

Luděk Skočovský
COMPUTER SCIENCE ENJOYER

Babce.

Luděk Skočovský
COMPUTER SCIENCE
ENJOYER

Kniha je určena specialistům netechnických oborů.

COMPUTER SCIENCE ENJOYER

© Luděk Skočovský

odborná oponentura

Jan Beran

Tomáš Hruška

čtenářská oponentura

Irena Velebilová

Karel Albrecht

Roman Švanda

redakce

Daniela Veškrnová

kresby

Vladimír Kokolia

Luděk Skočovský

obálka

Pavel Brabec s použitím kresby Vladimíra Kokolii

sazba

Vladimír Kokolia

Ivo Pecl

tisk

Arch – polygrafické práce, spol. s r.o.

vydal

Luděk Skočovský, Brno, 2006

náklad 250 ks na papíře, digitálně zdarma na www.skocovsky.cz/enjoyer, vydání první

zvláštní poděkování Daniele a grafickému studiu petit

Kniha obsahuje názvy programových produktů, firem apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

ISBN 80-902612-1-3

ÚVOD	ix
BASE OF	
I.1 výpočet, digitální údaj, von Neumann, CPU, paměť, periferie, disky	13
I.2 program, programovací jazyky, algoritmus, Turingův stroj, programovací techniky	27
I.3 operační systémy, servery a koncové stanice	37
I.4 uživatelé a jejich data, průzkum, adresáře a soubory	49
CONNECT TO	
II.1 propojení počítačů, multiplexing, vrstvení, technologie internet, DNS	63
II.2 klient a server	81
II.3 síťové aplikace: přenos dat, vzdálená práce, sdílení zdrojů, komunikace, publikování	91
ENJOY IT	
III.1 kód, text a obraz	125
III.2 desktop, okna, grafické aplikace, grafické formáty, zvukové formáty, video, kompozice	135
III.3 informační systémy a databáze, SQL, XML, datové sklady	167
III.4 bezpečnost, viry, červi, ochrana, šifrování	191
ZÁVĚR.	211
PŘÍLOHY	
A kódování: ASCII, UNICODE	215
B adresace v Internetu: URL, IP	221
C tagy HTML, značkovací jazyky	225
D editor vi	233
E klávesové zkratky	239
F SQL: přehled dotazů	253
G standardy a typy licencí v IT	259
LITERATURA.	263
REJSTŘÍK	267



V polovině 60. let 20. století především firma IBM vyráběla výpočetní stroje střediskového typu (sálové počítače, mainframe), jejichž cena se pohybovala řádově v milionech dolarů. Firma nebo organizace, která nákup výpočetní techniky tohoto typu provedla, musela pro správný chod počítače připravit klimatizovanou místnost, sál výpočetní techniky. Sál byl určen výhradně pro výpočetní techniku, měl v průměru okolo 100 m², speciální podlahu, protipožární vstupní dveře, bezpečnostní jištění atd. V okolí sálu bylo však nutno ještě vybudovat řadu místností a kanceláří pro obsluhující personál, který zahrnoval nejméně 10 osob. Provoz takového výpočetního střediska přišel tedy opět na pěknou sumu ročně. Životnost stroje přitom nebyla nijak závratná, obvykle se počítala na několik let, po jejichž uplynutí bylo nutno počítač vyměnit za jiný, většinou modernější. Střediskový počítač v té době byl přitom schopen provádět evidenci mzdových, inventárních nebo klientských agend, obvykle ve velmi primitivní podobě. Vedení organizace nemělo přístup k výsledkům prostřednictvím obrazovky s terminálem, požadavky se děrovaly na štítky nebo pásky a výsledky byly vytištěny na papír. Když vedení specifikovalo svůj požadavek, pak pokud šlo vše dobře a nebylo nutno problém nově programovat, výpočetní středisko dodalo výsledek na papíře nejdříve do druhého dne. Důvodem nebyla lenost personálu výpočetního střediska nebo neprůhlednost jeho práce, ale fakt, že to počítač rychleji vypočítat nedokázal. I přes tuto nemotornost ovládání a obrovské investice organizace počítače kupovaly a používaly.

V polovině 70. let nastal rozmach prodeje minipočítačů (minicomputer, zejména řada PDP firmy DEC nebo computery firmy Hewlett Packard). Jednalo se o pokrok. Výpočetní výkon byl sice nižší než výkon v té době i současně stále vyráběných střediskových počítačů, ale pro řadu aplikací postačoval. Např. se dobře hodil pro obsluhu pokladního systému v obchodě. Systém zvládl několik přípojných míst, tj. mohlo s ním komunikovat několik lidí současně prostřednictvím obrazovky a klávesnice (přibližně 4 i více připojení). Počítač nepotřeboval klimatizovanou místnost a vešel se do prostoru o velikosti několika ledniček. Stál pouze několik set tisíc dolarů.

V polovině 80. let se situace opakovala ve vlně osobních počítačů (personal computer, PC, původní koncepce od firmy IBM). Postupem času každý najednou PC potřeboval. Proč? Na to dokázal logicky a věcně odpovědět jen málokdo. Argument pracovního využití vedoucí k efektivitě působil směšně ve světle potřebné finanční investice (cena PC tehdy byla několik tisíc dolarů) a především času, který kupující ztratí, když se PC učí ovládat. Na rozdíl od sálových počítačů a minipočítačů si totiž uživatel musel s výpočetní jednotkou poradit sám nejenom jako uživatel, ale i jako správce. Lidé po půlroce od nákupu přiznávali, že tento krok byl zcela neefektivní, vůbec jim neušetřil čas a navíc je okradl o mnohonásobně víc času, než kdyby potřebné úkony prováděné na počítači dělali ručně. Současně ale s pokrčením ramen dodávali, že teď, ať už je výsledek jakýkoliv, se bez počítače již neobejdou.

Od té doby vždy po několika letech koupi opakují, aby získali v principu totéž PC s vyšším výkonem a kapacitou.

Zaujatost lidské populace pro výpočetní techniku v celém průběhu jejího vývoje nelze zdůvodňovat jejím současným využíváním v digitálním publikování, elektronické poště, fotografii, filmu, atd., tedy všestranností digitální informace jako média. Tento užitek (snad ekonomický, snad ekologický) lidé vědomě nesledovali, a to ani v nejodvážnější literatuře science fiction. I dnes jsme rozpačití, máme-li hovořit o dalším vývoji a smyslu.

Zcela jistě je ve fenoménu výpočetní techniky zaklet jiný význam, než který lze vysvětlit racionální úvahou. Je to paradoxní, protože počítače jsou matematické stroje, jsou vyrobeny na principech výsledků chladné úvahy techniků a přírodovědců a s uměním, filozofií nebo dokonce s iracionálními myšlenkovými pochody jako je víra, city, láska nemají ve svých principech nic společného. Tak dramatický zájem o ně, a to právě v netechnických a nevědeckých oborech lidské činnosti však ukazuje, že společného něco mají, něco, co je iracionální a podstatné. Co to je?

Programátoři a specialisté na výpočetní techniku vůbec jsou dnes lidé povětšinou jednoduší. Ti nejlepší jsou absolventi univerzit se zaměřením na přírodní vědy a techniku. Základy humanitní a filozofické přezírají jako nepotřebné, přestože je v rámci svého vzdělání absolvovali. Jejich život se omezuje na kódování programů, čtení monitorů s návody o nových způsobech kódování a zpracování dat. Ti nejkulturnější z nich navštíví jednou týdně biograf (pokud si film neprohlédnou v pirátské kopii z Internetu), kdy tématem filmu je opět ať už příjemná nebo znepokojivá budoucnost jejich oboru. Čtení beletrie, návštěva divadla či dokonce účast na výtvarné výstavě nebo přednášce na jiné téma než jsou počítače je pro ně činnost zcela absurdní a mnohdy směšná. Přesto dnes pod tlakem zájmu zbytku společnosti podléhají vědomí vlastní výjimečnosti a společenské nadřazenosti. Náhle se začínají, a to zcela běžně již i v masmédiích, vyjadřovat nejenom k politickým a společenským tématům a problémům, ale ke vši hrůze i k tématům ryze filozofickým, uměleckým a humanitním. Vždy však vycházejí ze svých pozic programátorů (a čtenářů science fiction) a mentorují okolní svět argumenty, na které např. vzdělaný filozof stěží dokáže kvalifikovaně odpovědět. Kvalifikovaný filozof totiž nezná principy informačních technologií a stydí se k nim vyjadřovat podobně neomaleně, jako to dělá programátor k jeho oboru. Zvláště když většinou potají zápasí s klávesnicí a myší svého domácího PC a systém oken (který je produktem právě programátorů) ho drze zrazuje v okamžiku, kdy jej nejvíce potřebuje a ponechává ho nechápajícího v jiném systému myšlení.

Znalost principů výpočetní techniky, digitalizace a informačních technologií vůbec je v současném stadiu vývoje lidské společnosti nezbytná pro každého jedince, který se podílí na rozhodování. Specialisté v různých oborech mají dnes k dispozici obecně zaměřené informační technologie, které mohou pro své potřeby použít a získat tak v oboru novou kvalitu. K takovému uzpůsobení je ale nutné principům a možnostem informačních technologií porozumět na

takové úrovni, aby následná komunikace s programátorem měla charakter spolupráce bez zbytečného přeceňování nebo podceňování.

Následující text je proto určen především specialistům netechnických oborů. Přestože připomíná sbírku esejí, je napsán přece jen formou technického odkazového textu (viz marginálie, rejstřík a přílohy). Tato techničnost k němu patří a začte-li se do něj čtenář, pak začíná pomalu rozumět tomuto zvláštnímu světu technicky smýšlejících programátorů naší doby.

Zbavme se frustrace a strachu před technologiemi, které jen čekají na naplnění svým obsahem. Obsah totiž spočívá v jiných oborech lidské činnosti než jsou počítače samy. Snažme se je však pochopit, bez tohoto pochopení totiž vláda technokratů přivede zajímavé nástroje na slepou cestu např. pouhých požitků a strhne k ní i celou společnost.

I.1 – výpočet, digitální údaj, von Neumann, CPU, paměť, periférie, disky

Základy výpočetní techniky vytváří matematická teorie, která před několika desetiletími přerostla v samostatnou partii vědeckého světa nazývanou Computer Science – věda o počítačích.

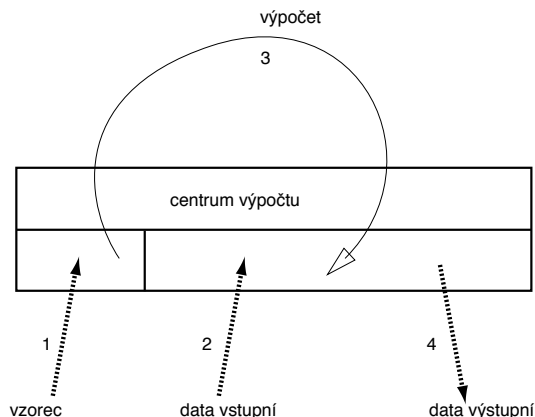
Computer Science

Jedná se o vědní obor, který pomocí matematického aparátu definuje a formalizuje kategorie, jako je výpočet, data atp. Výstupem Computer Science je digitalizace dat, programovací jazyky, operační systémy a počítačí stroje.

Jako každý vědecký obor, i Computer Science vznikla původně na základě požadavků empirie. Pro potřeby vědeckých výpočtů byl sestaven první matematický stroj, který požadované údaje akceptoval ve formě nul a jedniček (dvojková číselná soustava, digitální záznam), prováděl nad nimi požadované výpočty a výsledky zobrazoval opět jako nuly a jedničky. Co víc, samotný výpočet (tedy operace nad daty z nul a jedniček) byl říditelný, měnitelný a zaznamenaný opět ve formě nul a jedniček. Centrum výpočtu (dnes označované jako procesor) bylo řízeno výpočtem, tedy v čase postupně čtenými nulami a jedničkami, a na základě jejich kombinací měnilo nuly a jedničky dat na nové kombinace nul a jedniček. Po ukončení výpočtu tak byla vstupní data proměněna na data výstupní prostřednictvím požadované proměny dané obecným matematickým vzorcem. Člověk, který dokázal data z nul a jedniček číst, tak získal potřebný výsledek výpočtu. A co víc, získal prostředek opakované aplikace matematického vzorce nad různými daty. Musel ale:

- rozumět dvojkové soustavě,
- převádět do ní vstupní data,
- vyjádřit v ní výpočet,
- přečíst z ní výsledek.

Pomineme-li řadu dobově na technickém vývoji závislých nedostatků, můžeme si dnes popisovanou situaci ukázat např. ve formě následující kresby I-1.



Kresba I-1: Výpočet

Přitom výpočet na uvedené kresbě si musíme představit jako v čase probíhající proces. Nejprve je do centra výpočtu nahrán digitalizovaný vzorec, poté jsou nahrána digitalizovaná vstupní data, proběhne výpočet a nakonec jsou z centra výpočtu přehrána výstupní data, opět v digitální podobě. Čas je na kresbě zaznamenán očíslováním šipek (1, 2, 3, 4).

Tím, že celý proces probíhá ve formě digitálních údajů, se snažíme zajistit jednoznačnou opakovatelnost výpočtu, tj. že nuly a jedničky dat budou převedeny na jiné nuly a jedničky vždy stejným způsobem. Výstupní data budou tatáž, opakujeme-li výpočet nad týmiž vstupními daty. Takový proces obecného výpočtu lze totiž vzhledem k jeho digitální povaze matematicky formalizovat.

digitalizace

U *digitalizace* se jedná o formu abstrakce. Mám-li tři klacíky a přidám k nim další dva, dostanu pět klacíků, a to i v případě, že se tak stalo před rokem, měsícem nebo že se tak stane teprve příští rok. Mohu diskutovat o tom, že klacíky vypadají jinak: jednou jsou ulomeny ze smrkového dřeva, podruhé jsou z ovocného stromu, potřetí se nejedná vůbec o klacíky, ale je to pět kamíneků. Principem je ale záznam tří a dvou, jejichž sloučení ukáže výsledek pět. Co tímto způsobem vyjádříme, zda tři klacíky znamenají ženu a dva muže (nebo opačně) a pět klacíků základ rodiny, je otázkou interpretace. Budeme-li ale dále zjemňovat, můžeme různým počtem klacíků nebo kamíneků zaznamenávat postupně různé detaily muže či ženy a to nejenom fyzické, ale i psychické, mentální, morální. Je pak opět otázkou aplikace operací nad takovými záznamy, co budeme odvozovat z různých kombinovaných počtů klacíků (sčítáním, odčítáním, porovnáváním...).

Od dob vzniku tohoto principu průběhu digitálního výpočtu vlastně již jen docházelo k hledání různých forem záznamů reálného světa do kombinací nul a jedniček tak, aby záznamy bylo možné zpracovat výpočtem. V době vzniku prvních výpočetních strojů se používaly např. děrné pásky, kde díra v materiálu označovala jedničku. Později se přešlo na magnetický záznam impulsů, kde impuls byl jednička. Dnes je jednička záznamem v materiálu o velikosti molekul. Postupné zmenšování záznamové jednotky je dokáže zhuštít na stále menší plochu a umožňuje snižovat čas potřebný k jejich čtení. Ale ne již lidskými smysly. Přestože toto neustálé hledání (a nalézání) menšího prostoru na záznam jedniček a nul má v principu opět vědecký charakter, velmi podstatná je při tomto procesu technika. Ta se totiž snaží o co nejvyšší dostupnost nových typů záznamových technologií co největšímu možnému počtu lidí.

Computer Science je proto snad nejvíce ze všech vědních oborů propojena s technickým vývojem a reálným životem vůbec. Je na technologiích dokonce principiálně závislá. Její technické potřeby jsou ale také určující pro využívání v technologiích. Počítací stroje musí být programovány na základě principů této vědy.

Digitální údaj je vyjádření určité skutečnosti v číselném tvaru dvojkové soustavy. Znamená to, že pomocí čísel popíšeme např. sklenici s vodou, sochu, pohyb ruky, oblohu nebo politickou stranu. Jak jsme se již zmínili, popis přitom bude zanedbávat některé pro účel digitalizace nepodstatné části skutečnosti. Např. k digitalizaci pohybu ruky potřebujeme definici samotné ruky pouze v hrubých obrysech. Digitální záznam ruky nás bude zajímat teprve tehdy, zaměříme-li svou pozornost např. na kůži ruky nebo na její stárání. Digitálním záznamem vlastně vytváříme určitý model skutečnosti. Tento model může sloužit k různým cílům. Např. k pouhé prezentaci, kdy digitalizovanou sochu vystavujeme na Internetu. Může se jednat o řízení výrobního procesu, kdy digitální údaje provozu výroby z čidel zpracujeme a na základě výsledků pak zpětně ovlivňujeme modelovanou skutečnost změnou vstupních parametrů technologické linky. Nebo nad digitalizovanou skutečností provádíme simulace, např. digitalizovaný pohyb ruky končí v nádobě s horkou vodou.

digitální údaj

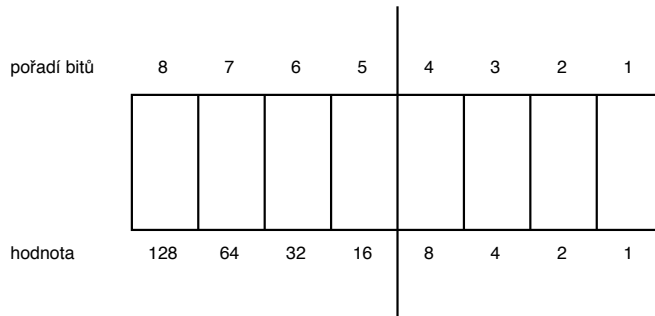
Základní digitální jednotkou je bit (binary digit, dvojková číslice). Označujeme jej zkráceně jako b. Bit může nabývat dvou hodnot, které nejjednodušeji vyjádříme ve formě protikladů. Dobro, zlo. Pravda, lež. Ano, ne. Nahore, dole. 1, 0. Protiklady ve zde uvedeném pořadí postupně ztrácejí citové zabarvení. U posledního již opravdu nelze říct, zda je 1 lepší než 0. To je podstatný prvek hodnoty jednoho bitu.

bit

Jak pomocí hodnot několika bitů vyjadřujeme skutečnost? Bity shromažďujeme do skupin o stejném počtu. 8 bitů tvoří bajt (byte). Označujeme jej zkráceně jako B. Osm nul a jedniček postupně za sebou vytváří hodnotu jednoho bajtu. Takto vznikají smluvené kombinace bitů na osmi po sobě jdoucích místech. Digitální údaj tak dokážeme produkovat nebo naopak číst po větších celcích. Tyto celky můžeme nazývat znaky, znaky mohou např. také být písmena abecedy.

bajt

Pořadí bitů v bajtu chápeme tak, jak je ukázáno na kresbě I-2.



Kresba I-2: Bity v bajtu.

8 bitů číslujeme zprava doleva, říkáme první bit, druhý bit..., osmý bit. Část bajtu od 1 do 4 nazýváme dolní polovina, 5 – 8 horní polovina. Podle umístění jedniček na takto smluvených pozicích dokážeme vyjadřovat hodnotu bajtu nikoliv pouze v nulách a jedničkách, ale i jako číslo v desítkové soustavě, tedy jak jsme v běžném vyjadřování zvyklí. Neříkáme totiž, že máme 1 a 1 klacík, říkáme, že máme 2 klacíky. Ohodnocení každé pozice bajtu v případě, že obsahuje údaj 1, je uvedeno na kresbě v dolní řadě čísel. 1. má hodnotu 1, 4. bit má hodnotu 8, 8. bit má hodnotu 128. V případě jedniček na více místech v bajtu hodnoty sčítáme, např.

0000 0001 je jednička,
 0000 0010 je dvojka,
 0000 0011 je trojka,
 0000 1111 je 15,
 0001 0000 je 16, že ano,
 1000 0001 je 129,
 1111 1111 je 255
 0000 0000 je nula.

Bajt tedy dokáže vyjádřit číselné hodnoty v rozmezí od 0 do 255. Nic víc. Každá hodnota bajtu nebo přesněji kombinace osmi nul a jedniček v pořadí za sebou dokáže udržet digitální informaci v tomto rozsahu. Každá hodnota vyjadřuje určitý znak. Je dohodnuto, že např. písmeno malé *a* má hodnotu 97 (0110 0001), velké *A* má 65 (0100 0001) a např. znak dvojtečka : má přiřazenou hodnotu 58 (0011 1010). Kódování je jednoznačné. Nedokážete vyjádřit hodnotu 58 jinak než jak je uvedeno v závorce.

ASCII Uvedená dohoda používání daných kombinací bitů v bajtu s odpovídajícím významem vyjádření různých znaků lidského světa byla obsažena v klíčovém

dokumentu amerického standardu s označením ASCII (American Standard Code for Information Interchange), viz [ASCII], a je platná dodnes. ASCII přitom nevyužívá všechny kombinace pro znaky, které jsou pro člověka čitelné. Ty čitelné soustřeďuje do dolní části tabulky kódování. Definuje hodnoty v rozsahu 0–127 (tedy bez využití pozice 8. bitu). Následující tabulka uvádí část ASCII, ve které jsou kódovány znaky lidskému oku běžně srozumitelné. Jedná se o písmena latinské abecedy, číslice a zvláštní znaky jako je čárka, dvojtečka, označení dolaru, křížek nebo procento. Ostatní kombinace i v této části mají význam speciálních značek a zde jsou uvedeny jen některé z nich (např. BEL, BS, LF). Úplná dolní část tabulky ASCII s popisem významu všech kombinací je uvedena v Příloze A.

0000 0000	0	NULL	0010 1011	43	+	0101 0110	86	V
0000 0001	1		0010 1100	44	,	0101 0111	87	W
0000 0010	2		0010 1101	45	-	0101 1000	88	X
0000 0011	3		0010 1110	46	.	0101 1001	89	Y
0000 0100	4		0010 1111	47	/	0101 1010	90	Z
0000 0101	5		0011 0000	48	0	0101 1011	91	[
0000 0110	6		0011 0001	49	1	0101 1100	92	\
0000 0111	7	BEL	0011 0010	50	2	0101 1101	93]
0000 1000	8	BS	0011 0011	51	3	0101 1110	94	^
0000 1001	9		0011 0100	52	4	0101 1111	95	`
0000 1010	10	LF	0011 0101	53	5	0110 0000	96	`
0000 1011	11		0011 0110	54	6	0110 0001	97	a
0000 1100	12		0011 0111	55	7	0110 0010	98	b
0000 1101	13	CR	0011 1000	56	8	0110 0011	99	c
0000 1110	14		0011 1001	57	9	0110 0100	100	d
0000 1111	15		0011 1010	58	:	0110 0101	101	e
0001 0000	16		0011 1011	59	;	0110 0110	102	f
0001 0001	17		0011 1100	60	<	0110 0111	103	g
0001 0010	18		0011 1101	61	=	0110 1000	104	h
0001 0011	19		0011 1110	62	>	0110 1001	105	i
0001 0100	20		0011 1111	63	?	0110 1010	106	j
0001 0101	21		0100 0000	64	@	0110 1011	107	k
0001 0110	22		0100 0001	65	A	0110 1100	108	l
0001 0111	23		0100 0010	66	B	0110 1101	109	m
0001 1000	24		0100 0011	67	C	0110 1110	110	n
0001 1001	25		0100 0100	68	D	0110 1111	111	o
0001 1010	26		0100 0101	69	E	0111 0000	112	p
0001 1011	27		0100 0110	70	F	0111 0001	113	q
0001 1100	28		0100 0111	71	G	0111 0010	114	r
0001 1101	29		0100 1000	72	H	0111 0011	115	s
0001 1110	30		0100 1001	73	I	0111 0100	116	t
0001 1111	31		0100 1010	74	J	0111 0101	117	u
0010 0000	32	SP	0100 1011	75	K	0111 0110	118	v
0010 0001	33	!	0100 1100	76	L	0111 0111	119	w
0010 0010	34	"	0100 1101	77	M	0111 1000	120	x
0010 0011	35	#	0100 1110	78	N	0111 1001	121	y
0010 0100	36	\$	0100 1111	79	O	0111 1010	122	z
0010 0101	37	%	0101 0000	80	P	0111 1011	123	{
0010 0110	38	&	0101 0001	81	Q	0111 1100	124	
0010 0111	39	'	0101 0010	82	R	0111 1101	125	}
0010 1000	40	(0101 0011	83	S	0111 1110	126	~
0010 1001	41)	0101 0100	84	T	0111 1111	127	DEL
0010 1010	42	*	0101 0101	85	U	1000 0000	128	

Je to, co bylo uvedeno, základem pro psaní textu např. dopisu elektronické pošty, článku, povídky? Přesně tak. Začnu-li psát do poštovního programu nového elektronického dopisu např. text

```
Mary had a little lamb;
its fleece was white as snow
```

počítač akceptuje úderky na klávesnici a znaky postupně registruje ve formě nul a jedniček podle uvedené tabulky. Vnitřní interpretace textu je pak bit po bitu (bajt po bajtu, znak po znaku) následující:

M	a	r	y		h	a	d
01001101	01100001	01110010	01111001	01100001	01101000	00100000	01100100
	a		l	i	t	t	l
01100001	01100001	01100001	01101100	01101001	01110100	01110100	01101100
e		l	a	m	b	;	
01100101	01100001	01101100	01100001	01101101	01100010	00111011	00001010
i	t	s		f	l	e	e
01101001	01110100	01110011	01100001	01100110	01101100	01100101	01100101
c	e		w	a	s		w
01100011	01100101	01100001	01110111	01100001	01110011	01100001	01110111
h	i	t	e		a	s	
01101000	01101001	01110100	01100101	01100001	01100001	01110011	01100001
s	n	o	w				
01110011	01101110	01101111	01110111				

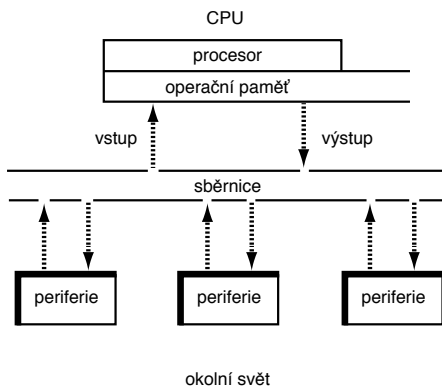
Vzhledem k tomu, že slova ve větě textu jsou oddělena mezerami, jsou i ty součástí záznamu (mezera, SP má binární kód 01100001). Všimněme si také znaku mezi středníkem a slůvkem *its*, text je přerušen přechodem na nový řádek. Tak jej vnímá čtenář, ale počítač tento efekt registruje použitím znaku 00001010, který má v tabulce ASCII označení LF (line feed). Pokud pak text počítač zobrazuje, uvedený znak interpretuje jako pokračování v textu na novém řádku. Registrovaných znaků v tabulce ASCII podobného významu je přitom více. Např. kód 00000111 (BEL) je znak, který počítač interpretuje jako zvukový signál (pípnutí). Uprostřed textu tak může být uveden znak se zvukovou signalizací (je ovšem interpretován v okamžiku kdy jej počítač zobrazuje a nikoliv kdy jej člověk čte).

Takto speciální význam mají i další znaky, jejichž zkratky jsme v ASCII ani neuvadli. Přestože jejich význam není nijak magický, jejich vysvětlení předpokládá např. znalosti programování nebo organizace dat v pamětech při jejich přesunu z místa na místo a proto se jimi zde nebudeme zabývat (viz Příloha A).

**Von Neumannovo
schéma**

Přestože základy počítačích strojů dnešního typu mají původ historicky mnohem starší (např. Keplerův model, Pascalův model, Leibnitzův model), za princip je dnes považováno a vždy citováno Von Neumannovo schéma. Jedná se o konstrukci, kterou v roce 1952 navrhl i realizoval matematik

John von Neumann. Jeho koncepci počítačícího stroje na úrovni binárního zpracování dat jsme v principu již uvedli na kresbě I-1. Základ Von Neumannova schématu ukazuje následující kresba I-3.



Kresba I-3: Von Neumannovo schéma.

Podstatná je jednotka provádějící výpočet, označovaná jako CPU (Central Processor Unit, dříve na kresbě I-1 jako centrum výpočtu), které se v praxi neřekne jinak než procesor. S procesorem úzce souvisí operační paměť. Procesor je zdrojem výpočtu, procesor mění data v této paměti, operuje, provádí operace v paměti, mění v ní nuly na jedničky a naopak, proto je tato paměť operační. Operační paměť je ale omezená svou velikostí vzhledem k dosahu procesoru, jeho adresaci. CPU tak potřebuje ke své činnosti dostávat do operační paměti postupně (různá) data a po jejich zpracování je z operační paměti dostávat opět ven. K tomu slouží kanály dat, kterým zkráceně říkáme vstup a výstup (input, output), rozhraní pro tento přenos dat je často zkráceně označováno I/O. Vzhledem k tomu, že data mohou být čerpána nebo naopak výsledky zaznamenávány na různých typech zařízení, kanály I/O spojuje CPU s okolním světem sběrnice (bus). Jedná se o univerzální propojení, kterým binární data procházejí z okolního světa do CPU a zpět. Okolní svět je na kresbě reprezentován periferiemi. Mohou být různých typů. V době von Neumanna se jednalo o děrné pásy. Dnes se jedná o diskety, CD, pevné disky různých typů, paměti typu flash, tiskárny, obrazovky, klávesnice, myši, tablety atd.

Základní princip je tedy v možnosti uchovávat digitální údaje na zařízení, kterému říkáme počítačová periferie (device), možnosti výběru z většího množství dat na této periférii, přenosu vybraných dat do CPU, zpracování těchto dat, tj. jejich přepočtu na data výsledná a konečně jejich přenosu zpět na periférii a jejich záznamu pro další uchování. Data tak procházejí výpočtem.

CPU - procesor

operační paměť

sběrnice

periferie

**vstupní
a výstupní data**

Vznikají data nová a jsou ukládána vedle dat původních. Získáváme tak termín pro označování dat jako data vstupní (input data) a data výstupní (output data).

CPU má omezenou velikost operační paměti a její obsah po ukončení výpočtu a vypnutí počítačového stroje zaniká. Pokud tato data před vypnutím stroje např. vyděrujeme na děrnou pásku nebo vypálíme laserem na CD, uchováme digitální kopii těchto dat pro jejich další použití. Toto je jeden z významů periférií a jejich připojování k CPU.

program CPU přitom musí mít zadán způsob, jak data v operační paměti měnit z jejich vstupní podoby na výstupní. Vykonání takového procesu je opět zaznamenáno ve formě nul a jedniček, ve formě sekvence bajtů. Takovému zápisu nul a jedniček, který řídí výpočet, říkáme program. Vzhledem k tomu, že se jedná o tentýž princip kódování jako u dat, je program rovněž přenášěn z periférií do CPU, kde je uložen a postupně interpretován procesorem. Po ukončení výpočtu jsou výstupní data přenesena na periférii, kde jsou zaznamenána a program může být v operační paměti zapomenut. Pro další zpracování dat pak načítáme do CPU jiný program, jiná data a následně probíhá další výpočet.

princip paměti Obecně je paměť schopnost udržet informaci v čase. Jak u informace, tak i času záleží na kapacitě. Je důležité, jak dlouho si podržíme vzpomínku na svatbu své dcery a do jakých podrobností. Protože je lidský mozek paměť poměrně špatnou (v čase informace zapomíná, neobjektivně si pamatuje pouze části, nové informace vytěsňují staré), pokouší se lidstvo od tohoto zjištění informace uchovávat na jiných záznamových médiích (hliněné tabulky, papír, malířské plátno, sochařský kámen atp.). U každého záznamu mimo lidský mozek se kvalita záznamu odvozuje od kapacity informací, kterou je médium schopno podržet, a od časového intervalu, po který je to možné. Časový interval uchování informace lidé prodlužují kopírováním, pokud je tato kopie vůbec možná (socha, malba).

Digitální informace potřebuje ke svému uchování médium, které je schopno jednoznačně rozlišovat dvě hodnoty, 0 a 1. Jako takovou ji tedy lze snadno kopírovat, ale ke svému záznamu v praxi vyžaduje poměrně rozsáhlé kapacity. Dnes prožíváme dobu, ve které je snahou digitalizovat (tedy kódovat do nul a jedniček) plošně bez abstrakce (jako je digitální fotografie, film, robotický prostor) a teprve následně digitalizovanou skutečnost analyzovat a vyhodnocovat. Jednak je to technicky možné a jednak to odpovídá snaze zaznamenat pokud možno vše tak, aby bylo možné zpětně vyhledávat (a později abstrahovat) informace, které jsme v okamžiku záznamu nepovažovali za podstatné.

Rozdíl v technických možnostech pamětí von Neumannovy doby a dneška se dá lehce vypočítat. Děrná páska z dnešního pohledu mohla mít šířku 8 děr, tedy bajt (ale i tak neměla!). Podle kvality pásky a způsobu, jakým byly díry raženy, bajt potřeboval i s odstupem k dalšímu bajtu nejméně 2,5 mm. Metr děrné pásky tak dokázal zaznamenat průměrně 400 bajtů. Pokud páska na kotouči byla dlouhá 20 m (ale zdaleka nebyla!), vychází nám 8000 bajtů,

tedy asi 8000 znaků záznamu informací. V dnešní řeči znalců by se jednalo přibližně o „osmi kilový“ (~8kB) soubor. Dnešní stále ještě používaná disketa (3,5“) má kapacitu 1,4MB, tedy 1400kB, tj. přibližně jeden a půl miliónu bajtů, znaků. Dohromady se na ni mnoho nevejde. CD už je lepší, jeho kapacita je 700MB, výpočet s představou, co obnáší 80 minut digitálního kódování zvuků na CD, ponecháváme na čtenáři. Také je možné si vypočítat, kolik vteřin záznamu digitální hudby dokážeme uložit na disketu a kolik by se jí vešlo na kotouč děrné pásky. Konečně DVD má dnes kapacitu 11GB (a více), tedy více než 11000MB trvalého záznamu.

Matematický stroj používá pro výpočet operační paměť. Bývá označována také jako vnitřní paměť nebo paměť RAM (Random Access Memory), což překládáme jako paměť s přímým přístupem, protože angl. random je zde použito ve smyslu stejnou rychlostí kamkoliv je požadováno (náhodně vybráno). Operační paměť má omezenou velikost. Omezení je dáno technickým vývojem. Operační paměť stroje je totiž elektricky udržována v takové organizaci, aby kterýkoliv bit byl procesorem dostupný stejnou rychlostí. To je náročné elektricky zajistit. Technická omezení velikosti operační paměti jsou také v délce slova, se kterým dokáže procesor pracovat jako s elementární jednotkou. Délka slova je počet bitů. Čtenář jistě zná termíny 8bitový procesor, 16bitový procesor, 64bitový procesor. Podle této délky slova je pak možné obsáhnout vypočitatelnou maximální velikost operační paměti. Také proto se výzkum ve vývoji hardware poměřuje schopnostmi vyvinout procesor na vyšších délkách slova. Znamená to získat větší operační paměť, možnost adresovat větší prostor. Současně to znamená změnit architekturu procesoru, tj. změnit (ale nejlépe pouze rozšířit) jeho způsob operování s bity v paměti.

**RAM – paměť
s přímým
přístupem**

CPU je jednotka centrálního výpočtu. Digitální data jsou jí předávána a opět z ní odebrána jako výsledky. K uchovávání dat i po vypnutí počítače používáme externí paměti. V principu jsou to periferie (viz kresba I-3). Kapacita a časový interval, po který tyto periferie informace uchovají, závisí technicky na záznamovém způsobu. Disketa nebo magnetická páska má schopnost udržet informace po dobu přibližně 10 let, záznam na CD se dnes odhaduje na více než 50 let. Počítací stroj pro průběh výpočtu je dnes technicky závislý na trvalém přísunu elektrické energie. U konstrukce externích pamětí vyhledáváme a používáme způsob uchovávání informací tak, aby tato závislost nebyla podmínkou jejich existence. Čas, po který je informace uchována, obvykle závisí na fyzické existenci hmoty v určité konzistentní podobě. Externí paměti také z tohoto důvodu někdy obecně nazýváme paměťmi trvalými.

externí paměť

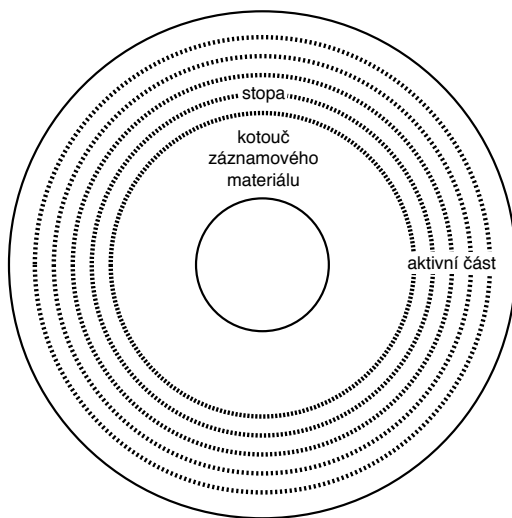
Externí paměti mají dnes pro práci s daty klíčový význam. Na těchto perifériích totiž udržujeme data, z nichž vybíráme ta, která potřebujeme zpracovat v CPU nebo která potřebujeme z archivních důvodů. V externích pamětech udržujeme také programy, které do operační paměti zavádíme pro realizaci samotného výpočtu.

Např. chci-li pokračovat v psaní textu na osobním počítači, vyberu si program jako je např. MS Word. Ten je z externí paměti přesunut do operační. Pomocí něj vyhledám v externí paměti data svého textu (ve kterém chci pokračovat) a přesunu je rovněž do operační paměti. Psaní textu na klávesnici, jeho registrování programem MS Word v CPU a zobrazování na obrazovce je průběh výpočtu. Ukončím-li psaní, musím pak ještě data textu z operační paměti přesunout zpět do paměti externí. Mohu přitom zachovat předchozí verzi textu a aktuální text uložit s jiným označením.

externí disková paměť

Při vývoji technických možností počítačích strojů od jejich vzniku až po dnešek bylo mimo jiné podstatné úsilí dosáhnout co největších kapacit externích pamětí tak, aby kterákoliv data v nich uložená byla snadno a rychle dostupná. Používání pásek se ukázalo při větším objemu dat jako velmi zpomalující. Přestože děrné pásy rychle vystřídaly magnetické o stále větších kapacitách, vždy se musela páska přetáčet na určitou pozici, kde byla uložena data nebo program. Jakkoliv se rychlost přetáčení zrychlovala, CPU i člověk čekali na data neúměrně dlouhou dobu. Lidé proto hledali a rychle našli technologii, která je označována jako *diskový princip*.

Disk je mechanismus, který eviduje data na kotouči v soustředných kruzích. Každému kruhu říkáme *stopa* (track), viz kresba I-4.

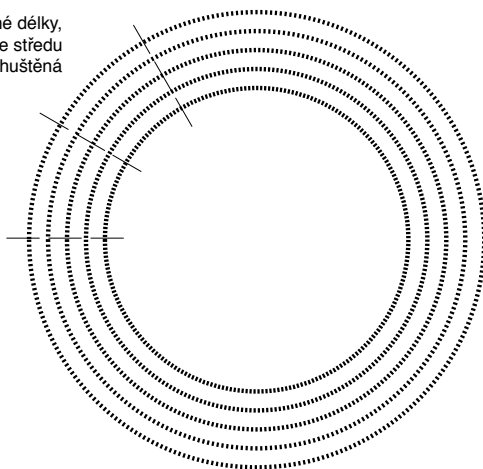


Kresba I-4: Stopy disku.

Stopa je část záznamového materiálu. Tento záznam vytváří a zpětně čte *hlava* (head), která je *raménkem* vysouvána nad určitou stopu. Materiál se záznamem se otáčí a hlava tak čte nebo zapisuje data ve vybrané stopě. *Povrch* disku (surface), tj. jeho aktivní plocha, je rozdělen na stopy, ve kterých jsou data uložena. Vysunutí raménka nad určitou stopu umožní rychlý přesun od dat na začátku a konci disku. Diskový princip tedy záznamový materiál

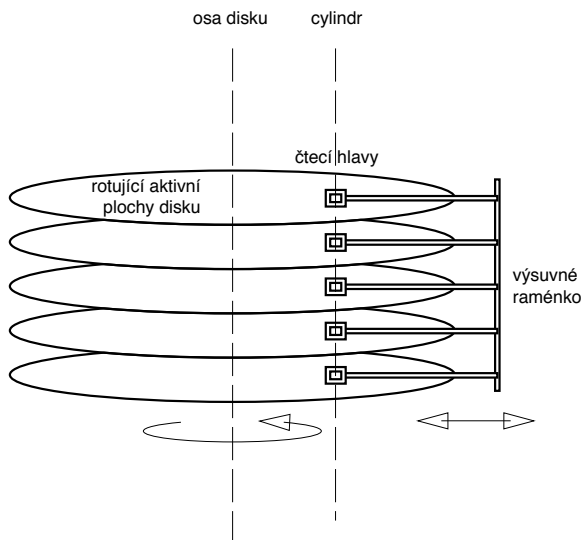
využívá po logicky oddělených částech, stopách. Jenomže stopy mají různou délku, směrem ke středu povrchu se jejich délky zkracují. Každá stopa je proto ještě dále rozdělena na sektory (sector). Sektor je na celém disku vždy stejné délky a můžeme si jej představit jako část povrchu, kam je možno zaznamenat např. 128 bajtů, viz kresba I-5.

sektory datově stejné délky,
fyzicky jsou směrem ke středu
data více zhuštěná



Kresba I-5: Sektory od sebe vzdálených stop disku.

Výkonné disky jsou konstruovány tak, že mají více aktivních ploch. Ty se při provozu otáčejí kolem společné osy. Každá aktivní plocha je čtena vlastní hlavou. Raménko, které ovládá pohyb hlav nad povrchem, je tak osazeno tolika hlavami, kolik má disk aktivních ploch. Systém hlav tak čte data z několika sektorů různých stop současně, jedná se o záznamový cylindr, viz kresba I-6.



Kresba I-6: Disk.

Jak je patrné, principem disku je snaha získat oddělené záznamové části (segmenty), které lze rychle střídát při čtení nebo zápisu. Toto rozdělení je důležité, uvědomíme-li si, že často nepořizujeme pouze data nová, ale často data doplňujeme nebo měníme (např. při psaní textu). Komplikace totiž nastanou, vložíme-li data doprostřed již existujícího textu. Od daného místa bychom na páskovém principu museli data odsunout o potřebný kus, abychom získali potřebný prostor. U diskové paměti manipulujeme s paměťovými kousky, které jsou pro souvislý text navzájem zřetězeny pomocným mechanismem ukládání dat na disku. U diskové paměti při vsouvání nového textu data v daném místě sekvenční sektorů v cylindrech rozpojíme a vložíme mezi ně potřebný počet zatím nepoužitých sektorů s obsahem nového textu. Propojení těchto dat tak, aby byla souvislá od začátku do konce, dále registrujeme v jiné (režijní) části disku. Takový pomocný mechanismus má speciální vnitřní organizaci dat diskových pamětí a ta může být poměrně složitá. Uživatel se těmito podrobnostmi nezabývá, pro něj disková paměť nabízí úložiště dat tak, že jsou různá data dostupná v průměru stejnou rychlostí. Využívá výsledku, data na disku mění, ruší nebo nově organizuje jejich umístění.

Diskový princip zrušil sekvenční systém zápisu dat tak, jak jej používal princip páskový. Páskový záznam se přesto stále používá, ale především k archivním účelům. Jeho princip ukládání dat je totiž jistě lépe čitelný, a to např. i při ztrátě způsobu organizace, jakým byla data na pásku zaznamenána.

Samotný fyzikální princip záznamu (tj. otisk nul a jedniček do materiálu), na kterém je diskový princip použit, může být různý. Magnetický záznam používá např. disketa, médium má dvě aktivní plochy: horní a dolní. Laserem

data zapisujeme na CD nebo DVD, které má pouze jednu aktivní plochu. Původní technologie ukládání dat na CD se ale odlišuje od popsaného diskového principu. Data jsou totiž ukládána sekvenčně za sebou nikoliv ve vyznačených sektorech, ale souvisle tak, že vytvářejí spirálu. To bylo vhodné pro data, která mají neměnný charakter, např. při distribuci zvukového záznamu (analogii lze spatřovat v páskových perifériích). Při využívání CD pro uchovávání měnících se dat (přepisovatelná CD) o něco později tak docházelo k nepříjemným komplikacím.

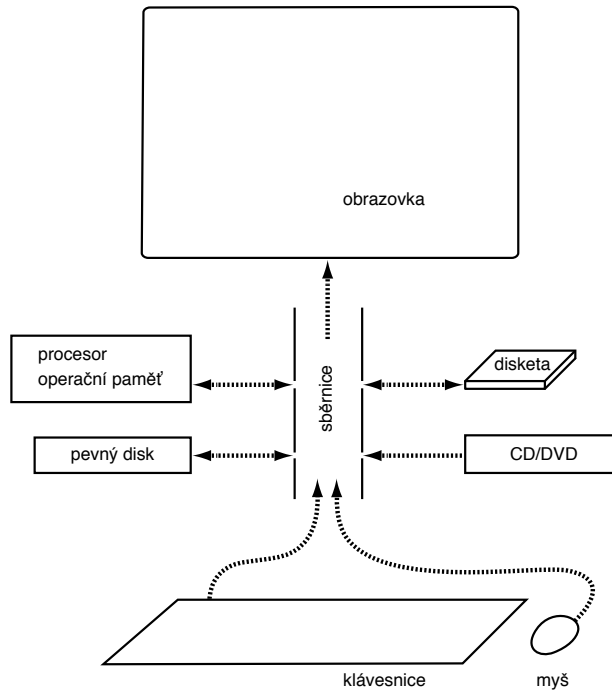
U CD a DVD rozlišujeme mechaniky pro jejich zápis a pro jejich čtení. Technologie totiž vznikla pro potřeby šíření většího objemu dat tak, aby jej nebylo možné změnit a dále vydávat za původní. Technologie také disponuje poměrně rychlým čtením již na médiu neměnných dat. Pro zápis je ale technologie vhodná méně a používá se proto především pro archivní a distribuční účely (digitální zvuk, obraz, film atd.). Protože jsou data na těchto typech médií již neměnná, označujeme tyto technologie jako paměti s trvalým zápisem. Formální je termín trvalá externí paměť, vlastně na diskovém principu. V protikladném významu je používán termín přepisovatelná externí paměť.

**trvalá
a přepisovatelná
externí paměť**

Princip diskové paměti byl ale využíván zejména při konstrukcích základních, tzv. systémových externích přepisovatelných pamětí. Má-li být počítač funkční a komfortní, jeho součástí musí být určitá externí přepisovatelná paměť. Po zapnutí stroje je totiž z ní do operační paměti okopírován (natažen, angl. load) základní program, pomocí něhož pak může člověk s počítačem komunikovat při zadávání výpočtů (takový program je principem tzv. operačního systému). Stroj by také měl být uzpůsoben k okamžitému použití, tj. i neodborník by měl mít možnost ihned využívat programy a svá data, aniž by musel zakládat média přenosných externích pamětí s potřebnými programy a další se svými daty. Každý počítač proto od počátku jejich širšího používání disponoval jedním nebo více diskovými přepisovatelnými pamětmi, které se později označovaly jako pevný disk (hard disk). Dnešní technologie vývoje fyzikálního principu záznamu diskových přepisovatelných pamětí se pohybují na úrovni různého využívání nejmenších známých a lidem dosažitelných elementů hmoty, na které dokážeme působit tak, aby plnily funkce proměnlivé digitální paměti. Důvodů je několik: zmenšování záznamového média, rychlost přístupu k uložené informaci, dosažení co nejvyšší kapacity a současně snadná přenositelnost média.

pevné disky

Základní způsob práce počítače ale zůstává od dob von Neumanna stejný. Ať už se jedná o počítač osobní nebo server sítě či superpočítač, vždy je stroj sestaven nejméně z CPU, operační paměti a externí diskové přepisovatelné paměti připojené k CPU sběrnici. Ke sběrnici jsou také připojeny všechny ostatní periferie, přenosné diskové paměti, páskové (vždy přenosné) paměti, obrazovky, klávesnice, tiskárny atd. Dobře je tento princip viditelný na konstrukci osobního počítače (personal computer, PC) na kresbě I-7.



Kresba I-7: Hardware PC.

I.2 – program, programovací jazyky, algoritmus, Turingův stroj, programovací techniky

Již na kresbě I-1 jsme uvedli princip výpočtu. Samotný výpočet je prováděn procesorem, procesor je řízen vzorcem výpočtu. Použití termínu vzorec je ale velmi zjednodušující. Jedná se totiž o definici práce procesoru, kterou obvykle zapisujeme daleko složitěji než jedním výrazem (vzorcem), proto používáme výraz program. Říkáme, že procesor provádí operace podle programu. Program je sekvence instrukcí, které procesor postupně provádí a mění tak bity operační paměti. Instrukce je pokyn pro procesor, aby změnil určitou část operační paměti. Běžné instrukce mají např. význam: sečti dva bajty, přesuň bajt na jiné místo, otestuj zda je na místě daného bitu nula nebo jednička a podle výsledku testu sečti nebo odečti nebo někam zapiš daný bajt, atp. Instrukce je kódována na několika bajtech a je vyjádřena jednoznačně definovaným systémem jedniček a nul. Jsou-li instrukce definovány na jednom bajtu, jedná se o 8bitový procesor. Má-li instrukce délku 8 bajtů, jedná se o 64bitový procesor. Délka instrukce přitom ale neznamená, že procesor využíváme na větším počtu instrukcí. Jedná se spíše o možnosti rozsáhlejšího popisu akce, kterou má instrukce vykonat. Instrukci také např. musíme sdělit, ve které části operační paměti mají být data zpracována. Umístění v operační paměti při delší instrukci tak může být vzdálenější. Znamená to, že instrukce má kromě svého pevně přiděleného kódu také variabilní část tzv. parametrů, pomocí nichž se mění konkrétní výsledek jejího vykonání. Seznam všech instrukcí vytváří instrukční repertoár procesoru. Instrukční repertoár procesoru vytváří programovací jazyk stroje nazývaný assembler. Assembler je považován za součást stroje, čili za součást hardware.

program

**assembler,
jazyk stroje**

Procesor může být konstruován na základě různých typů stavby assembleru. Dodnes není známa optimální varianta konstrukce obecného procesoru. Základní problém je jednoznačně v definici všech potřebných a současně dostačujících instrukcí včetně významu jejich parametrů. Další význam je v technologickém růstu podpory procesorů. Znamená to, že např. assembler 32bitových procesorů musí dokázat interpretovat i programy procesorů 16bitových, aby při výměně stroje za výkonnější nebylo nutné programy psát znova. Tomuto dodržování již jednou stanovených principů provádění instrukcí říkáme zpětná kompatibilita procesorů. V praxi to ale také znamená, že výrobci různých procesorů jsou vzájemně nekompatibilní (mají jiný assembler). Tato nekompatibilita ale není pouze dána obchodní konkurencí, ale odráží stav současné Computer Science: není znám obecně přijatelný a efektivní princip procesoru. Dobře vymyšlený a prakticky propracovaný návrh assembleru může navíc výrazně zvýšit výkon samotného stroje.

Program je sekvence instrukcí. K tomu, aby programátor nemusel programovat v jazyce stroje, assembleru, a zabývat se složitými přesuny dat na úrovni bitů a bajtů, vznikly v průběhu vývoje programovací jazyky vyšší úrovně. Jedná se o vyjádření programu textem, který je přístupnější myšlení

**programovací
jazyky vyšší
úrovně**

člověka. Programátor ve vyšším programovacím jazyce se tak věnuje více samotnému problému, který má vyjádřit ve formě programu.

Dnes nejrozšířenější programovací jazyky jsou např. FORTRAN (vznikl v r. 1957, viz [Vogel1976]), Pascal (vznikl v r. 1970, viz [Wirth1970]), C (vznikl v r. 1970, viz [Ritchie1970]) a Java (vznikl v r. 1991, viz [Flanagan1996]).

Text programu v jazyce C, který vypočítá součet dvou jedniček, je následující:

```
main()
{
    int a;
    a = 1 + 1;
}
```

Přestože jsme se právě prohršeli několika formálními nedostatky vůči standardu jazyka, každá implementace jazyka C by tento text programu akceptovala. I tak jsme museli ovšem uvést, že se jedná o samostatný program (slovem `main`). Levou a pravou kulatou závorkou jsme se vzdali parametrů programu. Levou složenou závorkou jsme zahájili sekvenci instrukcí, pravou složenou závorkou jsme sekvenci ukončili a uzavřeli tak zápis programu. Sekvence instrukcí obsahuje dvě instrukce, a to součet dvou jedniček, který je vložen do místa v operační paměti s naším pojmenováním písmenem `a`. Pojmenování jsme stanovili první instrukcí. Všechny instrukce v jazyce C jsou ukončeny středníkem.

Ekvivalentní kód v jazyce assembler pro procesor PC typu Intel v operačním systému Linux pak má podobu:

```
.file    "program1a1.c"
.text
.globl main
.type    main,@function
main:
    pushl    %ebp
    movl     %esp, %ebp
    subl     $8, %esp
    andl     $-16, %esp
    movl     $0, %eax
    subl     %eax, %esp
    movl     $2, -4(%ebp)
    leave
    ret
.Lfel:
    .size    main,.Lfel-main
    .ident   "GCC: (GNU) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)"
```

Uvedený text je dnes již stěží srozumitelný i erudovanému programátorovi. Přesto, text je rozdělen do několika sekcí tak, aby jej zaváděcí program do operační paměti rozpoznal a určil správně ke zpracování, tj. především odlišil samotný program od dat. Samotný kód práce programu začíná za řádkem s označením `main`. Použité instrukce jazyka assembler jsou zde `pushl`, `movl`, `subl`, `andl` atd., každá z nich je parametrizována pro práci s danou částí operační paměti.

Uvedený program je introvertní. To znamená, že jsme při jeho programování vypočítali součet 1 a 1, výsledek jsme ale vůbec nezobrazili a ani nepřenesli z operační paměti do externí. Náš program také neumožňuje zadávat jiné hodnoty k součtu než jsou dvě jedničky. Dopracování programu do podoby komunikace s externími periferiemi tak, aby interaktivně počítal variabilní zadávané hodnoty (součet dvou libovolných čísel, základ kalkulačky) by dále znamenalo použít v kódu instrukce pro čtení vstupních dat z klávesnice a výpis obsahu paměti s označením a na obrazovku. Program by nebyl složitý a bez komentáře si uveďme jeho podobu v jazyce C:

```
main()  
{  
    int a,b,c;  
    printf("zadej a: ");  
    scanf("%d",&a);  
    printf("zadej b: ");  
    scanf("%d",&b);  
    c = a + b;  
    printf("c je: %d\n",c);  
}
```

Význam jazyků vyšší úrovně je tedy zřejmý. Programovat v instrukcích assembleru je mnohem obtížnější, přestože z jazyka vyšší úrovně je nutné vždy program převádět na podobu, kterou procesor dokáže interpretovat. To programátor provádí pomocí speciálního programu, kterému říkáme *kompilátor* (compiler), překladač. Kompilátor je program, který provádí překlad vždy pro určitý typ procesoru. Pro stejné jazyky vyšší úrovně ale také vznikají různé kompilátory pro tentýž typ procesoru. Příkladem může být kompilátor jazyka C firmy Borland a firmy Microsoft pro procesory typu Intel v prostředí PC a MS Windows.

kompilátor

Převodem kódu programu do assembleru a konstrukcí (umělých) programovacích jazyků vůbec se zabývá *teorie programování*.

Program je popis algoritmu, který má počítač jako matematický stroj interpretovat a vytvářet tak varianty modelu skutečnosti. Model je abstrakce skutečného světa, který takto vzniká. Vstupem algoritmu jsou data, která jsou v průběhu zpracování strojem proměněna na data výstupní. Model v rámci tohoto procesu zpracování dat nabývá jedné z mnoha podob, která je dána právě použitými vstupními daty. Jedná se o instanci výpočtu, říkáme, že probíhá proces výpočtu nad daným modelem (programem daným algoritmem).

algoritmus

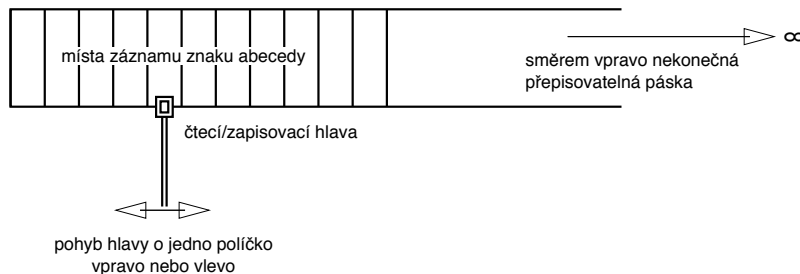
Model může vyjadřovat různé problémy: jedním je zápis textu, vstupem jsou pak data vkládaná člověkem z klávesnice a výstupem digitální záznam; dalším je simulace chemického procesu za vstupu různého poměru chemikálií, výstupem je grafické zobrazení chování procesu pozorované člověkem na obrazovce; třetím může být např. řízení technologického procesu, kdy teplota prostředí, rychlost pohybu válcovací stolice, konzistence zpracovávaného

materiálu jsou vstupní údaje, výstupem jsou řídicí impulsy motorům pro snížení nebo zvýšení snímané rychlosti, zvýšení teploty tavicí pece atp.

Pro obecné vyjádření modelu skutečnosti tak, aby model nabýval různých instancí na základě různých vstupních dat, používáme programovací jazyky. Jedna ze základních partií Computer Science je věnována právě teorii výpočtu. Je to matematické vyjádření teorie programovacích jazyků tak, aby programovacím jazykem bylo možné vyjádřit obecně člověkem vnímaný model skutečnosti. Jinými slovy, matematici se pokoušejí vyjádřit teorii programování tak, aby každý požadovaný model podléhal formálnímu vyjádření (napsanému programu), které dokazuje, že model jednoznačně nabývá určité instance nad jakýmkoliv vstupními daty. Ještě jednou a lidsky řečeno, programátor použije programovací jazyk a vznikne program, který pracuje bezchybně, ať jsou mu předložena jakákoliv vstupní data. Princip takové teorie není v umu programátora, ale v použitém programovacím jazyce, který zajistí, že v něm vyjádřený model je bezchybný.

Turingův stroj

Teorie výpočtu zajímala matematiky už dávno. Základní práce popisující obecného algoritmu vyjádřil již v r. 1936 matematik Alan Turing (Von Neumannovo schéma je z r. 1952!). Definoval *Turingův stroj* (Turing machine) jako obecný výpočetní prostředek, který je dodnes platný (viz např. [Manna1974], [Turing1936]). Turingův stroj se nejlépe vysvětluje podle kresby I-8 a následného popisu.



Kresba I-8: Turingův stroj.

Turingův stroj je algoritmus pohybující hlavou nad záznamovou přepisovatelnou páskou. Před startem stroje je na pásce zaznamenáno slovo používané abecedy (vstupní data). Výstupem je slovo téže abecedy, které vznikne přepisem vstupního slova tak, že algoritmus pohybuje hlavou vždy o jedno políčko vpravo nebo vlevo ze současné pozice s určitým znakem a po posunu na nové pozici přepíše obsah novým, algoritmem daným znakem. Algoritmus Turingova stroje je tedy vyjádřitelný sekvencí trojic ve tvaru (a , kam , b), tj. a je znak na pásce, kam určuje zda doprava nebo doleva a b je znak, kterým je políčko na pásce v nové pozici hlavy přepsáno. Podle vstupního slova tedy algoritmus pracuje a produkuje výstupní slovo. Říkáme, že v průběhu své práce se nachází postupně v různých stavech, které jsou dány jak vstupním slovem, tak daným algoritmem. Výsledkem práce

Turingova stroje je potvrzení nebo zamítnutí vstupního slova. To znamená, že pokud je sekvence definovaných trojic ukončena, slovo je potvrzeno, nastane-li ale situace, kdy se stroj ocitne ve stavu, odkud nemůže pokračovat, vstupní slovo je zamítnuto. To předpokládá, že algoritmus stroje má ještě označen výchozí stav a stavy ukončení. Pokud dosáhne jednoho ze stavů ukončení, slovo je potvrzeno, stroj se zastavil s potvrzením vstupního slova, pokud žádného z koncových stavů nedosáhne a přesto se zastaví (nemá k dispozici odpovídající sekvenci trojic pro pokračování), slovo je odmítnuto, stroj se zastavil s odmítnutím vstupního slova.

Turingovým strojem lze popsat jakýkoliv algoritmus reálného světa.

Obecně však nelze matematicky dokázat, že se Turingův stroj vždy zastaví nad obecně danou množinou slov používané abecedy, a to ať již ve stavu potvrzení nebo zamítnutí vstupního slova. Taková situace znamená, že stroj stále pracuje, přechází do dalších stavů a nelze říct, zda bylo zatím stavů pouze příliš málo a nebo zda je práce stroje nekonečná. Říkáme, že problém, zda se Turingův stroj zastaví po konečném počtu stavů nebo ne, je matematicky nerozhodnutelný.

Programovací jazyk je umělá řeč, kterou vyjadřujeme potřebný algoritmus pro strojové zpracování. Jedná se tedy o návrh koncepce zápisu množiny stavů (vstupních, výstupních a přechodových) a způsobu přechodu mezi nimi (výše uvedené trojice). Programovací jazyk vyšší úrovně tedy dokáže přepsat zápis algoritmu programátora do takového systému, a to navíc tak, že jej převede do assembleru konkrétního stroje (CPU).

Vzhledem k matematické nedokazatelnosti obecného algoritmu (Turingova stroje) to ale znamená, že ať bude programovací jazyk sebelépe navržen, pokud má být použitelný v praxi, programy v něm napsané nelze považovat za bezchybné. Co víc, nikdy nelze jednoznačně říct, že program neobsahuje žádnou chybu. Možná je po čase programování již bez chyb, ale člověk to nezjistí.

Vzhledem k tomuto žalostnému matematickému základu výpočtu se Computer Science již po řadu let snaží o kategorizaci algoritmů do několika vzájemně nadřazených tříd. Matematici našli část algoritmů, které lze skutečně matematicky dokázat. Jednotlivé třídy pak popisují pomocí strojů různé úrovně. Jedná se ale o velmi malou množinu prakticky nepoužitelných algoritmů, které pracují většinou nad velmi omezenou množinou vstupních dat, tedy algoritmy pro praxi obvykle málo užitečné. Mezi touto množinou dokazatelných algoritmů, které lze vyjádřit deterministickými konečnými automaty a Turingovým strojem, lze ještě vymezit další třídy algoritmů, jejichž teorie je kompromisem mezi možnostmi použít je na jakýkoliv algoritmus a jejich dokazatelností (lze říct, které části algoritmů v nich kódovaných jsou bezchybné). Je takto známa klasifikace algoritmů, nejčastěji citovaná Chomského klasifikace gramatik, viz [Chomsky1957]. Pojem gramatika se používá jako ekvivalent pro obecné vyjádření stroje dané třídy. Gramatika totiž vyjadřuje programovací jazyk (systém přepisovacích pravidel), tedy formální matematický popis stroje určité třídy. Na kresbě 9 je uvedena

**Chomského
klasifikace
gramatik**

základní klasifikace gramatik podle Chomského a ekvivalent termínu stroje, který popisuje.

typ gramatiky	určení	typ stroje
0	neomezené (recursively enumerable)	Turingův stroj (Turing machine)
1	kontextové (context-sensitive)	Turingův stroj s lineárním prostorem (linear-bounded non-deterministic Turing machine)
2	bezkontextové (context-free)	zásobníkový nedeterministický automat (non-deterministic pushdown automaton)
3	regulární (regular)	konečný automat (finite state automaton)

Kresba I-9: Chomského klasifikace gramatik.

Syntaxe programovacích jazyků, které jsou dnes běžně používané (a již zmiňované), jako jsou Pascal, FORTRAN, C, Java, jsou definovány jako typ gramatiky 2 a i 3 podle kresby I-9 (regulární gramatiky a bezkontextové gramatiky). Existují ale programovací jazyky, jejichž gramatiky vycházejí z dokazatelnosti v nich vyjadřovaných algoritmů, typicky Dijkstrův dokazovací (verifikační) jazyk. Programy v nich napsané ale nemají praktický užitek. Využitím programovacích jazyků, které jsou navrženy pomocí těchto gramatik vyšších typů, se pak programátor vystavuje problémům při kódování některých praktických algoritmů. Vybraný programovací jazyk mu však tuto nevýhodu vynahrazuje případným menším počtem chyb ve výsledném programu. Výběr programovacího jazyka při úvaze (analýze) nad typem kódovaného algoritmu je tedy mnohdy důležitý. Konstrukce a návrh nových programovacích jazyků pak vždy principiálně vycházejí z uvedené matematické teorie.

Poznání této situace vedlo v praktickém programování k vytvoření určitých postupů, umožňujících programovat tak, aby byl výsledný program dobře čitelný, aby jej bylo možné dobře sledovat při jeho provádění a aby do něj mohl snadno vstupovat i jiný programátor než jeho tvůrce. Takto se v rámci technologií programování v Computer Science vymezily pojmy jako je strukturované programování, modulární programování, objektové programování, vrstevnost, ladění programů. Všechny tyto techniky mají za úkol co nejlépe programovaný algoritmus strukturovat, rozložit na části (moduly) a umožnit tak snadněji vyhledávat objevující se chyby při testování a používání programů.

strukturované programování

Strukturované programování (structured programming) je metodika konstrukce programů, při které používáme hierarchickou stavbu programu jako celku. Při analýze problematiky, kterou máme naprogramovat, určíme základní cestu dat (tok dat, data flow), který vyjádříme v hlavní části při psaní programu. Při spuštění takový program vykoná jen základní manipulace s daty. Toto vykonávání testujeme na typických (a i méně typických) datech. Po odladění, tj. zjištění, že program produkuje data na základě vstupních dat požadovaným způsobem, dále definujeme úroveň nižší, tj. tu, která

je z různých míst hlavní části programu odkazována pro dílčí zpracování dat tekoucích hlavním programem. Jednotlivé části nižší hierarchie jsou vytvářeny *procedurami* (nebo funkcemi), které mohou být na různých místech odkazovány opakovaně nebo pouze jednou. Při používání metodiky strukturovaného programování jsou jednotlivé procedury vzájemně datově nezávislé. Jakým způsobem jsou jim z hlavního programu předávána data ke zpracování a jakým způsobem jsou jimi odevzdávány výsledky, je striktně definováno (vstupními a výstupními parametry). Programátor tak v další práci při programování nižší hierarchie programuje takové procedury a ladí je nejprve samostatně, později v celkovém kontextu programovaného algoritmu. Je věcí programátora, na kolik hierarchií program při výchozí analýze problému rozloží a jak je konstrukčně vyřeší. Jisté je, že např. opakující se problémy takto programuje pouze jednou. Nehledě na systém knihoven programovacích jazyků, kdy má programátor k dispozici již řadu procedur, které jsou součástí používaného programovacího jazyka. Jedná se např. o různé matematické funkce (mocniny, odmocniny, siny, kosiny...), ale i o vstupně výstupní funkce (čtení dat z klávesnice uživatele – např. s názvem `scanf`, vytvoření grafického okna, výpis textu do jednoho z oken na obrazovce – např. `printf`, tisk na tiskárně...) a řadu dalších. Součástí dispozic programovacího jazyka, které ovlivňují jeho výběr při rozhodování zda je na daný problém dobře použitelný, jsou také možnosti, které jazyk nabízí právě z pohledu obsahu svých knihoven procedur a funkcí.

Při strukturovaném programování používá programátor sekvenční zpracování tekoucích dat (sequential data flow), postupně krok za krokem vyjadřuje, co se s daty provádí. Vyjma sekvence zapisovaných akcí strukturuje program také pomocí *podmínek* (conditional data flow) a *iterací* (iterative data flow). Podmínky větví program při rozhodování nad stavem dat, kterou částí programu se bude algoritmus ubírat. Iterace je opakování určité části programu po stanovený počet (počet iterací, cyklů). Opuštění tohoto opakování je dáno vznikem určitého stavu dat, tj. např. je spočítán počet cyklů a ten je již maximální a nebo se objevila v datech sledovaná hodnota, jejíž stav je podmínkou k vystoupení z cyklu.

Modulární programování (modular programming) je důsledkem používání metodiky strukturovaného programování. Modul je definován jako samostatný funkční celek, který je používán programem pro vykonání daných operací nad danými daty. Modulem může být systém procedur parciálně volaných z různých částí programového celku. Ale může se také jednat o nezávislý programový celek, který je používán různými programy při jejich provádění tak, že je modul aktivován jako samostatný program, který se napojí na volaný program. Převezme data, zpracuje je, data volajícímu programu odevzdá a skončí svoji činnost. Programy jsou děleny do modulů a již existující moduly jsou vyhledávány a používány při psaní nových programů. Bez techniky modulárního programování je prakticky nemožné, aby se na programování větších programů účastnilo více programátorů.

**modulární
programování**

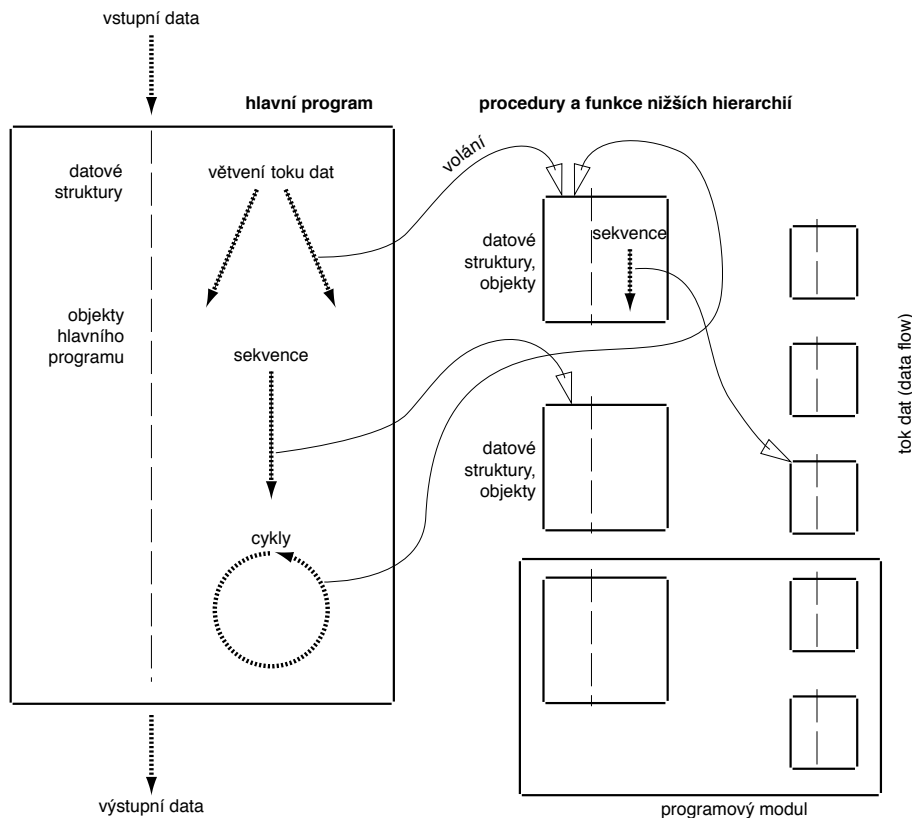
**objektové
programování**

Pokud se programátor rozhodne využívat technologii objektového programování (object-oriented programming), začne používat i jiný způsob myšlení než ten, který je minimální podmínkou strukturovaného programování (sekvence, podmínky, iterace). Pro práci s vyjádřením požadovaného algoritmu modeluje skutečnost pomocí objektů. *Objekt* je vyjádřením struktury dat, nad kterými probíhá výpočet. Znamená to, že model dat je uvažován ve smyslu jeho definovatelných částí (objektů), které jsou atomicky nezávislé ve vyjadřovaném celku dat. Např. chceme-li programovat agendy školní instituce nebo firmy, při objektovém přístupu uvažujeme studenta jako objekt, pedagoga jako objekt, učebnu jako objekt atd. Podobně zaměstnanec firmy je objekt, jemu svěřený klient firmy je objekt, zboží je objekt, dodavatel firmy je objekt, firma je objekt atp. Objekt je vyjádřen datovou strukturou a činnostmi, které lze nad touto datovou strukturou provádět. Objekt se tváří vůči zbylému programu jako nezávislý, je s ním možné komunikovat pouze stanoveným způsobem a sám se nachází v určitém stavu, který je součástí stavu všech dat probíhajícího výpočtu. Objekty jsou typizovány *třídami* (classes). Třídy vytvářejí hierarchii a vzniká tak silně kauzálně definovaný model skutečnosti. Při objektově orientovaném programování používá programátor objektově orientované programovací jazyky. Ty poskytují aparát pro snadné definování tříd a objektů, jejich skládání do programových modulů a programových celků. Objektové programování je technikou, řadu oblíbených programovacích jazyků bylo proto možné pouze rozšířit pro podporu této techniky. Vznikl tak např. jazyk C++ jako rozšíření jazyka C. Programátor v C++ přitom ale může v C++ programovat bez využití této techniky. Typickým současným představitelem objektového programování je jazyk Java, ale i při jeho používání můžeme vytvořit program, který není objektový.

ladění programů

Hlavním cílem uvedených programovacích technik je odstranění co největšího počtu případných chyb, které se mohou při vyjadřování algoritmů Turingova stroje objevit jako vedlejší efekt programátorovy práce. Programátor při své práci postupuje tak, že po napsání prvních verzí programu (obvykle pouze jeho základního skeletu) program testuje nad vybranou skupinou typických dat. Testování nazýváme také ladění programu (debugging, doslovně odstraňování obtěžujícího hmyzu). Možnostmi sledování průběhu výpočtu, případně (statistickým) vyhodnocováním zpracovaných dat programem, obvykle disponují používané implementace programovacích jazyků (vývojová prostředí programovacích jazyků). Programátory obvykle zajímají *dynamické ladící moduly* vývojového prostředí (v žargonu debugger), protože jejich prostřednictvím může programátor program sledovat při jeho běhu, zastavuje jej na určitých místech a prohlíží si stav rozpracovaných dat. Debuggery programátorovi umožňují vyhledávat chybné části programu.

Kresba 10 zachycuje základní principy uvedených programovacích technik.

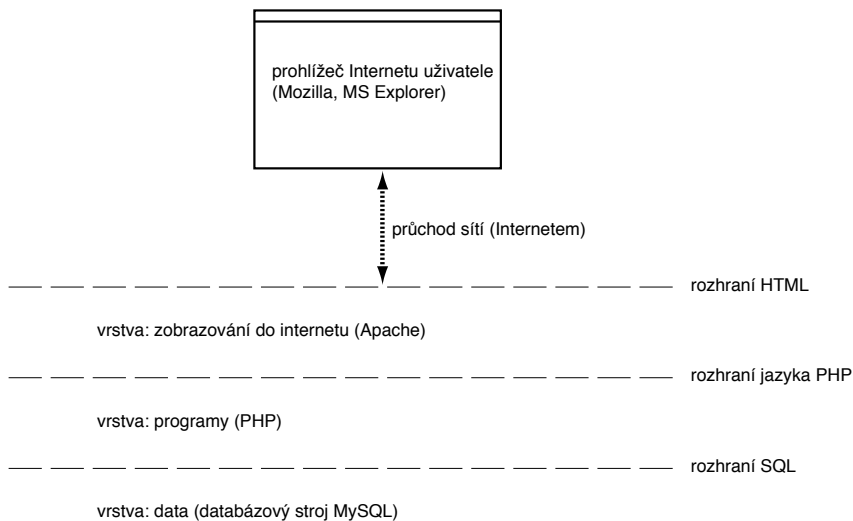


Kresba I-10: Programovací techniky.

Programování složitějších aplikací, programů větších rozměrů, programových systémů (tzv. informačních systémů) a programových systémů řídících samotný výpočetní stroj nebo více strojů současně se účastní více programátorů a více týmů programátorů. Pro každý tým je výhodné využít již existující produkty jiných týmů, zvláště jedná-li se o produkt, který je modulárně využitelný na úrovních různě vzdálených od samotného stroje. Např. tým programující agendy obchodní firmy použije pro ukládání dat databázový software, který programoval tým zabývající se speciálně databázemi. Pro komunikaci s databází existuje smluvený programovací jazyk (dotazovací jazyk, např. SQL), který lze používat stejným způsobem, přestože byl databázový software vyměněn za novější nebo dokonce za produkt jiné firmy. Programový systém agendy obchodní firmy se tak nemusí přepisovat do jazyka jiného databázového software. Říkáme, že větší programové celky jsou konstruovány ve vrstvách. Databázový software je vrstva, aplikace je vrstva. Moduly aplikace pak mohou být dále poskytovány jinému programu (např. statistice). Při jasné definici jazyka jednotlivých vrstev (rozhraní, interface) se zárukou pouze jeho rozšiřování (kompatibility) lze jednotlivé vrstvy vyměňovat za jiné. To, že jsou jednotlivé vrstvy od sebe odděleny

**programování
ve vrstvách**

a komunikují pouze prostřednictvím rozhraní, umožňuje snadněji lokalizovat objevující se chyby a zvyšuje tak kvalitu celého produktu. Kresba I-11 ukazuje vrstvy informačního systému, který uživatel používá prostřednictvím Internetu.



Kresba I-11: Příklad vrstev informačního systému v Internetu.

I.3 – operační systémy, servery a koncové stanice

V centrální jednotce výpočtu (CPU) probíhá výpočet. Procesor postupuje podle instrukcí programu uložených v operační paměti a postupně mění data vstupní na data výstupní. Probíhá proces (zpracování dat). K tomu, aby tento proces mohl být spuštěn a k tomu, aby mohla být výstupní data úspěšně převzata, zobrazena člověku a uložena na externí periférii, je potřeba vykonat řadu podpůrných úkonů. Do operační paměti musí být nahrán potřebný program a rovněž požadovaná data. Program musí být spuštěn. Musíme rozpoznat, kdy program skončil. Výstupní data musíme odebrat. Tyto podpůrné činnosti musí být snadné pro každého člověka, musí být lehce opakovatelné a přirozené. Člověk také musí mít dobrý a snadný přehled o všech programech, které potřebuje pro svou práci a musí je snadno a jednoduše vybírat pro zavedení do operační paměti a spuštění. Musí mít také k dispozici snadný, přehledný a pohodlný způsob evidence svých dat jak vstupních, tak výstupních. Uživatel musí mít snadný, pohodlný a přirozený přístup k perifériím, jako je tiskárna, disketa, CD a počítačová síť. A konečně musí mít uživatel k dispozici snadný, pohodlný a přirozený způsob, jak svá data poskytovat jiným uživatelům a současně musí mít zajištěnou ochranu svých soukromých dat.

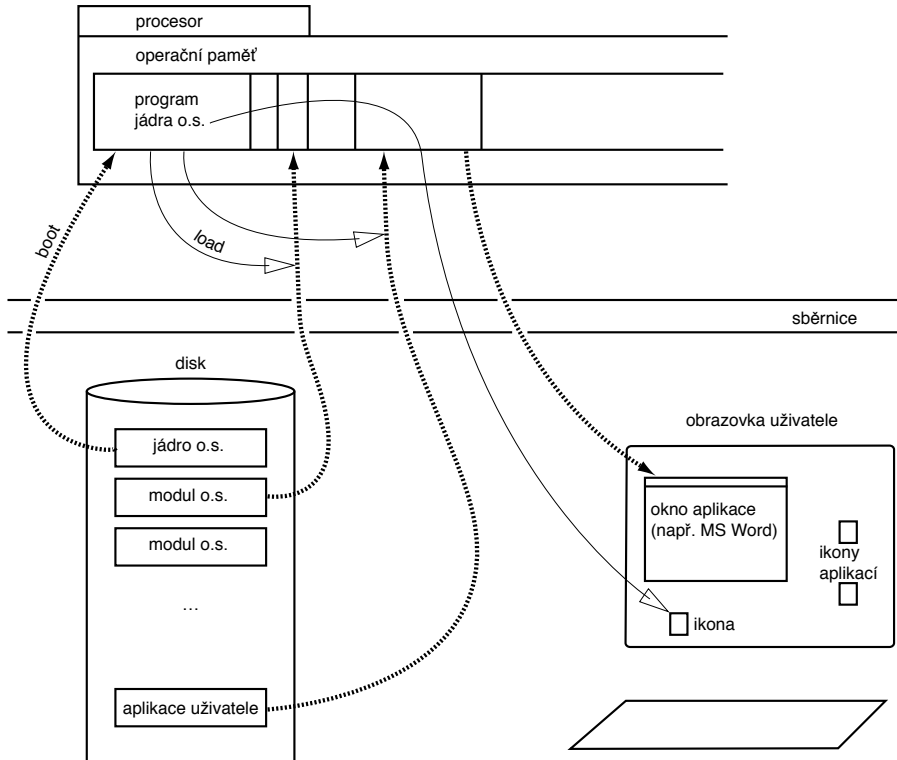
Od počátku praktického používání počítačů lidmi, kteří nejsou specialisté na výpočetní techniku, až po dnešek představuje návrh, řešení systému takových programů a jeho následné uvedení do praxe jeden ze základních problémů Computer Science. Vlastně pouze podpůrný programový celek takového zaměření nazýváme operační systém (operating system).

operační systém

Z technického pohledu se jedná o samovolný start základního programu takového programového celku, jádra (kernel), při zapnutí počítače. Jádro operačního systému je prvním programem, který je zaveden do operační paměti a spuštěn. Říkáme, že tímto krokem je zaveden (boot) operační systém. Jádro „obalí“ hardware a nabízí jej uživateli. Aby stroj po zapnutí rozpoznal právě ten správný program, který je jádrem, je nutno operační systém odborným způsobem na počítač instalovat. Výrobci operačních systémů (neboť operačních systémů je vskutku mnoho) se ale mnohdy snaží dodávat operační systém na vybraný typ počítače tak, aby jeho instalace nepotřebovala specialistu.

jádro

Stav počítače po rozběhu jádra operačního systému a jeho následné práce pro uživatele schématicky ukazuje kresba I-12.



Kresba I-12: Operační systém v operační paměti slouží uživateli.

proces

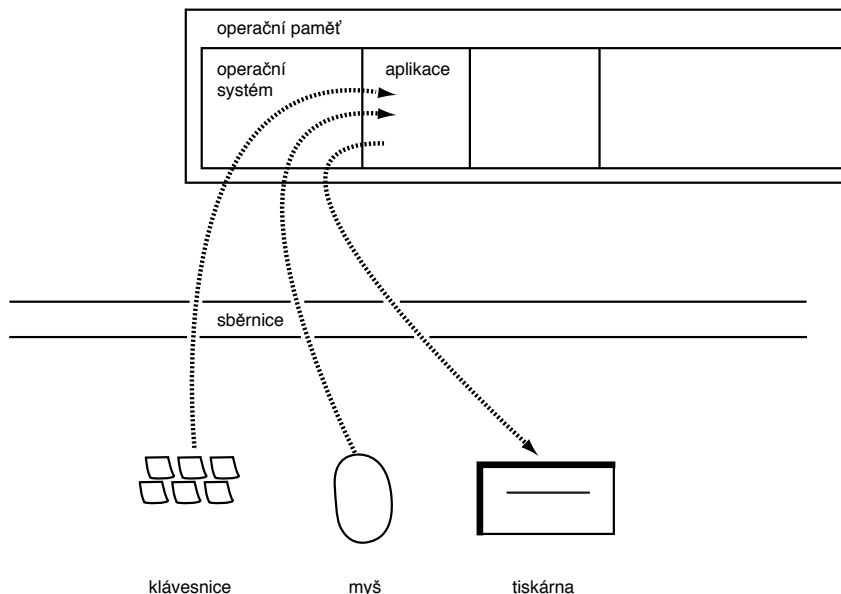
Uživatel prostřednictvím ikony na obrazovce (kterou nabízí jádro operačního systému) dává jádru pokyn ke spuštění aplikace nebo další části operačního systému. Operační systém realizuje pokyn uživatele a z jemu známého místa na disku zavede do operační paměti odpovídající program a předá mu řízení. To znamená, že je program ihned interpretován procesorem tak, aby tento průběh uživatel vnímal. Uživateli se na obrazovce otevře nové okno, je vyzván pro zadání vstupních dat atd. Program prováděný v CPU nazýváme *proces*.

Operační systém je velké rozhraní (interface) mezi samotným strojem (hardware) a uživatelem.

Jak jsme již několikrát zmínili (kresba I-3, I-7 atp.), hlavní průběh zpracování dat probíhá v CPU. Tato činnost je ale úzce spojena s periferiemi, a to ať už z hlediska ukládání, tisku nebo pořizování dat (snímání, stisky kláves) nebo s napojením na reálný svět (čidla snímání fyzikálních hodnot, ovládání strojů, robotů).

Jednou z hlavních funkcí operačního systému je zajišťování komunikace mezi CPU a periferiemi. Obvyklá situace je dána potřebou uživatele, kterou uživatel vyjádří kliknutím myši v určitém místě obrazovky, stiskem klávesy nebo pokynem jiného vstupního zařízení. Periferie na tento pokyn upozorní operační systém. Operační systém toto upozornění od periferie převeze a předá je odpovídající aplikaci, která na ně reaguje např. tak, že operačnímu

systému zpětně předá data, která má vytisknout na tiskárně. Operační systém osloví tiskárnu, které (pokud tiskárna odpoví) předá data k tisku. Situaci jsme znázornili na kresbě I-13.



Kresba I-13: Uživatel – periferie – operační systém – aplikace – operační systém – periferie.

Operační systém je o aktivitě periferie (kliknutí, dokončení tisku) obeznámen částí hardware, kterému říkáme systém přerušení (interrupt handler). Každá periferie, která je ke stroji připojena, má přidělenou paměťovou oblast, do které oznamuje své aktivity. Je věcí programátora operačního systému (systémového programátora), aby zajistil vysokou pozornost operačního systému změnám v této části hardware. Není to úloha snadná. Periferie mohou upozorňovat na své aktivity (a vyžadovat obsluhu operačním systémem) současně. Např. stisk klávesy od uživatele přichází v tomtéž okamžiku, kdy tiskárna ohlašuje ukončení tisku části dat a požaduje příjem dat pro pokračování v tisku. Operační systém například musí také ještě přenášet pohyb myši na monitor uživatele, který je také periferií, která také upozorňuje na již vykreslená data a požaduje data další.

systém přerušení

Výkon CPU je obvykle o několik řádů vyšší než je výkon samotných periferií. Operační systém proto dokáže odpovídat periferiím a uspokojovat jejich požadavky tak, že jejich čekání je z pohledu lidského vnímání prakticky nepostřehnutelné. I přesto ale musí být stanovena priorita obsluhy periferií. Tato priorita je závislá jednak na rychlosti periferií a jednak na potřebách uživatele. Tiskárna je periferie velmi pomalá. Tisk jednoho bajtu zabere tiskárně tolik, co zápis několika tisíce bajtů na diskovou periferii. Operační systém proto zvýhodňuje diskové operace. Také ale zvýhodňuje uživatelskou interakci, tj. každému stisku klávesy nebo kliknutí myši je věnována okamžitá pozornost, přestože prodlevy mezi stiskem jednotlivých kláves při psaní textu vydají na

řadu obsluhu požadavků z jiných periférií. Zcela jistě ale bude operační systém počítače autopilota nadzvukové stíhačky věnovat nejvyšší pozornost čidlům orientace v okolí, které upřednostní před požadavkem zobrazování probíhající navigace na displejích v řídicí kabině.

**sdílení času,
multitasking**

Vzhledem k vysokému výkonu CPU oproti obvyklé reakci reálného světa prostřednictvím periférií (i externí paměti jsou oproti CPU výrazně pomalé), probíhá v CPU běžně vykonávání několika úloh současně. Znamená to, že v operační paměti je zavedeno několik programů současně a v době, kdy některé jim odpovídající procesy čekají na přesun dat z periférií, je prováděn proces, jehož aktivita v současném okamžiku komunikaci s perifériemi nevyžaduje. Situace se ale výrazně komplikuje, protože vytváření fronty pro obsluhu v systému přerušování není sekvenčně řešitelné z hlediska daných priorit. Současně několik procesů totiž může požadovat přesun dat z téže periférie (např. z téhož disku na jeho jiném místě). Operační systém musí proto rozhodovat nejenom mezi perifériemi, ale i mezi procesy a jejich případným zvýhodňováním, a to jak pro potřeby jejich komunikace s perifériemi, tak pro přidělování procesoru, tj. který proces stejných požadavků na operační systém bude vykonáván dříve. Procesů pro další provádění bez požadavku přístupu k perifériím může být ale více. Takovým procesem se navíc stává proces, který data právě od periférie převzal a chystá se je zpracovat. Distribuci výkonu procesoru mezi několik procesů a distribuci jejich přístupu k perifériím nazýváme sdílení stroje a operační systémy, které tyto funkce splňují, nazýváme *operační systémy sdílení času* (time sharing operating systems). Funkci současného provádění několika úloh (procesů) označujeme výrazem *multitasking* nebo *multiprocessing*.

V okamžiku, kdy je procesu přidělen procesor, by tato úloha měla být vykonávána až do okamžiku, kdy dojde k požadavku přenosu dat mezi procesem a periférií. Sdílení strojového času více procesy ovšem znamená také funkci nezanedbání žádného z procesů přijatých ke zpracování. Proto operační systém navíc obvykle sám přerušuje prováděný proces, i když situace přerušování od periférie nenastane. Procesů, které jsou současně prováděny z pohledu uživatele a které nepotřebují v daném okamžiku komunikaci s perifériemi, totiž může být více a operační systém jim rozdělí strojový čas v reálném čase přiměřeně. Iluze o současném (a spravedlivém) provádění více procesů najednou je pak výraznější.

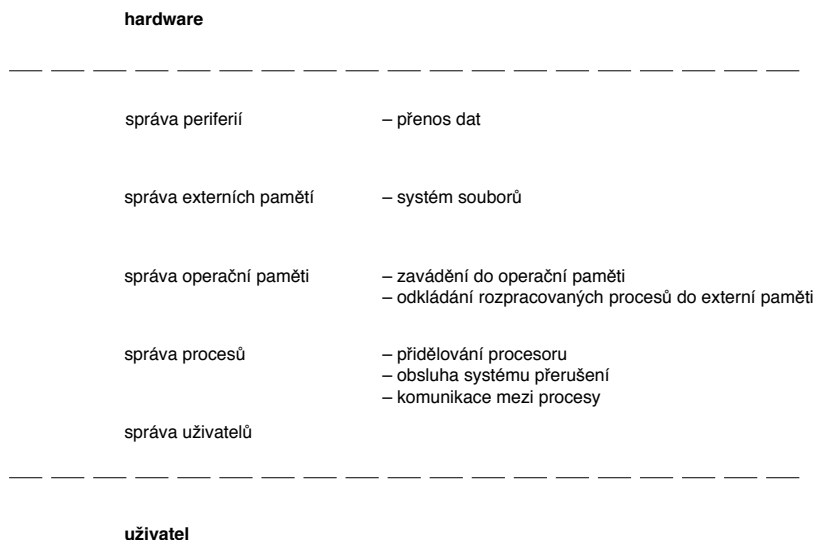
Příkladem sdílení času je běh dvou programů, kdy jsou výsledky jejich výpočtů souběžně zobrazovány ve dvou oknech uživatelské obrazovky. Vykonání obou programů současně by mělo být rychlejší než jejich postupné provedení, protože v době, kdy operační systém přenáší data na periférii obrazovky do jednoho okna, tedy v době čekání na dokončení tohoto přenosu, může probíhat výpočet druhého procesu a opačně.

**více uživatelů,
multiuser**

Předložený princip operačních systémů sdílení času má všechny předpoklady také pro práci několika uživatelů na jednom stroji současně. Z pohledu hardware jde pouze o připojení dalších periférií pro komunikaci s lidmi, tj. obrazovky, klávesnice, myši atp. Z pohledu operačního systému

jde o zajištění především odlišení jednotlivých uživatelů. Toto vzájemné odlišování může být poměrně složité. Např. uživatel, který spustil aplikaci čerpání a zpracování dat nad objemnou databází, musí mít k dispozici mechanismus, pomocí něhož tuto aplikaci předčasně ukončí (např. když si uvědomí, že zadání výběru dat provedl chybně), neboť celkový čas průběhu takového procesu může být dlouhý. Takové vynucené přerušení ale nemůže uplatňovat i vůči procesu jiného uživatele. Operační systém proto musí uživatele registrovat a při jejich práci je vzájemně odlišovat. Podle významu jejich práce vzhledem k provozu operačního systému jim poskytuje nebo naopak odmítá poskytovat aplikace nebo výjimečné služby operačního systému (např. sledování práce operačního systému a tedy i sledování práce jiných uživatelů). Situace je prakticky řešena tak, že uživatelé se před započítím práce musí operačnímu systému prokázat svojí identifikací. V současné době je nejjednodušším způsobem zadání přístupového kódu, který se skládá ze jmenné identifikace uživatele a z tajného hesla. Obě tyto informace uživatel zapisuje před zahájením práce a operační systém mu následně umožní spouštění procesů v rámci stanovených dispozic. Operační systémy, které umožňují pracovat více uživatelům současně na jednom stroji, nazýváme operační systémy víceuživatelské (multiuser).

Kresba I-14 ukazuje seznam základních modulů jádra víceuživatelského operačního systému.



Kresba I-14: Moduly jádra víceuživatelského operačního systému.

Operační systém přitom koordinuje práci uvedených modulů a zajišťuje jejich případnou spolupráci. V rámci každého modulu musí rozhodovat o řadě neobvyklých situací. Např. když několik uživatelů současně požaduje přístup ke stejným datům v případě, že všichni mají právo tato data jak číst, tak je prepisovat. V rámci přidělování operační paměti (v modulu správy

swap

operační paměti) musí např. řešit situaci, kdy je procesů příliš mnoho, takže nedostačuje velikost operační paměti. Pak dochází k odkládání (swap) procesů na externí periférii do speciálního místa (odkládací oblast disku, swap area).

Operační systém musí procesům také umožnit vzájemnou komunikaci. Ta je běžná na úrovni vzájemné synchronizace procesů, vzájemného předávání dat, případně přímé práce nad společnými daty v operační paměti.

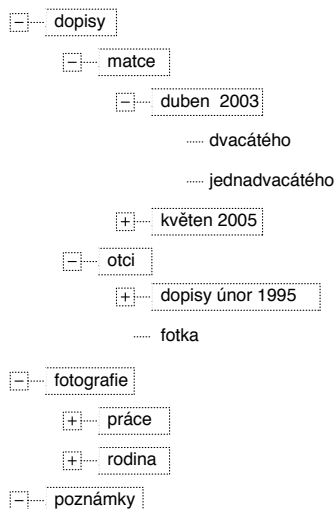
system souborů

S prací operačního systému také souvisí efektivní ukládání dat na externích pamětech. Jedná se o systém souborů (file system), jehož principy úzce souvisí s diskovým principem externích pamětí, který jsme popsali již dříve. Systém souborů ale musí nejenom data výkonně ukládat na externích diskových pamětech, ale musí také zajišťovat jejich distribuci současně pracujícím uživatelům a zamezit neoprávněným požadavkům čtení nebo jiné manipulace s daty jiných uživatelů než jejich majitelů.

soubor

V průběhu vývoje uživatelského přístupu k počítačům se pro práci s externími diskovými pamětmi etabloval hierarchický systém souborů a adresářů. Jedná se vlastně o princip organizace dat. Data jsou ukládána do celků, ve kterých je předkládáme programům. Takový celek nazýváme soubor (file). Obsahem souboru je např. textový dokument nebo obrázek z digitálního fotoaparátu. Soubor je sled bajtů různé délky. Jakým způsobem se s proměnlivou délkou souboru vypořádá operační systém, je právě věcí konstrukce modulu správy systému souborů. Délka souboru se totiž může pohybovat od několika bajtů (dopisy) běžně do stovek MB (digitalizované filmy) a soubor může být prodlužován nebo zkracován. Každý soubor má své jméno, dnes již obvykle libovolné délky. Adresář (directory nebo folder) je evidence více souborů na určitém pojmenovaném místě disku. Adresář má také jméno, takže dokážeme říct, že určitý soubor je uložen na disku pod určitým jménem v některém adresáři. Např. soubor může mít jméno „Dopis mé matce“ a je uložen v adresáři se jménem „dopisy“. Takové určení dat uživatelem na disku by již mnohdy mohlo stačit k tomu, aby operační systém data tohoto souboru poskytl pro právě používaný program (např. aplikaci pro psaní textů). Takto můžeme mít na disku další adresáře, jako např. „poznámky“, „fotografie“ atp. Při delším používání ale přivítáme možnost mít soubory umisťovány do určité hierarchie, např. v adresáři „dopisy“ bychom uvítali členění na „otci“, „matce“ atd. Operační systém poskytuje na každé diskové periférii možnost vzniku takové hierarchie, takže můžeme mít např. situaci podle kresby I-15.

adresář



Kresba I-15: Hierarchie adresářů systému souborů.

Každý adresář může evidovat jak soubory, tak další adresáře (podadresáře, subdirectory), a to obecně do libovolné hloubky hierarchie (soubor `dvacátého` je umístěn v adresáři `duben2003`, ten pak v adresáři `matce`, který je v adresáři `dopisy`). Aplikace prostřednictvím uživatelského výběru pak požaduje po operačním systému data s určením postupně všech jmen adresářů a nakonec i jména souboru s daty. Takto je dáno jednoznačné umístění souboru, přestože samotná jména souborů a adresářů se mohou v různých částech hierarchie shodovat. Uvedení hierarchické sekvence postupně všech adresářů a jména souboru pro označení dat nazýváme *úplná cesta* (full path) k datům. Úplná cesta začíná ve výchozím adresáři diskové periferie (kořenový adresář, root). Některé operační systémy vyžadují jako součást cesty také označení samotné diskové periferie, např. písmenem a dvojtečkou, např. `C:` nebo `D:`, za kterým následuje označení dat na periférii, např.

úplná cesta

`D:\dopisy\matce\duben2003\jedenadvacátého`

Jiné operační systémy sdružují všechny diskové periferie do jednoho celku, operační systém je automatizovaně spojuje do jedné hierarchie, takže v operačním systému UNIX by uvedené označení bylo

`/home/petr/dopisy/matce/duben2003/jedenadvacátého`

Adresář `home` je výchozím adresářem více uživatelů, `petr` je výchozí adresář uživatele. V této oblasti jsou mu data čitelná a dostupná pro změny. Zde se pak uživatel o rozlišování diskových periférií nestará, mnohdy ani neví, kde jsou jeho data fyzicky uložena.

Oba v příkladu použité typy operačních systémů používají jiný znak zvláštního významu pro určení hierarchických úrovní adresářů. Je to buď znak `\` nebo `/`. V odpovídajícím operačním systému jej pak nelze použít jako jeden ze znaků jména souboru či adresáře.

Uvedená hierarchie adresářů je uživatelem vnímána jako strom, na jehož různých místech jsou jako listy umístěvány soubory s daty, tj. nejenom v koncových adresářích hierarchie (na kresbě I-15 viz např. soubor *fotka* v adresáři *otci*). Adresář tak eviduje soubory s daty a podadresáře, kterými hierarchie pokračuje.

Systém adresářů je organizací umístění dat na diskových periferiích. Tuto organizaci zajišťuje operační systém jako režijní datovou část na fyzickém médiu. A modul systému souborů operačního systému jej ve formě stromu předkládá uživateli.

Uživatel má možnost do hierarchie adresářů a umístění souborů zasahovat, adresáře a soubory vytvářet, přejmenovávat a odstraňovat. Tyto operace může provádět všude tam, kde jsou určena jeho práva pro manipulaci s daty a tak, aby nepoškodil jednak data samotného operačního systému a jednak data ostatních uživatelů.

vrstvy operačního systému

Operační systém jako programový celek je sestaven ve vrstvách. Vrstvy jsou uvažovány ve vzdálenosti vytvoření umělé reality od hardware a jejich umístění odpovídá rozložení na kresbě I-14.

Vrstva nejbližší k hardware je označována jako strojově závislá část (machine dependent) operačního systému. Jedná se o práci s CPU a o práci sběrnice na úrovni assembleru a o části, které se věnují práci s periferiemi, tj. ovladače (drivers). Je-li operační systém konstruován pro určitý typ hardware, musí být strojově závislá část napsána speciálně pro něj. Vzhledem k tomu, že k hardware ale mohou být připojovány různé periferie, musí být také stanovena obecná metoda, jakým způsobem je možné ovladač pro určitou periferii naprogramovat a připojit k operačnímu systému. Výrobce periferií totiž nemusí být současně výrobcem počítače a výrobce operačního systému může být opět někdo jiný.

Horní vrstvy operačního systému jsou dnes již obvykle programovány ve vyšších programovacích jazycích. Umožňuje to tak jejich lepší ladění a také vývoj pro různý hardware.

Nad strojově závislou vrstvou operačního systému jsou umístěvány moduly vrstvy komunikace s periferiemi. S ní úzce souvisí moduly správy systému souborů.

Další vrstva je odpovědná za samotný provoz operačního systému, tedy operace s procesy v rámci jejich umístění v operační paměti a řízení jejich provádění.

Jako vrstva nejbližší uživateli je označována správa uživatelů. Je to paradoxní, protože uživatel přistupuje k počítači prostřednictvím hardware (klávesnice, myši, obrazovky, polohovacího zařízení). Principiálně je ale výpočetní systém vytvářen pro jeho potřeby, člověk je tedy v rámci hierarchie konstrukce operačního systému umístěn na nejvyšší přístupovou vrstvu jako hlavní konzument a řídicí element celého výpočetního systému.

Zvládnout implementaci operačního systému je náročné. Computer Science je věda, která je velmi úzce spojena s praxí, a to např. také ve smyslu návrhu konstrukce a praktického naprogramování operačního systému. Snaha vytvořit teoretické zázemí principů operačních systémů pochází od doby vzniku prvních hardware. Dodnes není uspokojivě ukončena, přestože se může na první pohled zdát jako nenáročná. Realizace modelování skutečnosti na reálných strojích podle principů digitalizace vstupních údajů je problém stavby inteligentně se chovající organizace hmoty anorganického původu. Současné operační systémy ovlivnila řada teoretických prací a praktických realizací mnoha významných osobností. Jejich přehled je uveden v části Literatura (viz např. [MadnickDonovan1974], [Bach1986], [Custler1993], [Plan9 2002] atd.).

V praktické Computer Science již po léta dominují dva základní problémy: návrh architektury procesoru a návrh architektury operačního systému. Obecný tlak uživatelů je na sjednocení. Návrháři ale neustále hledají technologie, které by zlepšily současný stav, a to bez ohledu na matematickou nedokazatelnost vyjádření obecného algoritmu. Přestože je kritériem mnohdy především obchodní úspěch, o kvalitní technologie je stále zájem. Již v době počítačové pravěky, na sklonku 60. let minulého století, vznikla (organizovaná i neoficiální) iniciativa sjednotit tehdy takřka nepřehledné množství různých typů operačních systémů různých výrobců výpočetní techniky. Do té doby byl operační systém vždy součástí stroje produkovaného výrobcem a koncepce, realizace a používání operačního systému byly vždy tímto výrobcem jednoznačně stanoveny. Vývoj v 70. letech ukázal, že sjednocení koncepce operačních systémů má v praxi výraznou odezvu. Začátkem 80. let s nástupem počítačů typu PC, tedy osobních počítačů, kdy uživatelů hodně přibýlo, tento tlak ještě více zesílil.

Přestože má návrh počítače a jeho operačního systému skutečně obecné principy, můžeme z dnešního pohledu rozlišit dva základní způsoby používání počítačů, které také ovlivňují charakter jejich operačních systémů (a i samotného hardware). Je to způsob jejich používání, který je osobní nebo serverový.

**osobní počítače
a servery**

Osobní způsob používání počítače je dnes jednoznačně spojován s počítači typu PC. Toto označení se vžilo pro konkrétní typ malých stále výkonnějších počítačů. S označením Personal Computer je uvedla na trh firma IBM začátkem 80. let. Velmi brzy se PC stal nejpopulárnějším na světě a výrazně ovlivnil veškerý trh s výpočetní technikou. Přesto musíme do kategorie osobního používání zařadit i jiné počítače, jejichž označení obvykle bývá Workstation (pracovní stanice). Jedná se o předchůdce PC, které nástup PC postupně v jejich funkci nahrazuje. U pracovních stanic se vždy jednalo a jedná o kvalitní výpočetní stroje zaměřené především na grafickou práci návrhářů různých oborů (strojní a stavební konstruktéry, grafiky atd.). Vyznačují se především koncepčním oddělením zpracování grafických a výpočetních částí programů (přidáním dalších grafických procesorů, koprocessorů), s čímž je spojen také rys velkých a kvalitních monitorů. Jejich koncepce je více propracovaná než

je tomu u PC, a to jak u hardware tak software (operačních systémů ale i samotných aplikací). Byly vždy a stále jsou výrazně dražší než PC a jsou stále pro PC koncepčním vzorem. Typické a nejznámější byly pracovní stanice firem Sun, SGI, IBM, Hewlett Packard. Kategorizace přitom není při dnes tak bouřlivém vývoji nikdy dost přesná, ukázkou nezařaditelnosti jsou počítače pro osobní použití firmy Apple (počítače Macintosh), které svým výkonem a koncepcí stojí na pomezí PC a pracovních stanic. Osobní způsob používání je charakterizován přizpůsobením koncepce pro využívání jednou osobou. U řady typů je tak mnohdy pomíjena vlastnost víceuživatelského přístupu a někdy i víceprocesového zpracování.

Serverový způsob používání počítačů je výrazně starší a z jeho koncepce vycházejí všechny současné trendy dalšího vývoje. Jedná se o využívání jednoho počítače, jehož výpočetní zdroje (data, periférie atd.) jsou poskytovány současně více lidem (dnes i několika tisíci) najednou. V době před zlevněním počítačů na cenu dostupnou jedné osobě ani nemohl být jiný než sdílený způsob používání počítače možný. Koncepce operačních systémů proto tehdy striktně předpokládala současný přístup několika uživatelů prostřednictvím více klávesnic a monitorů k jednomu CPU a sběrnici připojených dalších periférií (především externích diskových pamětí, tiskáren atd.). Operační systém tak byl striktně koncipován jako víceuživatelský a víceprocesový a jeho údržba byla a je stále provozně náročná. Vzhledem k možnosti přístupu uživatelů prakticky kdykoliv je nutno zajistit nepřetržitý provoz. Vzhledem ke sdílení diskových pamětí musí být zajištěno jejich pravidelné zálohování, musí být zajištěna vzájemná oddělenost uživatelů a jejich vzájemná ochrana a bezpečnost. Význam serverového způsobu používání je dnes především v komunikaci lidí a v přímé dostupnosti zveřejňovaných informací. Přestože v rámci zlevnění technologií existuje snaha výrobců PC nabízet i serverová řešení, dominantní stále zůstávají silní výrobci původních serverových řešení jako jsou IBM, Sun a Hewlett Packard.

Sjednocení koncepce operačních systémů serverů se ustálilo v operačním systému UNIX (viz např. [Skočovský1998]). Je původním produktem firmy AT&T a jeho rozšíření a přijetí mezi významné výrobce serverů proběhlo v 80. letech minulého století. UNIX je obecné označení, každý výrobce dnes používá pro praktickou realizaci (implementaci) vlastní značku. Např. IBM používá označení AIX, Hewlett Packard HP-UX nebo True64 UNIX, SGI IRIX, Sun Solaris. Výchozí snaha autorů UNIXu (Ken Thompson, Denis Ritchie, Brian Kernighan, A.S. Bourne) byla v definici a praktické realizaci prostředí práce člověka s počítačem, a to jak uživatele, tak programátora. Výsledkem ale bylo navíc přijetí i obecných principů funkcí jednotlivých vrstev operačních systémů, jako je např. systém souborů nebo moduly správy řízení a plánování procesů. Dnes slavné publikace, které popisují vnitřní strukturu UNIXu [Bach1986] platí prakticky beze zbytku na každou jeho současnou implementaci. Na bázi UNIXu byla také počátkem 90. let definována a vydána norma ISO operačních systémů s obecně známým názvem POSIX, viz [POSIX2003]. Přestože je UNIX pro serverová řešení jednoznačně dominantním operačním systémem, stále jsou různými výrobci

podporovány a v praxi uživateli vyžadovány i jiné operační systémy, které se etablovaly před UNIXem nebo v průběhu jeho vývoje (mateřské operační systémy výrobců hardware). Např. je to operační systém VMS dnes již neexistující firmy DEC (Digital Equipment Corporation), v současné době patřící firmě Hewlett Packard a jí také stále podporovaný. Jeho autorem byl v 70. a 80. letech Dave Cutler, později vedoucí týmu operačního systému Windows NT firmy Microsoft.

Dominantní operační systémy osobního způsobu používání počítačů pochází od firmy Microsoft pro počítače PC. Jejich vývoj byl vždy zaměřen především na používání jedním člověkem. První operační systém bez grafické podpory nesl název MS DOS a umožňoval jednoduchým způsobem startovat programy bez možnosti běhu takových programů současně. Obchodní úspěch firmě Microsoft přineslo ale především uvedení na trh operačního systému MS Windows, který je na této planetě snad nejznámějším prostředím pro grafické manipulace s výpočetními zdroji. Jeho koncepce byť grafických operačních systémů však po léta zůstávala stále na úrovni jednouživatelské a kupodivu prakticky i jednoprosesové. Bylo možné zavést do operační paměti několik procesů, ale prováděn byl pouze ten, na který uživatel zaměřil svoji pozornost určením aktivního okna. Průlom přinesla koncepce operačního systému s označením MS Windows NT (kterou navrhoval Dave Cutler). Jednalo se o ambiciózní projekt, který se snažil o obecné zajištění požadavků praxe, pohodlného známého uživatelského rozhraní a zajištění obecného sdílení a bezpečnosti dat a periférií více uživateli (multitask, multiuser). Přestože jej firma Microsoft implementovala především na procesory firmy Intel, bylo striktně dodrženo oddělení strojově závislé části od vyšších vrstev, takže bylo možné jej implementovat (a také implementován byl) i na jiný hardware, např. některých pracovních stanic. Jeho podřízenost jednomu uživateli, kterému, jak byli uživatelé operačních systémů MS Windows zvyklí, je dovoleno vše, jej ale stále nedokázala postavit na úroveň operačních systémů pracovních stanic a kvalitě operačních systémů serverového přístupu (přestože MS Windows NT je také klonován na serverové řešení).

Třebaže jsou dnes nejznámější operační systémy MS Windows, práce s grafickými okny ještě před jejich vznikem byla použita v operačním systému počítačů Macintosh s označením Mac OS (do své verze s označením X rovněž jednouživatelský a jednoprosesový), přestože definice oken, jak je dnes známe, pochází z laboratoří firmy XEROX. Mac OS je mateřský operační systém firmy Apple a je významným konkurentem platformy PC po celém světě. Operační systémy pracovních stanic se kupodivu ustálily především na platformě operačního systému UNIX. V UNIXu v průběhu 80. let bylo koncipováno a implementováno obecné grafické prostředí s označením X Window System, viz [XWindow1993], které tomuto ustálení výrazně napomohlo, nehledě na skutečnost, že výrobci pracovních stanic byli v mnoha případech také výrobci serverů, vývoj stejného operačního systému pro oba způsoby používání výpočetní techniky byl pro ně podstatný. Tomuto trendu se přiklonila i firma Apple. Dříve s operačním systémem A-UX, později s Mac OS X, oba jsou koncipovány na bázi UNIXu.

Podobně jak operační systémy pro osobní používání (MS Windows) aspirují na serverové využívání, jak již bylo uvedeno, i serverové operační systémy získávají oblibu u uživatelů osobního způsobu používání počítačů. Velmi vážným konkurentem operačních systémů firmy Microsoft je dnes implementace operačního systému UNIX pro hardware typu Intel, která má označení Linux. Jedná se o otevřené společenství volně (tj. bezplatně) šířeného operačního systému především pro grafický jednouživatelský způsob práce s počítačem. Žádná z funkcí výchozích principů UNIXu zde ale nebyla porušena, Linux je proto vybaven výkonným zajištěním současné práce více procesů a uživatelů s dodržením všech současných technologií pro bezpečnost dat. Proto je dnes Linux také používán jako serverový operační systém podnikového řešení menších místních sítí.

Při bližším zkoumání se skutečně rozdíl mezi osobním a serverovým způsobem používání postupně vytrácí. Specializace na osobní přístup byla vynucena jednak historicky a jednak stále nedostačujícím výkonem. Grafické operace vyžadují vysoký výkon, který nemůže být odebírán procesům pro zajištění práce ostatních uživatelů, byť pracujících mimo grafické potřeby. Řešení je mimo jiné i v distribuci výkonu mezi navzájem propojenými počítači, v počítačové síti, která je jedním z předmětů další samostatné části tohoto textu.

I.4 – uživatelé a jejich data, průzkum, adresáře a soubory

Hlavní součásti použitelného počítače jsou hardware a software.

Software je nehmotná organizace průběhu výpočtu. Hardware je vyjádření potřeb této organizace hmotnými prostředky.

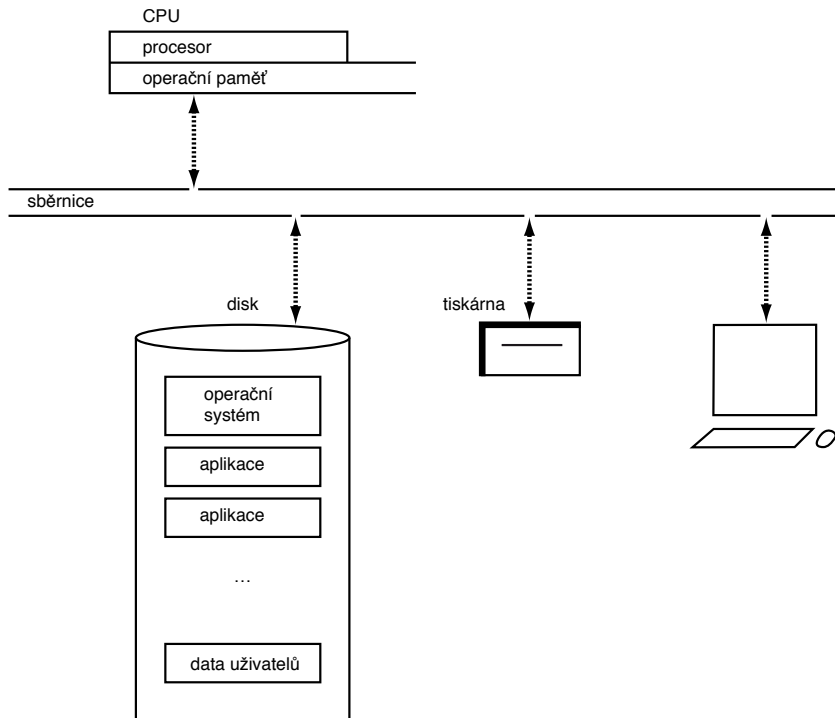
Software s hardware pracuje na úrovni operačního systému tak, že umožňuje provádět člověkem požadované výpočty. Výpočet je realizován aplikacemi, které si uživatel vybírá pro své potřeby nebo jsou programovány jeho potřebám na míru. Z praktického hlediska je v okamžiku předávání počítače uživateli součástí počítače hmatatelný hardware a nehmotný software.

Nehmotný software se dělí na operační systém a aplikace. Běžný výrobce počítače předává uživateli hardware, na kterém jsou všechny potřebné součásti software instalovány, a to dle požadavků uživatele. Produkt hardware z továrny je totiž *holý stroj* (machine). Operační systém nebo aplikace jsou vyrobeny jako software na jiných počítačích do podoby instalační sady. Instalační sada je několik CD nebo DVD, dříve to byly diskety nebo magnetické pásky. Specialista z instalační sady na holém stroji instaluje nejdříve operační systém. Použije k tomu programy z instalační sady, které jsou uzpůsobeny k práci na holém stroji. Po instalaci operačního systému následně specialista instaluje požadované aplikace.

Součástí dodávaného počítače jsou ale také použité instalační sady operačního systému i kupovaných aplikací. Nezbytná je také dokumentace, jejímž prostřednictvím se člověk učí s aplikacemi a operačním systémem pracovat.

Při práci s počítačem uživatel produkuje svá data, která jsou ukládána na externích pamětech a jsou velmi cenná. Prodávané aplikace a operační systémy jsou rozmístěny po celém světě v milionech kopiích, uživatelova data jsou ale jediná na světě. Je proto velmi důležité, aby uživatel věděl, kam umísťuje svá data v počítači, jak jsou pro něj cenná a zda od nich má vytvořenu kopii. Digitální záznam v případě přepisu je totiž nadobro ztracen, podobně jako jsou ztraceny informace zachycené na papíře, který shoří.

Na holý stroj – hardware je instalován operační systém, aplikace a po určité době používání jsou v něm postupně uložena i uživatelova data tak, jak se snaží zachytit kresba I-16.



Kresba I-16: Hardware, software, data.

Operační systém je uložen na externí diskové paměti. Není-li počítač zapnut, CPU a sběrnice jsou zcela nečinné, operační paměť je prázdná. Po zapnutí holého stroje má vnitřně CPU nastavenou aktivitu vyhledání místa, kde je na externí diskové paměti uloženo jádro operačního systému. Zavede jej do operační paměti a spustí jej jako první program. Jádro operačního systému provede inicializaci CPU, operační paměti, sběrnice a periférií a otevře počítač uživateli tím, že mu umožní se přihlásit. Operační systém při svém startu také rozbíhá programy, které pracují pro podporu jednak samotného chodu stroje (např. program zařazování požadavků na tisk tak, aby se vzájemně nesmíchaly) a jednak pro podporu aplikací (např. příjem a odesílání pošty). Po přihlášení pak na pokyn uživatele operační systém rozbíhá, řídí a ukončuje uživatelem požadované aplikace.

**obyčejný
a privilegovaný
uživatel**

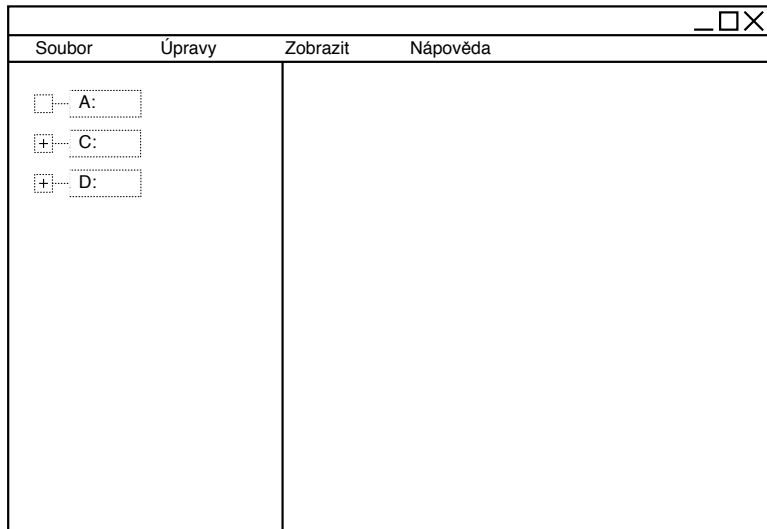
Operační systém, aplikace i uživatelova data jsou uložena na externí diskové periférii v systému souborů a adresářů tak, jak jsme si ji stručně popsali v předchozím textu. V operační paměti se pracuje s jejich kopiemi. Tato práce přitom přináší proměnu dat, a to zejména dat uživatelských. Obvykle je část externí diskové paměti obsahující operační systém a aplikace také viditelná pracujícímu uživateli. V dobrém operačním systému mu je ale pro přepis nedostupná, protože nekvalifikovaný zásah do části s uloženou aplikací nebo dokonce s operačním systémem může paralyzovat chod aplikace nebo i celého počítače. Ke kvalifikovaným zásahům do instalovaných

aplikací nebo operačního systému slouží přístup prostřednictvím speciálního uživatele, jehož práva jsou za tímto účelem vymezena. V operačním systému UNIX je to uživatel se jménem `root`, v operačním systému MS Windows NT je jeho jméno `Administrator`. Oproti jiným uživatelům, které řadíme do kategorie obyčejný, tento uživatel je privilegovaný, jedná se o přístup správce operačního systému. V serverových instalacích údržba práce počítače často zamezí obyčejným uživatelům přístup pro čtení i k částem s operačním systémem a aplikacemi. Obyčejní uživatelé pak při své práci tyto části diskových periférií vůbec nevidí. Souvisí to s bezpečností provozu. Je také věcí vzájemné diskrétnosti a ochrany soukromí dat mezi uživateli, aby údržba zajistila viditelnost pouze dat přihlášeného uživatele. Spolupráce uživatelů na společných projektech a sdílení dat se pak řeší speciálními aplikacemi. Každopádně je jisté (a nutné), že jakákoliv data jsou viditelná privilegovanému uživateli. Kvalita údržby serverů (ale i osobních počítačů) tedy úzce souvisí s kvalitou osoby provádějící údržbu a měla by být vždy zajištěna smluvně právní formou.

Hranice mezi aplikacemi a operačním systémem je mnohdy z pohledu obyčejného uživatele velmi obtížně stanovitelná. Je to tím, že součástí operačního systému je nikoliv pouze jeho jádro, které je při chodu počítače vždy v operační paměti, ale i pomocné programy, které mohou pracovat v pozadí (background), jako např. zmiňovaný program řazení tisků. Jedná se ale např. i o programy, které se chovají jako typické aplikace, a přesto jsou výrobcem dodávány jako součást operačního systému. Nazýváme je systémové aplikace. Takovým programem je jistě program výchozího zobrazení pracovní plochy (desktop), který se rozběhne uživateli po jeho přihlášení. Je jím ale také program, pomocí něhož může uživatel prohlížet externí diskové paměti. Vzhledem k tomu, že se jedná o základní přístup k orientaci v počítači, musí jej uživatel při své práci dobře ovládat jako základní potřebný vědomostní aparát. Tento program bývá označován různými jmény. V UNIXu je jeho obecné označení shell nebo file manager, ve světě MS Windows je to Průzkumník (PC Explorer). Velmi rozšířený a dobře pracující je ve světě PC produkt firmy C. Ghisler & Co. s názvem Total Commander (do listopadu 2002 Windows Commander), viz www.ghisler.com, jehož předchůdcem je Norton Commander. Jméno Norton (viz [Norton1986], [Norton1994], [Norton2002], [Symantec]) je slavné podobně jako název firmy Microsoft, a to z pohledu kvalitních systémových aplikací programovaných do prostředí právě operačních systémů firmy Microsoft. Aplikacím, které nejsou součástí uceleného produktu operačního systému a přesto zastupují jeho funkce, říkáme produkty třetí strany. Total Commander a Norton Commander jsou jedny z nejznámějších aplikací třetí strany pro operační systémy MS Windows. Aplikace Norton Commander byla inspirací také pro další programy tohoto typu, proslulý byl Volkov Commander a v operačním systému UNIX dodnes velmi oblíbený Midnight Commander (spouštěný příkazem `mc`).

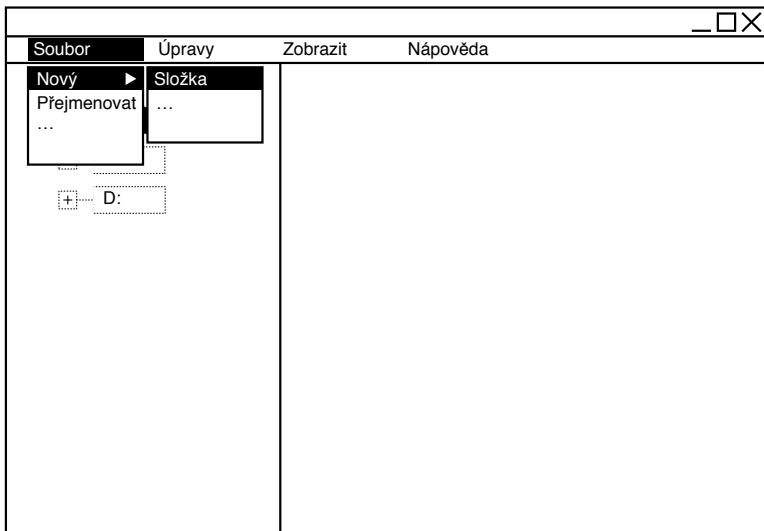
Organizace ukládání dat na externích diskových pamětech (pevné disky, CD, diskety, výměnné disky, paměti typu flash) vychází ze stromové struktury adresářů,

jak jsme uvedli již dříve. Každá externí disková paměť nabízí práci s takovou strukturou. Např. prostřednictvím programu typu Průzkumník je viditelná struktura, která je podobná našemu obecnému vyjádření na kresbě I-17.



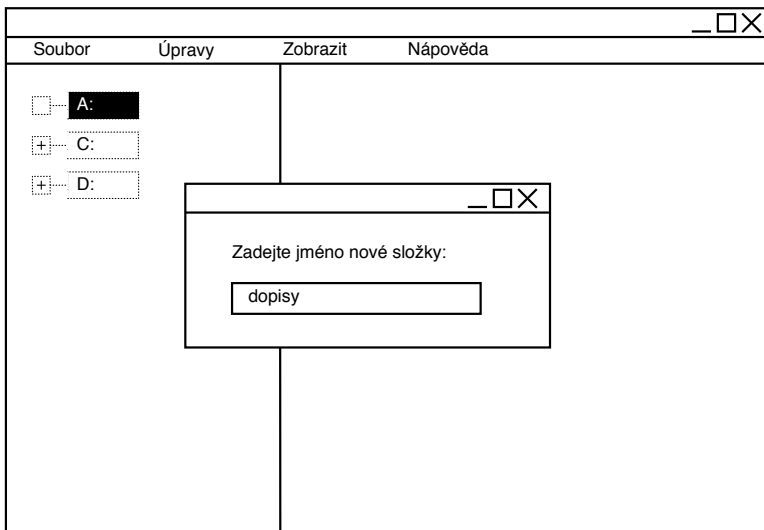
Kresba I-17: Program zobrazování obsahu diskových pamětí.

Okno je svisle rozděleno na dvě části. V levé části jsou zobrazeny přítomné diskové paměti (disketa s označením `A:`, pevný disk `C:`, disk typu CD s označením `D:`). Pokud vložíme do disketové mechaniky disketu a klikneme na ikonu mechaniky myši, program čte disketovou mechaniku a v pravé části okna zobrazí její obsah. Pokud je disketa nová a prázdná, je prázdná i pravá část okna aplikace, pouze je zvýrazněna ikona disketové mechaniky. Na prázdnou disketu můžeme ukládat svá data, např. dopisy, které později tiskneme a posíláme je matce nebo otci. Vzhledem k tomu, že chceme dopisy matce od dopisů otci pořádkumilovně oddělit, můžeme si pro každého z nich na disketě vytvořit zvláštní přihrádku, k tomu použijeme adresář. Kliknutím na text `Soubor` v záhlaví okna je vysunuto menu, z něhož použijeme první možnost, získáme vysunutí dalšího menu a kliknutím potvrdíme jeho první položku, tedy `Nový -> Složka`, viz kresba I-18.



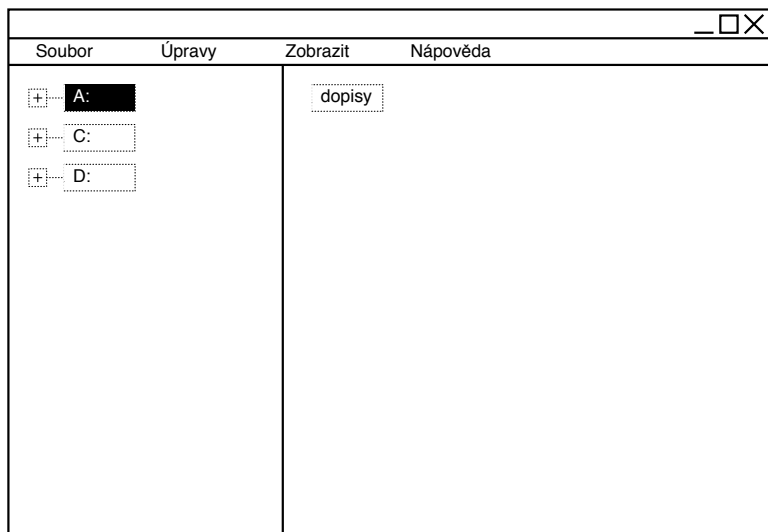
Kresba I-18: Vytvoření nového adresáře.

Po kliknutí na vybranou možnost vytvoření adresáře program požaduje vysání jména nově vznikajícího adresáře, jak ukazuje kresba I-19.



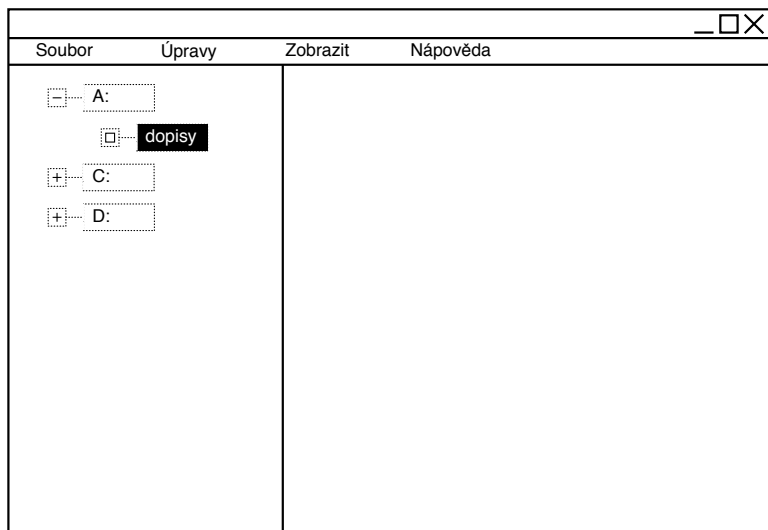
Kresba I-19: Pojmenování nového adresáře.

Napíšeme text `dopisy`. Okno aplikace zobrazí novou situaci, a to v detailu místa, kde pracujeme, tj. diskety, viz kresba I-20.



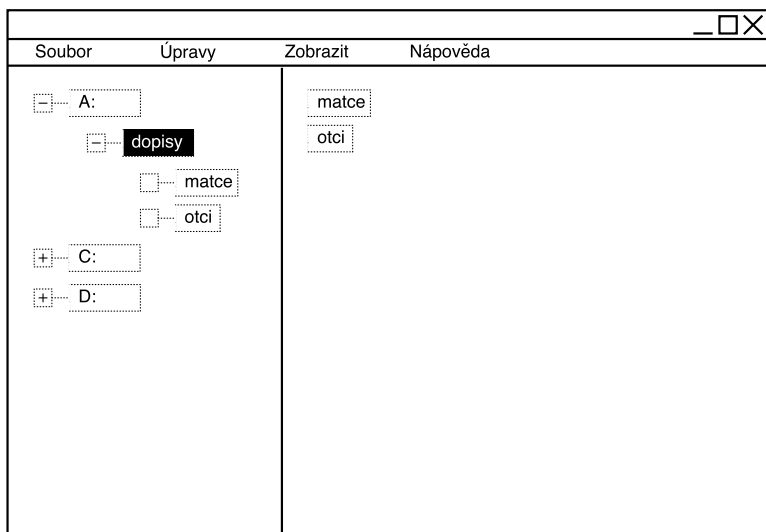
Kresba I-20: Adresář na disketě.

Disketa je zobrazena i v levé části a vzhledem k tomu, že jsme na ní vytvořili nový adresář, kterým na ní začíná stromová struktura dat, objeví se před její ikonou znak + jako symbol pro možnost rozvinutí struktury stromu adresářů diskety také v levé části okna. Udělejme to a přejděme kliknutím na adresář `dopisy`, jak ukazuje kresba I-21.



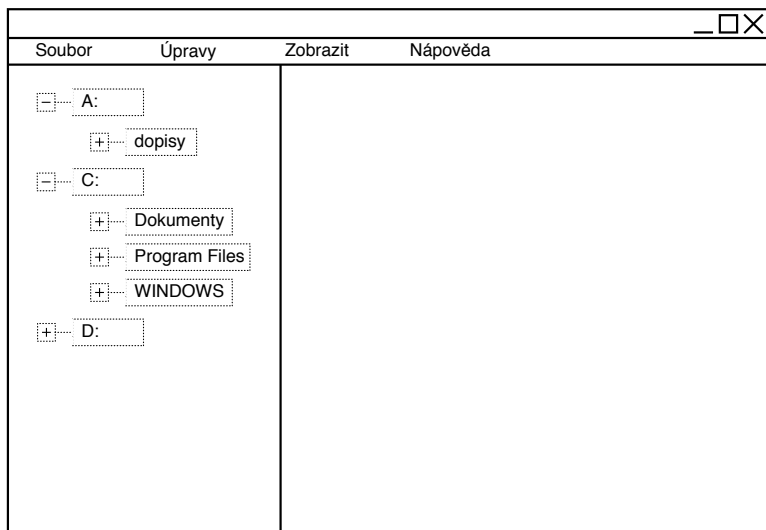
Kresba I-21: Vznikající struktura dat na disketě.

V místě, kde se právě nacházíme (`dopisy` – adresář, který je zvýrazněn, protože jsme si jej vybrali kliknutím myši), můžeme opět stejným postupem vytvořit další nové adresáře, postupně se jmény `matce` a `otci`, jak je uvedeno na kresbě I-22.



Kresba I-22: Struktura pro dopisy matce i otci.

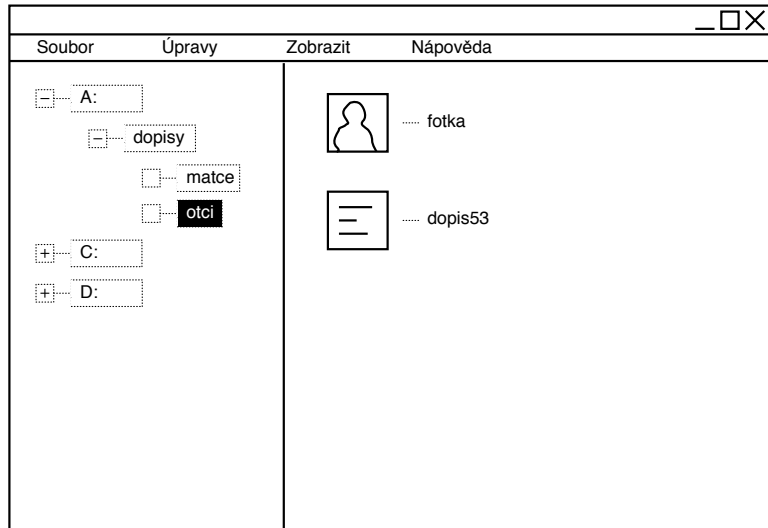
Kliknutím myši na různé symboly (ikony disků, znaky + a -, ikony adresářů) v levé části okna této aplikace můžeme otevírat, prohlížet, případně měnit stromovou strukturu adresářů na diskových periferiích jako je disketa A:, disk C: nebo CD jako D:. Např. viz kresba I-23.



Kresba I-23: Příklad stromové struktury systémového disku C:.

Na vnitřním disku s označením C: jsou již vytvořeny adresáře. Oblast operačního systému je v adresáři WINDOWS, instalované aplikace jsou uloženy v adresáři Program Files. Uživatel může pracovat s aplikacemi a ukládat svá data do adresáře Dokumenty.

Adresáře určují stromovou strukturu organizace ukládání dat. Ikony souborů s daty jsou obvykle výrazně odlišeny od ikon adresářů. Ovšem pohled do diskových pamětí počítače uživatel využívá především pro nalezení svých souborů s daty. Ikony souborů s daty dnes již obvykle vyjadřují typ jejich obsahu, takže je možné kliknutím myši na soubor s daty nastartovat aplikaci, která je operačním systémem určena pro zpracování dat obsahu takového souboru. Např. kresba I-24 zobrazuje průzkum části disku s adresářem `otci`.



Kresba I-24: Soubory adresáře `otci`.

V levé části okna jsme myší klikli na adresář `otci`. Klikneme-li dále na pravé straně na soubor `fotka`, naše aplikace průzkumu disku iniciuje start aplikace, která fotku zobrazí, případně ji umožní editovat. Rovněž obsah textu v souboru `dopis53` přečteme v aplikaci, kterou průzkumník nastartuje jakmile na soubor klikneme myší.

Cesta od využívání prohlížečů diskových oblastí k vyhledání svých dat a následném startu odpovídající aplikace je základní cesta uživatelského využívání počítače. Vlastně každý prohlížeč slouží pouze k průzkumu diskových pamětí a pro samotné zpracování dat jsou používány jiné aplikace, které prohlížeč rozeznává a startuje.

řády délek ukládaných dat

Při zobrazování obsahu adresářů bývá u souborů uváděna také jejich velikost. Často je to důležitá informace, protože např. posílat elektronickou poštou příliš rozsáhlá data znamená zahltit si připojení k Internetu, případně je občas nutno chápat fakt, že se soubor takové velikosti na disketu nevejde. Velikost souborů s daty je uváděna v počtech znaků, tedy v počtech bajtů. Při zkráceném výpisu se pak používají násobky počtu bajtů stupňované vždy o tři řády: kilobajt (zkráceně KB), megabajt (MB), gigabajt (GB) nebo terabajt (TB). Počet bajtů v 1 kilobajtu ale není 1000, nýbrž 1024. V Computer Science se totiž při uvádění délek souborů používá měřítko dvojkové soustavy, a to pro

snadnější zpracování dat v CPU. Při využívání kódování do nul a jedniček je výhodné používat délky, které jsou mocninami dvojky, tedy v řadě 1 (2^0), 2 (2^1), 4 (2^2), 8 (2^3), 16 (2^4), 32 (2^5), 64 (2^6), 128 (2^7), 256 (2^8), 512 (2^9), 1024 (2^{10}) atd. Odtud je převzata řádová terminologie, říkáme, že soubor má velikost 2KB obsahuje-li 2048 bajtů, 1MB je $2^{20}=1048576$ bajtů. Vypočtete si sami kolik bajtů má 1GB nebo 1TB.

V MS Windows je zvykem, že uživatel může vytvářet nové adresáře pro ukládání svých dat i libovolně v různých místech disku, dokonce i v oblasti instalace samotného operačního systému. V případě chráněných operačních systémů (např. UNIX) je každému uživateli vymezena disková oblast pro jeho data, která obvykle začíná ve stromové struktuře určitým adresářem. Typicky v LINUXu existuje za takovým účelem adresář `/home`, jehož podadresáře mají tento význam. Např. v adresáři `/home/mary` může uživatel, kterému je tento adresář přiřazen jako výchozí, svá data ukládat, může zde vytvářet podadresáře a dále tak košatit svůj podstrom adresářů, může adresáře a data případně i odstraňovat. Ostatní části celkového stromu adresářů tento uživatel pouze vidí, ale nemá k nim přístup pro přepis nebo ukládání nových dat. Je to důležité, protože odstranění adresáře Windows včetně celého jeho obsahu představuje velmi jednoduchý způsob, jak paralyzovat instalovaný operační systém MS Windows.

Adresář, jeho podadresáře a stromová struktura hierarchie adresářů je pouze způsob organizace diskových periférií. Data, programy, jádro operačního systému i kusy software jsou ukládány v souborech. I soubor má jméno, které je jeho základním atributem z pohledu jeho rozpoznání uživatelem. Jména souborů se objevují ve struktuře adresářů, říkáme, že soubory jsou uloženy v určitém adresáři. Pro jednoznačnou identifikaci bychom pak měli správně používat úplné jméno souboru (úplnou cestu, viz také předchozí část kapitoly), tj. před jeho jméno uvádět i jeho umístění ve stromové struktuře adresářů. Např. soubor se jménem *dvacátého* v adresáři *duben2003* (viz kresba I-15), který máme uložen na disketě v adresáři *matce*, bychom měli správně určit vypsáním `A:\dopisy\matce\duben2003\dvacátého`.

Pohled do obsahu souboru závisí vždy na typu obsahu tohoto souboru. Ať už je v souboru cokoliv, program, obrázek nebo dopis matce, jde vždy o sekvenci bajtů. Pouze význam těchto bajtů je pokaždé jiný. Jakým způsobem je s obsahem souboru zacházeno pak záleží na operačním systému nebo na aplikaci, která s daty pracuje. Často je zvykem pro jednodušší orientaci používat v rozlišení typů obsahů souboru příponu jména souboru, tj. část za poslední tečkou ve jménu souboru. Např. jméno `program.c` může být jméno souboru, v němž je uložen program napsaný v jazyce C, `dubnový.doc` může být jméno souboru s obsahem textu aplikace MS Word. Každá aplikace nebo dobrý operační systém ale ověřuje vnitřní strukturu (formát dat) obsahu takto označovaných souborů před jejich zpracováním (spuštěním aplikace) a v případě nesouladu jej odmítne zpracovat. Např. v operačních systémech MS Windows je přípona souboru vodítkem pro zobrazení odpovídající ikony souboru v průzkumníku,

v sofistikovanějších operačních systémech (UNIX, Mac OS) průzkumník před zobrazením odpovídající ikony nejprve prověří smysluplnost obsahu souboru.

Základní rozdělení obsahu souborů je na programy a data. Otvírání souborů s programy je věcí operačního systému a uživatel může programy startovat z nabídky jeho pracovní plochy (desktop). Soubory s daty lze otevírat pomocí průzkumníků. Je ale také možné po spuštění odpovídající aplikace v jejím prostředí vyhledat soubor s daty na disku a načíst je (jakýmsi malým průzkumníkem vloženým do aplikace). Nebo, začínáme-li pořizovat nová data, v aplikaci můžeme založit nový soubor obvykle pomocí nabídky z menu *Soubor->Nový*. Opět malým průzkumníkem aplikace určíme umístění nového souboru s daty a jeho jméno.

Obsahem souboru může být text

```
Mary had a little lamb;  
its fleece was white as snow
```

jehož digitální podobu jsme uvedli v úvodu kapitoly. Takto uložený text může být načten aplikací, kterou je některý z textových editorů (editovací program), např. v MS Windows je to Notepad (Poznámkový blok), v UNIXu je to např. editor *vi* (viz Příloha D). V uvedené digitální podobě takového textu ale nejsou obsaženy informace např. o tloušťce písma, jeho řezu (fontu) atd. Textový editor umožňuje také pouze doplňovat nebo měnit obyčejný digitalizovaný text ve významu jeho obsahu. Pokud použijeme pro editaci program, který požadavky na sazbu textu umožňuje, mezi textem jsou pak navíc uloženy informace o typografických značkách. Text tak přestává být jednoduše čitelný a pro jeho interpretaci je nutno používat prostředek, který typografickým značkám rozumí. Bohužel to dnes ale znamená číst a měnit text při další práci pouze pomocí programu, ve kterém jsme typografii textu dodali, protože standardní způsob vytváření takových značek není obecně stanoven. Editory takových dokumentů často nabízejí převodníky mezi svým formátem a formátem jiných editorů, obvykle ale se ztrátou některých informací.

V případě, že je obsahem souboru grafická informace, tj. např. obrázek beránka, soubor je rovněž sekvence bajtů v rozsahu ASCII, obrazová informace je ale kódována pro grafické programy. Principiálně se kódování grafických informací provádí buď rastrovým nebo vektorovým kódováním. Rastr (graphic raster), bitová mapa je evidence malých bodů (pixelů), které je možné zobrazit na displeji s jejich barevnými vlastnostmi. Vektorové kódování (vector graphics, coordinate graphics, slangem kódování do křivek) je promyšlený způsob záznamu obrazu, kdy je grafická informace uložena pomocí matematického aparátu, např. systému diferenciálních rovnic, obecně matematických příkazů jak zobrazit daná data. Každopádně kódování textu a kódování obrazu má jiný princip, jiný formát dat souboru. Proto jsou fotografie a filmy umísťovány do souborů odděleně od souborů s texty a uživatel musí takto principiálně k ukládání informací v počítačích přistupovat.

Dobrý editor ale dnes umožňuje kombinovat obraz i text. Jednou z cest jak toho dosáhnout je vyjadřovat i text jako obrázek, a nikoliv kódováním pomocí

ASCII doplněným o řez písma. Problém je ale v obtížném zpracování takto uloženého textu. Například vyhledat kus textu by znamenalo provádět poměrně složitou analýzu grafické informace. Proto současné výpočetní systémy používají kombinace textů a obrázků. Jako směsici textu a obrázků dnes ukládají data např. aplikace určené pro sazbu knih nebo aplikace pro práci s designem vůbec. Vnitřní formát takových souborů také rozlišuje začátek obrazové, textové nebo doplňující informace. Data jsou uložena v jednom větším souboru. Jiný způsob spojování grafických a textových dat se vyskytuje u hypertextů, což je dnes známý způsob používaný především pro prezentaci dat v Internetu. Je zde používán datový soubor s textem, který je proložen značkami způsobu zobrazení textu. Některé takové značky se nevztahují k textu, ale odkazují na soubory s grafickými informacemi. Datový textový soubor je ve formátu značkovacího jazyka (např. HTML, viz Příloha C). Prohlížeč nebo editor hypertextu čte tento výchozí textový soubor a podle značek zobrazuje uživateli jak text, tak odkazované obrázky, které načítá z jiných souborů.

O způsobu uložení obsahu souboru často také mluvíme jako o formátu souboru. Myslíme tím způsob ukládání informací. Soubor s pouhým textem nazýváme textový soubor (plain text). Grafické soubory mají formát např. JPEG, GIF nebo TIFF, případně BMP. Pro vyjádření jednoznačné sazby byl vytvořen formát grafického záznamu s označením PostScript a byl akceptován tiskárnami a osvitovými jednotkami různých výrobců právě pro zachování jednoznačného grafického vyjádření. V rámci editace se s ním ale nedalo rozumně pracovat. Byl proto používán jako finální způsob předávání sazby pro tisk do průmyslových tiskáren. V současné době je jeho pokračovatelem formát PDF. Více než pro vyjádření jednoznačného tisku se u PDF ale také jedná o jednoznačné grafické zobrazení v různých operačních systémech.

První digitalizace textů pracovala s abecedou anglického jazyka. Kód textu `little lamb je 0110110001100001011011010110001001101100011010010111010011010001101100`, který skládáním do osmic dokážeme (někdo i z hlavy) skutečně rozlišit. Ekvivalent v češtině je `beránek`. Pohledem do tabulky ASCII (viz Příloha A) máme ale s jeho digitální podobou problém, protože písmeno `á` v ní nenajdeme.

národní abecedy

V době expanze používání výpočetní techniky mimo anglosaský svět bylo nutno vytvářet digitální podoby textů psaných v národních abecedách, které ale nebyly v ASCII zcela obsaženy, protože některá písmena chyběla (např. uvedené `á`). Kódování tabulky ASCII využívá pouze 7 dolních bitů z bajtu (zprava doleva) a pokrývá tak 128 z celkových 256 možností. Použití osmého bitu otevírá cestu k využití horních 128 možností. Skutečnost, že tato část byla již využívána při např. interpretaci programů nebo ukládání obrázků, nebyla na překážku. My také již víme, že se jedná o jiné typy obsahu souborů, jejichž význam rozliší operační systém nebo aplikace (text zobrazí v textovém editoru, obrázek v grafickém editoru). V různých částech světa se proto začala horní část ASCII při záznamu textů přizpůsobovat potřebám národních abeced, které se jak známo vzájemně liší. Přestože v komunistickém Československu vznikla poměrně brzy místní norma s označením KOI-8, při

praktickém používání se ukázala jako nevyhovující. Její nevhodnost byla především v umístění jednotlivých písmen za sebou, což komplikovalo např. třídění textů. Proto se s nástupem vlny PC (stále ještě v komunismu) prosadil způsob kódování slovně označovaný jako Kód bratří Kamenických, který byl zaměřen právě tímto směrem. Bohužel vzájemně nekoordinovaně (tehdejší politické informační bariéry tomu vydatně napomáhaly) shodnou situaci řešili i v jiných zemích. Výsledkem byla vzájemná neslučitelnost např. českého a španělského textu, tedy, nebylo možné promístit český text španělskými citacemi tak, aby jej mohl číst svým editovacím programem autor ve Španělsku a spoluautor zase svým editorem v Československu. Situaci novodobé Babylónské věže se snažila vyřešit firma IBM, která nabídla kód, ve kterém vyjádřila všechny nejvíce používané znaky latinky tak, že bylo možné jej akceptovat na mezinárodní úrovni. Tento kód nesl označení LATIN2. Z něj se také odvinula definice standardu s označením ISO LATIN2, bohužel s jemnými (byť smysluplnými) rozdíly vzhledem k původnímu zdroji. Konečně s rozšířením grafických operačních systémů MS Windows firma Microsoft implementovala svůj vlastní kód pro různé národní abecedy, pro oblast Čech kód s označením Windows1250. Uživatelé získali možnost osobního výběru, který ale obvykle končil výběrem používaného editoru nebo spíše operačního systému. Tento výběr byl v posledních letech významně zúžen ve prospěch produktů firmy Microsoft, nejvíce používaným kódem je proto Windows1250. Pokud si ale uživatelé národních abeced vybrali české prostředí firmy Apple (počítače Macintosh), přijali opět jiný způsob kódování.

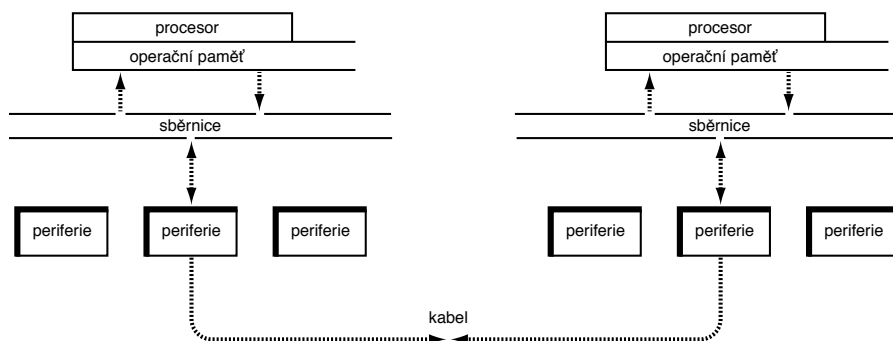
Dále se ukázalo, že využití horní části ASCII není obecné řešení. Jednak z důvodu, že počet znaků smysluplně seřazených pro všechny národní abecedy prostě nedostačuje, ale také proto, že některé používané abecedy jako je čínština nebo japonština se v ASCII vůbec vyjádřit nedají. Nijak geniální, ale přesto jediné rozumné řešení nabídlo Konsorcium pro UNICODE (viz Příloha A a [UNICODE]), které je vlastníkem normy pro vyjádření národních abeced. Přestože mezinárodní norma ISO (viz [ISOIEC]) je uváděna jako jiný dokument, jedná se o shodný způsob kódování. Princip je v rozšíření kódování z 1 na 2 bajty, tedy 16 bitů. Kódování je přitom provedeno velmi citlivě k dosavadním způsobům národních abeced (ovšem dle ISO), a to tak, že původní obsah bajtu se znakem zůstává tentýž, zařazení v rámci světa je dáno doplňujícím novým bajtem. Kombinací je přibližně 64 tisíc (víte kolik přesně?), což, jak se ukazuje, stačí. V průběhu postupné implementace normy UNICODE se začaly také používat její zjednodušené varianty. Jejich označení je např. UTF-8, UTF-16 atp. Využívají se např. v poštovních podsystémech, ale i v databázovém prostředí.

Bohužel, implementovat UNICODE do již provozovaného operačního systému v mnoha desítkách tisících instancích různě po světě je obtížné. Nejedná se totiž pouze o textové editory, ale uživatel má přece např. právo označit si jméno souboru ve svém jazyce a obsah adresářů s takovými jmény zobrazovat seřazený podle pořadí písmen své abecedy. A pokud uživatel současně mluví více jazyky, má právo zajistit takové prostředí pro každý z nich, a to současně. Implementace UNICODE při zachování zpětné kompatibility

je proto vleklá, přestože je již v operačních systémech patrná. Syndrom Babylónské věže však zůstává a uživatel se v mezinárodní komunikaci musí připravit na problémy a nutnost mnohdy použít převodní programy nebo složité nastavování aplikací na správný kód používaný v dané lokalitě.

II.1 – propojení počítačů, multiplexing, vrstvení, technologie internet, DNS

Záznamy na hliněných tabulkách sloužily k uchovávání informací, a sloužily nejenom jejich pisatelům, ale i těm, kteří byli schopni a ochotni informace číst. Záznamová forma komunikace mezi lidmi přinášela dále navíc archivní aspekt. Informace bylo možné číst i po uplynutí času, a to jak pro připomenutí původního významu, tak pro nový kontext např. za několik tisíc let. Snaha vzájemného sdělení je pro lidské jedince významná. Podobně i v počítačové době sdělit, tedy poskytnout výsledky výpočtu někomu jinému, bylo možné prostřednictvím papíru, na který byla data vytištěna, ta však byla už dále obtížně použitelná pro další výpočet jiným člověkem na jiném počítači. Kromě možnosti přenášet data od počítače k počítači na výměnných discích se jako daleko výhodnější způsob nabízelo přímé spojení (connection, connect, konektivita) počítačů. Přímým spojením rozumíme využití specializované periferie v každém z připojovaných počítačů tak, že na jedné straně je periferie součástí stroje a na straně druhé komunikuje se svým protějškem v jiném počítači. Specializované periferie jsou mezi sebou propojeny kabelem, tak aby vzájemná výměna dat mezi nimi byla co nejrychlejší. Podívejme se na kresbu 25.



Kresba II-1: Periferie spojující počítače.

Ve smyslu Von Neumannova schématu jsou počítače obohaceny o periferii, která neslouží pro místní manipulaci s daty, ale dokáže data předat prostřednictvím kabelu periférii téhož typu, která je vložena do jiného počítače. Tyto specializované periferie musí být začleněny do celkové funkce počítače, a to jak do hardware tak do software. Znamená to, že na pokyn uživatele jsou určitá data v operační paměti stroje předána periférii pro přenos do jiného počítače. Na protější straně je nutné, aby na příchod dat periferie reagovala jejich převzetím a prostřednictvím podsystému přerušování na jejich příchod upozornila. Toto upozornění registruje operační systém, který data přesune z periferie do operační paměti, a podle požadavku počítače, který data poslal, je zpracuje (např. uloží na diskovou periferii).

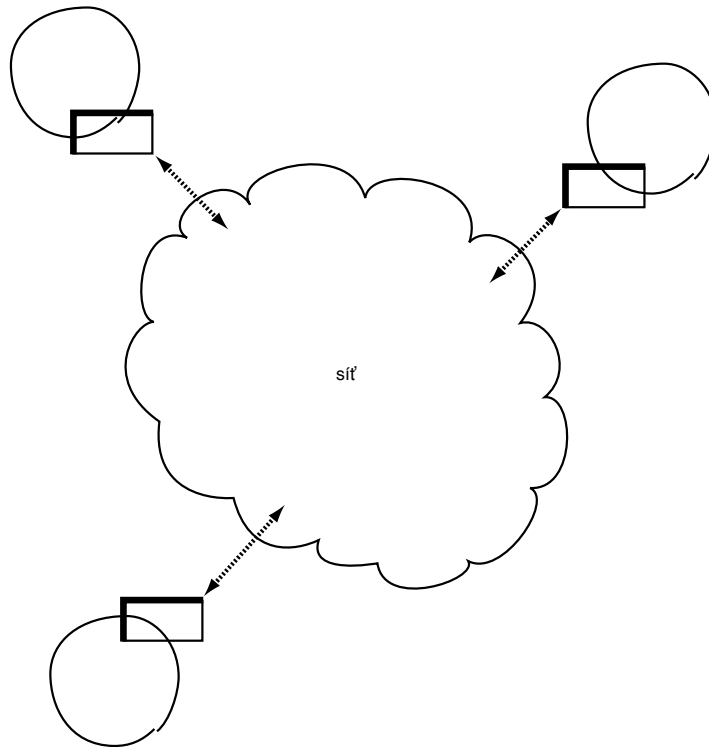
Uvedený způsob komunikace musí pro praktické potřeby zajišťovat přenos dat, který je oboustranný a pokud možno velmi rychlý. V optimálním případě

tak rychlý, aby rychlost přenosu dat byla srovnatelná s přenosem dat mezi např. diskovou periferií a CPU.

Pro účely obecné komunikace je ale potřeba spojovat mezi sebou více než dva počítače. Princip spojování by měl obecně zajistit propojení libovolného počtu počítačů, vzájemně mezi sebou a současně. Teprve pak vznikají sítě počítačů. Periferie, které se v těchto technologiích používají, nazýváme síťové periferie.

rámce (frames)

Hardware počítačových sítí je proto koncipován tak, že síťová periferie komunikuje se svými protějšky způsobem odesílání a přijímání určitých kousků dat, kterým říkáme *rámce* (frames). Vzhledem k tomu, že na jednom počítači může být současně několik požadavků na síťovou komunikaci (více současně pracujících uživatelů, tentýž uživatel současně prohlíží Internet a tiskne na tiskárně ve vedlejší místnosti), jednotlivé rámce mohou být určeny různým protějškům síťové periferie. Úkolem síťové periferie je odesílat zapisované rámce do sítě, označovat je jejich adresátem a opačně z rámců právě procházejících sítí přijímat ty, které jsou jí adresovány. Podívejme se na kresbu II-2.



Kresba II-2: Síťové periferie.

most (bridge)

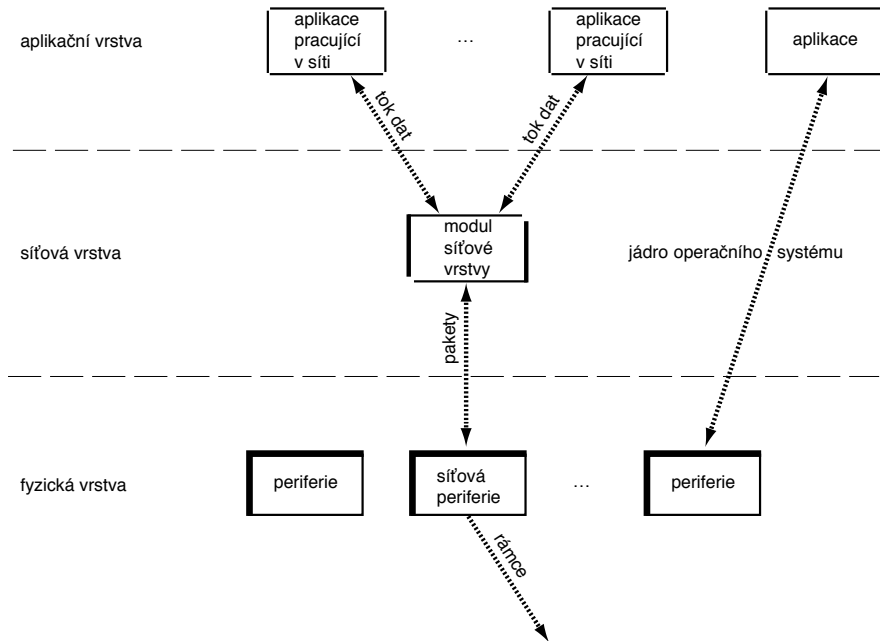
Obláčkem schematizujeme síť. Zde se jedná o hardware síť. Přesněji je každá síťová periferie každého počítače připojeného k síti součástí tohoto hardware sítě. Obláčkem zde ale rozumíme zajištění komunikace tak, aby

síťové periferie měly možnost odesílat rámce všem svým protějškům v síti a přijímat je. Toto zajištění komunikace přitom může být poměrně složité. Základem ale je několik aktivních prvků, které jsou umístěny na cestě mezi síťovými periferiemi, komunikují s nimi a jejich rámce vzájemně distribuují. Příkladem může být hvězdicové spojení všech síťových periferií do jednoho centra. Centrum je aktivní síťový prvek, kterému říkáme opakovač (repeater), např. je to hub nebo switch, který je vlastně násobným přepínačem síťových periferií. Taková topologie hardware sítě vynucuje ale koncepci síťových periferií téhož typu, např. s označením Ethernet. V případě, že v některém počítači je síťová periferie typu jiného, je potřeba připojení tohoto počítače k ostatním podpojit síťovým prvkem, který zajistí dorozumění síťových periferií nebo opakovačů takto různého typu. Takovému aktivnímu síťovému prvku říkáme most (bridge). Most je tedy síťový aktivní prvek, hardware, který umožní dorozumění síťových periferií na různých principech hardwarového spojení.

**opakovač
(repeater)**

Z výrazu adresace rámců, který jsme použili, je zřejmé, že síťové periferie musí být jednoznačně identifikovány. Je tomu skutečně tak, každý výrobce hardware síťových periferií vyrábí každou novou periferii s označením unikátním na celém světě. Adresa periferie je číslo poměrně velkého rozsahu a každý výrobce má přidělenou určitou část tohoto rozsahu. Shoda adres např. periferií typu Ethernet je kolizní. Důsledkem stejných adres v téže síti (až po most) by bylo nefungující spojení počítačů, ve kterých jsou periferie použity.

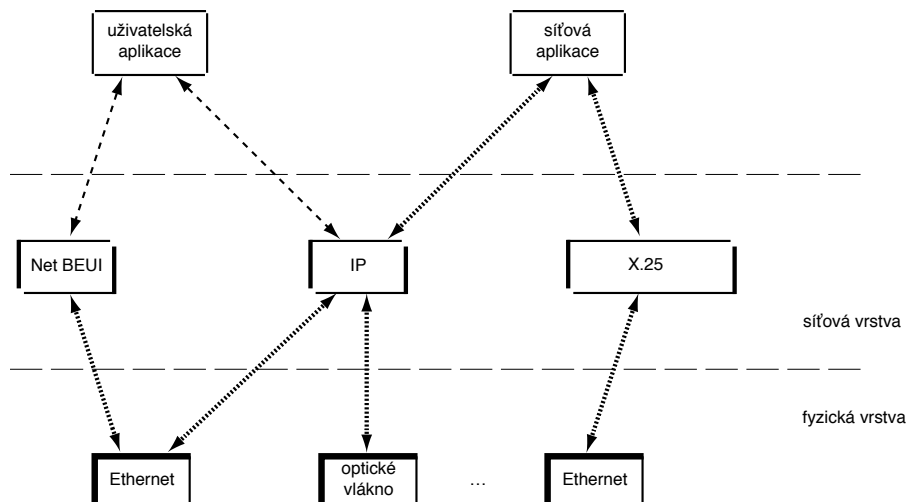
Síťová periferie v počítači je ovládána pomocí software, který je součástí operačního systému. Obvykle se jedná o moduly jádra, které řídí předávání dat od uživatelů k síťové periferii a naopak. Znamená to, že je-li určitý obsah souboru zapisován do sítě (po jednotlivých rámcích), tedy potažmo na diskovou periferii vzdáleného počítače, zajišťuje takto operační systém sekvenční tok dat, v něm jsou obsažena data požadovaného souboru. Tento tok dat musí být duplexní, tj. oboustranný a dále musí operační systém zajistit více takových toků dat současně (tak, aby se různá data nesmíchala, že ano). V operačních systémech je proto síťové spojení budováno na základě programové vrstvy, která komunikuje s vrstvou hardware. Princip je na kresbě II-3.



Kresba II-3: Vrstva fyzická, síťová, aplikační.

pakety (packets)**vrstvy (layers)**

Thick line separates hardware and software. Part of the operating system is a program module with the label network layer module. This module ensures data transfer between network peripherals and applications, which use the network for communication with other computers. The network layer module also controls how data is written to the network, so that data from incoming networks, which belongs to the responding application, and that in the context of a unique network connection between two applications on different computers (the introduced scheme has its counterparts on other computers in the network). When writing data to the network, the network layer module takes data from applications, packages it into *packet* (packets), i.e. again pieces of data, which are then sent to hardware, which further packages the pieces into frames and sends them to the network. In drawing II-3, there are also names of layers, i.e. logically separated functional units, which pass data to the network successively from hardware to applications. We denote them as *physical layer* (physical layer), *network layer* (network layer) and *application layer* (application layer). Such a principle allows using e.g. different types of network connections simultaneously through one network peripheral. E.g. on PC in operating systems MS Windows, there are commonly installed modules of two types of networks. NetBEUI is a network layer module, which the company Microsoft realized as a network PC originally, IP is a network layer module, which later became a standard of the Internet and without it the user of the PC could not use the Internet. The application addresses the responding network layer module according to the type of connection, or possibly the network layer module determines the operating system (on the basis of any possible connection). Also, each network layer module can work with several network peripherals of different types. Let's look at drawing II-4.

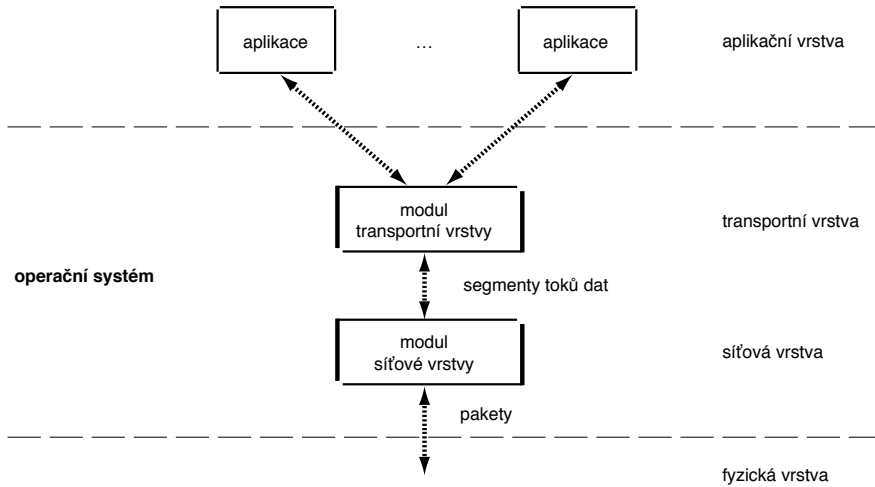


Kresba II-4: Význam vrstev.

Moduly síťové vrstvy mohou komunikovat s různými typy spojení na úrovni hardware, aplikace přitom sama určuje výběr způsobu průchodu dat sítí (tedy skladbu vrstev). Z kresby a z toho, co bylo řečeno, také vyplývá, že síťová periférie jednoho typu přijímá pakety různých typů síťové vrstvy. Tedy dvě různé aplikace mohou využívat tutéž síťovou periférii (např. Ethernet) pro např. přístup k tiskárně na vedlejším počítači (pomocí NetBEUI) a současně pro přístup ke stránkám WWW v Internetu (pomocí IP).

Příklad fyzické vrstvy na kresbě II-4 je sice možný (a vyskytující se), ale netypický. Přítomnost síťových periférií různého typu je nutné zajistit při použití počítače jako zvláštního propojovacího uzlu mezi sítěmi různých typů na úrovni hardware, tedy zajistit funkci mostu pomocí software (na kresbě je tento software označen jako síťová aplikace). Naopak přítomnost několika různých typů modulů síťové vrstvy pro práci v různých typech sítí (obvykle nabízené různými výrobci operačních systémů) je dnes běžná. Uživatelská aplikace na kresbě využije jeden z možných síťových modulů, ne však oba současně.

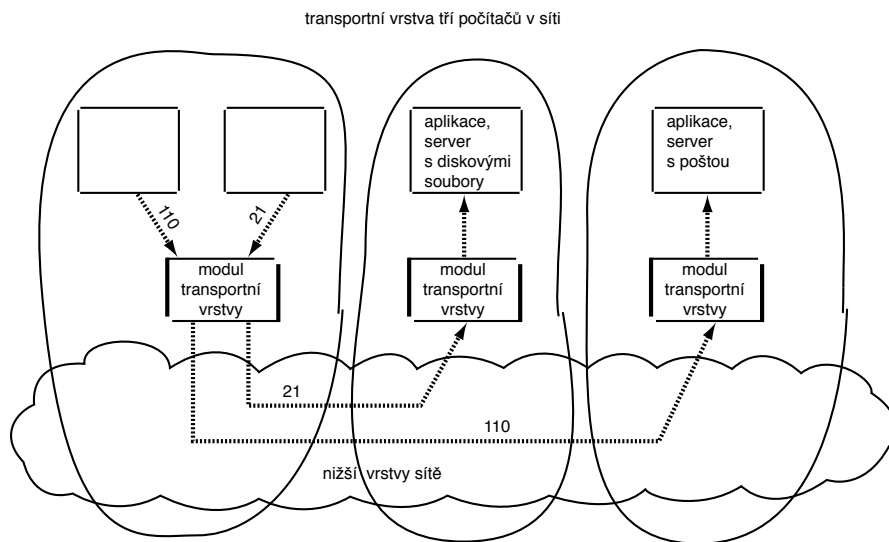
Při dalším přiblížení ovšem zjistíme, že vrstvení v sítích je jemnější. Mezi síťovou a aplikační vrstvou je umisťována ještě vrstva transportní (transport). Její základní funkcí je vytváření souvislých datových toků síťových spojení. Síťová vrstva totiž zajišťuje nalezení a komunikaci počítačů v síti a předáváníí paketů. Skládání paketů do souvislého duplexního toku dat vzájemně propojených aplikací ale zajišťuje vrstva transportní. Situace je uvedena na kresbě II-5.



Kresba II-5: Transportní vrstva.

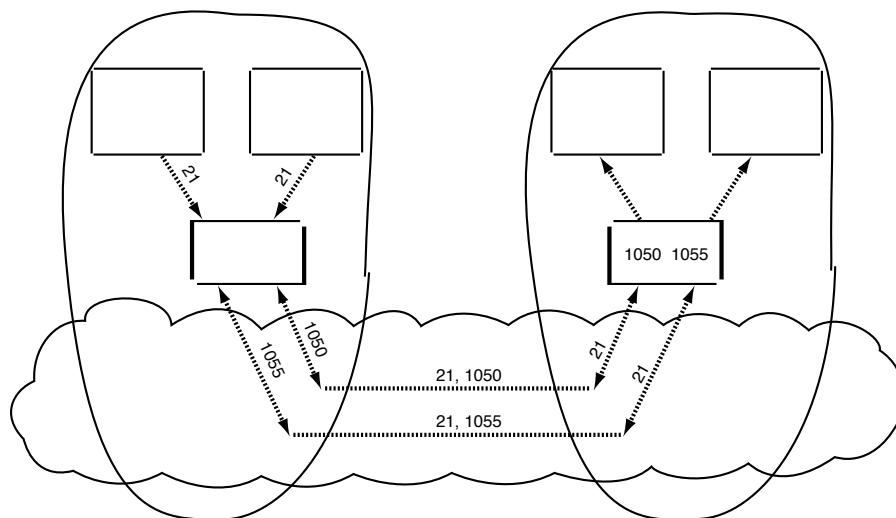
port transportní vrstvy

Příkladem transportní vrstvy je v Internetu používaný modul TCP (Transmission Control Protocol). Modul transportní vrstvy na kresbě zajišťuje současnou práci dvou aplikací v síti. TCP v této situaci odlišuje aplikace prostřednictvím portů. Číslo portu (port number) je celé číslo od 1 do několika tisíc, které je domluvené, tedy obecně stanovené. Znamená to, že pokud např. jedna z aplikací bude číst novou poštu ze serveru (v žargonu stahovat poštu), ohlásí to transportní vrstvě požadavkem na přidělení spojení s číselným označením 110. Druhá aplikace bude např. přenášet soubory s daty z diskového serveru, ohlásí to transportní vrstvě požadavkem na port s číslem 21. S číslem portu navazuje transportní vrstva spojení na vyžádaném protějšším počítači. Transportní vrstva protějšku dohodí spojení s odpovídající aplikací právě na základě např. čísla 110 tak, že se probudí aplikace, která poštu uživateli poskytne. Podrobněji je tento příklad znázorněn na kresbě II-6.



Kresba II-6: Výchozí čísla portů.

Pomocí čísel portů jsou tedy označovány toky dat, které transportní vrstva zajišťuje. Výchozí číslo portu, které aplikace používá pro označení typu svého toku dat, ale transportní vrstvě nestačí k tomu, aby tok dat jednoznačně identifikovala. Je totiž legitimní, aby transportní vrstva zajistila např. přenos souborů s daty pro několik uživatelů současně, a to i uživatelů, kteří jsou přihlášení na tomtéž počítači a přenášejí data z téhož protějšku. Kresba II-7 ukazuje příklad, kdy dvě různé aplikace vyžadují tok dat téhož typu (číslem portu 21), protějškem je přitom tentýž počítač.

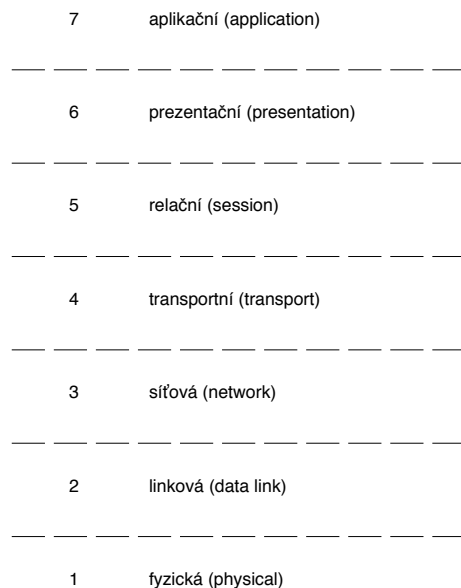


Kresba II-7: Dva toky dat téhož typu.

Transportní vrstva protějšku registruje tok dat typu 21 a odpovídá přiřazením opačného portu s číslem např. 1050 (dočasné číslo portu, ephemeral port number). Příchod dalšího požadavku na tentýž typ toku registruje přiřazením dalšího dočasného čísla portu, zde 1055. Tato čísla portu oznamuje zpět výchozímu počítači. Oba toky dat jsou takto identifikovány dvojicí čísel, která je pro každé spojení jiná, byť stejného typu. Po ukončení spojení jsou dočasná čísla portu uvolněna a transportní vrstva je použije při dalších požadavcích na síťový tok dat odkudkoliv ze sítě a pro třeba i jiný typ toku dat. Síťový tok dat je tedy vždy jednoznačně určen dvojicí čísel portů mezi dvěma počítači v síti.

segment Kromě zajištění identifikace toků dat aplikací má transportní vrstva např. také na starosti rozklad toku dat na *segmenty* (odcházející), které předává síťové vrstvě a naopak skládání (příchozích) segmentů téhož toku dat v souvislý celek pro zajištění duplexního spojení. Transportní vrstva může předávat síťové vrstvě najednou několik segmentů dat téhož toku současně, takže některý segment může předběhnout v pořadí jiné segmenty, transportní vrstva je ale musí složit ve správném pořadí (segmenty jsou číslovány). Takové předbíhání je výhodné v případě, že se rozhodneme stornovat již probíhající tok dat. Je totiž nepříjemné, stiskneme-li tlačítko storno a někdy třeba i několik vteřin ještě stále probíhá přenos dat, který nás již nezajímá. Dobrá aplikace využije příznaku vysoké urgency u segmentu a dokáže tak již odeslaná data do sítě předběhnout a upozornit protějščí stranu o stornování komunikace, jejíž data se ještě pohybují sítí.

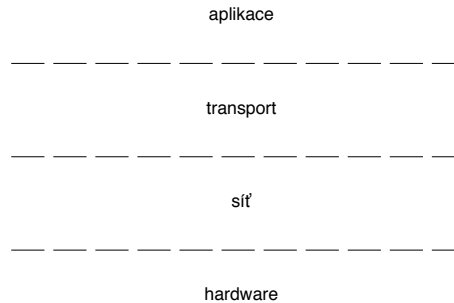
**standard
síťových vrstev** Princip vrstvení se v technologiích počítačových sítí ukázal jako výhodný pro integraci různých síťových principů již velmi dávno. Vrstvy přinesla snaha začlenit do sítí jak různý síťový hardware, tak software různých operačních systémů. Na konci 60. let 20. století, kdy se objevily první jasné signály o nutnosti počítače vzájemně propojovat, bylo v USA velké množství prakticky používaných výpočetních celků, které uživatele podporovaly v práci operačními systémy implementovanými různým způsobem. Manipulace s hardware byla mnohdy velmi rozdílná. Úvahy o struktuře počítačových sítí tak vedly k zajištění propojení heterogenních výpočetních systémů. Přestože praktické realizace do konce 70. let rychle předbíhaly teorii a laboratorní výzkumy, IEEE (Institute of Electrical and Electronics Engineers) definovala obecně pak respektovaný a z velké části dodržovaný model vrstvení sítí s označením OSI Reference Model (Open Systems Interconnection) registrovaný teprve v roce 1984 (viz [ISOOSI1984]). Standardizovaný model má celkem 7 vrstev, jak ukazuje kresba II-8.



Kresba II-8: 7 vrstev podle ISO OSI.

Kromě nám již známých vrstev, včetně jejich provozního významu, je normou vyžadováno používání vrstvy *linkové* nad vrstvou fyzickou, tedy zajištění obecné komunikace s hardware daného typu. Mezi samotný program, který komunikuje prostřednictvím sítě (síťovou aplikaci), a vrstvu transportní jsou vkládány ještě vrstvy *relační* a *prezentační*. Jedná se především o podporu programování v sítích a zajištění případné konverze různých interpretací dat (např. pořadím bajtů) mezi různými operačními systémy. Typickým síťovým programovacím jazykem je např. historicky známý RPC (Remote Procedure Calls). Uživatelé však definice prostředí programátorovy manipulace obvykle nezajímá, zvláště když pro samotný provoz tato vrstva není nijak patrná.

Implementace síťových vrstev podle OSI je mnohdy zužována právě ve smyslu jasného určení jednotlivých vrstev. V praxi se mnohdy setkáme s určením pouze 4 vrstev, jak jsme popsali síť na předchozích několika stránkách i v tomto textu. Také známá definice požadované síťové komunikace podle Ministerstva obrany Spojených států (DoD – Department of Defense) se zaměřuje pouze na tyto 4 vrstvy (viz kresba II-9) a my si s takovým zúžením v dalším textu rovněž vystačíme.



Kresba II-9: 4 vrstvy podle DoD.

Počítačová síť je tedy koncipována na principu komunikace dvou programů, síťových aplikací, které vzájemnou výměnou dat zajistí uživateli požadovanou službu, jako např. přenesení pošty, kopii dat z disku jednoho počítače na druhý nebo tisk dokumentu na tiskárně připojené k jinému počítači. Pro vzájemnou komunikaci programy využívají síťové duplexní kanály, které na jejich požádání vytvoří a dá jim k dispozici transportní vrstvu. Transportní vrstva k vyhledání odpovídajícího počítače a vytvoření spojení, ať již ve stejném nebo jiném typu operačního systému, použije vrstvu síťovou. Síťová vrstva oslovuje hardware, který data přenáší tam, kam je určeno. Toky dat jsou přitom realizovány pomocí jejich rozdělení na různých úrovních na segmenty, pakety a rámce. Dosáhne se tak řízení teoreticky libovolného počtu toků dat obecně mezi libovolným počtem počítačů (tzv. multiplexing).

technologie internet, IP

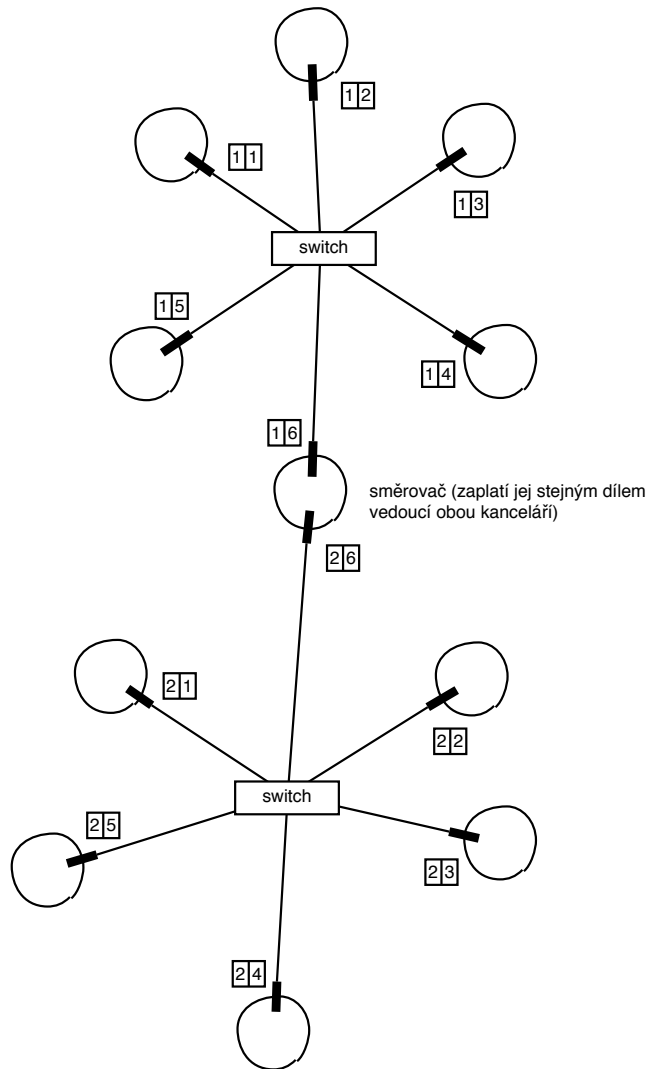
Historicky se současná celosvětová síť ustálila na principu síťové vrstvy s označením IP (Internet Protocol). Adjektivum internet v anglickém názvu bylo použito jako zkratka původního internetworking ve smyslu propojování sítí různého typu mezi sebou. Dnes používáme termín internet (s malým i) jako označení technologie, Internet (s velkým I) pak pro označení celosvětové sítě na bázi této technologie. Definice principů síťové vrstvy typu IP sahá až do druhé poloviny 60. let 20. století. V rámci projektu DARPA (Defense ARPA, projekt, na kterém se podílel opět zájem DoD) byl definován způsob propojování počítačových sítí různých výrobců a různých typů, které tehdy ve Spojených státech v praxi již existovaly a byly provozovány. Vložení programového modulu takto definovaného typu IP do operačního systému mělo umožnit výrobcům různých operačních systémů a sítí vzájemnou propojitelnost (interconnection). Požadavky, které se podařilo realizovat a které znamenaly rychlé následné rozšíření IP do veřejných sítí a vznik celosvětové sítě s označením Internet, byly zejména následující:

- síť počítačů neovlivní výpadek jednoho z jejich počítačů,
- sítě vzájemně mezi sebou propojené neovlivní výpadek jedné z těchto sítí,
- sítě vzájemně mezi sebou mohou mít obecně libovolný počet vzájemných propojení a cestu k protějšku lze dynamicky měnit,
- připojování a odpojování počítače nebo celé sítě neovlivní ostatní sítě a počítače,
- počítače a sítě mohou měnit libovolně své fyzické umístění.

Princip IP vychází z definice označování počítačů a sítí pomocí adresy IP (IP Address). Jedná se o unikátní číslování každého počítače, a to tak, že je v tomto čísle obsaženo také jejich sdružování do jednotlivých sítí. Uvedme si příklad. Mám vyřešit propojení několika počítačů v mé kanceláři vzájemně mezi sebou. Je-li jich celkem 5, nabízí se označit je postupně čísly 1, 2, 3, 4 a 5. Totéž zjevně ale napadne i kolegu ve vedlejší kanceláři téže firmy. Naší oddělené realizaci to ovšem v ničem nebrání a počítače si takto číselně označené můžeme propojit a používat je při vzájemné výměně dat. Ovšem pouze do okamžiku, kdy se dále rozhodneme propojit vzájemně i počítače obou kanceláří. Jisté je, že je budeme muset přečíslovat, a to nejméně v jedné z kanceláří. Vzhledem k tomu, že je pravděpodobné, že kanceláře budou mezi sebou komunikovat méně než účastníci jednotlivých kanceláří, bylo by výhodnější sítě mezi sebou propojit pouze v jednom místě a přenášet mezi nimi jen data požadavků vzájemné mezikancelářové (internetworking) komunikace (dvou kancelářských sítí). V rámci adresace IP je v jednoznačném očíslování každého počítače zahrnuto také označení sítě, do které počítač patří. V takovém případě může očíslování v kancelářích zůstat totéž, odlišíme je ale v části označení sítě.

adresa IP

Skutečně pak lze obě již existující sítě spojit a definovat jejich propojení pomocí specializovaného počítače s označením směrovač (router). Tento počítač má namontovány dvě síťové periferie, každá z nich je přitom obrácena do jedné ze sítí ve dvou kancelářích. Vytváření toků dat mezi počítači je pak v každé kanceláři místní, tedy bez jakéhokoli zatížení sítě ve druhé kanceláři. Pouze dojde-li k požadavku toku dat mezi počítači v různých kancelářích, vyřizuje takové propojení směrovač, přes který tok dat pak prochází. Situaci máme na kresbě II-10.



Kresba II-10: Počítače ve dvou sítích typu IP.

Jednotlivé sítě mají každá svůj fyzický switch, na oba je připojen směrovač. Každá síťová periferie na obrázku je vybavena značkou s dvojím očíslováním. Zleva doprava je to nejprve číslo sítě a dále číslo počítače v této síti. Určení počítačů je tak jednoznačné. Co víc, jsme připraveni do směrovače vložit další síťovou periferii a připojit síť třetí kanceláře. Nebo můžeme třetí síťovou periferii využít pro připojení k jiným síťovým celkům. To vše za předpokladu, že nebudeme mít kolizní čísla sítě, což se ale může stát, připojíme-li se např. k Internetu, kde jsou čísla sítě již přidělena vzájemnou domluvou po celém světě. Pokud budeme ale předem plánovat číslování sítě vzhledem ke zbytku světa, připojení nebude vyžadovat další úpravy. Pokud budeme

takto napojení na celý svět a jedna z kanceláří zvedne kotvy a odpluje do jiného města v jiném státě, může použít identifikaci své sítě pro připojení na jiný směrovač v jiném přístavu této planety. Po připojení se pak ve virtuálním světě sítě počítačů objeví zdroje této sítě tak, jak je znal zbytek světa doposud, přestože se fyzicky přemístily jinač.

Poskytovatelé (providers) Internetu přidělují čísla sítí. Číslo sítě pak místní správci sítí doplňují identifikací jednotlivých počítačů své sítě. Adresa IP má rozsah 32 bitů. Čteme ji po 8 bitech, jedná se tedy o 4 čísla. Čísla oddělujeme tečkou. Např. 147.251.0.0 je číslo sítě, 147.251.34.10 je počítač v této síti (přesněji 34.10). Uvedli jsme adresu sítě, která je kódována do dvou osmic, zbylé dvě zůstávají pro počítače. Nemusí tomu ale tak být. Číslem sítě vždy adresa IP začíná. Podle toho, zda je síť kódována v jedné nebo více osmic zleva doprava říkáme, že používáme adresu IP typu A (jedna osmice), B (dvě osmice), C (tři). V příkladu uvedená adresa je typu B. Pro zájemce je rozsah jednotlivých částí adresace IP s dalšími podrobnostmi uveden v Příloze B.

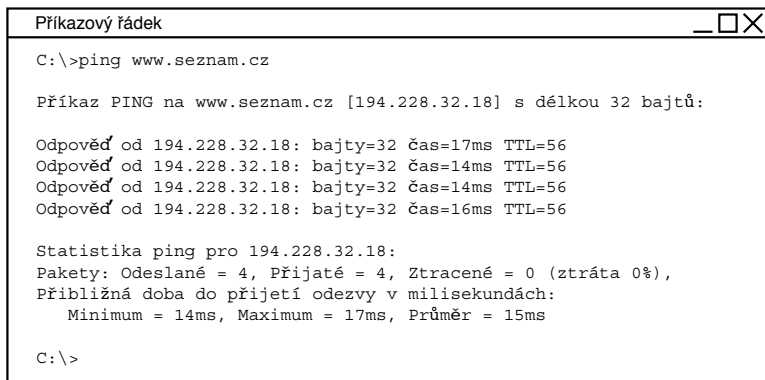
Modul IP v operačním systému musí znát adresu IP síťové periferie. Tu mu určí správce sítě v odpovídající tabulce správy operačního systému. Principiálně nic jiného modul IP nepotřebuje, protože z této adresy ví, ve které síti se nachází, ví, že v rámci ostatních počítačů v této síti jim může posílat pakety toků dat. Pokud ale bude potřebovat podpořit tok dat s aplikací běžící na počítači mimo jeho síťovou adresu, je bezradný. Proto musí mít ještě stanoveno místo v síti, kudy je síť připojena k jiným sítím, tedy adresu IP směrovače. Spojení směrovaná mimo jeho síťovou adresu pak posílá na tento uzel a očekává, že spojení, tedy cestu toku dat, vyřeší někdo za něj, a sice směrovač.

Jisté je, že skutečná legrace nastává na směrovačích. Ty totiž musí kvalifikovaně zajistit předávání paketů mezi sítěmi, které znají. Ty, které neznají, oslovují tak, že odesílají pakety na jednu z adres, kde pro ně svět sítě IP pokračuje. Navíc, takové předávání dat může probíhat různými cestami, které jsou dány poskládáním sítí a směrovačů mezi sebou. Lze tak dynamicky vypočítávat v daném okamžiku nejvhodnější průchod sítí a pro různé pakety je navíc měnit. Krásná hračka!

Zdali Internet pracuje na bázi takových adres, lze zhlédnout a následně i vyzkoušet při využívání např. prohlížeče stránek WWW. V prohlížeči (Mozilla Firefox, MS Internet Explorer atp.) do místa, kde běžně zadáváme textovou adresu některého informačního serveru např. www.seznam.cz, můžeme namísto symbolického jména napsat přímo adresu IP počítače, na kterém jsou stránky WWW uloženy. Skutečně totiž dochází k doposud popsané situaci, kdy náš prohlížeč stránek prostřednictvím síťových vrstev kontaktuje počítač s požadovanými informacemi a ten mu v navázaném spojení mu postupně pošle stránky WWW. Prohlížeč je uživateli zobrazí v okně. Obvykle u dolního levého okraje okna prohlížeče se objevují doplňující informace o probíhajícím síťovém přenosu dat. Tam lze také zhlédnout text o kontaktování počítače s určitou adresou ve tvaru IP. Znamená to, že prohlížeč si naše zadání informačního serveru převádí do tvaru číselného označení odpovídajícího

počítače se stránkami WWW. Pokud bychom zachytili toto číslo a opsali je do místa, kam jsme dříve zapsali `www.seznam.cz`, prohlížeč adresu IP bez mrknutí oka akceptuje a provede tutéž práci.

Adresu IP určitého počítače v síti Internet podle umístění stránek WWW můžeme získat prakticky na každém počítači. Např. v operačních systémech MS Windows můžeme otevřít okno Příkazového řádku (tj. MS DOS prompt, Command line) a za text `C:\>` můžeme psát text `ping www.seznam.cz` dle kresby II-11.



```
Příkazový řádek

C:\>ping www.seznam.cz

Příkaz PING na www.seznam.cz [194.228.32.18] s délkou 32 bajtů:

Odpověď od 194.228.32.18: bajty=32 čas=17ms TTL=56
Odpověď od 194.228.32.18: bajty=32 čas=14ms TTL=56
Odpověď od 194.228.32.18: bajty=32 čas=14ms TTL=56
Odpověď od 194.228.32.18: bajty=32 čas=16ms TTL=56

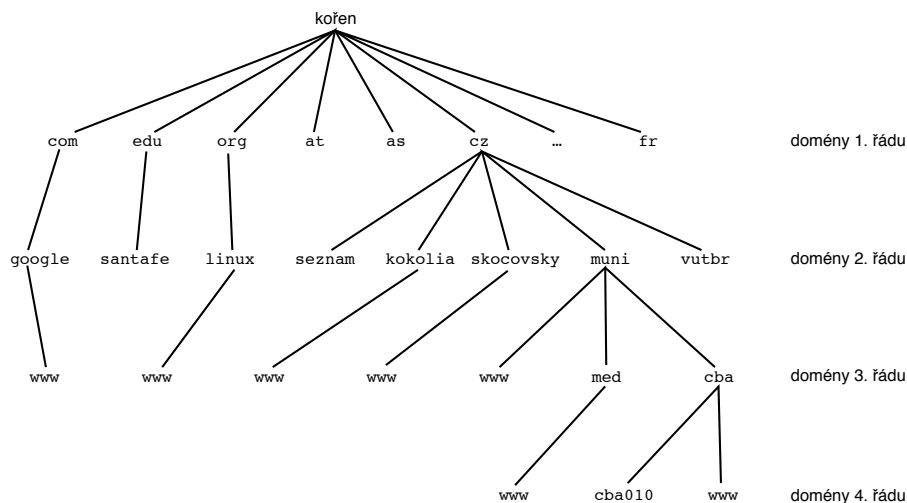
Statistika ping pro 194.228.32.18:
Pakety: Odeslané = 4, Přijaté = 4, Ztracené = 0 (ztráta 0%),
Přibližná doba do přijetí odezvy v milisekundách:
    Minimum = 14ms, Maximum = 17ms, Průměr = 15ms

C:\>
```

Kresba II-11: Program `ping` v MS Windows.

ping `ping` (Packet InterNet Groper) je program prověřování dostupnosti počítačů v síti typu IP. Při nalezení cesty k označenému počítači (zde `www.seznam.cz`) vypisuje informace o rychlosti průchodu k nalezenému počítači tak, že si s ním vymění několik paketů na úrovni modulů IP a čas této výměny uvede v textu (zde postupně 15ms, 13ms). Jeho první aktivitou je ale převod jména počítače na jeho adresu IP, výsledek ihned vypíše do okna (zde `212.80.76.3`).

doména Každá síťová aplikace (`ping` nebo Mozilla Firefox) tedy využívá pro samotné síťové spojení adresu IP a pokud ji uživatel zná, akceptuje ji aplikace v zadání. Obvykle uživatelé ale adresy IP neznají a věříme, že je ani znát nechtějí. Jak aplikace zjišťuje adresu IP na základě jmenného určení? Co to vlastně je jmenné určení počítače v síti? Symbolické pojmenování počítačů v síti typu internet vychází ze stromové struktury přiřazování domén v hierarchickém systému. Podívejme se na kresbu II-12.



Kresba II-12: Hierarchie domén Internetu.

V rámci aplikací, které síť typu IP nabídla v průběhu svého vývoje, vznikla také síťová aplikace s označením Domain Name System, zkráceně DNS. Programy této aplikace jsou instalovány roztroušeně (distribuovaně) na různých nepřetržitě běžících počítačích v propojených sítích typu IP tak, že si vzájemně vyměňují informace. Každý instalovaný program DNS poskytuje informace o jménech počítačů v jemu vymezených částech sítí a v případě, že si neví rady, má určeno, který jeho ekvivalent mu poradí. Informační databanka roztroušená po všech vzájemně propojených sítích (potažmo po celém Internetu naší planety) celkově vyjadřuje hierarchickou strukturu stromu domén, jak vidíme na kresbě. Doména je typ zařazení jména do určité hierarchické úrovně. Např. doména prvního řádu je jistě `com` nebo `org`, ale i `fr` nebo `cz`. Doména druhého řádu je např. `google` nebo `kokolia`, čtvrtého řádu `cba010`. Nejlépe se řád každé domény vyloupne ze zapsaného jména identifikace určité oblasti. Např. v označení `www.google.com` je `www` jméno domény třetího řádu, `google` druhého a `com` prvního řádu. Striktně vzato existuje i doména nultého řádu, je pouze jedna, neuvádí se a na kresbě je označena jako kořen (root). Domény jednotlivých řádů jsou přitom ve správě různých programů síťové aplikace DNS tak, že např. kořen DNS v Internetu zajišťuje jeden běžící program, kde jsou převážně evidovány odkazy na domény prvního řádu, tj. jaké adresy IP mají počítače, na kterých běží programy DNS, které mají na starosti podstrom daný touto doménou. Počítače s běžícím programem pro doménu prvního řádu (např. `cz`, že) obvykle evidují odkazy na domény druhého řádu, tj. např. `kokolia.cz` nebo `skocovsky.cz` jsou umístěny obě na jednom počítači (to vím) s adresou IP, kterou registruje centrální počítač s doménou `cz`. Nic ale nebrání (a běžně se používá) registraci domén nižšího řádu na téže adrese IP neboli na totéž počítači. Např. by mohla doména se jménem `www.med.muni.cz` být bez problémů registrována na počítači s centrálním uzlem domény prvního

DNS

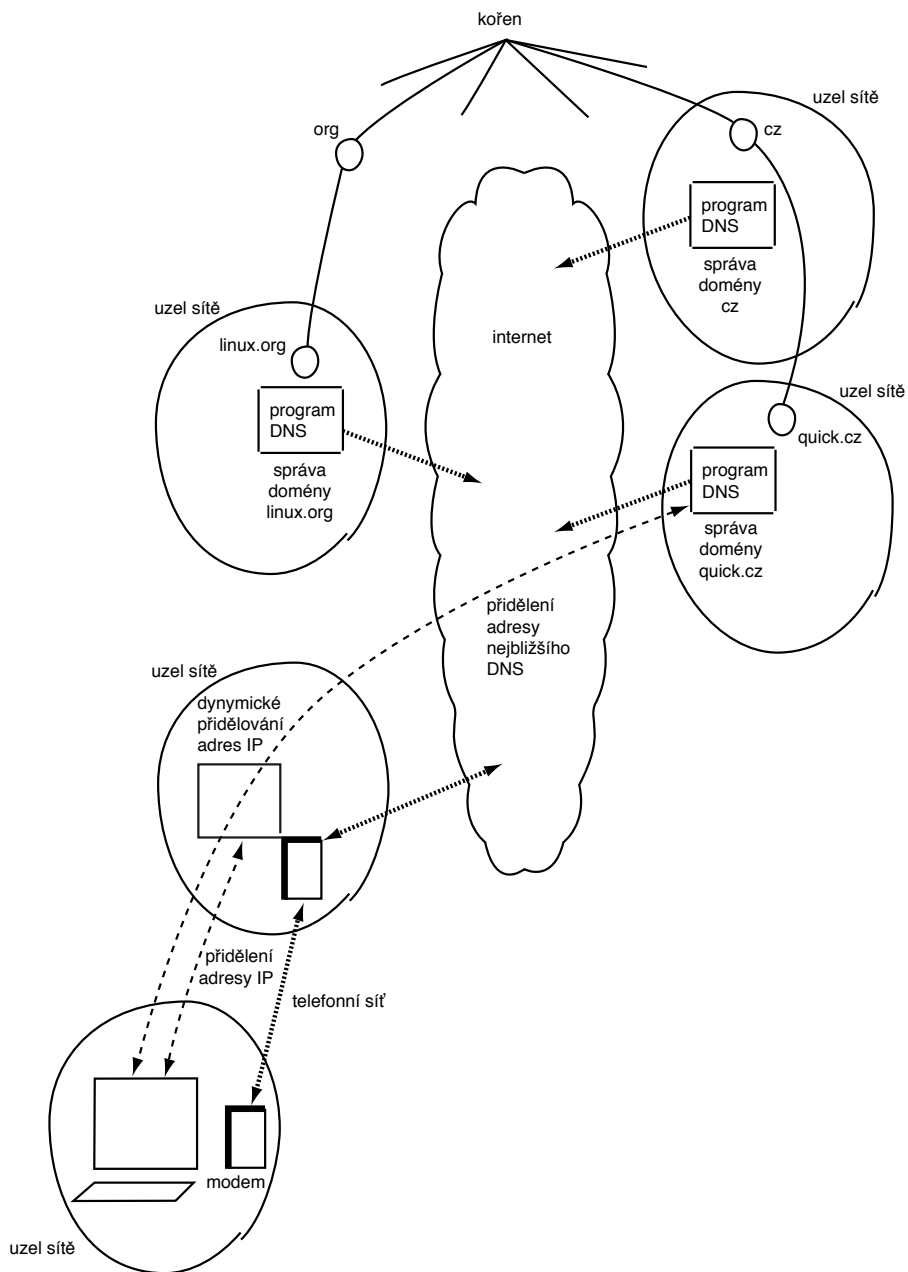
řádu `cz` (ale není, Masarykova univerzita v Brně má svůj vlastní server DNS s doménou druhého řádu `muni.cz`, to vím taky). Znamená to, že kolik domén je ve správě určitého programu DNS v síti, které to jsou a jakého jsou řádu, je věcí dohody lidí. Je to pohodlné, protože např. všechno, co je pod doménou `muni.cz` (např. `med.muni.cz`, `www.muni.cz`, `www.cba.muni.cz`), je ve správě počítače s doménou `muni.cz` a za správnou definici jejích podřádů a jejich zavěšení na správné adresy IP na různých počítačích v celé síti Masarykovy univerzity je zodpovědný správce této domény. On rozhoduje, kdo bude mít celou podstrukturu definovanou u něj a komu naopak svěří nižší doménu (např. `med.muni.cz`) do vlastní správy.

Síťová aplikace DNS je typickým příkladem provozu podpory sítě, i když si to běžně neuvědomujeme. Např. sedím u počítače, který je zařazen v DNS do oblasti `med.muni.cz`. Napíšu-li tedy do svého prohlížeče Internetu odkaz `www.linux.org`, podle toho co bylo řečeno, se nejbližší počítač s programem DNS (můj server DNS, zde na uzlu se správou domény `med.muni.cz`) např. vyjádří, že takový odkaz v životě neviděl a požádá o radu server správy domény vyššího řádu, do které je zařazen (`muni.cz`). Pokud si i tento počítač neví rady, předá dotaz síti na správce vyšší domény. V nejhorším případě se můj dotaz dostane na počítač se správou domény `cz`, který pak pokračuje do kořene DNS v Internetu a dále zase dolů např. do správy domény `org` a `linux.org`. Je vyhráno, zde se potvrdí existence `www.linux.org` a adresa IP, na kterou je `www.linux.org` zavěšena, se žene zpět síti k mému počítači, aby se můj prohlížeč Internetu mohl na takto získané adrese IP zajímat o nabízené stránky. Nutno ještě dodat, že všechny běžící programy DNS v síti si již vyřešené odkazy pamatují, a pokud se v doméně `med.muni.cz` najde v průběhu několika dalších hodin někdo, kdo požaduje rovněž návštěvu domény `www.linux.org`, program správy `med.muni.cz` tuto adresu IP vytáhne ze svého zápisníčku. Každý běžící program, který je součástí DNS, má takový zápisníček, kterému říkáme vyrovnávací paměť (buffer cache). Každý takový zápisníček má nastavenou ale expirační dobu informací, takže např. pokud nikdo doménu `www.linux.org` z oblasti `med.muni.cz` nenavštíví po dobu 8 hodin, je informace o její adrese IP zapomenuta a příští odkaz na ni projde opět již uvedeným mechanismem. Aniž si to my běžní uživatelé uvědomujeme, použijeme-li např. chybně odkaz ve jménu zdroje v Internetu, např. `www.nikde.nikdo.santafe.edu`, proletí náš dotaz v průběhu nejvýše několika vteřin na druhou stranu zeměkoule, aby se zjistilo, že si vymýšlíme a nic takového neexistuje (alespoň doufám). To vše bez kontroly pasu na hranicích nebo zaplacení cla a DPH za využití služeb výpočetní techniky v zahraničí. Pravdou je, že pokud někdo můj výmysl vzápětí zaregistruje a vytvoří reálný odkaz na adresu IP, rázem bude odkaz řešitelný. Tento ráz ale není zase rázem příliš rázným, protože si můj nejbližší server DNS pamatuje, že takový odkaz v posledních osmi hodinách nenašel a odmítá se jím znova zabývat (po dobu, která zbývá do vypršení 8 hodin). Stejně tak jeho spolupracující servery, které touto stupiditou obtěžoval. Proto nově vznikající domény nemusí být viditelné v síti ihned a každý poskytovatel Internetu vás ve chvíli kdy vám zaregistruje doménu upozorní, že viditelná beze zbytku

bude přibližně do 2 dnů, tj. kdy cache všech programů DNS projdou expirací, tedy se ze všech zápisníků vymažou i dřívější pitomosti, které pitomostmi být přestaly a staly se skutečností.

DNS je typickým představitelem síťové služby, která pracuje skrytě a přitom usnadňuje uživateli práci. Převádí běžně používaná jména na číselné adresy (IP), na kterých je internet vystavěn. Za jménem oblasti si přitom nemusíme představovat určitý počítač, protože např. zadáme-li neúplné a přesto existující jméno oblasti DNS (např. *.cz*), není vyhodnocena žádná adresa IP (ale např. *www.cz* již existuje). Dále je běžné umisťovat více odkazů na jeden počítač, jinými slovy na tomtéž počítači může být na téže síťové periférii registrováno třeba i několik stovek různých jmen, a to různých hierarchických úrovní. Pokud bude takový počítač dostatečně výkonný (včetně výkonu síťové periférie), ani to nikdo nemusí při běžném provozu pocítit. Pro nepřetržitý provoz (tam, kde je teď dopoledne na planetě večer, bude večer brzké ráno, představte si!) každá doména, pokud má být živá, musí tedy aktivně odpovídat na adrese IP. Počítač s takto zavěšenou doménou musí běžet nepřetržitě. Při využívání služeb poskytovatelů Internetu (hosting) se to rozumí samo sebou a my můžeme v klidu odpojit svoje PC od telefonní sítě a vypnout je, jít si schrupnout, a přesto tam, kde je v průběhu našeho spaní zářivý den, si mohou pro své pobavení nebo práci zobrazovat informace z naší domény. Naše doména je umístěna na nepřetržitě běžícím počítači (což je význam termínu hosting), který je trvalou součástí Internetu a tento stav nijak nesouvisí s tím, odkud fyzicky a z jaké domény se momentálně připojujeme k Internetu my (z domácího PC, z práce nebo z kavárny), protože naše připojení je klientské a nikoliv serverové.

Také je důležité si uvědomit, že ať už je naše registrovaná doména kdekoli, nesouvisí to s nastavením např. našeho domácího PC připojovaného k Internetu třeba prostřednictvím vytáčené telefonní linky nebo kabelové televize. I takto připojovaný počítač musí mít přidělenou adresu IP a být součástí celosvětové sítě, jinak si v Internetu ani neškrtneme. Dynamicky se připojující počítače z různých telefonních čísel jsou ošetřovány tak, že síťové vrstvě našeho operačního systému je vždy po navázání telefonického spojení s počítačem za telefonní ústřednou přidělena adresa IP dynamicky z určitého provozovaného rozsahu a také se dynamicky přidělí adresa serveru DNS, na kterou naše síťové aplikace posílají dotazy na převedení jmen v síti. Situaci zachycuje kresba II-13.



Kresba II-13: Počítač využívá služby internetu prostřednictvím telefonu.

II.2 – klient a server

Uživatelé v síti zastupují programy, jejichž prostřednictvím uživatel pracuje s výpočetními zdroji jiných počítačů. Výpočetní zdroje jiných počítačů jsou nabízeny do sítě také prostřednictvím programů, ty očekávají zájem připojujících se uživatelů. Např. prohlížeč Internetu (Mozilla) na pokyn uživatele hledá zdroj informací stránek WWW tak, že kontaktuje na odpovídající adrese IP počítač a na něm čekající program pro poskytnutí požadované stránky. Program, který požaduje službu nad výpočetním zdrojem nazýváme *klient* (client), program, který tuto službu poskytne je *server* (v překladu doslova sluha). Je zřejmé, že tyto dva programy si musí mezi sebou vyměňovat požadovaná data, ale i doplňující informace o vzájemné komunikaci (např. o která data vlastně jde nebo pozastavování toku přijímaných/odesílaných dat v případě pomalé přenosové rychlosti sítě). Způsob takové výměny informací nazýváme *komunikační protokol*, podobně jako lidé spolu hovoří určitým jazykem. Komunikační protokol vlastně určuje typ síťové služby. Je podstatný a každý program, ať už server nebo klient, který jej dodrží, se může stát účastníkem dvoustranné komunikace tohoto typu. Znamená to, že v počítačových sítích po dvojicích vzájemně komunikuje vždy klient a server, běžně přitom každý naprogramovaný někým jiným. Pokud programátoři respektovali obecnou úmluvu, tedy definici komunikačního protokolu, nic tomu nestojí v cestě. Proto se při určení typu spolupráce klientu a serveru uvádí především jejich komunikační protokol. Např. klientem stránek WWW může být Mozilla nebo MS Internet Explorer, serverem může být Apache nebo IIS. Podstatný je ale HTTP (Hypertext Transfer Protocol – protokol přenosu hypertextu), což je obecně definovaný způsob přenosu dat při zobrazování stránek WWW, který obě strany při komunikaci dodržují. Definice komunikačních protokolů různých služeb veřejných sítí (Internetu) jsou obecně dostupné jako průmyslový standard na adrese www.w3.org.

komunikační
protokol

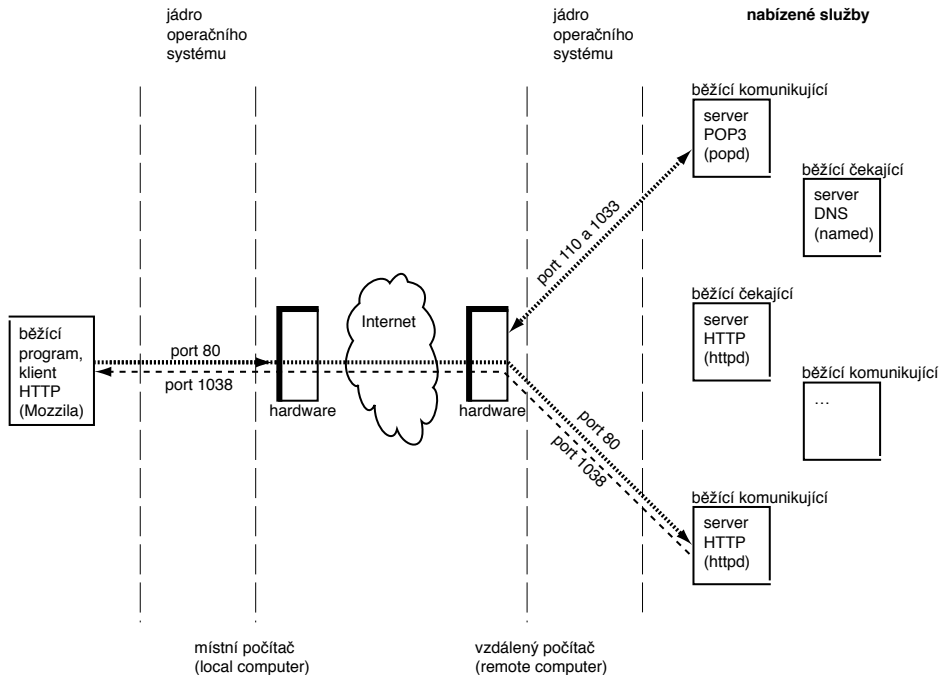
Protokolární spolupráci programů klient a server nazýváme *architektura klient – server* a je pro současné počítačové sítě klíčová.

architektura
klient – server

Vzpomeneme-li na čísla portů transportní vrstvy na kresbách II-5 až II-7, uvědomíme si jejich význam v komunikaci klient – server. Síťová aplikace je současný běh dvou spolupracujících programů na různých počítačích. Jejich spolupráce je určena komunikačním protokolem. Aby operační systém klientovi přidělil do sítě server správného typu, prokazuje se klient při prvním kontaktu číslem portu. Číslo portu příchozího požadavku ze sítě identifikuje komunikační protokol a tím také odpovídající server. Přidělování čísel portů komunikačním protokolům je opět věcí obecné domluvy. Jsou uvedena opět na www.w3.org, ale každá instalace operačního systému se skupinou síťových protokolů TCP/IP zahrnuje všechny doposud definované porty. V operačním systému UNIX jsou uloženy v souboru `/etc/services`, v MS Windows NT (tj. také 2000, XP) v souboru `C:\WINNT\system32\drivers\etc\services`. Jedná se o obyčejný textový soubor, ve kterém lze číst, že např. HTTP má přidělen port s číslem 80 nebo protokol stahování pošty POP3 má číslo portu 110, šifrovaný

services

přenos dat stránek WWW, tj. protokol HTTPS, má přidělen port číslo 443, komunikace v DNS číslo portu 53 (jeho jméno je `domain`) atp. Kresba II-14 je ukázkou nalezení požadované síťové komunikace.



Kresba II-14: Klient HTTP a služby vzdáleného počítače.

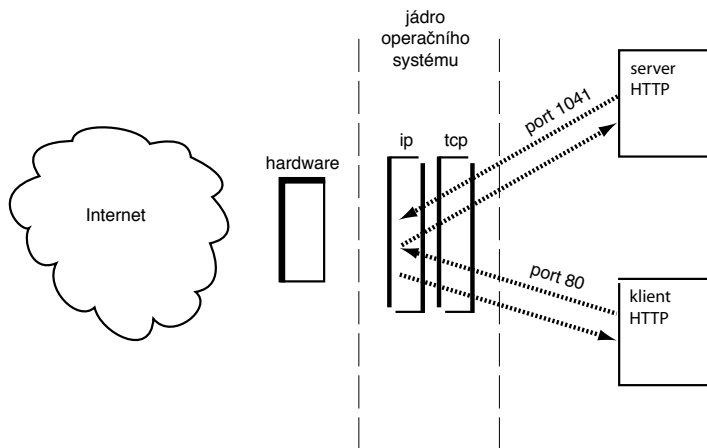
Klient komunikace HTTP oslovuje vzdálený počítač oznámením svého výchozího portu č. 80. Vzhledem k tomu, že ve vzdáleném počítači je převodní tabulka `services`, operační systém, který tento požadavek ze sítě zachytí (prostřednictvím transportní vrstvy TCP) ví, že má jako server tomuto novému síťovému spojení přidělit program serveru HTTP, zde program `httpd` (Apache, a je zkratka démon, angl. `daemon` – jedná se o terminologii probouzejících se a neviditelně pracujících procesů v UNIXu). Současně (viz kresba II-7) tomuto síťovému spojení přidělí dočasný port č. 1038. Spojení našeho klienta a požadovaného serveru je ustanoveno a oba programy si mohou začít vyměňovat data. V okamžiku příchozího klienta již ovšem probíhalo síťové spojení pro stahování pošty, a to výchozím portem číslo 110 a dočasně přiděleným portem číslo 1033. Našeho klienta získal jeden z běžících a čekajících serverů HTTP, kterých je vždy v počítači nabízejících tuto službu více (obvykle nejméně 16). Získal toto nové spojení, protože o tom rozhodl operační systém. Současně je podle kresby také připraven server DNS a čeká na příchozího klienta také. A tak podobně.

V operačním systému serverů (síťových služeb) mohou být samotné servery operačním systémem spravovány různým (pro uživatele ale nepodstatným) způsobem. Samotný program serveru může být po příchozím požadavku

klientu vyhledán na disku a teprve pak spuštěn nebo může být připraven již běžící program (je to rychlejší, pokud ale víme, že jej skutečně někdo bude často chtít). Také se používá metoda jednoho nebo více serverů téže služby, tj. pokud je operační systém vystaven současně několika požadavkům na spojení téhož typu, může je pokrýt několika nezávisle běžícími servery (typicky httpd) nebo může několik síťových požadavků současně zajišťovat jeden server (typicky DNS). I zde záleží na frekvenci a mohutnosti požadavku na přenos dat, podle níž je služba vytipována a spravována odpovídajícím způsobem. To vše je v moci správce operačního systému poskytujícího síťové služby.

Na kresbě II-14 jsme také poprvé použili termíny místní a vzdálený počítač. Pojmy místní (local) a vzdálený (remote) jsou používány jako adjektivum nejenom u pojmu počítač, ale i u pojmů služba, síť atp. Jedná se o síťovou terminologii ve smyslu rozlišení strany klientu a serveru. Vzdálený je přitom vždy prvek strany, která je na opačném konci komunikace, místní naopak. Druhý konec síťového spojení ale nemusí být vždy fyzicky vzdálený (!). Typickým příkladem této zdánlivé podivuhodnosti může být požadavek uživatele zobrazit stránky WWW nabízené serverem počítače, na němž právě pracuje. Lépe situaci vyjádříme na kresbě II-15.

**místní (local)
a vzdálený
(remote)**



Kresba II-15: Klient a server místního počítače.

Klient oslovuje server na tomtéž počítači. Jedná se o využití místní služby. Spojení mezi klientem a serverem je přitom ustanoveno i v případě, že nemáme fyzické připojení do sítě. Internet dokonce pro takový typ spojení vyhradil adresu IP, tou je 127.0.0.1, které je přiřazeno jméno localhost (a to mimo servery DNS, v překladu doslovně místní hostitel). Mechanismus takového propojení klienta se serverem označujeme termínem loopback (zpětná smyčka). Takové místní spojení sice prochází modulem IP, ale vrací se zpět jako příchozí požadavek ze sítě. Každý počítač se síťovým software má tedy k dispozici zpětnou smyčku, pomocí které lze testovat např. síťovou aplikaci při jejím programování. Nejjednodušeji prověříme existenci software TCP/IP příkazem `ping` dle kresby II-16.

**localhost,
loopback**

```

Příkazový řádek

C:\>ping localhost

Příkaz PING na skocovsky [127.0.0.1] s délkou 32 bajtů:

Odpověď od 127.0.0.1: bajty=32 čas < 1ms TTL=128
Odpověď od 127.0.0.1: bajty=32 čas < 1ms TTL=128
Odpověď od 127.0.0.1: bajty=32 čas < 1ms TTL=128
Odpověď od 127.0.0.1: bajty=32 čas < 1ms TTL=128

Statistika ping pro 127.0.0.1:
Pakety: Odeslané = 4, Přijaté = 4, Ztracené = 0 (ztráta 0%),
Přibližná doba do přijetí odezvy v milisekundách:
    Minimum = 0ms, Maximum = 0ms, Průměr = 0ms

C:\>

```

Kresba II-16: Prověření síťového software typu internet pomocí loopback.

Architektura klient – server tedy obecně umožňuje na počítači provozovat současně jak klienty tak servery. Zda je jejich protějšek umístěn na jiném nebo tomtéž počítači je věcí samotného vedení provozu. Architektura klient – server je totiž obecně definována jako komunikace dvou běžících programů, kdy jeden z nich poskytuje výpočetní zdroj druhému. Poskytování zdroje je obecně možné více způsoby: nejenom sítí, ale např. souborem, sdílenou pamětí (shared memory), frontou zpráv (messages), rourou (pipe) atd., podrobněji viz [Skočovsky1998]. Teprve u komunikace sítí používáme termíny místní a vzdálený, abychom určili odpovídající protějšek komunikace.

počítač
jako server
a jako klient

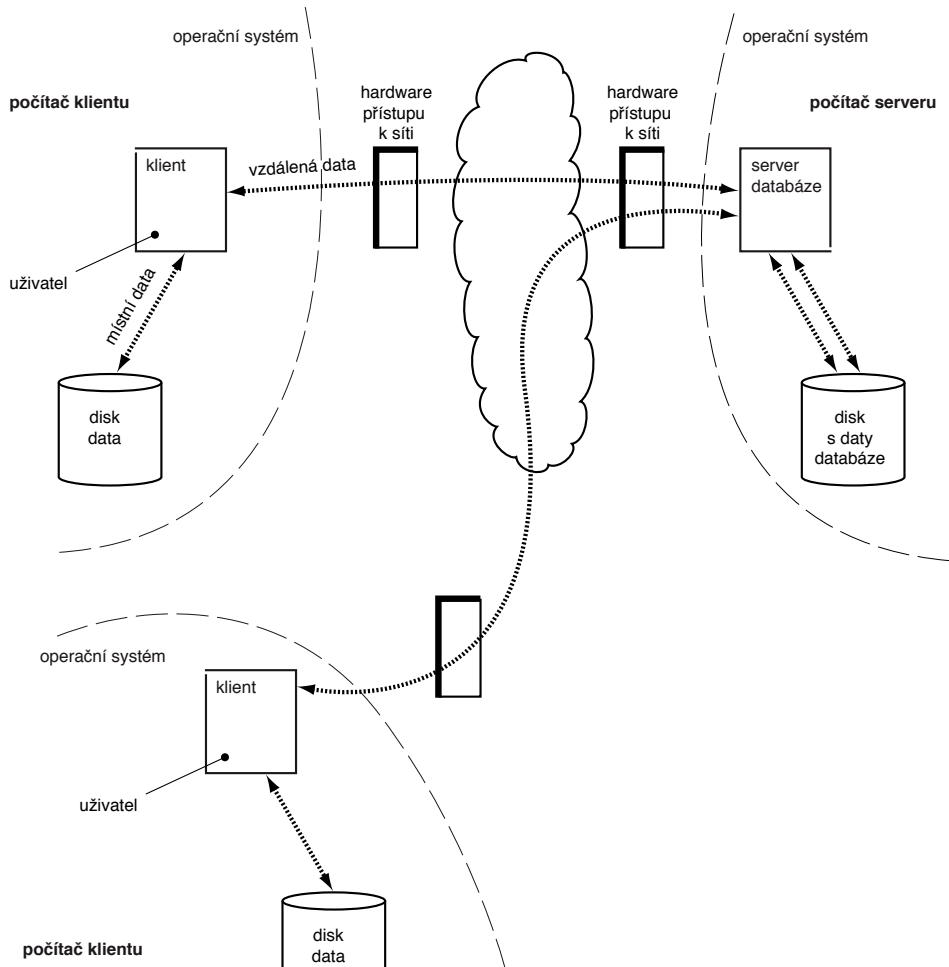
Skutečně, všechny současné používané operační systémy disponují možností provozu síťových serverů, a to i v případě, že se jedná o pracovní stanici určenou pro práci uživatele s účelově zaměřenou aplikací (návrhové systémy, kancelářské systémy atp.). Umožňuje to víceprocesovost současných operačních systémů. Počítač dokáže vykonávat souběžně provozované programy, aniž si toho interaktivně pracující uživatel vůbec všimne. To, jakým způsobem je pak síť provozně koncipována, záleží na úvaze správce sítě, protože pro klienty je výhodné spolupracovat se serverem běžícím na spolehlivém a výkonném počítači a je možná lepší, když samotný uživatel např. stanice typu PC není frustrován nejistotou, zda může svůj počítač vypnout a jít na oběd, když možná některý z běžících programů jeho počítače je serverem klientů v síti. Od koncepce síťového provozu se odvíjí terminologie označování některých nepřetržitě běžících počítačů v síti jako servery a pro počítače koncových uživatelů pracujících ve výkonném grafickém prostředí klientů jako klienty. Použijeme-li výraz server, v obecném povědomí si uživatelé představují počítač, odborník běžící program, u termínu klient si uživatel obvykle raději nepředstaví nic (nikoliv však po přečtení tohoto textu!). Terminologie je zde vágní. V tabulce *services* můžeme najít řadu aplikačních síťových protokolů, kterým jsou přidělena obecně používaná čísla rezervovaných portů. Obecné síťové služby, které mají tuto podporu, slouží pro známé typy síťových aplikací, na kterých vlastně je založeno současné využívání Internetu. V případě, že tyto síťové služby nestačí uspokojit nároky na využití sítě a jejích výpočetních

zdrojů, programují se nové síťové aplikace za využití jiných, doposud nepoužívaných portů. Čísla portů nad 3000 jsou totiž volná. Může se ovšem stát, že se programátoři nových síťových aplikací sejdou na shodných číslech portů, při instalaci nové síťové aplikace v provozu je proto úkolem správce sítě ve spolupráci s programátorem případné kolize vyřešit (přidělením jiných čísel portů).

Síťová aplikace nemusí být postavena pouze na dvou komunikujících procesech. Při dodržení architektury klient – server může provozovaná aplikace zahrnovat několik různých klientů a různých serverů, které mezi sebou vzájemně komunikují různými protokoly. Proto jsou síťové programovací jazyky uzpůsobeny typicky pro tyto možnosti. Jeden z nejvíce citovaných a doposud stále používaných programovacích systémů pro síťové aplikace je RPC (Remote Procedure Calls - Volání vzdálených procedur). RPC umožňuje programovat klienty tak, že jeho procedury (což je obecněji definovaný termín pro funkci) jsou realizovány serverem, tedy jiným procesem. Klient tak může při své práci postupně nebo i současně volat procedury různých serverů. V jazyce RPC na uvedeném principu spolupráce několika různých klientů a serverů je např. naprogramována aplikace NFS (Network File System). NFS umožňuje uživateli pracovat se strukturou adresářů ve vzdáleném operačním systému.

**síťové
programovací
jazyky (RPC)**

Klient a server si vyměňují data, což je účelem síťové komunikace. V principu se přitom jedná o poskytování dat serverem, přestože obecně tomu tak není. Např. tisk na vzdálené tiskárně je předání dat klientem serveru tiskárny, tedy server poskytuje spíše výpočetní zdroj než obecně data. Z pohledu centrálního shromažďování dat tomu tak ale je. Server může být programem, který je součástí databázové podpory práce informačního systému organizace. Je dotazován klienty z různých částí sítě, pro které data z databáze čte nebo je případně podle jejich pokynů mění. Znamená to, že každý klient může pracovat se svými místními daty a pro společná vzdálená data se dotazuje na vzdálený server. Podívejme se na kresbu II-17.



Kresba II-17: Místní a vzdálená data klienta.

Uživatel je ten, kdo spouští klienta, po kterém požaduje zpracování dat. Klient, ať už s uživatelským vědomím nebo bez něj, čerpá data pro zpracování jak z místního disku, tak ze vzdálené databáze. Klient přitom není jediným klientem, který data od serveru přijímá, jeho místní data nejsou ale poskytována nikomu jinému. Může si tak např. uchovávat kopii dat od serveru pro pozdější opětovné zpracování nebo si uschovávat výsledky svých výpočtů pro pozdější přehledy. Každopádně při používání místních dat riskuje, že oproti datům v centru mohou být zastaralá. Jsou ale naopak data, která mají archivní charakter, a tedy jsou neměnná. Síťová aplikace ale může také být naprogramována tak, že klient vůbec nepoužívá místní data a všechny informace čerpá ze serveru a své výsledky rovněž umísťuje prostřednictvím vzdáleného serveru do centrální databáze. To je z hlediska požadavku aktuálních dat velmi výhodné. Je to také výhodné při aktualizaci dat tímto klientem v databázi, protože pokud výsledky práce klienta nemají

soukromý charakter, jsou ihned viditelné a akceptovatelné i jinými klienty v síti. Potíže nastávají pochopitelně při přepisu týchž dat různými klienty současně. Souběh změny stejných dat různých klientů je kolize (např. při rezervaci jízdenky ze dvou míst současně), kterou musí řešit server (např. tzv. zamykáním dat).

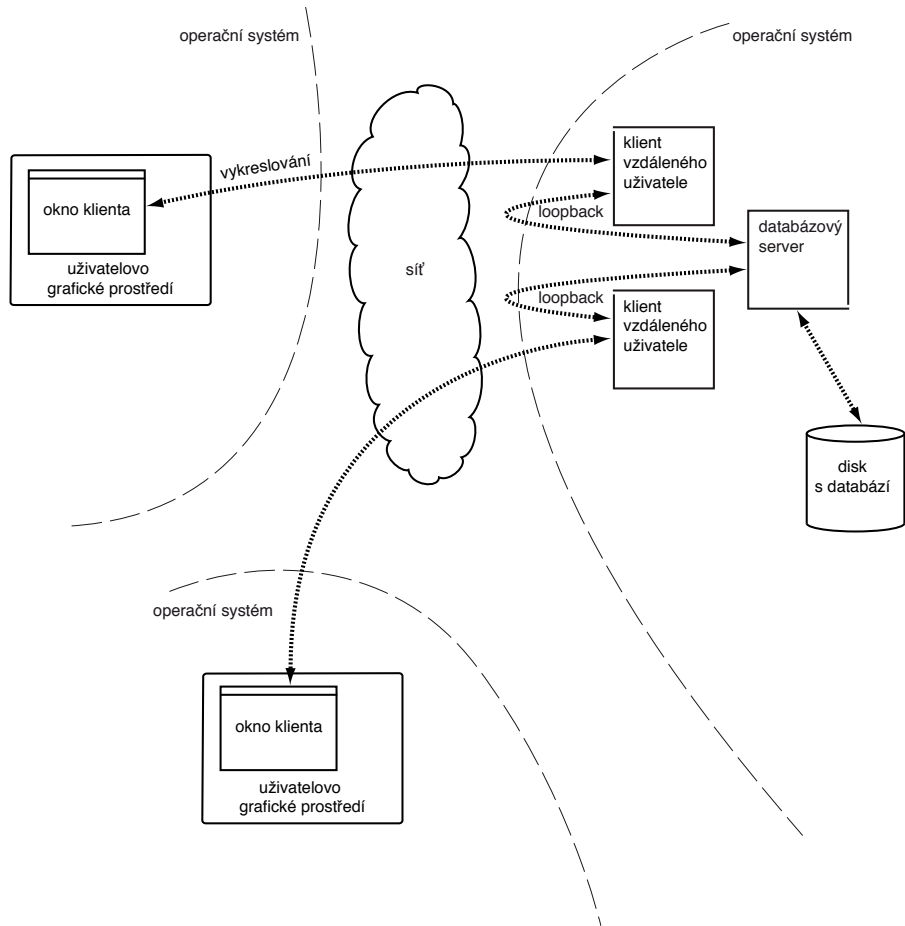
Míra práce klienta současně s místními i vzdálenými daty určuje míru jeho tloušťky (fat). *Tlustý klient* (fat client, také rich-client) je takový, který využívá zpracování místních dat v co nejvyšší možné míře, *tenký klient* (thin client) je ten, který pracuje pouze se vzdálenými daty. Také server může být různě tlustý, a to podle míry zpracování dat, kterou po něm klienty požadují. Princip konstrukce síťových aplikací na bázi tenkých klientů vyžaduje trvalé, rychlé a kvalitní síťové spojení s *tlustým serverem*. Předpokládá rovněž vysoký výkon, stabilitu a minimální výpadky počítače s centrální databází a databázovým serverem. V současné době je ale v teoretické i praktické Computer Science považován za progresivní. Představitelem tenkého klienta je prohlížeč stránek WWW, přestože i on používá Java applets, cookies a vyrovnávací paměti pro rychlejší interpretaci již jednou přenesených dat. Představitelem tlustého klienta je např. většina poštovních programů v prostředí PC, které využívají pro příjem a odesílání pošty protokoly POP (Post Office Protocol) a SMTP (Simple Mail Transfer Protocol). Od serveru data (poštu) přijímají (stahují) nebo mu je předávají po celcích. Čtení pošty, manipulaci s ní nebo psaní nové pošty provádí klient místně bez potřeby stálého spojení se serverem.

tenký a tlustý
klient

Na principu různě tlustých klientů si můžeme také dobře uvědomit význam obecně používaných termínů *on-line* (nutnost být připojen) a *off-line* (bez potřeby být připojen). Tenký klient je vždy on-line. Práce off-line pro něj nemá smysl, protože všechny jeho podstatné procedury zajišťuje server. Tlustý klient může pracovat off-line třeba hodiny i dny, práce on-line je pro něho výjimečný stav.

on-line, off-line

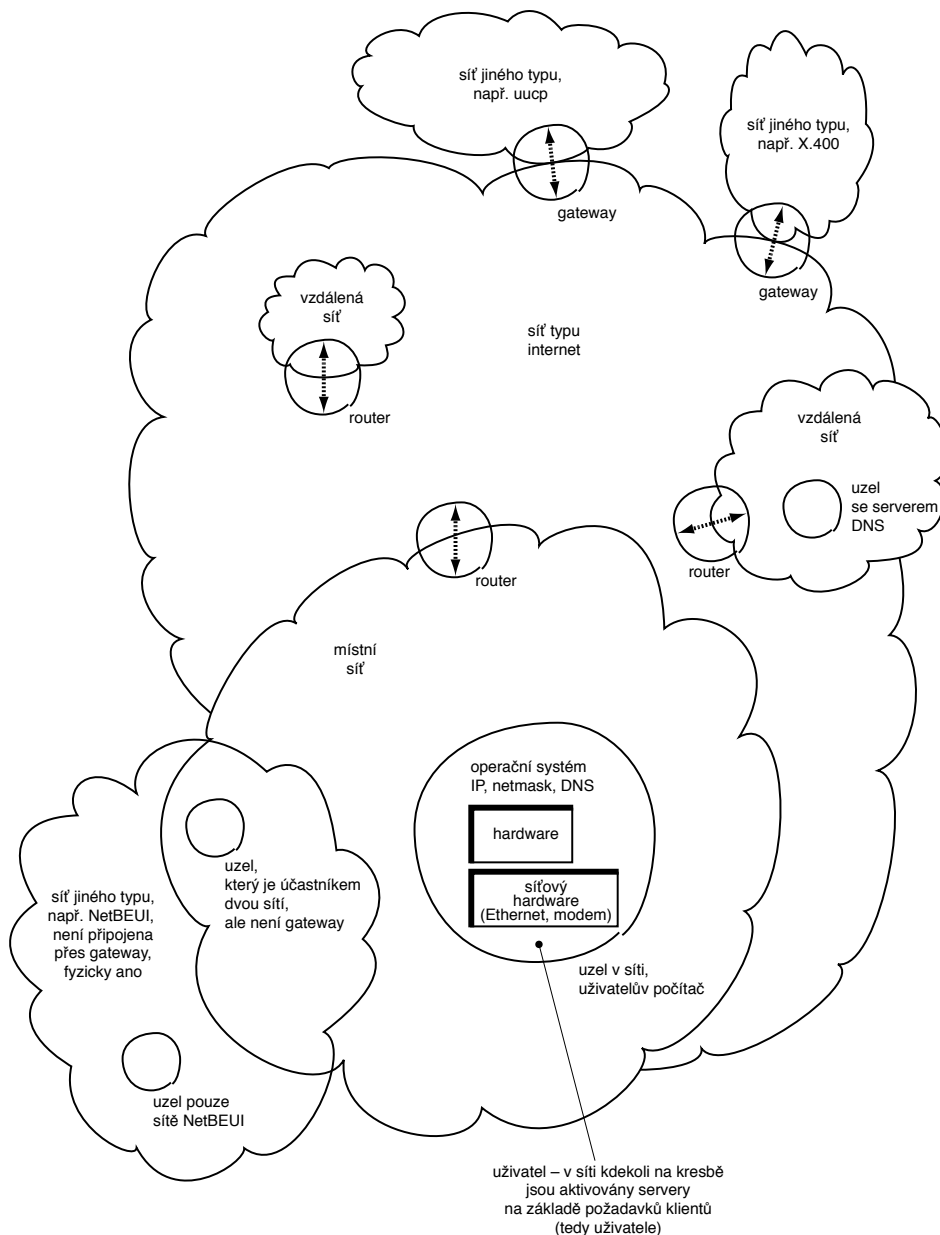
Zajímavým příkladem velmi tenkého klienta je práce grafických programů v prostředí X Window System, což je grafické prostředí každého UNIXu, označující se často pouze termínem X. Uživatel spouští program tak, že o něj požádá vzdálený počítač. V okamžiku, kdy se ve vzdáleném operačním systému program rozběhne, spojí se s počítačem, který o něj požádal a v grafickém prostředí uživatele vykreslí grafické okno, ve kterém zobrazuje výsledky výpočtu. Strana klienta je takto odlehčena od samotného provádění programu, musí pouze rozumět pokynům grafického vykreslování. Situace je na kresbě II-18.



Kresba II-18: Klient a server v X Window System.

Vidíme, že uvedená technologie využívá principy architektury klient – server po svém, ale nikoliv v rozporu s ní. Klient i server běží na tomtéž počítači, veškeré výpočty se odehrávají v centru. Pouze grafická část je přesunuta na místní počítač. Komunikace sítí je přitom redukována na smluvené pokyny grafickému prostředí, což snižuje požadavky na její rychlost. Požadovaný výkon je striktně rozdělen na část výpočtovou a grafickou.

Kresba II-19 je souhrnem nutného povědomí, které v současné době uživatel potřebuje mít o práci svého počítače v síti typu internet.



Kresba II-19: Uživatel v síti.

Podstatné síťové informace na kresbě jsou uvedeny u textu operační systém uživatelského počítače. Je to adresa IP, vazba na síťovou masku (netmask) – tedy určení rozsahu místní sítě a určení adresy IP se serverem DNS. Kromě funkčního hardware a korektní instalace operačního systému se síťovými moduly nepotřebuje uživatel pro práci v síti nic jiného. Router je spojení sítí téhož typu, v našem případě typu internet.

gateway Kruh, kterým na kresbě oddělujeme počítač od sítě, je operační systém. Uzel je v síti typu internet účastníkem určité podsítě. Současně ale může na fyzickém síťovém spojení používat i protokolární komunikaci se sítí jiného typu (např. NetBEUI firmy Microsoft). Klient spuštěný uživatelem tak může pracovat tímtež aplikačním protokolem v sítích různého typu, využívat server, který nabízí služby v jiném typu sítě. Přitom existuje fyzické spojení obou počítačů komunikace, ale není vytvořena gateway mezi dvě sítě různého typu. Nemusí být. Gateway (brána) je výpočetní zdroj, který spojí dvě sítě tak, že dojde k překladu síťové komunikace jednoho typu na typ jiný. Síť jiného typu je pak pro uživatele transparentní, vidí služby jiného typu sítě jako kdyby se jednalo o síť, již je účastníkem. V uzlu, který je účastníkem dvou sítí různého typu (operační systém umí oba typy sítí) a současně není gateway, může uživatel využívat přístup ke zdrojům obou sítí. Nelze však přes něj projít z jiného počítače do druhé sítě.

uzel, host, hostitel Na kresbě II-19 se objevují nové pojmy. Uzel je termín, jemu odpovídající anglický termín je host, který je do češtiny také překládán jako hostitel. Je jím obvykle myšlen počítač se serverem, ale obecně je takto chápán počítač, který akceptuje síť a odpovídá jejím protokolům, a to na různé úrovni, třeba jen síťové.

Uživatel používá klienty, jim slouží servery, které čekají a na pokyny klientů se aktivují. Je to pěna sítě, pěna dneška.

II.3 – síťové aplikace: přenos dat, vzdálená práce, sdílení zdrojů, komunikace, publikování

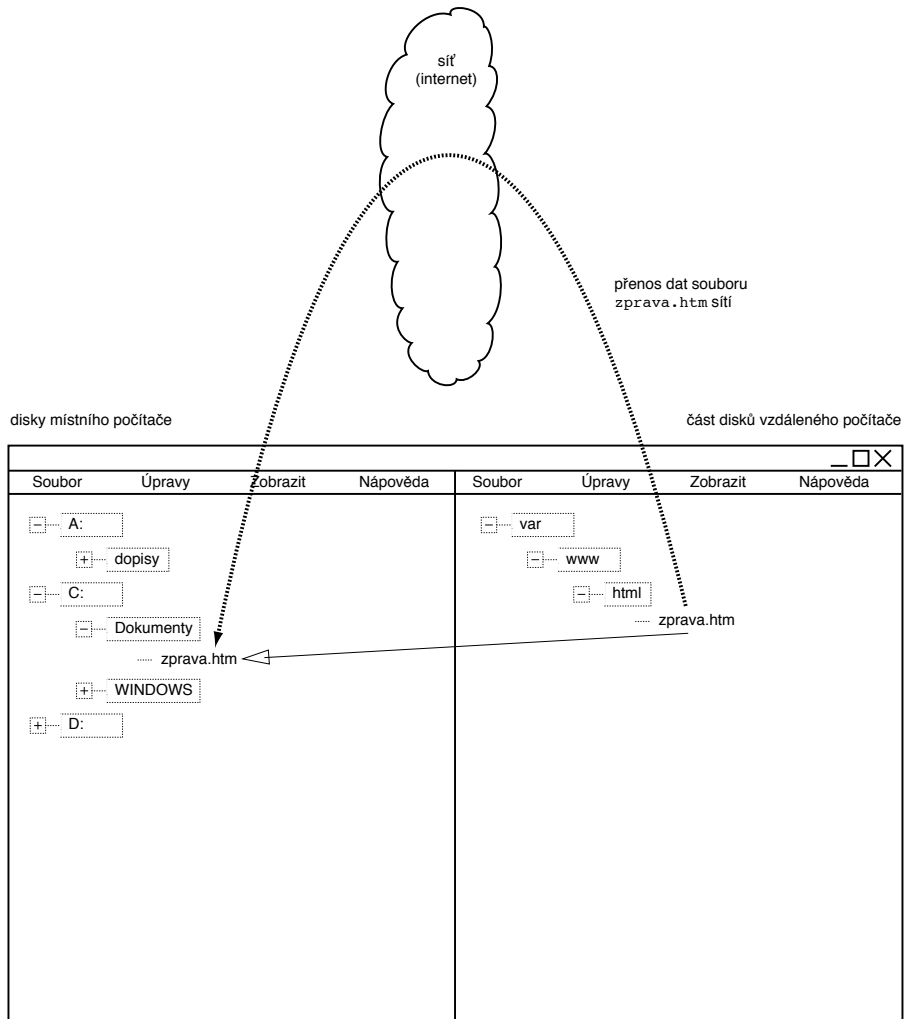
Pro potřeby vzájemné komunikace jedinců si lidská společnost v rámci vývoje manipulace s digitalizovanými informacemi vytvořila propojení jednotlivých výpočtů mezi sebou. Charakter každého takového propojení přitom byl (a je) vždy vynucen konkrétní potřebou uživatele. Přestože potřeby různých uživatelů mohou být výrazně odlišné, v rámci technického řešení se v průběhu let ustálilo několik typů síťových služeb, které obecně vyhovují daným možnostem a potřebám lidí používajících výpočetní techniku. Lze je vymezit následujícími požadavky:

- přenos souborů,
- práce ve vzdáleném uzlu,
- sdílení periferií,
- elektronický rozhovor a pošta,
- elektronické publikování.

K jednotlivým odrážkám lze přiřadit veřejně podporované komunikační protokoly síťových aplikací dnešního Internetu (viz tabulka *services*). Vhodné protokoly pak používají klienty síťových aplikací a ve spolupráci se svými servery tak zajišťují uživateli jeho potřeby.

Abychom mohli pokračovat v práci na rozepsané zprávě, dopisu, povídce nebo vůbec na jakémkoli textu, pokračovat v grafickém návrhu, kresbě, zkrátka pokračovat v doplňování obsahu souborů po příchodu z pracoviště domů, musíme mít tyto soubory dat k dispozici. Základním předpokladem je pochopitelně vlastnictví počítače, který je adekvátní tomu, který používáme na pracovišti s instalací potřebných programů, na kterých byla data souborů pořízena. Dalším předpokladem je síťová konektivita, tedy možnost připojení domácího počítače k síti, jejímž uzlem je také uzel na pracovišti, který disponuje serverem s nabídkou potřebných souborů. V okamžiku, kdy naše lidské děti usnou a domácí počítač je konečně v tichu a klidu domácnosti k dispozici pro soustředěnou práci, můžeme využít klienta síťové služby FTP (File Transfer Protocol – protokol přenosu souborů). Ve světě PC a MS Windows je FTP součástí již zmiňovaného software Total Commander (viz www.ghisler.com). Schématicky si můžeme ukázat princip přenosu souborů ze vzdáleného uzlu podle kresby II-20.

přenos souborů



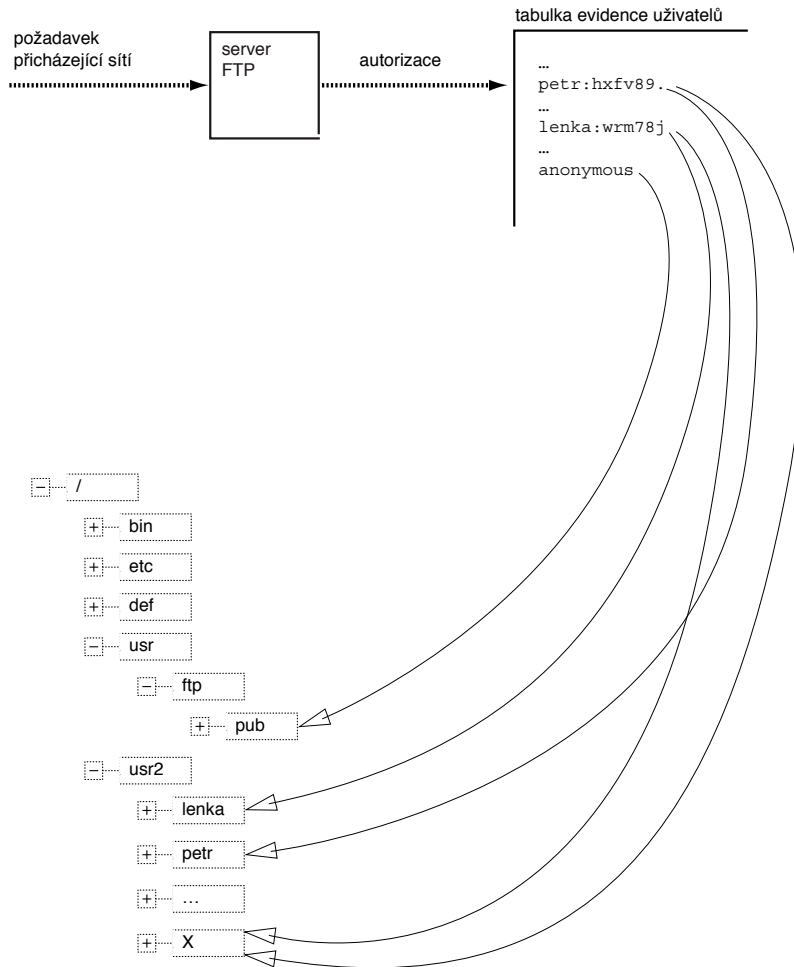
Kresba II-20: Princip použití FTP v okně klienta uživatele.

Kresba ukazuje situaci samotného přenosu dat (zde souboru `zprava.htm`) ze vzdáleného počítače do místního. Levá část uživatelského okna zobrazuje diskový prostor místního počítače, pravá část okna diskový prostor počítače vzdáleného. Je to situace, kdy již bylo ustanoveno spojení mezi oběma počítači. Uživatel nyní ovládá přístup jak k datům svého, tak k datům vzdáleného počítače. Kliknutím myši na některý ze souborů a tahem myši do druhé části okna pak po uvolnění tlačítka myši (drag and drop, táhni a pusť) spustí přenos odpovídajícího souboru z jednoho počítače na druhý. Provedeme tak kopii dat, nikoliv jejich přesun. Podobně přetažením adresáře z jedné části okna do druhé kopírujeme celý podstrom dat. Uživatelský přístup je přizpůsoben práci průzkumníků s diskovým prostorem (viz část Base Of I.4) počítače, u kterého uživatel právě sedí, a to tak, že se mu v jedné části okna aplikace

zobrazuje diskový prostor vzdáleného počítače. Pochopitelně je přenos dat zatížen problémem rychlosti přenosu dat. Kopie dat ze vzdáleného počítače může trvat daleko déle než pouhá kopie dat na místním diskovém prostoru.

Přístup k datům vzdáleného počítače je serverem prověřován na základě klientem zadaného přístupového jména a hesla. Pro uživatele tato certifikace přístupu k datům v síti proběhne prostřednictvím úvodního dialogového okna, ve kterém je vyzván k vyplnění kolonky se jménem (login) a heslem (password). Jedná se o přihlašovací jméno na straně serveru. Přihlašovací jména a k nim přiřazená hesla eviduje operační systém. Údržba tabulek (textových souborů) s těmito informacemi patří k typické činnosti správce operačního systému odpovídajícího uzlu v síti. Každý takto evidovaný přístup k datům může mít definovanou vlastní oblast diskového prostoru tak, že použije-li klient jiné přístupové jméno, vidí jinou část disku. Kombinovat lze pochopitelně různé varianty, kdy různě evidované klienty mohou vzájemně svá data číst, ale přepisovat mohou pouze data vlastního adresáře. Může přitom navíc existovat veřejná komunikační oblast, která je dostupná všem, a to jak pro čtení, tak pro zápis. Všem znamená každému klientu, který prošel certifikací jménem a heslem. Organizaci způsobu přiřazení dat na disku se serverem FTP pro evidované klienty ukazuje kresba II-21.

**přihlašování
v serveru**



Kresba II-21: Příklad poskytovaného diskového prostoru serverem FTP.

Klient, který použije při certifikaci serverem jméno `lenka`, bude mít zabezpečen přístup k datům od adresáře `/usr2/lenka` a adresáře `/usr2/x`. Do prvního podstromu má unikátní a neomezený přístup jak pro čtení, tak zápis, do druhého může mít přístup pouze pro čtení. Využitím přístupového jména `petr` je k dispozici tatáž varianta, pouze soukromá oblast dat je umístěna od adresáře `/usr2/petr`. Popis přístupu k datům jednotlivých registrovaných jmen je uveden také v dalších systémových tabulkách.

download Zvláštním případem vzdáleného přístupu k datům prostřednictvím FTP je anonymní přístup (anonymous). FTP může nabízet do internetu adresář, který obsahuje data volně dostupná každému klientu celé sítě, a to bez zvláštní autorizace. FTP takto reaguje na definovaný adresář v případě, že je v tabulce operačního systému uveden přístup se jménem `anonymous`. Na kresbě je uveden také adresář `/usr/ftp/pub`, kterým právě začíná oblast takto volně zveřejněných dat. Prostřednictvím FTP je pak možné do Internetu

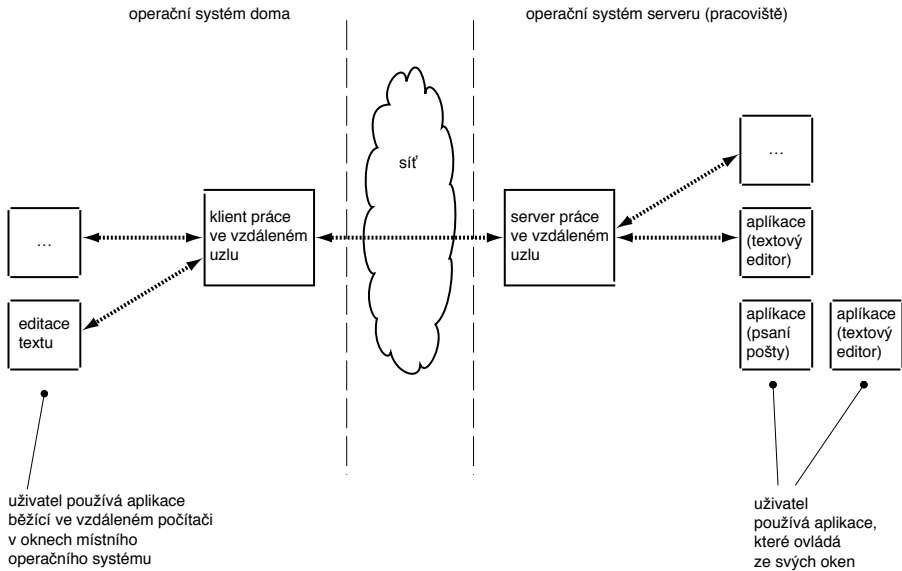
nabízet ke stažení (download) volně software, obrázky, texty. Je sice možné nastavit oblast `pub` také pro zápis, v drtivé většině případů se ale volně nabízí pouze pro čtení. Důvodem je bezpečnost. První hacker, který objevil takto volně dostupnou oblast, do ní umístí program, který pokud je spuštěn, následně paralyzuje operační systém serveru. Klient FTP je také součástí klienta stránek WWW (HTTP, viz dále). Znamená to, že uživatel při prohlížení stránek Internetu při stahování některých balíků dat přechází z klienta HTTP na klienta FTP, aniž si to musí uvědomit.

Oproti přenášení dat prostřednictvím výměnného média (disketa, CD, přenosný disk) je u síťové služby přenosu souborů výhodou možnost stažení i souborů, které jsem si na výměnné médium neuložil v domnění, že je nebudu potřebovat. Při práci s FTP si také musíme uvědomit, že pořizujeme kopii dat ze serveru na počítač klienta. Po ukončení práce na stažených datech bychom proto neměli zapomenout nově upravená data přenést zpět do uzlu serveru, tj. tak, aby byla přístupná další den na pracovišti. Pro přenos dat od serveru ke klientu se používá angl. termín download, pro opačnou cestu upload.

Pravděpodobně lepší variantou pokračování v přerušené práci na domácím počítači by jistě byla možnost nikoliv pouze stažení dat pomocí serveru FTP, ale možnost úplné nabídky pracovního prostředí počítače na pracovišti. Prakticky by to znamenalo např. spustit v novém okně aplikaci (klienta), která by zajistila připojení na server počítače na pracovišti, získala od něj všechny potřebné informace pro nastavení prostředí a v dalším podporovala práci uživatele jako kdyby seděl u vzdáleného počítače. Jinými slovy, uživatel by vstoupil do uzlu serveru a využíval by jej zcela plnohodnotně. To znamená, že nemusím data přenášet z počítače v kanceláři na domácí počítač, pokud mi síť umožní z domácího počítače vstoupit do počítače v kanceláři a pracovat přímo v jeho prostředí. Takové síťové službě říkáme práce ve vzdáleném uzlu. Podívejme se na kresbu II-22.

upload

**práce ve
vzdáleném uzlu**

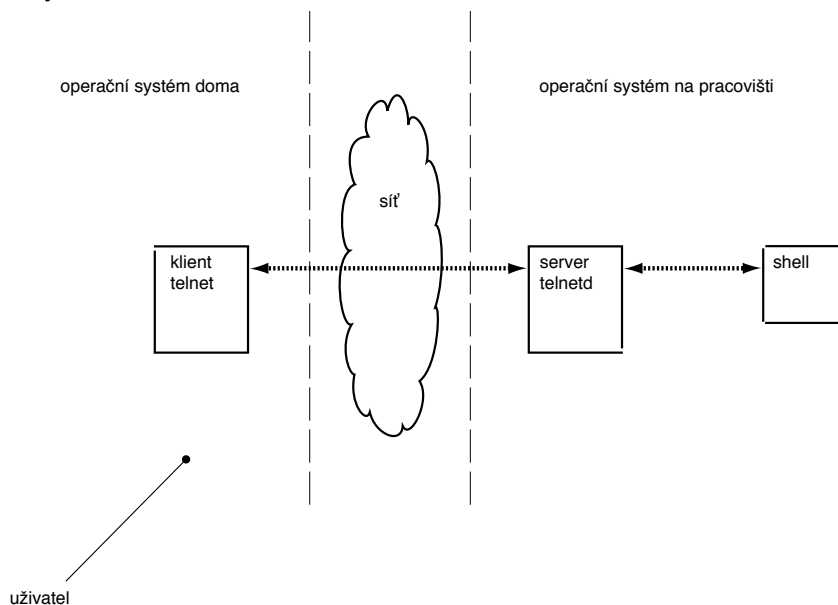


Kresba II-22: Práce ve vzdáleném uzlu.

Klient v počítači uživatele si vyměňuje data se serverem ve vzdáleném uzlu tak, že se např. uvedená aplikace editace textu zobrazuje uživateli v okně jeho pracovního prostředí jako kdyby text editoval na pracovišti. Z kresby je patrné, že klient a server mohou zajišťovat takovou práci současně více aplikacím uživatele. Mezitím může jiný uživatel sedící přímo u počítače na pracovišti využívat program pro editaci textu nad jiným textem nebo může současně přijímat či odesílat poštu, jeho aplikace jsou přitom prováděny přímo připojenými periferiemi a bez kontaktování serveru práce ve vzdáleném uzlu.

telnet Přestože se síťová služba práce ve vzdáleném uzlu objevila velmi brzy jako jedna ze základních aplikací technologie internetu, v současné době ji prakticky disponují pouze sofistikované operační systémy jako je např. UNIX. Ve svém principu je totiž tato síťová služba závislá na vlastnosti operačního systému, které říkáme víceuživatelský přístup (viz část Base Of I.3). Jedná se o podporu práce více uživatelů na tomtéž počítači současně. Původní síťová služba práce ve vzdáleném uzlu měla (a dodnes má) název `telnet` a umožňovala práci v řádkově orientovaném prostředí operačních systémů přelomu 60. a 70. let minulého století. Dodnes službu `telnet` používají zejména správci sítí a operačních systémů např. typu UNIX, protože jim umožňuje vstupovat sítí do vzdálených uzlů a opravovat nebo měnit jejich nastavení a funkce. Neméně známá je později a za stejným účelem konstruovaná síťová služba `rlogin` univerzitních operačních systémů UNIX BSD (Berkeley System Distribution), která je součástí jejich varianty implementace sítí v UNIXu pod názvem Berkeley Sockets. `rlogin` je jeden ze skupiny *r-příkazů* (r-commands) této sítě. Písmeno r je odvozeno od slova remote, vzdálený.

Uvedenou kresbu II-22 můžeme pro `telnet` nebo `rlogin` zjednodušit podle kresby II-23.



Kresba II-23: Síťová služba `telnet`.

Uživatel pracuje přímo v okně klienta `telnet`, který komunikuje se serverem `telnetd` (a jako daemon). Ten zprostředkuje využití terminálového řádkového přístupu s programem, který je na kresbě označen jako `shell`, což je obecný název pro programy řádkové (textové) komunikace uživatele s počítačem (shellem je v UNIXU např. Bourne Shell, KornShell, C-shell a řada dalších). I uživatel PC v operačních systémech firmy Microsoft má klienta `telnet` standardně k dispozici. Zde po stisku tlačítka `Start` vybereme možnost `Spustit...` a do kolony s označením `Otevřít:` napíšeme např.

```
telnet animal.ffa.vutbr.cz
```

což znamená, že požadujeme práci v uzlu s označením `animal.ffa.vutbr.cz`. Po rozběhu klienta se nám otevře okno a klient se pokouší navázat spojení se serverem na určeném uzlu. Pokud máme síťové spojení v pořádku, v okně klienta se pravděpodobně objeví:

```
animal.ffa.vutbr.cz
```

```
login:
```

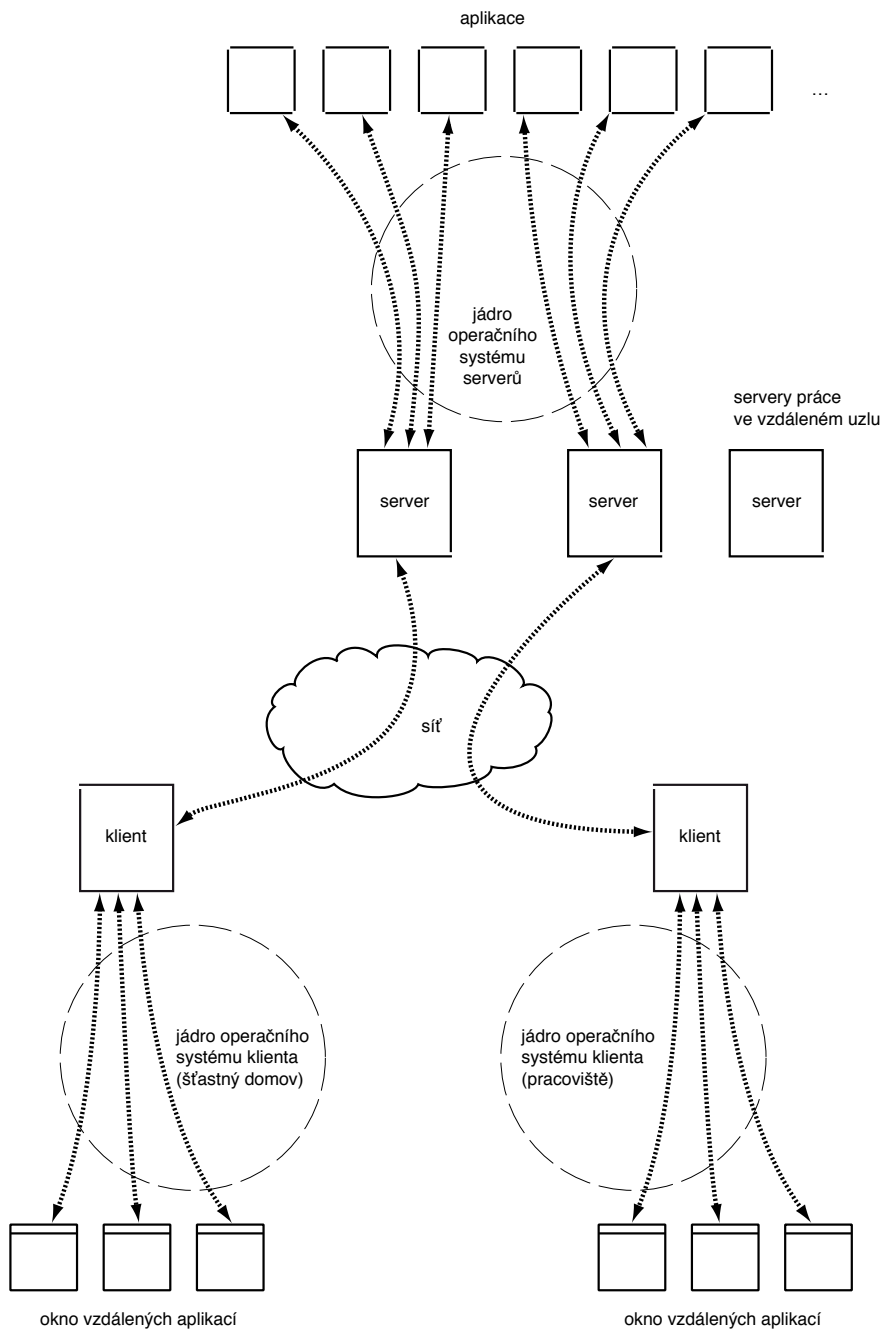
a server `telnetd` očekává zadání vstupního jména uživatele (následovaného stiskem klávesy `Enter`, jak je v řádkovém režimu zvykem). Následně požaduje heslo (server vypíše text `password:`) pro prověření oprávnění vstupu do vzdáleného uzlu.

S příchodem grafického systému oken jako běžného způsobu komunikace uživatele s počítačem vznikla v operačním systému UNIX podpora práce ve vzdáleném uzlu na bázi grafického rozhraní. Tento produkt pracuje již

podle schématu kresby II-22 a jedná se o jeden z principů již zmiňovaného grafického podsystému X Window System (viz [XWindow1993]). Uživatel využívá aplikace ve vzdálených uzlech sítě. Ovládá je přitom v prostředí svého počítače, protože výstup aplikace je vykreslován do jednoho z oken jeho pracovního prostředí. Dokonce i uživatel světa PC může mít takovou podporu práce ve vzdálených uzlech s operačním systémem UNIX. Součástí operačních systémů firmy Microsoft ale není standardně klient, který by takovou práci zajistil. Potřebný klient nese obecné jméno x-server (paradoxně) a lze si jej instalovat dodatečně jako produkt dodávaný třetí stranou. Za tím účelem existuje řada volně šiřitelných i zpoplatněných software nabízených v Internetu, např. Cygwin/X, Exceed, XVision, X-Win32, WeirdX. Na platformě počítačů firmy Apple je od verze operačního systému Mac OS X 10.3 implementován standardně nejenom x-server, ale i úplné prostředí strany síťového uzlu, který vzdálenou práci umožňuje.

Ve světě serverů operačních systémů firmy Microsoft umožňujících serverový provoz (MS Windows 2000, MS Windows XP) lze pro podobnou práci použít např. produkt firmy Citrix Systems s názvem Citrix MetaFrame, který zajistí přenos grafických informací ke klientu tak, že je okno běžící aplikace serveru zobrazováno vzdálenému uživateli. Princip způsobu chování je v posledních verzích podobný principu X Window System. Není však součástí operačního systému a ani firmou Microsoft není nijak výrazně podporován.

K tomu, abychom mohli používat službu práce ve vzdáleném uzlu, je zapotřebí trvalý provoz vzdáleného uzlu. Jinými slovy, je nutné, abychom při odchodu z práce domů počítač na pracovišti nevypínali, pokud jej chceme využívat na dálku z domova. Dále je nutné, aby v tomto uzlu byl provozován operační systém, který dokáže takovou službu zajistit, což, jak vyplývá z předchozího textu, není u většiny používaných osobních počítačů možné. Pracovní prostředí uživatelů, kteří se přemisťují v různých přípojných bodech počítačové sítě, se řeší centralizovanou správou aplikací a dat těchto uživatelů. Za takovým účelem je provozně nasazen uzel v síti s dostatečným výkonem, který nabízí především možnost vzdálené práce. Uživatelé jsou pak vždy registrováni jako vzdálení uživatelé připojící se v identifikaci svým určeným jménem a heslem, připojení na pracovišti nebo doma je pak shodné. Uživatel si v tomto případě nemusí zajišťovat vzájemnou provozní dostupnost jednotlivých počítačů (jejich nevypínáním). Současně je za vedení nepřetržitého provozu zodpovědný pouze správce centrálního uzlu, přičemž tento centrální uzel může sloužit (serverovat) třeba i několika desítkám uživatelů současně. Situace je znázorněna na kresbě II-24.



Kresba II-24: Uzel v síti pro podporu práce uživatelů vždy v tomtéž prostředí.

Uživatel tak získává úplnou shodu svých dat, ať již usedá ke koncovému počítači s přijatelnou konektivitou dnes už kdekoli na světě. Datové a aplikačně

se jedná o úplnou shodu a uživatel přestává být uživatelem počítače a stává se uživatelem počítačové sítě.

sdílení periferií Uvážíme-li princip možnosti přístupu k datům na disku vzdáleného počítače prostřednictvím serveru FTP kdekoliv v síti, jedná se vlastně pouze o speciální případ obecné práce se vzdáleným výpočetním zdrojem.

výpočetní zdroj Termínem *výpočetní zdroj* (computer resource) nazýváme datově zpracovatelný podsystém elementů potřebný pro zajištění určité operace. Tato suchopárná technická definice znamená, že výpočetním zdrojem je např. každá periferie počítače, ale také např. ucelená část operační nebo diskové paměti, samostatně se chovající probíhající software atp. Jinými slovy se zde otiskuje základní pojetí Computer Science jako vědy definované na matematické teorii, která se snaží definovat počítač a výpočet jak celek složený z částí a celek definovat pomocí potřebných a případně nepotřebných částí. Částí pak je onen výpočetní zdroj.

Výpočetním zdrojem tedy může být obecně skutečně cokoli, co souvisí s výpočtem. Mohou to být data jako část disku nebo databáze či její část, tiskárna, CD, archivní magnetická páska, zálohovací systémy, probíhající proces, část operační paměti nebo i CPU.

Pro využití výpočetního zdroje v síti se tedy jedná o poskytnutí (serverování) určitého výpočetního zdroje k jeho využití jako části jiného celku (počítačová síť se dnes skládá z navzájem propojených samostatných celků).

FIFO Slétneme-li z nebe zpátky na pevnou zem technické skutečnosti a současně vrátíme-li se k termínu sdílení periferií, zajímá nás zde především možnost nabízet do sítě část hardware, který je obsluhován určitým softwarem tak, aby se zájemci o tuto periferii vzájemně nepoprali. To znamená, že např. budu-li tisknout na tiskárně nabízené v síti a tutéž službu bude na téže tiskárně požadovat i jiný uživatel sítě, já i on bychom měli neradi své tisky vzájemně promíchané. Součástí síťové služby tedy není pouze klient, který zprostředkuje spojení se vzdálenou periferií, ale zde např. také obsluha tiskové fronty, což je intelligence, která zajistí tisk požadavků postupně všech uživatelů, a to obvykle metodou kdo dřív přijde, ten dřív mele (čili první dovnitř, první ven, angl. FIFO, first in first out). Tisky jsou souvisle prováděny v tom pořadí, v jakém k tiskárně přicházejí.

Sdílení každé periferie musí disponovat částí, která ochrání uživatele vzájemně mezi sebou. Primitivní forma, kterou je zajištěn uvedený vzdálený tisk, se ale nehodí u sdílení periferie např. diskové. Poskytujeme-li veřejný diskový prostor obecně několika uživatelům najednou, nemůžeme v případě požadavku přístupu k periferii na základě požadavku jednoho uživatele postavit do fronty všechny ty, kteří přišli později s tím, že je postupně všechny obslužíme. Každý uživatel požaduje obvykle jinou část téže diskové periferie, ale disková periferie je obsluhována jedním operačním systémem, jedním modulem přístupu k diskům (file system) a tedy v dané chvíli musí docházet k multiplexované obsluze takové periferie. Tento násobný přístup k jedné periferii jsme si ale vysvětlili již v části Base Of jako nutnou součást operačního systému, který slouží více uživatelům (více probíhajícím

programům, procesům) současně. U sdílení diskového prostoru sítí proto např. při kopírování téhož velkého objemu dat různými síťovými klienty nemůžeme po dobu přenosu pro jednoho z nich zablokovat síťový požadavek ostatních tím, že je postavíme do fronty. Přenos dat musí být rozprostřen mezi všechny klienty tak, že se síťová služba postupně mezi nimi přepíná a zdánlivě tak v čase probíhá přenos současně pro všechny. Skutečná kolize při sdílení disků nastane v okamžiku, kdy se současně nejméně dva klienty pokouší zapisovat data do téhož místa, tedy např. rozšiřovat soubor téhož jména. Mají-li oba oprávnění takového zápisu, pak obsah souboru, jak teoretici suše praví, nelze zajistit. Pokud ale operační systém pracuje správně, soubor nebude nijak poškozen, pouze jeho obsah bude podivně promícháný daty různých uživatelů.

Znamená to, že souběh uživatelů nad diskovým prostorem musí být chápán jako souběh nad např. souborem jako výpočetním zdrojem nebo dokonce nad částí souboru jako výpočetním zdrojem. Operační systém a systém procesů pro obsluhu sdílených periférií používá pro výhradní přístup k výpočetnímu zdroji zamykání (locking). Zdánlivě elementární prvek se po úvaze stává oříškem hodným matematické teorie. Musíme si totiž uvědomit, že zamkne-li někdo (nebo něco) výpočetní zdroj, musí tomu tak být nikoli na nekonečnou dobu. Ale odemknout by jej měl také ten, kdo zdroj zamknul a už jej nepotřebuje. Ale co když zapomněl? Nebo neočekávaně skončil a nestačil odemknout? Jak ale systémově zjistit a rozpoznat, že je zdroj již zamknut zbytečně? Systém zámků tedy musí podléhat obecnému schématu, které je ale natolik flexibilní, že dokáže nabízet např. vypršení platnosti (timeout) zámků proměnlivě. Rovněž požadavek na přístup k zamknutému zdroji musí mít možnosti varianty buďto čekání (zařazení se do fronty) a nebo odložení pokusu na později. Také je nutno řešit uváznutí (deadlock), kdy procesy drží zdroje a současně se ucházejí o další zdroje, ale vzájemně zamknuté. A tak podobně. Obecný způsob zamykání výpočetních zdrojů řeší v Computer Science semafor (semaphores). Jedná se o problém návrhu přístupu konstruktérů operačních systémů a sítí ke sdíleným výpočetním zdrojům. Obyčejný uživatel může pouze požadovat jejich spolehlivost, z hlediska využívání vzdálených periférií jej ale nemůže ovlivnit (pouze nadávkami správci systému a sítě).

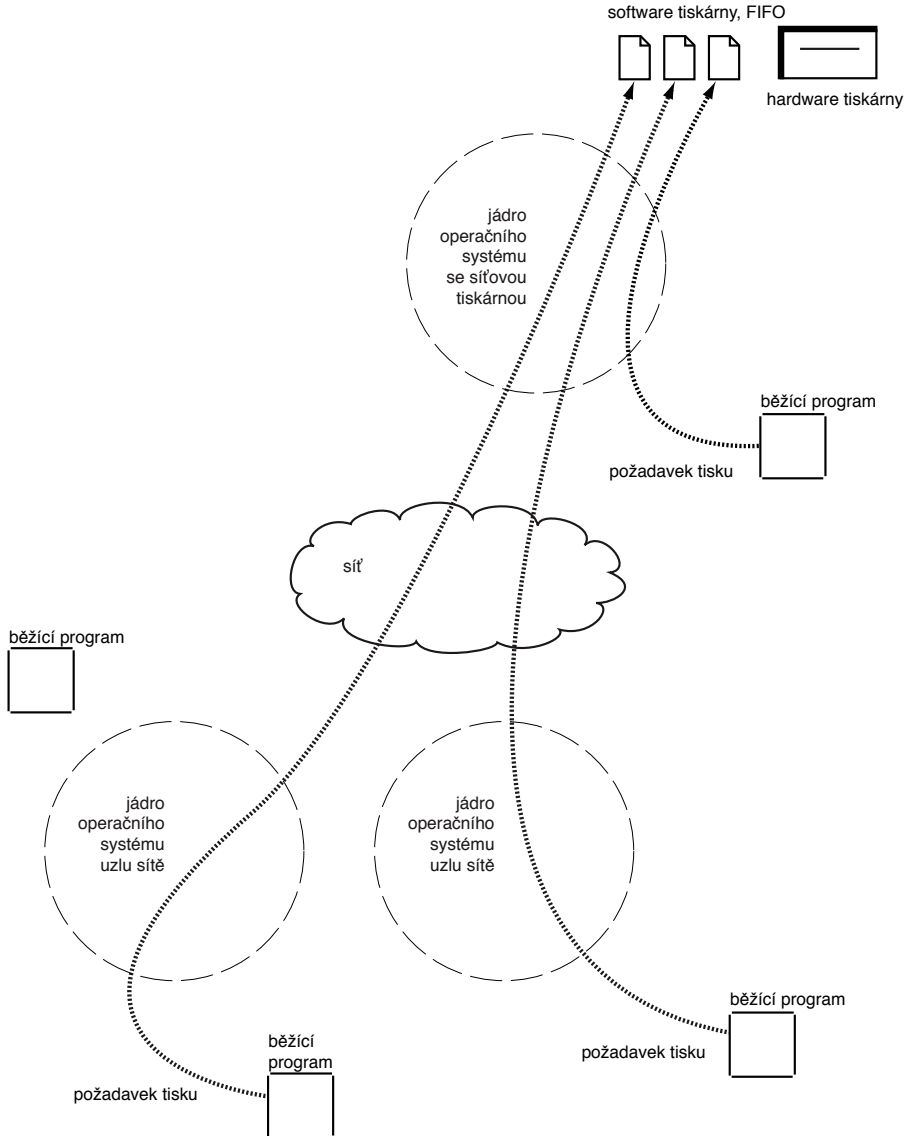
**zamykání
výpočetního
zdroje**

V běžné uživatelské praxi se však většinou jedná o typizované periferie, které jsou sdíleny, zejména, jak už bylo uvedeno, o tisk a diskový prostor. Obecně lze sdílet cokoliv, ale mnohdy se jedná o periferie, které vyžadují přítomnost člověka, jako je např. skener nebo vypalovací mechanika CD, disketová mechanika nebo magnetopásková jednotka. Fyzická přítomnost člověka u periferie podmiňuje také výhradní přístup jednoho uživatele v průběhu delšího časového úseku, který se většinou pohybuje v desítkách minut. To pak obvykle v organizaci práce lidí vlastně vylučuje poskytování takových periférií sítí automatizovaně.

Vzdálená tiskárna je periferie, která je připojena k jiném počítači než je náš a je nám přístupná prostřednictvím sítě. Každá tiskárna, ať už vzdálená

vzdálená tiskárna

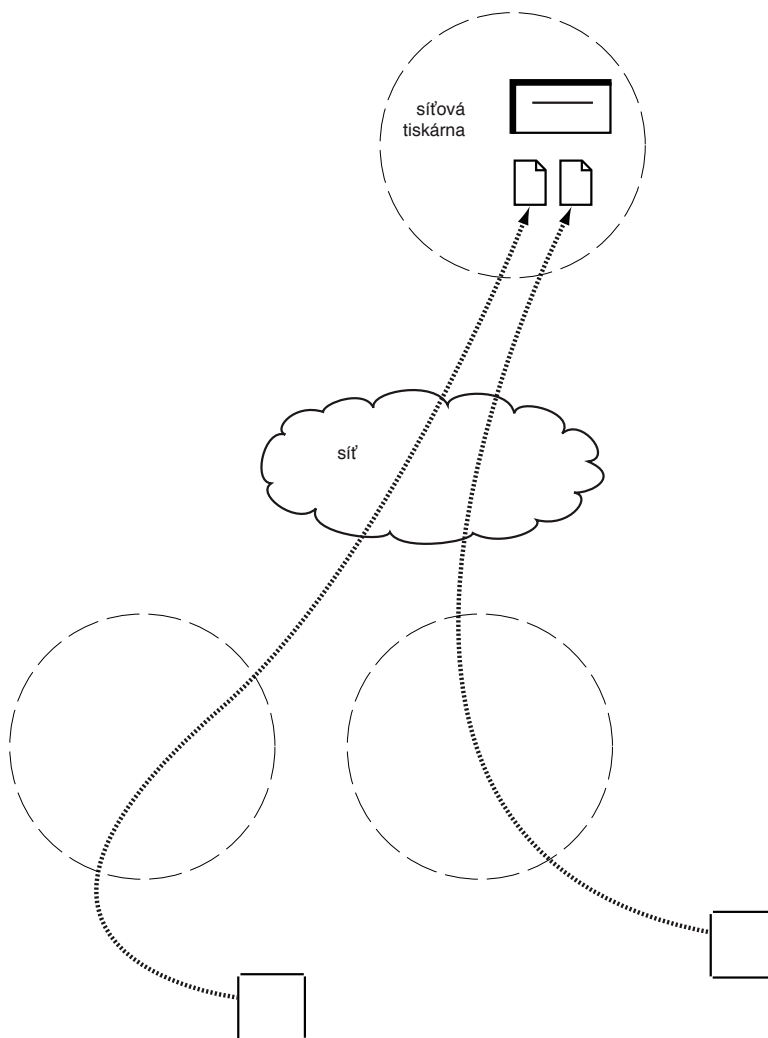
nebo místní, je obsluhována pomocí software, který je součástí operačního systému a který zajišťuje jednotlivé tisky prostřednictvím již zmiňované fronty FIFO. Tento software přitom přijímá požadavky na tisk od vlastně libovolného běžícího programu téhož počítače. Je-li tiskárna zveřejněna jako výpočetní zdroj do sítě, její obslužný software tyto požadavky přijímá i od programů, které běží na připojených počítačích. Situaci vidíme na kresbě II-25.



Kresba II-25: Síťová tiskárna.

Uvedené schéma se ale upotřebí v tom případě, že v síti disponujeme uzlem, který je neustále v chodu a k němu připojená tiskárna je fyzicky

přístupná lidem, kteří na ní tisknou. Většinou je takovým místem chodba na pracovišti, kopírovací centrum atp. Dále musí být operační systém uzlu s tiskárnou provozně spolehlivý. Tímto způsobem sdílené tiskárny se proto provozují především v kombinaci se serverováním dalších síťových služeb (např. diskový prostor) v operačních systémech UNIX. Připojení a nastavení sdílené tiskárny (nebo několika sdílených tiskáren) v UNIXu ale není jednoduché a pro kanceláře s několika počítači v síti s MS Windows je to řešení zbytečně náročné. Proto se již řadu let vyrábějí tiskárny, které se připojují jako samostatný uzel sítě. Jedná se vlastně o tiskárnu, která je vybavena síťovou periferií, potřebným komunikačním software a software obsluhy fronty příchozích požadavků na tisk. Podívejme se na kresbu II-26.



Kresba II-26: Síťová tiskárna jako samostatný uzel sítě.

Pro počítače v síti zůstává situace stejná jako v prvním případě. Tiskárna je viditelná jako součást vzdáleného uzlu, uzel je ale součástí hardware tiskárny. Samostatné síťové tiskárny jsou dodávány se software, který rozumí různým typům sítí (v síti IP např. přidělujeme tiskárně vlastní adresu IP). Tiskárna komunikační částí přijímá pakety třeba i různých sítí současně, skládá je do příchozích požadavků na tisk a předává části software pro obsluhu fronty na tisk. Vzhledem k tomu, že se jedná o specializovaný hardware společně se software, který je produktem výrobce tiskárny, je to řešení mimořádně spolehlivé. Přitom jediné, co je nutno mít elektricky zapnuto, je samotná tiskárna.

Pro tiskárny, jejichž hardware je konstruován pouze jako periferie počítače (pro místní tisk), lze síťový hardware pro osamostatnění tiskárny dokoupit, a to jako aktivní síťový prvek, který je připojen jednak na síť a jednak na tiskárnu. Obvykle umožňuje takový tiskový server připojení i několika tiskáren současně. Řešení má vyšší flexibilitu, protože jako síťovou tiskárnu můžete použít libovolnou tiskárnu, která nemusí být součástí trvale zapnutého počítače, který je typu PC a který se chystá jeho uživatel třeba právě restartovat.

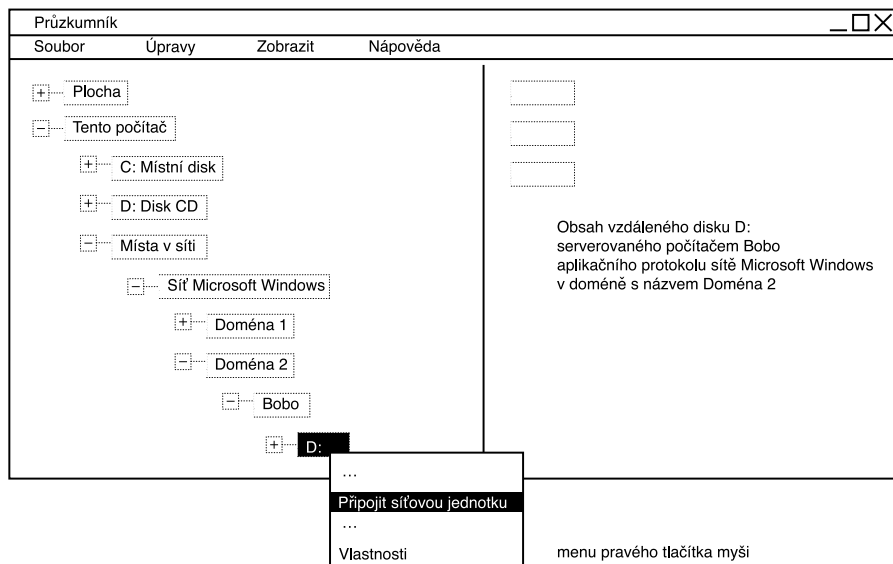
vzdálené disky

Dalším velmi rozšířeným příkladem sdílení periferií je sdílení disků (disk sharing). V termínu tkví jistá nepřesnost, protože tato síťová služba umožňuje do sítě poskytovat nikoliv nutně obsah celých disků, ale pouze jejich části, které jsou dány např. podstromem v hierarchii adresářů (viz část Base Of I.4). Správný výraz by tedy měl být sdílení souborů (file sharing), což je skutečně termín správný a používaný, ale na komolení odborných výrazů a zanášení neoborných nepřesností vysokou účastí neoborníků jsme si již zvykli.

Principem sdílení disků je opět nabídka částí disků do sítě jako části serverové. Klient využívá poskytovaných částí disků a zařazuje je pro uživatele mezi místní disky. Sdílení disků je transparentní, tedy nezávislé na operačním systému, pokud se různé operační systémy domluví na aplikačním protokolu, jak my víme. V praxi jsme ale zvyklí, že operační systémy ukazují uživatelům diskový prostor různým způsobem (různými protokoly). Např. MS Windows zobrazuje vzdálený diskový prostor jako adresáře vzdálených počítačů (viz program Průzkumník) a umožňuje označit si tyto adresáře jako samostatné disky, na což se zde používá obvykle termín připojit síťovou jednotku. Připojená disková jednotka se tak označuje některým dalším písmenem abecedy a při práci se vzdáleným diskovým prostorem uživatel používá písmen s dvojtečkou pro přístup k souborům ve vzdáleném počítači, např.

X:\honza\dokument.doc

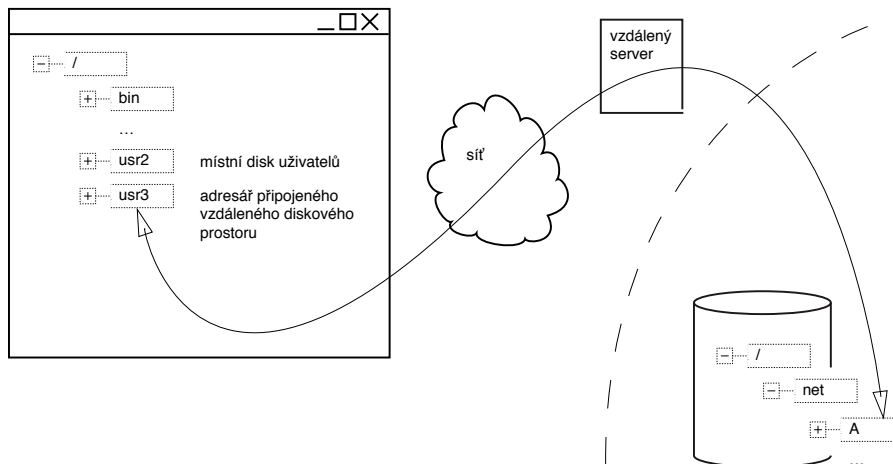
je odkaz na diskovou jednotku x:, která byla připojena způsobem zobrazeným na kresbě II-27



Kresba II-27: Připojení vzdáleného diskového prostoru v MS Windows.

a je tedy také viditelná v části Tento počítač jako **D:** na disku **Bobo (X:)**.

V typech operačních systémů, kde je diskový prostor uživateli zobrazován jako jednolitá plocha (typicky UNIX) začínající jedním adresářem (/, viz část Base Of I.4), se vzdálený diskový prostor připojuje na některý z volných adresářů. V UNIXU se za tímto účelem používá příkaz `mount`, který může provádět pouze správce operačního systému, ale ten jej může integrovat do části, která určuje pracovní prostředí počítače s klientem diskové síťové služby, takže se `mount` automatizovaně provádí při startu počítače a nastavuje tak uživatelům vzdálený diskový prostor. Vstupem do odpovídajícího adresáře pak uživatelé pracují se vzdáleným diskovým prostorem, jak ukazuje kresba II-28.



Kresba II-28: Práce se vzdáleným diskovým prostorem v UNIXu.

Síťové služby předváděné na kresbách pracují různými aplikačními protokoly, třebaže se jedná o jednu a tutéž síť (např. typu IP). V MS Windows je to protokol s označením Síť Microsoft Windows a v UNIXu je běžně používaný protokol s označením NFS (Network File System). Pokud by se uživatel počítačů v MS Windows rád připojil k serveru typu NFS, potřeboval by k tomu klienta do tohoto operačního systému. Klient NFS se pro MS Windows vyrábí, ale není součástí operačního systému a kvalitně jej obvykle podporují opět výrobci z třetí strany. Ani opačné přizpůsobení se serveru MS Windows pro mount UNIXu nebylo nikdy úspěšně realizováno. To vše pravděpodobně z důvodů obchodní konkurence. Produkt z třetí strany s názvem Samba (viz www.samba.org) je software pracující v operačním systému UNIX (velké škály různých výrobců), který emuluje server disků sítě MS Windows. Je velmi používaný, protože se jedná o volně šiřitelný software, který umožňuje kombinaci spolehlivého operačního systému na straně síťového serveru (UNIX) pro osobní používání vzdáleného diskového prostoru v MS Windows (a nejen v nich), aniž by bylo potřeba jakkoliv stávající instalaci MS Windows doplňovat. Pro připojení se totiž používá klient aplikačního protokolu Síť Microsoft Windows.

Podmínkou práce klientů je přítomnost serveru na druhém konci síťového spojení. Pokud pracujeme jako klient a dojde k zastavení serveru, uživatelsky se jedná o totéž, jako když při práci na disketě vyjmete disketu z mechaniky. Dojde ke ztrátě vzdáleného diskového prostoru. Podobně jako při opětovném vložení diskety, i zde po obnovení spojení lze pokračovat v práci, protože síťové protokoly se vzájemně dohodou. To platí v případě, že byl výpadek spojení způsoben např. chvilkovým odpojením propojovacího kabelu počítačové sítě. V případě havárie počítače se serverem je situace o něco dramatičtější, protože po novém startu počítače se vzdáleným diskovým prostorem jsou protokoly ve výchozím stavu a data před havárií nemusí být uložena na vzdáleném disku všechna a konzistentně (soubor, který jsme editovali, může mít poškozenou vnitřní strukturu).

bezdiskové pracovní stanice

Správce sítě je zodpovědný za topologii propojení počítačů, jejich zařazení do různých sítí, určení síťových periférií, klientů, které je budou využívat a zajištění dostupnosti síťových periférií těmto klientům. Provoz sítě je tedy podmíněn nepřetržitým chodem serverů, které zajišťují sdílení periférií. U diskového prostoru je dokonce možnost (a průmyslově používaná) provozu bezdiskových stanic (diskless machines). Myšlenka vychází ze silného serveru diskového prostoru s nepřetržitým provozem. Operační systém s klientem pak nemá žádný místní disk. Po zapnutí bezdiskového počítače je proveden dotaz na vzdálený server, který poskytne klientovi operační systém sítí (obvykle aplikačním protokolem TFTP – Trivial FTP, jednodušší obdoba FTP). Po zavedení takto získaného operačního systému ze serveru se rozbíhá jeho jádro nad nejprve připojeným vzdáleným diskem (již např. protokolem NFS). Uživatel se přihlašuje do operačního systému a pracuje pak pouze se vzdáleným diskovým prostorem. V kombinaci s dalšími síťovými perifériemi (tiskárnou) se tedy bezdisková stanice z pohledu hardware redukuje na

kvalitní obrazovku, klávesnici, myš, procesor s operační pamětí a síťovou periferii. Vše ostatní je poskytováno sítí.

Pomineme-li elegantní řešení síťového provozu s bezdiskovými počítači, je sdílení diskového prostoru určeno k rozšíření pracovního prostoru uživatelů a dále pro výměnu dat v souborech mezi různými uživateli různých operačních systémů. Je možné je používat i pro zpřístupnění určitých aplikací. Jejich instalace se provádí na serveru a uživatelé v klientských operačních systémech je pouze využívají. Šetří se tak diskový prostor tolikrát, kolik vzdálených stanic instalaci používá. Jedná se ovšem o podporu síťového provozu na úrovni výpočetních zdrojů, nikoliv tedy na úrovni sdílení aplikací. Sdílením aplikací zde rozumíme využívání přístupu k organizované struktuře provozních dat, a tou je databáze. Každý majitel firmy nebo vedoucí nějaké organizace po určité době vedení agendy v souborech (faktury, dopisy, adresáře) pocítí potřebu nikoliv neustále prohledávat soubory dat sdílených disků, ale přehledně pracovat se specializovaným software určeným pro podporu potřeb jeho organizace. Takový software nazýváme informační systém organizace a jeho základem je ukládání dat prostřednictvím databáze. Jedná se o systém ukládání dat, který řeší řadu úskalí současné práce s daty mnoha uživatelů se stejnými nebo podobnými zájmy. Informačními systémy a databázemi se budeme zabývat v části Enjoy It III.3. Jako příklad jsme si ale přitom takovou aplikaci již strukturálně uvedli v části Base Of I.2 na kresbě I-11.

Sdílení periferií je fenomén počítačových sítí, jehož dopad na další vývoj Computer Science a její praxe je výraznější než by si kdo myslel.

S vědomím možností sdílení vzdálených periferií nebo lépe vzdálených výpočetních zdrojů totiž začíná ta pravá legrace, kterou dnešní sítě nabízejí. Principem i zde zůstává poskytování výpočetního zdroje serverem, který je dotazován klientem. Obecně si ale můžeme představit např. využití výpočetního výkonu vzdáleného počítače pro zpracování části výpočtu (distribuovaný výpočetní výkon), tedy sdílení CPU, části operační paměti, sdílení paměťových kanálů atd.

síťové operační
systémy

Vnucuje se tedy myšlenka, zda je možné obecně všechny výpočetní zdroje sítě chápat samostatně a slučovat je do celku, který nerozlišuje uzly v síti jako počítače. Jedná se o partii Computer Science, které říkáme síťové operační systémy.

Síť lze chápat jako systém vzájemně propojených nezávislých celků, kdy každý celek je tvořen částmi nazývanými výpočetní zdroje. Lze ale také síť chápat jako systém takových výpočetních zdrojů, které jsou v ní k dispozici a kdy je nutno vyřešit jejich poskytování a využívání tak, že vzniká jeden výpočetní celek. Z pohledu technického lze tedy dispozičně stanovit k užívání např.: 15 procesorů, 15 operačních pamětí, 23 monitorů, 20 klávesnic, 15 myší, 2 skenery, 3 tiskárny, 52 disků, 1 magnetopáskovou mechaniku, 3 přepisovatelná CD, z toho jedno DVD atd., atd. Malé jádro síťového operačního systému je přitom přítomno a prováděno v každé CPU takové množiny výpočetních zdrojů. Pomocí komunikačních protokolů nabízí

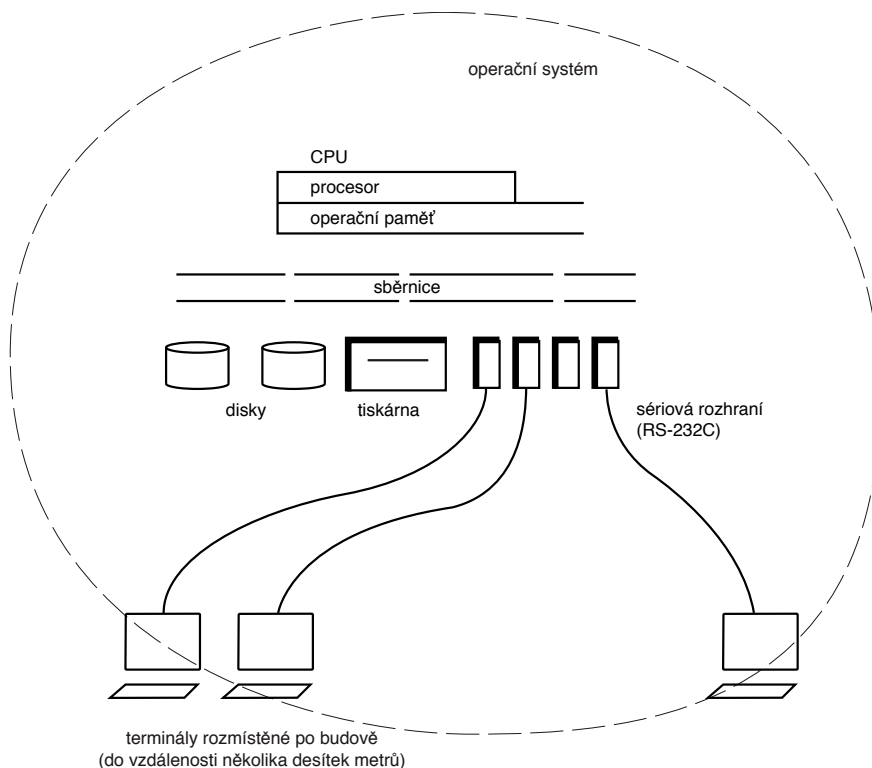
uživatelům přístup ke každému výpočetnímu zdroji, který je protokolem dostupný. Vzniká tak distribuovaný síťový operační systém, který uchopí výpočetní zdroje sítě jako celek. Uživatel se stává uživatelem počítačové sítě. Předmětem zájmu pak začíná být výkon nikoliv místního počítače (a vlastně pouze CPU), ale celkový výkon celé sítě, tedy především síťového spojení. Výkon sítě je poměřován a dimenzován počtem současně pracujících uživatelů.

rozhovor, pošta

Propojování počítačů vzájemně mezi sebou do počítačových sítí umožňuje uživatelům využívat výpočetní zdroje celé sítě a zprostředkovat jim tak tentýž datový a výpočetní prostor nezávisle na tom, kde se uživatelé právě fyzicky nacházejí. Člověk ale i jako uživatel počítačové sítě není ve své jedinečnosti osamocený. Současně s ním využívá zdroje sítě bezpočet dalších jemu na roveň postavených uživatelů. Již několikrát jsme se v textu přiblížili problematice sdílení dat více uživateli současně. Mnohdy to přitom není náhodné setkání pro výlučný přístup, ale je koncepčně definováno jako řízená spolupráce. Typicky se jedná o např. využívání centrálního skladu zásob při prodeji zboží na území celého státu nebo rezervace a prodej letenek různých leteckých společností po celém světě. Uživatelé se zde setkávají nad daty a svou přítomnost vzájemně pocítují jako vedlejší průvodní jev.

Jedním z dalších využití vzájemně propojených výpočetních zdrojů je přímá mezilidská komunikace.

Využívání telefonů je dnes běžná praxe. Přímý rozhovor prostřednictvím výpočetní techniky je aplikace, která je známá již prakticky od začátku využívání počítačů. V době práce na operačním systému UNIX v 70. letech 20. století, tedy v době, kdy sítě existovaly v základních konceptech a výzkumných laboratořích, byla naprogramována a využívána aplikace *write*, která umožňovala uživateli psát text ze své klávesnice na obrazovku uživateli jinému. V té době se počítače používaly v režimu *sdílení výpočetního času* (time sharing), tj. k jednomu počítači bylo připojeno více obrazovek a klávesnic, které byly rozmístěny po budově pro interaktivní práci uživatelů tak, jak ukazuje kresba II-29.



Kresba II-29: Interaktivní práce více uživatelů na tomtéž počítači.

Uživatelé se prostřednictvím klávesnice přihlašovali ke stroji a byli tak registrováni operačním systémem. Jejich práce nutně vyžadovala vzájemné sdílení periferií, ale i operační paměti a neustálé střídání práce CPU tak, aby vznikl pro uživatele dojem, že pracují bez přerušení. Po dobu, kdy uživatel zapisoval svůj požadavek na klávesnici nebo uvažoval nad dalším postupem práce, se CPU pod řízením operačního systému věnovala jiným uživatelům.

Program `write` propojoval dvě klávesnice a dvě obrazovky přihlášených uživatelů proti sobě tak, že si uživatelé mohli přímo vyměňovat psaný text. Komunikace byla textová a řádková, např.

```
$ write ludek
ahoj, večer v sedm
U průmyslovky
^d
$
```

je příklad, kdy přihlášený uživatel píše uživateli se jménem `ludek` text na dalších řádcích. Znak `$` je reakce operačního systému, kombinací `^d` (Ctrl-d) uživatel oznamuje konec odesílaného textu. Přihlášený uživatel `ludek` obdrží na obrazovce tuto zprávu:

```

Message from bohous on tty01 at 10:07 ...
ahoj, večer v sedm
U průmyslovky

```

a může odpovídat:

```

$ write bohous
O.K., přivedu T.
^d
$

```

Jeho odpověď se uživateli se jménem `bohous` objeví na obrazovce takto:

```

Message from ludek on tty02 at 10:09 ...
O.K., přivedu T.

```

Program `write` má dodnes každý UNIX a programátoři a ostatní specialisté jej používají. Jeho použití je možné i prostřednictvím sítě, typicky za využití služby přihlášení ve vzdáleném uzlu aplikací `telnet`. Ve vzdáleném uzlu tak lze hovořit s ostatními přihlášenými uživateli.

Později vzniklá aplikace `talk` má tentýž účel a je populárnější, protože při startu rozdělí obrazovku na dvě části, v horní píšeme text a v dolní části se objevuje text psaný uživatelem na druhém konci spojení.

chat Programy textového rozhovoru `write` a `talk` předznamenaly další vývoj komunikace lidí prostřednictvím počítačů, dnes např. s označením *chat*, kdy se psaný text jednoho připojeného uživatele současně zobrazuje několika (někdy i mnoha) ostatním. I telefonický rozhovor počítačovou sítí, ať už fonický nebo dokonce vizuální, je jen otázkou kvality a rychlosti spojení, specializovaných periférií a komfortu spuštěné aplikace, která mnohdy velmi závisí na výkonu místní pracovní stanice. Vizuální chat je nazýván videokonferencí.

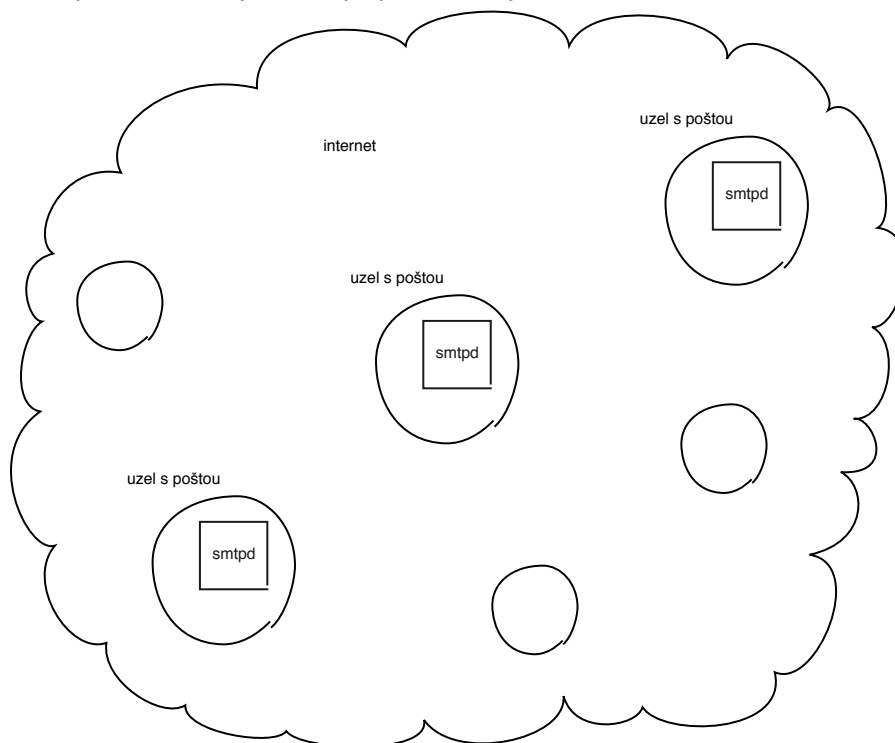
Na rozdíl od uvedeného rozhovoru dvou lidí se vzájemná přímá komunikace několika lidí současně implementuje za použití vzdáleného serveru, se kterým komunikuje každý účastník konference. Server oznamuje uživateli, kdo je právě také připojen a zprostředkuje mezi ně distribuci snímaných dat (psaný text, zvuk zaznamenaný mikrofonem nebo obraz kamerou). Takto pracuje např. ICQ (I seek you) nebo Skype (viz www.icq.com, www.skype.com).

Rozhovor dvou lidí vyžaduje vždy jejich mentální přítomnost, a to ať už jsou kdekoli na světě. Jinou formou komunikace uživatelů mezi sebou je zanechávání zpráv, které si mohou lidé číst později a věnovat se souvislé činnosti bez nutného vyrušení vyžádaným rozhovorem. Zanechávání zpráv formou digitálních údajů má svoji analogii v klasické papírové poště, proto tato počítačová aplikace dostala název *elektronická pošta* (electronic mail).

elektronická pošta

Elektronická pošta prožila největší rozkvět v průběhu vzniku Internetu jako celosvětové sítě. Všude, kam tato síť sahá, si mohou uživatelé jejím prostřednictvím zanechávat zprávy ve svých poštovních schránkách (mail box). Síťová aplikace elektronické pošty se opírá o vzájemnou komunikaci serverů v různých uzlech sítě, které si vyměňují poštu. Tyto servery se často také nazývají agenty transportu. Nejznámějším je server se jménem `sendmail`, jehož původ je v UNIXu typu BSD.

V síti v uzlech nepřetržitého provozu jsou instalovány servery sendmail tak, aby byly schopny vzájemně si vyměňovat poštu na základě požadavků uživatelů. Vzájemně mezi sebou komunikují protokolem s označením SMTP (Simple Mail Transfer Protocol). Program, který se pak za účelem příjmu nebo odeslání pošty aktivuje, je proto v UNIXu pojmenován `smtpd` (SMTP daemon). Běžící programy `smtpd` jsou rozmístěny v síti např. podle kresby II-30. **SMTP**



Kresba II-30: Agenty transportu pošty sendmail – démoni `smtpd`.

Kresba zachycuje několik uzlů internetu, které jsou označeny jako uzly s poštou, což znamená, že na jejich diskovém prostoru leží obsahy poštovních schránek uživatelů. Např. může být provozován uzel s označením `smtp.posta.cz`. Tomuto jménu jistě odpovídá určitá adresa IP. Na takto registrovaném uzlu je instalován program `smtpd`, který dokáže odeslat poštu podle jejího adresáta.

Adresace poštovních schránek v internetu je notoricky známá, je rozdělena na část poštovní domény a část vlastního jména člověka, kterému je pošta posílána. Obě části jsou odděleny znakem @. Např. `ludek@posta.cz` je adresa člověka se jménem `ludek`, jehož poštovní schránka je umístěna v poštovní doméně `posta.cz`.

V případě, že na tuto adresu přichází pošta, nejprve odesílající program `smtpd` dotazuje nejbližší proces DNS (viz předchozí text), aby mu sdělil adresu IP uzlu, na který je nastavena poštovní doména `posta.cz`. Tou může být podle našeho příkladu právě uzel `smtp.posta.cz`. Odesílající program `smtpd`

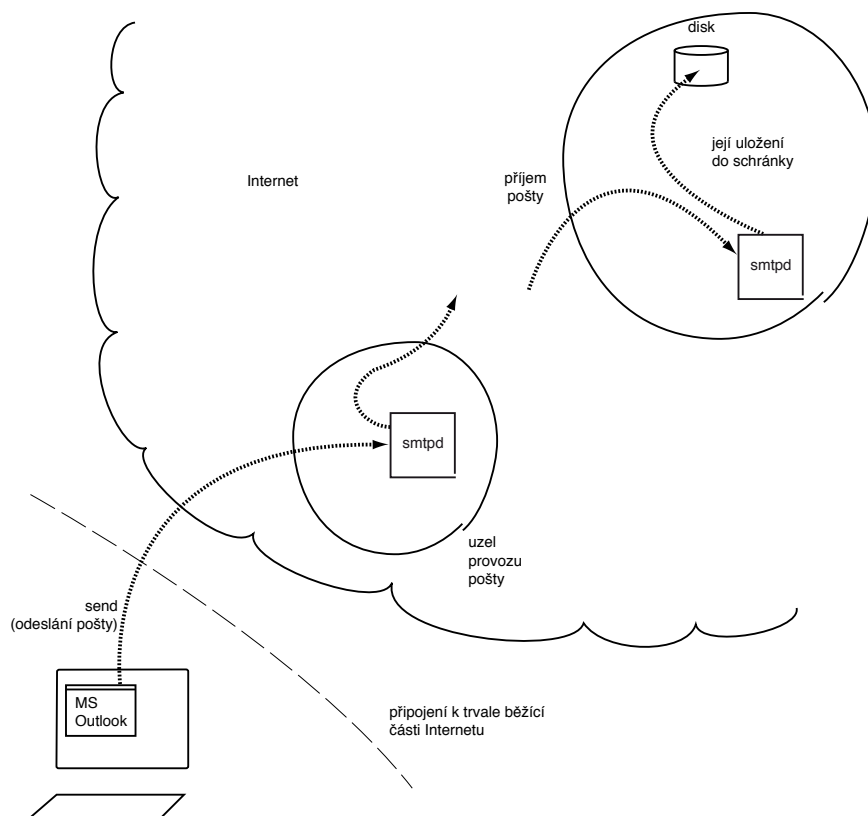
tedy zamíří na tento uzel a naváže spojení se zdejším programem `smtpd`. Ten prozkoumá, zda odpovídající poštovní schránka (se jménem `ludek`) je zde umístěna. Pokud je zde adresát skutečně evidován, poštu přijme a uloží do odpovídající poštovní schránky na disku.

Při odesílání pošty mohou tedy nastat dvě kolize. Jednak se můžeme přepsat ve jménu poštovní domény, což zjišťuje odesílající program u serveru DNS a nám se pošta vrací s poznámkou (host not found, tj. uzel nebyl nalezen) a nebo poštu odmítá `smtpd` na cílovém uzlu s tím, že požadované jméno neexistuje (user not found, uživatel nebyl nalezen). Třetí typ kolize může ještě nastat v případě, že dojde k přílišnému prodlení mezi dotazem na poštovní doménu a existencí schránky v ní. To je případ, kdy je poštovní doména registrována v podsystému DNS, ale uzel, kam tato doména směřuje, neodpovídá (`smtpd` neběží nebo častěji je celý počítač vypnutý nebo odpojený od sítě). Délka snesitelného prodlení je nastavována na odesílajícím `smtpd`, který obvykle po dvou dnech opakovaných pokusů poštu doručit ji vrací zpět s odpovídajícím komentářem.

finger Poštovní seznamy jmen evidovaných v poštovních doménách nejsou zveřejňovány, protože pak podléhají častému zahlcování (spam) reklamou a jinou nevyžádanou poštou. Přestože tento úzus platí od nepaměti, v sítích internet je používána služba s názvem `finger`, která zjišťuje existenci uživatele, majitele poštovní schránky. Znamená to, že pokud si nejsem jist, zda poštovní adresa je registrována, mohou ji prověřit tímto programem. V UNIXu je jeho použití obecně z příkazového řádku, takže např.

```
$ finger ludek@posta.cz
```

je příkazový řádek, na základě kterého se aktivuje klient `finger`, ten naváže spojení se serverem `fingerd` v odpovídajícím poštovním uzlu s poštovní schránkou uživatele a uživatel jej oslovuje jen v okamžiku, kdy odesílá poštu. Podívejme se na kresbu II-31.

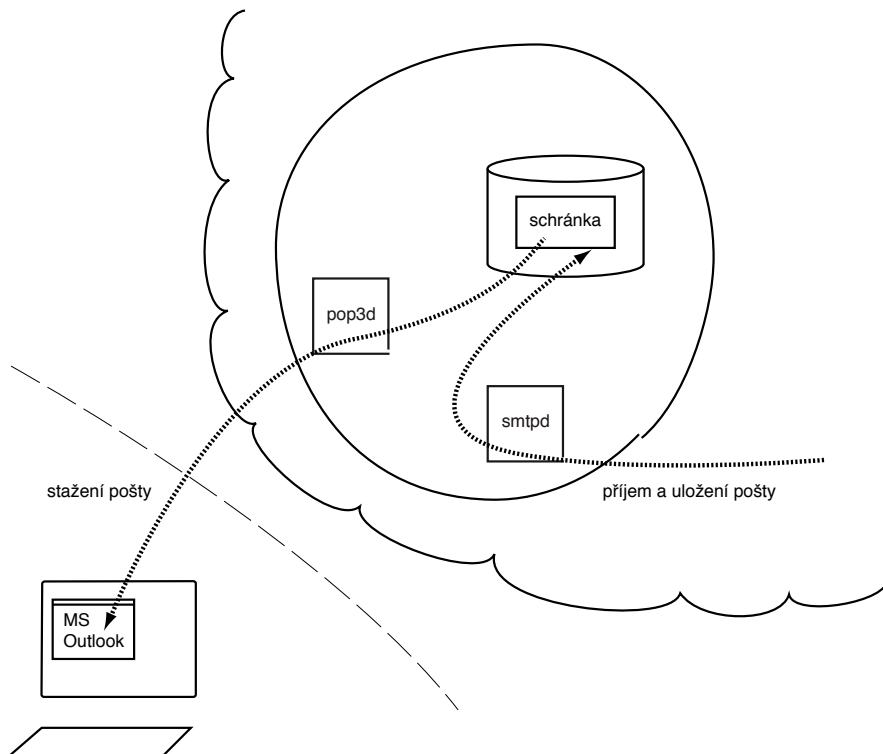


Kresba II-31: smtpd odesílá poštu na pokyn uživatelského klienta.

Uživatel odesílá a přijímá poštu programem jeho místní správy. Tím je např. na PC s MS Windows program MS Outlook. V jeho grafickém prostředí vytváří uživatel poštu a tlačítkem s nápisem *Odeslat* (*Send*) poštu předává programu `smtpd`. K tomu, aby předání pošty proběhlo, ale potřebuje spojení mezi svým počítačem a počítačem se `smtpd`, což je běžné internetové připojení (které ale nemusí být aktivní v okamžiku psaní pošty, ale pouze v okamžiku jejího odesílání). Typicky se jedná o případ připojení k síti vytáčenou telefonní linkou. Uživatel takového způsobu práce s poštou tak může bez připojení k Internetu připravit několik dopisů, poté navázat spojení vytočením odpovídajícího telefonního čísla a v průběhu několika desítek vteřin veškerou poštu odeslat a počítač od telefonu odpojit. Poštu převezme `smtpd`, který dále prověřuje doménu, jméno adresáta a ve spolupráci se `smtpd` cílového poštovního uzlu poštu umísťuje do poštovní schránky adresáta.

Uživatel píše a čte poštu prostřednictvím svého oblíbeného programu na svém počítači, který může mít připojení k Internetu trvalé nebo občasné. Jako příklad jsme uvedli program MS Outlook, ale přestože je pravděpodobně nejpoužívanější, možností je daleko víc (Netscape Messenger, Eudora, Pegasus Mail, Thunderbird, atd., atd.). Obvykle se jedná o programy, **POP**

které jsou zkonstruovány pro různé operační systémy a jsou v Internetu k dispozici zdarma. Čtení pošty takovým programem představuje požadavek navázání spojení s uzlem v Internetu, na kterém je umístěna naše poštovní schránka, kterou `smtpd` nepřetržitě doplňuje příchozí poštou. Aktivita čtení pošty z poštovní schránky a její přenos do programu na PC již nesouvisí s odesíláním a přijímáním pošty. Proto čtení pošty (v žargonu používaný termín stažení pošty, *download the mail*), tedy její přenos ze schránky na disk místního počítače, probíhá jiným způsobem komunikace. Tím je POP (Post Office Protocol), poštovní protokol, který je v současné době v 3. verzi označován jako POP3. Každý program uživatele k přístupu k poště ovládá jak protokol SMTP, tak protokol POP3. Pomocí SMTP poštu odesílá, pomocí POP3 poštu stahuje. Stažení pošty probíhá podle schématu na kresbě II-32.



Kresba II-32: Uživatel stahuje poštu pomocí POP3.

Záleží na uživateli, jak často jeho poštovní klient poštu ze schránky přenáší k jeho dispozici. V případě trvale připojeného počítače to může být každých 10 minut, v případě občasného je stažení dáno pokynem uživatele a okamžikem připojení a může se pohybovat v intervalu jednoho i více dnů. Příjem pošty do schránky tedy ještě neznamená, že uživatel o poště ví a že ji četl. Proto je možné si vyžádat potvrzení o přjetí odeslané pošty teprve samotným uživatelem, což umožňují všechny uživatelské poštovní programy.

Součástí poskytovaných služeb správcem poštovního uzlu bývá také možnost používání brány GSM (GSM gateway). Jedná se o automatizované zaslání textové zprávy SMS na mobilní telefon v okamžiku, kdy byla nová pošta do schránky doručena.

Nepříjemnost vyplývající ze systému tzv. stahování pošty je zřejmá: po přenosu nově doručené pošty do počítače s uživatelským poštovním klientem je pošta dostupná již pouze v tomto počítači. Provedu-li tedy stažení pošty např. do internetové kavárny, jsem-li zrovna na cestách, pošta zůstane na počítači v kavárně, a pokud ji neodstráním, bude v obvykle nechráněných operačních systémech přístupná i dalším návštěvníkům kavárny. Protokol POP3 sice umožňuje zanechání pošty na serveru (leave messages on the server), což znamená, že je stažena pouze kopie pošty, ale tato možnost problém neřeší, protože POP3 nedokáže určit, kam už byla pošta stažena. Využití možnosti stahovat poštu na jeden z počítačů úplně a na jiné pouze v kopii tedy POP3 nedokáže zajistit. Problém lze řešit přenosným počítačem, přeposláním pošty zpět na naši adresu po jejím přečtení a nebo využitím poštovních uživatelských programů, které pracují přímo s poštou v poštovní schránce a nikdy nikam poštu nestahují.

Práci s poštou přímo na počítači s poštovní schránkou umožňuje z uživatelského PC protokol IMAP (Internet Message Access Protocol), který je již součástí většiny klientů poštovních programů a uživatel si jej může nastavit pro komunikaci s obsahem své poštovní schránky ve vzdáleném uzlu namísto programu POP. IMAP (dnes IMAP4) je mladší, zahrnuje funkce POP a byl zkonstruován právě za účelem doplnění chybějících vlastností, které by mohly zpříjemnit práci s poštou. Uživatelé umožňuje mít nastavený režim přímé práce s poštovní schránkou, uživatel může poštu číst z různých počítačů a přesto si IMAP udržuje stopu, která zpráva byla kam stažena, takže je stažená pošta na více místech zcela konzistentní. Přesto se IMAP používá méně než POP, pravděpodobně proto, že jeho výhodám uživatelé nerozumějí, neboť i samotný princip mechanismu pošty (odesílání, přijímání, stahování, ponechání ve schránce atd.) působí zmateně a přinesla jej nutnost většiny uživatelů pracovat off-line.

IMAP

Další způsob práce s poštovní schránkou přímo na uzlu, kam pošta chodí, je pomocí prohlížeče Internetu. Poštu uživatel takto vidí prostřednictvím stránek WWW, do kterých obsah poštovní schránky převádí speciální software. Předpokladem ale je opět práce on-line, tj. být trvale připojen k Internetu a dále provoz speciální aplikace na straně serveru, což si ještě ukážeme později.

Při používání poštovních uživatelských programů s občasným připojením k serverům způsobem SMTP a POP3 vyžaduje takový klientský software několik nastavení. Je to především určení uzlu v síti, který přijímá požadavek na odesílání pošty, tedy server odchozí pošty SMTP. Jedná se o vyplnění kolonky s označením uzlu v konvenci DNS, tedy např. `smtp.posta.cz` (nebo `smtp.volny.cz`). Dále je nutno určit uzel s naší poštovní schránkou, což bývá v programech označováno jako server příchozí pošty POP3 (např. `pop3.`

volny.cz). Poštovní schránka je autorizovaná, tzn. že do software stahování pošty zadáváme jméno schránky (obvykle část před znakem @) a po kontaktu se serverem POP3 jsme požádáni o zápis hesla, kterým je přístup ke schránce chráněn. Nastavení adresy odesílatele může být libovolné (!). Jedná se především o adresu, kam bude adresát odesílat případnou odpověď. Běžně uživatel vyplní svoji poštovní adresu v Internetu (aby mu odpověď přišla, že ano). Spamer, tj. člověk, který rozesílá hromadné emaily (např. s reklamou), používá falešnou, často neexistující adresu. Pokud spamerem uvedená adresa navíc existuje, je takto zneužitý falešný odesílatel napadán za aktivitu, se kterou nemá nic společného. Využívání některého z veřejných serverů SMTP tuto činnost spamérům jen zlehčuje, neboť určení, kdo požádal o odeslání pošty, je u serverů SMTP sice možné registrovat, ale registrovaná adresa IP je dynamicky přidělována různým telefonickým spojení. V digitálních ústřednách lze ovšem zjistit použité telefonní číslo, ale takové zjišťování již podléhá zákonům lidské společnosti a lidským právům zejména. Pro sledování telefonických hovorů je totiž nutno mít příkaz soudce, který je vydáván pouze na základě důvodného podezření na trestný čin. Spam a ani zneužití cizí poštovní adresy v mnoha zemích světa ještě trestným činem není (také z důvodu, že je obtížné definovat, co ještě spam není a co už ano). Bezpečnosti v kontextu zákonů lidské společnosti se budeme více věnovat v dalších částech tohoto textu. Pro posílení identifikace odesílatele pošty byl protokol SMTP rozšířen o možnost vyžadovat při odesílání pošty identifikační jméno a heslo. I na volně dostupných serverech SMTP musí být proto každý uživatel, který chce služby použít, registrován. Způsob registrace je ale často omezen na vyplnění jednoduchého formuláře elektronickou cestou a bez právního podkladu. Přesto, zlobí-li pak takto registrovaný uživatel spamerem, může jej správce systému vymazat z registrace. Spamerovi ale nic nebrání si okamžitě vytvořit registraci novou. V současné době probíhá neustálý boj mezi hackery a správci systémů, které komplikují obchodní zájmy (veřejné servery žijí z reklam a o nové uživatele mají zájem) a nedostatečné zákony.

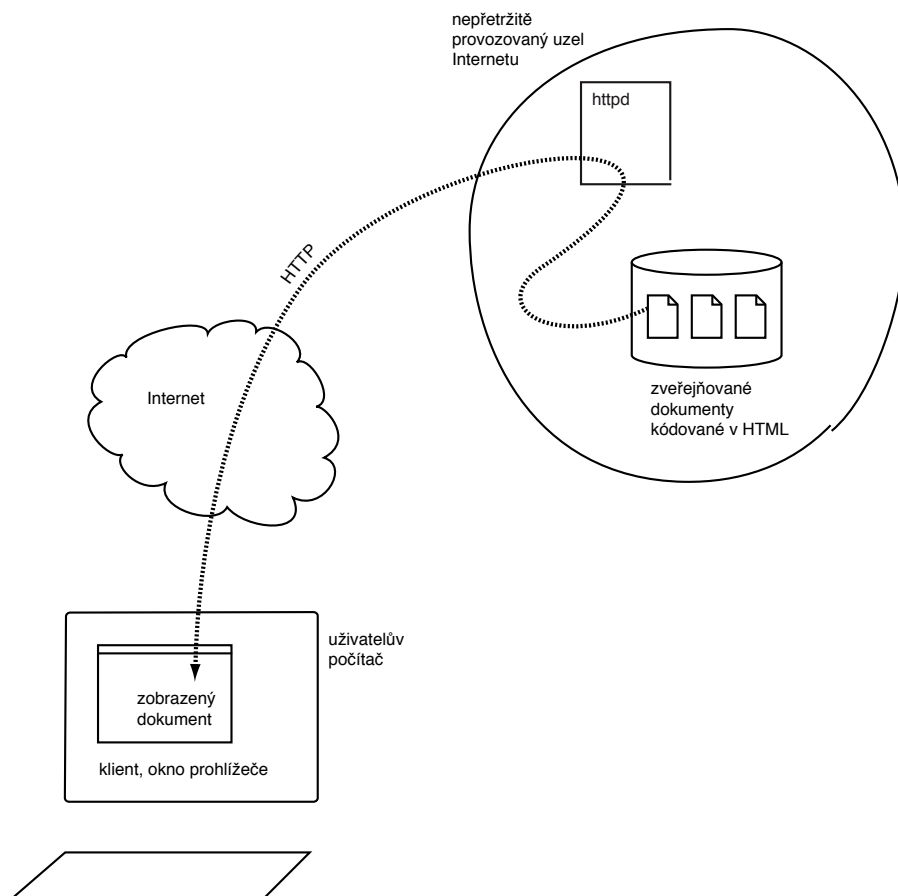
WWW -
elektronické
publikování

Internet dnes pro většinu jeho uživatelů představuje síťová služba poskytování stránek WWW (World Wide Web, v překladu celosvětová pavučina). Obecně ji lze nazývat elektronické publikování, mnohdy se také možná přesněji označuje termínem informační servery.

V principu se jedná o zveřejňování informací v podobě dokumentů. V Internetu nabízí dokumenty server WWW instalovaný v uzlu s nepřetržitým provozem. Každý uživatel, jehož počítač je k internetu připojen a má k dispozici prohlížeč nabízených dokumentů, tedy klienta WWW, pak může dokumenty prohlížet a číst. Vzhledem k povaze této síťové aplikace se často také hovoří o části autorské a čtenářské.

HTTP, HTML

Aplikace stránek WWW je postavena na aplikačním protokolu HTTP (Hypertext Transfer Protocol), dokumenty jsou psány v jazyce HTML (Hypertext Markup Language), jak ukazuje kresba II-33.



Kresba II-33: Síťová aplikace stránek WWW.

Klient si vyžádá od serveru dokument. Server dokument čte z disku a posílá ho sítí klientu. Klient dokument přijme a interpretuje jej podle značek jazyka HTML. HTML není programovací jazyk, je to jazyk, který vyšel z tradic pokynů pro fotosázecí stroje při výrobě knih a publikací. Předchůdcem HTML jsou jistě programy `ntroff` a `troff` operačního systému UNIX, ve kterých je dokumentace v UNIXu kódována `dodnes`. V programu `troff` se totiž dokumenty připravují pro tisk nezávisle na dále použitém fotosázecím stroji. Jejich pokračovatelem je program `tex`, jímž nabízené možnosti sazby dosáhly pravděpodobně maximálního komfortu. HTML ale definičně vyšel z jazyka SGML (Standard Generalize Markup Language, viz Příloha C). Princip HTML je jednoduchý. Publikovaný text je proložen smluvenými značkami, kterými jsou označovány např. nadpisy, začátky odstavců, způsob zarovnání textu, typy písma, jeho velikost, tloušťka atp. Značky jsou často *párové*, tj. jedna z nich označuje začátek např. tučného písma a druhá jeho konec. V jazyce HTML značkám říkáme *tagy*. Jsou povinně uzavřeny mezi znaky `<` a `>`. Druhá část párového tagu začíná `</`, např. text

```
<P><B>Mary</B> had a little <I>lamb</I>;<BR>its fleece was white
as snow</P>
```

obsahuje tři párové tagy: **P** (vymezení odstavce), **B** (tučné písmo), **I** (kurzíva) a jeden nepárový tag **BR**, kterým označujeme konec řádku. Text bude prohlížeč WWW zobrazovat např. takto:

```
Mary had a little lamb;
its fleece was white as snow
```

V dokumentu se umisťují obrázky uvedením jména souboru, ve kterém jsou uloženy, např.

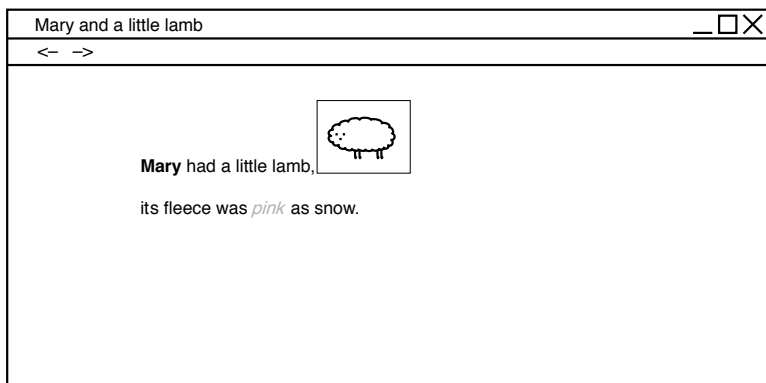
```
<P><B>Mary</B> had a little <I>lamb</I>;
<IMG SRC="lamb.jpg">
<BR>its fleece was white as snow</P>
```

kdy uprostřed textu se objeví obrázek beránka ze souboru `lamb.jpg`.

V HTML musíme ještě dokumentu vyznačit jeho začátek a konec a dodržet definice jeho částí, především záhlaví (**HEAD**) a tělo (**BODY**) dokumentu. V následujícím příkladu je uveden úplný funkční dokument, který v záhlaví definuje titulek dokumentu (**TITLE**). Tagem **CENTER** požadujeme vystředění textu s obrázkem v okně prohlížeče. V příkladu je beránek růžový, což vyjadřujeme růžovým písmem v definici tagu **FONT**.

```
<HTML>
<HEAD>
<TITLE>Mary and little lamb</TITLE>
</HEAD>
<BODY>
<CENTER>
<P><B>Mary</B> had a little <I>lamb</I>;<IMG SRC=lamb.jpg><BR>
its fleece was
<FONT COLOR=pink>pink</FONT> as snow</P>
</CENTER>
</BODY>
</HTML>
```

a v okně prohlížeče bude zobrazen dle kresby II-34.



Kresba II-34: Dokument v okně prohlížeče.

Základní a důležité tagy, jejichž význam je podstatný pro pochopení možností jazyka HTML, uvádíme v Příloze C.

Dokumenty jsou koncipovány na principu hypertextu. Znamená to, že sdělovaný text je rozdělen na výchozí stránku (home, domovská), která obsahuje obvykle základní informace, jako je abstrakt a obsah dokumentu. V textu stránky se pak vyskytují odkazy (links, anchors). To jsou citlivá místa. Klikneme-li na ně myší, klient požádá server o zaslání stránky, která je s odkazem spojena a po přijetí svým textem vyplní okno prohlížeče. Dokumenty HTML jsou takto obvykle čteny sledováním určité série odkazů tak, že přecházíme ze stránky na stránku. Styl rychlého pohybu v textu prostřednictvím elektronické podpory odkazů je typický pro programátora, který hledá v dokumentaci informace potřebné pro svůj záměr, např. název potřebné funkce nebo celého programu. Zjevně ale usnadňuje přístup k informacím specialistům také z jiných oborů, protože hypertextové publikování prakticky nastartovalo obecný zájem o Internet. Odkaz v kódu textu stránky určujeme párovým tagem `A`, např.

odkazy

```
<P><B><A HREF="mary.htm">Mary</A></B> had a little <I>lamb</I>;  
<A HREF="lamb.htm"><IMG SRC="lamb.jpg"></A>  
<BR>its fleece was white as snow</P>
```

Začátek a konec tagu `A` určuje text *Mary*, který bude klientem zvýrazněn modře a podtržením. Bude přitom citlivý. Kliknutí myši způsobí, že klient požádá server o stránku, která je uložena na tomtéž místě jako stránka s naším odkazem, ale se jménem souboru *mary.htm*. Dále je v příkladu uveden tag `A` u obrázku. Celý obrázek je citlivý a odkazuje na stránku dokumentu v souboru *lamb.htm*.

V textové kolonce okna klienta, prohlížeče stránek WWW, určujeme umístění dokumentu v síti. Umístění dokumentu podléhá formátu URL (Uniform Resource Locator, určení umístění zdroje), který je jednodušší podobou formátu URI (Uniform Resource Identifier, identifikace určení zdroje). Příkladem URL je

URL

`http://www.skocovsky.cz`

kde jsou uvedeny dvě povinné části URL, tj. komunikační protokol HTTP a určení adresy uzlu podle DNS `www.skocovsky.cz`. Pokud je v uvedeném uzlu podle DNS uložen dokument se jménem `home.htm`, server jej klientu pošle a ten jeho obsah zobrazí v okně podle konvencí HTML. Uvedený odkaz má tedy stejný význam jako

`http://www.skocovsky.cz/home.htm`

V tomto výchozím (home) dokumentu může být umístěn odkaz na další stránku WWW dokumentu, která je uložena např. v souboru `enjoyer/home.htm`, tedy v podadresáři se jménem `enjoyer`, ve kterém je soubor se jménem `home.htm`. Odkaz v textu výchozí stránky dokumentu je určen takto:

```
<A HREF="enjoyer/home.htm">Computer Science Enjoyer</A>
```

a text `Computer Science Enjoyer` bude citlivý a zvýrazněný při zobrazení stránky klientem. V textové kolonce okna klienta zobrazování stránek WWW můžeme ale zadat přímo tuto adresu jako adresu výchozí stránky požadovaného dokumentu:

`http://www.skocovsky.cz/enjoyer`

(`home.htm` nemusíme vypisovat). Vyhneme se takto výchozí stránce na adrese `www.skocovsky.cz`, dokument tedy bude začínat stránkou v podadresáři `enjoyer`.

Jak je patrné, určení celistvosti dokumentu je vážné. Dokument totiž můžeme chápat jako množství menších dokumentů navzájem propojených odkazy. Odkazy mohou pochopitelně směřovat zpět na stránky, které jsou výchozí na adrese s označením pouhého uzlu atp. Obecně vzato má text uvedený v uvozovkách za návěštím `HREF` v definici odkazu v kódu stránky stejný formát jako ten, který zapisujeme do okna prohlížeče stránek, tj. URL. Odkazy totiž mohou směřovat kamkoliv v rámci adresace Internetu, což se hojně používá. Je ovšem věcí autora dokumentu udržovat odkazy aktuální. Odkazy se totiž často mění, zanikají a vznikají na jiných místech. Uvedením pouhé části URL v odkazu znamená, že klient vyřizuje odkaz podle relativního umístění poskytnuté stránky s tímto odkazem. Je-li v odkazu za `HREF` uvedeno pouze jméno souboru (např. `mary.htm`), hledá klient soubor tohoto jména v tomtéž adresáři, ve kterém je umístěn dokument s odkazem, a to v tomtéž uzlu a za použití stejného protokolu HTTP. Je-li zde uvedeno označení uzlu, oslovuje server na takto dané adrese a požaduje odkazovanou stránku.

Cesta k části dokumentu může být dále podmíněna odkazem do souboru se stránkou. Takto se přenesená stránka zobrazí od určeného místa, nikoliv od svého začátku (např. nastavení na kapitulu delší stránky). V odkazu se používá znak `#` k rozlišení části pro samotnou stránku, např.

`http://www.skocovsky.cz/brosko89/51cz.htm#51`

je odkaz na stránku se jménem `51cz.htm`, ale do části, která je označena návěštím `51`. Návěští (používá se také termín kotva, anchor) pak musí být v dokumentu uvedeno použitím tagu `A` takto:


```
<A NAME="51">5.1 Úvod do jazyka C</A>
```

a klient zobrazí dokument od takto vyznačeného textu.

Část, kterou URL začíná, je určení komunikačního protokolu. U zobrazování stránek WWW se jedná vždy o text `http`. URL ovšem aspiruje na určení obecného výpočetního zdroje kdekoli v Internetu, proto zde můžeme použít i jiné protokoly. Je jisté, že zadáním jiného protokolu oslovuje klient jiný server a samotný klient si s takovým protokolem bude muset umět také poradit. Např. při

```
ftp://www.skocovsky.cz/archiv/upp9.pdf
```

je startována komunikace prostřednictvím protokolu FTP (prohlížeč stránek WWW ji umí). V uzlu `www.skocovsky.cz` je osloven server FTP a je požádán o dokument uložený v souboru se jménem `upp9.pdf` v adresáři `archiv`. Takto lze prohlížečem stránek stáhnout požadovaný soubor, ať již se jedná o dokument, hru nebo databázi.

Uvedený příklad komunikace protokolem FTP je anonymní. Znamená to, že server nepožaduje zadání jména a hesla. Pokud by ale o anonymní komunikaci nešlo, musel by být náš odkaz URL rozšířen o přístupové jméno a heslo např. takto:

```
ftp://kokolia:naseVec@www.skocovsky.cz/archiv/upp9.pdf
```

kde zadáváme `kokolia` jako jméno, které je evidováno v uzlu `www.skocovsky.cz`, je kryto heslem `naseVec` a takto evidovaný uživatel má přístupová práva nejméně pro čtení dále určeného souboru s dokumentem.

Úplný syntax definice odkazů URL je uveden v Příloze B.

Kromě ryze publikačních možností obsahuje ale jazyk HTML i prvky zpětné komunikace klienta se serverem tak, aby čtenář mohl pro autora zanechávat např. své poznámky k prezentované publikaci. Přenos dat od klienta k serveru se využívá také např. pro sběr informací o čtenářích nebo registraci elektronických objednávek zboží nabízeného na komerčních stránkách Internetu. Je možné jí dosáhnout využitím *formuláře* (form), a to použitím párového tagu `FORM`, např.:

```
<FORM ACTION="schovej_mi_to" METHOD="post">
```

```
Zde napište stručně Váš názor, ale žádná sprostá slova,  
prosím:<BR>
```

```
<INPUT TYPE="text" NAME="nazor" SIZE=50>
```

```
<INPUT TYPE="submit" NAME="tlacitko" VALUE="odeslat">
```

```
</FORM>
```

je jednoduchý formulář uprostřed textu vymezeným tagem `BODY`, v okně prohlížeče se v tomto případě objeví dle kresby II-35.

Kresba II-35: Formulář v okně prohlížeče.

PHP, JSP

A čtenář může psát do zobrazené kolonky formuláře. Kolonka je v HTML určena nepárovým tagem `INPUT` s vlastností `TYPE` jako text. Velikost jsme určili na 50 znaků vlastností `SIZE`. Po jejím vyplnění čtenář stiskem tlačítka s nápisem `odeslat` potvrzuje vyplněnou kolonku a její obsah je klientem odeslán serveru. Tlačítko s nápisem `odeslat` je také v textu HTML určeno tagem `INPUT`, ale jeho typ je `submit` (česky potvrzení). Text na tlačítku jsme stanovili vlastností `VALUE`. Oba použité tagy `INPUT` (vstupní pole) jsme také pojmenovali, a to vlastnostmi `NAME` jako `nazor` a `tlacitko`. Tato jména jsou důležitá proto, aby server rozlišil více takových vstupních polí (formulář by např. také mohl obsahovat kolonku pro jméno čtenáře jako jeho podpis). Pro server je také důležitá vlastnost `ACTION` u tagu `FORM`. Text `schovej_mi_to` je totiž jméno programu, který server použije na počítači se zobrazovanými dokumenty k tomu, aby přijímaný formulář takto zpracoval. Co tento program udělá s textem, který čtenář zapsal do kolonky s názvem `nazor`, je pouze na něm a HTML nijak nedefinuje tyto manipulace. Program (`schovej_mi_to`) text od klienta obvykle ukládá do určitého místa na disk, kam má přístup autor dokumentu. Zda je toto místo na disku soubor nebo databáze, je jeho věcí. Každopádně se již jedná o programování v nějakém programovacím jazyce, což je činnost výrazně specializovanější než je označení textu tagy a jeho umístění na server a obyčejný uživatel ji běžně není schopen zvládnout. Znamená to, že při využívání jiných než čistě publikačních činností u HTML je nutné spolupracovat se specialistou programátorem. A HTML se dnes skutečně hodně používá pro interaktivní spolupráci čtenáře a autora. Prostřednictvím formulářů se dokonce staví složité informační systémy, ke kterým uživatelé přistupují prostřednictvím prohlížečů stránek WWW. Zde je jazyk HTML využit jen jako špička ledovce, jeho pomocí dochází k zobrazování a přijímání dat uživatelů. Za obsahem vlastností `ACTION` tagů `FORM` se ale skrývá prostředník sofistikovaného přístupu k databázi a jejím aplikacím. Jednou z variant programovacího jazyka, který je velmi používaný, je např. PHP (PHP Hypertext Preprocessor, zkratka je rekurzivní akronym), ale dnes prakticky každý výrobce databáze a databázových aplikačních podpor (aplikační servery) nabízí vlastní řešení, z nichž výrazně používané je např. JSP (Java Server Pages). PHP je ovšem zdarma a nabízí možnost práce s libovolnou známou

databází, zejména s MySQL, což je opět výkonná, kvalitní a volně použitelná aplikace. PHP původně vznikl právě na podporu uživatelů, kteří potřebovali programovat jednoduché úkony v našem příkladu uvedeného typu. Původní zkratka PHP je totiž odvozena od Personal Home Page tools, česky nástroje pro osobní stránky WWW. PHP nebo jiné programovací jazyky tohoto typu však zvládne obyčejný uživatel obtížně. V současné době je proto snaha vytvářet různé graficky orientované nástroje pro návrh jednoduchých interaktivních systémů sloužících uživatelům Internetu ke komunikaci (např. redakční systémy a aplikace práce s poštou přímo na serveru). Složitější informační systémy ale musí vždy řešit, navrhovat a realizovat skupina odborníků, která si při dnešní mimořádné poptávce nechává za své produkty dobře platit.

Používat HTML pro vstupně výstupní komunikaci s uživatelem je provozně velmi výhodné. Prohlížeče stránek WWW jsou dnes součástí každého operačního systému. Pracují s texty i obrázky nezávisle na typu hardware a uživatelé přístup ke stránkám WWW znají. Nevýhody jsou v omezeních, která jsou dána v původním publikačním zaměření HTML. Obtížně se vytvářejí animace a interaktivní grafický systém, v němž je potřeba vykreslit křivku, upravit fotografii atp., je vyloučen zcela. K nápravě nastartovaného trendu internetového (tedy síťového) přístupu k datům umístovaným na různých vzájemně propojených počítačích strojích přispěla již před lety definice síťového programovacího jazyka Java (firmou Sun, viz např. [Flanagan1996]).

Současnou definici HTML (s označením XHTML, Extensible HTML) podle standardů Internetu Konsorcia W3C (na www.w3.org) je možné studovat a používat ji. Silná firma Microsoft ale mnohdy tyto a podobné standardy neuznává a vytváří definice vlastní. Vzhledem k jejímu postavení na trhu se softwarem pro PC je tento vliv velmi podstatný a definice Konsorcia W3C se musí mnohdy přizpůsobovat tlaku uživatelů firmy Microsoft, přestože vývoj dialektu HTML u Microsoftu je mnohdy v rozporu s nutností zachovávat kontinuitu vývoje a s potřebami směřování informačních technologií. Pro uživatele tak vzniká neradostná situace, kdy použití HTML jako obecného rozhraní prezentace dat není ve všech prohlížečích totožné. Doufejme, že je tato situace pouze dočasná a do budoucna zvítězí inteligence a snaha vzájemně si porozumět nad snahou monopolizovat tuto část lidské činnosti na naší planetě.

XHTML

Úspěch protokolu HTTP a jazyka HTML vedl ke vzniku komunikačního datově orientovaného jazyka XML (Extensible Markup Language). Používá se pro vzájemnou komunikaci různých databázových strojů při výměně dat, ale je také definičním jazykem pro značkovací jazyky a jejich rozšiřování (jako je např. HTML). Používat XML je složitější než HTML a souvisí již více s programováním (nebo s jeho analýzou). Zápis v něm je ale přehledný a srozumitelný, je zobrazitelný dnes prakticky ve všech prohlížečích stránek WWW. V dalších částech knihy se o něm ještě zmíníme.

XML

III.1 – kód, text a obraz

Lidé dnes pro vzájemnou komunikaci, ale vůbec pro vyjadřování produktů své mysli, většinou používají elementární částice. Jedním z typů takových částic jsou znaky abecedy, ze kterých se za použití navykklé syntaxe odvíjí užívání přirozeného jazyka. Kódujeme svá sdělení a snažíme se tímto způsobem dosáhnout co nejvyšší přesnosti, byť, jak naznačuje teorie Turingových strojů zmíněná v části Base Of I.2, je tato přesnost nedosažitelná.

Fungování veškerého software je neodmyslitelně spojeno s takovým typem vyjadřování. Kódování určuje chování všech počítačových programů, které kdy byly dosud napsány. Základním způsobem práce hardware je interpretace strojového kódu (machine code) v reakci na sekvenci znaků binární povahy, tedy nul a jedniček. Říkáme sekvenci pro zdůraznění významu umístění bitů ve skupině. Strojový kód je elementárně vyjádřen jako soupis chování stroje právě na základě jednoznačně definovaných reakcí na takové různé sekvence. Jazyk, který tak vzniká pro ovládání stroje, nazýváme assembler (viz část Base Of I.2).

strojový kód

Programátor může psát program přímo ve strojovém kódu, ale jeho práce by byla pomalá a velmi náročná. V průběhu vývoje software se v rámci Computer Science etablovala partie programovacích jazyků vyšší úrovně. Vyšší úroveň znamená, že programátor vyjadřuje (kóduje) instrukce pro počítač v jazyce, který je mu bližší z hlediska myšlení člověka, tedy přirozeného jazyka lidí. Zápis v programovacím jazyce do strojového kódu převádí překladač (kompilátor). Vznikají tak dvě verze programu. Jedna je nazývána program ve zdrojovém kódu (source code) a druhá program v binárním kódu (binary code). Binární kód se od strojového liší velmi málo. Struktura binárního kódu je však určena typem operačního systému a jeho způsobem, jak s kódem pracuje, tedy např. jak program zavede do operační paměti nebo jak sleduje jeho provádění. Program v binárním kódu je nicméně prováděn procesorem. Zdrojový kód mění programátor a nově ho překládá do binární podoby v období vývoje programu a jeho ladění nebo v případě potřeby změn v chování programu. Překladač je také program a byl prvotně naprogramován ve strojovém kódu. Příkladem vyšších programovacích jazyků je jazyk C, Pascal, Java, viz opět část Base Of I.2. Každý konkrétní překladač je vždy spojen s jedním jazykem a určitým typem počítače, tedy typem procesoru. Přestože se může chovat velmi příbuzně (např. v operačních systémech UNIX), nejméně jeho část pro generování strojového kódu musí být specifická. Pro jeden typ procesoru a i operačního systému ale může existovat řada vzájemně si konkurujících překladačů různých výrobců, což přináší zájem trhu. Pokud však existuje závazná definice programovacího jazyka (nejlépe standardem ISO) a programátor překladače ji respektuje, dochází k přenositelnosti (portability) programů na úrovni zdrojového kódu mezi různými typy procesorů a tedy i počítačů. Znamená to, že tentýž program může být přeložen různými překladači pro

jazyky vyšší úrovně

zdrojový a binární kód

přenositelnost programu

různá prostředí různých operačních systémů na různých typech architektury různých hardware. Je to jeden z významů partie umělých jazyků v Computer Science. Bohužel jej mnozí výrobci, ať již záměrně (pro účel udržení klientely na své platformě hardware) nebo nevědomě (pracují bez erudice v Computer Science) nerespektují. Současně je ale jisté, že dosáhnout kompatibility je mnohdy velmi pracné, protože řadu nutných věcí standardy stále nedefinují. Jedná se např. o přístup k perifériím, grafickým podsystémům atd.

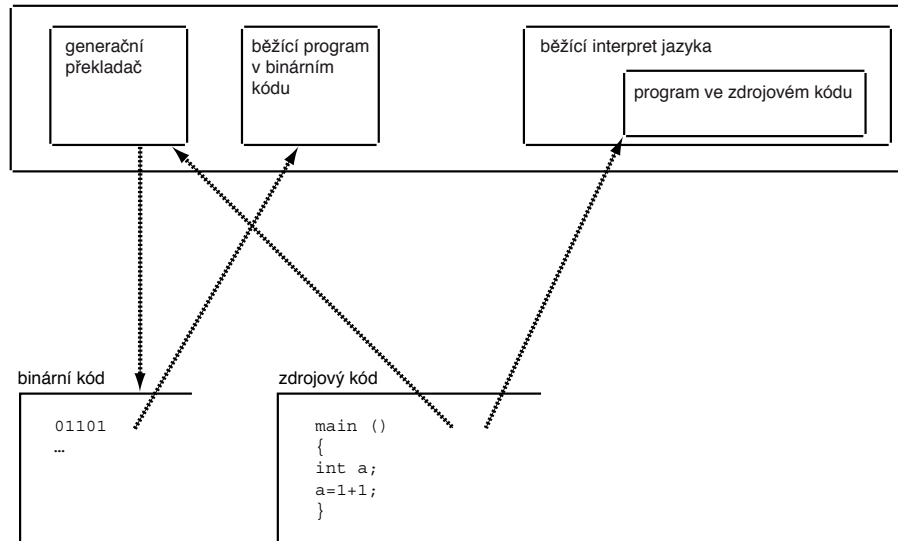
generační
překladače

Překladače, které na základě zdrojového kódu generují binární podobu programu, nazýváme generační překladače.

interprety

Druhý významný typ překladačů představují překladače interpretační. Zdrojový kód programu ve vyšším programovacím jazyce je přímo prováděn. Provádění přitom není svěřeno procesoru, ale právě překladači, který je programem prováděným počítačem. Situaci ukazuje kresba III-1.

operační paměť



Kresba III-1: Generační a interpretační překladače.

Na rozdíl od generačního překladače, u kterého je nutná fáze spuštění programu pro získání binární podoby zdrojového kódu, čte interpretační překladač přímo soubor zdrojového kódu a provádí jej. Jedná se tedy o přímý převod instrukcí vyššího jazyka do instrukcí kódu stroje. Tento způsob překladače je pro vývoj programu příjemnější, má ale vyšší nároky na samotný počítač, protože interpret vyžaduje větší prostor v operační paměti a běh programu takto zprostředkovávaný bude jistě pomalejší. Záleží také na složitosti programovacího jazyka, která je dána jeho schopnostmi vzhledem ke komfortu programování. Proto také byly první interprety programovány pro jednodušší jazyky vyšší úrovně, jako byl např. Basic. Při dnešním stále se zvyšujícím výkonu strojů jsou ale mnohdy i složitější programovací jazyky zajišťovány interpretačním způsobem. Příkladem jsou skriptovací jazyky.

jako třeba Perl nebo Python. Interpretaci ale např. také provádí programovací jazyk PHP pro dynamické stránky WWW při součinnosti s publikačním jazykem HTML.

Dalším příkladem, který ukazuje, kam se ubírá vývoj programování, je bezpochyby strojová podpora jazyka Java. Překladače firmy Sun Microsystems (již autorství Java patří) nabízí jednak generování do binární podoby a jednak možnost generování do *mezikódu*, tedy do určité obecné podoby strojového kódu, který je snadněji a výkonněji interpretován pomocí Java engine, tedy interpretačním překladačem. Výhodou je shodný mezikód pro různé platformy počítačů a jejich operačních systémů a vyšší výkon programu bez ztráty všech progresivních vlastností jazyka. Nevýhodou je nutná přítomnost *engine* pro provádění mezikódu, tj. jeho instalace pro danou platformu. Pro psaní nutně přenositelných programů je však takové vývojové prostředí velmi užitečným pomocníkem. Samotná distribuce konečných aplikací obvykle ale obsahuje i binární podobu aplikace pro různé platformy, tedy produkty adekvátního generačního překladače. Tento princip vývoje programů také převzaly mnohé skriptovací jazyky. Stejně tak např. v programovacím jazyku Perl lze vyvíjet za přímého provádění zdrojového kódu, buď v prostředí MS Windows nebo UNIXu. Pro distribuci konečného programu je ale možné vytvořit binární kód pro odpovídající platformu a uživatelům předávat pouze binární kód.

engine

Ve světě Computer Science má význam také *binární přenositelnost*. Jedná se o dosažení přenositelnosti programů na úrovni binárního kódu při změně platformy, na které je binární kód prováděn. Platformou je zde operační systém a hardware. Binární přenositelnosti je dnes docíleno víceméně u procesorů stejné vývojové řady, tedy u procesorů, jejichž strojový kód je pouze rozšiřován a nebo u těch, které jako podмноžinu svého instrukčního repertoáru obsahují strojový kód svých předchozích verzí. Zde se ukazuje, jak důležitý je návrh prvotní koncepce architektury procesoru. Přestože úspěšný obchod může takový argument potlačit, v konečném důsledku se po čase projeví např. v neadekvátně malém nárůstu výkonu vzhledem k fyzikálním parametřům nového produktu a nebo v nutnosti dále nepodporovat dříve používané programy v jejich plném rozsahu. Některé programové sofistikované systémy *emulují* chování jiných procesorů, interpretují strojový kód a nabízejí tak proveditelnost programů psaných původně pro jiné procesory. Jedná se vlastně o analogii interpretačních překladačů, hovoříme ale o emulaci, protože takový engine musí simulovat všechny části původního počítače a převádět je na třeba zcela jiný typ architektury.

binární
přenositelnost

K tomu, aby programátor mohl kódovat program, potřebuje prostředek pro jeho pohodlné psaní. Programy pořizování zdrojového kódu programů nazýváme textové editory. Textovým editorem je třeba i jednoduchý poznámkový blok (notepad) v prostředí MS Windows, přestože kvalitní textový editor vždy obsahuje řadu potřebných funkcí pro komfort psaní samotného textu programu, jako je snadná kopie bloku textu, výměna textu za jiný, pohyb v textu po různých částech a v různém kontextu atd. Podstatné ale je, že programátor při práci s textovým editorem čte skutečně každý

textové editory

bajt, tedy všechny znaky, které pořizovaný textový soubor obsahuje. Je to zapotřebí, protože každý bajt textu je součástí kódu programu, který musí jednoznačně odpovídat syntaktické skladbě dané gramatikou jazyka. Dobrý textový editor proto nezamlčí žádný znak a programátor může kterýkoliv znak pohodlně vyměnit za jiný. Výjimkou jsou *white spaces*, prázdná místa, tj. znak mezery, nového řádku nebo tabulátoru. Překladače je všechny shodně považují významově za oddělovače slov a editory je zobrazují jako prázdná místa. I tyto znaky lze ale obvykle určitým ovládním editoru zobrazit v jednoznačném rozlišení. Obvyklá práce s textovým editorem připomíná psaní na mechanickém psacím stroji. Na papíře jsou pouze ty znaky, které úderem klávesy otiskneme. Příkladem holého textu je

```
Mary had a little lamb;  
its fleece was white as snow
```

který jsme uvedli v části Base Of I.1 jako příklad digitalizace textu. V ukázce jsme uvedli jeho digitální podobu a právě s takovou úrovní textové editory pracují. Textovým editorem dohlédneme až na obsah každého bajtu a můžeme jeho obsah vyměnit za jiný obsah stiskem klávesy s odpovídajícím znakem. Principiálně lze říct, že textové editory pracují na úrovni bajtů, tedy znaků, které člověk vnímá jako srozumitelný kód.

binární editory

Existují i editory, které nám umožňují modifikovat soubory obsahující jakýkoliv znak ASCII, říkáme jim binární editory. Jsou to programy, jimiž nahlédneme do bajtů souboru např. s programem v binárním kódu nebo souboru obsahujícího obrázky. Binární editory na rozdíl od textových umožňují programátorovi (a i kvalifikovanému uživateli) měnit dokonce jednotlivé bity v každém bajtu digitálního obsahu souboru. To vše (a především) v bajtech, které nemají pro člověka význam textu (textových znaků je v ASCII přibližně 100 z celkového počtu 256).

Programátor a uživatel však v drtivé většině případů používají nejvýše úroveň textových editorů, protože binární kód je určen pro interpretaci samotným strojem. Oprava binárního kódu je tedy speciální aktivita potřebná pouze např. v krizových situacích fungování počítače, jako je např. kontrola práce překladačů, zásah do programů, ke kterým není k dispozici zdrojový kód ve vyšším programovacím jazyce nebo oprava jádra operačního systému či souvisejících systémových tabulek po havárii software počítače.

Příkladem výkonného textového editoru je editor *vi* (*ex*) operačního systému UNIX. Jeho ovládání je v rozporu s pohodlným uživatelským rozhraním jednoduché editace textu, kterým je např. již zmíněný notepad. *vi* disponuje rozsáhlým aparátem funkcí od jednoduché výměny textů až po filtraci speciálními aplikacemi (Tools). Každý začátečník si na něm vyláme zuby, ale programátoři jej již po dobu 20 let milují. Rychle v něm totiž dosáhnou složité úpravy textu a jeho implementace je zcela shodná ve všech typech operačního systému UNIX. Základní úvod práce s textovým editorem *vi* uvádíme v Příloze D.

Jiným příkladem komfortních textových editorů jsou editory, které jsou součástí vývojového prostředí některého programovacího jazyka vyšší úrovně. Jejich komfort je dán především vazbou na syntax používaného programovacího jazyka, takže editor při práci programátora kontroluje jeho zápis z pohledu pravidel gramatiky, která jazyk definují. Znamená to, že editor opravuje chyby, např. definice začátku a konce programového bloku, nebo na ně alespoň programátora upozorňuje (např. červenou barvou). Editor může i určitý programový kód odmítnout zapsat, ale současně také může doplňovat povinné struktury jazyka ve správné syntaxi, takže programátor se více věnuje obsahové části psaní programu. Tyto textové editory nazýváme syntaxí řízené editory a jejich význam nemusí být oceněn pouze při programování. Kód v jazyce HTML, který interpretuje klient zobrazování stránek WWW, dnes vytváří nejenom programátor, ale většinou i erudovaný výtvarník. Pro něj jsou vytvářeny speciální aplikace komponování stránek WWW, které generují samotný kód HTML (Macromedia Dreamweaver, Adobe GoLive, Mozilla Composer, Microsoft FrontPage atd.). Všechny tyto aplikace výtvarníkovi umožňují pohled do textové podoby zápisu stránky WWW v kódu HTML v průběhu práce a jeho ruční modifikaci. Pro snadnější orientaci a kontrolu jeho ručních vsuvek pracuje pohled do textu jako syntaxí řízený editor gramatiky jazyka HTML. Např. napíšu-li text

```
<HTML>
<BODY>
<B>tučný titulek <I>následuje kurzíva</I>
<P>a text</P>
</BODY>
</HTML>
```

chybí zjevně zakončení tagu . Syntaxí řízený editor červeně zobrazí text , tedy začátek tagu, který nikde nekončí.

Příkladem kvalitního textového editoru, který disponuje kontrolou syntaxe různých programovacích jazyků, je i editor Emacs používaný ve své textové podobě především v operačním systému UNIX. Jeho kontrola syntaxe je nastavitelná podle potřebného programovacího jazyka, což většina syntaxí řízených editorů operačních systémů MS Windows neumožňuje, protože jsou úzce spojeny z celkovým vývojovým prostředím konkrétního programovacího jazyka (Pascal, C, Java), které příslušný výrobce prodává.

Tlak praxe je však neúprosný. Koho by zajímaly problémy kódování v programovacím jazyce, byť vyšší úrovně, a kdo by rád měnil bity v binárním souboru fotografií z rodinného výletu! Snaha poskytnout co nejvyšší pohodlí a zvyšovat tak výkon programátorů se promítá do nabídky produktů software, pomocí nichž lze provádět konstrukci výsledného programu bez nutnosti psát zdrojový kód. Obecným názvem takových systémů je CASE (Computer-aided software engineering) a jeho různé podoby mají za sebou dnes již nejméně dvacetiletý vývoj. V současném světě výpočetní techniky se jedná o inteligentní návrhové systémy pro výrobu software, kde programátor vytváří strukturu modelu

textové editory
řízené syntaxí

CASE, grafická
podpora vývoje
software

programu požadovaného praxí obecněji než je zápis v programovacím jazyce. Co má program modelovat a jakým způsobem má komunikovat se skutečností (obvykle člověkem) v běžné praxi zjišťuje *programátor analytik*. Výsledkem jeho analýzy je takový popis požadovaného software, který umožňuje samotnému programátorovi model kódovat. Výsledkem práce analytika je tedy podrobná dokumentace s řadou schémat, obrázků a matematických specifikací řešených problémů. Cílem CASE je umožnit vypracování takové dokumentace přímo v poskytovaném (grafickém) prostředí. Hlavní prací CASE je ale převoditelnost tohoto popisu do kódu programovacího jazyka. Vlastně je tak vyřazena mechanická práce programátora kodéra. Rovněž změny při vývoji aplikace v průběhu jejího používání v praxi se provádějí v CASE, který vyprodukuje novou verzi zdrojového kódu. Zdrojový kód je produktem, protože před samotnou kompilací do binárního kódu je často nutný ještě zásah programátora v místech, která CASE nedokáže zvládnout. Tyto návrhové systémy pro vývoj software jsou velmi drahé a pro mnoho potřeb praxe jsou stále nepoužitelné. Jejich vývoj a postupné nasazování je ale jistě jednou z alternativ budoucnosti Computer Science.

CASE vlastně trochu připomíná návrhové programy stránek WWW. Přestože jsou stránky WWW výrazně jednodušší (jde o stavbu zobrazování informací na základě prezentace textu a obrazu), jedná se i zde o produkci zdrojového kódu, zde pro interpret klienta stránek WWW. I zde do zdrojového kódu lze zasahovat v případě speciálních požadavků mimo návrhový program. Konstrukce stránek WWW i CASE představují předpřípravu samotného zpracování, produkci kódu algoritmického zpracování dat, tedy produkci zdrojového kódu.

textová varianta
vývoje software

skript

V rozporu s uvedeným příklonem ke grafické orientaci aplikací pro vývoj software stojí ryze textová podoba práce programátora na vývoji software. Myslím tím *programovací jazyky velmi vysoké úrovně*, lépe a častěji nazývané jako *skriptovací jazyky*. Jak jsme již uvedli, patří mezi ně dnes oblíbený Perl nebo Python. Původ skriptovacích jazyků lze vysledovat v operačních systémech textového charakteru, jejichž nejvýraznějším představitelem je UNIX. Zde se skriptovací jazyky, jako je např. *awk*, *sed*, *sh*, *m4* atd., používaly především pro úpravy textových souborů. Umí třídít, modifikovat, filtrovat, spojovat a porovnávat texty různých souborů a to tak, že pro ně programátor napíše poměrně krátký textový soubor direktiv, *skript*, který určuje práci s texty souborů na vstupu. Soubor direktiv je programem, jeho kód je ale mnohdy velmi koncentrovaný. Skriptovací jazyky mají dále velmi dobrou schopnost vzájemně se propojovat, takže pro určitou část zpracování dat lze použít jiný jazyk a obecně tak vytvářet tok zpracování dat vzájemně si data předávajících interpretů. Programování ve skriptovacích jazycích je proto neefektivnější v prostředí podpory obecně definovaného zpracování textu pomocí Tools. Takové prostředí není výmysl několika hackerů, ale bylo popsáno již na konci 60. let 20. století a později knižně publikováno v [KernighanPaluger1976]. Kernighan jako jeden z tvůrců operačního systému UNIX toto prostředí v UNIXu také implementoval a dále se podílel na jeho vývoji. Že se jedná o smysluplný produkt, potvrdil i zájem standardu

ISO pro operační systémy s názvem POSIX (viz [POSIX2003]), jehož verze již ze začátku 90. let přijala Tools jako závazný standard uživatelského a programátorského prostředí operačních systémů vůbec. Přestože implementace tohoto prostředí v mnoha operačních systémech kulhá (např. MS Windows, že), vzhledem k tomu, že skriptovací jazyky začaly využívat pro grafické zobrazování jednoduše ovladatelnou a současně výkonnou knihovnu Tk (viz www.tcl.tk/doc) skriptovacího jazyka Tcl, byly portovány do prostředí i jiných operačních systémů jako produkt mimo nabízený operační systém. To umožnilo textově programovat na vysoké úrovni pro různé operační systémy v tomtéž programovacím jazyce, jak jsme se již zmínili.

Na rozdíl od systémů typu CASE je vývojové prostředí skriptovacích jazyků v drtivé většině poskytováno zdarma.

Obyčejný uživatel ale na rozdíl od programátora obvykle nepotřebuje a ani nechce zobrazovat nebo editovat kód výsledku své práce. Píše-li sekretářka dopis spolupracující firmě, potřebuje mít k dispozici hlavičkový papír své firmy přímo na displeji a dále možnost psát text s různou velikostí, zvýrazněním nebo typem písma. Výsledný kód dopisu, který sekretářka buďto vytiskne a odešle papírovou nebo elektronickou poštou adresátovi, bude mít binární charakter. Kromě samotného textu bude binární soubor dopisu obsahovat také označení právě požadované velikosti a typu použitého písma, bude v něm uložen i obrázek loga hlavičkového papíru atp. To vše vyjádřeno pomocí 256 možných kombinací každého bajtu. Přestože je jisté, že jednotlivé bajty takového souboru můžeme skutečně binárním editorem číst a také měnit, podstatný však je zde kontext sledu více bajtů, který určuje daný prvek dopisu (nebo obecně dokumentu), jako je např. způsob zvýraznění textu nebo typ písma. Způsobu organizace výsledného binárního kódu v takových sekvencích bajtů, které určují jednotlivé jeho části a prvky, říkáme *formát dokumentu*. Formát dokumentu je dán použitým programem, a tedy výrobcem programu. Pokud sekretářka předává dokument v digitální podobě, pak ten, kdo jej bude chtít číst v prostředí svého počítače, musí mít tentýž program nebo alespoň program, který formátu dokumentu rozumí. Obecně hovoříme o datové kompatibilitě, protože obecně takový dokument nepředstavuje nic jiného než právě data.

editory dokumentů

formáty ukládání dokumentů

Snaha standardizovat formáty dokumentů je letitá, stanovit obecně závazný formát je však stále obtížné, protože praxe přináší neustále další požadavky na obsahy dokumentů a mnohdy se používaná syntax vyjadřující obecnou strukturu dokumentu dá na tyto požadavky aplikovat pouze obtížně a neefektivně. Navíc výrobce software, jehož editor dokumentů se hodně rozšíří mezi uživatele, často diktuje většinově používaný formát, a ten nemusí být právě dobře navržen. Každá nová verze programu přitom musí akceptovat i formáty svých předchozích verzí jednoduše proto, aby uživatel své dokumenty z dřívějších v nové verzi otevřel a mohl je dál rozšiřovat nebo měnit.

V současné době každý editor dokumentů pracuje ve svém formátu, nabízí ale převod (export dokumentu) do formátů jiných obecně známých, rozšířených a používaných editorů. Dokáže rovněž takové formáty akceptovat na vstupu

práce (import dokumentu), tedy umožní uživateli přijmout dokument v jiném formátu a převést jej do svého. Příkladem formátu psaných dokumentů je jistě MS Office (viz www.microsoft.com/office), který je v současné době snad jedním z nejrozšířenějších. Formát jeho přímé konkurence Open Office (viz www.openoffice.org) má z pohledu koncepce a následného vývoje ovšem nesporně vyšší kvalitu, navíc je úspornější a obecně zveřejněný. Každý z těchto balíčků programů pro práci v kanceláři ale obsahuje také možnost importu a exportu dokumentů jiných aplikací, přestože takové dokumenty mají jiný formát (z důvodů historických nebo konkurenčních), což práci uživatele usnadňuje. Do principů formátů dokumentů můžeme nahlédnout opět např. prostřednictvím kódování stránek WWW. Ve své podstatě se totiž jedná o příklad formátu dokumentu, zde na textové úrovni. Podobně jako se v HTML dá dnes již vyjádřit typ a velikost písma, umisťování do odstavců atd. a prohlížeč zobrazí stránku podle těchto direktiv, podobně pracuje i každý editor dokumentů, pouze neumožní zobrazit přímo samotný kód dat dokumentu ve svém vlastním formátu. HTML je tedy jedním z vlastních formátů práce s dokumenty. Pro označení vlastního formátu dat dokumentů aplikací používáme také obecný termín *vnitřní formát dat*.

vnitřní formát dat

grafické editory,
formáty ukládání
grafických dat

Podobně jako pracujeme s dokumenty, které jsou vyšší úrovní práce s texty kombinovanými s obrázky, můžeme pracovat i s formáty jiných digitalizovaných prvků světa kolem nás. Zcela jistě je to práce s obrazem, ať již statickým (fotografie, malba, kresba) nebo dynamickým (film, video, interaktivní simulace). Programům umožňujícím editaci obrazu říkáme *grafické editory*. Přestože se podrobněji způsobem kódování a manipulace s obrazem budeme zabývat v následujícím textu, jednoduše si lze představit kód obrazu opět jako řadu sekvencí bajtů, jejichž jednoznačně stanovené kombinace určují, co má prohlížeč takového formátu zobrazit. I zde rozlišujeme aspekt autorský (editor) a čtenářský (prohlížeč). Přestože dnes již existuje řada standardizovaných formátů obecně zobrazovaných na všech známých platformách (JPEG, TIFF, GIF atd.), grafické editory většinou tyto formáty pro samotnou editační práci s obrazem nepoužívají. Grafické editory pracují ve vlastním vnitřním formátu a jako výsledek umožňují export do formátů obecně stanovených. Adobe Photoshop používá formát EPS, Corel CDR atd.

Opět si musíme uvědomit, že skutečnost digitalizujeme na úrovni obrazové a textové. I text můžeme vnímat jako obraz a pak s ním tedy manipulovat pomocí grafického editoru. O takový přístup ale nestojíme, stále odlišujeme informaci obrazovou a textovou, protože textová informace může představovat efektivní kód našeho vyjádření s výrazným obsahem, který obrazem nelze vyjádřit. Že tomu tak je, prokazuje právě systém digitalizace používaný v Computer Science, který je postaven právě na znakovém kódu.

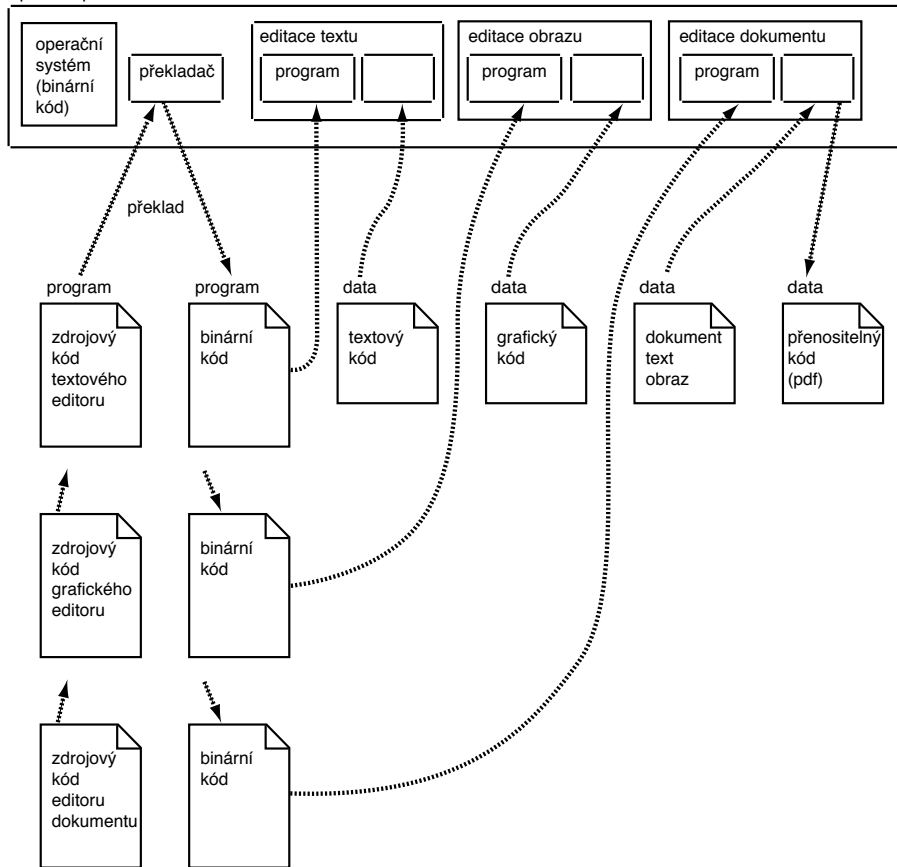
PostScript

Příkladem obecného kódu interpretace obrazového přístupu ke skutečnosti je formát používaný pro výstupní zařízení fotosázecích strojů a tiskáren s označením PostScript. Byl vyvinut již v době prvních software, které pracovaly se sazbou publikací určených k tisku na papír. Na jeho jednoznačné definici popisu výsledku, kterého mělo být na papíře dosaženo, se shodli

různí výrobci výstupního hardware. Software pracující ve svém vnitřním formátu exportoval v poslední fázi přípravy pro tisk podklady pro sazbu do formátu PostScript, který se sice nehodil pro další editaci, ale obecně byl srozumitelný kterékoliv tiskárně. Byla tak posílena možnost volby vydavatele, který mohl pracovat odděleně na sazbě a samotném tisku knihy. Kód ve formátu PostScript je mimo jiné také charakterizován výrazným navýšením svého objemu oproti kódu vnitřních formátů sázecích programů. S rostoucím zájmem o obecný formát PostScript pro výslednou prezentaci již nikoliv pouze na tiskových strojích, ale i na obrazovkách uživatelů vyvinula firma Adobe obecný formát kódování s označením PDF a uvolnila software s názvem Acrobat Reader, který jej interpretuje na obrazovku uživatele nezávisle na používané platformě. Obecně je Acrobat název software firmy Adobe, který pracuje s dokumenty přímo ve vnitřním formátu PDF. Každý software pro editaci dokumentů může na svém výstupu disponovat modulem pro převod do tohoto formátu. Principiálně se obvykle jedná o tiskový modul (výsledkem tisku není potištěný papír na tiskárně, ale soubor s daty ve formátu PDF), který dodává firma Adobe (s názvem Acrobat Distiller). Řada software dnes již ale nabízí i možnost exportu do tohoto formátu bez nutnosti nákupu a instalace tiskového modulu firmy Adobe (např. Open Office).

Kód (code) je základním principem fungování v Computer Science. Kresba III-2 stručně ukazuje princip používání tohoto kódu na úrovni programů, textů a obrazové informace. kódování

operační paměť



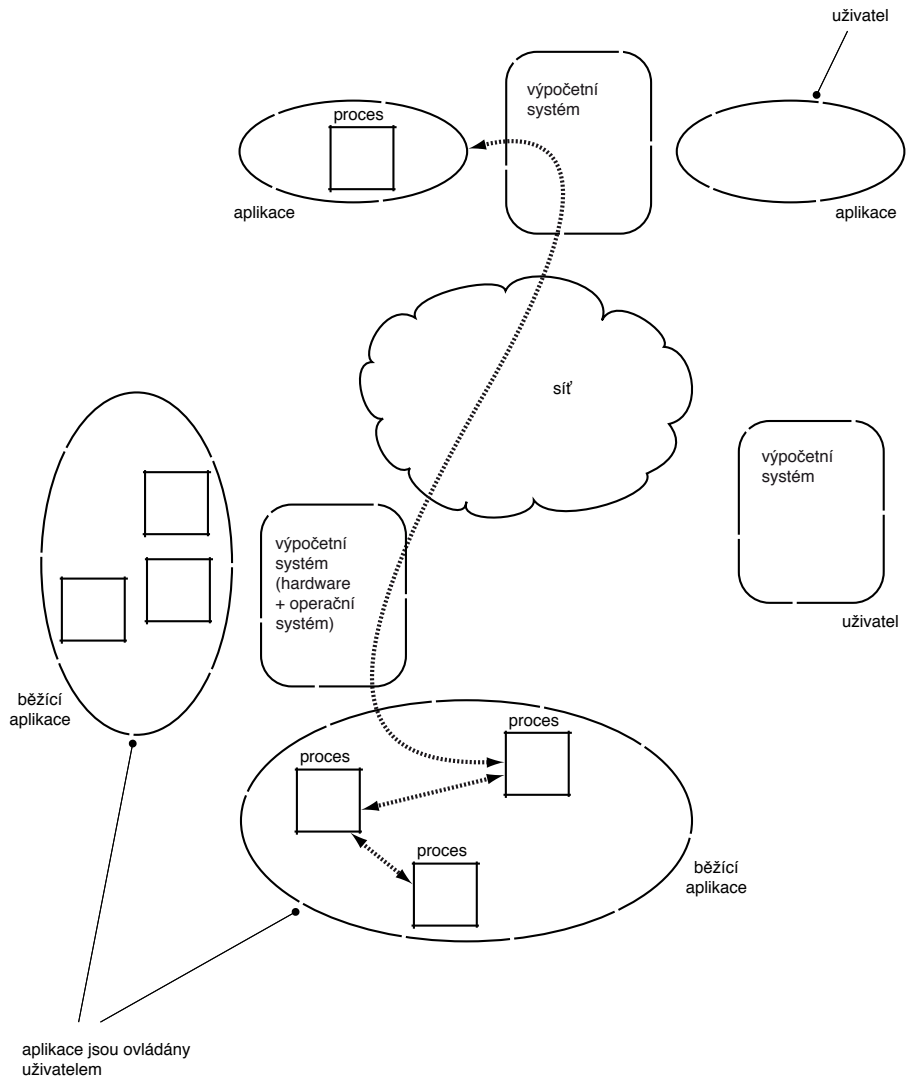
Kresba III-2: Kód jako nositel digitální informace.

III.2 – desktop, okna, grafické aplikace, grafické formáty, zvukové formáty, video, kompozice

Zpracování digitální informace probíhá v prostředí výpočetního celku, v nejjednodušším případě na výpočetní jednotce hardware podle Von Neumannova schématu řízené základním software určitého operačního systému na základě principů, které jsme se pokusili vyjádřit v předchozím textu. Uživatel tento fakt akceptuje a mnohdy je ochoten jeho principy i zkoumat, každopádně ho ale primárně nezajímá. Uživatele zajímá především aplikování jeho potřeb do prostředí výpočtu, což je práce programů realizujících algoritmy modelu reálného světa. Může se jednat o specializované požadavky, např. o podporu řízení tryskového letounu nebo výrobní linky, chirurgické operace atp. V mnoha případech se ale jedná o obecně požadované algoritmy práce s daty, jako je např. psaní textu, hraní her, práce s Internetem, kreslení nebo úprava digitální fotografie. K realizaci algoritmu může postačovat práce jednoho programu, často je ale zapotřebí více programů ve vzájemné součinnosti. K pojmenování celku realizujícímu požadovaný model říkáme *aplikace*.

aplikace

Aplikaci jako jeden program jsme již principiálně popsali v části Base Of. Obecněji se jedná o systém programů vzájemně spolupracujících podle požadavků uživatele. Prostředky vzájemné komunikace běžících programů jsou obvykle dány nabízenou podporou operačního systému a existuje i jejich standard (viz [POSIX2003]). Pro uživatele jsou však nepodstatné, protože se vlastně jedná o programovací techniky. Spuštěný program, běžící v prostředí operačního systému stroje, nazýváme proces. Běžící aplikace je tedy systém vzájemně spolupracujících procesů podle interakce s uživatelem. Např. editor dokumentů při kontrole gramatické správnosti psaného textu (spell checking) používá spolupracující program, který startuje až v okamžiku potřeby této kontroly jako nový proces. Běžící aplikace je tedy obvykle systém procesů, který se snaží efektivně využívat výpočetní zdroje bez zbytečné zátěže např. operační paměti nebo času procesoru na úkor jiných současně běžících aplikací uživatelů, kteří v tomtéž čase pracují na stejném výpočetním celku. Je jisté, že náhle hovoříme nikoliv o samostatné výpočetní jednotce, jako je např. PC, ale o využívání i jiných síťově dostupných výpočetních celků. Uživatel tak může ovšem pracovat s několika aplikacemi současně a ty ať už s jeho vědomím nebo bez něho mohou pracovat a spolupracovat v síti. Podívejme se na kresbu III-3.



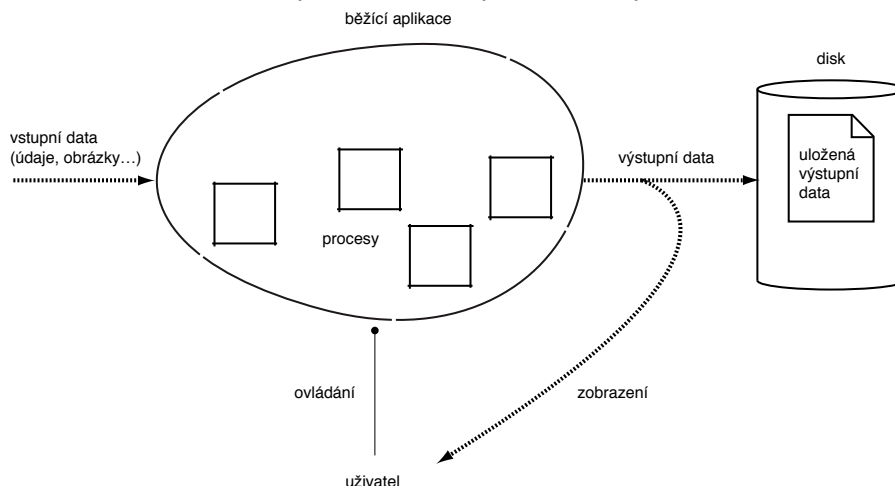
Kresba III-3: Uživatelé používají aplikace.

Vzájemná komunikace procesů aplikací na tomtéž stroji nebo v síti probíhá na základě předávání dat smluveným protokolem (aplikačním protokolem), tedy vzájemně srozumitelným kódováním předávaných informací. Rovněž pokyny uživatele jsou procesům předávány jako instrukce toku dat. Stisk klávesy je textový kód jednoho bajtu, posun a stisk myši nebo jiného ukazovacího zařízení rovněž. Výjimkou je zobrazování výsledků práce procesů uživateli, které může probíhat ve formě (i měnícího se) obrazu.

grafické aplikace

Víme, že běžící programy (procesy, a tedy běžící aplikace) přijímají vstupní data a produkují výstupní data. Výstupní data jsou ukládána na externích pamětech, případně jsou zobrazována na obrazovce uživatele.

Toto zobrazení výstupních dat může mít tvar srozumitelného kódu nebo formu obrazové informace. Podle zaměření aplikace na typ výstupních dat rozlišujeme aplikace grafické a ostatní. Grafická aplikace je zaměřena především na zpracování obrazu. Ostatní aplikace pracují s daty jako s určitým typem kódu a jejich výstupem je pak např. pokyn pro robotický prvek ovládání technologického procesu nebo důležitý údaj reálného světa v kontextu popsaného organizačního podsystemu vkládaného do databáze. Uživatelskou interakci a zpracování dat aplikací ukazuje kresba III-4.



Kresba III-4: Zpracování dat aplikací a uživatelská interakce.

Uživatel dává pokyn ke spuštění aplikace a její běh řídí. Aplikace přitom nemusí vždy použít digitální vstupní data z externí paměti. Uživatel může pomocí aplikace data pouze produkovat, např. při komponování hudby začíná uživatel svou práci bez výchozích dat. Rovněž interakce uživatele vstupujícího do chování aplikace může být minimální nebo žádná. V tom případě se ale jedná spíše o automatizované aplikace, např. cyklického vyhodnocování dat zaznamenaných snímači teplot nebo jiných měření. Výstupní data pak ukládáme na externí paměť k dalšímu použití v budoucnu. Většinou ale přítomný uživatel požaduje jejich zobrazení, protože on rozhoduje, zda jsou smysluplná. Často také uživatel nesleduje pouze úplný výstup, ale i určité mezifáze vývoje zpracování dat, které ovlivňují jeho další ovládání aplikace. Výstupní data uložená v externí paměti může uživatel později i zobrazit a dále je aplikací upravovat (slouží pak jako vstupní data). V dalším použití téže aplikace vznikají nové verze výstupních dat podle uživatelské vůle a on může rozhodovat, zda původní výstupní data budou nahrazena novou verzí nebo je postupně ukládá pod různými jmény s možností návratu např. k určité verzi své rozpracované hudební skladby, obrazu nebo dokumentu.

V ovládání běžících aplikací uživatelem dnes jednoznačně vítězí grafický princip. Uživatel má k dispozici obrazovku, ukazovací zařízení a klávesnici. Ukazovacím zařízením, jako je myš, touchpad, tablet, joystick nebo grafické

pero, si vybírá část obrázku a kliknutím jej potvrzuje jako vybranou aktivní plochu. Tento úkon je převeden do formy datového pokynu a po jeho zpracování získáváme jako výstup na obrazovku interpretaci ve formě proměny obrazu. Pomocí grafického editoru můžeme např. nakreslenou přímkou změnit jednoduše v křivku postupnou deformací, tedy tím, že popotahujeme za různé body na přímce. Jednotlivá popotahování si editor obvykle pamatuje jako postupné kroky v editaci a my můžeme kráčet zpět i vpřed v rámci již provedených úkonů a optimálně tak vyhledávat zamýšlenou variantu zakřivení. K popotahování přitom používáme ukazovací zařízení. Způsob manipulace s ním se přitom snaží vycházet uživateli vstříc do té míry, do jaké to technický chlad ovládacích páček, tlačítek a hmatem nedostupné obrazovky co nejlépe umožňuje. U práce s tabletem lze např. u dobrých grafických editorů navodit iluzi kresby uhlem, štětcem nebo pastelkou. Za výhodu přímé digitalizace naší představy platíme prozatím stále nepříjemnou vzdáleností od probíhajícího procesu, na rozdíl od přímého fyzického kontaktu našeho těla s měnící se hmotou kolem nás v průběhu obyčejné akce při našem dosavadním způsobu existence.

uživatelské
rozhraní

Přístup člověka k počítači byl pojmenován termínem *uživatelské rozhraní* (user interface). Už od svého vzniku byl tento přístup zatížen technokratickým pojetím, protože od samého počátku, vlastně od vzniku první realizace Von Neumannova schématu, jej po dlouhá léta spoluvytvářeli pouze lidé s technickým vzděláním a zaměřením. Tento princip byl po masovém rozšíření výpočetní techniky následně vnucen všem, kdo o digitální způsob vyjadřování projeví zájem. V drtivé většině tak došlo k obecné frustraci nových uživatelů netechnického zaměření, a to nejenom odborníků z humanitních a výtvarných oborů, ale i lidí, kteří vedou běžný plnohodnotný a všestranný život, technika je pouze jeho jednou a možná pouze dočasnou částí. I když existuje viditelná snaha uživatelské rozhraní stále více přibližovat přirozenému myšlení člověka, stěží se v dohledné době změní jeho již nastolené principy. Principiální proměna je velmi obtížná, protože s ní souvisí mnoho podstatných partií Computer Science. A tak většině uživatelů dnes nezbyvá nic jiného než se přizpůsobit, přestože to dělají často s nechutí a s tichým úžasem nad podivuhodným způsobem ovládání aplikací.

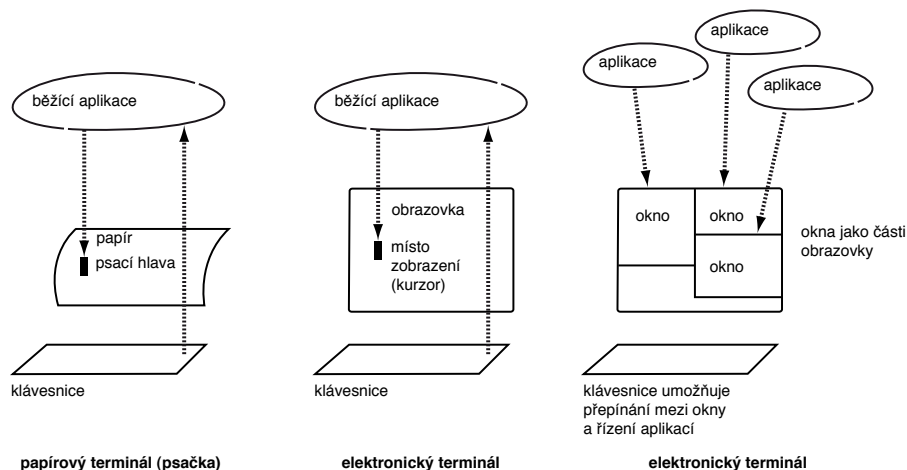
elektronický
alfanumerický
terminál

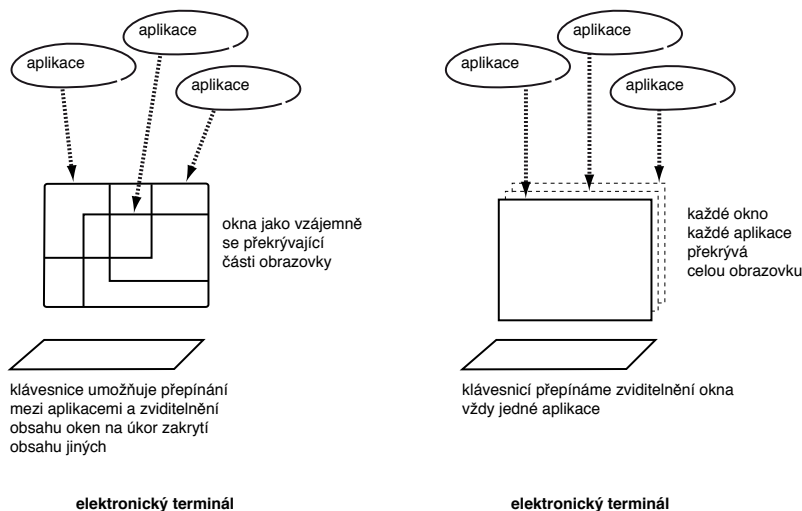
Okno (window) je prvek uživatelského rozhraní. Přestože jeho původ není v grafických systémech, je pro nás dnes hlavním prvkem při práci s grafickými aplikacemi. Okno bylo definováno jako část *alfanumerické obrazovky*. Alfnumerická obrazovka je zobrazovací plochou bajtů v kódu ASCII. Bajty jsou zobrazovány ve 24 řádcích a 80 sloupcích, obdélník obrazovky lze takto pokrýt celkem $24 \times 80 = 1920$ znaky textového vyjádření. Každý znak má na jednom z 1920 míst obrazovky svoji interpretaci, např. bajt s obsahem 01000001 je zobrazen ve tvaru písmena A (viz Base Of I.1 a Příloha A). Některé znaky celé škály ASCII nebyly na alfanumerických obrazovkách interpretovány, protože nemají pro lidské oko informační význam. Alfnumerické obrazovky vznikly jako elektronická podoba dálnopisů, což byla papírová komunikační zařízení (paper terminal, v českém žargonu psáčka), prostřednictvím nichž operátor v 60. a 70. letech 20. století komunikoval s počítačem jako

s jediným připojeným interaktivním terminálem a řídil tak průběh výpočtu. Tato systémová konzola (system console) se vbrzku stala vyhledávaným komunikačním zařízením. Jednalo se o řádkovou komunikaci. Uživatel napsal řádek požadavku pro počítač (příkaz, příkazový řádek, command line), odeslal jej stiskem klávesy Enter (tehdy označovanou jako carriage return, tj. návrat vozíku psacího stroje nebo psačky na první znak řádku) a počítač po zpracování odpověděl výpisem výstupních dat. Obrazovka se oproti papíru stala pohodlnějším prostředkem pro komunikaci, přestože se dialog vzájemné komunikace člověka a stroje po dosažení 24. řádku ztrácel. Především se šetřil papír popsaný často množstvím zbytečných informací, ale elektronická podoba psačky, tj. obrazovka s klávesnicí, navíc umožnila přesouvat umístění psací hlavy na různá místa obrazovky a tam přepisovat obsah znaku. Současné umístění výpisu znaku na obrazovce dostalo název kurzor (cursor) a vypisovaný znak začal mít i jiný význam než znaky abecedy přirozeného jazyka. Např. bylo možné aktivně používat znaky tabulace textu a zobrazovat výpisy běžících procesů v několika sloupcích, používat znaky šipek pro vyjádření např. vývoje dat atp.

Při vývoji operačních systémů a prohlubování významu interakce uživatele s počítačem se začal používat běh více procesů nebo celých aplikací současně řízených jedním uživatelem. K tomu byla řádková forma komunikace nepohodlná, zvláště když běžící aplikace začaly provádět výpis výstupních dat na obrazovku současně a data se tak nepříjemně promíchala. Vzniklo tedy rozdělování obrazovky na části, kterým se vzhledem k různým pohledům do dění v různých aplikacích začalo říkat okna. Na kresbě III-5 je zobrazen princip řízení více aplikací jedním uživatelem na jedné alfanumerické obrazovce.

okno





Kresba III-5: Psáčka, elektronický terminál a okna.

Jak ukazuje třetí a čtvrté schéma kresby, okna ihned po svém vzniku přestala být pouze striktně vymezenými částmi alfanumerické obrazovky, mohla se překrývat a dokonce měnit svůj rozměr od velikosti několika řádků a sloupců až po velikost celé obrazovky. Okna tak mohou být pohledem do běhu aplikací, přepínáním mezi jednotlivými okny aplikací můžeme měnit obsah celé obrazovky za jiný (za obsah jiného okna), jak je uvedeno na pátém schématu kresby. Pro přepínání mezi aplikacemi, tak aby klávesnice byla aktivní v požadovaném okně, se používaly speciální klávesy nebo stisk smluvené kombinace více kláves najednou.

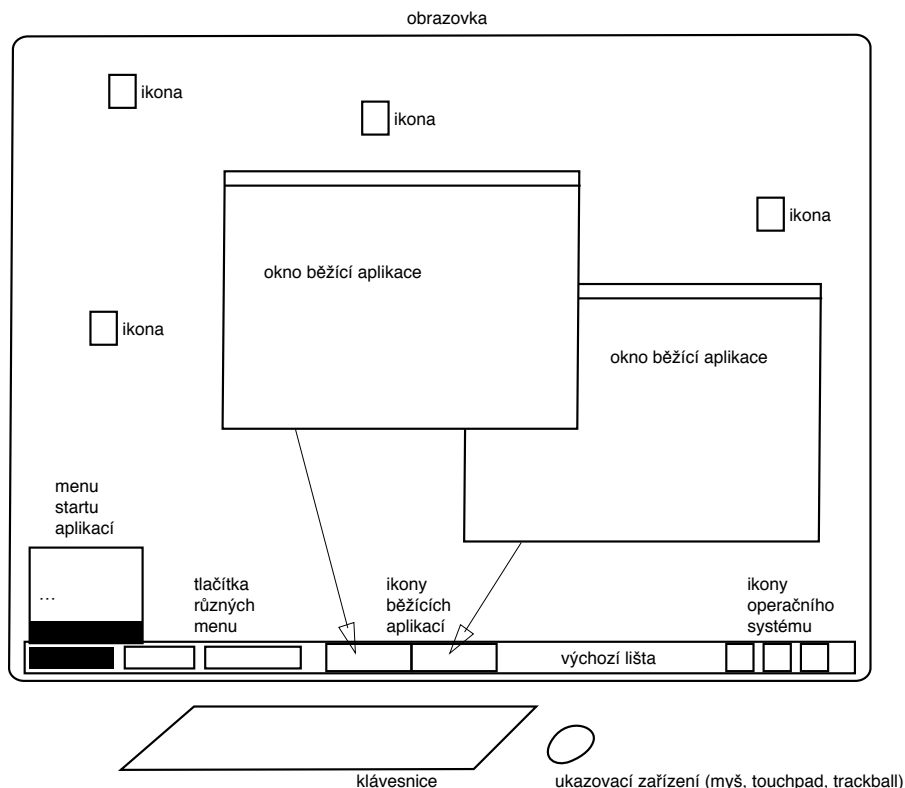
**grafický
elektronický
terminál**

Vznik *grafických elektronických terminálů* umožnil počítat velikost okna nikoliv v počtu sloupců a řádků, ale v počtu *pixelů*, tedy nejmenšího bodu barevného zobrazení a dále především umožnil vyjadřovat výstup aplikace v obrazové podobě a nikoliv pouze textové. Co se ale nezměnilo, byl princip pravoúhlosti oken a manipulace s nimi, jako je jejich zvětšování a zmenšování popotahováním za jejich strany a rohy nebo vzájemné překrývání. I když i alfanumerické terminály zobrazovaly určité bajty jako části rámečku oken (svislá čára, vodorovná čára, čtyři rohy), výrazné ohraničení oken grafických terminálů a jejich další části, např. nevyužitá plocha obrazovky, je přece jenom hezčí a svou variabilitou umožňuje spoustu legrace, přestože se často jedná o nepodstatné hříčky, které odvádějí pozornost od obsahu informací v oknech běžících aplikací.

desktop

Při vývoji uživatelského rozhraní pro grafický přístup k počítači se okna začala používat v rámci grafické aplikace, která je označována termínem *desktop* (pracovní plocha). Jedná se skutečně o jednu běžící aplikaci, která komunikuje s uživatelem pomocí grafických možností hardware. Tato aplikace je uživateli v okamžiku jeho přihlášení do operačního systému spuštěna ve výchozí podobě, která má skutečně připomínat psací stůl včetně

způsobu práce člověka na něm. Běh jednotlivých aplikací uživatel sleduje v jim přiřazených oknech. Na kresbě III-6 je schématicky uveden základní princip používaných prvků pracovní plochy.



Kresba III-6: Desktop a jeho prvky.

Grafická aplikace desktop překreslí obsah celé obrazovky. Obrazovka je pokryta *pozadím* (background), tedy v obyčejném případě je každý pixel obarven neutrální neodráždivou a na sebe neupozorňující barvou, podobně všední a nenápadnou, jako to požadujeme u našeho pracovního stolu.

V rámci desktopu se uživatel pohybuje pomocí ukazovacího zařízení, kterým je např. myš. Pohyb myši desktop sleduje a současně upozorňuje na umístění v rámci plochy, a to obvykle pomocí šikmé šipky, jejíž špička označuje místo (až na úrovni pixelu), kde může být vyžadována aktivita. Kromě možnosti měnit tuto polohu šoupáním po podložce disponuje myš nejméně jedním tlačítkem. Stisk tohoto tlačítka (je-li jich více, je to první tlačítko zleva) nazýváme *klik* (click) a představuje upozornění pro desktop, je to formulace uživatelské požadavku. To, o jaký požadavek se jedná, je dáno právě umístěním myši. Šipka sledující pohyb myši může být totiž umístěna na různých grafických prvcích, které desktop zobrazuje a jejichž umístění eviduje. Pokud klikneme pouze do prázdné plochy pozadí, přenesli jsme pozornost desktopu na toto pozadí, pro které není definována žádná akce.

click

Pokud ale pozornost desktopu byla předtím zaměřena na jiný grafický prvek, např. okno nebo ikonu, je mu tato pozornost odebrána, protože je převedena na pozadí. Kliknutí myši má obvykle charakter pouze změny pozornosti. Akci, která je s prvkem desktopu spojena, uživatel vyžaduje dvojklikem (double-click), tj. dvojnásobným stiskem tlačítka myši v krátkém časovém intervalu po sobě. Typickým příkladem akce, která je s ikonou jako prvkem desktopu spojena, je např. start aplikace. Desktop odlišuje klik a dvojklik na základě změření časového intervalu mezi dvojnásobným stiskem tlačítka. Začátečník je proto nucen si rozdílit mezi dvojnásobným kliknutím a dvojklikem nacvičit. Stisknu-li totiž dvakrát tlačítko myši v intervalu, který je delší než interval dvojkliku, desktop tyto stisky rozliší jako dva samostatné, což má za následek dvojnásobné přenesení těžké pozornosti do místa desktopu, a tedy se vizuálně pro uživatele nic nestane.

otevření okna

Aplikace desktopu startuje na základě uživatelského pokynu další aplikace a otevírá pro ně ve svém grafickém prostředí nová okna. Otevřením okna zde rozumíme vznik okna. Desktop do něj v okamžiku jeho otevření přenáší aktivitu ukazovacího zařízení. Uživatel proto může ihned komunikovat s nově spuštěnou aplikací. K tomu, aby mohl v další práci pokračovat nejenom v tomto okně, ale i v oknech jiných aplikací nebo obecně kdekoli v desktopu, je umístění myši dále monitorováno jednak v rámci aplikace okna a jednak v rámci celého desktopu. Umístěním ukazatele myši do okna aplikace a kliknutím předáváme aplikaci aktivitu ovládacích prvků myši a aplikace na tyto aktivity reaguje svým způsobem. Teprve až je kurzor přesunut mimo okno aplikace a pozornost je kliknutím přenesena na jiný prvek desktopu, je aktivita okna odebrána a řízení myši přebírá opět desktop.

Uživatel musí mít při práci možnost volby, musí mít k dispozici výběr startu různých aplikací. K tomu mu slouží především výchozí lišta. Na kresbě III-6 je uvedena jako vymezená dolní část obrazovky po její celé šířce, což je způsob operačního systému MS Windows. Často je ale lišta jednak kratší a jednak může být umístěna po celé délce naopak horní části obrazovky, jako je tomu v operačních systémech firmy Apple. Nebo může být výchozí lišta dokonce kdekoli v prostoru obrazovky ve tvaru obdélníku s tlačítkem s textem `start` nebo `klikni sem (click here)`, případně s textem `hlavní menu (main menu)`, jako je tomu u pracovních stanic v prostředí operačních systémů UNIX. A tak podobně. Výchozí tlačítko hlavní lišty (s textem `start...`) je tlačítkem pro rozbalení nabídky možností aplikace desktopu. Tyto možnosti jsou dány tím, co je uživateli definováno v rámci jeho práce v operačním systému. Jedná se především o start aplikací, úpravy pracovní plochy a ovlivňování běhu operačního systému.

menu

Menu (nabídka) je výběr z nabízených možností. Graficky se tento prvek desktopu chová jako výsuvný seznam možností, z nichž můžeme zvolit pouze jednu. Možnosti jsou vyjmenovány na několika řádcích obdélníku jednoduchým a výstižným textem. Uživatel je může využít tak, že přejíždí myší po řádcích menu. Desktop přitom každý řádek, přes který myš přejíždí, zvýrazní, tj. pozadí textu řádku ztmaví a text projasní. Uživatel je tak upozorňován, kterou možnost z menu by nyní volil. Volbu provádíme kliknutím

myši. Za položkou menu je ukryta akce, kterou desktop na základě volby provede. Menu často představuje dynamicky se objevující prvek desktopu. Obvykle je vysouváno z různých textů lišt, z ikon a tlačítek. Uživatel tak může např. z nabídky programů tlačítka hlavní lišty vybrat ten, o kterém si myslí, že je potřebný pro jeho práci.

Vedle oken běžících aplikací, výsuvných menu a lišt je prvkem pracovní plochy také *ikona*. Ikona ve tvaru malého obrázku je zástupný prvek pro provedení určité akce, kterou běžně provádíme pomocí menu. Zástupnost spočívá v propojení ikony s určitým výpočetním zdrojem operačního systému, který se tak aktivuje. Např. dvojklikem na ikonu uživatel spouští aplikaci a její start se projeví vznikem nového okna, ve kterém dochází ke komunikaci aplikace s uživatelem. Ikona ale může obecně zastupovat výpočetní zdroj. Znamená to, že za ikonou se neskrývá pouze start aplikace v novém okně, ale odkaz na zdroj dat, která jsou označena pro zpracování určitou aplikací. Např. může ikona na ploše zastupovat data kartotéky osob. V takovém případě dvojklikem nad ikonou spouští desktop aplikaci, která tuto kartotéku dokáže zobrazit a dále s ní pracovat na základě uživatelského ovládání. Rozdíl je v tom, že nestartujeme pouze aplikaci, ale typ dat sám určuje aplikaci, která je startována s těmito daty na vstupu. Obrázek ikony je symbolem aplikace, se kterou je ikona spojena, a to ať už odkazem na samotnou aplikaci nebo na data aplikace.

ikona

Uvedené možnosti práce uživatele na pracovní ploše vycházejí z principu manipulace uživatele s průběhem současně několika běžících aplikací. Důležitá je tedy také manipulace s otevřenými okny běžících aplikací. Každé okno má typizované ohraničení, v rámci daného typu desktopu je ohraničeno čarou určité barvy. Horní ohraničení okna je tlustší. Jedná se o *záhlaví okna* a je barveně odlišeno. Za záhlaví můžeme okno myši uchytit a přemístit je do jiné části pracovní plochy. Uchytit okno znamená stisknout levé tlačítko myši a držet jej za pohybu přesunu myši (drag). Okno se přemisťuje, dokud levé tlačítko myši nepustíme (drop). Uvedená metoda *drag and drop* (táhni a pusť) je jednou ze základních metod ovládání desktopu (doposud jsme používali termín popotahování myši). Touto metodou totiž můžeme přemisťovat nejenom okna, ale i ikony a všechny další prvky a části prvků desktopu vůbec (lišty nevyjímaje), podobně jako vezmeme do ruky předmět na stole a položíme jej na jiné místo. Metodou drag and drop uzpůsobujeme desktop svým potřebám a zvykům.

drag and drop

Kliknutím myši na záhlaví okna aktivujeme přístup k jeho aplikaci. Protože i celé záhlaví okna může být překryto jinými prvky desktopu (zejména jinými okny), často u většiny desktopů stačí, aby uživatel klikl na jakoukoliv část ohraničení okna nebo do obsahu okna. Protože ale může být celé okno aplikace překryto jinými okny, každý desktop nabízí uživateli místo, ve kterém jsou všechna okna evidována. Někdy se jedná o evidenci ve výsuvném menu (Apple, Unix), někdy jsou okna evidována jako tlačítka základní lišty (MS Windows), jak je uvedeno na kresbě III-6. Kliknutím na výběr z menu nebo na tlačítko přesuneme odpovídající okno aplikace do *popředí* (foreground),

foreground

objeví se tedy před ostatními prvky desktopu. Takto lze získat kontrolu nad všemi běžícími aplikacemi a dokonce i tehdy, je-li každá z nich zobrazována v okně, které překrývá celý desktop. Dochází tak k situaci, kterou zobrazuje páté schéma kresby III-5.

uzavření okna

Záhlaví okna obsahuje několik tlačítek. Obvykle na pravém konci jsou umístěna tlačítka rychlé manipulace s velikostí a existencí okna. Např. v MS Windows kliknutím na tlačítko s obrázkem čtverečku dosáhneme skokem zvětšení okna na největší možnou velikost. Okno využije celou obrazovku s výjimkou základní lišty. Opětovným stiskem tohoto tlačítka získáme jeho původní velikost. Původní velikost je dána výchozím nastavením aplikace a nebo tím, jak jsme velikost okna změnili taháním myši (metodou drag and drop) za jeho strany a rohy. Tlačítko se znakem křížku (nebo tečky) je umístěno poblíž na základní liště okna a slouží k uzavření okna. Uzavřením okna zde rozumíme jeho zánik. Uzavřením okna ukončíme aplikaci vlastně nestandardní cestou, protože tím, že aplikace ztrácí prostor pro komunikaci s uživatelem, ztrácí svůj vstup a výstup. Přestože tomu tak striktně nemusí být, aplikace skutečně na základě uzavření okna uživatelem končí svoji činnost. U dobře napsaných aplikací ještě následně dochází k dialogu aplikace s uživatelem, zda je jeho požadavek na násilné ukončení běhu legitimní, zda se nejedná o náhodný klik, zda mají být případná rozpracovaná data z operační paměti přepsána také na externí paměť (protože jinak jsou ztracena) atp.

Všechny způsoby manipulace uživatele s oknem jsou často ukryty za tlačítko v levé části záhlaví okna. Obvykle se jedná o tlačítko s pomlčkou nebo obrázkem symbolu aplikace. Klikneme-li na něj, získáme menu, ve kterém jsou obsaženy všechny dostupné možnosti manipulace s oknem, tj. jeho maximální zvětšení, změna velikosti, zmenšení pouze do ikony, uzavření atp. Zmenšení pouze do ikony znamená, že okno není na desktopu vůbec vidět a je dostupné pouze prostřednictvím tlačítka na základní liště s ikonou aplikace nebo v menu seznamu oken aplikací.

hot keys

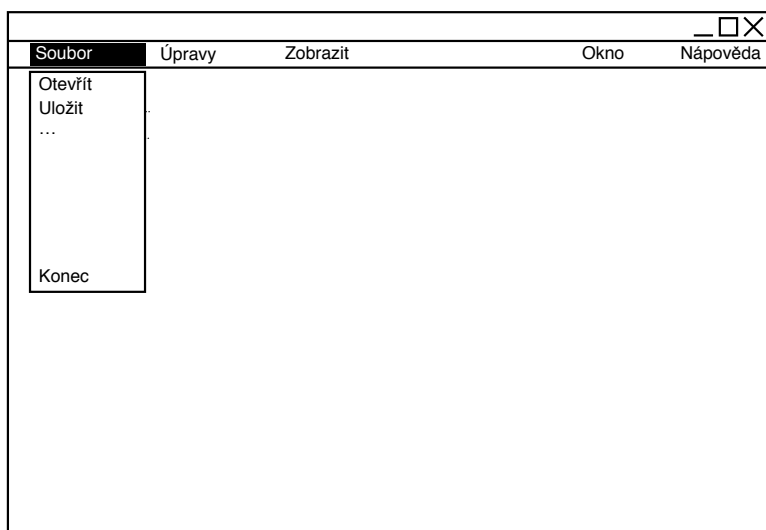
Každý desktop byl vždy koncipován tak, aby jej bylo možné ovládat i bez ukazovacího zařízení, tj. pouze pomocí klávesnice. Základní ovládací klávesy jsou šipky, tabulátor, mezera a speciální klávesy, jako je Ctrl, Alt, klávesa s jablkem, klávesa s oknem, klávesa s menu. Jedná se spíše o nouzové ovládání, u každého desktopu je ale vždy plně zachováno. Je to ovšem i z toho důvodu, že pokynem pro desktop z klávesnice lze rychleji dosáhnout požadované akce. Jedná se o klávesové zkratky (hot keys), které mají v oblibě nejenom programátoři, teenageři a hackeři, ale i zkušení uživatelé. Bohužel, klávesové zkratky většinou nejsou dokumentovány v souhrnné podobě a mezi uvedenými skupinami uživatelů se šíří většinou osobní radou. Seznam klávesových zkratk některých operačních systémů uvádí Příloha E.

pop-up menu

Obsah okna běžící grafické aplikace, tedy způsob zobrazování dat a reakce na vstupní pokyny, je zcela v kompetenci programátora aplikace. Jsme proto často svědky různého chování aplikací různých výrobců, a to mnohdy zcela podstatným způsobem. Bohužel se každý, kdo má větší úspěch s prodejem

software, snaží poznamenat svět svým vlivem, namísto kázně a využití již mnohdy velmi dobře definovaných způsobů ovládání. Přesto se časem vyvinul, jednak vlivem používaných vývojových prostředí a jednak vlivem stylů mateřských aplikací úspěšných desktopů, určitý jednotný způsob programování uživatelského rozhraní aplikací.

Výchozí okno spuštěné aplikace obsahuje základní lištu umístěnou pod záhlavím okna po jeho celé délce. Texty na této liště jsou výchozími místy různých menu. Menu, které je skryto za určitým prvkem desktopu (zde za textem na liště) a objevuje se kliknutím na tento prvek, je nazýváno *výsuvné menu* (pop-up menu). Texty na liště a jejich pop-up menu odpovídají všem možným akcím práce s aplikací. Kategorizují přitom její ovládání tak, jak ukazuje kresba III-7.



Kresba III-7: Výchozí lišta okna aplikace.

Texty uvádějící menu na kresbě jsou obecně používány. V prostoru mezi *Zobrazit* a *Okno* dále bývají umísťovány texty pro menu, která jsou specifická pro danou aplikaci.

Práci se vstupem a výstupem dat umožňuje menu za textem *Soubor* (File). Menu obsahuje především možnost pro načtení dat z externí paměti disku, a to výběrem textu *Otevřít* (Open). Jak již dobře víme, spuštěná aplikace je umístěna v operační paměti, a k tomu, aby mohl uživatel pracovat s daty, je musí do aplikace načíst, tj. přemístit je z diskové paměti do operační. Přestože jsou data v průběhu práce měněna pouze v operační paměti, aplikace udržuje spojitost se jménem souboru na disku, ze kterého byla data do operační paměti okopírována. Pro aktualizaci dat na disku, tedy přepis současného stavu dat v operační paměti do tohoto souboru, slouží položka menu *Uložit* (Save). Jedná se o nutné povědomí uživatele o práci nad kopií dat v operační paměti a její aktualizaci na externí paměť, kde jsou data uložena i po ukončení aplikace, zatímco je operační paměť uvolněna pro jinou práci a je dynamicky přepsána

save as

jinou aplikací a jinými daty. Některé aplikace tuto aktualizaci dat provádějí automatizovaně vždy po uplynutí určitého času (např. každých 10 minut). Obecně je ale výhodnější data aktualizovat vědomě. Položka menu `Uložit jako` (`Save as`) totiž dává možnost zpracovaná data uložit do jiného souboru, než ze kterého jsme je původně načetli. Běžně takto pracuje úřednice, když k napsání dokumentu použije některý již dříve sepsaný dokument, otevře jej, načte jej do aplikace, změní jméno souboru pomocí `Uložit jako` a dokument pouze upraví. Vznikla tak kopie výchozího dokumentu pomocí jeho otevření v aplikaci a změny směru výstupu dat. S aplikací můžeme pracovat také bez existence vstupních dat. Pokud začínáme kreslit nový typ obrázku, nemáme žádná výchozí data, neotevíráme žádný soubor. V menu `Soubor` je položka `Nový` (`New`), pomocí které otevíráme prostor pro práci od začátku. Vznikají tak data v operační paměti, která nesouvisí s žádným souborem diskové paměti. V rámci struktury systému souborů jsou tato data Nepojmenovaná (`Untitled`, `Unnamed`), jak je aplikace také eviduje. Uživatel by měl dbát na to, aby pro tato data brzy určil jméno souboru a jejich umístění na disku (možností `Uložit jako`), aby v průběhu práce mohl provádět jejich přepis na externí paměť. V případě kolize operačního systému nebo výpadku elektrického proudu jsou totiž bohužel tato data ztracena. Pokud to uživatel neudělá v průběhu své práce s daty a jméno souboru a jeho umístění na disku neurčí, aplikace ho při ukončování práce k této činnosti vyzve a varuje ho před nebezpečím ztráty dat.

Položkou menu `Soubor` je také `Tisk` (`Print`) dat, `Náhled` (`Preview`) dat před tiskem, `Vlastnosti` (`Options`) tisku a `Konec` (`Exit`), který jako obvykle poslední položka menu `Soubor` znamená ukončení aplikace.

clipboard

Menu, které se vysouvá kliknutím na text `Úpravy` (`Edit`) výchozí lišty aplikace, obsahuje kategorii základních editačních manipulací s daty. Pro začátečníka se jedná o záhadu, protože by na tomto místě očekával nabídku možností orientovanou na zaměření aplikace, ale není tomu tak. Základem je označení dat. Např. kus textu dokumentu označím tahem myši se stisknutým (levým) tlačítkem přes požadovaný text. Po uvolnění tlačítka myši zůstane tento vybraný kus textu zvýrazněn (obvykle text zbělá a jeho podklad ztmavne). Označená data jsou takto celkem určeným pro manipulaci. Můžeme je jedním stiskem klávesy `Delete` smazat. Můžeme je ale také zdvojit, tedy vytvořit jejich kopii. V menu `Úpravy` vyhledáme položku `Kopírovat` (`Copy`) a potvrdíme ji. Zdánlivě se nic neděje, protože vizuálně se nic nezměnilo. Označený text byl ale uložen do schránky (clipboard) v rámci datového prostoru desktopu, který není běžně viditelný. Data ve schránce jsou připravena být vložena do námi dále určeného místa. Toto určení provedeme nastavením kurzoru v datech aplikace myši a kliknutím a nebo pohybem kláves se šipkami. Důležité také je, že v průběhu této práce zvýrazněný text, okopírovaný do schránky, své zvýraznění ztratí. Použitím položky menu s textem `Vložit` (`Paste`) text ze schránky kopírujeme do určeného místa. Akci vložení můžeme přitom opakovat několikrát po sobě, protože schránka vložním jejím obsahem do určeného místa není vyprázdněna. Data tak můžeme kopírovat několikrát za sebou. Další položka menu `Úpravy` má textové označení `Vyjmout` (`Cut`). Možnost `Kopírovat` označená data do schránky okopírovala, `Vyjmout` je do

schránky přesune, tedy z okna aplikace zmizí. Přesunem do jiného místa v datech a použitím možnosti *vložit* tato data ze schránky kopírujeme na jejich nové místo. Data jsme takto vlastně přemístili jinam.

Obecně je metoda práce se schránkou nazývána *cut and paste* (vyjmi a vlož), doslova jako bychom na pracovním stole vystřihli kus papíru a nalepili jej na jiné místo. Metoda cut and paste má v rámci desktopu obecný charakter. Tím je myšleno, že data takto můžeme přesouvat nikoliv pouze v rámci aplikace, ale také navzájem mezi různými aplikacemi. Text z dokumentu v aplikaci Word firmy Microsoft tedy mohou okopírovat pro popis obrázku v kreslícím software firmy Adobe, obrázek nebo fotografii z prohlížeče stránek WWW Internetu mohou vložit mezi své poznámky v aplikaci Word atp.

cut and paste

Důležitou položkou v menu *Úpravy* je *Zpět* (*Undo*). Aplikace totiž běžně od startu do svého ukončení udržují historii úprav dat. Znamená to, že nejsem-li spokojen s právě provedenou úpravou dat (opravou textu, změnou formátu obrázku, prosvětlením fotografie), mohu se vrátit k variantě, ve které byla data před ní. Aplikace přitom běžně umožňují návrat v několika krocích. Kráčet přitom můžeme jak *Zpět* tak i *Vpřed* (*Step Backward*, *Step Forward*), dokud nevyčerpáme počet uložených provedených změn.

Součástí menu *Úpravy* jsou také položky umožňující vyhledávání ve zpracovávaných datech (položka *Hledat*, *Find*) nebo jejich vyhledání a výměnu za jiná (*Výměna*, *Replace*). Hledat i vyměňovat můžeme opakovaně, a to potvrzováním (nebo zamítáním) v nově vytvořeném okně s odpovídajícími tlačítky.

Menu za textem *Zobrazit* (*View*) umožňuje uživateli nastavovat prostředí práce v aplikaci. Např. lze přidávat další ikony do prostředí aplikace jako zkratk akcí z různých menu, což urychluje uživateli práci. Součástí menu jsou také položky pro přibližování nebo naopak zvyšování odstupu od dat (*Zoom in a Zoom out*), tj. využívání lupy při zobrazování dat. Je důležité např. přiblížit detail kresleného obrázku, abychom čáry a křivky spojili co nejpresněji. Součástí menu *Zobrazit* bývají i možnosti zviditelnění některých běžně skrytých částí dat, např. záhlaví stránky dokumentu nebo naopak možnost zobrazení pouze dat bez jakýchkoliv dalších menu, lišt a ikon okna aplikace (*Celá obrazovka*, *Fit on Screen*).

Uživatel může mít v rámci jedné aplikace rozpracována data více souborů, tj. může např. upravovat několik dokumentů současně. Jednotlivá data aplikace zobrazuje v různých oknech, ta jsou ale její součástí, přestože se chovají jako samostatně běžící aplikace. Použitím *Konec* v menu *Soubor*, tj. ukončením aplikace, zanikají všechna tato okna. Přesouváním pozornosti do jednotlivých oken těchto dat uživatel střídá práci např. na jednotlivých dokumentech (používá třeba metodu cut and paste). Organizace rozmístění a změny velikosti těchto oken jedné aplikace, případně střídání těchto oken (zviditelňování pokud jsou zcloněna jinými okny) dosahujeme využíváním menu za textem *Okno* (*Window*).

Konečně menu za textem *Nápověda* (*Help*) je věnováno *systému přímé nápovědy* (on-line help) provozované aplikace. V jeho položkách najdeme

on-line help

možnosti doprovodného textu k uživatelskému ovládání aplikace v tématech, v požadovaném kontextu nebo podle abecedního seřazení pojmů aplikace (rejstřík, index). Přestože systém přímé nápovědy bývá často velmi rozsáhlý, uživatelé jej čtou jen málo. Snaží se totiž aplikaci ovládat především intuitivně a často na sáhodlouze popisované partie aplikace v nápovědě přijdou sami. Vyhledávají většinou až speciality, které nemohou pohybem v menu objevit. Ty však v nápovědě často nenaleznou (byť tam většinou jsou, ovšem utopeny v balastu banalit).

Mezi menu *Zobrazit* a *Okno* bývá vloženo několik dalších menu, která jsou specifická pro konkrétní aplikaci. Jejich textové označení bývá stanoveno výrobcem software, přestože se mnohdy výrobce u software se stejným zaměřením snaží používat známé texty a podobnou strukturu odpovídajících menu. Tato menu jsou pak pro obsah práce s aplikací podstatná. Ostatní (zde uvedené) menu souvisí více se způsobem práce v desktopu a v operačním systému.

typy desktopů

Uvedený způsob ovládání aplikace v okně grafického prostředí pracovní plochy uživatelem není jediný možný ani jediný existující a nemá dokonce ani vývojové prvenství. Byl zde uveden, protože je nejvíce rozšířený a uživatel se s ním setká jako s prostředím na nejlevnějších sestavách s grafickým rozhraním, tedy na platformě PC v operačních systémech MS Windows. Vážným konkurentem je stále grafické prostředí uživatele na počítačích firmy Apple s označením Macintosh, z něhož firma Microsoft svého času také čerpala inspiraci při konstrukci svého prostředí desktopu. Přestože se pro uživatele jedná v řadě ovládacích prvků o jiný princip myšlení, základní popsané metody práce (cut and paste, drag and drop, clipboard, plocha, ikona atd.) jsou tytéž. Odlišná je práce v prostředí samotné aplikace v oknech, která vlastně vždy překrývá dříve spuštěné aplikace a jiný je i způsob přepínání se mezi takto běžícími aplikacemi. Bystrý uživatel ale dokáže tyto odlišnosti rychle pochopit a naučit se pracovat i v tomto prostředí.

Způsobů grafického rozhraní práce uživatele při komunikaci s pracovní plochou a aplikacemi provozovanými v systému oken je celá řada. Jistě bychom neměli zapomenout na firmu IBM, která se na vývoji v tomto smyslu výrazně podílela, přestože výsledky jejího výzkumu byly často s obchodním úspěchem aplikovány jiným výrobcem. Architektura prvního PC pochází od firmy IBM a donedávna každé PC s procesorem Intel bylo v textu uváděno jako IBM PC kompatibilní. Rovněž první jednoduchý operační systém na této architektuře byl jako zakázka od IBM realizován firmou Microsoft. Firma IBM se také pokusila už v době používání systému oken na alfanumerických terminálech navrhnout obecný způsob uživatelského přístupu a jeho popis zveřejnila. Dokument definoval již tehdy vznikající systém oken a dokonce i systém pop up menu základní lišty pod záhlavím okna aplikace, a to ve smyslu typů menu popsaných v předchozím textu.

Pro uživatele je ale lhostejné, kdo dosáhl obchodního úspěchu, kdo si zasluhuje prvenství a kdo by měl být majitelem autorských práv. Uživatele

zajímá především co nejpodobnější nebo dokonce shodný způsob ovládání jeho desktopu.

Grafický podsystém X Window System (zkráceně jen X) byl vyvinut především pro operační systém UNIX a v části Connect To II.2 a II.3 jsme se o něm již zmínili. Vznikl v polovině 80. let minulého století především v MIT (Massachusetts Institute of Technology), ale významnou měrou se na něm podílela i řada komerčních firem počítačového světa, jejichž grafické stanice pracovaly pod UNIXem (byla to např. firma Sun nebo firma DEC, kterou později pohltila firma Compaq a tu pak zase Hewlett Packard), ale i řada akademických a vůbec nezávislých specialistů v Computer Science. Na rozdíl od již uvedených systémů oken uživatelova rozhraní byl X Window System podporován sofistikovaným operačním systémem tak, jak byl uveden v části Base Of I.3. Další odlišující se vlastností byla flexibilita při programování aplikací. Tím je myšlena podpora možnosti různého stylu grafického návrhu aplikace nebo dokonce celého desktopu, a to na různých úrovních, které jsou přesně vymezeny. Pokud programátor přijme grafický styl (např. Motif, OPEN LOOK, Tk atd.), aplikaci programuje sestavením z již připravených grafických prvků (okna, tlačítka, atd.). Může ale přitom také vybudovat prostředí vlastní a nabídnout je k použití jiným programátorům. Konečně, X byl konstruován na podobném principu jako operační systém UNIX, tj. především pro docílení nezávislosti na platformě hardware, u X ale i nezávislosti na operačním systému. X je tak možné provozovat v každém UNIXu, ať již se jedná o AIX firmy IBM, IRIX firmy SGI nebo Solaris firmy Sun. Je ale provozován i v operačním systému VMS firmy DEC, který není UNIX. Z těchto hlavních uvedených důvodů se mohlo grafické prostředí operačního systému LINUX na platformě PC vizuálně přiblížit desktopu MS Windows, nebo mohl být grafický systém oken firmy Apple přeprogramován do UNIXu s označením Mac OS X bez patrných změn pro uživatele.

X Window System

Jisté je, že vývoj X musel být finančně velmi nákladný. Také jeho prodejní cena byla ještě na konci minulého století výrazně vyšší než byla cena MS Windows, což jistě rozhodování uživatelů ovlivňovalo.

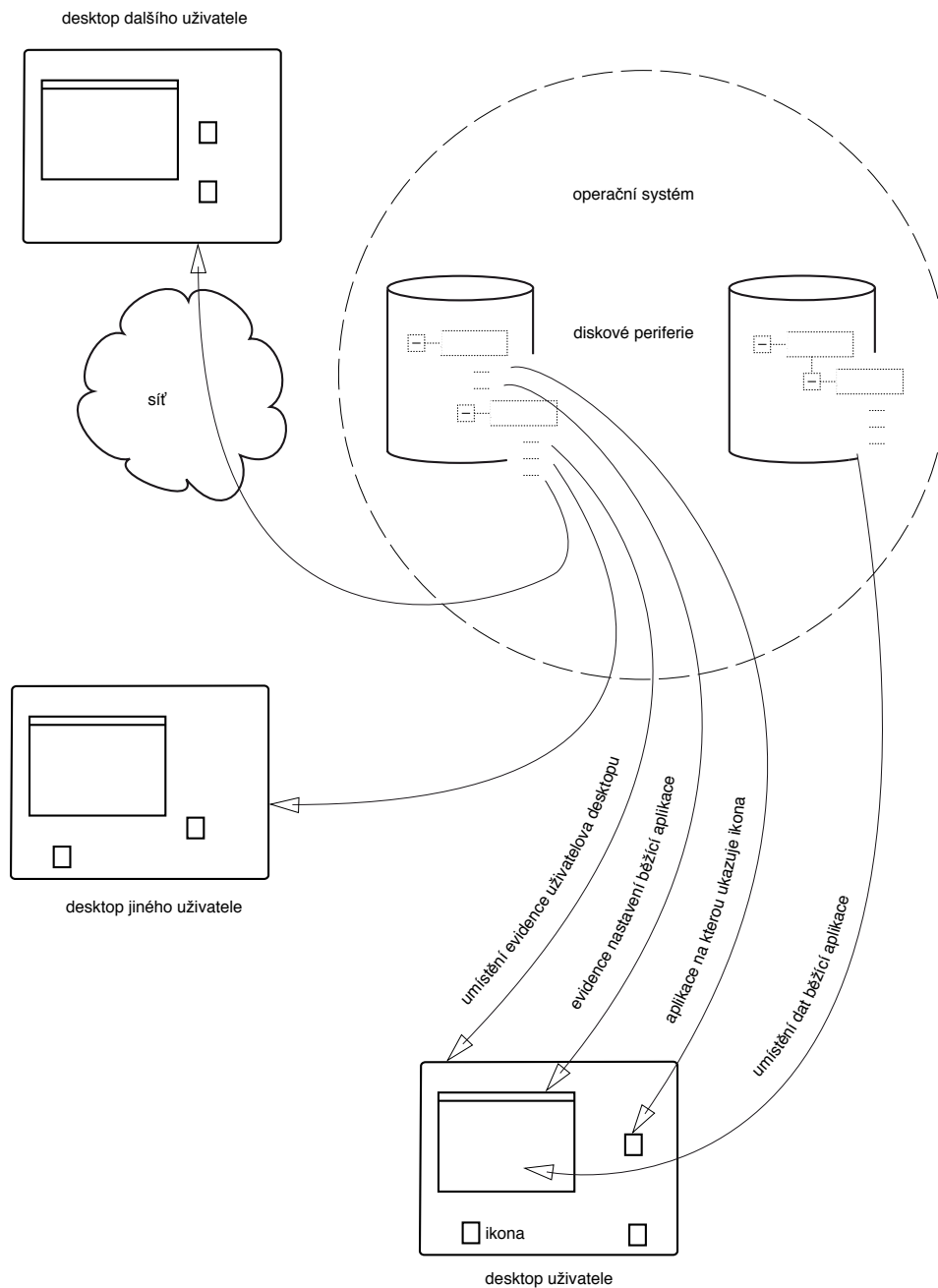
Firmy produkující pouze systém oken začaly ve své době nazývat tento software termínem grafický operační systém, protože např. MS Windows byl v té době skutečně pouze jeden grafický software, který běžel prakticky přímo na holém stroji a zajišťoval všechny potřeby, které jsou očekávány od operačního systému. Tento rys nevzdělaných programátorů se ale nevyplatil z řady důvodů. Software zaměřený především na obsluhu myši a obrazovky bez vnitřní struktury a vrstevnosti často chybuje, hrouť se a v mnoha případech je zbytečně pomalý. S nutností zajistit práci v sítích, bezpečnost dat a podporu střídajících se uživatelů u téhož PC nebo dokonce současně pracujících na tomtéž serveru pak takový grafický operační systém prakticky nelze provozovat. Také proto firma Microsoft výrazně investovala do vývoje operačního systému tzv. nové technologie (New Technology, NT), který se chová uživatelsky podobně jako předchozí Windows 95, 98, ME atd., z pohledu vnitřní struktury má však charakter sofistikovaného

operačního systému. MS Windows NT firma Microsoft dokonce klonovala i na hardware pracovních stanic např. firmy SGI nebo Hewlett Packard, její dnešní zakončení vývojové řady (MS Windows XP) je ale určeno prakticky výhradně pro platformu Intel PC. Nezávislosti grafického podsystému na operačním systému podobně jako je tomu u X zde ale nebylo dosaženo a pravděpodobně ani nebyla v plánu.

Přestože v laboratořích dalšího vývoje grafického prostředí v Computer Science lze zhlédnout i řadu jiných variant, druhou nejrozšířenější platformou podpory grafického podsystému je u řady komerčních i nekomerčních výrobců operační systém UNIX, a to právě díky uvedeným hlavním vlastnostem software grafického podsystému X.

desktop uživatele

Uživatel grafického prostředí přichází k obrazovce stolního PC, pracovní stanice nebo grafického terminálu výkonného serveru a do formuláře o dvou kolonkách zobrazeného v jednoduchém okně zapisuje své přihlašovací jméno a heslo, kterým je jeho přístup k pracovnímu prostředí chráněn. Operační systém čte z uživateli vyhrazené části disku nastavení pracovní plochy a na obrazovce vystaví výchozí stav. Výchozím stavem rozumíme rozmístění ikon, lišt, jejich velikost a vzájemné vztahy a vazby. Je obnoven z naposledy použitého přihlášení uživatele, který tedy používá své vlastní nastavení pracovní plochy tak, jak mu nejlépe vyhovuje. Oproti tomu běh aplikací v otevřených oknech uživateli běžně nastaven nezůstává. Znamená to, že pokud uživatel ukončí své sezení, tedy odhlásí se v průběhu práce aplikací a pokud nepoužije zvláštní postup, jsou tyto aplikace ukončeny operačním systémem a zpracování dat není aplikacemi dokončeno. Hlavní důvod takového ukončení je v principech chování operačního systému. Aplikace zpracovávající data totiž obvykle potřebují pro dokončení práce také obrazovku a ovládací periferie jako klávesnici a myš. Odhlášením uživatele zaniká v operačním systému jejich propojení s aplikacemi. Každopádně si i aplikace udržují kontext rozpracování dat. Dobrý editor např. uchovává jako součást dat také informaci o naposledy editovaném místě textu a při pokračování práce uživateli nastaví toto místo jako výchozí místo další editace. Podobně i prohlížeč diskových pamětí nás nastaví do naposledy zobrazovaného adresáře a zobrazí jeho obsah. Tyto informace jsou přitom udržovány ve víceuživatelského přístupu, znamená to, že každý uživatel, který se dokáže přihlásit, má nastaveno své vlastní prostředí práce (jak desktopu, tak aplikací), které v čase prochází změnami dle jeho potřeb. Více uživatelů znamená nejenom střídání uživatelů u téže obrazovky, ale používání různých přípojných míst prostřednictvím několika různých grafických pracovišť u téhož výpočetního systému, a to několika uživateli současně. Celou situaci si lze představit pomocí kresby III-8.



Kresba III-8: Prostředí práce uživatele.

Z kresby také vyplývá, že evidence celého uživatelského pracovního prostředí je rozmístěna v různých částech disků. Po přihlášení uživatele je grafická aplikace desktop postupně načítá a převádí je do vizuální podoby pracovního prostředí

na obrazovce, u které uživatel právě sedí. Evidováno je jak rozmístění ikon, tak ikony uživatelem vytvořené a spojené s daty nebo aplikacemi. Prostředí různých uživatelů je ukládáno ve stejném kontextu. Každý uživatel má však svá vlastní nastavení, která jsou rozpoznávána podle jeho přihlašovacího jména. Je to nastavení desktopu a jednotlivých aplikací. Ve stejném okamžiku může pracovat z jiného grafického pracoviště jiný přihlášený uživatel a pokud je výpočetní systém dostatečně výkonný, uživatelé o své přítomnosti nemusí vzájemně vědět. Jejich pracovní prostředí je pro každého jedinečné, stejně jako jejich data, která jsou proti vzájemnému zásahu chráněna operačním systémem. Zda se přitom jedná o uživatele připojené přímým hardware grafického rozhraní (kterých může mít jeden počítač více) nebo sítí, je u dobře navrženého operačního systému a software grafického podsystému lhostejné (viz koncept X Window System výše).

**desktop
a standard**

Snaha o standardizaci prostředí uživatele v různých operačních systémech je letitá. Důvodů, proč k ní nedochází, je celá řada. Nejedná se pouze o snahu výrobců udržet si klientelu spotřebitelů. Jak jsme již uvedli, rozhraní práce uživatele určovali vždy technicky zaměření programátoři. Na vývoji se nikdy nepodíleli např. psychologové, filozofové, lékaři a umělci, kteří by společně pracovali jako koncipovaný tým. Také z tohoto důvodu není známo, které z používaných prostředí je nejvíce smysluplné. Ať už samotní uživatelé říkají cokoli, při prosazování jimi používaného prostředí se jedná často pouze o zvyk. Konečně, vývoj takového rozhraní uživatele není u konce, protože vývoj používání výpočetní techniky ve světě lidí je jistě teprve v počátcích a nikdo nedokáže ani přibližně stanovit, k čemu všemu bude tento nástroj v budoucnu používán i v běžné praxi. Přesto i z pohledu práce uživatele s desktopem již vzniká snaha o standard, protože základní prvky jako je ikona, okno, menu atp. jsou již léta ustáleny. O jednom z progresivních pokusů o standard můžeme číst např. na www.freedesktop.org.

grafická informace

Přístup uživatele k výpočetnímu systému se zakládá na manipulacích s daty, které a která mají pro samotného uživatele informační výpověď s různým významem. Dnes si pod termínem desktop stěží čtenář představí obrazovku alfanumerického terminálu, jak byla popsána při vzniku systému oken v předchozím textu. Uživateli se vybaví barevně pestrá plocha obrazovky grafického podsystému, na které je řada ikon, lišt, vysouvajících se menu a oken. Rozdíl mezi podobnou koncepcí v textové podobě ale není v informační výpovědi, je pouze v jemnosti zobrazení. Tento příklad spíše snížení informační výpovědi nadbytečným vykreslením je typický, přestože v používání grafických podsystémů není zdaleka jediný. Dalším příkladem je způsob používání barvy. Barva je zcela jistě nositelem informace, jak ví každý student informatiky. Informační výpověď se jí ale v rámci desktopů nepřikládá žádná, a sice proto, že se s ní v tomto pojetí nikdy nepracovalo. Přesto, objeví-li se v nově otevřeném okně barevná fotografie z nedávného výletu, informace se na nás vysypou takřka okamžitě.

Rozdíl mezi vyjádřením slovem nebo obrazem je obrovský. Prakticky nelze stanovit jednoznačné spojení mezi textem a obrazem ve smyslu totožné

informační výpovědi. Co vyjádříme textem je na obraz nepřevoditelné a naopak. I ovládání desktopu má vlastně textový charakter. A teprve zpracovávání nasnímaného obrazu světa kolem nás můžeme považovat za využívání grafického podsystému počítačů.

Grafická informace je ukládána v souboru. Jedná se o soubor binárního charakteru, pro zobrazení jeho obsahu je proto nutno mít program, který struktuře jedniček a nul v 256 kombinacích systému ASCII rozumí a dokáže ho interpretovat na grafické obrazovce (nebo jiném výstupním zařízení, např. tiskárně). Strukturu ukládání grafických informací v obsahu souboru nazýváme grafický formát dat. Opět i v této partii Computer Science je používáno velké množství formátů, jejichž vznik byl a je dán nikoliv pouze snahou odlišit se od konkurence. Definovat obecně přijatelný formát grafických dat totiž komplikuje např. způsob další manipulace s ním. Důležitá je podrobnost záznamu, kterou může komplikovat nadměrná velikost výsledného souboru a určitě následná analýza takového obrazu.

grafický formát dat

Počítačová grafika je založena na systémech skládání tří základních barev, které v daném systému nemohou vzniknout z barev jiných, jedná se o trichromatický kolorimetrický systém. Každá barva je pak při digitalizaci vyjádřena trojicí – tripletem zastoupení těchto barev. Podle toho, o jaký systém se jedná, je triplet interpretován buď jejich skládáním, tj. aditivní systémy, např. systém RGB, nebo jejich odpočtem, tj. subtraktivní systémy, např. systém CMY. V systému RGB jsou tři základní barvy: červená (Red), zelená (Green) a modrá (Blue). Jejich číselné vyjádření v tripletu určuje, v jaké míře je barva přidána do černé. U systému CMY jsou základní barvy azurová (Cyan), purpurová (Magenta) a žlutá (Yellow) odečítány od bílé. RGB je systém používaný pro zobrazování na displejích, CMY pak pro tisk a papírovou interpretaci fotografií. V zájmu spravedlnosti bychom měli jmenovat i další systémy, které např. lépe vyhovují interpretaci malířských prací, což je např. HSV (Hue-Saturation-Value, barva-sytost-jas) nebo systém vhodný pro televizní techniku YUV (signály Y, U, V). Systém RGB, případně CMY však souvisí se základními počítačovými perifériemi, které barevný obraz interpretují, proto jsou v počítačové grafice používány především. Pro systém CMY pak došlo při praktickém používání k modifikaci na CMYK, a to tak, že K reprezentuje čtvrtou složku, která je černá. Vyjádření černé pomocí tří základních barev je totiž prakticky obtížné a také provozně zbytečně drahé.

RGB, CMYK

Při digitalizaci obrazu v informačních technologiích rozlišujeme dva základní způsoby kódování obrazu. Je to vektorový formát a formát bitové mapy grafických dat.

kódování obrazu

Formát bitové mapy vychází z pojmů samotného zobrazování dat na displeji. Nejmenší zobrazitelnou jednotkou na grafické obrazovce je pixel. Jedná se skutečně o nejmenší elementy, ze kterých je na obrazovce poskládán obraz. Fyzický pixel je barevný bod na obrazovce. Pixel je matematicky vyjadřován hodnotou, která určuje jeho barvu. Přesněji se jedná o zachycení tří hodnot odkazů do palety barev systému RGB. Pokud (u pracovních stanic) určíme

pixel

pro vyjádření hodnoty každé barvy 48 bitů (6 bajtů), každá složka této trojice může nabývat hodnoty (na 2 bajtech) od 0 do 65535, např. 0,0,0 je černá, 65535,65535,65535 je bílá. Celkem tímto způsobem můžeme obecně používat 2^{48} barev což, jak se tvrdí, již přesahuje schopnost vnímání lidského oka a je dosud také obtížně zajiřitelné technickými možnostmi displeje. V praxi PC je používán termín *pravá barva* (true color), který je používán při možnostech fyzického zobrazení 2^{24} barev (každá hodnota tripletu na 1 bajtu, rozsah 0–255). Termín vznikl na základě odhadu rozlišení takového množství barev lidským okem. Dřívější termín *vysoká barva* (high color) je používán pro displeje s možnostmi fyzického zobrazení nejvýše 2^{15} nebo 2^{16} barev.

bitová mapa Z principu pixelů vycházejí *rastrované displeje* (obrazovky), proto se bitovým mapám dříve říkalo také rastrové formáty. Skutečně, formát bitové mapy (bitmapy) vychází z primitivního zobrazování dat na rastrovém displeji tak, že v souboru s obrazem jsou zachyceny informace spojené s každým pixelem na obrazovce.

vektor Druhý nejpoužívanější formát vektorového kódování obrazu je založen na modelování obrazu jeho vyjádřením matematickou aproximací. Obraz je rozložen na pokud možno nejjednodušší geometrické tvary, které jsou v ideálním případě popsány systémem rovnic diferenciálního počtu. Teorii takové metody se zabývá numerická matematika v rámci každého studia informatiky na vysoké škole. Popis obrazu pomocí vektorů je uložen v souboru a umožňuje jej souvisle zobrazit v různé velikosti aniž by se objevil viditelný rastr. Každopádně je však i zde důležitá velikost digitalizované předlohy. Přestože je tento formát výrazně sofistikovanější a progresivnější, jeho vznik časově spadá před vznik formátu bitových map. Důvod není jenom v postupné vulgarizaci v Computer Science, ale je také ryze pragmatický. Záznam ve vektorech je výrazně menší a umožňuje zobrazení v různé kvalitě na různých typech rastrových obrazovek. Manipulace s ním je ale náročnější, protože před vlastním zobrazením musí být vždy proveden výpočet. Naopak bitová mapa je zobrazována prakticky pouhým přepisem obsahu souboru na obrazovku. Bitová mapa je proto oblíbená všude tam, kde se šetří na rychlosti zobrazení, např. u počítačových her, animací, přímé práce s obrazem tj. např. u CAD/CAM systémů (Computer-Aided Design/Computer-Aided Manufacturing), což jsou návrhové systémy produktů průmyslu. Přestože samotné vnitřní výpočty tyto programy provádějí ve vektorech, výsledkem pro zobrazení je ale bitová mapa. Naopak vektorový formát je stále oblíbený všude tam, kde je potřeba šetřit na velikosti dat, což je např. jistě přenos dat sítí (zobrazování v Internetu), nebo všude tam, kde se pracuje s různou úrovní vykreslení obrazu. Výhoda bitové mapy je také v ostrém vykreslení změny barvy, a to na úrovni pixelů, což se vektorově dosahuje obtížně. Vektorový záznam se proto také příliš nehodí na záznam fotografií atp.

rozlišení Jak vektorový, tak bitmapový způsob kódování obrazové informace je na fyzickém rastrovém displeji zobrazován podle barevných možností pixelů (16, 24, 48 bitů) a podle *rozlišení* (resolution). Rozlišení je míra podobnosti. Je vyjadřována v počtu pixelů (šířka) krát počet vzorkovacích řádků (délka).

V obchodě s obrazovkami nebo grafickými kartami se setkáváme s údaji o rozlišení, které tyto periferie umožňují. Např. 1280x1024 znamená, že displej dokáže zobrazit 1280 pixelů na 1024 řádcích. Fotografie, která pokryje celou obrazovku, je zobrazena v tomto rozlišení. Je-li zobrazena v okně, odpovídá rozlišení výseku obrazovky daného vnitřní velikostí okna. Při samotném zobrazování přitom může docházet jak ke ztrátě informací o barvě, tak naopak k nevyužití technických možností zobrazovacího zařízení (kterým je jak obrazovka, tak grafická karta), a to podle toho, jak podrobně byl obraz digitalizován. Převod z původně podrobně digitalizovaného obrazu na méně kvalitní (a mnohdy celkově mnohem menší) je hitem dnešních dnů. Je to dáno především omezenou rychlostí přenosu dat sítí Internet. Čtenář stránek WWW nechce čekat příliš dlouho na zobrazení všech obrázků umístěných na stránce, zároveň však očekává jistou kvalitu obrázku. Převody a manipulace s původně digitalizovaným obrazem jsou věcí výtvarníka, jehož specializace pak pokračuje studiem pokročilejší literatury než je tento text, jako např. [MurrayRipper1994] nebo [AdobePhotoshop].

Kvalita obrazu ale není určena pouze jeho možností zobrazení na displeji, důležitý je také jeho tisk. Je třeba vycházet z možností tiskařské techniky, a to jak stolní, tak průmyslové. Základním poměřováním kvality vytištěného obrazu je jednotka DPI (dots per inch, bodů na palec), která určuje, jakého počtu nejmenších elementů tisku jsme schopni dosáhnout na jednotce plochy. U počítačových aplikací, které jsou určeny pro přípravu podkladů pro papírový výstup (reklama, papírové grafické publikace, knihy, periodika atp.) má taková jednotka klíčový význam již při samotném návrhu. Také proto je možné u těchto aplikací používat určování barev v systému CMYK. Znamená to, že vyjádření téže barvy v různých systémech (RGB a CMYK) je pro praxi důležité, přestože samotný převod může být zatížen chybou (výsledná barva je jiná). I zde se dostáváme na okraj říše zpracování obrazu určeného pro tisk na papír, které se podrobně věnují jiné texty než je tento.

Typickým představitelem bitmapového formátu je např. BMP (firmy Microsoft, Windows Bitmap), PCX (PC Paintbrush File Format), TIFF (Tag Image File Format) a TGA (Targa Image File). Velmi používaným formátem je ale také JPEG (Joint Photographic Experts Group, často označovaný pouze jako JPG), který se jednoznačně etabloval v digitalizační technice (fotoaparáty, skenery). Konečně uvedme i velmi progresivní formát PNG (Portable Network Graphic Format), který dokáže ukládat data až do 48 bitové úrovně a byl navržen speciálně pro přenos a ukládání dat v počítačových sítích. Příkladem vektorového formátu může být AutoCAD DFX (Drawing eXchange Format) a Microsoft STYLK.

BMP, JPEG, PNG

Pokus o využití výhod obou typů formátů je v *metasouborových* formátech, které mají možnost ukládat jak bitovou mapu, tak vektorový záznam. Jejich reprezentantem je jistě WPG (WordPerfect Graphics Metafile), Macintosh PICT a ISO CGM (Computer Graphics Metafile).

Používaným formátem záznamu obrazu zejména v Internetu je také GIF (Graphics Interchange Format), který patří do hypertextových jazyků (patří

GIF

mezi ně např. i DXF, ale i PostScript). Tento formát byl vyvinut pro zobrazování při postupném načítání dat (při zobrazování stránek WWW, známé postupné zjemňování obrázku). Výhodou takového formátu je také možnost práce s jednoduchou animací. Takové a podobné hypertextové jazyky jsou dnes používány např. pro záznam zvuku atp.

Typů grafických formátů je celá řada a v neustálém vývoji je i samotný způsob kódování digitalizovaného obrazu. Přestože do tohoto vývoje vstupuje snaha ISO o standardizaci, je vývoj a vznik nových grafických formátů stále podmiňován především způsobem zpracování samotného obrazu. Požadavky na zpracování digitalizovaného obrazu přitom dnes zdaleka nejsou u konce, naopak, každý obor lidské činnosti je dnes teprve začíná definovat.

Vznik nových formátů nebo použití digitalizovaných obrazů určených původně k jinému účelu si vynucuje možnosti převodu (konverzi) mezi jednotlivými formáty. I zcela laickým přístupem si dokážeme představit, že při převodu obecně musí často docházet k určité ztrátě informací, pokud nemá být další práce s konvertovanými daty zbytečně těžkopádná.

vnitřní formát grafických dat

V současné době každá výkonná počítačová aplikace umožňující práci s obrazem ukládá data ve svém vnitřním formátu, který je dán zaměřením vlastní aplikace. Formáty uvedené v předchozím textu často vznikly jako výsledek kompromisu mezi vnitřními formáty různých používaných aplikací tak, aby bylo možné nad data vytvořenými v jedné z aplikací pokračovat v práci v aplikaci jiné (jakkoliv to výrobcům software není po chuti). Proto každá z aplikací nabízí import nebo export dat z (do) obecně definovaných formátů. Každý výtvarník by ale měl mít před zahájením vlastní práce přehled jednak o obecné struktuře práce s obrazem (v intencích alespoň tohoto textu) a jednak o možnostech a zaměření jednotlivých aplikací, které má k dispozici. K tomuto přehledu přitom bohužel nestačí mít doporučení kamaráda, který je na nějakou aplikaci zvyklý (jistě, dobrá rada je drahá).

Práci s obrazem se věnují grafické aplikace různého zaměření. Grafickou aplikací asi nebude vývojové prostředí programátora, který v několika oknech programuje přístup k datům textové povahy, jako je např. diář nebo cestovní deník či jiné tabulkově orientované programy, jakkoliv může být součástí jeho výsledného programu řada ikon či zajímavých pestrobarevných tlačítek. Grafické prvky totiž programátor i tak skládá z výběru nabízeného vývojovým prostředím programovacího jazyka, nebo je pro něj vytváří výtvarník, ale již v jiném programu. Nebude se jednat ani o CAD/CAM návrhové systémy konstrukce průmyslových výrobků, protože i zde se pracuje s návrhem technického charakteru, přestože i design má zde své opodstatnění. Také u designu průmyslových výrobků vstupujeme do diskutabilního prostoru, protože každý průmyslový výrobce by měl mít tým výtvarníků. CAD/CAM systémy samy nabízejí možnosti způsobu designu výrobku, ale jedná se vždy o řešení typizované. Do prostředí samotného technického návrhu výrobku je možné vkládat grafické prvky, při jejichž použití je akceptován design nově navržený výtvarníkem, nikoliv ten, kterým CAD systém disponuje implicitně.

Není to snadné, ale jedná se o tvůrčí proces, jemuž výpočetní technika pouze napomáhá, jako je to koneckonců i u všech jiných způsobů jejího použití.

Za grafické návrhové systémy lze považovat především ty, které používá pro svou práci výtvarník. Principiálně přitom vznikají podklady pro práci dvojího typu. První představuje otisk světa kolem nás, a to na základě fotografování, scanování atp. Druhý je založen na použití grafického programu pro samotný vznik obrazu. Dobré aplikace přitom umožňují pracovat s oběma typy, přestože v současné době jde obvykle o dvě různé aplikace, které si vzájemně mohou data poskytovat. Typickým příkladem aplikací pro zpracování obrazu jsou kvalitní a výkonné produkty firmy Adobe, a to jednak Photoshop a jednak Illustrator (viz [AdobePhotoshop2004] a [AdobeIllustrator2004]), přestože vytvořit jednoduchou kresbu lze i pomocí řady kancelářských aplikací, jako je Open Office (viz www.openoffice.org) nebo MS Office. Analogii produktům firmy Adobe (viz www.adobe.com) do jisté míry představuje hodně rozšířený software s názvem CorelDraw a s ním související programy firmy Corel (viz [CorelDraw] a www.corel.com). Z oblasti volně šiřitelných programů by bylo jistě hříchem opomenout GIMP (GNU Image Manipulation Program, viz www.gimp.org), který se software Adobe Photoshop v mnohém nejméně vyrovná.

Přestože hlavní způsob využívání výpočetní techniky pro zpracování obrazu je stále více zaměřován na lidské vnímání přímým způsobem, výtvarníci ji používají stále především na úrovni přípravy podkladů pro tiskařské stroje. Výsledný obrazový materiál je pak prezentován lidskému oku klasickým způsobem jako tiskový výstup. Výtvarník se proto zajímá o prostředky, které mu umožní realizovat jeho představu tisku plakátu, reprodukce nebo celé publikace především na papírový materiál. Znamená to, že vnitřní formáty interpretace jeho obrazů jsou pro něj důležité natolik, nakolik ovlivňují výstupní tiskařské stroje, na které je také jeho práce zaměřena. I dříve byl tento způsob práce s digitálním obrazem používán a proto je jeho součástí množství dnes již nepoužívaných a těžkopádných metod. Formátem dat snažícím se sjednotit způsob ovládání výstupních tiskových zařízení byl svého času jistě PostScript, který definoval John Warnock v roce 1976 (tehdy jako zaměstnanec firmy Evans and Sutherland, později, v roce 1982, se stal jedním ze zakladatelů firmy Adobe). Dodnes používaný grafický formát a současně jazyk pokynů pro práci tiskařského stroje byl založen na jednoznačném popisu obrazu v rámci jeho tiskové prezentace. Tiskařská zařízení, která jej dokázala interpretovat, pak produkovala výstup tak, jak jej výtvarník v počítačové aplikaci plánoval. Jazyk PostScript tak umožňoval poskytovat digitální zdroj dat jakékoliv tiskárně, která vlastnila stroje, které tomuto jazyku rozuměly. Nebylo tedy zapotřebí přizpůsobovat se vždy poměrně komplikovaným tiskařským strojům různých formátů, producenti výsledných tisků si tak mohli vybírat vždy mezi více realizujícími firmami, a to i podle jejich technických možností, zatímco dřívější způsob práce znamenal orientaci pouze na jednoho nebo několik producentů, na nichž pak vznikla závislost.

PostScript

PDF Podobné závislosti na typu operačního systému nebo na grafické aplikaci a grafickém formátu se pokusil zabránit formát grafických dat definovaný firmou Adobe jako formát PDF (Portable Document Format). Podobně jako u PostScript se i zde jedná o výstupní formát zpracovaných dat určených pro čtenáře, jako jsou např. ilustrované publikace, vysázené dokumenty atp. Znamená to, že po fázi konstrukce grafického díla v některé ze sázecích nebo grafických aplikací je provedena konverze z vnitřního formátu aplikace do formátu PDF, který je obecně čitelný v kterémkoli operačním systému a v kterékoli aplikaci, která má za úkol výsledek práce interpretovat pro čtenáře, a to jak na tiskovém zařízení, tak především na obrazovce. Firma Adobe podpořila export ze svých grafických aplikací do formátu PDF a současně zdarma poskytla k dispozici program, který obsah souboru v tomto formátu zobrazuje v okně různých grafických podsystémů různých operačních systémů. Jméno čtenářského programu je Acrobat Reader (viz www.adobe.com), přitom jeho rozšířená a poměrně levná verze s názvem pouze Acrobat umožňuje i editaci souborů s obsahem ve formátu PDF. Součástí programu Acrobat je konverzní program Acrobat Distiller, který slouží jako výstupní zařízení v operačním systému. Acrobat Distiller je uživateli v operačním systému přístupný jako jedno ze zařízení, na kterém je možné tisknout, jeho výstup je ale ukládán do souboru ve formátu PDF. Uživatel tak při zakoupení programu Acrobat získá možnost ukládat výstup v obecně čitelném a interpretovatelném formátu PDF z kterékoli používané aplikace jeho operačního systému. Samotný Acrobat Reader uživateli umožňuje také tisk zobrazovaného dokumentu na jeho tiskárně, a to v grafické podobě odpovídající tomu, co uživatel vidí na obrazovce. Formát PDF byl úspěšný a autor tohoto textu je přesvědčen, že to bylo způsobeno úsilím o snadnou čitelnost téhož grafického výstupu na kterémkoliv počítači na světě. Čtenáři tak přestali být závislí na koupi mnohdy poměrně drahých grafických a sázecích aplikací jen proto, aby si mohli přečíst vyprodukovaná data různých vydavatelství v elektronické podobě.

fonty Specifickou částí grafických podsystémů jsou *fonty*, česky řezy písma (typy písma) neboli znakové sady. V části Base Of I.1 jsme definovali práci s písmeny abecedy a jejími národními odchylkami ve smyslu jejich kódování v rámci ASCII, případně UNICODE. V obou případech se jedná pouze o digitální určení významu obecného znaku abecedy, tedy např. o určení, že znak A má v ASCII sled bitů v bajtu hodnotu 01000001. Není zde stanoveno, zda bude tento znak vyjádřen písmenem s patkami nebo obloukovým písmem a v případě zobrazení znaku na obrazovce záleží na programu, který tento vlastně holý text zobrazuje. Aplikace grafického prostředí určené pro psaní a sazbu dokumentů přináší ale možnosti vyjádřit tvar písma v definitivní podobě, která bude předložena čtenáři. Typy písma již od doby Gutenberga souvisejí s profesí sazby a realizace tisků a také s otázkou přijímání písma člověkem. Sazba některým typem písma se čte hůř, pro určité literární žánry se v průběhu století ustálilo používání některých typických fontů atp.

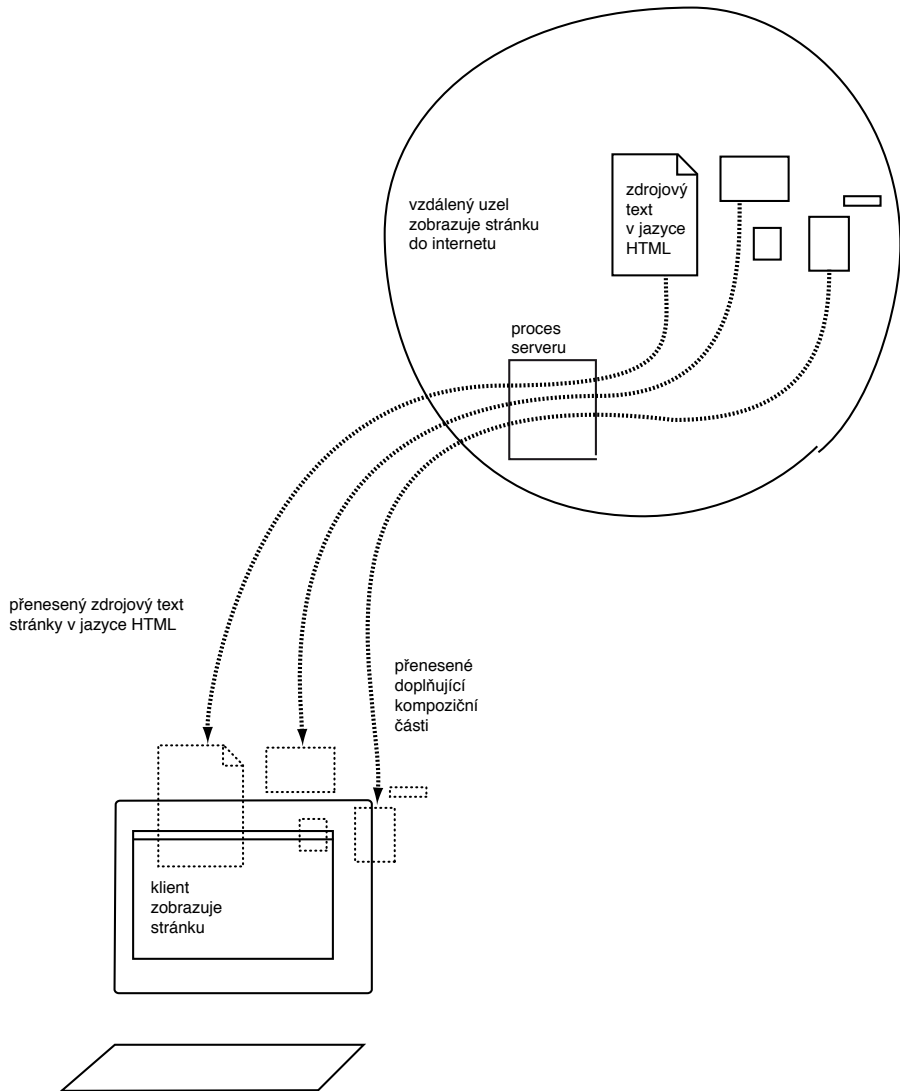
Sazeč, autor nebo grafik kombinující obraz a písmo dnes proto při práci na počítači určuje používaný řez písma z nabízeného výběru, kterou poskytuje

aplikace. Fonty se tedy zdají být součástí sázecí aplikace, ale obecně tomu tak není. Fonty jsou grafické soubory, které poskytuje operační systém, přesněji jeho grafický podsystém. Grafické aplikace je ze smlouveného místa načítají a užívají nebo je nabízejí uživateli k použití. Operační systém často disponuje pouze základní sadou běžně používaných fontů. Větší škála různých fontů pak přichází s instalací konkrétní grafické aplikace, tyto fonty jsou ale obvykle dostupné pro všechny jiné používané grafické aplikace. Návrhem a vývojem fontů se zabývají samostatné firmy, které pak na výsledky své návrhářské a mnohdy velmi obtížné práce uplatňují autorská práva. Obvykle jim to ale není mnoho platné, dobré fonty se stále často běžně šíří a používají i v prodávaném software bez uvedení jména jejich autora. Font je vlastně soubor s grafickým obsahem, říkáme, že soubor obsahuje grafická data písmového formátu (font files). Formátů pro ukládání fontů je opět celá řada, v principu se ale v průběhu času začaly rozdělovat na bitmapové (bitmap), úsečkové (stroke) a obloukové (spline). Příkladem typů fontů jsou třeba PostScript nebo TrueType. Popisem typů fontů se zde nebudeme zabývat, protože je uživatel nepotřebuje znát, pro něj je důležitá pouze přítomnost určitého fontu v grafickém podsystému a jeho dostupnost v aplikaci, kterou používá. Každopádně je konkrétní font definicí tvarů všech písmen používané abecedy přirozeného jazyka (včetně písmen s háčky a čárkami), která je vyjádřena v některém z grafických formátů tak, aby jí aplikace dokázaly porozumět a pracovat s ní. V běžně psaných dokumentech, ale i v kolážích textu a obrazu, je ve vnitřním formátu aplikace pouze odkazováno na odpovídající font vybraný uživatelem pro daný úsek textu. Znamená to, že v případě přenosu dat na jiný počítač do téže aplikace ještě nemusí být data správně čitelná. „Nemáš tam font“, zní okřídlená zdrcující odpověď povýšeného autora grafického produktu, tato věta ovšem nebohému uživateli nijak nepomůže. Již zmiňovaný formát PDF je však pro poskytování grafických dat obecně nejvhodnější, protože při exportu dat do tohoto formátu jsou do výstupu ukládány i popisy použitých typů písma. Je potřeba si ale uvědomit, že výstup ve formátu PDF není zdrojovým vnitřním formátem, ve kterém byl grafický produkt vytvořen. Při převzetí práce ve formátu PDF od grafika je nutné mít tento zdrojový vnitřní formát používané aplikace k dispozici pro potřeby dalšího pokračování, dále mít k dispozici aplikaci, která v tomto formátu pracuje a také mít k dispozici fonty, které byly v poskytnutém produktu použity.

Uvedený způsob práce se znakovými sadami při sazbě dokumentů vcelku dobrým způsobem ilustruje práci v grafickém prostředí elektronické sazby a orientaci v něm. Prakticky vždy jde o kompozici. Výsledný produkt práce je sestaven z grafických prvků a textů a dále pokynů, jakým způsobem má docházet k jejich interpretaci podle autorova záměru. Pokročilý uživatel (a enjoyer zejména) přitom dokáže sestupovat od úrovně této kompozice ke změnám např. v grafických prvcích. Upravit odstín nebo světlost fotografie v dobrém programu (Adobe Photoshop) není tak obtížné, ovšem vytvořit ikonu, kterou pak lze používat při programování, nebo navrhnout a rozkreslit nový font představuje už činnost náročnější, každopádně však možnou.

**kompozice
textu a obrazu**

Dobrým příkladem uživatelské kompozice je jistě stavba stránek WWW v jazyce HTML. Podívejme se na kresbu III-9.



Kresba III-9: Kompozice stránky WWW.

Stránka, o kterou požádáme vzdálený server, je přenesena sítí na stranu našeho klienta. Tato stránka je obsahem textového souboru, její část např. může obsahovat:

```
<FONT "Helvetica CE">
a když jsme <B>tam</B> přišli, co jsme <B>tam</B> neviděli:
</FONT><BR>
<IMG SRC="obrazky/kvetina.jpg">.
<BR>Květinu.
```


kde tag `IMG` určuje vložení obrázku mezi text při zobrazování stránky klientem. Tento obrázek není součástí textového souboru. Obrázek je uložen na serveru v samostatném souboru se jménem `kvetina.jpg`. Oproti souboru s textem stránky je uložen v podadresáři s názvem `obrazky`. Klient je řízen tagy textu stránky. Např. text `tam` bude na dvou místech zobrazen tučným písmem (tag `B`), tagem `FONT` můžeme definovat font, který klient použije při zobrazení textu takto označeného tagu. Nemá-li ovšem klient takový font k dispozici z grafického podsystému, použije font náhradní. Komponování stránky v HTML je proto věc citlivá. Ne každá stránka WWW bude zobrazena (zkomponována) tak, jak ji zobrazí náš program pro kompozici (Microsoft FrontPage, Mozilla Composer), zvláště je-li stránka zobrazována ve zcela jiných operačních systémech klientů (MS Windows, UNIX, MacOS), kde se vývoj fontů ubíral každý trochu jinou cestou.

Kompozice textu a obrazu je výhodná, a nejen pro publikační činnosti. Umožňuje používat prvky grafického návrhu opakovaně pro různé produkty. Při vlastní práci jsou tyto prvky pouze odkazovány. Neopakují se vždy ve své grafické definici u každého produktu, ale jsou odkazově přístupné ze všech takových produktů výsledné práce. Podobně také pracují např. již zmiňované návrhové systémy CAD/CAM. Z vnitřního formátu dnes každá aplikace dokáže vytvářet zobrazení pro výstupní zařízení tiskařského průmyslu (PostScript) nebo náhled pro uživatele prakticky v jakémkoliv prostředí (PDF).

Doposud jsme se v rámci digitalizace obrazu zabývali pouze statickým obrazem. Již zmínka o zvukových formátech (u formátu GIF) nás ale upozornila na nutnost podíídit také formáty grafické práce časovému průběhu. Jedná se o digitalizaci informací plynoucích v reálném čase.

Digitální záznam pohyblivého obrazu ve smyslu termínu *animační formát* vychází z principu klasického způsobu výroby filmu na celuloidový pás. Animace je simulace pohyblivého obrazu určená pro lidské oko, která vytváří pohyblivý obraz zobrazováním řady rychle po sobě jdoucích statických obrazů. Dokážeme-li takto zobrazovat v průměru alespoň od 15 do 20 obrazů za vteřinu, vytvoříme tak pro lidské oko iluzi pohybu. Tento klasický způsob výroby filmu byl vlastně beze zbytku použit při digitalizaci obrazu u animačních formátů. Grafické animační formáty tak kódují plynulý obraz primitivně uložením *snímků* (frames) - částí pohybu - v podobě bitových map za sebou. Snahou je dosahovat určité optimalizace vzhledem k objemu takto vznikajících dat, a to kódováním pouze toho, čím se po sobě jdoucí snímky odlišují. Pokud je změna příliš odlišná (střih), dochází k nové definici obrazu, který se dále proměňuje určováním jeho změn.

animační a video
formáty

Animační formáty jsou používány nejenom pro výrobu animovaných filmů, ale např. také pro zobrazování průmyslového produktu v návrhovém systému jeho plynulým otáčením nebo při práci se simulacemi pochodů přírodních jevů, třeba u předpovědi počasí nebo průběhu chemických reakcí.

Analogový videozáznam byl v době svého vzniku podmíněn principem televizního vysílání. Jedná se o kompoziční signál (z několika různých signálů je smíchán jediný), který je vyslán prostorem. V televizním přijímači je jeho

obrazová část interpretována v podobě rychle se pohybujícího paprsku, který obraz kreslí na stínítku obrazovky postupně po řádcích odshora dolů (řádků je 625, každý řádek obsahuje 830 samostatných bodů). Paprsek by měl podle klasické metody animace vykreslit 625 x 830 bodů za čtyřicetinu vteřiny. Vzhledem k tomu, že se po jejich vykreslení musí zabývat ihned dalším snímkem, obraz by se shora dolů postupně proměňoval, což lidské oko registruje jako blikání. Proto se používá půlsnímkování, což je vykreslování paprskem nejprve lichého a pak sudého řádku jednoho statického obrazu. Pro vykreslení 25 snímků za vteřinu se tedy používá zobrazovací kmitočet 50 půlsnímků za vteřinu (v normě PAL, viz dále). Princip je tedy trochu jiný než u animace, obraz se postupně překresluje vždy tam, kde se právě paprsek nachází, a to ještě jenom jeho střídavá polovina. Na celém světě se přitom stále používá několik různých systémů pro skladbu takového signálu. V Evropě známe systém PAL (Phase Alternation Line) a SECAM (Sequential Colour Avec Memoire), Amerika používá NTSC (podle National Television Standards Committee).

U digitalizace videa se při digitalizaci signálu postupuje tak, že se v určitém počtu vzorků za jednu vteřinu (vzorkovací sekvence) vyjadřuje hodnota signálu číselně na 16 bitech ($16b = 1$ vzorek). Podle typu kompozičního videosignálu se pak vzorkovaný signál rozkládá na jednotlivé signály pro tři základní barvy pro RGB nebo YUV, podobně jako tomu je např. u formátu JPEG záznamu statického obrazu. Zobrazování takové digitalizace se děje v pixelech obrazovky počítače. Hodnoty pixelů se ale neustále proměňují, a to 50 nebo 60 krát za vteřinu. Ve výsledku se jedná o zobrazování přibližně 30 statických snímků za vteřinu.

Digitalizace pohyblivého obrazu je ale obvykle nemyslitelná bez digitalizace jeho ozvučení. U digitálního filmu používáme termín multimediální formát a myslíme tím technologii kombinující záznam pro působení na více lidských smyslů současně, zde ovšem pouze zrak a sluch. Termín multimediální v sobě obecně zahrnuje mimo jiné např. i interakci.

Záznam pohyblivého zvukového obrazu uvedeme až za částí o digitalizaci zvuku. Ta se vyvinula historicky dříve, ale její vývoj několik let probíhal souběžně s vývojem digitalizace pohyblivého obrazu. Digitální zvukový film je jejich kombinací.

zvuk Záznam zvuku souvisí úzce s časem. Nikoho z nás ani nenapadne uvažovat o zvuku bez jeho plynutí v čase (na rozdíl od obrazu). Digitalizace zvuku je méně náročná na výpočet než digitalizace pohyblivého obrazu. Kódování snímaného zvuku a jeho zpětná reprodukce se proto také začalo používat v běžném lidském životě dříve než kódování digitálního obrazu. Jednalo se o technologii hudebních CD (compact disc). Tato technologie čtení digitální informace laserem byla vyvinuta pro distribuci zvukových nahrávek. Vlastně teprve následně začala být CD používána i ve výpočetní technice jako periferie diskového typu pro uchovávání dat. Při vývoji technologie hudebních CD byl mimo jiné definován dodnes používaný zvukový formát s označením CDA, který byl standardizován ISO.

U digitalizace zvuku a zvukových formátů hovoříme o vzorkování (sampling), vzorkovací frekvenci (sampling rate) a velikosti vzorku (sample size). Již z uvedených termínů je zřejmé, že se jedná o stanovení nejmenšího časového intervalu pro zachycení zvuku a jeho datového rozsahu. Pochopitelně podobně jako u pohyblivého obrazu jde i zde o navození takové iluze, kdy by lidské ucho již nemělo rozpoznat rozdíl mezi skutečností a interpretovaným záznamem. Počet vzorků za vteřinu je pro formát CDA standardem stanoven na 18900, ale pro dosažení vyšší kvality se běžně používá 44100 a 48000, některé současné systémy ale i 192000. Používaná velikost vzorku je přitom 8, 16, ale i 24 bitů. Výsledkem digitalizace zvuku (obsah vzorku) je přitom jejich prostý binární přepis podobně jako bitová mapa u digitalizace obrazu. Digitalizace zvuku snímáním určitého počtu vzorků je metoda pulsně kódovaná modulace (PCM, Pulse Code Modulation). Jedná se o lineární převod analogových signálů na digitální vzorky. Vstupní i výstupní zařízení pro práci se zvukem mají totiž vždy analogovou povahu.

pulsně kódovaná
modulace – PCM

Možnost využívání rychle se rozšiřujících PC pro práci se zvukem, a to nejenom pro jeho interpretaci, ale také produkci (dnes zvuk vzniká nejenom záznamem, ale i simulací hudebních nástrojů ve zvukových počítačových aplikacích), přinesla mimo jiné i vznik dalších zvukových formátů. Veřejným (ale ne obchodním) zájmem se stala především distribuce zvuku prostřednictvím Internetu. Vzhledem k vysokým nárokům na velikost samotného záznamu zvuku se pro dosažení přijatelných časů přenosu dat sítě začaly používat různé způsoby kódování, které byly vedeny snahou o dosažení co nejvyšší komprese (zhuštění) při co nejmenší ztrátě kvality. Kompresní metody mohou fungovat na různých principech. Jedním z nich je podobně jako u pohyblivého obrazu kódování pouze rozdílů mezi jednotlivými vzorky (vzorek tak může mít např. ne 16 ale jen 4 bity). Toto schéma je označováno jako DPCM (Differential Pulse Code Modulation). Schéma ADPCM (Adaptive DPCM) je schéma progresivnější. Je adaptivní vzhledem k právě kódovanému zvuku tak, že mění velikost vzorku podle obsahu kódovaného zvuku. Dynamická povaha ADPCM přináší vyšší úsporu a je používána u většiny způsobů kompresí jako základní algoritmus.

Snad nejvýrazněji se pro potřeby komprese uplatnil formát MPEG (Moving Picture Experts Group, je součástí ISO). Formát MPEG byl přitom vyvinut nejen pro zvuk (ve známé verzi s označením MPEG-3, v žargonu často mp3), ale i pro kódování videa (které přinesla verze MPEG-4, mp4). Při kódování lze používat různé frekvence, ty jsou pak uvedeny v samotném výsledném kódu. Dosažitelná komprese formátu MPEG tak bývá až 10:1. 80 minut plného hudebního CD o velikosti 700MB se tak např. zmenší na 70MB.

MPEG

Dalším příkladem používaného zvukového formátu jsou např. AIFF (Audio Interchange File Format) a AIFC, které vyvinula a ve svých operačních systémech používá firma Apple pro profesionální práci se zvukem. Byly přijaty také na pracovních stanicích firmy SGI a jiných. Kód v AIFC umožňuje kompresi, na rozdíl od AIFF, který kompresi nemá.

- WAV, WMA** Nelze jistě nezpomenout formáty firmy Microsoft, které zaplavily svět, přestože jejich obliba není příliš velká. Je to např. formát WAV (Waveform Audio File), který je implementací standardu RIFF (Resource Interchange File Format) vyvinutého firmou IBM a Microsoft. Dalším formátem nabízeným firmou Microsoft je WMA (Window Media Audio), o jehož struktuře je ale známo velmi málo.
- Real Audio** A konečně nelze pominout ani zvukový formát Real Audio vyvinutý firmou Real Networks, který se dnes používá pro přímé vysílání rozhlasových stanic do Internetu (tzv. streaming). Bohužel je málo podporován v běžných přehrávačích a konkurence (např. Microsoft) nabízí (a používá) pouze formáty vlastní (ASF, Advanced Streaming Format).
- MIDI** U digitalizace zvuku stojí jistě za zmínku také digitální formát komunikace s hudebními periferiemi, kterým je dnes standard MIDI (Musical Instrument Digital Interface). Nebývá považován za zvukový formát, protože neukládá vzorky dat, ale podobně jako u vektorových formátů obrazu pouze zachycuje instrukce, které definují ovládání daného hudebního nástroje. Přehrání záznamu je následně možné také způsobem odeslání dat v tomto formátu zpět na periferii. Zvuk obecně však nepopisuje.
- zvukový film** Když používáme termín film, myslíme tím záznam pohyblivého se obrazu doprovázený zvukem. Pro původní význam termínu film dnes spíše používáme němý film, termín zvukový film naopak používáme málo. Přesto i v počítačových technologiích prožívaly digitalizace zvuku a digitalizace obrazu dost často každá svůj vlastní vývoj, podobně jako tomu bylo u klasických technologií souvisejících s filmem. Interpretace zvuku na osobních počítačích např. zaujala uživatele až nějaký čas po nástupu grafických systémů, přestože, jak jsme uvedli, se digitální zvuk vyvíjel nezávisle na jeho dostupnosti na PC. Spojit digitální pohyblivý obraz s digitálním zvukem znamenalo vyvinout vnitřní formát souboru s takovou digitální informací, která i zde vycházela z původního pojetí záznamu u klasických technologií. Ke kódovanému obrazu přibyla zvuková stopa. Snaha o dosažení co největší věrnosti vedla k požadavku používat více zvukových stop současně. Podobně jako CD pro distribuci zvuku se také digitální technologie pro distribuci videa s označením DVD (Digital Video Disc) prosadila vlastně odděleně od samotného zpracování videa na počítači (tím myslíme jeho střih, triky a další úpravy). Vzhledem k tomu, co bylo již řečeno před částí zabývající se digitalizací zvuku, musíme rozlišovat dva způsoby pořizování digitálního videa. Jedním je digitalizace televizního signálu a druhým je snímání a přímé kódování v digitálním formátu digitální videokamerou. Formát DV (Digital Video) je ten, který digitální kamery používají přímo, další alternativy jsou VCD (Video CD) a SVCD (Super VCD), svět ovšem směřuje k formátům DVD (kterých je opět řada). Každý formát používá při digitalizaci určitou kompresi dat, protože kapacitní možnosti současných výpočetních systémů jsou pro digitální video stále omezující. Způsobům komprese, tedy zhuštění tak, aby přenos byl co nejkratší, obecně říkáme *kodeky* (z angl. codec = compression + decompression). Komprese dat je dnes velmi významná, respektive zajímavá

velkou část populace planety, protože umožňuje pracovat s filmem takovým způsobem, který je jinak možný pouze s náročným vybavením hardware, který je stále drahý (software je často zdarma). K tomu, aby např. bylo možné stříhat 30 minut filmu při plné digitalizaci v rozlišení, které představuje běžný videosignál, potřebujeme přibližně 60GB místa na disku a výkonný procesor i rozhraní přenosu dat mezi operační pamětí a diskem. Použijeme-li kompresi, je možné film stříhat a dále distribuovat přibližně i při desetinovém objemu dat. Rovněž přenášení filmových dat Internetem při velikosti 5GB a více je náročné a přímé promítání do Internetu v těchto velikostech je vlastně dnes stále nemožné. Kodeků je celá řada a nejedná se pouze o výstřelky hackerů. Jsou vyvíjeny a podporovány seriózními výrobci software pro zpracování a promítání videa a také bývají součástí hardware, který převádí analogový videosignál na digitální záznam již v komprimované podobě, mnohdy bohužel se ztrátou možnosti návratu k plné kvalitě. Kodeky skutečně rozlišujeme na bezztrátové a ztrátové. Dobrý bezztrátový kodek je např. HuffYUV, typický ztrátový kodek je Microsoft H.261 a H.263. Otevřenou a současně efektivní kompresi nabízí formát MPEG již zmiňovaný u kódování zvuku. Jeho podoba pro film ve verzi MPEG-4 je dnes velmi významná pro jakoukoliv distribuci filmů Internetem, a to dokonce i při přímém vysílání (streaming), kdy klient zobrazuje data poskytovaná serverem plynule bez nutnosti posečkání na dokončení přenosu celého filmu. Již princip verze MPEG-1 byl standardizován a uvolněn pro obchod výrobců softwaru a hardware. Také již uvedený formát firmy Microsoft pro streaming zvuku s označením ASF byl rozšířen na video. Vychází z jejich původního formátu pro video s názvem AVI (je kombinován s technologií MPEG-4), jeho novější verze nese označení WMV. Podstatná je ale jeho uzavřenost, tedy nemožnost převodu do jiného formátu. I přes toto omezení se hackerům podařilo jej zlomit a vytvořit tak kodek, jehož sláva se dotkla hvězd a má označení DivX (tehdy s označením 3.11a Alpha). Na základě jeho popularity byla vytvořena jeho nová verze DivX 5, která už ale nebyla k dispozici se zdrojovými kódy a za její používání se muselo platit. Jako reakce na zkomercializování DivX pak vznikl kodek Xvid, který z DivX vychází.

Ve výpočetní technice se ale u digitálního filmu předpokládá větší prostor a svoboda než kterou představuje jeho pouhá konzumace. Ozvučený digitální film je totiž jenom jedna část využití *multimediálních formátů*. Multimediální formáty ve svém principu zahrnují možnosti digitálního kódování i jiných vjemů člověka, které dokáže počítač obecně simulovat prostřednictvím vhodné periferie. Multimediální formáty také navíc zahrnují princip interakce, tj. reakce na požadavek člověka vhodnou odpovědí. Určitá inteligence, kterou pak výsledný software disponuje, ale není věcí samotného formátu. Ten má za úkol pouze poskytovat možnosti při realizaci různých algoritmů současného používání statického obrazu, pohyblivého obrazu, zvuku, ukazovacích zařízení nebo i polohovacích zařízení. Nejznámější doménou multimediálních formátů jsou dnes především počítačové hry, ale jejich uplatnění má daleko širší charakter, který sahá na hranice simulací procesů výzkumu různých oborů lidské činnosti nebo k využívání virtuální reality.

**multimediální
formáty**

Multimediální formáty, které se dnes komerčně používají pro kódování filmů, jsou jistě QuickTime firmy Apple a RIFF firmy Microsoft. Denně lze ovšem zaznamenat proměny této partie, která stojí rozkročená mezi Computer Science a výhodnou komercí.

scénické
a trojrozměrné
formáty

Zmínkou o virtuální realitě jsme otevřeli otázku implementace simulací a umělých světů v počítačových technologiích vůbec. Zvukový film není jediným způsobem popisu vnímání člověka. Orientací a pohybem v prostoru, případně ovlivňováním tohoto prostoru v počítačové simulaci, začínají formáty *virtuální reality*. Předstupněm těchto formátů jsou jistě *scénické formáty* a *formáty trojrozměrné*, které slouží pro popis scény prostoru. Jejich principem je vlastně vektorový způsob kódování třírozměrného podkladu tak, aby jej bylo možné využít pro jeho interpretaci nebo orientaci v něm. Tyto formáty jsou opět využívány také v počítačových hrách, ale jejich vznik jistě iniciovala především robotika, bez níž si dnešní stav průmyslu stěží dokážeme představit.

formáty virtuální
reality

Virtuální realita je chápána jako umělé působení na lidské smysly tak, aby samotné mozkové centrum člověka bylo uvedeno v omyl. Přestože i to je možné (a dosažitelné i jinými prostředky), význam virtuální reality je od jejího počátku jiný. Smysl je v simulaci, ve snaze pochopit případný vývoj světa kolem nás, který je již později nevratný. Příkladem využití virtuální reality je možnost procházky po navrhovaném architektonickém objektu nebo simulace neurochirurgické operace. Jako nový způsob vyjadřování se začíná ale používat i v umění a správný enjoyer hledá její uplatnění také např. v sexu. Snahou pro jednotné a dostupné řízení software virtuální reality je definice jazyka VRML (Virtual Reality Markup Language), který je ve své syntaxi pokračováním jazyka HTML světa informačních serverů Internetu.

III.3 – informační systémy a databáze, SQL, XML, datové sklady

Tématický postup vývoje textu této knihy nás postupně zavedl k popisu služeb, které jsou zaměřeny na požadavky jednoho koncového uživatele a k jejich ovládní. Je to v souladu se současným stavem vývoje přístupu člověka k výpočetní technice, jejího vnímání a využívání v běžném životě. Takové vnímání výpočetní techniky ale člověka pochopitelně silně ovlivňuje a ovlivňuje i představu o systémovém zpracování potřebných algoritmů modelované skutečnosti. Obecně tak vzniká představa potřeby silného výkonného hardware na místě přístupového místa koncového uživatele s odůvodněním, že vše podstatné k algoritmickému zpracování se musí odehrávat právě zde. To je jistě v rozporu s tématem popisovaným v druhé části tohoto textu (v části Connect To), a sice s distribucí a využíváním výkonu prostřednictvím principů počítačových sítí.

Pod pojmem *informační systém* rozumíme realizaci projektu souvislého poskytování a distribuce informací organizované části lidské společnosti za účasti jejích lidských, technických a finančních zdrojů tak, aby sloužil k podpoře samotného smyslu organizace. Tato suchopárná definice vlastně nehovoří o nutnosti využití výpočetní techniky a skutečně je hlavní princip obsažen především v samotném systému organizace. Že je při organizované činnosti lepší vzájemně si poskytovat informace elektronickou cestou než například papírovou, je jisté už od časů realizace počítačových sítí. Také se již dříve ukázalo, že stroj dokáže provádět mnohdy i primitivní výpočty výrazně rychleji než člověk a současně bez chyb. Člověk je tak oproštěn od monotónních a únavných činností a při jeho zařazení v organizaci je mu poskytnut větší prostor pro činnost tvůrčí a rozhodovací. Výpočetní technika se tak stala dnes již zřejmou součástí informačních systémů, přestože její výpočetní zdroje samotným informačním systémem nejsou.

informační systém

Pro vedení provozu informačního systému pomocí výpočetní techniky se předpokládá vzájemné propojení lidí v organizaci tak, aby každý člověk měl k dispozici všechny potřebné údaje související s jeho činnostmi a žádné jiné. Jedná se jistě o ukládání dat v potřebných souvislostech, jejich zpracovávání, poskytování a ochranu před neoprávněným použitím. Pro potřeby uchovávání dat tímto způsobem byly v rámci Computer Science vyvinuty databázové systémy.

Databázový systém (database system) je intelligence ukládání elektronických dat v jejich vzájemných obsahových souvislostech, tedy princip, jakým jsou data ukládána v (externí) paměti počítače tak, aby přístup k nim byl v jejich vzájemném kontextu co nejrychlejší a umožňoval je v různých souvislostech snadno číst, měnit a odstraňovat. Databázový systém také musí zajistit řešení problému současného přístupu několika uživatelů k datům. Pokud se spokojíme se situací, kdy pouze jeden uživatel data vkládá a mění, zatímco ostatní je pouze čtou, nejedná se o velký problém. Situace se stane složitější v okamžiku, kdy data může vkládat a v průběhu jejich využívání je i měnit několik uživatelů. Příkladem může být systém prodeje autobusových jízdenek

databázový
systém

přepravní společnosti, která umožňuje pasažérům nakupovat jízdenky na libovolnou linku v kterémkoliv městě, kde má společnost kancelář. Všechny kanceláře jsou připojeny počítačovou sítí k centrální databázi společnosti, kde se evidují všechny linky a jejich obsazenost, tedy prodané místenky. Každý prodejce místenek v každé kanceláři musí mít k dispozici současný stav všech linek, aby mohl volnou místenku odkudkoliv kamkoliv prodat. Prodej ale trvá přibližně minutu a v průběhu této doby může mít o místenku zájem i někdo jiný. Riziko souběhu se zvyšuje s blížícím se odjezdem autobusu a s počtem prodejních kanceláří. Tento a další podobné problémy inteligentního uchovávání přístupu k datům při vedení provozu informačních systémů řeší databázový systém.

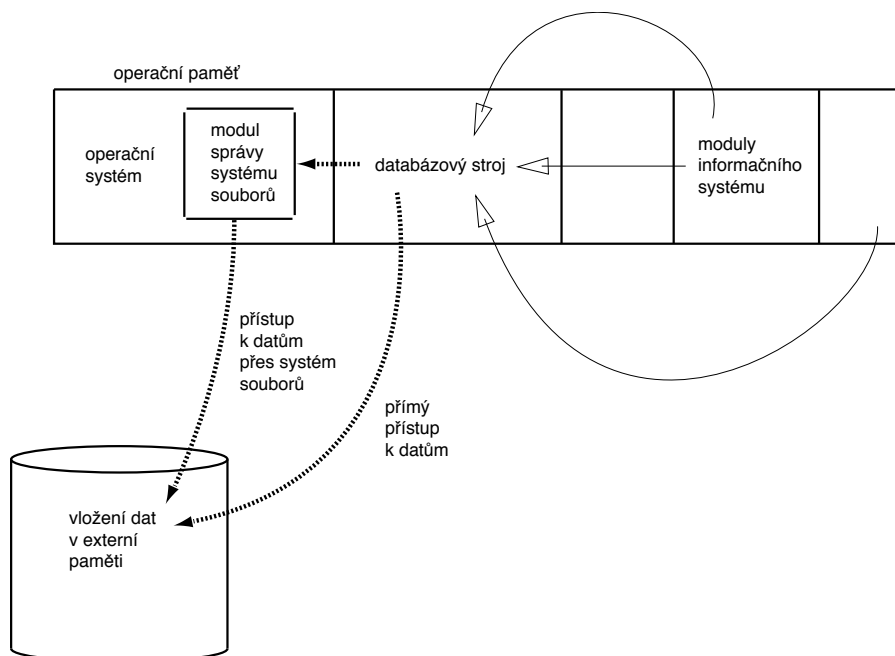
DBMS
a databázový stroj

Striktně vzato, odborníci rozlišují jako hlavní součásti databázového systému jednak jeho databázi, tj. způsob uložení dat, a jednak systém řízení báze dat (database management system, DBMS), tj. metody přístupu k datům databáze, které jsou implementovány aplikací s označením databázový stroj (database engine). Dalším pojmem, který se v praxi často používá, je databanka (data bank), tím je myšlen samotný obsah databáze, tedy data informací z pohledu jejich obsahového významu, které databáze obsahuje.

databáze
a databanka

Databázové systémy vznikly a slouží pro potřeby různých aplikací tam, kde dochází ke složitějším vztahům mezi samotnými daty, typicky při denním provozu např. obchodní firmy, průmyslového podniku, ale i školy nebo státní instituce. Používají se i tam, kde se přímý přístup aplikace k datům jednoho nebo více souborů stává příliš komplikovaným, např. již při programování promyšlenějších kartoték osob a institucí. Přestože jsou databázové systémy zaměřeny obecně pro současný přístup více uživatelů, jsou také dnes běžně používány při programování aplikací, které slouží i pouze pro jednu osobu pracující na osobním počítači bez účasti v počítačové síti.

Implementace umístování dat s databázovým přístupem je vlastně určitá vrstva mezi samotnou aplikací sloužící uživateli a systémem souborů operačního systému, jak ukazuje kresba III-10.



Kresba III-10: Umístění databázového systému mezi operační systém a aplikaci.

Databázový systém využívá podporu systému souborů, ale pro organizaci dat, kterou poskytuje samotné aplikaci prostřednictvím databázového stroje, si vytváří vlastní systém přístupu, který je zaměřen především na rychlou odezvu. Principiálně je dosahováno rychlého přístupu k datům speciálními algoritmy třídění dat, které jsou obvykle obecně známé, ale jejichž naprogramování a optimalizaci si výrobci databází obvykle úzkostlivě střeží. Dalším způsobem posilování výkonu databáze je využívání pouze základního poskytování dat operačním systémem pro přístup k datům na externích pamětech a vytváření vlastní speciální organizace dat na externím médiu mimo systém souborů. V rozporu s tímto uvedeným trendem je v současné době extrémně používaná progresivní databáze MySQL, která jednak striktně využívá výkonové vlastnosti systému souborů operačního systému (UNIX) a jednak je k dispozici zdarma i se zdrojovými kódy, její přístupové algoritmy k datům si tedy může kdokoliv přečíst.

Databázový systém je software, který slouží pro stavbu základů informačních systémů. Databázový systém je komerční produkt, který je vybírán z nabídky podle potřeb organizace, pro kterou je informační systém stavěn. Databázové systémy jsou poměřovány jednak množstvím dat, které dokážou efektivně ukládat, jednak inteligencí systému řízení báze dat a konečně také počtem přístupových míst, které databáze dokáže obsloužit najednou ve snesitelném časovém intervalu. Množství dat a počet přístupových míst jsou také podmíněny výkonem hardware a operačního systému a jejich vzájemným sladěním. Intelligence databáze ovšem vždy úzce souvisí s výrobcem a jeho

kvalitou. Pochopitelně je nejlépe mít výkonný hardware, dobrý operační systém a sofistikovaný databázový systém. To je možné, drahé, ale také pro řadu jednodušších informačních systémů i zbytečné. Typickým představitelem takového řešení je jistě databázový systém firmy Oracle nebo databázový systém DB2 firmy IBM. Nehledě na chytrou obchodní strategii těchto firem se jedná skutečně o databázové systémy s vysokou kvalitou DBMS. Naučit se je používat a programovat efektivně je ale i pro vzdělaného programátora věcí několika měsíců. Jejich instalace a sladění s platformou operačního systému a hardware je obtížné a časově náročné.

Tyto (a mnohé další) silné databázové systémy slavily úspěch zejména v době, kdy byl pro provoz informačního systému nutný hardware počítačů typu mainframe nebo supercomputer, jejichž celkový výkon se ale dá srovnat s výkonem hardware na principu dnešních technologií osobních počítačů. Údržba a provoz takových velkých výpočetních systémů tehdy vyžadovaly provozní tým několika specialistů a instalace databázového systému probíhala jiným způsobem než je tomu dnes. V té době také nebyl přístup k databázovým systémům z pohledu programátorů a potažmo uživatelů sjednocen tak, jak to přinesl tlak praxe do dnešních dnů. Každý databázový systém nabízel svůj vlastní jazyk pro programování a ovládání (viz další text). V průběhu postupné standardizace musely tyto starší databázové systémy podporovat jak dřívější verze, tak zaváděné standardy, protože musely zajistit v nových verzích podporu již běžících informačních systémů a současně získávat nové zákazníky. Tato dlouhodobá podpora zákazníků za současného vývoje standardů rozhraní komplikovala implementace dříve narozených databázových systémů. Naopak nárůst výkonu a kapacity hardware při jeho současném zlevnění umožnil vznik nových databázových systémů, které s sebou navíc ve své implementaci nevlekly svoji historii a které se přizpůsobily poptávce podpory i méně bohatých organizací. Takovým typickým představitelem je MySQL, který je navíc bez garance a podpory možné získat a používat zdarma.

dotazovací jazyky

Programátor aplikace oslovuje databázový stroj určitým strukturovaným způsobem, kterému říkáme *dotazovací jazyk*. Jedná se o typ programovacího jazyka, který je zaměřen především na přístup k datům s obecně zjednodušenou algoritmickou částí, kterou ponechává na samotné aplikaci. Dotazovací jazyk byl dříve definován výrobcem databáze a v jednotlivých databázích se výrazně lišil. Tlak praxe jednak na jednotný způsob programování a jednak na možnost snadného programování aplikací pro různé databáze (pro přenositelnost aplikací) přinesl snahu dotazovací jazyky standardizovat. Princip dotazovacího jazyka je ovšem také spojen se způsobem *datového modelování* (data modeling). Jedná se o princip vnímání souvislostí mezi daty. Pro *relační způsob modelování dat* byl definován a později standardizován dotazovací jazyk SQL (Structured Query Language). Jeho první návrh vznikl již v roce 1974 na půdě laboratoří firmy IBM (pod názvem Sequel). V roce 1987 byl mezinárodně standardizován, poslední vydaný standardizující dokument je z roku 2003, viz [ISOSQL2003]. Přestože princip relačního modelování není jediný, je v dnešní době nejrozšířenější. Vyjma SQL je za progresivní (a také poměrně rozšířený) považován jazyk QBE (Query-By-Example), jehož princip je také relační. Jiné principy organizace

SQL a relační model báze dat

databázi jsou např. deduktivní. Používá se síťové modelování dat nebo hierarchické modelování dat. Velkému zájmu se také těší objektové databáze, jejichž princip vychází z objektového přístupu k algoritmům (viz část Base Of I.2). V praxi však obvykle umí databázový systém pouze jeden uváděný způsob návrhu modelu dat. V případě dalšího zájmu je z české literatury jistě nejlepší zaměřit se na publikace [Pokorný1994] a [Pokorný1998], z nichž pak lze kvalifikovaným způsobem odkazově studovat principy databází.

Relační algebra je matematický prostředek pro práci s relacemi (operace s relacemi). Relace druhého stupně je kartézský součin dvou neprázdných množin. Mezi základní operace s relacemi patří projekce, selekce a spojení. Relaci dokážeme zobrazit formou tabulky o m řádcích a n sloupcích.

relační algebra

Pojem tabulka je také používán v praktických implementacích relačních databázových systémů. Každý řádek tabulky reprezentuje záznam (record) a každý sloupec tabulky pole (field). Položka je sloupec záznamu. Každá tabulka v databázi musí mít jedinečné jméno, každé pole musí mít v tabulce jedinečné jméno, pole má jednoznačný datový typ (např. pole znaků, číselná hodnota atp.). Primární klíč je definice podmnožiny polí tabulky tak, že jednoznačně určuje záznam tabulky. Např. několik sloupců: rodné číslo, jméno, příjmení a místo narození je primárním klíčem. Primárním klíčem je ale třeba také jenom sloupec s číselnou hodnotou, která je vždy o 1 zvýšená přidáním nového záznamu. Každá tabulka musí mít alespoň jeden primární klíč.

tabulka,
záznam, pole

primární klíč

Vlastnosti relace tedy jsou:

- v každém sloupci tabulky jsou položky téhož typu,
- v tabulce (relaci) nemohou být stejné řádky (záznamy),
- záznamy jsou identifikovány podle primárního klíče, na jejich pořadí tedy nezáleží,
- sloupce jsou pojmenovány, nezáleží na jejich pořadí.

Autorem relačního modelu báze dat je E. F. Codd, který byl jedním z členů týmu vývoje databázového systému s označením R firmy IBM v 70. letech dvacátého století, kde byl tento způsob poprvé použit.

Vyjadřovaný relační datový model definujeme diagramem E-R (entity – relationship) překládaným jako diagram entit a vztahů. Základem jsou entity, tedy objekty reálného světa, o kterých v databázi evidujeme určité údaje. Vzájemné vztahy mezi jednotlivými entitami nazýváme vztahy entit. Např. entitou je zaměstnanec organizace, u něhož evidujeme informace o jeho jménu, příjmení, datu narození, pracovním zařazení, příslušnosti k určitému pracovišti, měsíčním platu atp. Tabulka takové entity může např. obsahovat tyto záznamy:

diagram E-R

entita, vztah

Novotný	Petr	1962	programátor	výpočetní středisko	10.000,-
Horák	Jan	1965	kotelník	kotelna	3.000,-
Nenasyta	Pavel	1955	ředitel	ústředí	80.000,-
Drábková	Jana	1966	sekretářka	ústředí	10.000,-

...

atribut Tabulka musí mít jméno (zde např. ZAMĚSTNANEC). Každý sloupec tabulky je také pojmenován (např. PŘÍJMENÍ, JMÉNO, RN, ...). *Atribut* je funkcí, jejíž definiční obor je množina entit nebo vztahů. Lidsky řečeno se jedná o popis sloupce tabulky nebo popis vztahu mezi tabulkami. Tabulku si tedy můžeme představit jako souhrn atributů, datový model je souhrn tabulek a vztahů mezi nimi. Vztahy souvisí s atributy různých tabulek.

Již v našem jednoduchém příkladu vidíme opakující se informace ve sloupci příslušnosti k pracovišti. V případě mnoha řádků (a databáze mnoho řádků obsahují) dochází k neefektivnímu ukládání opakujících se informací. Další nevýhodou takového opakování textů je, že při změně názvu pracoviště musí být přejmenován každý text, který pracoviště takto identifikuje. Proto se např. zde pro pracoviště zavádí v relačním modelu nová entita pracovišť organizace a v příslušném sloupci tabulky zaměstnanců je uváděn pouze odkaz do této nové tabulky, např.

tabulka entity ZAMĚSTNANEC:

Novotný	Petr	1962	programátor	2	10.000,-
Horák	Jan	1965	kotelník	1	3.000,-
Nenasyta	Pavel	1955	ředitel	3	80.000,-
Drábková	Jana	1966	sekretářka	3	10.000,-

...

tabulka entity PRACOVISTĚ:

- 1 kotelna
- 2 výpočetní středisko
- 3 ústředí

Mezi entitou ZAMĚSTNANEC a PRACOVISTĚ jsme tak vytvořili vztah.

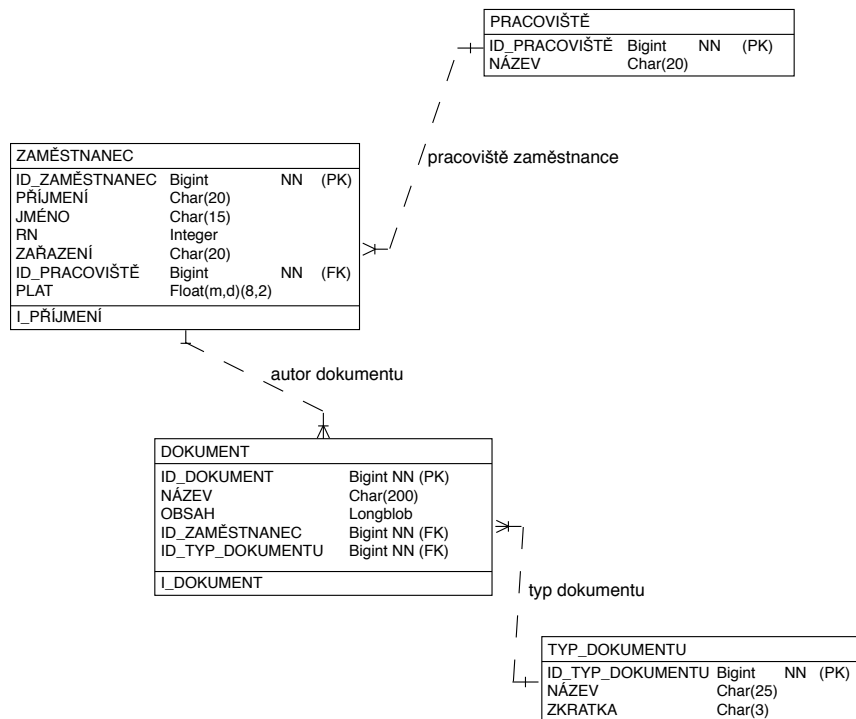
datový model Diagram E-R je grafické zobrazení všech entit a jejich vzájemných vztahů potřebného modelu. Říkáme také, že diagram E-R definuje *datový model* konstruovaného informačního systému. Terminologicky striktně vzato bychom pak měli používat výraz diagram ERA (entity – relationship – attribute), kde se také vyskytuje pojem atributu, praxe oba termíny ale obvykle nerozlišuje.

Na první pohled je zřejmé, že datový model může být velmi složitý a že jeho návrh je pro další práci na informačním systému velmi důležitý.

Popis datového modelu může být dán obrázkem, ale pro databázi je nutno jej vyjádřit formálně relační algebrou. SQL je programovací jazyk, pomocí něhož relační datový model vyjadřujeme. Diagramy E-R se dnes běžně vytvářejí v prostředí grafických nástrojů, kde konstruktér informačního systému definuje postupně všechny entity, jejich atributy a vztahy mezi entitami tak, že vzniká přehledné schéma datového modelu v grafické podobě. Jednou z funkcí takového grafického nástroje je převedení grafické podoby na textový zápis v SQL (generace skriptu SQL). Dnes obvykle každá komerční databáze takový grafický nástroj nabízí. Přestože má již SQL řadu let svůj standard, pro jednoznačnou a vhodnou definici v praxi stále nedostačuje a databázové systémy se ve své syntaxi SQL navzájem liší. Grafický nástroj pro definici

datového modelu výrobce určitého databázového systému proto produkuje kód SQL pouze pro mutaci podporovanou daným výrobcem. Důvody jsou ale i obchodní a proto se již běžně v praxi používají grafické nástroje dodávané třetí stranou, tj. ty, které umí modelovat pro databáze různých výrobců včetně exportu definice modelu do SQL v dané mutaci. Příkladem je jistě [Kunz2004].

Diagram E-R našeho příkladu ukazuje kresba III-11.



Kresba III-11: Ukázka fragmentu diagramu E-R.

V kresbě jsou definovány další tabulky, které vyjadřují produkci písemných dokumentů zaměstnanců. Vztahy jsou v kresbě III-11 vyjadřovány dle konvence spojnicí s rozdělením na jednom svém konci. Rozdělení symbolizuje více záznamů, ze kterých lze vybírat pro určení např. pracoviště, do kterého je zaměstnanec zařazen. Na jednom pracovišti pracuje více zaměstnanců, ale zaměstnanec nepracuje na více pracovištích současně. Jedná se o kardinalitu vztahu entit, tedy vzájemné zastoupení počtu entit ve vztahu. Uvedená kardinalita je typu 1:N (1 pracoviště, N zaměstnanců) a vyskytuje se nejčastěji. Roztřepení na obou stranách grafického vyjádření vztahu se používá při vztahu M:N. Pokud připustíme, že autorů jednoho dokumentu může být více, což je v praxi běžné, měli bychom mezi tabulkou zaměstnanců a jejich dokumentů vytvořit právě vztah M:N. Konečně je také možné používat vztah 1:1, kdy např. zaměstnanec sedí právě na jedné židli

**kardinalita
vztahu entit**

a nám se vzhledem k různým potřebám struktury datového modelu nehodí židli evidovat přímo v tabulce zaměstnanců, ale v samostatné tabulce židlí.

datový model ve
třech úrovních

Teorie návrhu datových modelů informačních systémů stanovuje 3 úrovně návrhu. Jsou to: úroveň *konceptuální*, *logická* a *fyzická (implementační)*. Jedná se vlastně o postupné přibližování od obecných úvah o datovém modelu až k jeho vyjádření v dotazovacím jazyce konkrétního databázového stroje.

Konceptuální návrh datového modelu je vyjádření diagramu E-R ve smyslu definice entit a vztahů mezi nimi.

klíč, index

Logický datový model obsahuje už seznam všech atributů entit, způsobů jednoznačného určení jednotlivých záznamů pomocí *klíčů* (key), definice vztahů pomocí atributů entit či způsobu potřeby setřídění záznamů v tabulkách pomocí *indexů* (index).

Fyzický datový model je zápis logického datového modelu v jazyce konkrétní databáze. Přestože v praxi se většinou při návrhu již takto uvažuje na úrovni logického modelu. Nezřídka se totiž používá při vyjadřování logického modelu již konkrétní mutace SQL vybrané databáze.

Obecně by přístup inženýra informačního systému neměl zanedbávat žádnou z uvedených úrovní. Abstrakce konceptuální úrovně totiž umožňuje diskusi s odborníkem, který je zadavatelem informačního systému a mnohdy s ním lze konzultovat části datového modelu ještě na logické úrovni. Informatik přitom z pozice své profese dokáže nejlépe posoudit úskalí požadavků návrhu pro různé typy databází. Konceptuální a částečně i logická úroveň návrhu datového modelu tak obvykle určí, který databázový stroj se pro potřebné řešení nejlépe hodí.

Uvedený fragment datového modelu kresby III-11 je v jazyce SQL databáze MySQL na fyzické úrovni vyjádřen takto:

```
Create table ZAMĚSTNANEC (
    ID_ZAMĚSTNANEC Bigint NOT NULL,
    PŘÍJMENÍ Char(20),
    JMÉNO Char(15),
    RN Int,
    ZAŘAZENÍ Char(20),
    ID_PRACOVISTĚ Bigint NOT NULL,
    PLAT Float(8,2),
    Primary Key (ID_ZAMĚSTNANEC)) TYPE = MyISAM
ROW_FORMAT = Default;
Create table PRACOVISTĚ (
    ID_PRACOVISTĚ Bigint NOT NULL,
    NÁZEV Char(20),
    Primary Key (ID_PRACOVISTĚ)) TYPE = MyISAM
ROW_FORMAT = Default;
```

```
Create table DOKUMENT (  
    ID_DOKUMENT Bigint NOT NULL,  
    NÁZEV Char(200),  
    OBSAH Longblob,  
    ID_ZAMĚSTNANEC Bigint NOT NULL,  
    ID_TYP_DOKUMENTU Bigint NOT NULL,  
    Primary Key (ID_DOKUMENT)) TYPE = MyISAM  
ROW_FORMAT = Default;  
Create table TYP_DOKUMENTU (  
    ID_TYP_DOKUMENTU Bigint NOT NULL,  
    NÁZEV Char(25),  
    ZKRATKA Char(3),  
    Primary Key (ID_TYP_DOKUMENTU)) TYPE = MyISAM  
ROW_FORMAT = Default;  
Create Index I_PŘÍJMENÍ ON ZAMĚSTNANEC (PŘÍJMENÍ);  
Create Index I_DOKUMENT ON DOKUMENT (ID_ZAMĚSTNANEC,NÁZEV);
```

Přestože je pro neodborníka uvedený text zcela jistě nečitelný, pokusme se v něm orientovat. Text `Create table` uvádí definici tabulky s následně uvedeným jménem. V příkladu jsou definovány tabulky se jmény `ZAMĚSTNANEC`, `PRACOVISTĚ`, `DOKUMENT` a `TYP_DOKUMENTU`. Jako primární klíč každé tabulky (definice textem `Primary Key`), tedy způsob jednoznačného určení každého záznamu v tabulce, jsme ve všech tabulkách použili celočíselnou identifikaci, která je běžnému uživateli při práci s databází a informačním systémem skryta (používá se termín vnitřní identifikace záznamu). Jedná se o atributy, jejichž jméno začíná textem `ID_`. Každý záznam tak má přiděleno celé číslo, které jej identifikuje. Každý atribut tabulky je definován samostatně na jednom řádku, za jeho názvem (např. `PŘÍJMENÍ`) následuje definice typu dat (`Char(20)`, řetězec s možností délky nejvýše o 20 znacích). Typy formátu atributů mohou být textové řetězce (`Char`, `Varchar`, `Text`), celočíselné hodnoty (`Integer`, `Bigint`) nebo číselné hodnoty s daným počtem desetinných míst (`Float`, `Double`), ale i datum a čas (`Date`, `Time`, `Datetime`), logické typy (`Bool`) nebo dokonce celé binární soubory (`Blob`, `Longblob`), jako jsou dokumenty nebo obrázky. Typy formátu atributů, jejich rozsah hodnot a jejich pojmenování jsou specifické pro daný typ databáze. Uvedené typy a struktura jejich použití je zde citací databáze MySQL, v jiných databázích je odlišná, ale mnemonicky podobná.

Vztahy v příkladu zápisu v SQL jsou vyjádřeny pouze uvedením odpovídajících primárních klíčů v jiných tabulkách. Říkáme, že tabulky obsahují cizí klíče (foreign key). Dále mohou vztahy kromě typu kardinality mít také parcialitu, tj. povinnost vyjádření u každého záznamu tabulky. V uvedeném příkladu je parcialita uvedena textem `NOT NULL` u atributů cizích klíčů, což znamená, že nemohou obsahovat prázdné položky (`NULL`), tedy případ, kdy by vztah u některých záznamů nebyl určen.

parcialita

index Poslední dva řádky příkladu definují úvodním textem `Create Index` způsob, jakým mají být záznamy dané tabulky ukládány v databázi tak, aby výběr určitého požadovaného záznamu byl co nejrychlejší. Říkáme, že definujeme index nad tabulkou. U tabulky `ZAMĚSTNANEC` je rozumný požadavek vybírání záznamů podle `PŘÍJMENÍ`, protože uživatel obvykle takto osoby v databázi hledá. Z druhého řádku definice indexu také vyplývá, že přístup může kombinovat i několik atributů. Podpora rychlého přístupu k tabulce dokumentů je zde podle současně určeného zaměstnance a názvu (nebo jeho části) dokumentu.

Databáze implicitně indexují tabulky podle primárních klíčů, nad tabulkou dále běžně definujeme více indexů pro podporu zrychlení různých způsobů přístupu k tabulce.

Zápis fyzického datového modelu v jazyce SQL je předložen databázovému stroji. Vzniká tak prázdná databanka, tedy struktura dat, do níž je možné následně data ukládat, číst je a měnit.

Následujícím krokem při práci s databankou je vložení základních dat do tabulek, které vytvářejí typizaci záznamů. Jedná se především o číselníky. V našem příkladu je číselníkem `TYP_DOKUMENTU` a `PRACOVIŠTĚ`.

číselník Obecně je číselník tabulka databáze, která obsahuje záznamy bazálních neměnných informací v té oblasti lidské činnosti, pro niž je informační systém určen. Číselník je při běžném provozu informačního systému označen jako tabulka, z níž je možné záznamy jen číst. Pouze je-li to skutečně nezbytné, je rozšiřována o další záznamy, a to zvláštním přístupem správcem databáze. Jedná se totiž o informace, které jsou v ostatních tabulkách používány pouze odkazem a obsah záznamu číselníku je určující pro další chování informačního systému. Tabulka `TYP_DOKUMENTU` je číselník. Pokud se při prvním návrhu datového modelu stanoví, že dokument s číselným kódem (primárním klíčem) ¹ je faktura, ² cestovní příkaz, ³ zápis z valné hromady, může již např. zobrazování obsahu těchto dokumentů uživateli při výběru některého z nich podléhat jiným dále dokumenty zpracovávajícím algoritmům.

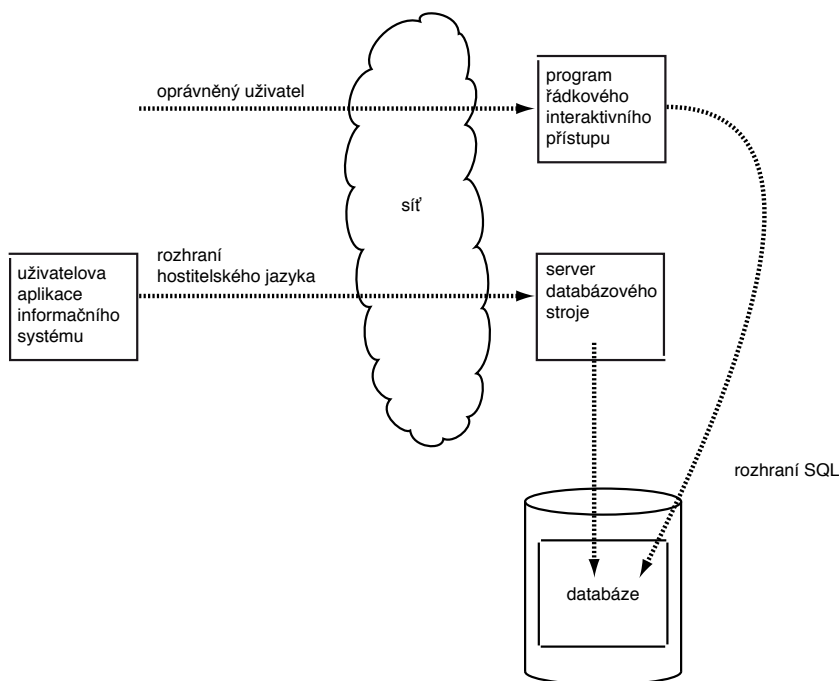
INSERT Dotaz jazyka SQL, kterým vkládáme záznam do tabulky databáze, je `INSERT`. Naše číselníky `TYP_DOKUMENTU` a `PRACOVIŠTĚ` můžeme např. naplnit takto:

```
INSERT INTO TYP_DOKUMENTU VALUES (1,'faktura','FAK');
INSERT INTO TYP_DOKUMENTU VALUES (2,'cestovní příkaz','CP');
INSERT INTO TYP_DOKUMENTU VALUES (3,'zápis z valné
hromady','ZVH');
INSERT INTO PRACOVIŠTĚ VALUES (1,'kotelna');
INSERT INTO PRACOVIŠTĚ VALUES (2,'výpočetní středisko');
INSERT INTO PRACOVIŠTĚ VALUES (3,'ústředí');
```

což bývá často součástí definičního skriptu databáze. Sémantika dotazů uvedené sekvence je jistě zřejmá. Za slovem `INTO` je uvedeno jméno již existující tabulky a za `VALUES` jsou v závorkách postupně uvedeny hodnoty

atributů oddělené čárkou. U textů používáme pro označení jejich začátku a konce znak apostrof.

Výchozím stavem pro práci informačního systému s databází je tedy databanka, která vyjma základních číselníků neobsahuje žádná data. Přestože všechny dále uváděné příklady dotazů jazyka SQL lze provádět v interaktivním řádkovém textovém režimu, který každá databáze nabízí, programátor dotazuje databázi ve verzi *hostitelského programovacího jazyka*. Jedná se o doplněk některého programovacího jazyka, o rozšíření pro přístup k databázi (např. knihovnou funkcí). Vzhledem k tomu, že samotný přístup k databázím různých výrobců je různý, hostitelský jazyk obsahuje zvláštní rozhraní (sadu funkcí) vždy pro určitou databázi. Tento princip je patrný v obecně dostupných programovacích produktech pro stavbu menších informačních systémů, jako je např. PHP. Pohledem do jeho dokumentace vidíme, že PHP nabízí přístup k databázím od MySQL přes Interbase, SyBASE až po Oracle. Vždy se jedná o sadu funkcí jiných jmen. Programátor využívá hostitelský jazyk pro přístup k databázi a současně jej používá jako programovací jazyk budovaného informačního systému. Výběr tohoto programovacího jazyka je věcí potřeb jednak rozhraní uživatele a jednak celkového zaměření informačního systému. Každopádně ale hostitelství jazyka vůči databázi spočívá ve schopnosti odeslání dotazu SQL databázi a získání její odezvy (např. obsahu tabulky) tak, aby s takto získanými daty mohl programátor pracovat, tj. zobrazovat je uživateli v grafické podobě nebo je dále zpracovávat (přepočítat atp.), jak ukazuje kresba III-12.



Kresba III-12: Přístup k databázi SQL z programovacího jazyka.

SELECT V jazyce SQL používáme pro získání záznamů tabulky dotaz `SELECT`. Např. chceme-li v programovacím jazyce PHP získat obsah tabulky `TYP_DOKUMENTU`, která je uložena v databázi MySQL se jménem `priklad` (databázový stroj běžně poskytuje několik databank), používáme jako programátor tento kód:

```
$link=mysql_connect('localhost','skocovsky','enjoyer');
mysql_select_db('priklad');
$result=mysql_query('SELECT * FROM TYP_DOKUMENTU');
while($typy_dokumentu=mysql_fetch_object($result))
{
    ...
}
mysql_close($link);
```

Program v PHP se připojuje k databázi, kde má uživatel `skocovsky` určen prostor pro práci s databankou s heslem `enjoyer` příkazem `mysql_connect`. Vybírá si databanku se jménem `priklad`. Pomocí funkce `mysql_query` odesílá databázovému stroji místního počítače (`localhost`) dotaz `SELECT`. V cyklu `while` programovacího jazyka PHP pak postupně zpracuje (uvádíme jen..., např. zobrazí v okně) výsledky tohoto dotazu, tedy předané záznamy z tabulky, protože je má postupně vždy k dispozici v proměnné se jménem `typy_dokumentu`. Příkazem `mysql_close` se od databázového stroje odpojuje. Dotazem `SELECT` zde přitom požadujeme obsah všech sloupců (tento význam určuje znak `*`) a vzhledem k tomu, že jsme nijak blíže neurčili záznamy požadované tabulky, získáme tak seznam všech záznamů. Omezení bychom např. stanovili takto:

```
SELECT * TYP_DOKUMENTU WHERE ZKRATKA='CP';
```

a získali bychom pouze ty řádky, které obsahují přesně text `CP` ve sloupci atributu `ZKRATKA`.

UPDATE, DELETE Dotaz `SELECT` má velmi široký formát použití. Je základním přístupovým mechanismem pro čtení dat z databáze. Umožňuje data seřadit dle zadaného sloupce (použitím `ORDER`), vybírat pouze záznamy podléhající určité podmínce (pomocí `LIKE`, např. že ve sloupci `PŘÍJMENÍ` je obsažen text „ová“) nebo obsahy tabulek dle daných kritérií spojovat a teprve potom výsledek předávat hostitelskému jazyku. `INSERT` a `SELECT` jsou základní dotazy programátora. Stejně důležitý je ale také dotaz `UPDATE`, kterým lze změnit určitou položku daného záznamu v tabulce databáze, dotazem `DELETE` záznam z databáze odstraníme. Výbornou publikací s kvalitním přehledem všech dotazů SQL a přitom srozumitelně vedeným výkladem i pro neprogramátory je jistě [Šimůnek1999]. Pro každého manažera, který uvažuje o vybudování a následném provozu informačního systému pro potřeby své organizace, je tato kniha výbornou pomůckou pro získání přehledu možností práce s daty. Pro první přiblížení je také základní seznam dotazů SQL a příkladů jejich použití uveden v Příloze F.

**třívrstvá
architektura**

Databáze ve svém principu znamenají především obecný prostředek pro efektivní ukládání dat v systému souborů tak, aby přístup k nim byl snadný a rychlý, a to podle definice potřeb provozu. Přístupové mechanismy se opírají o rozhraní, které databáze poskytuje. V některých případech dokonce

z nedávné historie víme, že takové rozhraní může být implementováno pouze pomocí knihovny funkcí, která je součástí uživatelského programu, takže data jsou požadována nikoliv po databázovém stroji, ale program data čte sám z databanky. Inteligence přístupu k datům byla v tomto případě celá součástí uživatelského programu. To zjevně neobstojí např. tehdy, má-li ke stejným datům přistupovat více uživatelů současně a kdy je potřeba klienty uživatelů vzájemně synchronizovat. Rovněž řízení přístupu více takových klientů k datům současně je opět přeneseno na systém souborů operačního systému, který vnitřní inteligenci ukládání dat databází nezná a postupuje pak neefektivně. Obsluha požadavků klientů databázovým strojem jako samostatně běžícím procesem představuje osvědčený a v tomto textu uváděný způsob řešení. Databázový stroj přebírá od klientů požadavky na dotazy databanky a sám jejich souběh nebo efektivitu přístupu organizuje do jednoho proudu práce s daty. Ukazuje se, že se jedná o přístup, který je sice z hlediska provozu centrální databáze náročnější, ale je také výkonnější, spolehlivý a bezpečný. Je označován termínem *třívrstvá architektura přístupu k databázi*. Instalace databázového stroje a jeho rozběh se dnes ale používá i u jednoduchých jednouživatelských aplikací, a to z důvodu jednotného přístupu, případné přenositelnosti a také možnosti rozšíření na víceuživatelský režim. Aniž si to uvědomujeme, přijímáme zde způsob myšlení paralelního průběhu různých procesů, které se v určitých chvílích setkávají nad týmiž daty. Takový víceprocesní způsob myšlení je sice obtížnější, denně jej ale prožíváme v praktickém světě, který se ve výpočetních systémech snažíme modelovat.

Architektura informačního systému vychází z výběru databáze, který je určen základní úvahou nad požadavky, které stanoví zadavatel informačního systému. Hovoříme zde o zadání problematiky, která má být řešena vybudováním informačního systému. Úkolem řešitele (informatika) požadovaného zadání je nejprve zadavateli předložit dokument, který se zabývá *analýzou*. Někdy je tento dokument také označován jako *projektová dokumentace*.

analýza
informačního
systému

Analýza musí obsahovat charakteristiku modelované skutečnosti s odkazem na požadavky zadání a poznatky získané z osobních diskusí se zaměstnanci organizace, kteří budou výsledný informační systém používat. Součástí analýzy je nutně logický datový model s podrobným popisem zdůvodňujícím jeho strukturu podle navrhovaných přístupových dotazů uživatelů. Struktura datového modelu určuje výběr databáze. Ale nejen ona. Analýza musí obsahovat také odhad objemu ukládaných dat a odhad objemu toků těchto dat. Tok dat vychází z *procesní analýzy*, která by měla být také součástí výsledného dokumentu, přestože ji mnohdy zadavatelé nepožadují. Procesní analýza je totiž rozpis základních algoritmů zpracovávání dat a ty jsou často v době zadání známy většinou jen neurčitě. Přesto není na škodu procesní analýzu provádět vždy. Jak řešitel, tak zadavatel a jeho zaměstnanci jsou totiž nuceni uvažovat z hlediska zpracovávání ukládaných dat, což je pohled na dynamickou proměnu dat, kdežto datový model je pohled statický a většinou si jej člověk kupodivu neuvědomuje v závislosti na plynoucím čase. Odhad nároků na velikost databanky a složitost a četnost přístupů k ní určuje

požadovaný výkon a inteligenci databázového stroje. Výběr databázového stroje a požadavek na jeho výkon souvisí i se stanovením požadavků na operační systém a hardware. Z hlediska informačního systému jako celku a jeho vnitřní struktury představuje analýza základ návrhu z pohledu potřeb centrálního zpracovávání dat organizace.

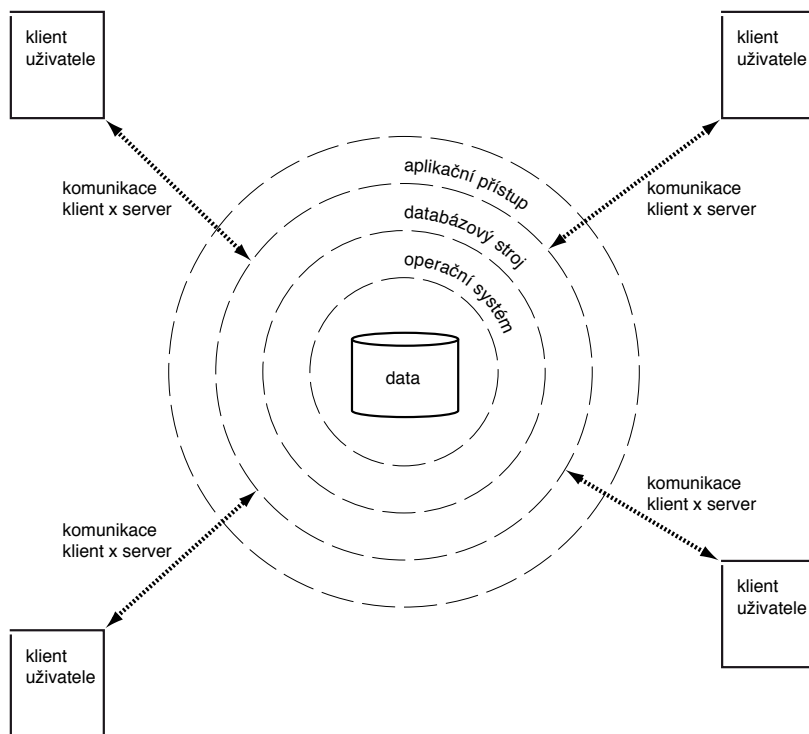
uživatel a role

Druhou, neméně důležitou částí návrhu informačního systému je vedle dokumentu analýzy přístup uživatelů. Podstatný je jistě jejich počet. Ten se odhaduje především z počtu přístupových míst, tedy terminálů informačního systému. Terminálem může být samostatná výpočetní jednotka (PC) nebo pouze displej grafického podsystému (v X, viz část III.2). Obvykle je dnes pro obsluhu přístupových míst navrhována počítačová síť, a to jednak z pohledu místního (budova ústředí organizace, budovy poboček), tak vzdáleného (propojení ústředí s pobočkami). Návrh tak může zahrnovat strukturu hardware a potřebného síťového software pro nutnou podporu požadovaného informačního systému. Návrh sítě ale souvisí také s její průchodností, tj. se zajištěním minimální časové odezvy centrální databáze na základě požadavku uživatele. Výpočet takové struktury je obtížný, ale analýza musí obsahovat alespoň zdůvodněný odhad. Pro přístup uživatelů je také podstatný návrh jejich rozhraní přístupu. Jde o způsob používání oken, menu, tlačítek, odkazů a jiných prvků dnešního obecného rozhraní interaktivního přístupu uživatele k výpočetní technice tak, aby sémanticky odpovídal záměru informačního systému. Znamená to, že informatik provádějící analýzu také určí a zdůvodní programovací jazyk, ve kterém bude uživatelský přístup naprogramován. Víme z předchozí části tohoto textu, že se jedná o určení programovacího jazyka klientů, které budou komunikovat se serverem databázového stroje v centru informačního systému. Přestože výrobci databází nabízejí své hostitelské jazyky pro programování klientů, ty nemusí být právě pro programování klientů vhodné. Skutečně, praxe je dnes již tak daleko, že připojení k databázovému serveru databáze určitého výrobce je možné z mnoha programovacích jazyků jiných výrobců (kteří dokonce mohou nabízet vlastní databáze). Alternativou z třetí strany jsou dnes velmi rozšířené a používané nástroje pro podporu prohlížečů stránek WWW Internetu (PHP, JSP atd.), pomocí nichž lze komunikovat s centrální databází. Jsou volně k dispozici bez nutnosti licenčních poplatků jak při programování, tak při samotném provozu. Výběr programovacího jazyka klientů uživatelů pak určuje požadavky na hardware terminálových pracovišť z pohledu jeho typu a potřebného výkonu. Analýza uživatelova rozhraní dále také musí obsahovat hierarchizaci uživatelů ve vztahu jejich přístupových práv k datům. Jedná se o stanovení obecných potřebných úrovní přístupu. Může to být uživatel, který pouze data pořizuje (což je např. prodáváčka v pokladně supermarketu). Nebo se jedná o uživatele, který má možnost prohlížet seznamy všech pořízených dat či dokonce vytvářet statistické výsledky vedení provozu (majitel řetězce supermarketů). Nebo to může být uživatel, který má právo manipulace s daty číselníků, přístupu k zálohování části nebo všech dat, jejich obnovy, odstraňování již nepotřebných dat atd. (správce databáze, database administrator). Návrh struktury typů uživatelů

jistě také ovlivní návrh datového modelu. SQL nabízí pro definici a práci s tímto prvkem databází uživatele (user) a jejich práci v rolích (role).

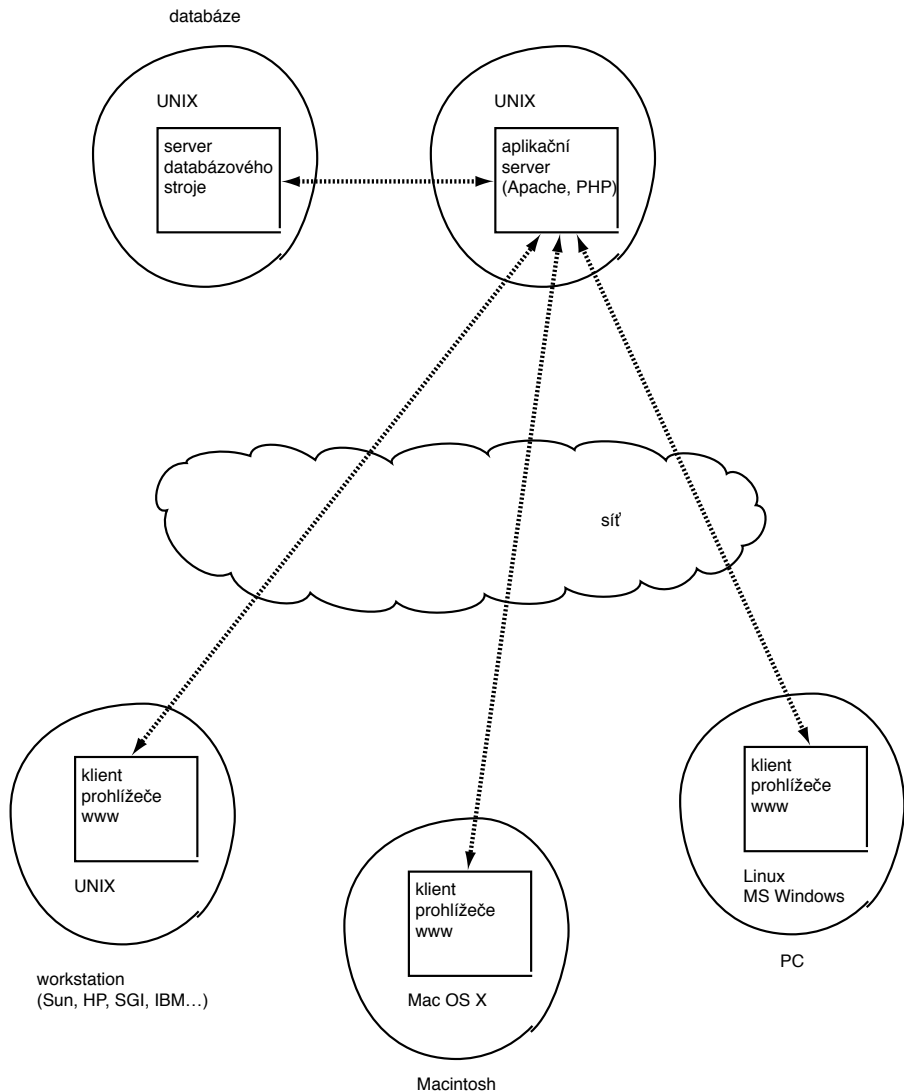
Nutnou součástí analýzy informačního systému je také časový harmonogram a cena všech navrhovaných částí, jak software (nákup databáze, práce programátorů), tak hardware. Není to časté, ale analýza by měla také alespoň orientačně obsahovat požadavky na samotné vedení provozu, a to jak ve smyslu inovace technologií, údržby a potřebného personálu, tak změn vycházejících z proměny struktury a chodu organizace, a vyjadřovat to vše finančně.

Informační systém organizace v dnešním pojetí jeho implementace v IT ukazuje kresba III-13.



Kresba III-13: Principy centralizovaného informačního systému.

Kresba ukazuje běžně používané schéma postupného vrstveného přístupu k datům v konečné podobě uložené na discích výkonného hardware centrální databáze. Dnešní velmi častou konkrétní podobu, tj. využití technologií internetu pro řešení provozu informačního systému, ukazuje kresba III-14.



Kresba III-14: Intranet centralizovaného informačního systému.

Jedná se vlastně o využití prohlížeče stránek Internetu jako tenkého klienta v komunikaci se silným serverem centrální databáze. Jde o využití technologií obecného internetu jak pro spojení klienta se serverem na úrovni síťové a transportní vrstvy, tak na úrovni aplikační. Síťovou aplikaci tvoří prohlížeč stránek a server HTTP, jehož příkladem je uvedený software Apache. Síťová aplikace bývá rozšířena o aplikační server, který zajišťuje spojení mezi serverem HTTP a databázovým strojem. Takovým propojovacím software je právě např. programovacím jazykem ovladatelný produkt PHP nebo JSP. Aplikační server, který dotazuje uživatele prohlížečem

aplikační server

internetu pomocí běžných formulářů, tedy obsahuje jak server HTTP, tak programovací rozhraní přístupu k databázovému stroji. Kresba ukazuje situaci, kdy je databázový stroj jako proces provozován na samostatném hardware, což někteří výrobci databází doporučují (např. Oracle dokonce podmiňuje) z důvodů potřeby zajištění dostatečného výkonu. Běžně tomu tak v praxi nemusí být. Jak server databázového stroje, tak aplikační server se dělí o výkon téhož hardware na jednom uzlu sítě TCP/IP. Termín *Intranet*, který jsme v textovém označení obrázku použili, je termín, kterým by měl být označován informační systém sloužící pro vnitřní potřebu určité organizace provozovaný za použití technologií internetu, a to nezávisle na tom, zda spojení mezi klientem a serverem prochází pouze vnitřním fyzickým rozvodem organizace (místní sítě) nebo veřejným Internetem. Často se v praxi používá termín Intranet pro označení software vnitřní sítě organizace, která je oddělena od Internetu (vnější sítě) pomocí bezpečnostní technologie firewall, a to nezávisle na síťových technologiích, což naznačuje obecné vnímání takového informačního systému. Ten je nedostupný veřejnosti, přestože kdokoli z této veřejnosti se může jednoduchým zásahem správce informačního systému stát jeho uživatelem, a to přidělením vstupního jména a hesla. Také z tohoto důvodu je dnes koncepce informačních systému na bázi Intranetu obecně oblíbená, vyhledávaná a podporovaná, přestože problémy s bezpečností jsou jistě značné (viz další část III.4).

Intranet

Jako příklad programování spojení aplikačního a databázového serveru nám může posloužit doplnění již uvedeného příkladu v jazyce PHP. Server HTTP po oslovení klientem čte požadovanou stránku z disku uzlu s aplikačním serverem. Stránka by měla být popisem v jazyce HTML, který má server sítě odeslat klientu a ten jej má podle značek (tagů) formátovat v okně uživateli ke čtení. Ještě před odesláním však server prochází text stránky a části textu uvedené speciálním tagem `<?php` a ukončené?> předá pro zpracování běžícímu procesu – serveru PHP. Server PHP interpretuje předaný text jako program. Součástí tohoto programu mohou být i dotazy SQL pro databázový server, které PHP skutečně databázovému serveru předá, a to v okamžiku, kdy k nim při sekvenčním zpracovávání programu přistoupí. Po vyhodnocení dotazu SQL získá PHP odpověď databázového serveru a pokračuje v interpretování programu. Výsledkem práce serveru PHP je kód jazyka HTML, ve který je proměněna část zdrojového kódu stránky uvedeného tagu `<?php`. Ta je předána serveru HTTP, který ji vsune namísto textu tagu, a stránku, která již obsahuje pouze kód HTML, předává sítě klientu. Např. zdrojový kód

```
<HTML>
<BODY>
<CENTER>Seznam všech možných typů dokumentu u nás je:</
CENTER><BR>
<?php
$link=mysql_connect('localhost','skocovsky','enjoyer');
mysql_select_db('prikklad');
$result=mysql_query('SELECT * FROM TYP_DOKUMENTU');
```

```

while($typy_dokumentu=mysql_fetch_object($result))
{
    printf($typy_dokumentu->ZKRATKA."": ".$typy_dokumentu-
    >NÁZEV."<BR>\n");
}
mysql_close($link);
?>
</BODY>
</HTML>

```

server HTTP klientu odesílá v HTML takto:

```

<HTML>
<BODY>
<CENTER>Seznam všech možných typů dokumentu u nás je:</
CENTER><BR>
FAK: faktura<BR>
CP: cestovní příkaz<BR>
ZVH: zápis z valné hromady<BR>
</BODY>
</HTML>

```

a prohlížeč stránek Internetu zobrazí:

Seznam všech možných typů dokumentu u nás je:

```

FAK: faktura
CP: cestovní příkaz
ZVH: zápis z valné hromady

```

Jazyk PHP, který programátor používá při programování, je mutace syntaxe jazyka C se změnami, které některá příliš striktní nařízení jazyka C zlehčují a programátorovi usnadní práci tak, že i pokročilý uživatel může PHP použít pro dynamické proměny svých stránek WWW. Jazyk PHP však nenabízí struktury, pomocí nichž lze konstruovat větší systémový celek tak, aby všechny jeho části zůstaly konzistentní, což je např. podpora modulárního nebo dokonce objektového programování. JSP (Java Server Pages), který byl navržen a implementován na principech progresivního programovacího jazyka Java, potřebné programovací techniky nabízí a podobně jako Java si je od programátora vynucuje. Nestudovaný programátor jim však těžko rozumí. Navíc řadu jednoduchých požadavků na dynamickou část stránek WWW je zbytečné v JSP programovat, když zde postačuje PHP. Tomu také odpovídá současné četné zastoupení využívání PHP v dnešním Internetu. Kromě uvedených dvou technologií lze však (rovněž bez nutnosti platit licenční poplatky) najít dnes v Internetu software téhož zaměření, jakým je např. Webware (viz www.webware.sf).

Využití prohlížeče stránek internetu pro realizaci klienta je výhodné. Zobrazovací software je dnes totiž implementován ve všech používaných operačních systémech koncových uživatelů. Síťová aplikace tak nepotřebuje mutace pro různé operační systémy, provozovatel informačního systému

není nucen se orientovat na jeden typ koncových stanic a také není pro nastavení uživatelského přístupu potřeba žádné mnohdy problémové a složité instalace klienta. Nevýhoda je ovšem v možnostech HTML, které jsou chabé vzhledem např. k chování aplikace pro potřeby grafického rozhraní. Nároky při konstrukci informačního systému, pokud se jedná o více než jen tabulkové zpracování a zobrazování přepočtených dat databáze, jsou pak limitovány možnostmi HTML.

Proto je součástí dnešního vývojového hledání obecně dostupná technika programování klientů, jejichž výsledná provozovaná aplikace má podporu všech operačních systémů a současně nabízí chování za využití všech dostupných výpočetních zdrojů uživatelského koncového rozhraní. Jistě je takovou technologií programovací jazyk Java. Alternativu ale dnes tvoří i skriptovací jazyky, jako je Perl nebo Python. Java jako fenomén je progresivní programovací technologií již od svého vzniku (především za podpory firmy Sun) a i přes odpor firem, které se snaží získat nad podobnými technologiemi monopol, je Java stále používána a implementována jako součást nových programovacích technik (jako je např. JavaScript jazyka HTML nebo právě již zmiňovaný JSP).

Každý atribut v databázové tabulce má určený svůj typ. Postupně s vývojem uchovávání dat v databázích a s vývojem manipulace s daty se ukázala potřeba obecnějších typů než je řetězec znaků omezené délky (typ `Char`), číselná hodnota (`Int`, `Float`) nebo datum a čas (`Datetime`). Rádi bychom např. v databázi ukládali v některých sloupcích text neomezené délky (tedy neomezený řetězec znaků) nebo soubor s dokumentem či dokonce obrázek v libovolném grafickém formátu. Jak jsme si ukázali v uvedeném příkladu, databáze např. pro ukládání celých dokumentů (ať již ve formátu MS Word či HTML atd.) nabízejí pro sloupec typ s obecným označením `BLOB` (Binary Large Object). V databázi tak mohou být ukládány jakékoliv obsahy binárních souborů, a to ať už obrázky, dokumenty nebo programy. Strukturální práce s takovým typem databázové položky je ale problematická. Obtížné např. dokážeme v rámci dotazu `SELECT` určit výběr pouze fotografií se známými osobnostmi nebo jen ty, kde převažuje zastoupení modré barvy. Rovněž indexování podle typu `BLOB` databáze odmítají právě z důvodů obsahu, který je sekvencí nul a jedniček bez daného kontextu. V databázi MySQL lze pro výhradně textové obecně libovolně dlouhé sloupce použít typ `Text`, v rámci kterého pak můžeme používat např. vyhledávání nebo porovnávání částí položek tohoto typu. I zde se však jedná o omezení, např. u tohoto typu nelze požadovat indexaci. Z důvodů obtížné manipulace se sloupci typu `BLOB` a z něj odvozených typů je nutno ve struktuře odpovídající tabulky či datového modelu vůbec ukládat popisné informace, které jsou v rámci konstruovaného informačního systému důležité. Vše se komplikuje, stavíme-li informační systém, kde je vyžadována potřeba rozložit daný grafický obraz na důležitá seskupení, jako je např. na satelitním snímku oddělení silnice od řeky, určení umístění mostu přes řeku nebo dráhu železniční tratě. Pro potřeby stavby (nejenom) grafických systémů s vazbou na kartografii byly v nedávné době vyvinuty nástroje pro programování informačních systémů

GIS

typu GIS (Geografic Information System). Ve spolupráci s databází tyto nástroje umožňují spojovat datovou a grafickou část v sémantický celek. Principem je práce ve vrstvách, které definují a popisují jednotlivé zájmové části plošného popisu, jako je např. síť silnic nebo vodních toků atp. V rámci překládání jednotlivých vrstev přes sebe a ve spojení s informacemi ohledně souřadnicemi stanovených lokalit v datových tabulkách databáze je možné uživateli nabízet poměrně komfortní způsob mapování povrchu Země.

Jistě si také dokážeme představit grafický výstup ze systému pracující na uvedeném principu, který dokáže zobrazovat simulace přírodních procesů pro potřeby vědeckého výzkumu nebo systému zpracování rentgenových snímků v lékařství, o architektuře a jejím začlenění v krajině nemluvě. S architekturou ale ještě např. výrazně souvisí třetí rozměr. A s lékařstvím např. analýza obrazu, tedy získávání strukturovaných informací na základě rozboru obrazu. Ve všech případech se jedná o poměrně složité systémy, které jsou v současné době jednak velmi drahé a jednak jsou stále teprve ve svých začátcích, přestože science fiction nás o dalším vývoji nenechává na pochybách.

Databáze s obsahem binárních dat je středem zájmu při skladování multimediálních objektů, tedy zvuků, pohyblivého obrazu, jejich spojování do filmů nebo interaktivních aplikací. V odborných kruzích se hovoří o podpoře multimediálních aplikací a rozšířené relační technologii pro jejich podporu. Jistě se jedná o oblast, která zaznamená rychlý růst, protože trh s produkty software tohoto typu je velký a hladový. Nejedná se totiž pouze o podporu stavby informačních systémů, ale především o koncová pracoviště multimediálního charakteru, jako jsou střížny filmů, hudební software atp.

ISO SQL Doba, kdy obsah provozované databanky daného typu určitého výrobce byl uzavřený celek dostupný pouze striktně danými mechanismy výrobce databáze se specializovanými typy sloupců tabulek či struktury datového modelu, je věcí historie. Již standardizace SQL v ISO sjednotila pohled programátorů a uživatelů na databáze a umožnila zaměřit se při výběru potřebné databáze spíše na omezení velikosti tabulek a počtu sloupců, na celkovou velikost databáze a její výkon. V konečném důsledku ale umožnila také obsah databanky určitého typu databáze vyjmout a převést do databáze jiného typu. Nejedná se přitom pouze o výměnu databáze používaného informačního systému poté co se původní databáze stala omezujícím faktorem pro další rozšiřování samotného informačního systému, ale jde např. i o situaci, kdy je nutno řešit vzájemné předávání dat mezi různými informačními systémy. Tento požadavek byl řadu let bolestivým problémem pro firmy, jejichž jednotlivé pobočky a provozovny jsou roztroušeny v různých částech státního útvaru nebo světa. Ani dnes není vždy možné vystavět informační systém centralizovaně s přístupem tenkých klientů odkudkoliv. Důvodem přitom nemusí být pouze problém rychlosti spojení a jeho cena, ale také např. bezpečnost přenosu dat (a jeho cena) atp. Dnešní distribuované informační systémy dokonce mohou používat ve svých různých částech různé typy databází různých výrobců. Vzájemná výměna dat mezi organizacemi různého typu, jejichž spolupráce byla otevřena v době,

kdy každá z nich má už vybudovaný a provozovaný informační systém, představuje také důležitý fenomén obecné potřeby vzájemné výměny dat. Ke komunikaci takového typu výrazně přispěla definice přenosového protokolu dat s označením XML.

XML (Extensible Markup Language) definovalo konsorcium W3C na základě vývoje a používání jazyka HTML a definice jazyka SGML. SGML (Standardized General Markup Language) byl navržen pro obecnou definici značkovacích jazyků, které se ukázaly důležité pro vzájemnou komunikaci při předávání dat serverem klientovi. Přílišná složitost SGML přinesla definici XML jako jednodušší a pro praxi flexibilnější variantu jak v rámci implementace používaných technologií u stránek WWW internetu, tak pro používání programátorem. Přestože lze XML interpretovat dnešními prohlížeči stránek WWW, jeho hlavní význam je komunikační. Ten je dán oddělením obsahu přenášených dat od definice jejich struktury. Je tím myšleno, že v okamžiku přenosu dat je protistraně zaslána struktura dat tak, aby je dokázala automatizovaně uložit např. do místní databáze. Zda uloží data v definované struktuře XML nebo provede jejich transformaci dle potřeb a struktury vlastní databáze, je věcí zpracování přijímaných dat a úschovy pro jejich další použití v rámci místního informačního systému. Programátor takové transformace přitom není odkázán na vlastní techniky určení obsahu příchozích dat, ale má dnes prakticky jako součást každé databáze prostředek, v němž zpracování přijímaných dat kóduje, *XML parser*.

XML

V XML pracujeme s tagy podobně jako u HTML. Podíváme-li se však do textového souboru se zdrojovým kódem (textovým editorem), párové značky nesou jména, která obvykle v HTML nepoužíváme. Jsou totiž definovány v pomocném textovém souboru s formátem DTD (Document Type Definition). Obsah tohoto souboru tedy skutečně určuje typ dat a způsob jejich manipulace a strukturálního zařazení, které jsou označovány v dokumentu XML. Obsahem souboru DTD je *deklarace entit a elementů*. *Entita* je pevná hodnota (konstanta) a *element* pak je definice párové značky. Součástí souboru typu DTD je pochopitelně také vyjádření gramatiky, která je definována nad entitami a elementy. Vyjádření gramatiky popisu definice dat uložených v XML není pro laika jednoduše pochopitelné. Programátorovi však při pohledu na způsob určování sekvencí, voleb, seskupení a opakování zaplesá srdce. Přestože se jedná o zúženou podobu, tak jak jsme popsali teorii gramatik a umělých jazyků v části Base Of I.2 jako základ programovacích technik, popis gramatiky definice práce s příchozími daty je zde určen makrojazykem a samotnou realizaci potřebných algoritmů gramatiky interpretuje XML parser.

DTD

Podstatou digitálních technologií jsou data. Jejich obsah je vždy unikátním bohatstvím jedince nebo organizace jedinců. Obsah dat má ovšem proměnlivou povahu, přesněji, vznik dalších dat pozměňuje vnímání dat jako celků, a to v různém kontextu. Oproti klasickým záznamovým způsobům jako je papír nebo hliněné destičky však mají digitální data výhodu opětovného použití bez extrémní námahy. Aby tomu tak skutečně mohlo být, způsob uchovávání dat musí být obecně srozumitelný, a to při

datové sklady

zachování rozmanitosti jejich fyzické struktury. Hovoříme o poskytování dat v obecně čitelných konvencích. HTML a později XML jsou takovou obecnou konvencí. Přestože mnohá data obecně zveřejnit nechceme a naopak se snažíme je utajit, potřebujeme i je čas od času transformovat např. do jiné aplikace nebo jiné databáze soukromého charakteru. Obecně tak skutečně vzniká informatická klenba lidské společnosti digitálního věku, o jejichž možnostech a významu lze usuzovat mnohé. Při provozu informačních systémů je často opomíjeným rozměrem čas, který zde naopak nabývá na významu. Informatická klenba se proměňuje v čas, přitom historii těchto proměn lze zpětně sledovat v digitálních archívech. A nejenom sledovat, ale i vyhodnocovat. O vyhodnocování dat z historie projevil zájem především obchod, protože jeho management se vždy snažil orientovat svůj další marketingový rozvoj podle již proběhlých úspěšných či neúspěšných aktivit. Proto má obor práce s historickými daty často trochu pejorativní označení business intelligence. Obecně se ale jedná o technologie, jejichž využitelnost je stejně široká jako digitální technologie vůbec. Jedná se o práci s datovými sklady (data warehouse), s vhledem (insight) do dat a těžbou dat (date mining). Datový sklad přitom musí mít určitou vnitřní strukturu a organizaci dat, aby poskytoval těžbu dat ve smyslu různých rozměrů (dimensions), kterým je např. čas. Vést řez datovým skladem podle určité dimenze tedy předpokládá nejenom data pro potřebné vyhodnocení shromáždit v databázi, ale shromáždit je podle potřebného datového modelu. Princip datových modelů pro datové sklady je používán v podobě např. kostky (cube), hvězdy (star) nebo vločky (snowflake). O jejich progresivitě se vedou diskuse na odborném fóru. Za klíčovou knihu této problematiky lze jistě považovat [Kimball1996]. Produktem těžby dat jsou výsledky, které jsou označovány jako studie (report). Jejich obsahem je pokud možno stručný a srozumitelný soupis výsledků požadované úlohy, jako je např. obchodní finanční obrat za posledních několik měsíců pro jednotlivé zákazníky, ale třeba také počet úspěšně léčených zhoubných nádorů různými léky za posledních několik měsíců, a to v různých regionech. Obtížná není v datových skladech pouze metoda technologie vzhledu do dat, ale také samotné shromažďování dat. Proces čerpání dat z provozovaných informačních systémů nebo z archívů dat nazýváme pumpování dat. Celý proces získávání dat a jejich transformace do datového skladu je označován termínem ETL (Extraction, Transformation and Loading). Pro potřeby určité organizace představuje stanovení způsobu čerpání dat proces, který se neopakuje. Samotné čerpání dat (a naplnění datového skladu) je ale proces, který je automatizovaně spouštěn vždy po uplynutí určitého časového úseku nebo po dosažení určitého stavu datové základny provozovaného informačního systému. Vyhodnocovací metody pro vytváření studií nad datovým skladem nesou označení OLAP (On-Line Analytical Processing), případně ROLAP (Relational OLAP). Přístup uživatele k možnostem vyhodnocování (tvorbě studií) je dnes obvykle prostřednictvím prohlížeče stránek internetu. Spustit navržený report (požádat o výsledky studie) je ale často možné i pomocí rozhraní mobilních telefonů atd.

OLAP

Datový sklad je mnohdy chápán jako součást informačního systému a uživatelsky tomu tak skutečně je. Informační systém jako celek je poměrně složitý software určený pro řízení chodu organizace lidí. Přestože se výrobci software snaží ujistit laiky o opaku, vždy se jedná o prakticky unikátní strukturu a použití informačních technologií, jimž musí předcházet kvalitní analýza chodu organizace. A nejenom to, s vývojem a změnami v organizaci je nutné také provádět úpravy a programovat nové části (moduly) používaného informačního systému. Informační systém není opakovanou aplikací, jako je tomu např. u tabulkových programů nebo textových editorů. Návrh, realizace a vedení provozu informačního systému je pro každou organizaci nákladné, a to nejenom finančně. Z těchto důvodů je nutné, aby každý používaný informační systém byl dobře popsán v uživatelské, implementační a provozní dokumentaci. Tým programátorů musí majiteli informačního systému poskytnout zdrojové texty s podrobným komentářem tak, aby bylo možné provádět změny a rozšiřování informačního systému nezávisle na výměně programátora nebo celého jejich týmu. Vlastnictví autorských práv ke kódování software informačního systému, případně návrhu jeho koncepce, dokumentace atp., je také nutné vyjasnit, a to jistě smluvně právní cestou.

III.4 – bezpečnost, viry, červi, ochrana, šifrování

Postupující digitalizace přinesla do mnoha činností člověka jeho závislost na dalším fenoménu, kterým je ochrana jeho dat. Zřejmé výhody digitální informace, jejího zpracování a šíření jsou v rozporu s nároky lidského ega, které touží po svém prosazení mezi ostatními, po jejich uznání a tedy získání lepšího prostoru pro svou existenci, silnějšího vlivu a postupné nadvlády. Podobně jako člověk přivykl činnosti směřující k ochraně své existence, existence svých bližních a ochraně majetku, vyžaduje dostupné analogické postupy, které by zajišťovaly také ochranu jeho dat související s požadavky jeho ega. Nikoho dnes nepřekvapí, požaduje-li okradený člověk soudní satisfakci, jedná-li se o hmotný majetek. Podobně jako se případ komplikuje u majetku nehmotného, např. vlastnictví popisu výrobního postupu či myšlenky obecně, komplikuje se podobný nárok u vzniku digitální kopie, která je nehmotná a jejím obsahem může být vlastně stále univerzálnější cokoli. U digitální kopie se navíc jedná o zcizení, o kterém vlastník ani nemusí vědět, protože původní zdroj zcizení zůstává neporušen. Dvěře ke zcizení digitální informace jsou dnes doširoka otevřeny a v mnoha případech se nejedná o snahu krást, ale pouze o nalezení cesty k takové krádeži a upozornění na tuto možnost. Toto upozornění může mít různé formy, z nichž nejhorší je škodolibá destrukce dat. Jedná se o určitý nový druh sportu, který je označován termínem *hack*, který ale při své naivní podstatě může mnohdy způsobit vážnou paralýzu progresivních činností lidstva. Zájmu se těší především z důvodů intelektuálního vzrušení a obtížného postihu, který je způsoben nesnadným zachycením v právním řádu lidské společnosti. Právní řád však kulhá i z důvodů chatrné vymahatelnosti právního nároku způsobené nedostatečnou technickou podporou samotných digitálních technologií, jak si přečteme v dalším textu.

Lidská společnost jako celek se dostala do závislosti na digitálních technologiích, protože je začala používat v provozech, které jsou pro ni existenčně důležité. Nejedná se pouze o finanční toky, ale také např. o řízení energetických a dopravně komunikačních sítí či přístupu k masmédiím, pokud odhlédneme od důležitosti soukromí dat v pojetí lidských práv. Bezpečnostní rizika si při vzniku principů digitalizace zjevně nikdo neuvědomoval. První, kdo na tento problém upozornil, byl v roce 1984 spisovatel science fiction William Gibson, který ve svém románu *Neuromancer* (viz [Gibson1984]) představil digitální virtuální svět lidské společnosti v Matrixu, období dnešního Internetu. Přestože se v té době ve skutečném světě Internet začínal teprve postupně implementovat, používané principy se již nedaly změnit, a to i v případech, že by Gibsona odborníci brali vážně a zároveň věděli o principech jiných. Na současném stavu hrozícího nebezpečí nechráněného toku digitálních informací nese jistě vinu také obchod, který v dalších letech s cílem prodat cokoli bez hlubší kvalifikace zaplavil svět šlendriánem. Zcela paradoxně jsou pak vždy k odpovědnosti volávání odborníci Computer Science. Ti, pokud je jim to vůbec umožněno, se usilovně snaží o nápravu, a to navzdory stále lehkomyšlnému obchodu, který v reklamních akcích bezpečnostní rizika

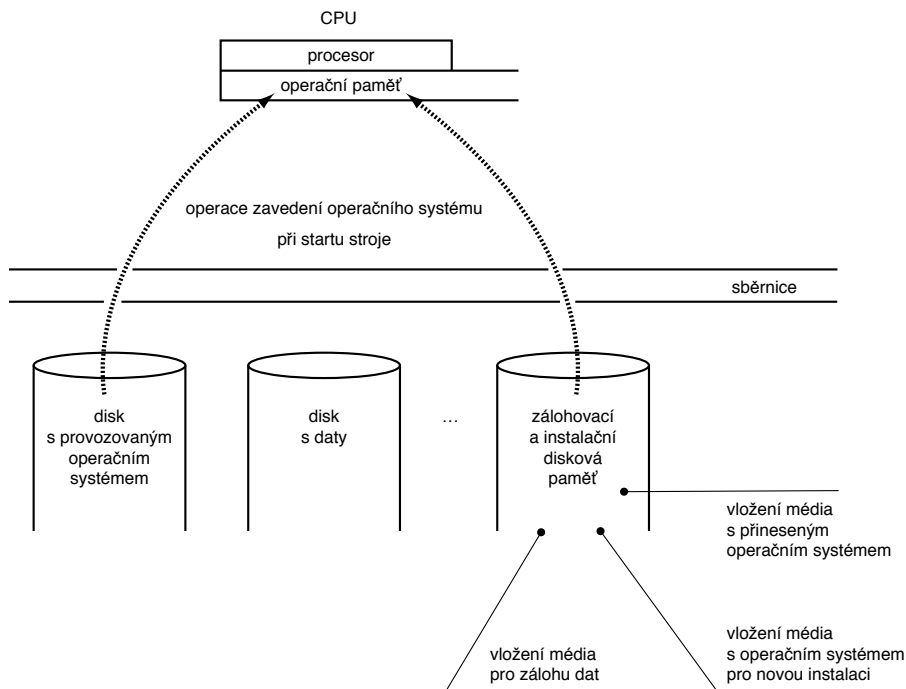
zamlčuje nebo je dokonce popírá. Řešení je z odborného hlediska obtížné, protože je nutné zachovat uživatelům současný stav a ten vybavit nadstavbou kvalitního zabezpečení tak, aby byla zachována kompatibilita technologií.

digitální kopie

Prvním principiálním nedostatkem ochrany digitální informace je možnost vytvoření její dokonalé kopie. Navíc výraz kopie bychom vlastně neměli používat, vzniká totiž druhá instance téhož. Na této možnosti je vlastně práce s digitální informací principiálně závislá. Víme, že pokud nezajistíme provedení kopie jádra operačního systému z externí diskové paměti do operační paměti při startu výpočetní jednotky, počítač nemůže vůbec pracovat. Kopii dat, uložených na výměnném médiu, kterou přenášíme k jinému počítači, musíme mít ve všech bitech věrnou originálu, pokud s ní máme pracovat stejným způsobem na jiném počítači. Konečně informace z Internetu zobrazované na našem displeji ve tvaru slov, obrazů a zvuků musí být přenesenou naprosto věrnou kopií těch, které autor nabízí ke zveřejnění ve vzdáleném serveru. Zabránit jejímu opětovnému využití lze ovšem pouze začátečníkům a naprostým laikům. Zabránit uložení digitální informace přicházející jakýmkoliv kanálem k našemu displeji je totiž principiálně nemožné. Ochrany média (jako je např. CD nebo DVD) před provedením kopie jeho obsahu se dosahuje různým způsobem, vždy lze ale používanou zábranu prozkoumat (je digitální) a buďto odstranit a nebo kopírovat včetně jí samotné. Zdá se a zcela právem, že řešení není v technických možnostech, a bude tedy muset být v právním řádu lidské společnosti, případně v rentabilitě prováděných kopií. Např. jistě nebude potřeba provádět kopie komerčních filmů a hudby, pokud za jedno přehrání prostřednictvím vysokorychlostního internetu zaplatím částku odpovídající ceně denního tisku, zvláště když bude archiv jakýchkoliv filmů a nahrávek v Internetu k dispozici nepřetržitě a uvědomím-li si, že skutečně oblíbené filmy a nahrávky si přehraji pouze několikrát za celý svůj život. Naopak, jistě uvítám vyklizení prostoru, který zabírají nosiče v mém bytě, na kterých je koneckonců pouhý zlomek toho, co by mohlo být v síťových archívech k dispozici.

Neustálé provádění digitálních kopií z externí do operační paměti a zpět při vedení provozu každého počítače podle Von Neumannova schématu je princip umožňující snadné zcizení dat. Fyzicky dostupná výpočetní jednotka je obvykle vybavena periferiemi externích diskových pamětí nejenom pro práci operačního systému, ale i diskovými periferiemi, prostřednictvím nichž se provádí záloha dat, instalace nových programů nebo nová instalace operačního systému. Jedná se dnes obvykle o mechaniky CD nebo DVD, obecně ale jde o externí diskovou paměť, jejíž pomocí lze do výpočetního systému vložit digitální informaci nebo ji z něj odnést. Přestože taková mechanika DVD může být vždy po provedení plánované zálohy nebo instalace fyzicky odpojena a odnesena, fyzická dostupnost počítače umožňuje specialistovi přinést si mechaniku svoji, a to včetně potřebného rozhraní pro připojení ke sběrnici. Pokud narušitel dále dokáže prohlásit tuto mechaniku za externí paměť (přepínačem hardware nebo pomocí základního software stroje), z níž má centrální jednotka zavést do operační paměti operační systém, po spuštění stroje bude pracovat počítač

s přineseným operačním systémem a ostatní diskové periferie bude mít dostupné zcela pro své potřeby. Situaci ukazuje kresba III-15.



Kresba III-15: Operační systém na přineseném externím médiu.

Přístup k datům je v uvedeném případě založen na získání výhradního přístupu k řízení stroje, na jehož diskových periferiích jsou zcizovaná data uložena. Získání výhradního přístupu k takovým datům nemusí být pouze cestou zavedení přineseného operačního systému téhož typu. Operační systémy struktury systémů souborů jiných operačních systémů dnes obvykle také rozeznávají a dokážou číst. Dokonce lze získat výhradní přístup k hardware i jednoduchým programem přístupu k periferiím, který lze použít namísto operačního systému, a odnést pak celý obsah disku v kopii a později jej zkoumat. Snaha řešit přístup k hardware cestou požadavku zadání hesla operátorem při startu stroje riziko omezuje. Vždy ale musí být pro případ zapomenutí hesla k dispozici také cesta jak jej zrušit, nehledě k tomu, že je potom nutno zajistit trvalou přítomnost operátora pro případ havárie a nutnosti opětovného startu stroje. Jednoznačně lze říct, že zamezit nežádoucí kopii dat lze pouze zamezením fyzického přístupu k počítači. U významných počítačových celků musí být bezpečnost zajištěna provozem kvalitního operačního systému na promyšleném hardware v kombinaci s pečlivou organizací fyzického přístupu osob k hardware.

Prozatím jsme ovšem uvedli pouze případ místního výpočetního celku. Situace se bezpečnostně velmi, velmi komplikuje jeho připojením k počítačové síti. Obejít se bez spojování počítačů do sítí je dnes nemožné. Dnes

i izolovaný výpočetní systém řídicí výrobní průmyslový proces je sestaven z několika výpočetních celků, které jsou vzájemně propojeny sítí. Produkce takového výrobního procesu přitom podléhá obchodním aktivitám firmy, do výpočetního systému tedy musí nepřetržitě vstupovat zadání pro výrobu a současně je nutno z něj odebírat informace o proběhlé nebo probíhající výrobě. Obchodníci nemohou být digitálně izolováni od svých zákazníků a ti mnohdy požadují informace ohledně stavu své zakázky ve výrobě. Jiným příkladem na opačném konci spektra je obyčejný osobní počítač izolovaný v dobře fyzicky zabezpečené pracovní nebo domácím bytě. Člověk si takový počítač obvykle pořizuje nikoliv pouze pro potřeby práce s domácími agendami nebo jako psací stroj, ale dnes také a především pro potřeby komunikační, jako je email, vstup do Internetu a přenos dat.

Bezpečnostní opatření pro práci počítačů v síti vychází z následujícího základního doporučení. Je nutné určit počítače, které musí být mezi sebou spojeny sítí a stanovit jejich umístění a cesty fyzického propojení (fyzickou vrstvu). Místních sítí může být stanoveno více, pro jejich vzájemné propojení je ale nutno definovat vždy pouze jednu fyzickou cestu. Nad síťovou vrstvou každého fyzického spoje mezi sítěmi navzájem pak povolit provoz pouze síťovým aplikacím (v aplikační vrstvě), které jsou pro práci uživatelů nezbytné. Vlastně totéž doporučení se také vztahuje na připojení takto vzniklé sítě místních sítí ke zbytku světa, tedy k Internetu. Místo propojení s Internetem musí být stanoveno pouze jedno a musí podléhat nepřetržité kontrole průchodů uživatelů, a to zejména ve smyslu odezvy vzdálených sítí a uzlů. Situace se komplikuje, používáme-li k propojení svých místních sítí navzájem právě Internet, tedy v případě geograficky vzdálených lokalit téže organizace, protože získat bezpečné privátní spojení napříč planetou je drahé.

zadní vrátka

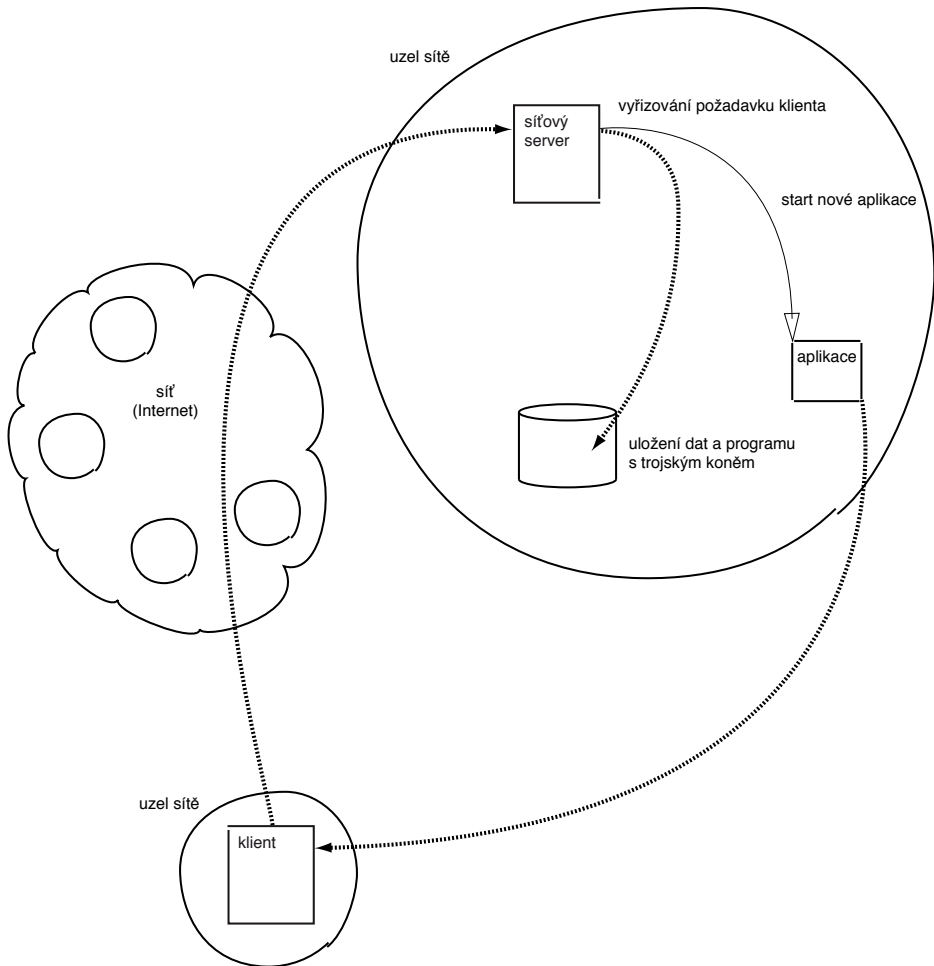
Průnik narušitele při využití síťového provozu je založen především na používané architektuře klient – server. V části Connect To II.2 jsme tuto architekturu podrobně uvedli v technologii TCP/IP, a to včetně vyjmenování všech zúčastněných vrstev síťového provozu. S potřebným odstupem si lze dnešní síťový provoz v drtivé většině případů představit jako systém běžících programů (serverů) v jednotlivých výpočetních celcích. Server Apache nabízí zobrazování stránek v určitém výpočetním celku, server POP nebo IMAP nabízí stažení pošty. Každý server očekává zájem klientů, každý klient může požádat o spojení principiálně kterýkoliv server v Internetu na světě. Dnešními prostředky neřešitelné nebezpečí zneužití tohoto principu je především v nerozhodnutelnosti algoritmů, jak jsme uvedli v části Base Of I.2. Server jako běžící program provádí algoritmus, o kterém nelze s matematickou jednoznačností rozhodnout, zda je bezchybný. Server komunikuje s klientem na základě domluveného aplikačního protokolu. Zpracovávat komunikaci s klientem pro server tedy znamená provádět požadované činnosti pro potřeby klienta. Narušitel se chová jako běžný klient, pokouší se ale v rámci komunikace vyvolat situaci, ve které je server uveden do neurčeného stavu. Takový stav je např. paralýza jeho dynamicky se měnící datové části běhu programu (zásobník, stack), na jejímž základě klient dokáže nastartovat jiný

program operačního systému, obvykle určený pro správu. Takovou chybu v serveru nazýváme dírou do operačního systému. Díra ovšem v některých případech v historii byla pouze neodpojenou částí komunikačního protokolu, kterou používal programátor při psaní a ladění samotné síťové aplikace. Díra zanechaná programátorem je nazývána zadní vrátka (back door) a je hackery vyhledávána především. Pomocí díry v serveru dokáže narušitel získat přístup ke vzdálenému počítači, který nemusí ani být privilegovaný, aby mu posloužil k získání kopií dat nebo k využití k dalšímu průniku do systémových služeb téhož nebo jiných uzlů sítě.

Další kategorie chyb, kterých narušitelé využívají, je dána nekvalifikovanou správou síťových serverů. Servery mohou data poskytovat, ale i přijímat a pokud přijmou a uloží nepovšimnutá data, jejichž součástí je program, spuštění takového programu některým z uživatelů operačního systému může způsobit paralýzu operačního systému nebo „pouze“ získání dalších možností pro narušitele. Dost často se tak děje zcela vědomě za naivní účasti uživatele, který hledá v Internetu zdarma jemu potřebnou aplikaci. Uživatelé často využívají produkty software z neznámých serverů Internetu. Součástí takto distribuované aplikace přitom ale může být i činnost, která později umožní průnik do operačního systému. Způsob narušení prostřednictvím programu určeného k jiné činnosti nazýváme průnik pomocí trojského koně (trojan horse).

trojský kůň

Kvalifikovaně spravovat síťový uzel a používat nabídku do sítě prostřednictvím serverů však není snadné. Stěží to může provádět uživatel bez odborné kvalifikace, který sice dokázal propojit několik osobních počítačů své kanceláře mezi sebou a k Internetu, ale o technologii síťových serverů nic neví. Navíc nemá ani žádné prostředky k tomu, aby získal potřebný přehled nad síťovými aplikacemi, které uzly jeho sítě nabízejí do Internetu po prvotní instalaci. Sám považuje za úspěch, podaří-li se mu síť vůbec zprovoznit, což je dnes možné za cenu právě nebezpečného otevření se do celého světa. Princip průniku prostřednictvím síťového serveru ukazuje kresba III-16.

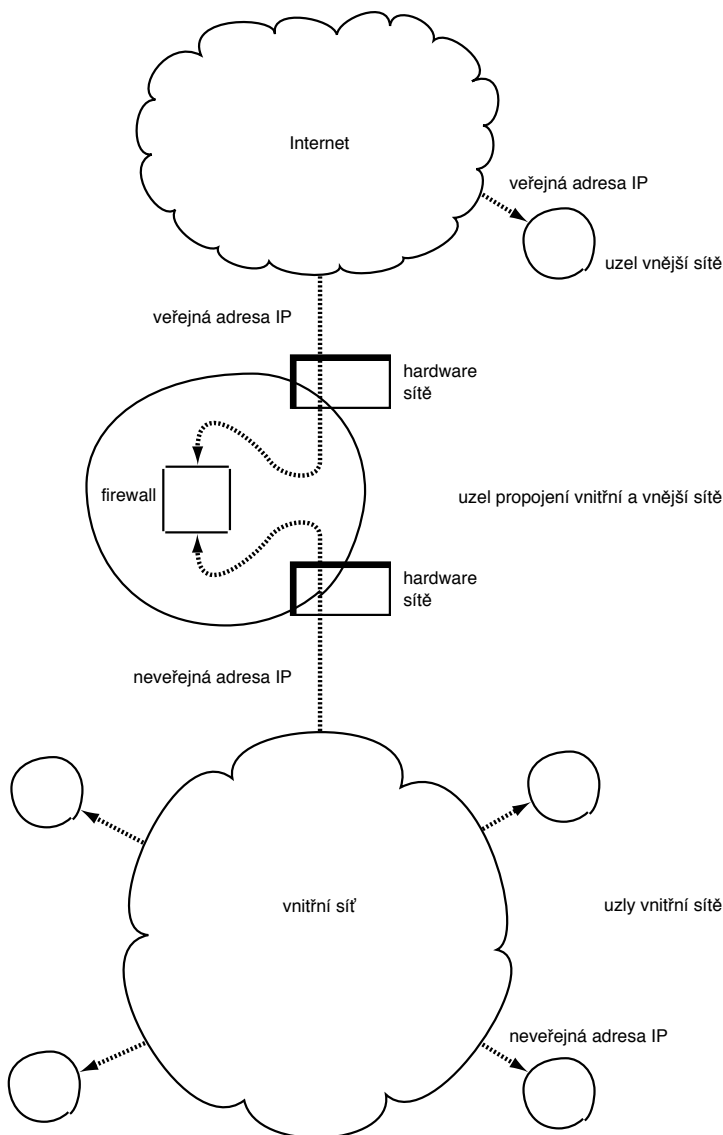


Kresba III-16: Průnik prostřednictvím síťového serveru.

firewall, proxy

Pro zajištění řízené bezpečnosti práce uživatelů v místních sítích oddělených od sítě veřejné (Internetu) jsou dnes používány ochranné podsystemy s označením *firewall* (ochranná zeď) nebo *proxy*. Jedná se o software, který zajišťuje právě požadovaný přehled nad průchodem uživatelů místní sítě do sítě veřejné. Výchozí stav je přitom takový, že každý průchod je zakázán a teprve definicí *pravidel průchodů* stanovujeme, které síťové aplikace mohou mezi sebou komunikovat a v jakém rozsahu. Výchozí zablokování firewall nebo proxy dokáže zajistit, přestože v uzlech místní sítě jsou aktivní servery pro nabízení služeb příchozím klientům odkudkoliv. V takovém případě je pak možné provozovat síťovou službu (jako je např. sdílení disků) v místní síti několika počítačů mezi sebou, ale nikoliv mimo tuto síť. Klienta z vnější sítě totiž firewall (či proxy) nepustí do vnitřní sítě. Takový klient se ani nedozví, zda je za ochrannou zdí server, který by dokázal odpovídat na požadovaném protokolu (portu, že ano). Firewall je software, který odděluje

vnitřní síť od zbytku světa na úrovni síťové nebo aplikační vrstvy. Na úrovni síťové vrstvy v internetu je to tedy na úrovni protokolu IP. Princip ukazuje kresba III-17.



Kresba III-17: Firewall – ochranná zeď vnitřní sítě na úrovni síťové vrstvy protokolu IP.

Základem ochrany vnitřní sítě je instalace potřebného software na uzel připojující vnitřní síť k Internetu. Takový uzel je směrovač (router), který má instalovány dvě síťové periferie (viz část Connect To II.1), jednu z nich používá pro připojení k uzlům vnitřní sítě, druhou k Internetu. Bez ochranné zdi jsou uzly vnitřní sítě viditelné z vnější pod identifikací odpovídající adresou IP v rámci určené síťové adresy. Firewall takovou viditelnost přetne. Síťová

vrstva uzlů Internetu končí adresou IP, již je vybaveno rozhraní hardware sítě směrovače obráceného do Internetu. Druhé rozhraní hardware sítě je vybaveno adresou IP, která je součástí síťové adresy uzlů vnitřní sítě. Adresa IP vnitřní sítě je zcela libovolná. Jedná se vlastně o samostatný internet v rámci celé škály všech typů adresace IP (viz Connect To II.1 a Příloha B). Propojení obou internetů pak zajišťuje firewall. Říkáme, že v případě komunikace mezi klientem a serverem firewall překládá výchozí a cílovou adresu IP. Uzel vnější sítě adresuje svůj požadavek na veřejnou adresu IP směrovače a firewall rozhodne, zda požadavek do vnitřní sítě vpustí a kterému uzlu v síti jej předá. Tento řez je omezující současně i pro uzly vnitřní sítě. I jejich požadavky firewall zkoumá a podle daných pravidel je buďto překládá do vnější sítě a nebo je zahazuje. Termín odmítnutí požadavku zde není na místě, protože firewall v rámci bezpečnosti neoznamuje odmítnutí požadavku, ale vůbec se v takovém případě klientu neozývá, mlčí.

Uvedený firewall kontroluje spojení na úrovni síťové vrstvy. Každý procházející požadavek na úrovni transportní, tj. na úrovni určitého portu a tedy typu síťové služby, musí mít průchod povolen pravidlem, které firewall prověří a teprve poté zajistí spojení na síťové vrstvě. Z uvedeného je patrné, že firewall není obecně lékem na průnik, protože pokud síťový server vnitřní sítě pracující na oznámeném portu pro vnější síť bude děravý, škůdce se do vnitřní sítě dostane stejným způsobem jako doposud. Firewall ale umožňuje explicitně vyjmenovat pouze služby, které si přejeme obecně zveřejňovat a žádné jiné. Kvalifikovaný správce sítě proto omezuje služby vnitřní sítě pouze na její rozsah a např. stránky WWW organizace zobrazuje na uzlu vnější sítě jiné veřejné adresy IP než je průchozí do vnitřní sítě (viz kresba). Obdobný veřejný uzel bývá také určen pro příjem pošty pracovníků organizace a pro její čtení pomocí protokolu POP3 nebo IMAP je na tento uzel vyhrazen pouze přístup uzlů vnitřní sítě v kontextu čtoucích klientů. Obvykle totiž lze síťové služby dnes omezovat na připojování klientů pouze z vyjmenovaných adres IP.

Software proxy server je často také označován jako firewall na úrovni aplikační vrstvy. Znamená to, že se jedná o hlídacího psa, který má ve své vnitřní konfiguraci nastaveny síťové služby, jejichž požadavky do vnitřní sítě vpustí (nebo z ní vypustí) nebo jim vstup (výstup) zakáže, a to bez ohledu na to, jak pracují na nižších síťových vrstvách. Proxy server disponuje i dalšími funkcemi, jako je např. vyrovnávací paměť atp. V terminologii je mnohdy firewall zaměňován za proxy a opačně. Typickým představitelem proxy jsou aplikace kontroly síťového provozu operačních systémů PC, a to především od firmy Microsoft (MS Windows NT, XP atp.). Většina dnešních PC s těmito bezpečnostně děravými operačními systémy totiž bývá často připojena k veřejné síti typu internet jako jeden uzel prostřednictvím vytáčené telefonní linky nebo jiného pevného spoje, kdy poskytovatel negarantuje žádnou ochranu. Při dynamicky přidělené adrese IP je takové PC otevřeno celému světu jako neviňátko. Firewall na aplikační vrstvě blokuje provoz nevyžádaných síťových aplikací a přinejmenším na vzniklé nebezpečí aktivace síťové aplikace upozorňuje uživatele. Obyčejný uživatel však musí

mít bohužel přehled jak o principech síťového provozu, tak o seznamu síťových aplikací, které sám používá, což je pro uživatele netechnicky zaměřené vědomostní kapitál náročný a obtížně dostupný.

Běžný způsob ochrany přístupu k požadované síťové službě (a tedy vstupu do počítače) byl odedávna založen na prokazování se jménem a heslem, jak jsme si ukázali v části Connect To II.3. I server stránek WWW může omezovat přístup k některým částem zveřejňovaného stromu na čtenáře, kteří znají přístupový kód. Vypadá to pěkně do okamžiku, než si uvědomíme, že přístupový kód musí tak jako všechna data mezi klientem a serverem projít sítí. Data procházející veřejnou částí sítě lze kopírovat, jinak řečeno komunikaci mezi klientem a serverem lze monitorovat. Pokud rozumím principům síťového přenosu dat a získám software pro takové monitorování (a že jich v Internetu volně ke stažení je!), dokážu číst nejenom přenášená data, ale i přenášené přístupové kódy. Touto cestou pak dokáže hacker získat přístup (dvojici jméno a heslo), který je registrován jako legální. Na rozdíl od serverů síťových služeb, které velmi často pracují (při chybné správě uzlu) v režimu vysoké priority přístupu k operačnímu systému, zde hacker obvykle získává přístup některého z neprivilegovaných uživatelů (pokud správce privilegovaný vzdálený přístup k uzlu rozumně nikdy nepoužije). I to mu ale umožňuje se v uzlu pohybovat a hledat další díry operačního systému, a to na odpovědnost odposlechnutého uživatele.

Řešení, které se dnes pro zamezení odposlechnutí dat přenášených sítí používá nejvíce, je šifrování.

Šifrování (encryption) neboli kryptografie je část vědní disciplíny s označením kryptologie stojící na rozhraní matematiky a Computer Science. Její prestiž byla velká v dobách, kdy při snaze dešifrovat text neznámé šifry nebyly k dispozici počítače a šifrování se provádělo ručně. Způsoby šifrování jsou různé a různě komplikované. Principiálně se ale vždy jedná o určitý algoritmus proměny původního textu (tedy dat) tak, aby při běžném čtení (otevření aplikací) nebyl srozumitelný, ani jej nebylo možné rozšifrovat do původní podoby bez znalosti klíče. Pro použití šifrovaných dat musí čtenář, jemuž jsou data určena, rovněž znát způsob, jakým lze data převést na původní. Triviální případ je ten, kdy zná šifrovací algoritmus a autor šifry mu sdělí klíč, kterým původní text šifroval. Potom rozšifruje text opačným postupem algoritmu. Takový způsob šifrování označujeme jako symetrický. Asymetrický způsob šifrování je založen na principu šifrování určitým algoritmem při použití soukromého klíče (private key) a rozšifrování algoritmem jiným při použití veřejného klíče (public key). Současná kryptografie je založena na obecné prezentaci algoritmů, kdy i při jejich znalosti je třetí strana bezmocná bez šifrovacích klíčů. Je ovšem vždy pouze otázkou výpočetního výkonu, jehož prostřednictvím se podaří třetí straně hledaný klíč uhodnout, nebo dokonce najít způsob (algoritmus) pomocí něhož lze klíč nad šifrou zjistit. Nalezení algoritmu k určení klíče k rozluštění dat za použití určitého způsobu šifrování je označováno termínem zlomení šifry (codebreaking) a v současné době je to oblíbený sport studentů informatiky a hackerů. Zlé jazyky praví, že jeden

šifrování

symetrické
a asymetrické
šifrování

zlomení šifry

z důvodů, proč se jim to daří vcelku vždy v rekordním čase od zveřejnění samotné šifry, je ten, že kryptografie byla historicky podporována především zájmem armády (která úzce souvisí s mocí), a ta je spokojena s výsledkem, který jí zaručí, že data nebudou rozluštna pouze po určitý omezený časový interval. V rychle se proměňující válečné situaci je totiž požadovaná doba utajení přenášené informace mnohdy jen několik desítek vteřin, a tu současné metody kryptografie splňují. V takovém případě je použita šifra na data, která lze zpětně získat třetí cestou v průběhu několika desítek i stovek minut, považována za bezpečnou.

S obecným vědomím nebezpečí, které přináší nešifrovaný tok dat Internetem, nabyla ovšem podpora kryptografie na významu a stala se tak osobním zájmem každého uživatele, ale i zájmem obchodu s takovým zbožím. Zjevně především proto se dnes kryptografie snaží přinášet nové výsledky prakticky denně, a to tak, aby bylo posíleno utajení přenosu dat jakoukoliv cestou, tedy nejenom sítí, ale i na přenosných médiích.

U používaného symetrického způsobu šifrování se jedná se o rychlé algoritmy, jejichž klíče se ale dají poměrně snadno třetí stranou uhodnout. K tomu, aby zjištění klíče *hrubou silou* (postupným generováním a zkoušením klíčů) bylo znesnadněno, je nutné, aby klíče měly dostatečnou délku. Délka klíče bývá v rozmezí 40–128 bajtů, ale mnohdy i více. Delší klíč ovšem samotný šifrovací algoritmus zpomaluje při běžném provozu. Typickým představitelem symetrického šifrování je DES (Data Encryption Standard) vyvinutý firmou IBM v 70. letech 20. století. Dnes je součástí amerického federálního standardu a i přesto, že je poměrně slabý, je stále hodně používán, a to ve své zesílené variantě 3DES (Triple-DES). Z dalších známých a používaných symetrických šifer je nutné jistě uvést šifry instituce RSA Data Systems, které mají označení RC2 a RC4, případně RC5. Jejich zajímavostí je používání *solí* (salt) při samotném šifrování jako doplňku ke klíči a délka klíče je proměnná (dokonce až do 256 bajtů). Algoritmy RC2 a RC4 jsou rychlé a poměrně bezpečné, a proto je používá síťový přenos dat, protokol SSL, jak uvidíme v dalším textu.

digitální podpis

Zajímavější způsob asymetrického šifrování bývá používán nejenom pro přenos dat, ale i pro *digitální podepisování*, protože asymetrie zde znamená certifikaci (potvrzení identity) jedné ze stran. Jak již bylo řečeno, principem je dvojice matematicky vzájemně souvisejících klíčů. Soukromý klíč vlastní v tajnosti tímto klíčem určená strana, která jej používá pro šifrování příchozích i odchozích dat. Soukromý klíč je nutné udržovat v naprosté tajnosti, jeho vyzrazení třetí straně znamená ohrožení nejen obsahu přenášených dat, ale i důkazu o jednoznačnosti majitele klíče. Veřejný klíč je poskytován majitelem soukromého klíče vlastně komukoliv, kdo projeví zájem se subjektem autorizovaným soukromým klíčem šifrovaně komunikovat. Komunikace probíhá tak, že majitel soukromého klíče používá k šifrování i rozšifrování soukromý klíč, ostatní komunikující protistrany používají pouze klíč veřejný.

RSA

Hlavním představitelem asymetrického šifrování je *šifrovací systém RSA* (Rivest-Shamir-Adleman, trojice příjmení autorů) a byl vyvinut v 80.

letech minulého století na MIT. Používá se k šifrování, ale také jako nástroj k digitálnímu podepisování. Protože ve svém algoritmu používá umocňování, je poměrně pomalý, ale jeho bezpečnost je velmi vysoká, zvláště používali poměrně dlouhý klíč (1024 bitů, tj. 128 bajtů, že). Pro vlastní šifrování dat se proto používá málo, jeho hlavní využití je např. pro výměnu privátních klíčů v síťové komunikaci atp.

Druhý významný algoritmus asymetrického šifrování je DSA (Digital Signature Algorithm). Byl vyvinut pro potřeby digitálního podpisu vládou USA, je možné jej ale používat pouze pro něj. Vzhledem k tomu, že byl v Americe také standardizován, nese také označení DSS (Digital Signature Standard).

DSA

MD2, MD4 a MD5 jsou asymetrické šifrovací algoritmy vyvinuté institucí RSA Data Security na začátku 90. let minulého století, a to rovněž především za účelem použití v digitálním podepisování. Jak naznačuje pořadí jejich číselného značení, jedná se o postupně bezpečnější šifry.

MD2, MD4, MD5

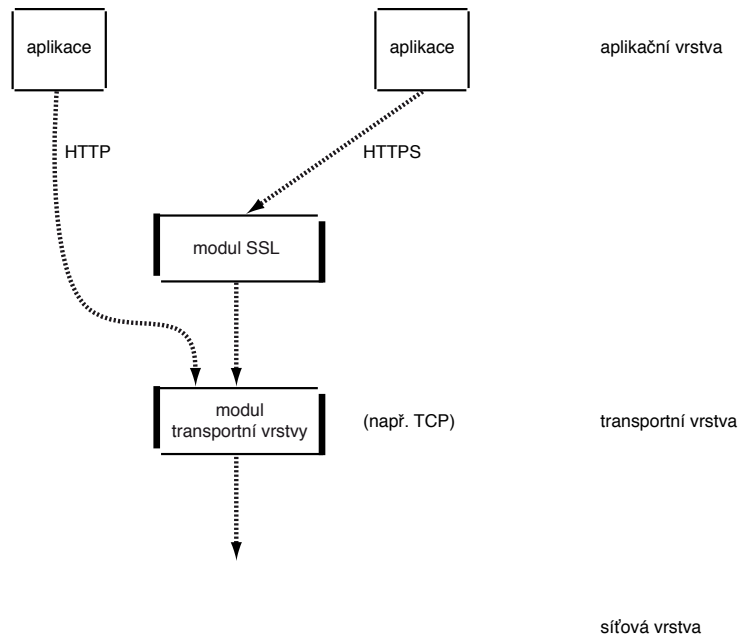
Konečně je důležitá také asymetrická šifra SHA-1. Je to algoritmus organizace National Institute of Standards and Technology z poloviny 90. let 20. století. Oproti MD5 je pomalejší, jeho bezpečnost je ale vyšší.

Asymetrické šifrování při přenosu dat (např. v protokolu SSL, viz dále) je používáno tak, že nejenom jedna strana je jednoznačně identifikována, ale oba účastníci šifrovaného provozu disponují soukromým klíčem, kterým šifrují a dešifrují odesílaná a přijímaná data. K tomu, aby celá komunikace měla smysl, je potřeba si vzájemně vyměnit klíče, a pro zvýšení bezpečnosti utajeně, tedy šifrovaně. Za takovým účelem se používají opět speciální algoritmy, jejichž představitelem je algoritmus Diffie-Hellman.

Digitální podepisování znamená, že digitální data jsou jednoznačně označena určitým subjektem tak, že digitální doplněk dat je s tímto subjektem jednoznačně spojen, nelze jej falšovat a ob stojí v systému práva lidské společnosti. Asymetrické šifrování umožňuje digitální data šifrovat soukromým klíčem tak, že je lze zpětně získat se znalostí veřejného klíče a toto zpětné získání dat potvrzuje autora zašifrovaných dat. Pro digitální podepisování se používá také princip *kryptografického kontrolního součtu*, který je principem uvedeného šifrování MD4 a MD5. Je to *šifrování jednocestnou hešovací matematickou funkcí*. Výsledkem práce této funkce nad podepisovaným textem je sekvence bajtů, která je k původnímu textu (datům) připojena. S šifrovanými daty je tato generovaná sekvence bajtů identifikována soukromým klíčem. Případná změna původních dat, ze kterých je takový kontrolní součet vytvořen znamená, že podpis neodpovídá (je vygenerován z původních, tedy jiných dat) a je tedy veřejným klíčem (a soudcem) odmítnut. Takový způsob digitálního podepisování vypadá elegantně, protože se velmi podobá podpisu lidskou rukou na papíře. Podepisovaný text totiž může zůstat v původní podobě a je pouze připojeným podpisem označen. Takto se dá certifikovat např. elektronická pošta. Samozřejmě nic nebrání tomu vše ještě šifrovat a tak si předávat digitální data, která jsou digitálně podepsána a určena pouze pro neveřejné potřeby.

Pokud má mít asymetrické šifrování a digitální podepisování v lidské společnosti právní platnost, předpokládá nutnost právního přidělování soukromých klíčů a s ním spojených dalších informací souhrnně nazývaných certifikátem. Vlastně se jedná o potvrzení veřejného klíče právním systémem, certifikační autoritou. Systém přidělování certifikátů certifikační autoritou je v USA a i jiných zemích ve světě již řadu let používán. Když certifikační autorita vydá certifikát, znamená to, že podepíše veřejný klíč a k němu připojené další údaje svým soukromým klíčem. Další údaje jsou např. jméno uživatele certifikátu, jeho internetová doména, sériové číslo certifikátu, ale např. i doba platnosti certifikátu.

SSL Implementace použití šifry pro potřeby bezpečného přenosu dat počítačovou sítí byla provedena definicí bezpečnostního protokolu s označením SSL (Secure Socket Layer) firmou Netscape. Jedná se o vloženou vrstvu síťové komunikace na úroveň mezi vrstvou aplikační a transportní, jak ukazuje kresba III-18.



Kresba III-18: Umístění SSL v hierarchii síťových vrstev.

Modul realizující SSL je v operačním systému využíván síťovými aplikacemi na vyžádání. Jak ukazuje kresba, nejedná se o povinnou vrstvu síťového spojení. Toto vyžádání je v praxi dáno přímo typem aplikace, která si modul zařadí do cesty dat tekoucích sítí. Vzhledem ke stáří klasických aplikací bez zabezpečení, jako je např. FTP nebo Telnet, používá uživatel ekvivalentní aplikace s novým jménem, např. FTPS nebo SSH. Ovládání je přitom obvykle takřka shodné s ovládáním v původní aplikaci. Kresba uvádí aplikaci s označením HTTPS, což je přesnější název zabezpečeného aplikačního protokolu komunikace klientu a serveru přenosu stránek WWW. Důležité je

přítom splnění podmínky, kdy zapnutí vrstvy SSL umí jednak klient a jednak server. Nedorozumění jsou přitom vyloučena stanovením komunikace na jiných portech transportní vrstvy než je tomu u původních nezabezpečených aplikací.

SSL je bezpečnostní protokol, který aplikacím umožňuje používat různé způsoby asymetrického šifrování, např. libovolnou z šifer uvedených v předchozím textu. Po navázání spojení dochází mezi serverem a klientem ke stanovení způsobu šifry a certifikačnímu procesu, kdy je potvrzena identifikace serveru. Pomocí veřejných klíčů si procesy připraví klíče vzájemného spojení a generují své soukromé klíče. Teprve poté je připraven šifrovaný přenos vlastních dat, a to včetně zadávání jména a hesla uživatele registrovaného v operačním systému serveru.

SSL je dnes jedním z nejpoužívanějších způsobů zabezpečení toku dat sítí. Principem je asymetrické šifrování s dynamickou generací všech klíčů, každé spojení má proto jedinečné klíče a pro hackera je proto obtížné spojení monitorovat. Jedinou možností je pro něj proces komunikace zcela okopírovat a později se pokusit zlomit šifru a najít dynamicky přidělené klíče, což je činnost poměrně náročná, nicméně možná, přestože výsledky jsou nejisté. Okamžitý odposlech je ovšem tímto způsobem znemožněn a také je znemožněn průnik do systému bez prověření certifikační autoritou.

V uplynulé historii vývoje počítačových sítí a jejich provozu byly a dodnes jsou používány bezpečnostní podsystémy, které se snaží zajistit prověření pravosti klienta k nabízenému serveru silněji než jen požadovanou dvojicí jména a hesla, která je přenášena čitelně sítí. Jedná se např. o systém Kerberos instituce MIT z 80. let 20. století. Tento bezpečnostní podsystém pracuje na základě prověřování vyhrazeným uzlem sítě a na základě přidělování *lístků* (tickets), jejichž doba platnosti je velmi omezená. Kerberos je podsystém, pro jehož používání je nutná zvláštní správa a používá se ve snaze zabezpečit provoz především vnitřní sítě organizace, přestože jeho nasazení může být poměrně rozsáhlé. Uživatel je při vstupu do Kerberem zajištěné sítě požádán o jméno a heslo a při dalším pohybu zabezpečenou sítí se již s touto dvojicí nepracuje, nýbrž je uživatel prověřen vnitřní databází speciálního uzlu sítě a přidělovanými bezpečnostními lístky.

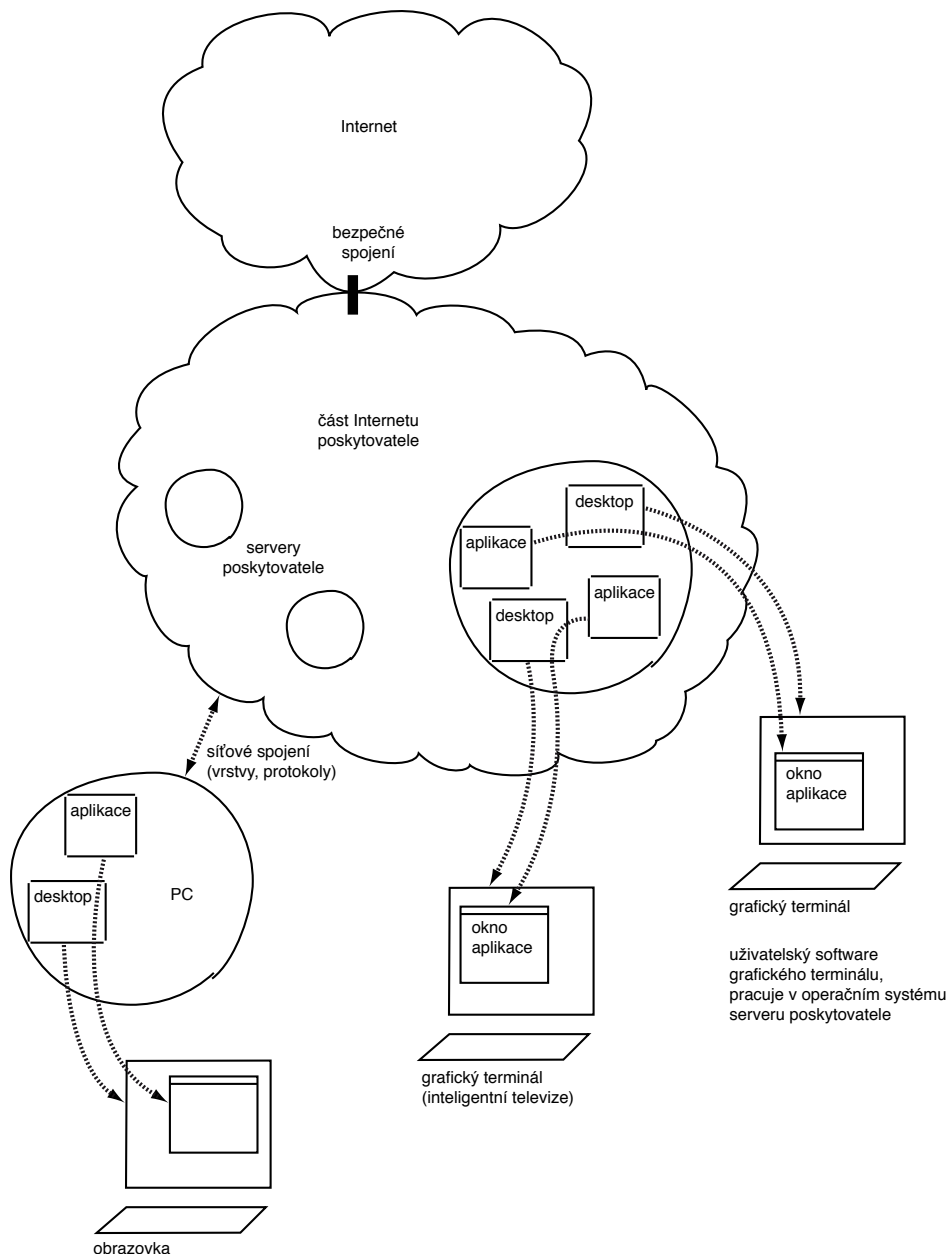
Kerberos

Na prověřování pravosti klientů při zvýšené bezpečnosti lze také koncepčně pamatovat již při programování síťové aplikace. Programovací jazyk síťových aplikací RPC (Remote Procedure Calls) ve své variantě Secure RPC (přibližně téhož stáří jako Kerberos) firmy Sun Microsystems tuto možnost nabízí. Vnitřně Secure RPC pracuje se standardní databází uživatelů a jejich hesel (bez nutnosti speciálního uzlu sítě), kterou je NIS (Network Information Services, dříve Yellow Pages). Při přístupu klientů k serverům Secure RPC vynucuje komunikaci prověřování přidělovaným konverzačním klíčem, a to na základě znalosti tajného klíče z databáze. Za znalosti tajného klíče a klíče sezení vytváří klient klíč konverzační a teprve poté je serverem akceptován. Podobně jako u Kerbera i zde platí časové omezení konverzačních klíčů.

Secure RPC

**správce
bezpečnosti**

Z uvedeného přehledu snah ochránit síťový provoz je zřejmé, že se jedná o komplikovaný aparát. Ani erudovaný a zkušený správce sítě nepostupuje bez problémů, zvláště proto, že topologie a rozsah jemu svěřených sítí se dle potřeb provozu organizace stále proměňují. S tímto vědomím dnes osvědčené vedení organizace určuje také osobu se zaměřením na bezpečnost, správce bezpečnosti. Obyčejný uživatel při své práci v organizaci se pak nemusí bezpečnostními riziky zabývat. Základní problém ovšem přináší charakteristika doby v masovém rozšíření výpočetní techniky a její neoddelitelné souvislosti se síťovým provozem. Snahou každého jedince lidstva je dnes vlastnit nejméně PC, pomocí něhož se stává účastníkem Internetu. Technologie, které k této účasti použije a které mu obchodníci bez mrknutí oka prodají, jsou přitom tytéž, které se používají pro zajištění práce náročných síťových celků a které i zde vyžadují uvedené bezpečnostní postupy. Uživatel sice nemusí řešit topologii sítě, násobný přístup uživatelů k serverům informačního systému atp., ale např. jeho výpočetní jednotka je identifikována přidělenou adresou IP a jeho operační systém disponuje možnostmi práce síťových serverů, které jsou mu navíc obvykle nastaveny tak, že své služby do sítě nabízejí. Rizika, která uživatel podstupuje připojením se k síti, jsou dnes velmi vysoká a zcizit uživateli data sítě je mnohdy docela snadné. Zmenšit tato rizika lze používáním kvalifikovaného poskytovatele připojení k veřejné síti, protože poskytovatel může zajistit firewall svým zákazníkům stejně jako sledování provozu sítě směrem k zákazníkovi a od něj tak, aby blokoval přístup nežádoucích klientů z veřejné sítě, případně aby rozpoznával aktivitu červů (viz dále). Technologicky je již dnes možné svěřit se zcela do péče poskytovateli a pro své koncové zařízení využívat pouze zobrazovací jednotku, grafický terminál, jak jsme si např. ukázali u technologie X Window System (není to ale technologie svého typu jediná), viz kresba III-19.



Kresba III-19: Koncový uživatel operačního systému a grafického terminálu.

Takovým způsobem by veškerá bezpečnostní rizika byla řešena kvalifikovaně poskytovatelem v rámci sítě jeho silných výpočetních celků, v jejichž prostředí by současně bylo přihlášeno dnes už třeba i několik tisíc uživatelů současně. Proti takovému zajištění dnes ale jistě stojí zájem obchodu, protože jednoduché zobrazovací zařízení uživatele je výrazně

levnější než celý výpočetní systém (včetně licence za operační systém), jejichž použití by se redukovalo na síťové uzly poskytovatelů. Rovněž by se výrazně snížil počet kupovaných licencí pro aplikační software, protože uživatel by si potřebný software pronajímal u poskytovatele, a to pouze ten, který by skutečně potřeboval a pouze na dobu, kdy by jej potřeboval a tedy v poslední verzi. Instalace software by byla vždy kvalifikovaná, protože by probíhala pouze na hardware poskytovatele a koncový uživatel by se jí vůbec nezabýval. Snížilo by se či dokonce by vymizelo používání software bez licencí. Uživatel by nepotřeboval znalosti pro údržbu svého výpočetního systému a pro zajištění bezpečnosti, nemusel by se obávat ztráty svých dat nebo jejich zcizení. Za jeho data by dle smlouvy zodpovídal a za jejich soukromí a bezpečnost tak právně ručil jeho poskytovatel. Ten by také prováděl pravidelnou zálohu dat svých uživatelů, která by jim jednotlivě byla vždy k dispozici třeba i několik let zpětně nebo dokonce bez omezení.

Takovou možnost dnes ale obyčejný uživatel nemá. Nabídka veřejného využívání strojového času poskytovatelem je dnes velmi malá. Topologii silných serverů s připojováním uživatelů pouze pomocí zobrazovací jednotky je dnes využíváno u kvalifikovaně vedených vnitřních sítí (obvykle komerčních) organizací, případně se uvedený typ *hostingu* nabízí opět celým organizacím, ale ne jednotlivcům.

Pro zamezení nebezpečí průniku do operačního systému by měl koncový uživatel za současné situace používat ochranný podsystém. Jedná se o software, který při běžném provozu výpočetní jednotky prověřuje nové programy a data vstupující do operačního systému. Průnik do operačního systému může být proveden buďto pomocí škůdce na přineseném výměnném médiu (disketa, CD etc.) nebo může přijít počítačovou sítí.

virus *Virus* je škůdce, který způsobí paralýzu operačního systému nebo některé jeho části (např. systému souborů) takovým způsobem, že operační systém a v něm používané aplikace chybují tak, že přestávají být použitelné. Cílem viru je často zničení uživatelských dat, která jsou umístěna na externích diskových perifériích počítače. Pravidelná záloha dat z počítače na CD, diskety nebo jiné výměnné zařízení zabraňuje ztrátě trvalého charakteru, přestože i zálohovaná data mohou být již napadena. Virus je principiálně program jako každý jiný, bývá ale ukrýván do jiných programů nebo dat programů tak, že při spuštění napadeného programu bývá virus v určité části práce algoritmu spuštěn. Je-li virus dobře naprogramován, uživatel nic nepozná, protože virus po ukončení svého běhu předá řízení napadeného programu do místa, odkud byl spuštěn. Co je náplní práce viru, záleží na cílech jeho autora. Virus může pouze sledovat stav operačního systému a v něm instalovaných aplikací a může tyto informace odesílat sítí autorovi viru. Virus se může aktivovat až po určité době nebo ve stanoveném datu a hodině a zničit data nebo celý operační systém (logická bomba). Základní funkcí viru je ale jeho rozmnožování, tedy umisťování kopií sama sebe do jiných zatím nenapadených programů v operačním systému nebo do částí operačního systému, jako jsou systémové knihovny nebo oddělené moduly. Virus dnes přichází počítačovou

sítí především prostřednictvím klienta elektronické pošty, který obsah pošty stahuje ze serveru do operačního systému uživatele. Virus přitom nemusí být samostatný program, ale pouze část dat, která mají pro některou z aplikací charakter algoritmu (skript, makro). Takový skript, je-li dopraven do aplikace sítí a je aplikací interpretován, může znamenat úpravu operačního systému, tj. jeho napadení. Skript si lze jednoduše představit jako část kódu určenou pro zobrazení prohlížečem stránek WWW. Prohlížeče mohou být ovládány pomocí např. jazyka JavaScript, dle autora stránky přenesené ze serveru. Taková interpretace ale může být naštěstí v konfiguraci prohlížeče zakázána a povolena pouze z vyjmenovaných uzlů. Příloha ve stažené elektronické poště může obsahovat program, který je virem. Ovšem i data, jako je např. dokument pro MS Word, mohou obsahovat makra pro zpracování určitých částí dokumentu, ve kterých může být obsažen nebezpečný algoritmus.

Jiným typem škůdce je červ (worm). Jeho základní funkcí je průnik do operačního systému především prostřednictvím síťových aplikačních protokolů. Červ zkouší na jednotlivých uzlech sítě odezvu serverů. Pokud červ zjistí určitý jemu známý server ve verzi, která obsahovala díru, tedy možnost vyvolání neobvyklého stavu serveru, snaží se touto dírou do operačního systému proniknout. Činnost červa po průniku je závislá na úmyslech jeho autora. Může se dál chovat jako virus, od viru se však odlišuje snahou dále se rozšiřovat sítí. Často se červ přemísť postupně mezi různými uzly a snaží se přitom za sebou odstraňovat data, která by mohla i později identifikovat jeho průchod uzlem. Po určité době pak např. zaútočí z mnoha v takto rozšířených uzlech veřejně na některé významné servery Internetu, např. vymění stránky WWW za jiné nebo k nim zahltní přístup mnohonásobným opakováním požadavků pro jejich čtení. Útok je ale již veden z uzlů, do kterých červ pronikl a nebyl odhalen správcem, přitom cesta odkud přišel je smazána nebo stačí, aby byla pouze přerušena.

červ

Paralýza operačního systému virem je vždy způsobena průnikem do části operačního systému, ke kterým by obyčejný uživatel neměl mít přístup pro zápis nebo by vůbec neměl mít právo jakkoliv je svévolně ovlivňovat. Jedná se o chráněný (supervizorový) režim operačního systému, který podléhá obvykle jednomu nebo několika způsobům přihlášení (jako root nebo Administrator) a který není určen pro běžnou práci, ale pouze pro správu výpočetního systému. Tuto situaci jsme podrobně popsali v části Base Of I.3. Pokud by každý operační systém pracoval striktně na takovém principu, virus by mohl způsobit škodu omezenou pouze na data uživatele, který virus do operačního systému stáhl sítí nebo přinesl na výměnném médiu. Samotný operační systém by ale neohrozil a neohrozil by ani data jiných uživatelů. Tuto slabinu obsahují především operační systémy firmy Microsoft. Přestože od verze Windows 2000 jsou navrženy a provozovány v takových dvou režimech, uživatelé je běžně nepoužívají. Důvodů je několik, především je to neznalost, dále jsou to vážné problémy při provozu řady aplikací v jiném než supervizorovém režimu a s tím související historický návyk uživatelů a jejich programů mít otevřený přístup ke všem částem hardware i software operačního systému. Uživatel proto nastaví režim MS Windows pro provoz vždy v supervizorovém režimu, protože se tak

vyhne problémům. Operační systém je ale takto otevřen každému způsobu případného útoku. Tato omezení jsou v současné době důvodem, proč je stále více upřednostňován operační systém UNIX (jeho varianta Linux) na osobních počítačích typu PC.

Kvalitní integrovaný podsystém ochrany práce v operačních systémech firmy Microsoft je vždy dodáván třetí stranou. Jako příklad lze uvést renomovaný Norton Internet Security (viz [Symantec]), ale firem a akademických institucí, které se na tuto činnost zaměřují, je celá řada a lze je snadno vyhledat v Internetu. Ochrana je dnes prováděna již obvykle na všech stupních možného průniku, ať již přílohou v elektronické poště, prohlížením stránek Internetu, průnikem síťovými servery nebo sledováním startu a běhu každé spuštěné aplikace a části operačního systému. Uživatel, byť neznalý principů operačních systémů a sítě, dnes nutně musí v operačních systémech MS Windows ochranné podsystémy používat a snažit se jim porozumět. Že to u drtivé většiny uživatelů prostě není možné pak vysvětluje současné rozmnožení virů a červů různého zaměření. Koncepční řešení je zřejmě jiné než snaha naučit lidskou populaci znát principy operačních systémů a sítě. Jednou možností je jistě přístup zabezpečený poskytováním informačních technologií odborníky jako služby každé jiné sítě (telefonní, elektrické, síť pohonných hmot atd.) ve smyslu termínu hosting, jak bylo uvedeno v předchozím textu.

Oranžová kniha

Obecně je chatrná bezpečnost výpočetních systémů vzhledem k principům informačních technologií známá, a to vlastně již od jejich samotného vzniku. Nejedná se pouze o zanedbaný aspekt, který v průběhu dalšího vývoje nebylo možné jednoduše řešit s ohledem na zpětnou kompatibilitu, přestože částečně tomu tak opravdu je. Bezpečnostní problémy lze ovšem popsat a lze je výrazně omezit sledováním a kvalifikovaným provozem. Jaké jsou požadavky na takovou praktickou bezpečnost, bylo definováno v dokumentech, které dnes mají mezinárodní charakter a uznávanou odbornou váhu. Především se jedná o Trusted Computer System Evaluation Criteria (TCSEC, Kritéria hodnocení důvěry výpočetního systému, viz [TCSEC1985]), který vydalo Ministerstvo obrany vlády Spojených států (Department of Defense, DoD) v roce 1983 a který je běžně označován spíše jako Oranžová kniha (Orange Book). Jednotlivé nároky na bezpečnost jsou zde kategorizovány do jednotlivých skupin s písmenným označením A, B, C a D, kterým říkáme bezpečnostní úrovně. Od písmene D směrem k A se jedná o vyšší bezpečnost, vyšší bezpečnostní úroveň přitom zahrnuje všechna kritéria všech úrovní nižších. Vzhledem k tomu, že úroveň je sestavena vždy z několika nezávislých celků, v rámci každé úrovně je ještě používáno označování doplňujícím číslem. Např. je známá bezpečnost s označením C2 nebo B1 a jejich konkrétní implementace (v UNIXu, ale i v MS Windows NT atp.). Úroveň A je definice, která je stále na hranici praktických možností, přestože je pro vojenské účely navrhována. Úroveň D si lze představit jako bezpečnostní podsystém chráněného operačního systému tak, jak jsme operační systém popsali v části Base Of. Jedná se o chráněný režim a oddělení přístupu uživatelů s různým oprávněním k provádění změn v operačním systému se zadáváním jména

a hesla. Úroveň bezpečnosti v kategorii B1 zase jako výpočetní systém, který je instalován a konfigurován pro jednoznačně dané požadavky provozu a poté je uzamknuta jakákoliv možnost manipulace a další konfigurace operačního systému nebo instalovaných aplikací, síťových serverů atp. Změna konfigurace výpočetního systému v úrovni B1 je možná jediné novou instalací na prázdný hardware.

Přestože je Oranžová kniha biblí zejména výrobců operačních systémů, není jediným mezinárodně uznávaným bezpečnostním dokumentem. V Evropě je mezinárodně respektován dokument Information Technology Security Evaluation Criteria (ITSEC, Kritéria hodnocení bezpečnosti informační technologie, označovaný také jako Bílá kniha, viz [ITSEC1991]) z roku 1990, jemuž za vzor posloužila jednak Oranžová kniha a jednak německý dokument ZSIEC z roku 1989. Rovněž ITSEC hierarchicky kategorizuje bezpečnost podobně jako TCSEC, označování úrovní je ale jiné (F1, F2... F5), F1 je nejslabší, odpovídá přibližně D, F5 přibližně A. I přes snahu starého kontinentu je ovšem takřka vždy požadováno označování bezpečnosti výpočetních systémů ve smyslu kategorií TCSEC, snad proto, že vše podstatné v Computer Science se událo a stále děje ve Spojených státech.

Bílá kniha

Používání média digitální informace se pro nás postupně stává běžnou každodenní činností. Otiskujeme obraz, text, zvuk či jiné smyslové vjemy vnějšího nebo vnitřního světa člověka. Provádíme to různými dnes již komerčně běžně dostupnými prostředky, jakými jsou fotoaparát, filmová kamera, mobilní telefon, scanner, ale i klávesnice, myš, dotekový displej našeho kapesního počítače, senzor pedálu paliva auta nebo otočný přepínač pračky či mikrovlnné trouby. Proces digitalizace probíhá tak, aby jeho produktem byla data, která lze algoritmicky zpracovávat, a to za různým účelem. Požadavky na algoritmické zpracování také určují míru podrobnosti digitalizace. Rodinná vzpomínková fotografie bude jistě určena pouze ke svému pozdější zobrazování v úzkém kruhu rodiny a přátel a požadavek na její kvalitu bude tedy určen jen mírou uspokojivosti pro oko těchto diváků. Oproti tomu podrobnost rentgenového snímku vnitřního orgánu nebo filmového záznamu chemické reakce bude dána přesnými požadavky nutnými pro jejich následnou analýzu, která umožní stanovit diagnózu pacienta nebo určit molekulární změny zkoumané látky. Podobně snímání hlasu mikrofonom, kterým rozmlouvám s kamarádem na druhém konci planety a který je přenášen v digitální podobě počítačovou sítí a u kamaráda interpretován reproduktorem, nebude potřeba digitalizovat na takové úrovni, jak by tomu bylo např. u záznamu koncertu operní pěvkyně.

Proces digitalizace, který je určen k produkci dat, je ale vždy předstupněm jejich algoritmického zpracování, jehož principy byly popsány v předchozím textu.

Na rozdíl od způsobů digitalizace a uchovávání dat, které je v neustálém vývoji co do rychlosti a objemu, jsou tyto principy v dosavadní Computer Science neměnné.

Vskutku, ohromení nad možnostmi práce s digitální informací prožívá lidstvo pouze několik let a seznamuje se tak s implementacemi, které byly prováděny již v 60. a 70. letech minulého století. Komerční dostupnost jak periférií digitalizace, tak výpočetních celků zpracování dat ruku v ruce s nárůstem jejich výkonu a schopnosti pojímat stále větší objem způsobuje devaluaci promyšleného přístupu k řešení problémů a hledání metod v současné Computer Science. Často je práce na zpracovávajících algoritmech nejenom v praxi, ale i v laboratořích firem produkujících software prováděna způsobem hrubé síly a nikoliv způsobem promyšleným. Mnohdy to dostačuje a většina uživatelů získává dojem, že řešení je vždy v bezmyšlenkovitém násobně opakovaném používání týchž algoritmů, přestože řešení může být zcela jiné, mnohdy jednodušší a myšlenkově nové.

Nové myšlenky ale do Computer Science přicházejí pomalu. Každý kvalifikovaný specialista si totiž uvědomuje omezení, která jsou zakleta v popsanych principech. Jedná se o stále stejné způsoby spojování, provádění výpočtu, stále stejný princip připojování periférií k centrální jednotce, stále tentýž způsob zpracování dat.

Jako příklad může posloužit vývoj zpracování (analýza) obrazu. Přestože v současné době dokážeme provádět digitalizaci za stále vyššího zjemnění, mnohdy tolik potřebná kvalitní analýza obrazu za využití inteligentních vědomostních systémů je zatím stále věcí science fiction. Partie umělé inteligence (AI, Artificial Intelligence), která je s analýzou obrazu úzce spojena, prošla v průběhu svého vývoje krizí. V jejím důsledku upustila od naivního hledání algoritmické simulace tohoto vůbec obtížně definovatelného chování lidského mozku (heuristickou funkcí) a postupně se věnuje algoritmickému zpracování chování neuronových sítí. Výsledky jsou prozatím malé, protože rozsah simulované sítě je řádově v jednotkách počtu neuronů (přestože v intencích Computer Science to lze považovat za výrazný úspěch).

Nedaří se také řízení paralelního provádění výpočtu ve vzájemně přímo propojených CPU tak, aby toto propojení přineslo zvýšení požadovaného výkonu (víceprocesorové architektury, clustery). Jedná se o jeden z kruciólních problémů, protože představíme-li si, že bychom obecně dokázali proces algoritmického zpracování dat rozložit na libovolný počet procesorů, mělo by dojít k posílení výkonu na základě zvýšení počtu center výpočtu a v důsledku toho k možnosti dynamického využívání nabízeného strojového času volných CPU. Problém ale spočívá především v tom, že klasický algoritmus je zpracováván sekvenčně, kdy současně probíhající výpočty procesorů lze nad prováděním jedné sekvence synchronizovat jenom velmi obtížně. Hledá se proto jiný princip zpracování než je sekvenční. To by pak mohlo změnit základní principy zpracování dat nebo je alespoň doplnit.

Ve výčtu základních problémů, které potřebují informační technologie vyřešit pro svůj další vývoj, lze pokračovat také např. virtuální realitou, simulacemi a vytvářením umělých světů vůbec. Souvislost s robotikou a ochranou lidského jedince před nebezpečným prostředím nebo nebezpečnou budoucností je pak více než jistá.

Informační technologie tedy přešlapují, dnes však nikoliv na možnostech své implementace, ale na svých možnostech mentálních.

Na druhé straně informační technologie vždy vystupovaly z věží ze slonoviny, kam se je občas snaží zahnat matematici, teoretikové a akademici, přestože bez jejich účasti by vůbec nevznikly a ani se nemohly vyvíjet. Doposud krátká historie počítačích strojů digitální povahy totiž ukazuje omyly cílené snahy řídit jejich vývoj odtazité od praxe a života. Např. v hlubinách času, na přelomu 70. a 80. let minulého století, bylo seriózní snahou vědců vyvinout obří výpočetní systém, který by prováděl řadu výpočtů, dokázal sám číst denní tisk, případně se mohl sám učit na základě scanu denního tisku a archivů lidské společnosti. Výsledkem měl být inteligentní stroj, který by napomáhal při řešení obecných problémů lidské civilizace. Tato představa byla tehdy popisována ve většině literatury science fiction, kde centrální počítač řídil kosmické lety a životy lidí. Nákladný projekt, který začal být realizován nezávisle na sobě na několika místech na Zemi, byl ale neúspěšný. Zájem o něj rychle vyprchal v průběhu vlny, která přinesla náhle objevené osobní počítače a ty do několika let zaplavily svět. Vznikly na základě zkušeností

z počítačů domácích jako projekt určený pro hry a domácí kutily. Distribuce výkonu v podobě samostatných výpočetních celků se ukázala jako potřebná pro řadu činností lidí různých oborů. Neustálý nárůst výkonu osobních počítačů a jejich úložných kapacit se spojil se snahou o datové komunikace a vyústil v komerční tlak na vývoj počítačových sítí. Výsledkem byl vznik Internetu. Náhle se ukázala pošetilost a zbytečnost vývoje obřího výpočetního systému. K čemu by měl sloužit? Jaký užitek by z něj plynul?

Druhým příkladem je snaha vypracovat uvažující systémy na bázi již zmíněné umělé inteligence. Po prvopočátcích, kdy inteligenci zastupovala pouze heuristická funkce, se vědci pustili do simulace neuronových sítí. Přestože se postupně daří je implementovat, zájem je malý. V praxi je totiž vždy požadován spíše systém poradenský, ve kterém se skrývají utříděné informace slovníkového charakteru, tedy systém sloužící jako prodloužená paměť člověka. Inteligenci k řešení problému totiž produkuje člověk sám, nikoliv stroj.

Jakoby lidé byli vždy při klíčových rozhodnutích vývoje v Computer Science bezohledně strženi vždy jinou cestou, než kterou lopotně s použitím racionálního myšlení sami stanovili. Podobně jako je smysl činnosti lidí stále nerozpoznatelný, jsou nerozpoznatelné i cesty vývoje informačních technologií a předměty dalšího zkoumání v Computer Science.

Brno, Rovensko pod Troskami, Vieste

2002–2005

ASCII (American Standard Code for Information Interchange) je výchozí **ASCII** způsob digitálního kódování znaků na 8 bitech.

Záhlaví sloupců v následující tabulce **2, 10, 8, 16** označuje zápis v číselné soustavě s odpovídajícím základem, tj. dvojkovým (binárním), desítkovým (dekadickým), osmičkovým (oktálovým), šestnáctkovým (hexadecimálním).

Ve sloupci se záhlavím *klávesnice* je v případě předřazeného ^ vyjádřena kombinace stisku klávesy Ctrl a jejího podržení při následném stisku dále uvedené klávesy. Jedná se o původní notaci u popisu zadávání řídicích znaků s odpovídajícím významem.

2	10	8	16	klávesnice	znak	význam
00 000 000	0	00	00	^@	NULL	binární nula
00 000 001	1	01	01	^a	SOH	(Start Of Header) následuje záhlaví
00 000 010	2	02	02	^b	STX	(Start Of Text) následuje text
00 000 011	3	03	03	^c	ETX	(End Of Text) končí text
00 000 100	4	04	04	^d	EOT	(End Of Transmission) končí přenos
00 000 101	5	05	05	^e	ENQ	(Enquiry)
00 000 110	6	06	06	^f	ACK	(Acknowledgement) potvrzení
00 000 111	7	07	07	^g	BEL	(Bell) pípnutí
00 001 000	8	10	08	^h	BS	(Backspace) zpět
00 001 001	9	11	09	^i	HT	(Horizontal Tab) horizontální tabulátor
00 001 010	10	12	0a	^j	LF	(Line Feed) nový řádek
00 001 011	11	13	0b	^k	VT	(Vertical Tab) vertikální tabulátor
00 001 100	12	14	0c	^l	FF	(Form Feed) nová stránka
00 001 101	13	15	0d	^m	CR	(Carriage Return) na začátek řádku
00 001 110	14	16	0e	^n	SO	(Shift Out)
00 001 111	15	17	0f	^o	SI	(Shift In)
00 010 000	16	20	10	^p	DLE	(Data Link Escape)
00 010 001	17	21	11	^q	DC1	(Device Control 1)
00 010 010	18	22	12	^r	DC2	(Device Control 2)
00 010 011	19	23	13	^s	DC3	(Device control 3)
00 010 100	20	24	14	^t	DC4	(Device control 4)
00 010 101	21	25	15	^u	NAK	(Native Acknowledgement) odmítnutí
00 010 110	22	26	16	^v	SYN	(Synchronous Idle)
00 010 111	23	27	17	^w	ETB	(End of Transmission Block) končí přenosový blok
00 011 000	24	30	18	^x	CAN	(Cancel) zrušení
00 011 001	25	31	19	^y	EM	(End of Medium)

2	10	8	16	klávesnice	znak	význam
00 011 010	26	32	1a	^z	SUB	(Substitute)
00 011 011	27	33	1b	^[, Esc	ESC	(Esacpe), opuštění, únik, výjimka
00 011 100	28	34	1c	^\	FS	(File Separator) ukončení souboru
00 011 101	29	35	1d	^[GS	(Group Separator) ukončení skupiny
00 011 110	30	36	1e	^^	RS	(Record Separator) ukončení záznamu
00 011 111	31	37	1f	^_	US	(Unit Separator) ukončení jednotky
00 100 000	32	40	20	mezera	SP	(space), mezera
00 100 001	33	41	21	!	!	vykřičník
00 100 010	34	42	22	“	“	uvozovky
00 100 011	35	43	23	#	#	křížek
00 100 100	36	44	24	\$	\$	dolar
00 100 101	37	45	25	%	%	procenta
00 100 110	38	46	26	&	&	ampersand, značka &
00 100 111	39	47	27	,	,	apostrof
00 101 000	40	50	28	((levá kulatá závorka
00 101 001	41	51	29))	pravá kulatá závorka
00 101 010	42	52	2a	*	*	hvězda
00 101 011	43	53	2b	+	+	plus
00 101 100	44	54	2c	,	,	čárka
00 101 101	45	55	2d	-	-	pomlčka
00 101 110	46	56	2e	.	.	tečka
00 101 111	47	57	2f	/	/	lomítko
00 110 000	48	60	30	0	0	číslice nula
00 110 001	49	61	31	1	1	číslice jedna
00 110 010	50	62	32	2	2	číslice dva
00 110 011	51	63	33	3	3	tři
00 110 100	52	64	34	4	4	čtyři
00 110 101	53	65	35	5	5	pět
00 110 110	54	66	36	6	6	šest
00 110 111	55	67	37	7	7	sedm
00 111 000	56	70	38	8	8	osm
00 111 001	57	71	39	9	9	devět
00 111 010	58	72	3a	:	:	dvojtečka
00 111 011	59	73	3b	;	;	středník
00 111 100	60	74	3c	<	<	levý zobák

2	10	8	16	klávesnice	znak	význam
00 111 101	61	75	3d	=	=	rovnítko
00 111 110	62	76	3e	>	>	pravý zobák
00 111 111	63	77	3f	?	?	otazník
01 000 000	64	100	40	@	@	zavináč
01 000 001	65	101	41	A	A	písmeno velké A
01 000 010	66	102	42	B	B	velké B
01 000 011	67	103	43	C	C	velké C
01 000 100	68	104	44	D	D	...
01 000 101	69	105	45	E	E	
01 000 110	70	106	46	F	F	
01 000 111	71	107	47	G	G	
01 001 000	72	110	48	H	H	
01 001 001	73	111	49	I	I	
01 001 010	74	112	4a	J	J	
01 001 011	75	113	4b	K	K	
01 001 100	76	114	4c	L	L	
01 001 101	77	115	4d	M	M	
01 001 110	78	116	4e	N	N	
01 001 111	79	117	4f	O	O	
01 010 000	80	120	50	P	P	
01 010 001	81	121	51	Q	Q	
01 010 010	82	122	52	R	R	
01 010 011	83	123	53	S	S	
01 010 100	84	124	54	T	T	
01 010 101	85	125	55	U	U	
01 010 110	86	126	56	V	V	
01 010 111	87	127	57	W	W	
01 011 000	88	130	58	X	X	
01 011 001	89	131	59	Y	Y	
01 011 010	90	132	5a	Z	Z	
01 011 011	91	133	5b	[[levá hranatá závorka
01 011 100	92	134	5c	\	\	obrácené lomítko
01 011 101	93	135	5d]]	pravá hranatá závorka
01 011 110	94	136	5e	^	^	stříška (karet)
01 011 111	95	137	5f	_	_	znak podtržení
01 100 000	96	140	60	`	`	obrácený apostrof

2	10	8	16	klávesnice	znak	význam
01 100 001	97	141	61	a	a	písmeno malé a
01 100 010	98	142	62	b	b	písmeno malé b
01 100 011	99	143	63	c	c	malé c
01 100 100	100	144	64	d	d	...
01 100 101	101	145	65	e	e	
01 100 110	102	146	66	f	f	
01 100 111	103	147	67	g	g	
01 101 000	104	150	68	h	h	
01 101 001	105	151	68	i	i	
01 101 010	106	152	6a	j	j	
01 101 011	107	153	6b	k	k	
01 101 100	108	154	6c	l	l	
01 101 101	109	155	6d	m	m	
01 101 110	110	156	6e	n	n	
01 101 111	111	157	6f	o	o	
01 110 000	112	160	70	p	p	
01 110 001	113	161	71	q	q	
01 110 010	114	162	72	r	r	
01 110 011	115	163	73	s	s	
01 110 100	116	164	74	t	t	
01 110 101	117	165	75	u	u	
01 110 110	118	166	76	v	v	
01 110 111	119	167	77	w	w	
01 111 000	120	170	78	x	x	
01 111 001	121	171	79	y	y	
01 111 010	122	172	7a	z	z	
01 111 011	123	173	7b	{	{	levá složená závorka
01 111 100	124	174	7c			svislá čára
01 111 101	125	175	7d	}	}	pravá složená závorka
01 111 110	126	176	7e	~	~	vlnovka (tilda)
01 111 111	127	177	7f	Delete	DEL	(Delete) odstranění

UNICODE je standardizovaný způsob (ISO/IEC 10646) kódování všech znaků pro všechny abecedy a to tak, že každý znak je digitalizován jednoznačně. Pro tuto jednoznačnost je nutné zvětšit rozsah kódování, používají se dva bajty. UNICODE je mezinárodní konvence pro digitální hodnoty všech známých používaných nebo historických znaků. Přestože UNICODE byl poprvé vydán v roce 1996, dodnes není plně implementován ve všech operačních systémech a nelze proto obsah souborů takto zpracovávat implicitně. Pokud by tomu tak ale bylo, text souboru by byl sice větší, ale mohl by současně obsahovat text český prokládaný německým, španělským nebo arabským, ruským nebo textem napsaným ve staré čínštině. To ovšem není pouze věcí kódu samotného, ale aplikačního programu (textového editoru, programu elektronické pošty), který musí dokázat znaky správně interpretovat, a to za podpory operačního systému.

UNICODE

Principy pro UNICODE jsou následující:

- kódování je prováděno na 16 bitech, které jsou všechny využívány pouze na vyjádření kódovaného znaku,
- prvních 8 bitů označuje mezinárodní zařazení,
- druhá část je na 8 bitech kódována tak, aby co nejvíce odpovídala umístění sémanticky odpovídajících znaků ASCII a znaků v jiných (národních) abecedách,
- kódování se snaží odpovídajícím zařazením dodržovat logiku používání znaků, která se uplatní např. při třídění nebo porovnávání textů,
- kódování sjednocuje duplicity, např. znak A je kódován pouze jednou.

URL (Uniform Resource Locator) je způsob, kterým dotazující se klient určuje některé výpočetní zdroje v rámci sítě typu internet. Pro odkazy v rámci WWW jej definoval Tim Berners-Lee, který je také zakladatelem a vedoucí osobností organizace World Wide Web Consortium.

URL bylo standardizováno v roce 1998 dokumentem RFC2396, ze kterého také vyplývá, že URL je zúžením obecněji definovaného URI (Uniform Resource Identifier). Pomocí URI lze jednoznačně identifikovat jakýkoliv výpočetní zdroj v rámci sítě typu internet. Vzhledem k tomu, že URI zahrnuje URL jako svoji součást, je termín URI někdy používán namísto URL.

Řetězec znaků odkazu URL definuje formát:

```
service://user:password@host:port/path
```

Klient může použít pouze část *host*. Určuje tak pouze uzel v síti, a to buďto tečkovým vyjádřením doménových jmen v rámci DNS, např. www.skocovsky.cz nebo přímo pomocí adresy IP, např. 212.65.244.7. Korektní je tedy např. URL

```
www.kokolia.cz
```

a jedná se o přístup pomocí protokolu HTTP na jeho standardním portu. Klient by totéž mohl vyjádřit také zápisem

```
http://www.kokolia.cz
```

kde *http* je požadovaný typ protokolu na straně serveru. Jedná se o část *service* (služba) z uvedeného formátu. *service* je určení komunikačního protokolu a tedy typu spojení. Na místě *service* lze použít:

<i>http</i>	zobrazování stránek WWW (Hypertext Transfer Protocol)
<i>https</i>	zabezpečené zobrazování stránek WWW (HTTP over SSL)
<i>ftp</i>	přenos souborů (File Transfer Protocol),
<i>ftps</i>	zabezpečený přenos souborů (FTP over SSL),
<i>file</i>	odkaz na zdroje místního systému souborů,
<i>ldap</i>	adresářová služba (Lightweight Directory Access Protocol),
<i>mailto</i>	poštovní služba,
<i>telnet</i>	služba podpory práce ve vzdáleném uzlu,
<i>rlogin</i>	rovněž služba pro práci ve vzdáleném uzlu,
<i>news</i>	noviny,
<i>gopher</i>	textový předchůdce podpory WWW,

případně další, které byly používány dříve nebo ty, které budou používány v budoucnu.

Každý protokol má smluvený komunikační port, který je pro klienta a server vyhrazen v rámci standardního použití. Proto část *port* nemusí být uvedena v URL, je-li tomu tak. Opět tentýž odkaz by tedy mohl být zapsán také

```
http://www.kokolia.cz:80
```

kde 80 je číslo portu. Jak klient tak server mohou ale využívat ke komunikaci i jiný port. Pokud klient ví, že vzdálený server je připraven ke komunikaci na jiném portu, může tuto komunikaci otevřít zadáním např.

`http://www.kokolia.cz:8008`

(což je běžný port pro správu vzdálené služby HTTP).

Služba může být chráněna přístupovým jménem a také heslem. To je ve formátu určeno částí `user:password@`. Např. správa služby `http` bude pravděpodobně vždy vyžadovat přístupové jméno a heslo, které si od uživatele vyžádá interaktivně pomocí okna klienta, pokud jej klient nezadá přímo v URL, např.

`http://administrator:pust_mne_dovnitř@www.wikipedia.org:8008`

Přístupové jméno však může být použito i bez hesla pouze jako identifikace uživatele, např. u poštovní služby:

`mailto://pavel@skocovsky.cz`

Konečně nepovinná je i cesta k dokumentu `path`. Je také implicitně určena u různých protokolů. Např. u `http` se jedná o dokument se jménem `index.html`:

`www.kokolia.cz/index.html`

Adresa IP **Adresa IP** (Internet Protocol Address) je jednoznačnou číselnou identifikací uzlu v síti typu internet. Má rozsah 32 bitů, zapisujeme ji ve tvaru čtyř čísel oddělených tečkou. Každé číslo reprezentuje 8 bitů (1 bajt).

Podle potřeby shromažďování uzlů do sítí o různém počtu uzlů rozlišujeme adresy typů A, B, C a D:

A:

rozsah `1.0.0.1 až 126.255.255.255`,
síť po první tečku zleva,
uzel v síti tři čísla zprava,
dá se vyjádřit 126 sítí, v každé síti může být umístěno až 16 milionů uzlů,
přiděluje se velmi málo.

B:

rozsah `128.0.0.1 až 191.255.255.255`,
síť dvě čísla zleva,
uzel v síti dvě zprava,
16256 sítí, v každé síti přibližně 65000 uzlů,
přiděluje se často velkým organizacím.

C:

rozsah 192.0.0.1 až 223.255.255.255,
sít' tři čísla zleva,
uzel jedno číslo zprava,
přibližně 2 miliony sítí, v každé jen 254 uzlů,
přiděluje se často pro menší sítě.

D:

rozsah 224.0.0.1 až 254.255.255.255,
adresa multicast,
uzel je vyjádřen všemi čtyřmi čísly,
používá se výjimečně, např. pro zajištění satelitního propojení
obrovského množství uzlů (interaktivní televize atp.).

Výjimky:

- *localhost*, tzv. loopback, adresa 127.0.0.1

Jedná se o virtuální odkaz na sama sebe, používá se pro testování správné funkce vrstev sítě nad vrstvou IP a také jejím prostřednictvím komunikují síťové aplikace, kdy klient a server běží v tomtéž uzlu.

- Síťová maska (netmask).

Je vypočtena modulem implementace IP a uživatele nemusí zajímat. Zajímá správce sítě, protože jejím jiným zadáním může síť rozdělit na několik podsítí. Používá se např. k přidělování pouze několika adres různým organizacím připojovaným k Internetu, které mají vlastní síť oddělenou pomocí technologie firewall. Pak vzniká vnější a vnitřní síť jako samostatná síť typu IP a z Internetu je vnitřní síť dostupná pouze prostřednictvím jedné nebo několika viditelných adres. Také se používá u adres typu B na síťové oddělení různých lokalit nebo oddělení firmy v různých zbylým Internetem oddělených sítích.

- Broadcast.

Adresa určená ke zkoumání připojených a nepřipojených uzlů sítě oběžníkem. Pokud není řečeno jinak, modul implementace IP přidělí adresu broadcast jako poslední adresu uzlu v síti, např. 192.10.255.255 je implicitní broadcast sítě 192.10.0.0 Každý uzel této sítě, zachytí-li v síti paket s touto adresací, odpovídá jeho odesílateli stavem svého připojení k síti.

- Implicitní směrování (default routing).

Je adresa, kudy se hledá cesta ven ze sítě. Je jí vyhrazena 0 v síťové adrese, (např. 192.10.0.0). Nemá-li uzel explicitně zadánu adresu směrování, prostřednictvím této adresy vede cesta k ostatním sítím internetu. Adresu tohoto implicitního směrování používá modul IP.

HyperText Markup Language je jazyk určený pro formátování dokumentů. Dokument je sestaven z jedné nebo libovolného počtu stránek. Stránka (page) je součástí dokumentu, je-li na ni z dokumentu odkazováno. U každého dokumentu hovoříme o stránce, která je výchozí, říkáme jí domovská stránka (home page). Každá stránka je samostatný celek, na který můžeme odkazovat prohlížečem internetu bez toho, abychom nejprve navštívili její domovskou stránku. Do dokumentu tedy můžeme vstoupit odkazem na libovolnou jeho stránku, nemusíme mít ale zaručeno, že tak přečteme celý dokument. Přistupovat k dokumentu z jeho domovské stránky se proto doporučuje, ale známe-li již dokument dobře, není to nutné. Dokument nemusí mít žádnou domovskou stránku. Libovolnou stránku kteréhokoliv zveřejněného dokumentu v Internetu můžeme zahrnout do svého dokumentu vytvořením odkazu na ni. Platnost této stránky ale záleží na autorovi dokumentu a změnil-li on její umístění, náš odkaz ztratí svůj cíl. Úplný odkaz na stránku se provede pomocí URL (viz Příloha B).

tag
Každá stránka je text, do kterého vkládáme obrázky, případně pohyblivé obrázky. Stránka může být obohacena zvukem. Výchozí je prostý text, který je prokládán odkazy na jiné stránky. *Tag* je formátovací pokyn pro prohlížeč stránky, uživateli se v okně prohlížeče nezobrazuje. Tag je text uvedený mezi znaky < a >, např.
 je pokyn pro zalomení řádku, na základě <HR> prohlížeč vykreslí vodorovnou čáru. Tagy mohou být nepárové a párové. Párový tag znamená, že je vymezena platnost způsobu formátování, text je uzavřen mezi označení <≡> a </tag>, kde *tag* zastupuje text konkrétního tagu. Některé tagy mají značku ukončení platnosti nepovinnou, vlivem vyvíjejících standardů jich ale valem ubývá, např. při formátování textu do tabulky je nový řádek uveden tagem <TR>, jemuž nemusí předcházet ukončení předchozího řádku (pomocí </TR>). Není důležité, zda je tag napsán malými nebo velkými písmeny. Text stránky je uložen jako prostý text, přípona jména souboru je .html a používá se také jen .htm. Domovská stránka určité URL bývá uložena pod jménem home.html nebo index.html, takže uživatel nemusí zadávat celé její URL, namísto www.skocovsky.cz/home.html lze pouze psát www.skocovsky.cz. Pokud máte v uzlu se serverem stránek (httpd) v tomtéž adresáři soubor home.html i index.html nebo dokonce i home.htm a index.htm, pak o tom, který ze souborů bude použit, rozhoduje server, resp. správce serveru.

Struktura textu stránky tak, aby se v ní prohlížeč dobře vyznal, je následující:

```
<HTML>
<HEAD>
...
<TITLE>...</TITLE>
...
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

Tag `HTML` začíná a ukončuje stránku. Tagem `TITLE` vymezujeme text, který se objevuje v horní liště okna prohlížeče. V oblasti tagu `HEAD` mohou být použity další specifikace stránky. Za `<BODY>` začíná text, který prohlížeč formátuje a zobrazuje uživateli. Mezi `</BODY>` a `</HTML>` nic není.

validator Zda je stránka dobře zkomponována prověříme jejím zobrazením v prohlížeči, a to použitím služby `file` v zadání URL. Každý prohlížeč je ale v dnešní době stále ještě proprietární, test, zda stránka odpovídá definici držitele licence současné oficiální verze definice HTML, lze provést na `validator.w3.org`.

Komentáře konstruktéra stránky, které se uživateli nezobrazují, jsou uvedeny mezi `<!--` a `-->`, např.

```
<!-- sem by měl přijít ještě nějaký obrázek, ale bůhví kam jsem
ho založil... -->
```

<A> Odkaz na jinou stránku provádíme párovým tagem `A`, např.

```
...
<BODY>
... je vydavatelství počítačové literatury <A HREF=www.ora.com>
O'Reilly & Associates, Inc.</A>, které vřele doporučujeme pro
další studium ...
</BODY></HTML>
```

text `O'Reilly & Associates, Inc.` bude uveden jako citlivý, odkazuje na stránku na adrese URL uvedeně za `HREF=` a po kliknutí uživatele a domluvě se serverem ji stáhne a zobrazí.

**** Obrázek vkládáme do textu nepárovým tagem `IMG`, např.:

```
...
<IMG SRC=logo.jpg>
...
```

kdy v adresáři vedle textového souboru se stránkou je také soubor se jménem `logo.jpg`, který obsahuje obrázek, který si prohlížeč stáhne následně po stažení stránky a zobrazí obrázek na odpovídajícím místě stránky mezi textem. Text může obrázek obtékat zprava, zleva atd., viz [MuscianoKennedy2000].

Obrázek může být jako odkaz, tj. celý obrázek je citlivý jako kdyby šlo o tlačítko, které kliknutím myši stlačíme. Obrázek lze tedy použít na místě odkazovacího tagu `A`, např.:

```
...
<A HREF=/kontakty.html><IMG SRC=logo.jpg></A>
...
```

Jeden obrázek může být citlivý ve svých různých částech na různé odkazy. Jedná se o konstrukci mapy obrázku. Obrázek můžeme rozložit na obdélníky, trojúhelníky a jiné geometrické obrazce a pro takto určené části obrázku můžeme stanovit odkazy vedoucí vždy na jinou stránku v internetu. Souřadnice v obrázku určujeme v počtech pixelů, musíme proto

mít obrázek naměřený v pixelech z některého grafického editoru. Je to náročné, ale působivé, podrobnější informace lze získat opět ve vynikající [MuscianoKennedy2000].

Písmo volně plyne na obrazovce. Není-li uvedeno žádným tagem, prohlížeč si jeho velikost, tvar, barvu a další vlastnosti stanoví sám svým výchozí nastavením (např. černé písmo na bílém pozadí, velikost 10 pixelů, font Times New Roman). Vlastnosti zobrazení písma můžeme ale ovlivnit, a to především pomocí tagu `FONT`. Např. písmo bude červené o velikosti 5 pixelů pomocí ****

```
...
a teď červeně: <FONT COLOR=red SIZE=5>to je co?</FONT>, hmm
..
```

Atributem `FACE` tagu `FONT`, např. ``, můžeme stanovit typ písma, který vyžadujeme, aby prohlížeč použil při zobrazování stránky. Vyžadovat můžeme, ale marná sláva, pokud odpovídající font uživatel v operačním systému svého prohlížeče nemá, prohlížeč použije některý zástupný font. Je tedy důležité znát obecně používané a implicitně přítomné fonty v operačních systémech a používat je.

K tomu, abychom mohli pohodlně vytvářet nadpisy různých úrovní, používáme tagy `Hn`, kde `n` je číselné označení úrovně nadpisu, `H1` je např. nadpis nejvyšší úrovně (např. název dokumentu), `H6` nejnižší, např. **<Hn>**

```
<H1>Padla vláda!</H1>
<H2>premiér konečně podal demisi</H2>
<P>Včera<BR>ve večerních hodinách ...</P>
...
```

kde jsme výjma dvou úrovní nadpisů použili také označení začátku a konce odstavce tagem `P`. Atributy tagu `P` mohou určovat zarovnání textu odstavce, např. `<P ALIGN=right>` znamená, že odstavec bude zarovnán na pravý okraj, implicitně je zarovnávání provedeno na levý okraj (`left`), jak je v kraji zvykem, zarovnání do bloku použitím `justify`. V příkladu jsme také použili nepárový tag `BR`, který určuje místo tvrdého ukončení řádku. Konce řádků v textu stránky totiž nemají žádný vliv na výsledné zobrazení, každý způsob formátu musíme určit tagem. **<P>** **
**

Odstavec určujeme také párovým tagem `DIV`, který je ve svých attributech `P` velmi podobný. Zkušenější autoři používají nejvíce pro odstavce párový tag `SPAN`, se kterým se dobře pracuje při využívání kaskádových stylů (viz dále) nebo skriptovacích jazyků. **<DIV>** ****

Text stránky můžeme prokládat formátovacími párovými tagy, tag `B` označuje začátek a konec textu, který bude uveden tučně, `I` je tag pro kurzívu, `U` pro podtržené písmo. **** **<I>** **<U>**

V textu můžeme používat tagy pro seznamy (výčty, odrážky). Tag `UL` používáme pro netřídný seznam, jednotlivé odrážky určujeme vždy tagem `LI`, např.: **** ****

```

Je potřeba<UL>
<LI>jednak jmenovat premiéra,</LI>
<LI>ten by měl sestavit vládu,</LI>
<LI>vládu musí potvrdit parlament.</LI>
</UL>

```

**** Vyměníme-li v příkladu tag `UL` za `OL`, použijeme číslovaný seznam, tedy odrážky se promění v očíslované body.

Tabulky jsou důležitým aparátem nejenom pro vytváření škatulek na stránce, ale i při zarovnávání textů, jejich umisťování do sloupců, vytváření seznamů o více sloupcích atd. Tabulku určíme párovým tagem `TABLE`, řádky tabulky tagem `TR` a sloupce `TD`. Namísto `TR` můžeme použít tag `TH`, kterým vyjadřujeme formátování textu v záhlaví sloupců. Záhlaví celé tabulky označíme tagem `CAPTION`, např.:

<TABLE>
<TR> **<TH>** **<TD>**
<CAPTION>

```

<TABLE>
<CAPTION>Co mám rád<CAPTION>
<TH>1. sloupec<TD></TD>2. sloupec<TD></TD><TD>3. sloupec</TD></
TH>
<TR><TD>hrách</TD><TD>luštěnina</TD><TD>chutná</TD></TR>
<TR><TD>čočka</TD><TD>luštěnina</TD><TD>nechutná</TD></TR>
</TABLE>

```

Textový obsah jednotlivých buněk tabulky můžeme zarovnávat, můžeme měnit barvu pozadí buněk, tabulku můžeme orámovat různě tučnou barevnou čarou, sloupce a řádky můžeme od sebe odsadit a stanovit přesnou šířku sloupců v pixelech atd.

<FRAME>

Rámce (frames) určují rozložení okna prohlížeče na jednotlivé části, ve kterých jsou zobrazovány dokumenty samostatných stránek. Jisté tedy je, že se jedná o obdélníkové vymezení částí stránek. Rámce mohou být uživatelem (čtenářem) v okně prohlížeče měnitelné (myší metodou drag and drop) a nebo je jejich vymezení v okně fixní. Dokument se využitím rámců větví, ale bez použití odkazů. Jednotlivé stránky rozdělovníku jsou přímo zobrazeny v okně prohlížeče. Stránka každého rámce je i zde uložena v samostatném souboru. Do jednoho zobrazení v rámcích tyto stránky sdružuje stránka s definicí rámců. Takový soubor s definicí rámců může být docela krátký, rámce totiž definujeme v oblasti tagu `HEAD`, např.:

```

<HTML>
<HEAD>
<TITLE>Ve stránce používáme rámce</TITLE>
<FRAMESET COLS="20%,*">
<FRAME NAME="seznam" SRC="seznam.html">
<FRAME NAME="obsah" SRC="obsah.html">
</FRAMESET>
</HEAD>
</HTML>

```

Párovým tagem `FRAMESET` definujeme rozložení rámců. V atributu `COLS` jsme v příkladu určili, že okno prohlížeče bude rozděleno na rámce jako na dva sloupce, a to tak, že první rámeček bude vždy mít šířku 20% z celkové šířky okna. V tomto levém sloupci bude zobrazována stránka s textem ze souboru `seznam.html` a v pravém sloupci okna prohlížeče bude zobrazována stránka ze souboru `obsah.html`. Rámce jsme v příkladu pojmenovali, protože pak se na ně můžeme příjemně odkazovat, např. použijeme-li ve stránce `seznam.html` odkaz, můžeme zobrazení tohoto odkazu nasměrovat atributem `TARGET` do rámce `obsah`, čímž se promění jiná část okna, např.:

<FRAMESET>

```
...
<A HREF=clenove.html TARGET=obsah>členové vlády</A>
...
```

Formuláře jsou nositelé dynamického chování, tedy vzájemné interakce uživatele a serveru při prohlížení stránek. Formulář ve stránce obsahuje kolony, výběry a volby, které uživatel vyplňuje, zaškrťává a vybírá. Po vyplnění formuláře odesílá tyto informace serveru stisknutím nějakého tlačítka. Server informace převezme a předá je ke zpracování programu, který je ve formuláři autorem stránky jednoznačně určen. Program informace zpracuje a vytvoří pro uživatele odpověď ve tvaru textu stránky HTML, kterou server uživateli ihned odešle a prohlížeč zobrazí. Uživatel se tak dozví výsledek zpracování svých údajů.

<FORM>

Formulář určíme párovým tagem `<FORM>`, místa, kam uživatel v prohlížeči wpisuje informace, nepárovým tagem `<INPUT>`. Např.:

```
...
<FORM ACTION=/cgi-bin/uloz_to>
Vaše jméno:<INPUT TYPE=text NAME=jmeno SIZE=15><BR>
Příjmení: <INPUT TYPE=text NAME=prijmeni SIZE=32><BR>
Jste
<INPUT TYPE=radio NAME=pohlavi VALUE=1> muž
<INPUT TYPE=radio NAME=pohlavi VALUE=2> žena<BR>
<INPUT TYPE=submit NAME=Odeslat>
</FORM>
...
```

je definice formuláře, kde do kolony s označením `jmeno` a `prijmeni` wpisuje uživatel volný text a může si vybrat pouze jednu z možností označenou textem muž nebo žena. Server na základě stisku tlačítka s textem `Odeslat` dostává jednak obsah textu kolonek `jmeno` a `prijmeni` a jednak buďto informaci 1 nebo 2. Všechny informace jsou pochopitelně serveru sděleny s označením jména `NAME` z formuláře. V příkladu má server za úkol přijaté informace z formuláře předat programu s názvem `uloz_to` a převzít od něj text, který tento program vyprodukuje. Zda je program na serveru přítomen a co s informacemi udělá, je věcí nejenom konstruktéra stránky, ale i programátora programu `uloz_to` a správce serveru, který program na odpovídající místo instaluje.

- PHP, JSP** Pro pohodlnější práci s dynamickými stránkami lze používat programovací jazyky, jako je např. PHP nebo JSP. Notace jejich použití vychází z uvedeného schématu používání formulářů a podmínkou jejich používání je jejich podpora na straně serveru. Konstruktor se ale programovací jazyk musí naučit, a přestože se jedná o cestu jednodušší než je tomu při využívání přímých
- CGI** programů v atributu `ACTION` (tzv. skriptů CGI, Common Gateway Interface, v příkladu `uloz_to`), jedná se o programování, které předpokládá znalosti a schopnosti schématicky popsané v kapitole Base Of tohoto textu.
- CSS** **Kaskádové styly** (CSS, Cascading Style Sheets) slouží pro rozšíření práce s HTML. Pomocí CSS lze předdefinovat atributy používaného tagu, a to v kaskádovitě vymezené oblasti celého dokumentu. Např. použijí-li tag `TD` při určení buňky tabulky, může být již předdefinována barva pozadí a font textu této buňky. Soubor s definovanými styly mívá příponu `.css` a autor jej zadává v oblasti určené tagem `HEAD` stránky, a to nepárovým tagem `LINK`, např.:

```
<LINK href='kokolia.css' rel=stylesheet type=text/css>
```

- JavaScript** **Skriptovací jazyky** představují ještě další rozšíření práce s HTML. Skriptovacím jazykem je např. JavaScript, který vychází z definice jazyka Java. U skriptovacích jazyků se jedná o ovládání textu stránek dokumentu pomocí programových konstrukcí. Např. pomocí něj dokážeme omezovat uživatele při psaní textu do okénka formuláře tak, že mu nedovolíme psát jiné znaky než alfabetycké nebo naopak pouze číselné. Můžeme testovat umístění kurzoru myši uživatele a podle toho proměňovat chování dokumentu. A podobně.

Správně by měl každý autor dbát na oznámení verze jazyka HTML, ve které je stránka kódována. Přesto se tento rys v běžné praxi takřka nepoužívá, protože autor většinou neví, jako verzi použil. Např.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
```

jako první řádek textu stránky (ještě před tagem `HTML`) stanovuje odpovídající verzi. Více viz www.w3.org a validator.w3.org.

- značkovací jazyky** **Značkovací jazyky** jsou definovaný způsob prokládání sdělovaného materiálu vyhrazenými značkami ve významu formátu, který formálně zpřesňuje obsah. Značkovací jazyky patří mezi metajazyky. Nejznámější je HTML, který byl definován pro elektronické publikování v Internetu. HTML byl odvozen ze standardu SGML (jehož předchůdcem je GML) a později definován i v XML. XML je podmnožinou SGML. SGML a XML jsou definiční jazyky pro jazyky značkovací (metajazyky značkovacích jazyků).

Značkovací jazyky se používají nejenom pro elektronické publikování, ale obecně slouží pro vzájemnou výměnu dat.

Definiční značkovací jazyky:

- GML (Generalize Markup Language), vznik 1960 IBM
- SGML (Standard Generalize Markup Language), ISO 8879:1986
- XML (Extensible Markup Language), 1998 W3C

Publikační značkovací jazyky:

- HTML (Hypertext Markup Language), ISO/IEC 15445/2000
- XHTML (Extensible Hypertext Markup Language), vznik 2000 W3C

Další značkovací jazyky na bázi XML:

- SOAP (Simple Object Access Protocol), vznik 1998 W3C, standardizovaný protokol předávání zpráv typu XML,
- RSS (Rich Site Summary nebo Really Simple Syndication), vznik 1996 Netscape, způsob udržování konzistence odkazů, používá se ve websites a weblogs,
- XML-RPC (XML-Remote Procedure Call), vznik 1995 Microsoft, síťový programovací jazyk implementovaný na bázi XML.

Editor vi má svůj původ v prostředí operačního systému UNIX. Původní verzi **vi** naprogramoval Bill Joy v jedné z prvních univerzitních verzí operačního systému UNIX s označením BSD na UCB (University of California, Berkeley) okolo roku 1976 v rámci snahy o vytvoření jednotného způsobu editace prostých textů na různých typech alfanumerických terminálů (viz Enjoy It III.2), jejichž technický způsob ovládání obrazovky se různil. Editor **vi** byl určen programátorům pro editaci zdrojových textů programů. Uživatelé je pomocí něho umožněna editace obsahu souboru jako prostého textu. Všechny znaky jsou totiž zobrazovány dle definice ASCII (viz Příloha A, sloupec s označením klávesnice), výjimkou jsou pouze znak tabulátor, který **vi** zobrazí v nahrazení několika mezer, a dva po sobě následující znaky CR a LF, které zobrazuje tak, že text přeruší a pokračuje na novém řádku. **vi** pracuje obvykle pouze v ASCII od 0 do 127, k editaci horní části ASCII jsou používány tzv. binární editory. **vi** ale také bývá upraven pro editaci v odpovídajícím národním prostředí, takže je možná editace i znaků od 128 do 255 dle odpovídající konvence národní abecedy. Editace souboru probíhá náhradou jednotlivých znaků dle ASCII za jiné, editor nekládá ani neubírá žádné znaky, které nejsou zobrazeny.

vi přežil dnes již bezmála půl století především díky svému mocnému aparátu práce s editovaným textem, jako je např. využívání regulárních výrazů pro účely hromadných změn, ale také garancí obsahu souboru zobrazeného uživateli. Jak je v UNIXu zvykem, **vi** může využívat kterýkoliv nástroj (tool) UNIXu tak, že jím obsah souboru nechá zpracovat a uživatel pokračuje v editaci s výsledným textem.

vi je součástí řádkového editoru **ex**. Předchůdcem editoru **ex** je řádkový editor **ed**. Slovo řádkový je používáno ve smyslu řádkově orientované komunikace zadávání příkazů v textovém prostředí (viz opět Enjoy It III.2), kdy uživatel zapisuje příkaz na klávesnici a odesílá jej ke zpracování stiskem klávesy Enter. Počítač jeho příkaz zpracuje a případný výsledek vypíše jako textové pokračování na řádcích za zadaným příkazem. Uživatel po ukončení příkazu zadá na novém řádku další příkaz. Editor **vi** je součástí **ex** jako možnost využívat při editaci nejenom jeden řádek, ale celou obrazovku, odtud zkratka **vi** pro visual.

ed, ex, vi

V UNIXu je **ex** aktivován nejlépe v prostředí shellu řádkovým příkazem (např. v prostředí okna aplikace **xterm**)

```
$ ex jméno_souboru
:
```

a uživateli se ohlásí výpisem znaku dvojtečka na následujícím řádku (znak **\$** vypisuje shell, aby oznámil svoji připravenost). Pokud řádkově orientované příkazy editoru **ex** známe, můžeme je postupně zadávat jejich zápisem za dvojtečku a potvrzením klávesou Enter. Jedním z příkazů editoru **ex** je příkaz **vi**

```
: vi
```

kdy aktivujeme obrazovkový režim editoru **ex**, editor **vi**.

Z obrazovkově orientovaného režimu `vi` aktivujeme řádkovou komunikaci s `ex` stiskem klávesy se znakem dvojtečka, který je žádostí o zadání jednoho příkazu pro `ex`.

Vstup přímo do režimu `vi` z shellu v UNIXu můžeme zadat samostatným příkazem takto

```
$ vi jméno_souboru
```

Při používání editoru `vi` rozlišujeme dva základní režimy. Je to režim zobrazování a režim vkládání.

**h, j, k, l,
Ctrl-f, Ctrl-b**

Režim zobrazování je výchozí při aktivaci editoru a projevuje se tak, že si editovaný soubor můžeme pouze prohlížet. Pro prohlížení souboru používáme klávesy s šipkami (přesouváme se o znak vpravo nebo vlevo, o řádek dolů nebo nahoru), klávesy `PgDn` (o stranu vpřed) a `PgUp` (o stranu zpět), případně klávesy `Home` (začátek souboru) a `End` (konec souboru). Vzhledem k tomu, že `vi` vznikl dříve než byly klávesnice terminálů vybaveny uvedenými klávesami (!), šipky dodnes alternují klávesy `h`, `j`, `k`, `l` a klávesy `PgDn`, `PgUp` kombinace kláves `Ctrl-f` a `Ctrl-b`.

Pro zobrazování obsahu souboru je k dispozici řada dalších kláves, např.:

<code>\$</code>	přesun na konec řádku,
<code>^</code>	přesun na začátek řádku,
<code>w</code>	přesun na následující slovo,
<code>b</code>	přesun na předchozí slovo,
<code>+</code>	přesun na první znak následujícího řádku,
<code>-</code>	na první znak předchozího řádku,
<code>(</code>	začátek věty,
<code>)</code>	začátek následující věty,
<code>{</code>	začátek odstavce,
<code>}</code>	začátek následujícího odstavce,
celé číslo	následované stiskem <code>G</code> přesune kurzor na odpovídající řádek pořadí v souboru,
<code>\$G</code>	přesune kurzor na konec souboru.

Režim zobrazování si můžeme představit tak, jako kdyby obsah editovaného souboru byl zobrazován za sklem a my si jej mohli pouze prohlížet.

i, I, a, A, o, O

Pro doplňování textu do souboru musíme vstoupit do režimu vkládání, a to stiskem klávesy `i` (zde ve významu insert), kdy začínáme dále psaný text na klávesnici vkládat před znak s kurzorem, nebo klávesu `a` (append), kdy dále vkládáme text za znak s kurzorem. Použijeme-li znak `I`, budeme vkládat text před první znak řádku, na němž se kurzorem nacházíme, `A` za poslední znak tohoto řádku. Přidání nového řádku, do něhož budeme vkládat nový text, tedy provedeme stiskem `A` a pak klávesou `Enter`, můžeme ale také stisknout pouze klávesu se znakem `o` (open), klávesou `o` zkráceně dosáhneme téhož jako při `I`, `Enter` a šipka nahoru.

Po stisku jedné z kláves i, a, I, A, o, O je text psaný z klávesnice vkládán do dříve kurzorem vybraného místa souboru. Režim editace ukončíme stiskem klávesy Esc a přejdeme tak zpět do režimu zobrazování. Poté se přesuneme na jiné místo, kam potřebujeme vložit nový text a opět použijeme jednu z kláves i, a, I, A, o, O.

Přímo v režimu zobrazování můžeme smazat znak, na kterém je umístěn kurzor, stiskem klávesy se znakem x. Smazat jeden řádek, na kterém je umístěn kurzor, můžeme stiskem klávesy d dvakrát po sobě (tedy dd). Zrušit zbytek řádku od umístění kurzoru můžeme stiskem D (nebo dš), zrušit slovo pomocí dw.

x, dd, D, dw

Kopii řádku, na kterém je umístěn kurzor, provedeme stiskem klávesy y, což je pouze výběr řádku ke kopii, kurzor přesuneme na místo, kam požadujeme řádek okopírovat a stiskem klávesy p nebo P označený řádek kopírujeme za nebo před řádek s kurzorem. Namísto klávesy y můžeme dvakrát stisknout klávesu Y (yy).

Y, yy, p, P

Změny textu souboru můžeme dosáhnout jistě tak, že nejprve text, který požadujeme změnit, odstraníme (pomocí x a dd) a pak použijeme režim vkládání. Jednodušší cestou je však použít v režimu zobrazování klávesy změn s (substitute), s, c (change) a c. Pro změnu znaku za jiný text použijeme klávesu s, pro změnu slova textu za jiný text kombinaci cw, pro změnu celého řádku cc (nebo s), pro změnu od místa s kurzorem do konce řádku c. Editor upozorní na rozsah místa, které se bude měnit, umístěním kurzoru na jeho začátek a doplněním znaku \$ na jeho konec. Výměnu textu ukončíme stiskem klávesy Esc.

s, S, c, C

Jeden znak přímo v režimu zobrazování vyměníme za jiný použitím klávesy r s následným stiskem klávesy s požadovaným znakem. Pomocí R dále přepisujeme zbytek řádku do chvíle stisku klávesy Esc.

r, R

Malé písmeno za velké a opačně vyměníme přímo v režimu zobrazování stiskem klávesy ~.

~

Uvedené klávesy pro editaci změn mohou předcházet určení oblasti, ke které jsou změny vztaženy. Tímto určením je číselná hodnota, kterou při uživatelské zapisování editor nikam na obrazovce neuvádí. Číselná hodnota uvádí počet opakování odpovídajícího editačního úkonu. Např. zápisem 10x smažeme 10 znaků od místa kurzoru, 2dd smažeme dva řádky atp. Podrobnosti je možné najít v dokumentaci v každém UNIXu nebo v [Skočovský1993].

Jakoukoliv provedenou editaci textu vrátíme stiskem klávesy u (undo). Opakovaným stiskem u se v editaci vracíme a obnovujeme text postupnou eliminací provedených změn.

u

Vyhledat určitý text v editovaném textu a přesunout se na jeho první výskyt kurzorem můžeme v režimu zobrazování stiskem klávesy / následovaným zápisem požadovaného textu. Editor po potvrzení klávesou Enter hledá text od umístění kurzoru směrem od začátku obsahu souboru k jeho konci (dolů). Použijeme-li ? namísto /, editor bude prohledávat soubor od konce k začátku (směrem nahoru). Po stisku / nebo ? se tento znak napíše na poslední

/, ?, n, N

řádek na obrazovce a my zapisujeme požadovaný hledaný text na tomto řádku. Po nalezení prvního výskytu uvedeného vzorku můžeme pokračovat vyhledáním dalšího stiskem klávesy `n` (next) nebo stiskem klávesy `N` hledáme výskyt předchozího. Editor nekončí prohledávání souboru dosažením jeho konce (v případě prohledávání vpřed) nebo začátku (zpět), ale pokračuje opět od začátku nebo konce.

Editovaný text je textem souboru se jménem, který jsme uvedli v příkazovém řádku. Nový text můžeme pořizovat a editovat také bez takového uvedení jména souboru. Slouží k tomu prosté uvedení

```
$ vi
```

Před ukončení editace text ale musíme s nějakým jménem souboru spojit. To provedeme příkazem editoru `ex` (ve `vi` stiskem klávesy `:`), kterým je příkaz `w`, např.

```
: w pokus.txt
```

ukládá právě pořízený text do souboru se jménem `pokus.txt` v pracovním adresáři editoru `vi` (to je ten, na který jsme byli v shellu nastaveni, když jsme `vi` spustili).

Editor `ex` ukončíme příkazem `q` takto

```
: q
```

Pokud jsme provedené změny v textu nepřepsali do souboru na disku, `ex` odmítne skončit a upozorní nás na možnou ztrátu změn v textu. Pokud jsme si vědomi této ztráty a vyžadujeme ji, provedeme to příkazem

```
: q!
```

Uložení změn do souboru a současně ukončení editoru můžeme provést příkazem

```
: wq
```

nebo jeho alternativ

```
: x
```

Přímo z editoru `vi` bez nutnosti zadávání příkazu `wq` editoru `ex` můžeme ukončit editaci textu s uložením do souboru stiskem klávesy `z` dvakrát za sebou (tedy `zz`).

Editovaný text je spojen s určitým souborem, jehož jméno a umístění na disku buďto zadáváme shellu v okamžiku spouštění editoru a nebo toto jméno zadáváme teprve v příkazu `w`. Informace o propojení editovaného textu se souborem získáme příkazem `f` (file) pro `ex` takto

```
: f
```

Ukončit editaci textu jednoho souboru a zahájit editaci souboru jiného bez ukončení editoru můžeme příkazem `e`.

Můžeme editovat několik textů ukládaných v různých souborech současně. Pro střídání editace jednotlivých souborů používáme příkaz `n` (next) editoru `ex`

```
: n
```

Uváděný stisk klávesy : je pokynem pro `vi` k aktivaci řádkového režimu editoru `ex`. Řádkový režim `ex` má možnosti, které uživatelé po vzniku části `vi` většinou přestali používat, možná ke své škodě. Znak : editor opíše na dolním řádku obrazovky a na zbytek tohoto řádku uživatel zapisuje příkaz podobně jak bylo již uvedeno u `w`, `q`, `f` nebo `n`. Typickým řádkovým příkazem v `ex` je `s` (substitute), pomocí něhož provedeme opakovanou výměnu textu. Např.

```
: 1,50s/jeden/dva/g
```

vymění text `jeden` za `dva`, a to v prvních 50 řádcích editovaného textu souboru. Znaky `/` zde slouží jako oddělovače starého a nového textu. Výměna se přitom provede ve všech výskytech na každém řádku, což je požadováno doplňkem příkazu `g` (global). Pokud by `g` nebyl uveden, výměna by se uskutečnila na každém z 50 řádků vždy nejvýše jednou, a to při prvním nalezení hledaného textu na řádku.

Adresace před příkazem (zde `1,50` u příkazu `s`) může být vyjádřena buďto jedním číslem, kdy se vztahuje k jednomu řádku nebo ke dvěma řádkům odděleným čárkou, což určuje rozpětí platnosti uvedenými čísly řádků (oblast několika řádků). `$` můžeme použít k označení posledního řádku souboru, `.` označuje současný řádek souboru, `+` je následující řádek, `-` předchozí. Kopii řádků docílíme použitím příkazu `t`, přesun pomocí příkazu `m` např.

```
: .t.
```

okopíruje současný řádek. Zadáním

```
: 1,$ t $
```

zdvojíme editovaný text.

Editovaný text může `vi` ukládat na disk do souboru v šifrované podobě a uživatel tak může chránit své texty před jejich zneužitím. K šifrování obsahu souboru používáme příkaz `x`, kdy jsme požádáni o heslo, které při novém otevírání editorem musíme znát. Šifrování je v operačním systému UNIX podporováno na různých úrovních (viz závěrečná část Enjoy It) a šifrování ve `vi` jim odpovídá. S šifrovaným obsahem souboru totiž můžeme (a potřebujeme) manipulovat i mimo jeho editaci při jeho zpracování v jiných aplikacích.

S vazbou editoru na operační systém (mateřský u `vi` je UNIX) souvisí také schopnost spouštět aplikace operačního systému z prostředí editoru a využívat je k samotné editaci. Např. lze ve `vi` příkazem `!` aktivovat přístup k Tools UNIXu. Zájemci mohou najít podrobnější popis a další informace v [Skočovský1993], [Skočovský1998] a dokumentaci editoru `vi` a `ex` v operačním systému UNIX. Její začátek lze získat v příkazovém režimu shellu (v okně `xterm`) pomocí

```
$ man vi
```


Klávesové zkratky umožňují uživateli ovládat aplikace v systému oken namísto používání myši, přestože to není možné ve zcela všech případech. Jsou uživateli oblíbeny, protože jejich použití urychluje ovládání. Klávesová zkratka je takřka ve všech případech kombinací stisku několika (obvykle dvou) kláves, které postupně tiskneme a držíme stisklé v uváděném pořadí zleva doprava. Klávesová zkratka je vyjádřena přidáním poslední klávesy v pořadí a uvolněním všech kláves najednou. V následujících tabulkách uvádíme jednotlivé klávesy klávesových zkratk oddělené čárkou pro nejrozšířenější systémy oken. Vždy se jedná o citaci odpovídající dokumentace a zkušenost autora tohoto textu.

Klávesové zkratky MS Windows

1. Obecné klávesové zkratky

klávesy	význam
Ctrl, c	kopírování
Ctrl, x	vyjmutí
Ctrl, v	vložení
Ctrl, z	zpět
Delete	odstranění
Shift, Delete	odstranění s vynecháním koše
Ctrl, tah myši	kopírování
Ctrl, Shift, tah myši	vytvoření zástupce
F2	přejmenování
Ctrl, →	na začátek následujícího slova
Ctrl, ←	na začátek předchozího slova
Ctrl, ↓	na začátek následujícího odstavce
Ctrl, ↑	na začátek předchozího odstavce
Ctrl, Shift, šipka	označení části textu

klávesy	význam
Shift, šipka	přidání k označenému
Ctrl, a	označí vše
F3	vyhledávání
Alt, Enter	vlastnosti
Alt, F4	uzavření grafického objektu
Alt, mezera	menu aktivního okna
Alt, Tab	přepínání mezi aktivními objekty
Alt, Esc	přepínání mezi aktivními objekty v pořadí jejich aktivace
F6	přepínání mezi objekty
F4	orientace v průzkumníku
Shift, F10	menu možností daného objektu (náhrada za pravé tlačítko myši)
Ctrl, Esc	menu Start
Alt, písmeno	možnost z menu odpovídající podrženému písmenu
F10	panel menu programu aktivního okna
→	další menu
←	předchozí menu
F5	refresh, aktualizace obsahu okna
Esc	zrušení akce
trvalý stisk Shift	zabrání automatickému přehrání vloženého CD
Alt, PrntScrn	snímek aktivního okna (pomocí Ctrl, v ho vložíme např. do dokumentu)

klávesy	význam
Ctrl, Alt, PrntScrn	snímek obrazovky

2. Klávesové zkratky pro dialogová okna

klávesy	význam
Ctrl, Tab	následující karta
Ctrl, Shift, Tab	předchozí karta
Tab	další
Shift, Tab	předchozí
Alt, písmeno	možnost odpovídající podrženému písmenu
Enter	provedení
mezera	zaškrtnutí
šipky	výběr přepínače
F1	nápověda
F4	menu aktivního seznamu
Backspace	předchozí úroveň dialogu

3. Klávesové zkratky při používání klávesnice typu Natural Keyboard

Klávesnicí typu Natural Keyboard je myšlena klávesnice, která je rozšířena o klávesu letícího okna (v tabulce označujeme jako W), které je logem MS Windows a o klávesu místního menu, což je klávesa s obrázkem menu s šipkou kurzoru (v tabulce jako M). V současné době každé prodané PC s MS Windows tyto klávesy obsahuje.

klávesy	význam
W	menu Start
W, Break	vlastnosti systému

klávesy	význam
W, d	plocha
W, m	minimalizace všech oken
W, Shift, m	obnovení minimalizovaných oken
W, e	průzkumník
W, f	hledání
Ctrl, W, f	hledání uzlu v síti
W, F1	nápověda systému oken
W, l	uzamknutí sezení, přepínání uživatelů
W, r	dialog spuštění aplikace
M	místní menu
W, u	správce nástrojů

4. Průzkumník MS Windows

klávesy	význam
End	přesun na konec
Home	přesun na začátek
NumLock, *	všechny podsložky
NumLock, +	rozbalení složky
NumLock, -	sbalení složky
←	sbalení složky nebo nadřazená složka
→	rozbalení podsložky nebo první podsložka

Klávesové zkratky operačních systémů Apple

se v novém systému Mac OS X mírně liší od předchozích verzí. Mac OS X je UNIX a grafický podsystém je tedy programovaný v X (viz také KDE v Linuxu, které následuje). V nižších verzích bylo také možné klávesové zkratky editovat pomocí programu ResEdit.

Klávesa Command v následujících seznamech je klávesa s jablkem.

1. V průběhu startu operačního systému

dlouhý stisk klávesy	význam
x	start operačního systému Mac OS X
Option, Command, Shift, Delete	zavedení alternativního operačního systému (z externího disku, CD ...)
c	zavedení alternativního operačního systému explicitně z CD
n	zavedení alternativního operačního systému ze síťového serveru (NetBoot)
r	Power Book screen reset
t	start operačního systému z disku připojeného pomocí FireWire
Shift	start operačního systému v režimu Safe Boot
Command, v	komentovaný výpis průběhu startu operačního systému
Command, s	operační systém bude startován v jednouživatelském režimu

2. V okně vyhledávače

klávesy	význam
Command, w	uzavření okna
Option, Command, w	uzavření všech oken
Command, →	rozbalení složky
Option, Command, →	rozbalení podstromu složky

klávesy	význam
Command, ←	sbalení složky
Option, Command, ↑	otevření nadřazené složky a uzavření okna

3. V menu

menu	klávesy	význam
hlavní (Apple Menu)	Shift, Command, q	odhlášení
	Shift, Option, Command, q	okamžité odhlášení
vyhledávání (Finder Menu)	Shift, Command, Delete	vysypání koše
	Option, Shift, Command, Delete	vysypání koše bez dotazu na uživatele
	Command, h	skrytí vyhledávače
	Option, Command, h	skrytí ostatního
soubor (File Menu)	Command, n	nové okno prohlížeče
	Shift, Command, n	nová složka
	Command, o	otevření
	Command, s	uložení
	Shift, Command, s	uložení jiným způsobem (save as)
	Command, p	tisk
	Command, w	uzavření okna
	Option, Command, w	uzavření všech oken
	Command, i	nápověda
	Option, Command, i	zobrazení atributů
	Command, d	vytvoření kopie

menu	klávesy	význam
	Command, l	vytvoření zástupce
	Command, r	zobrazí původní
	Command, t	přidá k oblíbeným
	Command, Delete	odstraní (do koše)
	Command, e	vysunutí
	Command, f	hledání
editace (Edit Menu)	Command, z	o krok zpět
	Command, x	vyjmout
	Command, c	kopírovat
	Command, v	vložit
	Command, a	označit vše
zobrazení (View Menu)	Command, 1	jako ikony
	Command, 2	jako seznam
	Command, 3	ve sloupcích
	Command, b	skryj lištu
	Command, j	zobrazí všechny možnosti
navigace (Go Menu)	Command, [zpět
	Command,]	vpřed
	Shift, Command, c	počítač
	Shift, Command, h	na začátek (Home)
	Shift, Command, i	iDisk

menu	klávesy	význam
	Shift,Command, a	aplikace
	Shift,Command, f	oblíbené
	Shift,Command, g	složka přejdi na (GoTo Folder)
	Command, k	připojení k serveru
okna (Window Menu)	Command, m	minimalizace okna
	Option,Command, m	minimalizace všech oken
	Command, ?	hlavní nápověda

4. Obecné klávesové zkratky

klávesy	význam
Option, Command, *	aktivace přibližování a vzdalování (zoom)
Option, Command, +	přiblížení (zoom in)
Option, Command, -	vzdálení (zoom out)
Control, Option, Command, *	inverzní zobrazení
Control, F1	zapnutí funkčních kláves (Full Keyboard Access), je-li zapnuto, je možné klávesové zkratky zbytku tabulky používat z aplikace vyhledávače (Finder)
Control, F2	vysunutí menu
Control, F3	vysunutí již použitého
Control, F4	přesun (aktivního nebo prvního zakrytého) okna do popředí
Control, F5	vysunutí lišty nástrojů
Control, F6	vysunutí nabídky aplikací

5. Myš

Panel obecného nastavení uživatelského přístupu umožňuje zapnout používání kláves namísto myši (Mouse keys). Myš je pak zastoupena klávesnicí a uživatel používá její číselnou část (pokud ji nemá, používá klávesy s předřazením klávesy Fn) s následujícím významem:

klávesy	význam
8	pohyb nahoru
2	pohyb dolů
4	pohyb doleva
6	pohyb doprava
1 nebo 3 nebo 7 nebo 9	diagonální pohyb
5	stisk tlačítka myši
0	trvalý stisk tlačítka myši
.	uvolnění trvalého stisku tlačítka myši (po stisku klávesy s 0)

6. Ostatní

klávesy	význam
Option, Command, d	vysunutí nebo skrytí nabídky použitých možností
Command Tab	přepínání mezi aplikacemi
Tab	zaměření pozornosti na další prvek
Command, ↑	posun o adresář zpět
Command, ↓	posun o adresář vpřed
PgUp nebo Ctrl, ↑	posun o stranu zpět
PgDn nebo Ctrl, ↓	posun o stranu vpřed
Option, tah myší	kopie do nového místa
Option, Command, tah myší	zástupce v novém umístění

klávesy	význam
Command, tah myši	přesun do nového místa
Shift, Command, c	paleta barev aplikace
Command, t	paleta fontů aplikace
Command, Shift, 3	snímek obrazovky
Command, Shift, 4	snímek vybrané části obrazovky
Command, Shift, 4 a Control po provedení výběru	snímek obrazovky s uložením k dispozici pro vložení (into Clipboard)
Command, Shift, 4 a mezera	snímek vybraného okna
Option, Command, Esc	násilné ukončení aplikace
Control, Eject	dialog pro restart, spánek, ukončení operačního systému
Control, Command, Eject	ukončení všech aplikací a restart operačního systému
Option, Command, Eject nebo Power	přechod do režimu spánku
Command, kliknutí na tlačítko lišty nástrojů okna (pravý horní roh okna)	postupné střídání možností způsobů zobrazování okna
Command, `	postupné střídání otevřených oken aplikace
Function, Delete (pouze PowerBook)	smazání znaku napravo od kurzoru

Klávesové zkratky KDE v operačním systému Linux

Každý grafický systém oken nejenom v Linuxu, ale v UNIXu obecně, je programován v prostředí X. Každý nově programovaný grafický podsystém může ovšem mít své vlastní klávesové zkratky. KDE podobně jako i další systémy oken v UNIXu se podřizují obecnému doporučení dle stavby

a ovládání X. Tato doporučení jsou určena především programátorům nových aplikací v UNIXu, ale i pokročilým uživatelům, kteří umí definici klávesových zkratk přenastavit.

Pro začátečníky je důležité vědět, že klávesové zkratky aplikací v UNIXu dodržují konvenci klávesových zkratk v notaci \wedge , písmeno (Ctrl, písmeno).

Pokud má klávesová zkratka význam zapnutí určité vlastnosti, adekvátní zkratka s \wedge , velké písmeno tuto vlastnost vypíná.

V aplikacích se nepoužívají kombinace kláves s klávesou Alt. Ta je rezervována pro ovládání pracovní plochy.

Obvykle se nepoužívají znaky, které vyžadují použití klávesy Shift (např. kombinace \wedge , %), a to z důvodu komplikací pro uživatele národnostních klávesnic.

Klávesové zkratky se nepřekládají při převodu aplikací do národních jazykových mutací. Např. \wedge , o bude klávesová zkratka pro otevření souboru i v případě, že slovo open je do jiného jazyka překládáno s jiným začínajícím písmenem.

Pro aplikace jsou doporučeny (a používány) následující klávesové zkratky:

1. File menu (menu Soubor) hlavní lišty aplikace

klávesy	význam
\wedge , n	nový (např. nový dokument)
\wedge , o	otevřít (např. otevřít existující dokument)
\wedge , s nebo \wedge , Shift, s	uložit (uložit provedené změny otevřeného dokumentu)
F5	opakuji načtení (např. přečti mi obsah dokumentu z disku, zapomeň co je v operační paměti)
\wedge , p	tisk
\wedge , w	uzavři (např. uzavři dokument)
\wedge , q	ukončí aplikaci

2. Edit menu (menu Úpravy hlavní lišty aplikace).

klávesy	význam
\wedge , z	o editační krok zpět (Undo)
\wedge , y nebo Shift, \wedge , z	o editační krok vpřed (Redo)

klávesy	význam
^, x	vyjmutí označené oblasti (cut and store in the Clipboard)
^, c	kopie označené oblasti
^, v	vložení do místa kurzoru (paste content of Clipboard)
^, a	označí vše
^, Shift, a	zruší označení všeho
^, f	hledání
F3	hledej následující výskyt
Shift, F3	hledej předchozí výskyt
^, r	hledej a vyměň
^, +	přiblížení (zoom in)
^, -	vzdálení (zoom out)
^, b	přidej k záložkám
F1	přímá nápověda (help)
mezerník	zobrazení následující stránky
Shift, mezerník	zobrazení předchozí stránky

3. Rezervované klávesové zkratky

jsou uvedeny v následující tabulce, a to z důvodu jejich používání v uvedeném významu v již existujících aplikacích operačního systému UNIX (které byly používány s klávesnicemi bez v tabulce uvedených kláves).

klávesy	význam
^, a	ekvivalent klávesy Home
^, e	ekvivalent klávesy End
^, h	ekvivalent klávesy Backspace
^, k	odstranění textu od kurzoru do konce řádku
^, u	odstranění textu od začátku řádku do umístění kurzoru

SQL (Structured Query Language) je nejrozšířenější dotazovací databázový jazyk. Jeho znalost zajišťuje uživateli poměrně jednoduchý přístup k datům, je ale určen programátorům, kteří jej používají pro konstrukci přístupových cest k datům a nabízejí je tak uživatelům pro různé manipulace podle jejich způsobu oborového myšlení. Přestože je SQL již poměrně podrobně standardizován (viz [ISOSQL2003]), při práci s ním zvláště programátoři často používají jeho části, které ve standardu obsaženy nejsou. Následující přehled dotazů je pouze základním seznamem a jejich zde uváděná podoba striktně podléhá standardu. V dále uvedeném tvaru jsou zcela použitelné v libovolném databázovém stroji, který standard splňuje.

Každá databáze disponuje řádkově orientovaným přístupem. V oknově orientovaných operačních systémech se jedná o aplikaci, která v okně umožní řádkovou komunikaci s obsahem databáze. Řádkově orientovaná komunikace znamená, že uživatel na řádku píše některý z dotazů a teprve po stisknutí klávesy Enter je tento dotaz odeslán databázovému stroji. Databázový stroj dotaz zpracuje a výsledky vypíše na několika řádcích za text uživatelského dotazu. Uživatel píše dotaz na základě výzvy ke komunikaci, který nazýváme prompt, např.:

SQL>

Uživatel píše na řádek dotaz a odesílá jej klávesou Enter. Dotaz je formálně ale ukončen znakem ; (středník), který uživatel napíše před stiskem klávesy Enter. Uživatel ale může pokračovat v zápisu dotazu na několika řádcích a dokud nenapíše znak ; následovaný klávesou Enter, dotaz není databázovým strojem zpracován. Např.

```
SQL> SELECT * FROM PRACOVIŠTĚ;
ID_PRACOVIŠTĚ  NÁZEV
```

```
-----
1              kotelna
2              výpočetní středisko
3              ústředí
```

(tučně psaný text je odezvou databázového stroje).

V dalším textu uváděné formální definice používají obvyklý definiční metajazyk. Význam použitých symbolů je následující:

- velkými písmeny uvádíme text jazyka SQL, text malými písmeny uživatel substituuje jménem tabulky, sloupce, uvedením konkrétního vztahu atp.,
- část uvedená v hranatých závorkách [a] je nepovinná,
- část uvedená ve složených závorkách { a } je jedna z uvedených variant, ty jsou odděleny znakem |.

Základní dotazy SQL jsou:

```
SELECT  pro výpis dat z databáze,
INSERT  pro vložení záznamu do databáze,
DELETE  pro odstranění dat z databáze,
UPDATE  pro změnu dat v databázi.
```

SELECT Dotaz SELECT má formální definici

```
SELECT sloupec, sloupec...
FROM tabulka, tabulka...
[WHERE podmínka]
[GROUP BY výstup, výstup...]
[HAVING podmínka]
[ { UNION | UNION ALL | INTERSECT | MINUS } SELECT ... ]
[ORDER BY kritérium_řádění]
```

Např.

```
SQL> SELECT NÁZEV FROM PRACOVISTĚ;
NÁZEV
-----
kotelna
výpočetní středisko
ústředí
SQL> SELECT NÁZEV FROM PRACOVISTĚ WHERE ID_PRACOVISTĚ > 2;
NÁZEV
-----
ústředí
```

byl proveden výpis názvů (položka `NÁZEV`) všech záznamů tabulky `PRACOVISTĚ` a všech záznamů tabulky `PRACOVISTĚ`, jejichž položka `ID_PRACOVISTĚ` (jednoznačná číselná identifikace záznamu) je větší než 2.

INSERT Dotaz INSERT vloží nový záznam do databáze, má formální definici

```
INSERT INTO tabulka
[ (sloupec [, sloupec] [, ...] ) ]
{ SELECT... | VALUES (hodnota [, hodnota] [, hodnota] ... ) }
```

Např.

```
SQL> INSERT INTO PRACOVISTĚ (ID_PRACOVISTĚ, NÁZEV) VALUES (4, 'obchodní oddělení');
```

SQL>

je totéž jako

```
SQL> INSERT INTO PRACOVISTĚ VALUES (4, 'obchodní oddělení');
```

a dotazem SELECT můžeme prověřit, zda se stalo, co jsme chtěli:

```
SQL> SELECT * FROM PRACOVISTĚ;
ID_PRACOVISTĚ  NÁZEV
```

```
-----
1              kotelna
2              výpočetní středisko
3              ústředí
4              obchodní oddělení
```

Dotaz DELETE odstraní data z databáze, formální definice je

DELETE

DELETE FROM *tabulka*

[WHERE *podmínka*]

Např.

SQL> DELETE FROM PRACOVIŠTĚ WHERE ID_PRACOVIŠTĚ=4;

odstraní každý záznam, jehož ID_PRACOVIŠTĚ je 4, ale bez uvedení podmínky

SQL> DELETE FROM PRACOVIŠTĚ;

odstraní všechny záznamy tabulky PRACOVIŠTĚ (!).

Dotaz UPDATE mění obsah položek záznamů, formální definice je

UPDATE

UPDATE *tabulka*

SET *sloupec=hodnota* [, *sloupec=hodnota*] [, *sloupec=hodnota*]...

[WHERE *podmínka*]

Např.

SQL> UPDATE PRACOVIŠTĚ SET NÁZEV='ústředí organizace' WHERE ID_PRACOVIŠTĚ=3;
nebo

SQL> UPDATE PRACOVIŠTĚ SET NÁZEV='ústředí organizace' WHERE NÁZEV='ústředí';

Při práci s definováním datového modelu používáme dotazy pro vytváření tabulek, jejich odstraňování, doplňování atd. Novou tabulku vytvoříme pomocí dotazu CREATE, tabulku zrušíme pomocí DROP. Oba dotazy se ale používají také na další prvky struktury datového modelu, jakým je např. index, uživatel, pohled aj.

Dotaz CREATE vytvoří novou strukturu dat (např. tabulku), formální definice pro vytvoření tabulky je

CREATE

CREATE TABLE *tabulka*

(*sloupec typ* [*omezení_sloupce*] [, *sloupec typ* [*omezení_sloupce*]]...)

omezení_tabulky

Např.

SQL> CREATE TABLE PRACOVIŠTĚ

(ID_PRACOVIŠTĚ DECIMAL(2,0) PRIMARY KEY, NÁZEV CHAR(20));

Standardizované datové typy, které musíme uvádět při definici položek tabulky, jsou především číselné (obsahují číselné údaje), znakové (obecné texty) a časové (datum a čas).

Číselné datové typy jsou:

INTEGER	položka bude obsahovat celá čísla (kladná i záporná),
SMALLINT	pro položky, kdy víme, že se bude jednat o malá celá čísla,
DECIMAL	určuje desetinné číslo, v parametrech definice určujeme celkový počet číslic a počet číslic za desetinnou čárkou, viz uvedený příklad u CREATE,
NUMERIC	je synonymum pro DECIMAL,
FLOAT	desetinné číslo s plovoucí desetinnou čárkou, v parametru je počet platných číslic (nejvýše 38), např. FLOAT(20),

REAL	malé desetinné číslo, v parametru i zde počet platných číslic, nejvýše 18,
BIT	binární data, není číselný typ, jedná se o manipulaci s daty na bitové úrovni, může tedy obsahovat cokoliv (viz Příloha A), v parametru celé číslo, udávající délku v bajtech, obvykle omezenou na 255.

Znakové datové typy:

CHAR	textový řetězec určené max. délky (nejvýše 255), v parametru uvedena délka,
VARCHAR	text proměnlivé délky, v parametru se uvádí max. délka, např. VARCHAR (2000).

Datové typy pro datum a čas:

DATE	položka bude obsahovat údaje o dnu, měsíci a roku, formát není standardizován a často jej lze v konkrétní databázi měnit na jiný,
TIME	čas, údaje o hodině, minutě a vteřině,
TIMESTAMP	datum a čas.

Dotazem `CREATE` vytváříme i jiné struktury dat datového modelu, jako jsou `INDEX`, `SEQUENCE`, `USER`, `ROLE`, `VIEW`, `TRIGGER`.

INDEX `INDEX` Obsah tabulky lze indexovat, tj. záznamy mají přiřazeno své umístění v databázi a jsou takto rychleji dostupné. Kritéria pro umístění stanovujeme ve chvíli definice indexu, a to tak, že určíme, na které sloupce tabulky má být index zaměřen. Používáním indexů se výrazně zvýší čtení (především dotazem `SELECT`) dat z tabulky. Formální definice pro vznik indexu je

```
CREATE [UNIQUE] INDEX index
ON tabulka
(sloupec [, sloupec ...])
```

Index se doporučuje znovu vytvářet vždy po nárůstu počtu záznamů v tabulce (zrušíme jej pomocí `DROP INDEX` a nově vytvoříme pomocí `CREATE INDEX`). Každá tabulka je indexována implicitně podle primárního klíče (viz Enjoy It III.3). Např.

```
SQL> CREATE INDEX IX_PRAC_NÁZEV ON PRACOVISTĚ (NÁZEV);
```

umožní rychlejší čtení, pokud vyhledáváme podle názvu pracoviště.

SEQUENCE `SEQUENCE` definuje sekvenci. Sekvence udržuje naposledy použitou číselnou hodnotu používané číselné řady, např. pro jednoznačné označování postupně vznikajících záznamů v tabulce (počítadlo záznamů). Použitím sekvence v dotazu `INSERT` je sekvenci přiřazena nová hodnota dle její definice. Formát je

```
CREATE SEQUENCE sekvence
[START WITH celé_číslo]
[INCREMENT BY celé_číslo]
[{MINVALUE celé_číslo | NOMINVALUE}]
[{MAXVALUE celé_číslo | NOMAXVALUE}]
[{CYCLE | NOCYCLE}]
```


Např.

SQL> CREATE SEQUENCE SQ_ID_PRACOVIŠTĚ START WITH 1 INCREMENT BY 1 NOCYCLE;
 USER K databázi mohou přistupovat uživatelé různých oprávnění pro čtení, vytváření nebo aktualizaci dat různých částí datového modelu. Výchozí nastavení databáze pracuje s nejméně jedním uživatelem, který má neomezená práva přístupu k databázi. Pomocí vytváření různých uživatelů a pomocí dotazu GRANT lze uživatele omezovat v přístupu k tabulkám, jejich sloupcům, atd. Vytvořit nového uživatele lze příkazem

**USER, GRANT,
REVOKE**

```
CREATE USER uživatel
```

jehož formát není stále striktně standardizován.

Např.

```
SQL> CREATE USER KOTELNÍK;  
SQL> GRANT SELECT ON PRACOVIŠTĚ TO KOTELNÍK;  
SQL> REVOKE DELETE ON PRACOVIŠTĚ FROM KOTELNÍK;  
vzniká uživatel se jménem KOTELNÍK, který má přístup k tabulce PRACOVIŠTĚ  
dotazem SELECT, nemůže ale v tabulce odstraňovat záznamy. Bude-li KOTELNÍK  
zlobit, lze mu odejmout i právo pro SELECT pomocí REVOKE takto  
SQL> REVOKE SELECT ON PRACOVIŠTĚ FROM KOTELNÍK;
```

ROLE Pro vypracování skupin různě silných uživatelů vzhledem k pravidlům přístupových metod k datovému modelu slouží role. Definujeme postupně různé role a pak je pomocí dotazu GRANT přiřazujeme vytvořeným uživatelům (nebo je pomocí REVOKE uživatelům odebíráme). Formát je

ROLE

```
CREATE ROLE role
```

Např.

```
SQL> CREATE ROLE VEDENÍ;  
SQL> GRANT ALL ON PRACOVIŠTĚ TO VEDENÍ;  
SQL> CREATE USER ŘEDITEL;  
SQL> GRANT VEDENI TO ŘEDITEL;
```

VIEW je pohled. Pomocí něj můžeme usnadnit uživateli přístup k datům. Jedná se také o bezpečnostní opatření, protože přístup k původní tabulce nebo tabulkám, nad kterými je pohled vytvořen, uživateli odpojíme a přiřadíme mu pouze přístup k pohledu (i zde pomocí GRANT a REVOKE). Pohled definujeme a vzniká tak zástupné jméno pro virtuální tabulku. Definovat pohled znamená určit, které položky kterých tabulek budou sloučeny v rámci jednoho pohledu VIEW. Formát je

VIEW

```
CREATE VIEW pohled  
[( sloupec [, sloupec [...]] )]  
AS dotaz_select
```

Např.

```
SQL> CREATE VIEW kdo_kde AS  
SELECT ZAMĚSTNANEC.PŘÍJMENÍ, PRACOVIŠTĚ.NÁZEV  
FROM ZAMĚSTNANEC, PRACOVIŠTĚ  
WHERE ZAMĚSTNANEC.ID_PRACOVIŠTĚ=PRACOVIŠTĚ.ID_PRACOVIŠTĚ;
```

a následně můžeme použít

```
SQL> SELECT PŘÍJMENÍ,NÁZEV FROM kdo_kde;
```

TRIGGER TRIGGER je vnitřní procedura v databázi, která je aktivována tehdy, nastane-li v dané tabulce vytvoření nového záznamu, změna záznamu nebo zrušení záznamu. Jedná se již o programování vnitřních procedur, které je věcí nikoliv obyčejných uživatelů. Navíc dosavadní definice vnitřních procedur a programování uvnitř databáze není standardizována a výrazně se liší napříč různými implementacemi SQL.

ALTER Dotaz ALTER mění dříve definovanou strukturu dat, upraví strukturu tabulky, databázového uživatele, sekvence atd. Formální definice pro změnu v tabulce je

```
ALTER TABLE tabulka  
[ ADD (sloupec [, sloupec] [...] ) ]  
[ MODIFY (sloupec [, sloupec] [...] ) ]  
[ DROP { sloupec | CONSTRAINT omezení } ]
```

Např.

```
SQL> ALTER TABLE PRACOVIŠTĚ ADD (POČET_ŽIDLÍ DECIMAL(2,0));
```

Doplnění sloupce nezpůsobí žádnou ztrátu dat v již existujících záznamech. Obsah nových sloupců v již existujících záznamech není dle standardu definován, obvykle ale bývá databázovým strojem naplněn hodnotou NULL (prázdné hodnoty).

DROP Dotaz DROP odstraní dříve definovaná data, např. celou tabulku (zmizí i její obsah), formáty:

```
DROP TABLE tabulka  
DROP INDEX index  
DROP VIEW  
DROP USER uživatel  
DROP ROLE role  
DROP SEQUENCE sekvence
```

Např.

```
SQL> DROP VIEW kdo_kde;
```

```
SQL> DROP TABLE PRACOVIŠTĚ;
```

Standard je v oblasti software IT dokument, který podrobným způsobem popisuje určitou ověřenou a obecnou technologii, jejíž implementace nebo použití v praxi má důležitý význam pro správnou funkci ostatních použitých prostředků a technologií. Standard je normativ, vyjádření obecné shody nad tím, co považujeme již za přirozeně normální.

Standard si může definovat skupina lidí vzájemně mezi sebou nebo určitá organizace, jakou je např. podnik, firma atd., bez vazby na oficiální státní nebo mezinárodní struktury. Problémy však nastávají při vzájemné komunikaci a spolupráci těchto lidských organizací.

I v oblasti IT jsou oficiální standardy definovány na úrovni nadnárodních organizací, jakou je ISO (International Organization for Standardization, založená v r. 1947 a sídlící v Ženevě ve Švýcarsku, iso je akronymem řeckého isos, tj. rovnost). ISO je producentem standardů doporučených celosvětově pro různé činnosti lidí. Jednotlivá světová teritoria ale mohou a nemusí standardy ISO respektovat, a to ať již částečně nebo zcela. V různých teritoriích platí standardy vydávané v rámci jejich politického uspořádání, které mohou také souviset s právním řádem teritoria. To se týká medicíny nebo práce s nebezpečnými chemickými látkami, ale platí to také pro software pro případ škody způsobené nesprávnou implementací, která neodpovídá nařízené normě nebo použitím zakázaných prostředků ke zcizení software, dat nebo jejich záměrného poškození. **ISO**

V IT jsou respektovány standardy ISO, mnohdy jsou ale zpochybňovány dalším vývojem a především rychlým a masivním rozšířením produktů software, jejichž programátoři o existenci odpovídajícího standardu ani nevědí. Pro mnohé produkty, a to zejména programovací jazyky, jsou obecně na celém světě akceptovány standardy ANSI (American National Standards Institute), které jsou závazné pouze na území USA a jiná teritoria je nezahrnují ve svých standardech a nebo definují standardy jiné. Příkladem je používání kódu ASCII (viz Příloha A) a jeho rozšíření na znaky české a slovenské abecedy Československým standardem KOI-8, který nikdo v Čechách a na Slovensku prakticky nerespektoval a používal kódování definované v USA pro naši oblast. **ANSI**

Vyjma oficiálně definovaných a dodržovaných standardů existují také otevřené standardy (Open standard). Otevřenost je ve smyslu možnosti kohokoliv se podílet na jejich doplňování, rozšiřování, případně i provádění oprávněných změn. Příkladem otevřeného standardu je HTML, PDF, TCP, IP a mnoho dalších, např. uvedených formou dokumentu RFC, viz www.rfc-editor.org nebo vůbec na stránkách organizace W3C (World Wide Web Consortium), viz www.w3.org.

V IT jsou standardy důležité také pro zajištění kompatibility nových verzí nebo celých nových software v již digitalizovaných oblastech.

Licence v IT Svolení s používáním vytvořeného digitálního díla dle jeho zaměření. Licenci uděluje majitel díla, obchodník přitom může hrát roli prostředníka. Autorovi díla po jeho vytvoření vzniká nárok na autorské právo, copyright. Auteurské právo může autor poskytnout k dalšímu využití nebo v některých částech světa také prodat.

Ochrana autorského práva, manipulace s ním a přidělování a poskytování licencí (licenční politika) jsou zakotveny v právním řádu každého státního útvaru. Většinou jsou i zde citovány zákony a předpisy USA, ale v konkrétním případě je vždy nutno číst zákony konkrétních zemí (např. České republiky). Mezinárodní právo určitý státní útvar totiž buďto přijímá mezinárodními úmluvami (kterou je např. Evropská unie) a přenáší je do svého právního řádu v systému svých platných psaných zákonů a nebo si je vytváří sám dle svých potřeb. V tomto smyslu jsou i dále uváděné typy licencí výčet aplikace autorského a licenčního práva USA (kolébky a stále předvoje výroby a používání software). V České republice podléhá tato problematika Autorskému zákonu (viz [ZákonČR121/2000]), který je v současné době přizpůsoben obecným ustanovením v rámci Evropské unie a jehož základní praktické dopady dále také uvedeme.

EULA Licence pro software je v USA smluvní vztah mezi producentem a uživatelem vytvořeného software, bývá označována zkratkou EULA (End User Licence Agreement, Licenční smlouva koncového uživatele). V principu jsou používány dva typy licencí, *patentovaná* (proprietary software licence) a *volná* (free software licence), a to v závislosti na tom, zda je zakázáno dále produkt šířit či nikoliv.

OEM Patentovaná licenční smlouva ve svých jednotlivých ujednáních tedy odkazuje na patentové právo, autorské právo, obchodní tajemství (je zakázáno zkoumat principy fungování), ale na druhé straně také záruky ze strany producenta jak na fungování software, tak na případné škody způsobené jeho používáním. Patentová licence je udělována za peníze, může být udělena na použití jedné instalace, ale může být udělena také se souhlasem k více instalacím současně. Je pak označována jako multilicence a pro koncového uživatele je cenově výhodnější než několik samostatných. U patentované licence je zakázáno dále software poskytovat a také do něj zasahovat a přizpůsobovat si jej. Zajímavý typ patentové licence je OEM (original equipment manufacturer), která je poskytována pro produkty spojené např. jednoznačně s konkrétním hardware, např. při nákupu PC poskytne jeho prodejce současně i licenci na používání operačního systému, který je součástí dodávky. Operační systém pak nesmí být používán na jiném PC a záruku za správnou funkci výpočetního systému jako celku nese výrobce hardware (tedy nikoliv výrobce software).

GPL Software, který je označen svým producentem jako podléhající volné licenci, je možné používat koncovým uživatelem bez uzavření odpovídajícího obchodu. Volná licence navíc může mít různé hloubky použití produktu. Hloubkou se rozumí nejenom používání a další šíření, ale i zkoumání principů

fungování a zásahů do těchto principů, tj. úprava zdrojového kódu (source code), který bývá v případě volné licence k dispozici jako součást distribuce. Zdrojový kód může být nezveřejněn (Closed Source Licence) u patentové licence, zveřejněn (Viewable Source Licence) u patentové i volné licence a otevřený (Open Source Licence) pro volnou licenci. Nejhlubší veřejně známá licence má označení GPL (General Public Licence) a dovoluje uživateli produkt kopírovat, dále šířit, upravovat jej a upravený dále šířit. Podmínkou je ale vždy uvedení původního autora, případně dalších, zvykem jsou také komentáře provedených změn. Licence GPL je obecný princip použitý volným sdružením osob kolem aktivit známých jako GNU (zkratka je akronymem GNU's is Not Unix) pro aktivitu volně šířitelného operačního systému Linux. Obecně je takový typ licence nazýván Public domain. Na software typu Public domain není poskytována žádná záruka.

U licence typu Freeware je povoleno volné šíření, ale není dovoleno jej upravovat. Znamená to mimo jiné také to, že si autor ponechává autorské právo (v některé z příštích verzí může licenci změnit na patentovou). U licence typu Freeware není poskytována žádná záruka.

freeware

Konečně jako Shareware je označována licence, která má omezenou platnost. Jedná se o určitou verzi demonstrace plné funkce nabízeného produktu (demo). Produkt se pod touto licencí může volně šířit, nesmí být upravován, může se zdarma využívat pro nekomerční účely a na jeho funkci nejsou poskytovány žádné záruky.

shareware

V České republice je platný Autorský zákon, viz [ZákonČR121/2000]. Z Autorského zákona ČR pro licenci na software (zejména §65 a §66) plyne pro licence především následující:

(Začátek citace)

Počítačový program je dílo, na které se vztahuje autorský zákon, autorem je fyzická osoba, která dílo vytvořila. Počítačový program je chráněn jako dílo literární. Dílem není námět díla sám o sobě, denní zpráva nebo jiný údaj sám o sobě, myšlenka, postup, princip, metoda, objev, vědecká teorie, matematický a obdobný vzorec, statistický graf a podobný předmět sám o sobě. Osobnostních práv se autor nemůže vzdát; tato práva jsou nepřevoditelná a smrtí autora zanikají. Myšlenky a principy, na nichž je založen jakýkoli prvek počítačového programu, včetně těch, které jsou podkladem jeho propojení s jiným programem, nejsou podle tohoto zákona chráněny.

Licenční smlouvou autor poskytuje nabyvateli oprávnění k výkonu práva dílo užít. Do práva autorského nezasahuje oprávněný uživatel rozmnoženiny počítačového programu, jestliže

- a) rozmnožuje, překládá, zpracovává, upravuje či jinak mění počítačový program, je-li to potřebné k užití počítačového programu v souladu s jeho určením, včetně opravování chyb programu, není-li dohodnuto jinak,

- b) zhotoví si záložní rozmnoženinu počítačového programu, je-li to potřebné pro jeho užívání,
- c) zkoumá, studuje nebo zkouší sám nebo jím pověřená osoba fungování počítačového programu za účelem zjištění myšlenek a principů, na nichž je založen kterýkoli prvek počítačového programu, činí-li tak při zavedení, uložení počítačového programu do paměti počítače nebo při jeho zobrazení, provozu či přenosu,
- d) rozmnožuje kód nebo překládá jeho formu při rozmnožování počítačového programu nebo při jeho překladu či jiném zpracování, úpravě či jiné změně, a to ať již sám nebo jím pověřená osoba, jsou-li takové rozmnožování nebo překlad nezbytné k získání informací potřebných k dosažení vzájemného funkčního propojení nezávisle vytvořeného počítačového programu s jinými počítačovými programy, jestliže informace potřebné k dosažení vzájemného funkčního propojení nejsou pro takové osoby jinak snadno dostupné a tato činnost se omezuje na ty části počítačového programu, které jsou potřebné k dosažení vzájemného funkčního propojení.

(Konec citace.)

Rozhodne-li se tedy autor, může licenční smlouvu uzavřít tak, že nepožaduje odměnu. Licenční smlouva však při používání počítačového programu musí existovat, nicméně o tom, zda lze za licenční smlouvu považovat zveřejnění díla autorem na stránkách v Internetu, lze spekulovat. Oproti americkému právu pak nelze autorsky chránit algoritmus či princip práce software. Rovněž je dovoleno uživateli licence software upravovat tak, aby vyhovoval účelu, k němuž se jeho licence vztahuje. Jistě za předpokladu, že mu autor poskytne za tímto účel potřebné podklady, tj. zdrojové kódy, což by mělo být (pro jistotu) uvedeno také v licenční smlouvě (ale zřejmě dle předchozí citace zákona nemusí).

LITERATURA

Následující seznam knih obsahuje všechny odkazované publikace textu knihy. Adresy Internetu jsou uváděny jako doplněk, pokud mají pro publikaci význam a jejich platnost vzhledem k dynamicky se měnící podobě Internetu není zaručena.

Renomované vydavatelství publikací z oblasti informačních technologií pro uživatele různých úrovní je O'Reilly & Associates, Inc., které lze najít na adrese www.ora.com.

Protože tištěné papírové knihy již nezmění svoji podobu, je lépe se při hledání aktuálních informačních zdrojů orientovat na Internet. V současné době je výborným zdrojem pro objasnění pojmů Computer Science a informačních technologií encyklopedie www.wikipedia.org.

Popis informačních technologií Internetu lze najít v technologickém centru jeho Konsorcia WWW na adrese www.w3.org.

[AdobeIllustrator2004]

Adobe Creative Team:

Adobe Illustrator cs Classroom in a Book

Adobe Press, 2004

www.adobepress.com

česky Adobe Illustrator CS – oficiální výukový kurz

SoftPress, 2004

www.softpress.cz

[AdobePhotoshop2004]

Adobe Creative Team:

Adobe Photoshop cs Classroom in a Book

Adobe Press, 2004

www.adobepress.com

[ASCII]

ANSI X3.4 US standard,

americká varianta ISO/IEC 646

www.asciitable.com

[Bach1986]

Maurice J. Bach:

The Design of the UNIX Operating System

Prentice Hall, Inc., 1986

česky Principy operačního systému UNIX,

Softwarové Aplikace a Systémy, Praha 1993

[Chomsky1957]

Noam Chomsky:

Syntactic Structures.

The Hague: Mouton, 1957, Berlin, New York, 1985

[CorelDraw2004]

Steve Bain:

CorelDRAW 12 : The Official Guide

McGraw-Hill/Osborne, 2004

www.osoborne.com

česky CorelDRAW 10: Oficiální průvodce,

SoftPress, 2001

www.softpress.cz

[Custler1993]

Helen Custler:

Inside Windows NT

Microsoft Press, 1992

česky Windows NT,

Grada Publishing, Praha 1994

[Flanagan1996]

David Flanagan:

Java in a NutShell

O'Reilly & Associates, Inc., 1996, 1997, 1999, 2002

česky Programování v jazyce Java,

Computer Press, Brno 1997

www.ora.com

[Gibson1984]

William Gibson:

Neuromancer

Ace Books, 1984

česky Laser, Plzeň 1992

[IBM-Dictionary]

George McDaniel:

IBM Dictionary of Computing

McGRAW-HILL, Inc., 1994

[ISOIEC]

ISO/IEC

10646-1: 1993, 2000

10646-2: 2001

10646-3: 2003

Information Technology – Universal Multiple-Octet Coded Character Set (UCS)

www.iso.org

[ISOOSI1984]

ISO 7498

Information processing system – Open Systems Interconnection – Basic Reference Model

1984

[ISOSQL2003]

ISO/IEC 9075

Information technology – Database languages – SQL

1999, 2003

[ITSEC1991]

Information Technology Security Evaluation Criteria

Luxemburg: Office for Official Publications of the European Communities 1991

[KernighanPlauger1976]

Brian W. Kernighan, P. J. Plauger:

Software Tools

Reading, Addison-Wesley, 1976

[Kimball1996]

Ralph Kimball:

The Data Warehouse Toolkit

John Wiley & Sons, Inc., 1996, 2002

[Kunz2004]

Radim Kunz:

Case studio

www.casestudio.com

[MadnickDonovan1974]

Stuart E. Madnick, John J. Donovan:

Operating Systems

McGraw-Hill, Inc., 1974

česky Operační systémy,

SNL, Praha 1981, 1983

[Manna1974]

Zohar Manna:

Mathematical theory of computation

McGraw Hill, 1974, Dover, 2003

česky Matematická teorie programů,

SNL Praha 1981

[MurrayRipper1994]

James D. Murray, William van Ryper:

Encyclopedia of Graphics File Formats

O'Reilly & Associates, Inc., 1994, 1996

česky Encyklopedie grafických formátů II,

Computer Press, Brno 1997

[MuscianoKennedy2000]

Chuck Musciano, Bill Kennedy:

HTML & XHTML The Definitive Guide

O'Reilly & Associates, Inc., 1998, 2000, 2002

česky HTML a XHTML Kompletní průvodce,

Computer Press, Praha 2000

www.ora.com

[Norton1986]

Peter Norton:

Insight The IBM PC/Peter Norton

Prentice Hall, Inc., New York, 1986,

2002 Scott Clark: Peter Norton's New Inside the PC

[Norton1994]

Peter Norton:

Peter Norton's Complete Guide to DOS 6.22

Prentice Hall, Inc., New York, 1994

[Norton2002]

Peter Norton:

Peter Norton's Complete Guide to Windows XP

Prentice Hall, Inc., New York, 2002

[Plan9 2002]

Plan 9 4th Edition Box Set

Plan 9, An Advanced Network Operating system

Vita Nuova, 2002

cm.bell-labs.com/plan9dist

[Pokorný1994]

Jaroslav Pokorný:

Dotazovací jazyky,

Science, Veletiny 1994

[Pokorný1998]

Jaroslav Pokorný:

Databázová abeceda,

Science, Veletiny 1998

[POSIX2003]

IEEE (Institute of Electrical and Electronics Engineers):

Portable Operating System Interface (POSIX),

IEEE Std 1003.1-2001 (Open Group Technical Standard, Issue 6)

IEEE, 1994, 1996, 2001, 2004,

[Ritchie1970]

Brian W. Kernighan, Dennis M. Ritchie:

The C Programming Language

Prentice Hall, Inc., 1978, 1988

[Skočovský1993]

Luděk Skočovský:

Principy a problémy operačního systému UNIX

Science, Veletiny 1993,

skocovsky.cz/sko93

[Skočovský1998]

Luděk Skočovský:

UNIX, POSIX, PLAN9

vydal autor, Brno 1998

skocovsky.cz/upp9

[Symantec]

www.symantec.com

[Šimůnek1999]

Milan Šimůnek:

SQL

vydavatelství Grada publishing, Praha 1999

[TCSEC1985]

Trusted Computer Systems Evaluation Criteria

DoD 5200.28-STD, Department of Defense, U.S.A., 1985.

[Turing1936]

Alan Turing:

On Computable Numbers, With an

Application to the Entscheidungsproblem

Proceedings of the London Mathematical Society,

Series 2, Volume 42, 1936,

NY: Raven Press, 1965

[UNICODE]

The Unicode Consortium

Mountain View, CA 94043 U.S.A:

The Unicode Standard

verze: rok vydání

1.0: 1991

1.1: 1993

2.0: 1996

2.1: 1998

3.0: 2000

3.1: 2001

3.2: 2002

4.0: 2003

4.0.1: 2004

www.unicode.org

[Vogel1976]

Jiří Vogel:

Programování v jazyku FORTRAN

SNTL Praha 1976

[Wirth1970]

Kathleen Jensen, Niklaus Wirth:

PASCAL – User Manual and Report, ISO

Pascal Standard

Springer Verlag 1974, 1985, 1991

[XWindow1993]

The Definitive Guides to the X Window

System. Volume 0 – Volume Eight

Nye, A. Volume 0 - X Protocol Reference Manual.

Nye, A. Volume One - Xlib Programming Manual.

Nye, A. Volume Two - Xlib Reference Manual.

Quarcia, V., O'Reilly, T. Volume Three - X Window System User's Guide.

Nye, A., O'Reilly, T. Volume Four - X Toolkit Intrinsic Programming Manual.

Flanagan, D. Volume Five - X Toolkit Intrinsic Reference Manual.

Heller, D. Volume Six - Motif Programming Manual.

Heller, D. Volume Seven - XView Programming Manual.

Mui, L., Pearce, E. Volume Eight - X Window System Administrator's Guide.

Sebastopol CA: O'Reilly & Associates, 1993.

[ZákonČR121/2000]

Zákon č. 121/200 Sb. České republiky:

Zákon ze 7. dubna 2000

**o právu autorském, o právech souvisejících
s právem autorským a o změně některých
zákonů (autorský zákon)**

Parlament České republiky, 2000

Symboly

<A> 226
 119, 129, 227

 225, 227
 <CAPTION> 228
 <DIV> 227
 227
 <FORM> 229
 <FRAME> 228
 <FRAMESET> 229
 <Hn> 227
 <I> 227
 226
 227
 227
 <P> 118, 119, 227
 227
 <TABLE> 228
 <TD> 228
 <TH> 228
 <TR> 225, 228
 <U> 227
 227

A

Acrobat 133, 158
 adresa IP (IP address) 73
 adresář (directory, folder) 42
 AIFF 163
 alfanumerický terminál 138
 algoritmus 29
 ALTER 258
 analýza informačního systému 179
 ANSI (American National Standards Institute) 259
 aplikace 135
 aplikační server 182, 183
 aplikační vrstva sítě 66
 Apple Computer 98, 142, 143, 148, 149, 163, 166, 243
 architektura klient – server 81
 Artificial Intelligence, AI (umělá inteligence) 212
 ASCII 16, 215
 ASF 164
 assembler (jazyk stroje) 27, 125
 atribut 172
 automaty 31
 Autorský zákon 261
 awk 130

B

back door (zadní vrátka) 195
 bajt 15
 bezdisková stanice (diskless machine) 106
 binární editor 128

binární kód (binary code) 125
 binární přenositelnost 127
 bit 15
 bitová mapa (bitmap) 154
 Bílá kniha (ITSEC) 209
 BMP 155
 brána (gateway) 90
 bridge (most) 65
 BSD (Berkeley System Distribution) 96, 110, 233

C

C 28, 29, 32, 34, 57, 125, 129, 184
 C++ 34
 CAD/CAM 154, 156, 161
 CASE 129
 CD (compact disc) 19, 20, 21, 25, 51, 52, 95, 107, 162
 CDA 162
 Centrum výpočtu 13
 certifikační autorita 202
 cesta (path) 43
 červ (worm) 207
 CGI (Common Gateway Interface) 230
 CGM 155
 chat 110
 Chomského klasifikace gramatik 31
 Citrix 98
 číselník 176
 click (klik) 141
 client (klient) 81
 clipboard 146
 CMYK 153
 Computer Science (věda o počítačích) 13
 CPU 19
 CREATE 255
 CSS 230
 cut and paste 147
 Cygwin 98

D

databanka (data bank) 168
 databáze (database) 168
 databázový stroj (database engine) 168
 databázový systém 168
 datové modelování (data modeling) 170
 datový model 172
 datový sklad 187
 DB2 170
 DBMS 168
 deadlock (uváznutí) 101
 debugger 34
 DEC (Digital Equipment Corporation) 47, 149
 DELETE 178, 255
 DES, 3DES (Data Encryption Standard) 200

desktop 140, 148, 150
device (periferie) 19
délka slova 21
diagram E-R 172
digitalizace 14
digitální kopie 191
digitální podpis 200
digitální údaj 15
directory, folder (adresář) 42
diskless machines (bezdiskové stanice) 106
disková paměť 24
disk sharing (sdílení disků) 101, 104
DNS (Domain Name System) 77
DoD (Department of Defense) 71, 208
doména (domain) 76
dotazovací jazyk 170
download (stažení) 94
DPI (dots per inch) 155
drag and drop 143
DROP 258
DSA (Digital Signature Algorithm) 201
DTD 187
DVD 21, 25, 49, 107, 164, 192
dvojková číselná soustava 13
DXF 156

E

ed 233
editor 127
elektronická pošta (electronic mail) 110
Emacs 129
encryption (šifrování) 199
engine 127
entita 171
EPS 132
Ethernet 65
EULA (End User Licence Agreement) 260
ex 233
externí disková paměť 22, 52
externí paměť 21

F

fat client (tlustý klient) 87
FIFO 100
file (soubor) 42
file sharing (sdílení souborů) 104
file system (systém souborů) 42
finger 112
firewall 196
folder, directory (adresář) 42
font 158
foreground (popředí) 143
FORTRAN 28, 32, 265
frame (rámeček) 64
freeware 261

FTP (File Transfer Protocol) 91, 92, 93, 94, 202
FTPS (FTP-SSL) 202
fyzická vrstva sítě 66, 194

G

gateway (brána) 90
generační překladač 126
GIF 132, 155
GIMP 157
GIS 185
GML (Generalize Markup Language) 231
GNU 261
gopher 221
GPL (General Public Licence) 260, 261
grafická aplikace 136
grafický editor 132
grafický formát dat 132
grafický terminál 140
GRANT 257
GSM gateway 115

H

hack 191
hard disk (pevný disk) 25
Hewlett Packard 46, 149
holý stroj (machine) 49
host (uzel sítě) 90, 103
hostitel (host, uzel sítě) 90, 103
hostitelský programovací jazyk 177
hot key (klávesová zkratka) 144
HSV (Hue-Saturation-Value) 153
HTML (Hypertext Markup Language) 116
HTTP (Hypertext Transfer Protocol) 81, 82, 95, 116, 120, 123, 182
HTTPS (HTTP-SSL) 82, 202

I

IBM (International Business Machines) 45, 46, 148, 149, 164, 170, 171, 200, 264
ICQ (I seek you) 110
IEEE (Institute of Electrical and Electronics Engineers) 70, 265
ikona (icon) 143
Illustrator 157, 263
IMAP (Internet Message Access Protocol) 115
INDEX 256
index 174, 176
input/output data, I/O (vstupní/výstupní data) 20
INSERT 176
Instrukce 27
Internet Explorer 81
interpret 126
Intranet 183
IP (Internet Protocol) 66
IP address (adresa IP) 73, 75, 76

ISO (International Organization for Standardization) 186
ITSEC (Bílá kniha) 209

J

Java 28, 32, 34, 122, 125, 127, 129, 184, 185
Java applet 87
JavaScript 230
jazyk stroje (assembler) 27, 125
jazyky vyšší úrovně 27
jádro (kernel) 37
JPEG 155
JSP (Java Server Pages) 184, 185, 230

K

kardinalita vztahu entit 173
Kerberos 203
kernel (jádro) 37
klávesová zkratka (hot key) 144
klient (client) 81
klik (click) 141
klíč 174
kodeky 164
KOI-8 59, 259
kompilátor (překladač) 125
komunikační protokol 81

L

ladění programů 34
LATIN2 60
ldap 221
linková vrstva sítě 71
local (místní) 83
localhost 83
locking (zamykání výpočetního zdroje) 101
login (přihlašování) 93
loopback 83

M

machine (holý stroj) 49
machine code (strojový kód) 125
Mac OS 47, 58, 98, 149, 243
mailto 221
Matrix 191
MD2, MD4, MD5 201
memory (paměť) 19
menu 142
Microsoft Corporation 29, 47, 48, 51, 60, 66, 90, 97, 98, 106, 123, 147, 148, 149, 164, 207
MIDI 164
místní (local) 83
modulární programování 32
most (bridge) 65
Mozilla 81, 129
MPEG 163

MS DOS 47
MS Office 132, 157
MS Windows 29, 47, 48, 57, 142, 143, 149, 161, 198, 239
multiplexing 72
multitasking 40
multiuser (víceuživatelský) 40, 96, 179
MySQL 123, 169

N

nápověda (on-line help) 147
národní abecedy 59
NetBEUI 66
netmask (síťová maska) 89, 223
news 221
NFS (Network File System) 85, 106
Notepad (Poznámkový blok) 58
nroff, troff 117
NTSC 162

O

objektové programování 34
obyčejný uživatel 50
OEM (original equipment manufacturer) 260
off-line 87
okno (window) 138
OLAP (On-Line Analytical Processing) 188
on-line 87
on-line help (nápověda) 147
opakovač (repeater) 65
Open Office 132, 133, 157
operační paměť (operating memory) 19
operační systém (operating system) 37
Oracle 170
Oranžová kniha (Orange Book, TCSEC)) 208
OSI (Open Systems Interconnection) 70
osobní počítače a servery 45

P

paket (packet) 66
PAL 162
paměť (memory) 19
parcialita 175
Pascal 28, 32, 125, 129, 265
path (cesta) 43
PCM (pulsně kódovaná modulace) 163
PDF 59, 133, 158, 159, 259
periferie (device) 19
Perl 127
pevný disk (hard disk) 25
Photoshop 132, 155
PHP 177, 180, 182, 183
PICT 155
ping 76, 83
pixel 58, 140, 141, 153, 162, 226

PNG (Portable Network Graphic Format) 155
POP (Post Office Protocol) 87
pop-up menu (výsuvné menu) 145
popředí (foreground) 143
port transportní vrstvy 68
POSIX 46, 131, 265
poskytovatel Internetu (provider) 75
PostScript 132, 156
Poznámkový blok (Notepad) 58
prezentační vrstva sítě 71
překladač (kompilátor) 125
přenositelnost (portability) programu 125
primární klíč 171
privilegovaný uživatel (superuser) 50
přihlašování (login) 93
proces 38
procesor 19
program 20, 27
programovací jazyk vyšší úrovně 27
programování ve vrstvách 35
provider (poskytovatel Internetu) 75
proxy 196
Průzkumník 51, 52
průzkumník 56
Python 127

Q

QuickTime 166

R

RAM 21
řády dat 56
rámeček (frame) 64
RC2, RC4, RC5 200
Real Audio 164
relační algebra 171
relační vrstva sítě 71
remote (vzdálený) 83
remote printer (vzdálená tiskárna) 101
repeater (opakovač) 65
resolution (rozlišení) 154
resource (výpočetní zdroj) 84, 85, 90, 100
REVOKE 257
RGB 153
RIFF (Resource Interchange File Format) 164
rlogin 96, 221
ROLAP (Relational OLAP) 188
ROLE 257
root 43, 51, 77, 207
router (směrovač) 197
rozlišení (resolution) 154
RPC (Remote Procedure Calls) 203
RSA (Rivest-Shamir-Adleman) 200
RSS (Rich Site Summary) 231

S

Samba 106
Save as 146
sběrnice (bus) 19
sdílení času (time sharing) 40
sdílení disků (disk sharing) 101, 104
sdílení souborů (file sharing) 104
SECAM (Sequential Colour Avec Memoire) 162
Secure RPC 203
segment dat 70
SELECT 178, 253
sendmail 110
Sequel 170
SEQUENCE 256
server 45
servery a osobní počítače 45
services 81
SGI (Silicon Graphics, Inc.) 46, 149, 150, 163
SGML 230
SGML (Standardized General Markup Language) 187
shareware 261
shell 51, 97, 233
sít'ová maska (netmask) 89, 223
sít'ová tiskárna 102
sít'ová vrstva sítě 66
sít'ové periferie 64
sít'ový operační systém 107
sít'ový programovací jazyk 71
skript 130
skriptovací jazyk 130, 185
Skype 110
směrovač (router) 197
SMTP (Simple Mail Transfer Protocol) 87
SOAP (Simple Object Access Protocol) 231
soubor (file) 42
source code (zdrojový kód) 125
spam 112
SQL (Structured Query Language) 170, 172, 186, 252
SSL (Secure Socket Layer) 200, 202
standard sít'ových vrstev 70
stažení (download) 94
strojový kód (machine code) 125
strukturované programování 32
STYLK 155
Sun Microsystems 46, 123, 127, 149, 185, 203
superuser (privilegovaný uživatel) 50
swap 41
systém přerušení 39
systém souborů (file system) 42
šifrování (encryption) 199

T

tag 117, 225
talk 110
Tel 131
TCP (Transmission Control Protocol) 68, 259
TCP/IP 81, 83, 183, 194
TCSEC, Orange Book (Oranžová kniha) 208
telnet 96, 202
tenký klient (thin client) 87
teorie programování 29
textový editor 58, 127
TFTP (Trivial FTP) 106
TGA (Targa Image File) 155
TIFF (Tag Image File Format) 155
tlustý klient (fat client) 87
transportní vrstva sítě 67, 182, 198
trojský kůň (trojan horse) 195
trvalá paměť 25
třívrstvá architektura 179
Turingův stroj 30

U

umělá inteligence (AI, Artificial Intelligence) 212
UNICODE 60, 219, 265
UNIX 43, 46, 47, 51, 57, 81, 87, 96, 103, 106,
108, 117, 130, 149, 208, 233, 248, 251,
265
UPDATE 178, 255
upload 95
URI (Uniform Resource Identifier) 119, 221
URL 119, 221
USER 256, 257
UTF (Unicode Transformation Formats) 60
uváznutí (deadlock) 101
uzel sítě (host) 90, 103
uživatel (user) 112
uživatelské rozhraní (user interface) 138

V

validator 226
VCD (Video CD) 164
vektor 154
věda o počítačích (Computer Science) 13
vi 58, 128, 233
víceuživatelský (multiuser) 46, 96, 179
VIEW 256, 257
virtuální realita 165
virus 206
VMS 47, 149
vnitřní formát dat 132, 156
Von Neumannovo schéma 18
VRML (Virtual Reality Markup Language) 166
vrstvy operačního systému 44
vrstvy sítě (layers) 66

vstupní/výstupní data, V/V (input/output data) 20
výpočet 13
výpočetní zdroj (resource) 84, 85, 90, 100
výsuvné menu (pop-up menu) 145
vzdálená tiskárna (remote printer) 101
vzdálený (remote) 83
vztah 172

W

W3C 187
WAV (Waveform Audio File) 164
white spaces 128
window (okno) 138
Windows1250 60
WMA (Window Media Audio) 164
worm (červ) 207
WPG (WordPerfect Graphics Metafile) 155
write 108, 109
WWW (World Wide Web) 116

X

XHTML (Extensible Hypertext Markup Language)
123, 231
XML (Extensible Markup Language) 123, 167,
187, 230, 231
XML-RPC (XML-Remote Procedure Call) 231
X Window System 47, 87, 98, 149, 152, 204, 266

Y

YUV 153

Z

zadní vrátka (back door) 194
zamykání výpočetního zdroje (locking) 101
zdrojový kód (source code) 125
značkovací jazyky 187, 230

