

Počítače a programování 1

Garant předmětu:

Doc. Ing. Ivo Provazník, Ph.D.

Autor textu:

Doc. Ing. Ivo Provazník, Ph.D.

Obsah

1	ÚVOD	5
2	ČÍSELNÁ INFORMACE V POČÍTAČI	6
2.1	ČÍSELNÉ SOUSTAVY	6
2.2	ČÍSELNÉ KÓDY	7
2.3	VYJÁDRĚNÍ ZÁPORNÝCH ČÍSEL	10
2.4	DATOVÉ STRUKTURY	11
2.4.1	<i>Celá čísla</i>	<i>11</i>
2.4.2	<i>Čísla s plovoucí čárkou</i>	<i>12</i>
3	ARCHITEKTURA POČÍTAČŮ	15
3.1	ARCHITEKTURA PC	16
3.1.1	<i>Základní deska</i>	<i>17</i>
3.2	KOMUNIKAČNÍ ROZHRAŇÍ PC	21
3.2.1	<i>Sériová rozhraní</i>	<i>21</i>
3.2.2	<i>Paralelní rozhraní</i>	<i>23</i>
3.2.3	<i>Rozhraní IDE</i>	<i>24</i>
3.2.4	<i>Rozhraní SCSI</i>	<i>28</i>
4	OPERAČNÍ SYSTÉMY	29
4.1	CO JE TO OPERAČNÍ SYSTÉM	29
4.1.1	<i>Funkce operačního systému</i>	<i>29</i>
4.1.2	<i>Moduly operačních systémů</i>	<i>30</i>
4.2	PROCESY	31
4.2.1	<i>Stavy procesu</i>	<i>31</i>
4.2.2	<i>Vykonávání procesu</i>	<i>32</i>
4.3	ZÁKLADY OPERAČNÍHO SYSTÉMU UNIX	34
4.3.1	<i>Vstup uživatele do systému</i>	<i>34</i>
4.3.2	<i>Základní komunikace</i>	<i>34</i>
4.3.3	<i>Soubory</i>	<i>35</i>
4.3.4	<i>Procesy</i>	<i>44</i>
4.3.5	<i>Jádro</i>	<i>47</i>
4.3.6	<i>Uživatelé</i>	<i>47</i>
5	PŘENOS INFORMACÍ A POČÍTAČOVÉ SÍTĚ	49
5.1	ZÁKLADY PROPOJOVÁNÍ PROSTŘEDKŮ	49
5.1.1	<i>Referenční model OSI</i>	<i>49</i>
5.1.2	<i>Informační jednotky přenosu dat</i>	<i>51</i>
5.2	TYPY SÍTÍ	52
5.2.1	<i>Sítě podle rozlehlosti</i>	<i>52</i>
5.2.2	<i>Sítě podle topologie</i>	<i>54</i>
5.2.3	<i>Sítě podle technologie přístupu k médiu</i>	<i>55</i>
5.2.4	<i>Sítě podle síťové architektury</i>	<i>56</i>
5.2.5	<i>Sítě podle typu média</i>	<i>57</i>
5.3	ARCHITEKTURA TCP/IP	58
5.3.1	<i>Vrstvy architektury TCP/IP</i>	<i>59</i>
5.3.2	<i>Adresace v architektuře TCP/IP</i>	<i>61</i>
5.3.3	<i>Datagramy v architektuře TCP/IP</i>	<i>64</i>

6	INTERNET A JEHO SLUŽBY	66
6.1	HISTORICKÉ MEZNÍKY INTERNETU	67
6.2	PRINCIPY INTERNETU	68
6.3	ZÁKLADNÍ SLUŽBY INTERNETU	69
6.3.1	<i>Přenos souborů přes Internet.....</i>	<i>70</i>
6.3.2	<i>Elektronická pošta.....</i>	<i>70</i>
6.3.3	<i>Vzdálený přístup k hostitelskému počítači</i>	<i>74</i>
6.4	SLUŽBA WEB	75
6.4.1	<i>Nástroje pro tvorbu webových stránek</i>	<i>77</i>
6.4.2	<i>Protokol HTTP.....</i>	<i>81</i>
6.4.3	<i>Žádosti klienta v protokolu HTTP.....</i>	<i>82</i>
7	JAZYK HTML.....	84
7.1	ZÁKLADNÍ WEBOVÁ STRÁNKA	84
7.2	ZÁKLADY HTML.....	86
7.2.1	<i>Tělo stránky.....</i>	<i>86</i>
7.2.2	<i>Formátování textu.....</i>	<i>88</i>
7.2.3	<i>Odkazy.....</i>	<i>92</i>
7.2.4	<i>Obrázky.....</i>	<i>92</i>
7.2.5	<i>Tabulky.....</i>	<i>94</i>
8	DODATKY	100
8.1	SEZNAM ZKRATEK.....	100

Seznam obrázků

OBRÁZEK 2.1:	STRUKTURA ČÍSLA SE ZNAMÉNKOVÝM BITEM.	10
OBRÁZEK 2.2:	SCHÉMATICKÉ ZOBRAZENÍ BYTU.	11
OBRÁZEK 2.3:	SCHÉMATICKÉ ZOBRAZENÍ 16-BITOVÉHO CELÉHO ČÍSLA.	12
OBRÁZEK 2.4:	SCHÉMATICKÉ ZOBRAZENÍ 32-BITOVÉHO ČÍSLA S PLOVOUCÍ ČÁRKOU.	12
OBRÁZEK 3.1:	KLASICKÉ ZPRACOVÁNÍ INSTRUKCÍ PROCESOREM.	18
OBRÁZEK 3.2:	ZŘETĚZENÉ ZPRACOVÁNÍ INSTRUKCÍ PROCESOREM.	19
OBRÁZEK 4.1:	MODEL VÝPOČTU ÚLOHY S UVEDENÍM STAVŮ.	31
OBRÁZEK 4.2:	ŽIVOTNÍ CYKLUS PROCESU A ÚLOHY V OPERAČNÍM SYSTÉMU.	33
OBRÁZEK 5.1:	SEDMIVRSTVÁ ARCHITEKTURA REFERENČNÍHO MODELU OSI.	50
OBRÁZEK 5.2:	SBĚRNICOVÁ TOPOLOGIE LOKÁLNÍ SÍTĚ.	54
OBRÁZEK 5.3:	STROMOVÁ TOPOLOGIE LOKÁLNÍ SÍTĚ.	54
OBRÁZEK 5.4:	KRUHOVÁ TOPOLOGIE LOKÁLNÍ SÍTĚ.	55
OBRÁZEK 5.5:	HVĚZDICOVÁ TOPOLOGIE LOKÁLNÍ SÍTĚ.	55
OBRÁZEK 5.6:	PROTOKOLY SADY TCP/IP VE VRSTVÁCH OSI A VRSTVÁCH TCP/IP.	59
OBRÁZEK 5.7:	STRUKTURA DATAGRAMU V ARCHITEKTUŘE TCP/IP.	64
OBRÁZEK 5.8:	DATAGRAM S HLAVIČKOU TCP.	65
OBRÁZEK 6.1:	ORIGINÁLNÍ NÁČRT SÍTĚ ARPANET SE 4 UZLY.	67
OBRÁZEK 6.2:	PRINCIP PŘENOSU SOUBORŮ V INTERNETU S VYUŽITÍM SLUŽBY FTP.	70
OBRÁZEK 6.3:	LOGICKÉ MODULY ELEKTRONICKÉ POŠTY.	71
OBRÁZEK 6.4:	PRINCIP PŘIPOJENÍ KE VZDÁLENÉMU POČÍTAČI POMOCÍ SLUŽBY TELNET.	75
OBRÁZEK 6.5:	PŘÍKLAD JEDNODUCHÉ STRÁNKY VYTVOŘENÉ V JAZYCE HTML.	77
OBRÁZEK 6.6:	PŘÍKLAD STRÁNKY S FORMULÁŘEM S POUŽITÍM SKRIPTU CGI.	79
OBRÁZEK 6.7:	UKÁZKA GRAFICKÉHO VÝSTUPU PROGRAMU V JAZYCE JAVASCRIPT.	81
OBRÁZEK 7.1:	PŘÍKLAD ZOBRAZENÍ ZÁKLADNÍ STRÁNKY. ZOBRAZENÍ SE MŮŽE LIŠIT PODLE TYPU PROHLÍŽEČE, JEHO VERZE, A NASTAVENÍ PROHLÍŽEČE.	85
OBRÁZEK 7.2:	PŘÍKLAD ZOBRAZENÍ ZÁKLADNÍ STRÁNKY S OBRÁZKEM NA POZADÍ.	86
OBRÁZEK 7.3:	PŘÍKLAD ZOBRAZENÍ ZÁKLADNÍ STRÁNKY S JEDNOLITÝM POZADÍM BARVY 0099FF A TEXTEM BARVY FFFF00.	87
OBRÁZEK 7.4:	PŘÍKLAD ZOBRAZENÍ STRÁNKY SE VŠEMI ÚROVNĚMI NADPISŮ.	89
OBRÁZEK 7.5:	PŘÍKLAD POUŽITÍ Odstavců se RŮZNÝM ZAROVNÁNÍM.	90
OBRÁZEK 7.6:	PŘÍKLAD NEČÍSLOVANÉHO SEZNAMU.	90
OBRÁZEK 7.7:	PŘÍKLAD VNOŘENÉHO NEČÍSLOVANÉHO SEZNAMU.	91
OBRÁZEK 7.8:	PŘÍKLAD VNOŘENÉHO ČÍSLOVANÉHO SEZNAMU S RŮZNÝM TYPEM ČÍSLOVÁNÍ NA RŮZNÝCH ÚROVNÍCH.	91
OBRÁZEK 7.9:	PŘÍKLAD ODKAZU NA EXTERNÍ STRÁNKU.	92
OBRÁZEK 7.10:	PŘÍKLAD VLOŽENÉHO OBRÁZKU.	94
OBRÁZEK 7.11:	PŘÍKLAD TABULKY S JEDNOLITÝM RÁMEČKEM.	95
OBRÁZEK 7.12:	PŘÍKLAD TABULKY S PLASTICKÝM RÁMEČKEM.	95
OBRÁZEK 7.13:	PŘÍKLAD FORMÁTOVANÉ TABULKY.	96
OBRÁZEK 7.14:	PŘÍKLAD FORMÁTOVÁNÍ ŘÁDKU TABULKY.	97
OBRÁZEK 7.15:	PŘÍKLAD FORMÁTOVÁNÍ BUŇKY TABULKY.	98
OBRÁZEK 7.16:	PŘÍKLAD SLUČOVÁNÍ BUNĚK TABULKY VE VERTIKÁLNÍM I HORIZONTÁLNÍM SMĚRU.	99
OBRÁZEK 7.17:	PŘÍKLAD POUŽITÍ HLAVIČKY TABULKY.	99

Seznam tabulek

TABULKA 2.1:	GRAYŮV KÓD.....	8
TABULKA 2.2:	KÓD BCD.....	9
TABULKA 2.3:	KÓD ASCII.....	9
TABULKA 2.4:	PŘÍKLAD VYJÁDŘENÍ ČÍSEL V JEDNIČKOVÉM DOPLŇKU.....	10
TABULKA 2.5:	PŘÍKLAD VYJÁDŘENÍ ČÍSEL VE DVOJKOVÉM DOPLŇKU.....	10
TABULKA 2.6:	VÝZNAM 4-BITOVÉHO SLOVA PŘI VYJÁDŘENÍ 1-DOPLŇKEM, 2-DOPLŇKEM A BEZ ZNAMÉNKA.....	11
TABULKA 2.7:	NĚKTERÉ STRUKTURY PRO VYJÁDŘENÍ CELÝCH ČÍSEL.....	12
TABULKA 2.8:	ROZLOŽENÍ BITŮ PODLE IEEE 754.....	12
TABULKA 2.9:	ROZSAH HODNOT PODLE NORMY IEEE 754.....	13
TABULKA 2.10:	SPECIÁLNÍ ČÍSLA PODLE NORMY IEEE 754.....	14
TABULKA 3.1:	GENERACE POČÍTAČŮ.....	16
TABULKA 3.2:	ZÁKLADNÍ PARAMETRY PROCESORŮ.....	17
TABULKA 5.1:	TABULKA ROZSAHU IP ADRES V TŘÍDÁCH A, B, C V BINÁRNÍM VYJÁDŘENÍ.....	62
TABULKA 5.2:	TABULKA ROZSAHU IP ADRES V TŘÍDÁCH A, B, C V DEKADICKÉM VYJÁDŘENÍ.....	62
TABULKA 7.1:	POJMENOVANÉ BARVY JAZYKA HTML.....	87

Seznam grafů

GRAF 6.1:	POČET POČÍTAČŮ S REGISTROVANOU IP ADRESOU PŘIPOJENÝCH K INTERNETU V LOGARITMICKÉM MĚŘÍTKU.....	68
GRAF 6.2:	POČET WEBOVÝCH SERVERŮ NA INTERNETU.....	76
GRAF 6.3:	VÝVOJ PODÍLU WEBOVÝCH SERVERŮ RŮZNÝCH VÝROBCŮ.....	77

1 Úvod

Elektronický text *Počítače a programování 1* je určen pro stejnojmenný předmět bakalářského studijního programu na Fakultě elektrotechniky a komunikačních technologií VUT v Brně. Cílem je předmětu je seznámit studenty s výpočetní technikou včetně jejího praktického použití. Součástí náplně předmětu je popis vnitřního hardwarového i softwarového uspořádání počítačů, základy algoritmizace a úvod do programování.

Kontakt: Doc. Ing. [Ivo Provazník](#), Ph.D.

☎ 541 149 562

✉ ivo@ieee.org

Adresa: Ústav biomedicínského inženýrství
Purkyňova 118
61200 Brno

2 Číselná informace v počítači

2.1 Číselné soustavy

Nejpoužívanějším číselným systémem dneška je arabský systém. Byl vyvinut Hindy přibližně ve 3. století před naším letopočtem. Součástí systému je nula, s jejíž pomocí je využívána množina hodnot rekurentním (opakujícím se) způsobem. Například desítková soustava obsahuje hodnoty 0 až 9, které se opakují. S každým opakováním se inkrementuje číslice, která je umístěna ve sloupci nejvíce vlevo. Kdykoliv má být inkrementována číslice s hodnotou 9, mění se její hodnota zpět na 0 a inkrementuje se nejbližší levý soused. Dlužno poznamenat, že v arabském systému je číslice s nejvyšší vahou umístěna nejvíce vlevo.

Základ

Základ číselného systému je dán počtem unikátních hodnot, které se vyskytují v množině čísel bez opakování. Například, desítková soustava (základ je roven 10) obsahuje 10 číslic. Ostatní soustavy používané v počítačové technice mají základ

- ▶ dvojková (binární) soustava: základ 2, hodnoty {0, 1},
- ▶ osmičková (oktalová) soustava: základ 8, hodnoty {0 - 7},
- ▶ desítková (dekadická) soustava: základ 10, hodnoty {0 - 9},
- ▶ šestnáctková (hexadecimální) soustava: základ 16, hodnoty {0 - 9, A-F},

a některé historické:

- ▶ duodecimální soustava: základ 12 (používáno Římany),
- ▶ vigesimální soustava: základ 20 (používáno May),
- ▶ sexagesimální soustava: základ 60 (Používáno Babyloňany).

Základ se často označuje subscriptem za číslem (není-li základ zřejmý z kontextu).

Váhový faktor

Váhový faktor je hodnota, kterou se násobí číslice na určité pozici v čísle. Například v desítkové soustavě je váhový faktor mocnina 10, kde mocnitel je dán pořadím číslice zprava počínaje 0. Příklad dekadického čísla 312_{10} :

$$\begin{array}{rcl}
 312_{10} = & & \\
 \begin{array}{l} \text{---} \\ \text{---} \\ \text{---} \end{array} & \begin{array}{l} 2 * 10^0 = 2 \\ 1 * 10^1 = 10 \\ 3 * 10^2 = 300 \end{array} & \begin{array}{l} = 2 \\ = 10 \\ = 300 \end{array} \\
 & & \hline
 & & 312 \text{ (součet)}
 \end{array}$$

Jiný příklad binárního čísla 1011_2 :

$$\begin{array}{rcl}
 1011_2 = & & \\
 \begin{array}{l} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} & \begin{array}{l} 1 * 2^0 = 1 \\ 1 * 2^1 = 2 \\ 0 * 2^2 = 0 \\ 1 * 2^3 = 8 \end{array} & \\
 & & \hline
 & & 11 \text{ (dekadický součet)}
 \end{array}$$

Konverze čísel mezi soustavami

Pro převod čísel mezi jednotlivými soustavami (změna základu) existuje mnoho metod. Uvedme dvě základní metody pro převod mezi nepoužívanější dekadickou a ostatními soustavami.

1. metoda - převod z dekadické do jiné soustavy: Dekadické číslo je poděleno základem, zbytek po dělení je zapsán. Podíl bez zbytku je opět podělen základem, dokud není podíl roven 0. Reverzně zapsané zbytky po dělení tvoří převedené číslo. Například číslo 254_{10} je převedeno do binární soustavy takto:

```
254 / 2 = 127 a zbytek 0
127 / 2 = 63 a zbytek 1
63 / 2 = 31 a zbytek 1
31 / 2 = 15 a zbytek 1
15 / 2 = 7 a zbytek 1
7 / 2 = 3 a zbytek 1
3 / 2 = 1 a zbytek 1
1 / 2 = 0 a zbytek 1
```

a výsledek je 11111110_2 . Pro převod do hexadecimální soustavy např. číslo 232_{10} :

```
232 / 16 = 14 a zbytek 8
14 / 16 = 0 a zbytek E ( $14_{10} = E$ )
```

a výsledek je tedy $E8_{16}$.

2. metoda - převod z jiné do dekadické soustavy: Každá číslice v čísle je na pozici s příslušným váhovým faktorem pro soustavu, do které převádíme. Sečteme všechny číslice vynásobené váhovými faktory. Například převod binárního čísla 10010010_2 :

$$10010010 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 146_{10}$$

nebo převod čísla $17F_{16}$:

$$17F = 1 \cdot 16^2 + 7 \cdot 16^1 + 15 \cdot 16^0 = 383_{10}$$

2.2 Číselné kódy

Grayův kód

Grayův kód je cyklický kód s proměnnými váhovými faktory. Je navržen tak, že přechod z jedné jeho hodnoty na následující je doprovázen změnou pouze jediného bitu.

Dekadicky	Binárně	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011

14	1110	1001
15	1111	1000

Tabulka 2.1: Grayův kód.

Převod z Grayova kódu do binární soustavy

1. Nejvýznamnější bit binárního čísla bude roven nejvýznamnějšímu bitu čísla v Grayově kódu.
2. Další bit binárního čísla je získán součtem modulo 2 dalšího bitu binárního čísla a dalšího bitu čísla v Grayově kódu.
3. Krok 2. je opakován tak dlouho, dokud nejsou všechny bity čísla v Grayově kódu přičteny.

Například číslo 1101101 v Grayově kódu je převedeno takto:

krok	Gray	binárně	
0.	1101101		
1.	1 101101	1	kopie nejvýznamnějšího bitu
2.	1 1 01101	1 0	1 modulo 2 1 = 0
3.	1 1 1 0101	1 0 0	0 modulo 2 0 = 0
3.	1 1 0 1 101	1 0 0 1	0 modulo 2 1 = 1
3.	1 1 0 1 1 01	1 0 0 1 0	1 modulo 2 1 = 0
3.	1 1 0 1 1 0 1	1 0 0 1 0 0	0 modulo 2 0 = 0
3.	1 1 0 1 1 0 1	1 0 0 1 0 0 1	0 modulo 2 1 = 1

Výsledek je binární číslo 1001001.

Převod z binární soustavy do Grayova kódu

1. Nejvýznamnější bit binárního čísla bude roven nejvýznamnějšímu bitu čísla v Grayově kódu.
2. Další bit čísla v Grayově kódu je získán součtem modulo 2 dalšího bitu binárního čísla a dalšího bitu binárního čísla.
3. Krok 2. je opakován tak dlouho, dokud nejsou všechny bity čísla v Grayově kódu přičteny.

Například binární číslo 1101101 je převedeno takto:

krok	binárně	Gray	
0.	1101101		
1.	1 1 0 1 1 0 1	1	kopie nejvýznamnějšího bitu
2.	1 1 0 1 1 0 1	1 1	1 modulo 2 0 = 1
3.	1 1 0 1 1 0 1	1 1 0	0 modulo 2 0 = 0
3.	1 1 0 1 1 0 1	1 1 0 1	0 modulo 2 1 = 1
3.	1 1 0 1 1 0 1	1 1 0 1 1	1 modulo 2 0 = 1
3.	1 1 0 1 1 0 1	1 1 0 1 1 0	0 modulo 2 0 = 0
3.	1 1 0 1 1 0 1	1 1 0 1 1 0 1	0 modulo 2 1 = 1

Výsledek je číslo v Grayově kódu 1101101.

BCD kód

Kód BCD (Binary Coded Decimal) slouží ke kódování číslic desítkové soustavy 4-bitovými slovy. Jeho struktura se shoduje s binární soustavou, pouze bitové kombinace 1010 až 1111 nejsou povolené.

<i>Dekadicky</i>	<i>Binárně</i>	<i>BCD</i>
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	-
11	1011	-
12	1100	-
13	1101	-
14	1110	-
15	1111	-

Tabulka 2.2: Kód BCD.

ASCII kód

Kód ASCII (The American Standard Code for Information Interchange) ASCII je 7-bitový kód pro vyjádření 128 níže uvedených znaků. Jednotlivé kódy jsou obvykle označovány v hexadecimální soustavě, např. kód znaku 'M' je 4C₁₆.

- ▶ znaky a až z a A až Z (bez diakritiky),
- ▶ speciální znaky, např. <, ., ?, :, atd.,
- ▶ číslice 0 až 9,
- ▶ speciální řídicí znaky (přechod na nový řádek, apod).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Tabulka 2.3: Kód ASCII.

2.3 Vyjádření záporných čísel

Záporná čísla mohou být v binární soustavě vyjádřena několika způsoby. Jsou to zejména: využití znaménkového bitu, jedničkový a dvojkový doplněk.

Znaménkový bit

Jako znaménkový bit je obvykle použit nejvýznamnější bit binárního čísla s tím, že jeho hodnota 0 představuje kladné číslo a hodnota 1 pak záporné číslo. Struktura čísla je znázorněna na **Obrázek 2.1**.

znaménkový bit	číslo
----------------	-------

Obrázek 2.1: Struktura čísla se znaménkovým bitem.

Využijeme-li znaménkový bit v 8-bitovém slově, je pro vyjádření vlastního určeno zbývajících 7 bitů. Rozsah takového čísla v dekadickém vyjádření je pak -127 až +127.

Jedničkový doplněk

Jedničkový doplněk je vytvořen tak, že hodnoty všech bitů jsou invertovány ($0 \rightarrow 1$ a $1 \rightarrow 0$). Příklad je uveden v **Tabulka 2.4**.

<i>Dekadicky</i>	<i>Binárně</i>	<i>1-doplněk</i>
7	00000111	11111000
32	00100000	11011111
114	01110010	10001101

Tabulka 2.4: Příklad vyjádření čísel v jedničkovém doplňku.

Dvojkový doplněk

Dvojkový doplněk je vytvořen tak, že číslo je převedeno do svého jedničkového doplňku a pak je přičtena 1. Příklad je uveden v **Tabulka 2.5**.

<i>Dekadicky</i>	<i>Binárně</i>	<i>1-doplněk</i>	<i>2-doplněk</i>
7	00000111	11111000	11111001
32	00100000	11011111	11100000
114	01110010	10001101	10001110

Tabulka 2.5: Příklad vyjádření čísel ve dvojkovém doplňku.

V **Tabulka 2.6** je ukázáno, jaký je význam hodnot 4-bitového slova, je-li použit jedničkový doplněk, dvojkový doplněk, obyčejný binární kód (bez znaménkového bitu) a binární kód se znaménkovým bitem.

<i>Binárně</i>	<i>Dekadicky (1-doplněk)</i>	<i>Dekadicky (2-doplněk)</i>	<i>Dekadicky (bez znaménka)</i>	<i>Dekadicky (se znaménkem)</i>
0111	7	7	7	7
0110	6	6	6	6
0101	5	5	5	5
0100	4	4	4	4
0011	3	3	3	3
0010	2	2	2	2
0001	1	1	1	1
0000	0	0	0	0

znaménka), word (16 bitů bez znaménka), short (8 bitů se znaménkem), byte (8 bitů bez znaménka), long (32 bitů se znaménkem), longint (32 bitů se znaménkem).

Příklad schématu datové struktury pro vyjádření celého čísla je na **Obrázek 2.3**. Z zde znamená znaménkový bit.



Obrázek 2.3: Schématické zobrazení 16-bitového celého čísla.

Podle délky slova a použití znaménka může pojímat čísla v různém rozsahu (viz **Tabulka 2.7**).

Název	bitů	znaménko	rozsah
<i>byte</i>	8	ne	$0 \div 255$
<i>integer</i>	16	ano	$-32768 \div +32767$
<i>word</i>	16	ne	$0 \div 65535$
<i>long</i>	32	ano	$-2147483648 \div +2147483647$

Tabulka 2.7: Některé struktury pro vyjádření celých čísel.

2.4.2 Čísla s plovoucí čárkou

Pro uložení reálných čísel se používají datové struktury s tzv. plovoucí čárkou. Tyto struktury téměř výhradně obsahují tři položky: znaménko, exponent a mantisa. Rozložení bitů pro jednotlivé položky a způsob stanovení jejich obsahu předepisuje norma. Obecné rozložení je naznačeno na **Obrázek 2.4**.



Obrázek 2.4: Schématické zobrazení 32-bitového čísla s plovoucí čárkou.

Dnešním nejpoužívanějším standardem pro popis čísel s plovoucí čárkou je IEEE Standard 754. Používá se pro všechny platformy včetně PC, Macintosh, a většiny UNIXových stanic.

Standard IEEE připouští dvojí přesnost - vyjádření 32-bitovým a 64-bitovým slovem (single precision a double precision). Rozložení bitů je naznačeno v **Tabulka 2.8** kde je uveden vždy počet bitů položky (znaménka, exponentu a mantisy) a v hranatých závorkách odpovídající čísla bitů.

přesnost	znaménko	exponent	mantisa	posun
<i>jednoduchá</i>	1 [31]	8 [30 - 23]	23 [22 - 0]	127
<i>dvojitá</i>	1 [63]	11 [62 - 52]	52 [51 - 0]	1023

Tabulka 2.8: Rozložení bitů podle IEEE 754.

Znaménko

Znaménkový bit svou hodnotou 0 označuje kladné číslo, hodnotou 1 záporné číslo.

Exponent

Bitové pole exponentu musí svou hodnotou reprezentovat kladné i záporné exponenty. Z tohoto důvodu je zaveden tzv. posun, jehož hodnota je uvedena v **Tabulka 2.8**. Posun je jednoduše přičten ke skutečnému exponentu.

Mantisa

Mantisa je složena z implicitního bitu a zbytku. Číslo, které má být uloženo, je upraveno do normalizované formy. Například číslo $12,5_{10}$ je upraveno do své normalizované formy $0,78125 \cdot 2^4$ takto

$$\begin{aligned} 12,5 & / 2 = 6,25 \\ 6,25 & / 2 = 3,125 \\ 3,125 & / 2 = 1,5625 \\ 1,5625 & / 2 = 0,78125 \end{aligned}$$

Číslo je děleno 2 tak dlouho, dokud není výsledek v rozmezí 0 a 1. Zbytek po dělení tvoří mantisu v dekadické soustavě, počet dělení pak hodnotu binárního exponentu.

Výhodnou optimalizací vyjádření čísla je, že první bit mantisy lze explicitně předpokládat nenulový (lze vždy dosáhnout volbou exponentu). Jedinou nenulovou hodnotou v binární soustavě je však pouze 1, proto ji nemusíme vyjadřovat. V případě jednoduché přesnosti tak máme 23-bitovou mantisu s 24-bitovým rozlišením.

Příklad 2.1: Uvažujme vyjádření čísla $12,5_{10}$.

Normalizovaná forma čísla je $0,78125 \cdot 2^4$.

Číslo je kladné, znaménkový bit je tedy roven 0.

Exponent je roven $127 + 4 = 131_{10} = 10000011_2$.

Mantisa $0,78125_{10}$ je převedena do binární soustavy a je rovna $0,11001_2$. První 1 nemusíme vyjadřovat (je implicitní) a celá 23-bitová mantisa je rovna $10010000000000000000000_2$.

Rozsah hodnot

Při uvažování rozsahu hodnot datových struktur pro vyjádření čísel v plovoucí čárce musíme uvažovat maximální hodnoty (největší kladná i záporná čísla), ale také minimální hodnoty (nejbližší 0). Rozsahy pro oba typy přesnosti jsou uvedeny v .

přesnost	rozsah	přibližný dekadický rozsah
jednoduchá	$\pm 2^{-126} \div (2 \cdot 2^{-23}) \cdot 2^{127}$	$\pm \sim 10^{-44,85} \div \sim 10^{38,53}$
dvojitá	$\pm 2^{-1022} \div (2 \cdot 2^{-52}) \cdot 2^{1023}$	$\pm \sim 10^{-323,3} \div \sim 10^{308,3}$

Tabulka 2.9: Rozsah hodnot podle normy IEEE 754.

Při bližším zkoumání lze zjistit, že pomocí výše uvedeného postupu nelze kódovat například číslo 0. Norma však zavádí tzv. speciální čísla pomocí kombinace hodnot bitů mantisy a exponentu. Seznam některých z nich je uveden v **Tabulka 2.10**.

<i>znaménko</i>	<i>exponent</i>	<i>mantisa</i>	<i>hodnota</i>
0	00 ... 00	00 ... 00	+0
0	11 ... 11	00 ... 00	$+\infty$
0 ... 1	11 ... 11	00 ... 01 : 01 ... 11	S NaN
0 ... 1	11 ... 11	10 ... 01 : 11 ... 11	Q NaN
1	00 ... 00	00 ... 00	-0
1	11 ... 11	00 ... 00	$-\infty$

Tabulka 2.10: Speciální čísla podle normy IEEE 754.

Speciální čísla zahrnují ± 0 , $\pm\infty$, a také NaN (Not a Number) pro vyjádření čísla, které neprezentuje reálné číslo. Existují dva druhy NaN: S NaN (Signaling Not a Number) a Q NaN (Quiet Not a Number). První z nich je používán při neplatných operacích, druhé reprezentuje operace, jejichž výsledky nejsou matematicky definovány.

3 Architektura počítačů

Von Neumannovo schéma bylo navrženo roku 1945 americkým matematikem (narozeným v Maďarsku) Johnem von Neumannem jako model samočinného počítače. Tento model s jistými výjimkami zůstal zachován dodnes.

Podle tohoto schématu se počítač skládá z pěti hlavních modulů:

- ▶ Operační paměť - slouží k uchování zpracovávaného programu, zpracovávaných dat a výsledků výpočtu.
- ▶ Aritmeticko-logická jednotka ALU (Arithmetic-Logic Unit) - jednotka provádějící veškeré aritmetické (např. sčítání, násobení, dělení) a logické (např. posuvy, logické AND a OR) operace.
- ▶ Řadič - jednotka, která řídí činnost všech částí počítače. Toto řízení je prováděno pomocí řídicích signálů, které jsou zasílány jednotlivým modulům. Reakce na řídicí signály, stavy jednotlivých modulů jsou naopak zasílány zpět řadiči pomocí stavových hlášení.
- ▶ Vstupní zařízení - zařízení určená pro vstup programu a dat.
- ▶ Výstupní zařízení - zařízení určená pro výstup výsledků.

Ve von Neumannově schématu je možné ještě vyznačit dva další moduly vzniklé spojením předcházejících modulů:

- ▶ Procesor: řadič + ALU.
- ▶ Centrální procesorová jednotka CPU (Central Processor Unit): procesor + operační paměť.

Princip činnosti počítače podle von Neumannova schématu

Do operační paměti se pomocí vstupních zařízení přes ALU umístí program, který bude provádět výpočet. Stejným způsobem se do operační paměti umístí data, která bude program zpracovávat. Proběhne vlastní výpočet, jehož jednotlivé kroky provádí ALU. Tato jednotka je v průběhu výpočtu spolu s ostatními moduly řízena řadičem počítače. Mezivýsledky výpočtu jsou ukládány do operační paměti. Po skončení výpočtu jsou výsledky posílány přes ALU na výstupní zařízení.

Základní odlišnosti moderních počítačů od von Neumannova schématu

- ▶ Podle von Neumannova schématu počítač pracuje vždy s jedním programem. Toto vede k špatnému využití strojového času. Je tedy obvyklé, že počítač zpracovává paralelně více programů zároveň - tzv. multitasking.
- ▶ Počítač může disponovat i více než jedním procesorem.
- ▶ Počítač podle von Neumannova schématu pracoval pouze v tzv. diskrétním režimu (do paměti počítače je zaveden program, data a pak probíhá výpočet; v průběhu výpočtu již není možné s počítačem dále interaktivně komunikovat).
- ▶ Existují vstupní/výstupní zařízení, která umožňují jak vstup, tak výstup dat (nebo programu).
- ▶ Program se do paměti nemusí zavést celý, ale je možné zavést pouze jeho část a ostatní části zavádět až v případě potřeby.

Počítače se historicky rozdělují do tzv. generací, kde každá generace je charakteristická svou konfigurací, rychlostí počítače a základním stavebním prvkem.

<i>Generace</i>	<i>Rok</i>	<i>Rychlost</i>	<i>Aktivní prvky</i>
0.	1940	1 - 10	relé
1.	1950	100 - 1.000	elektronky
2.	1958	1.000 - 10.000	tranzistory
3.	1964	10.000 - 100.000	integrované obvody
3. 1/2	1972	100.000 - 1 mil.	integrované obvody LSI
4.	1981	10 mil.	integrované obvody VLSI

Tabulka 3.1: Generace počítačů.

3.1 Architektura PC

Počítač třídy PC se skládá z následujících částí:

- ▶ Základní jednotka: část uzavřená do skříně počítače, obsahuje (nebo může obsahovat) tyto komponenty:
 - základní deska,
 - procesor,
 - numerický (matematický) koprocessor,
 - operační paměť,
 - cache paměť,
 - CMOS paměť,
 - mechaniky pružných disků,
 - pevné disky,
 - řadiče disků,
 - interní mechaniky diskových médií (CD-R, CD-RW, DVD-R, DVD-RW, ZIP, JAZ, SyJet,...),
 - grafická karta,
 - zvuková karta,
 - I/O karta,
 - síťová karta,
 - další zařízení.
- ▶ monitor,
- ▶ tiskárna,
- ▶ klávesnice,

- ▶ polohovací zařízení (myš, trackball, ...),
- ▶ externí mechaniky diskových médií,
- ▶ PCMCIA zařízení,
- ▶ scanner,
- ▶ další zařízení.

3.1.1 Základní deska

Základní deska je deska plošného spoje tvořící základ celého počítače. Základní deska obsahuje především tyto prvky (moduly):

- ▶ procesor (mikroprocesor),
- ▶ numerický koprocessor (u starších počítačů),
- ▶ obvody čipové sady,
- ▶ rozšiřující sběrnici,
- ▶ paměti,
- ▶ vyrovnávací cache paměť,
- ▶ sloty umístěné na rozšiřující sběrnici pro připojení rozšiřujících karet.

Zařízení jako jsou procesor, numerický koprocessor, řadič cache paměti, paměti a obvody čipové sady jsou společně propojeny pomocí tzv. systémové sběrnice, která umožňuje jejich rychlou vzájemnou komunikaci.

Procesor

Procesor je integrovaný obvod zajišťující funkce CPU. Do značné míry ovlivňuje výkon celého počítače (čím výkonnější procesor, tím výkonnější počítač). Obsahuje rychlá paměťová místa malé kapacity nazývané registry. Základní parametry procesoru:

<i>Parametr</i>	<i>Popis</i>	<i>Jednotka</i>
<i>Taktovací frekvence</i>	Počet kroků (operací) provedených za jednu sekundu	Hz
<i>Počet instrukčních kanálů</i>	Maximální počet instrukcí proveditelných v jednom taktu procesoru	-
<i>Šířka slova</i>	Maximální počet bitů, které je možné zpracovat během jediné operace	bit
<i>Šířka přenosu dat</i>	Maximální počet bitů, které je možné během jediné operace přenést z (do) čipu	bit
<i>Kapacita interní cache paměti</i>	Kapacita rychlé interní paměti integrované přímo na čipu procesoru	byte
<i>Velikost adresovatelné paměti</i>	Velikost paměti, kterou je procesor schopen přímo adresovat (používat)	byte

Tabulka 3.2: Základní parametry procesorů.

Procesor je synchronní zařízení. To znamená, že ke generování jednotlivých řídicích signálů i k jeho dalším činnostem nedochází nahodile, ale v souvislosti s tzv. hodinovým

signálem. Ten je generován vnějšími obvody a je rozváděn obvykle k více jednotkám počítače. Díky tomuto hodinovému signálu je zabezpečeno správné časování veškeré komunikace mezi jednotlivými zařízeními i činností probíhající uvnitř procesoru. Zpracování jedné instrukce programu obvykle trvá několik taktů hodinového signálu. Počet taktů potřebný pro provedení celé instrukce bývá označován jako instrukční cyklus.

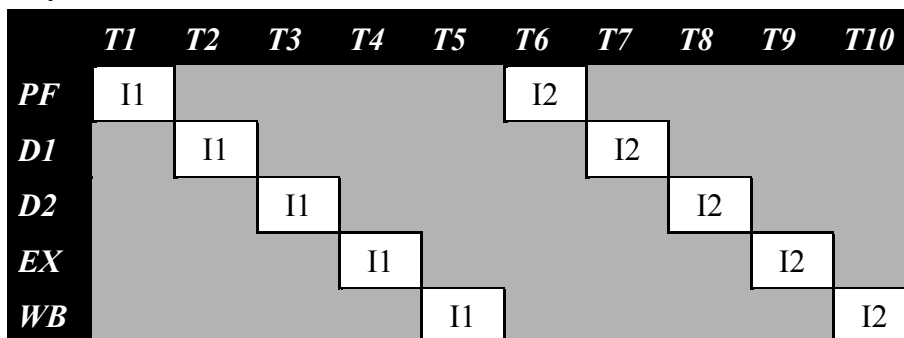
Důležitou součástí procesoru jsou registry. Jsou to velmi rychlá paměťová místa malé kapacity (řádově jednotky bytů) umístěné většinou uvnitř procesoru počítače. Registry je možno rozdělit na registry univerzální a registry se stanoveným významem. Registry univerzální jsou určeny pro uchovávání operandů, mezivýsledků i výsledků podobně jako paměť počítače. Jejich výhodou oproti operační paměti je, že informace v nich uložené jsou přístupné okamžitě bez nutnosti přístupu mimo procesor. Druhou skupinou registrů jsou registry s určitým pevně stanoveným významem. Každý z těchto registrů realizuje určitou specifickou funkci a změnou obsahu těchto registrů je možno ovlivňovat činnost procesoru.

Výkon procesoru také závisí na efektivitě mikrokódu jeho instrukční sady. Prakticky to znamená, jak efektivně jsou jednotlivé instrukce naprogramovány. Například k vydělení dvou čísel může procesor potřebovat 25 ale i 70 taktů.

Ve skutečnosti je však možné provést téměř každou instrukci během jediného taktu. Způsob, kterým se tato jednotaktová operace provádí, se nazývá zřetězené zpracování instrukcí (tzv. pipelining). Tento způsob vychází ze skutečnosti, že zpracování každé instrukce procesorem lze rozdělit do pěti základních fází:

- ▶ PF (Prefetch) - výběr instrukce (další zpracovávaná instrukce se bere buď z paměti RAM, nebo z vyrovnávací cache paměti),
- ▶ D1 (Decode1) - dekodování instrukce (určí se délka a typ instrukce),
- ▶ D2 (Decode2) - určení adres operandů, se kterými instrukce pracuje,
- ▶ EX (Execution) - vlastní provedení instrukce,
- ▶ WB (Write Back) - zápis výsledků zpracované instrukce.

Každou z těchto fází může provádět samostatně pracující jednotka a v okamžiku, kdy je tato jednotka se svou prací hotova, předá svůj výsledek jednotce provádějící následující fázi zpracování a pokračuje ve své práci nad další instrukcí. Klasické a zřetězené zpracování jsou dokumentovány na **Obrázek 3.1** a **Obrázek 3.2**.



Obrázek 3.1: Klasické zpracování instrukcí procesorem.

	<i>T1</i>	<i>T2</i>	<i>T3</i>	<i>T4</i>	<i>T5</i>	<i>T6</i>	<i>T7</i>	<i>T8</i>	<i>T9</i>	<i>T10</i>
<i>PF</i>	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
<i>D1</i>		I1	I2	I3	I4	I5	I6	I7	I8	I9
<i>D2</i>			I1	I2	I3	I4	I5	I6	I7	I8
<i>EX</i>				I1	I2	I3	I4	I5	I6	I7
<i>WB</i>					I1	I2	I3	I4	I5	I6

Obrázek 3.2: Zřetěžené zpracování instrukcí procesorem.

Jednotlivé sloupce značí takt procesoru, v řádcích jsou vždy uvedeny instrukce v patřičné fázi zpracování. Při klasickém zpracování se tak zpracuje v prvních pěti taktech kompletně první instrukce. V dalších pěti taktech se pak kompletně zpracuje druhá instrukce. Při zřetěženém zpracování je v prvních pěti taktech opět zcela zpracována první instrukce, ale další instrukce jsou již rozpracovány, takže v každém dalším taktu je pak zpracována vždy jedna další instrukce. V optimálním případě tedy po deseti taktech je úplně zpracováno šest instrukcí.

Při zřetěženém zpracování však vyvstává problém v okamžiku, kdy některá z instrukcí způsobí skok. V tomto okamžiku je nutné provést vyprázdnění fronty. Předzpracované instrukce totiž vůbec nebudou prováděny, protože chod programu bude pokračovat na jiném místě. Od tohoto místa se tedy začne opět plnit fronta instrukcí.

Procesor, který má právě jednu frontu pro takové zřetěžené zpracování instrukcí, se nazývá skalární procesor. Zřetěžené zpracování instrukcí je možné provádět i ve více než jedné frontě. Procesor, který má více než jednu frontu pro zřetěžené zpracování, se nazývá superskalární procesor. Díky této technice je možné, aby procesor během jednoho taktu zpracoval více než jednu instrukci.

Čipová sada

Čipová sada je jádrem základní desky. Jejím úkolem je zajištění komunikace mezi jednotlivými částmi základní desky a spolupráce s dalšími komponenty počítače. Čipová sada limituje parametry periferních zařízení a komponent počítače, které se připojují k základní desce. Tím ovlivňuje také výkonnostní profil sestaveného počítače.

Čipová sada se dělí na dva základní moduly, z nichž první je systémový řadič (tzv. North Bridge, System Controller) a druhý řadič periférií (South Bridge, Peripheral Bus Controller). Systémový řadič je blíže procesoru a zajišťuje rychlé přesuny dat mezi klíčovými oblastmi počítače. Sběrnici FSB (Front Side Bus) je připojen k procesoru a zajišťuje tak jeho veškerou interakci s ostatními částmi základní desky i PC samotného. K systémovému řadiči je také připojena paměťová sběrnice. Její rychlost bývá většinou stejná nebo vyšší, než je rychlost FSB. Druhým modulem čipové sady je řadič periférií, který se stará o připojení periférií k základní desce. Propojení mezi oběma moduly zajišťuje sběrnice PCI, jejíž sloty jsou na základní desce vyvedeny pro připojení rozšiřujících karet.

K perifernímu řadiči je připojen především diskový systém. Parametry modulu, resp. celé čipové sady, rozhodují o tom, jakou max. přenosovou rychlost budou moci disky připojené k základní desce využívat. Periferní řadič dále ovládá USB, sériové a paralelní porty a zajišťuje služby BIOSu.

Paměti

Paměť počítače je zařízení, které slouží k ukládání programů a dat, s nimiž počítač pracuje. Paměti lze rozdělit do tří základních skupin:

- ▶ Registry - paměťová místa na čipu procesoru, která jsou používána pro krátkodobé uchování právě zpracovávaných informací.
- ▶ Vnitřní (interní, operační) paměti - paměti osazené většinou na základní desce. Bývají realizovány pomocí polovodičových součástek. Jsou do nich zaváděny právě spouštěné programy (nebo alespoň jejich části) a data, se kterými pracují.
- ▶ Vnější (externí) paměti - paměti realizované většinou za pomoci zařízení používajících výměnná média v podobě disků. Záznam do externích pamětí se provádí většinou na magnetickém nebo optickém principu. Slouží pro dlouhodobé uchování informací a zálohování dat.

Základní parametry pamětí jsou:

- ▶ kapacita - množství informací, které je možné do paměti uložit,
- ▶ přístupová doba - doba, kterou je nutné čekat od zadání požadavku, než paměť zpřístupní požadovanou informaci,
- ▶ přenosová rychlost - množství dat, které lze z paměti přečíst (do ní zapsat) za jednotku času,
- ▶ staticčnost / dynamičnost,
 - statické paměti - uchovávají informaci po celou dobu, kdy je paměť připojena ke zdroji elektrického napětí,
 - dynamické paměti - zapsanou informaci ztrácí i v době, kdy jsou připojeny k napájení (informaci je nutné periodicky obnovovat, aby nedošlo k jejich ztrátě),
- ▶ destruktivnost při čtení,
 - destruktivní při čtení - přečtení informace z paměti vede ke ztrátě této informace (přečtená informace musí být následně po přečtení opět do paměti zapsána),
 - nedestruktivní při čtení - přečtení informace žádným negativním způsobem tuto informaci neovlivní,
- ▶ energetická závislost,
 - energeticky závislé - uložené informace jsou po odpojení od zdroje napájení ztraceny,
 - energeticky nezávislé - uchovávají informace i po dobu, kdy nejsou připojeny ke zdroji elektrického napájení,
- ▶ přístup,
 - sekvenční - před zpřístupněním informace z paměti je nutné přečíst všechny předcházející informace,
 - přímý - je možné zpřístupnit přímo požadovanou informaci,
- ▶ spolehlivost - střední doba mezi dvěma poruchami paměti,
- ▶ cena za bit - cena, kterou je nutno zaplatit za jeden bit paměti.

3.2 Komunikační rozhraní PC

Počítače třídy PC obsahují řadu rozhraní pro komunikaci zařízení uvnitř počítače i komunikaci s vnějšími zařízeními. Základní dělení rozhraní je provedeno podle toho zda jsou data předávána v celé šířce slova, tzn. paralelně, nebo postupně bit po bitu (tzn. sériově).

Sériová rozhraní zahrnují především:

- ▶ sériový port,
- ▶ USB,
- ▶ FireWire,
- ▶ IrDA.

Paralelní rozhraní zahrnují především:

- ▶ paralelní port,
- ▶ IDE,
- ▶ SCSI.

Sériová rozhraní a paralelní port jsou diskutovány v kapitolách 3.2.1 a 3.2.2. Rozhraním IDE a SCSI jsou věnovány samostatné kapitoly.

3.2.1 Sériová rozhraní

Sériový port

Asynchronní sériové rozhraní bylo navrženo jako port pro komunikaci mezi dvěma zařízeními. Asynchronní znamená, že není použito synchronizace nebo hodinového signálu, takže jednotlivé znaky se posílají s libovolným časováním.

Každý znak, poslaný přes sériové rozhraní, je definován standardním signálem počátku a konce: znakem počátku je tzv. start-bit, mající hodnotu 0, a znakem konce je jeden či dva tzv. stop-bity. Po každém start-bitu tedy následuje dalších 8 bitů (1 byte), tvořících vlastní přenášená data. Přijímající zařízení pak jednotlivé znaky rozpoznává právě podle signálů počátku a konce. Celé rozhraní je tedy orientováno znakově. Režie přenosu dat, daná počátečními a koncovými signály, přes sériové rozhraní je asi 20%.

Pojem sériový znamená, že data jsou posílána po jednom vodiči, kde každý bit je zařazen do bitového toku. Sériové porty počítače jsou v PC obvykle umístěny v jednom z multifunkčních adaptérů nebo na kartách, obsahujících alespoň paralelní port. Součástí základní desky moderních počítačů je vestavěný čip Super I/O, vytvářející jedno či dvě sériová rozhraní. To znamená, že u těchto počítačů nejsou nutné žádné rozšiřující adaptéry.

K sériovým portům lze připojit různá zařízení jako např. modemy, plottery, tiskárny, další počítače, čtečky čárových kódů či obvody pro řízení dalších zařízení.

Sériová komunikace je definována standardem Reference Standard 232 revize C (RS-232C). Z tohoto důvodu se někdy standardní sériové porty nazývají porty RS-232.

Rozhraní USB

Hlavním cílem při vývoji sběrnice USB (Universal System Bus) bylo umožnit připojování externích zařízení k počítači bez nutnosti nastavení systémových zdrojů a bez nutnosti restartu počítače. Dalším cílem bylo vyhnout se používání dalších rozšiřujících karet.

První verze standardu USB označená 1.0 by zveřejněna v roce 1996, zanedlouho i její první modifikace USB 1.1. Sběrnice USB 1.1 podporuje přenosovou rychlost až 12 Mb/s (1,5 MB/s), přičemž data jsou přenášena po jednoduchém kabelu, tvořeném čtyřmi vodiči. Sběrnice umožňuje připojení až 127 zařízení a je založena na topologii stupňovaných hvězdic (tiered-star). Základem této topologie jsou rozbočovače, které mohou být umístěny v samotném počítači, v jakémkoliv jiném zařízení připojeném ke sběrnici, či mohou být tvořeny samostatnými zařízeními. Všechna připojená zařízení však sdílejí jedno přenosové pásmo o šířce 1,5 MB/s. Pro připojení pomalejších zařízení, jakými jsou např. myši, klávesnice apod., má sběrnice USB vyhrazený subkanál o přenosové rychlosti 1,5 Mb/s.

Data přenášená po sběrnici USB jsou kódována metodou NRZI (Non Return Zero Invert). Metoda je založena na tom, že jednotlivé bity jsou reprezentovány opačnými a střídajícími se vysokými a nízkými úrovněmi napětí. Přitom mezi jednotlivými bity nenásleduje žádný návrat k nulovému napětí (nebo nějakému jinému referenčnímu napětí). U kódování NRZI je 1 přenášena bez jakékoliv změny napětí a 0 je představována změnou napětí. To znamená, že řetězec 1 nepůsobí žádnou změnu napětí, zatímco při přenosu 0 bude napětí po každé přenesené 0 znovu změněno. Tento způsob přenosu dat eliminuje potřebu dodatečného časovacího signálu.

Podle specifikace USB lze jednotlivá zařízení rozdělit do tří kategorií: rozbočovače, koncová zařízení a kombinovaná zařízení. Rozbočovače umožňují připojení dalších zařízení ke sběrnici, koncová zařízení jsou ta, v nichž je sběrnice ukončena (klávesnice, myš, fotoaparát apod.). Počáteční porty v počítači jsou počátkem celé sběrnice, a proto jsou nazývány kořenovým rozbočovačem.

Výhodou sběrnice USB je podpora technologie Plug-and-Play, t.j. snadná instalace zařízení bez nutnosti konfigurace. Celá sběrnice USB využívá pouze jediné přerušení. Jednotlivá zařízení mohou být také odpojována či připojována během chodu počítače.

V současné době existuje také sběrnice USB 2.0, nabízející až 40násobné zvýšení přenosové rychlosti. Sběrnice využívá stejné kabely a konektory, pouze rozbočovače musí být navrženy pro tuto verzi. Je zaručena plná kompatibilita směrem k nižší verzi. Pokud některé ze vzájemně komunikujících zařízení nepodporuje vyšší přenosové rychlosti, budou tato dvě zařízení spolu komunikovat sníženou rychlostí, tj. 1,5 MB/s.

Rozhraní FireWire

Standard IEEE-1394 (též i.Link či FireWire) definuje vysokorychlostní sériovou sběrnici. Přenosová rychlost tohoto rozhraní je max. 400 Mb/s. V současné době definuje standard IEEE-1394 tři varianty sběrnice, odlišující se přenosovou rychlostí: 100 Mb/s, 200 Mb/s a 400 Mb/s. Kabely sběrnice IEEE-1394 jsou založeny na technologii kabelů herní konzole Nintendo a používají i obdobné konektory: 4 vodiče slouží pro přenos dat, zatímco další dva vodiče zajišťují napájení. IEEE-1394 plně odpovídá standardům technologie Plug-and-Play, tj. jednotlivá zařízení lze připojovat za chodu počítače a není nutné je nijak konfigurovat systém. IEEE-1394 nevyžaduje žádné zakončování sběrnice. Jednotlivá připojená zařízení mohou ze sběrnice odebírat proud až 1,5 A.

K jediné kartě s rozhraním IEEE-1394 lze připojit až 63 uzlů sběrnice (zařízení), a to v jednom řetězci nebo větvením. Přitom ke každému uzlu lze připojit maximálně 16 zařízení. Pokud ani toto není dostatečné množství, standard IEEE-1394 definuje možnost propojení až 1 023 sběrnic pomocí mostů, což znamená, že teoretický max. počet připojených uzlů je 64000.

Rozhraní IrDA

IrDA je standard vytvořený konsorciem IrDA (Infrared Data Association), který definuje metodu bezdrátového přenosu dat pomocí infračerveného záření. IrDA definuje standardy jak fyzických koncových zařízení, tak komunikačních protokolů.

Zařízení IrDA komunikují pomocí záření vydávaného infračervenými LED diodami s vlnovými délkami vyzařovaného světla $875 \text{ nm} \pm 30 \text{ nm}$. Přijímačem jsou PIN fotodiody

IrDA zařízení podle normy IrDA 1.0 a 1.1 pracují do vzdálenosti 1.0 m při bitové chybovosti BER (Bit Error Ratio, poměr chybně přenesených bitů ke správně přeneseným) 10^{-9} a maximální úrovni okolního osvětlení 10klux (zhruba denní svit slunce). Tyto hodnoty jsou definovány pro nesouosost vysílače a přijímače 15° , pro jednotlivé optické prvky se měří výkon do 30° . Max. rychlost přenosu rozhraní IrDA v. 1.0 je od 2400 do 115200 kb/s při využití pulsní modulace 3/16 délky původní doby trvání bitu. Formát dat je stejný jako u standardního sériového portu, tedy asynchronně vysílaná slova se start-bitem a stop-bitem.

IrDA zařízení využívají těchto komunikačních protokolů: IrLAP (Infrared Link Access Protocol), IrLMP (Infrared Link Management Protocol), Tiny TP (Tiny Transport Protocols), IrOBEX (Infrared Object Exchange Protocol), IrTran-P (Infrared Transfer Picture Specification).

3.2.2 Paralelní rozhraní

Fyzické vlastnosti paralelního portu včetně režimů přenosů dat a elektrických parametrů byly nejnověji stanoveny standardem IEEE 1284 z roku 1994. Standard IEEE 1284 definuje několik režimů činnosti portu, podpora všech režimů však není vyžadována.

Standard IEEE-1284 předpokládá, že ke komunikaci budou využívány standardní paralelní kabely, ale také kabely typu kroucené dvoulinky. Další součástí uvedeného standardu je definice konektorů pro paralelní porty, zahrnující kromě dvou běžných typů (nazývaných typ A a typ B) i konektor typu C, jehož rozměry byly sníženy díky výraznému zmenšení vzdáleností mezi jednotlivými vývody. Typ A odpovídá konektoru DB25 (25 vývodů), který má většina PC, typ B je běžný konektor Centronics (36 vývodů), nacházející se obvykle na tiskárně a konečně typ C (36 vývodů) je novější typ konektoru, používaný na některých typech tiskáren (zejména firmy Hewlett-Packard).

Z hlediska komunikace standard IEEE 1284 definuje celkem pět různých režimů, přičemž důraz je kladen na režimy EPP a ECP (viz níže). Některé z režimů slouží pouze pro příjem, zatímco jiné pouze pro vysílání. Různé kombinace těchto pěti režimů jsou základem pro definici čtyř různých typů paralelních portů.

Standardní paralelní porty (SPP)

SPP (Standard Parallel Port) jsou původní paralelní porty, určené pouze k vysílání dat. Protože se však na trhu objevila zařízení vyžadující i příjem dat (např. postscriptové tiskárny), byl port upraven tak, aby mohl data i přijímat. Výsledkem byl port umožňující 8bitové vysílání dat (kompatibilní režim) a 4bitový příjem dat (režim Nibble). U moderních PC však tyto porty nejsou využívány.

Obousměrné paralelní porty

Obousměrný paralelní port byl původně součástí počítačů řady PS/2 firmy IBM. Tyto porty jsou u levnějších počítačů používány dodnes. Port umožňuje kvalitní komunikaci mezi počítačem a zařízením připojeným k jeho paralelnímu portu.

Paralelní porty typu EPP

Jedná se o standard vyvinutý firmami Intel, Xircom a Zenith Data Systems a označovaný někdy jako rychlý paralelní port. Lze říci, že všechny soudobé počítače, u nichž je paralelní port vytvářen obvodem Super I/O, podporují port typu EPP. Hlavním cílem při návrhu tohoto portu bylo zvýšení přenosové rychlosti, které by umožňovalo připojit přes paralelní port taková zařízení, jako jsou páskové mechaniky, síťové adaptéry či přenosné pevné disky.

Byly publikovány celkem dvě verze specifikace portu EPP verze 1.7 (původní) a verze 1.9 (součást IEEE 1284).

Paralelní porty typu ECP

Paralelní port ECP byl společně vyvinut firmami Microsoft a Hewlett-Packard. Stejně jako EPP i ECP nabízí výrazné zvýšení přenosové rychlosti portu. Hlavním cílem při vývoji ECP však nebyla podpora přenosných zařízení připojovaných k paralelnímu portu. Port byl navržen proto, aby umožnil komunikaci s vysoce výkonnými tiskárnami a skenery s velmi vysokým rozlišením. Kromě toho port ECP je založen na využívání jednoho z kanálů přímého přístupu do paměti (DMA kanál), což může vést ke vzniku konfliktů při využívání systémových zdrojů.

Většina soudobých počítačů je vybavena obvodem Super I/O schopným vytvářet jak porty typu EPP, tak i ECP, přičemž konkrétní typ je možné si vybrat v programu Setup systémového BIOSu. Platí však, že nejvyšší přenosovou rychlost vám zajistí porty ECP.

Obousměrné paralelní porty se velmi často využívají k přenosu dat mezi dvěma počítači, např. mezi přenosným a stolním počítačem. Jsou-li oba systémy vybaveny porty typu EPP/ECP, může komunikace probíhat rychlostí až 2 MB/s. K propojení dvou počítačů přes paralelní porty je však nutný speciálně upravený kabel, nazývaný také null-modemový kabel.

3.2.3 Rozhraní IDE

Zkratka IDE (Integrated Drive Electronics) je poměrně obecná a může jí být označena kterákoliv (disková) mechanika s vestavěným řadičem. Rozhraní IDE se v té podobě, v jaké se dnes využívá, označuje správně ATA (AT Attachment) a je standardizováno organizací ANSI. Lépe řečeno, jedná se o vyvíjející se standard, který má několik verzí. Písmena AT ve zkratce ATA jsou odvozena od původního IBM AT, u kterého byla poprvé použita 16-bitová sběrnice ISA. To znamená, že zkratka ATA obecně označuje jakýkoliv pevný disk, který se připojuje přímo k některé verzi sběrnice AT neboli k 16-bitové sběrnici ISA.

V současnosti se používá mnoho různých druhů disků s integrovanými řadiči. Obvykle je u takového disku řadič pevně připojen k samotnému disku a tato kombinace disku a řadiče se pak připojuje ke konektoru sběrnice umístěnému na základní desce či na rozšiřující kartě. Toto pevné propojení disku s řadičem má několik výhod: nemusí se instalovat zvláštní vodiče pro signály a data, vedoucí z řadiče do disku, je snížen celkový počet součástí a vodiče pro jednotlivé signály jsou kratší a tedy odolnější vůči rušení. Díky integraci řadiče a disku je také možné zvýšení rychlosti převodu digitálního signálu na analogový a zvýšení kapacity disku.

Samotný konektor IDE, nacházející se na základních deskách, je v podstatě jen ochuzenou verzí standardního konektoru sběrnice. Obvykle má tento konektor 40 vývodů z 98, které tvoří standardní konektor 16-bitové sběrnice ISA. Existují však i menší, 2" mechaniky ATA, používající konektor se 44 vývody. Tyto konektory totiž obsahují další vývody pro vodiče napájení a konfigurace. Vývody konektoru a jejich význam jsou stejné i v

případě, že rozhraní ATA je připojeno k obvodu řadiče periférií čipové sady základní desky a pracuje rychlostí sběrnice PCI.

Je také nutné zdůraznit, že pojmem IDE může být označena jakákoliv mechanika, u které je řadič vestavěn do mechaniky. Naopak pojmem ATA lze označit pouze některé typy mechanik s rozhraním IDE. Protože však v současné době jsou nejvíce využívány mechaniky ATA, oba pojmy se často zaměňují, a to i přesto, že z technického hlediska tato záměna není správná. Zřejmě největší výhodou mechanik IDE ve srovnání s novějšími alternativami, jako např. rozhraním SCSI nebo FireWire, je cena. Náklady na pořízení mechaniky IDE jsou výrazně nižší než náklady na pořízení odděleného disku a řadiče.

V dnešní době se používá jediná verze rozhraní IDE - ATA IDE, určená pro 16-bitovou sběrnici ISA. Byly také vyvinuty a specifikovány rychlejší a výkonnější verze rozhraní ATA IDE, které se pak označují číslicí (např. ATA-2, ATA-3 apod.). Jiným označením pro tyto výkonnější varianty jsou EIDE (Enhanced IDE), Fast-ATA, Ultra-ATA či Ultra-DMA.

Rozhraní ATA IDE

Za první disky s rozhraním ATA IDE lze považovat disky vytvořené v roce 1986 firmami CDC, Western Digital a Compaq. Jednalo se o disky CDC, mající kapacitu 40 MB, k nimž byly pevně připojeny řadiče Western Digital a pomocí konektoru se 40 výhody byly připojeny do počítačů Compaq 386.

Později byl návrh konektoru a rozhraní disku předložen standardizačním výborům organizace ANSI, která ve spolupráci s dalšími výrobci pevných disků některé prvky původního návrhu upravila a posléze v roce 1989 publikovala pod označením CAM ATA (Common Access Method AT Attachment). Tento standard však není zcela striktní a úplný: některé jeho části si může výrobce disku upravit podle svého. Typickým důsledkem jsou pak potíže při nízkoúrovňovém formátování pevných disků – většinou je nutné použít speciální program přímo od výrobce konkrétního disku, neboť právě při tomto formátování se používají některé příkazy pro přepis hlaviček a sektorů, které si každý výrobce může definovat sám.

Standardy ATA

Rozhraní ATA a jeho vývoj je dnes řízeno nezávislou skupinou představitelů významných výrobců PC, disků a dalších komponent. Tato skupina je označena jako Technical Committee T13 a odpovídá za standardy všech rozhraní, vztahujících se k rozhraní AT Attachment. Výbor T13 je součástí amerického výboru National Committee on Information Technology Standards (MCITS), pracujícího podle pravidel schválených americkou organizací pro standardizaci American National Standard Institute (ANSI).

Celkově bylo zatím standardizováno 6 variant rozhraní ATA:

- ▶ ATA-1 (1986-1994),
- ▶ ATA-2 (1996, Fast-ATA, Fast-ATA-2 či EIDE),
- ▶ ATA-3 (1997),
- ▶ ATA-4 (1998, Ultra-ATA/33),
- ▶ ATA-5 (1999, Ultra-ATA/66),
- ▶ ATA-6 (2000, Ultra-ATA/100),
- ▶ ATA-7 (2001, Ultra-ATA/133).

Každá varianta tohoto rozhraní je navržena tak, aby byla zpětně kompatibilní. Novější verze ATA jsou obvykle vytvářeny na základě předcházející verze, k níž jsou přidány další, rozšiřující funkce.

Rozhraní ATA-1

Rozhraní ATA-1 se využívalo od roku 1986, avšak standardizováno bylo až r. 1994. Mezi základní charakteristiky specifikace ATA-1 patří:

- ▶ konektory a vodiče se 40, resp. 44 vývody,
- ▶ možnost konfigurovat disk jako Master/Slave či možnost nastavit roli disku podle toho, ke kterému konektoru je připojen,
- ▶ časování signálů pro základní režimy PIO a DMA,
- ▶ překlady parametrů disků CHS (Cylinder Head Sector) a LBA (Logical Block Address).

Standard ATA umožňuje provozovat v jednom systému dva disky, které jsou k rozhraní připojené za sebou. Primární (řídící) disk (disk 0) se nazývá master, zatímco sekundární disk (disk 1) se nazývá slave. Funkci určitého disku lze určit nastavením speciálních spojek na tělese disku.

Rozhraní ATA-2

Toto rozhraní bylo standardizováno roku 1996 a představuje významnou modernizaci původního standardu ATA. Pravděpodobně nejdůležitější změna je spíše filozofického rázu: rozhraní ATA-2 bylo definováno tak, aby je bylo možné využít jako obecné rozhraní mezi hostitelským systémem a jakýmkoliv zařízením pro ukládání dat, nikoliv jen pevným diskem. Mezi další významné úpravy obsažené ve standardu ATA-2 patří:

- ▶ podpora rychlejších režimů PIO a DMA,
- ▶ podpora správy napájení,
- ▶ podpora vyjímatelných zařízení,
- ▶ podpora karet standardu PC Card (PCMCIA),
- ▶ podpora pevných disků o kapacitě až 136,9 GB,
- ▶ definice standardních metod překladu CHS/LBA (Cylinder Head Sector / Logical Block Address) pro disky o kapacitě až 8,4 GB.

Rozhraní ATA-3

Jedná se jen o menší úpravu standardu ATA-2, uveřejněnou v roce 1997. V podstatě jedinými podstatnými novinkami jsou:

- ▶ vyřazení všech 8bitových přenosových protokolů využívajících kanály DMA,
- ▶ přidání podpory technologie S.M.A.R.T. (Self-Monitoring, Analysis, and Reporting Technology), umožňující určité předpovídání snížení výkonu zařízení,
- ▶ přidána možnost ochrany přístupu k zařízení heslem,
- ▶ přijata doporučení zaměřená na snížení vlivu rušení při vyšších přenosových rychlostech.

Rozhraní ATA-2 a ATA-3 jsou často nazývána Enhanced IDE (EIDE), což je ryze marketingový termín, prosazený firmou Western Digital. Naopak firma Seagate spíše používala pojmy jako Fast-ATA či Fast-ATA-2, které později přejala i společnost Quantum.

Rozhraní ATA-4

Roku 1998 bylo standardizováno rozhraní ATA-4, které se stalo další významnou modernizací rozhraní ATA. Součástí standardu se totiž stala i vlastnost Packet Command, která byla stěžejní součástí definice odděleného rozhraní ATAPI (AT Attachment Packet Interface). Díky tomu je možné k rozhraní ATA-4 připojovat mechaniky CD-ROM, disketové mechaniky typu LS-120, páskové mechaniky apod. Druhou významnou novinkou tohoto standardu se stala podpora přenosové rychlosti 33 MB/s, kterou rozhraní umožňuje v režimu Ultra-DMA či Ultra-ATA. Níže následuje přehled všech podstatných změn tohoto rozhraní:

- ▶ podpora režimu Ultra-DMA, umožňujícího přenášet data rychlostí až 33 MB/s (režim je označován jako UDMA/33 či Ultra-ATA/33),
- ▶ integrovaná podpora rozhraní ATAPI,
- ▶ podpora vylepšené správy napájení,
- ▶ definice volitelného kabelu s 80 vodiči a konektory se 40 vývody, zlepšujícího odolnost kabelu vůči rušení,
- ▶ podpora karet standardu Compact Flash.

Kromě zvýšení přenosové rychlosti přináší režim UDMA/33 i snížení zatížení procesoru, což vede k dalšímu zvýšení výkonu systému. Součástí této specifikace se stala i podpora příkazů pro práci s frontami, čímž se rozhraní ATA-4 přiblížilo rozhraní SCSI-2 (viz kapitola 3.2.4). Tyto příkazy vylepšují podporu multitaskingu, neboť jsou schopně zpracovat současné požadavky několika programů na přenosy dat.

Rozhraní ATA-5

Tato verze rozhraní ATA byla zveřejněna na počátku roku 2000 s následujícími úpravami:

- ▶ podpora režimu Ultra-DMA umožňujícího přenášet data rychlostí 66 MB/s (režim je označován jako UDMA/66 či Ultra-ATA/66),
- ▶ kabel s 80 vodiči podmínkou pro režim UDMA/66,
- ▶ automatická detekce kabelu se 40 či 80 vodiči.

Přenosová rychlost rozhraní ATA-5 byla oproti ATA-4 zdvojnásobena díky zkrácení časů pro nastavení a zvýšení rychlosti rozhraní. Avšak zvýšení rychlosti rozhraní s sebou přineslo zvýšení interference, která znemožnila používání standardních kabelů se 40 vodiči. Jak již bylo řečeno v předcházejících částech kapitoly, kabel obsahuje navíc 40 vodičů pro uzemnění, čímž se zvyšuje ochrana jednotlivých signálů proti rušení. Tento kabel je však možné použít i se staršími zařízeními nepodporujícími režimy UDMA. Současně se u tohoto kabelu stala standardem podpora signálu Cable Select a byla také normalizována barva jednotlivých konektorů. Modrý konektor by měl být připojen k hostitelskému rozhraní, tj. k základní desce. Černý konektor, umístěný na opačném konci kabelu, je určen pro řídicí mechaniku (master) a šedý konektor by měl být využit pro sekundární mechaniku (slave).

Spolehlivost přenosových režimů UDMA je zvýšena v důsledku využívání mechanismu detekce chyb, označovaného zkratkou CRC (Cyclic Redundancy Checking). Jedná se o algoritmus používaný k výpočtu kontrolních součtů v proudu dat. V případě režimů UDMA je kontrolní součet vypočítáván jak na straně hostitelské, tak na straně disku. Je-li zjištěn nějaký rozdíl mezi daty přijatými hostitelem a vyslanými diskem, může být hostitel požádán o snížení přenosové rychlosti nebo může hostitel znovu požádat o zaslání shodných dat.

Omezení kapacity disků

Jednou z nevýhod rozhraní ATA je omezení maximální kapacity disku na 136,9 GB. V závislosti na použitém BIOSu se tato maximální velikost disku může ještě snížit na 8,4 GB či dokonce jen na 528 MB. Tyto hodnoty jsou dány kombinací omezení daných rozhraním ATA a BIOSem.

3.2.4 Rozhraní SCSI

SCSI (Small Computer System Interface) označuje rozhraní, používané k připojování obecných zařízení k počítači. Principiálně je jediná sběrnice SCSI schopna podporovat až 8 či 16 fyzických zařízení, z nichž každému je přiděleno identifikační číslo (SCSI ID). Ke sběrnici je vždy připojen řadič SCSI, který má funkci brány mezi sběrnici SCSI a systémovou sběrnici PC. Každé zařízení na sběrnici má svůj vestavěný řadič. Sběrnice SCSI pak nekomunikuje přímo se samotnými zařízeními, ale s řadiči těchto zařízení. Některé řadiče jsou pak schopny po povelu hosta převzít kontrolu nad sběrnici a komunikovat s ostatními zařízeními.

Standardy rozhraní SCSI definují fyzické i elektrické parametry paralelní vstupně/výstupní sběrnice, použité k připojení všech zařízení a počítače do jednoho řetězce. První standard SCSI byl schválen roku 1986, o osm let později následoval standard SCSI-2 a první část definice SCSI-3 byla zveřejněna v roce 1995. Standard SCSI-3 má mnoho různých variant.

V původním standardu SCSI-1 byly mnohé příkazy definovány jako volitelné. Nebylo tedy možné se spolehnout na to, že určité zařízení bude tyto příkazy podporovat. Proto se výrobci zařízení SCSI dohodli na sadě standardních příkazů CCS (Common Command Set), jejichž podpora byla vyžadována. Tato sada příkazů se pak stala základem standardu SCSI-2.

Standard SCSI-2 navíc podporuje podstatně více typů zařízení (např. mechaniky CD-ROM, páskové mechaniky, vyjímatelná zařízení apod.). Současně byly definovány i rychlejší varianty Fast SCSI-2 (8-bitová), a Wide SCSI-2 (16-bitová). Standard SCSI-2 také zavedl podporu řazení příkazů do fronty, což znamená, že zařízení je schopno přijmout několik různých příkazů současně, uložit je do fronty a následně je vykonat v tom pořadí, které je z hlediska zařízení optimální. Specifikace SCSI-2 také obsahuje některé volitelné součásti a ne všechna zařízení SCSI-2 je podporují. K těmto volitelným součástem patří např. zmíněný rychlejší režim Fast SCSI-2.

Standard SCSI-3 sestává z několika dalších specifikací. Jednou z nich je SPI (SCSI Parallel Interface), definující paralelní propojování SCSI zařízení. Doposud byly zveřejněny čtyři verze tohoto standardu, kterými jsou SPI, SPI-2, SPI-3 a SPI-4. Přitom verze SPI-4 zatím existuje pouze v návrhu. Jednotlivé verze SPI se pak odlišují především přenosovou rychlostí.

Rozhraní SCSI je zpětně i dopředně kompatibilní což znamená, že je-li k pomalejšímu hostitelskému adaptéru připojen rychlejší pevný disk, bude celá soustava pracovat ne menší z obou rychlostí. Pomocí speciálních adaptérů je dokonce možné připojit k jedné sběrnici zařízení o různé datové šířce (8 či 16 bitů).

4 Operační systémy

4.1 Co je to operační systém

Srdcem každého počítače je skupina programů, nazývajících se obvykle **operační systém** (OS). Představuje souhrn mechanismů a nástrojů, které slouží k ovládání systémových prostředků, řízení vstupně/výstupních zařízení a ke spouštění programů. Společně s OS je k dispozici skupina standardních utilit (podprogramů), které zajišťují běžné funkce, jako kopírování souborů a výpis obsahu adresáře.

OS je souhrn programů, které

- ▶ inicializují hardware počítačového systému,
- ▶ provádí základní rutiny obsluhy zařízení,
- ▶ zajišťuje správu úloh,
- ▶ zajišťuje integritu počítačového systému.

Existuje velké množství typů OS, jejich složitost a rozsáhlost je různá podle typu funkcí, které zajišťuje.

Příklad 4.1: *Velmi jednoduché OS, například v domácích bezpečnostních systémech, jsou uloženy v pamětech typu ROM. Jejich práce začíná zapnutím systému (přivedením napájecího napětí do hardware). Jejich prvním úkolem je resetování (a často i testování) hardwarových senzorů a hlásičů, po té OS vstupují do procesu kontinuálního monitorování vstupních senzorů. Změna stavu senzorů obvykle vyvolá spuštění příslušné rutiny v OS.*

4.1.1 Funkce operačního systému

Operační systém pracuje především jako:

- ▶ správce zdrojů (resource manager), a
- ▶ virtuální počítač (virtual machine).

Správa zdrojů je funkcí, která zajišťuje správné a efektivní nakládání se systémovými zdroji. Zdroje jsou V/V zařízení, soubory, procesor, paměť, apod. Operační systém vlastní jednotlivé systémové zdroje a přiděluje je a odebírá jednotlivým procesům.

Virtuální počítač je pojem charakterizující způsob práce s počítačovým systémem. Operační systém skrývá detaily ovládání jednotlivých zařízení a definuje standardní rozhraní pro volání systémových služeb. Programátor se může věnovat vlastní úloze a nemusí znovu programovat V/V operace. Program může díky "odizolování" od konkrétních zařízení pracovat i se zařízeními, o které jeho autor v době vytváření programu neznal (program se o detaily ovládání V/V zařízení nestará).

4.1.2 Moduly operačních systémů

Ve velkých více-uživatelských počítačových systémech s mnoha terminály je nutné použít složité OS. Takové OS bývají obvykle modulové, tzn. skládají se z nezávislých komunikujících částí. Některé hlavní moduly s jejich částmi jsou např.:

Modul přidělování procesoru

Plánovač úloh - sleduje a eviduje stav všech úloh v systému, které si uchovává ve frontě. Jednotlivé úlohy mohou mít různou prioritu, např. systémové úlohy mají vždy vyšší prioritu než uživatelské.

Plánovač procesu - sleduje frontu procesu a rozhoduje, který proces a na jak dlouho dostane přidělen procesor.

Dispečer - sleduje procesor a stav procesů.

Modul přidělování periférií

V/V dispečer - sleduje stav vstupně/výstupních periferních zařízení, kanálů, a řídicích jednotek.

V/V plánovač - rozhoduje o efektivním přidělení periferních zařízení. Pokud mají být zařízení sdílena, rozhoduje o tom, kdo ho dostane a v jakém rozsahu. Přiřazuje periférii a zahajuje V/V operaci. Požaduje navrácení prostředků (většinou se ale V/V operace ukončuje automaticky).

Systém správy souborů

Sleduje každý soubor, jeho umístění na disku, stav použití, apod. Rozhoduje, komu bude soubor poskytnut, realizuje požadavky na ochranu dat a operace přístupu k souborům. Otevírá a uzavírá soubory.

Jádro (Kernel)

Srdcem operačního systému je jádro (jinak kernel). Jeho úkolem je zajišťovat základní funkce systému, většinou na hardwarové úrovni. Některé důležité funkce jádra jsou:

- ▶ přepínání mezi procesy a jejich ovládání,
- ▶ ovládání hardwarových komponentů,
- ▶ správa a ovládání paměti,
- ▶ plánování,
- ▶ komunikace mezi procesy,
- ▶ zpracování přerušení.

Příklad 4.2: *Jednoduchý bezpečnostní systém z Příklad 4.1 nebude využívat všechny funkce jádra uvedené výše. Jde o jednoúlohový systém (zpracovává jediný program současně) a je ovládán většinou jen jedním uživatelem současně. Proto nebude potřeba funkce plánování úloh a procesů a také komunikace mezi procesy. Správa paměti není nutná, programy jsou většinou umístěny v paměti typu EPROM nebo ROM.*

4.2 Procesy

Procesem se v operačním systému rozumí prováděný program včetně dat. Je důležité rozlišovat proces a program. Spustí-li současně několik uživatelů též program, jejich provádění zajistí odpovídající počet procesů vytvořených jádrem. **Program** je definován jako posloupnost instrukcí bez vztahu k stavu jeho vykonávání.

4.2.1 Stavy procesu

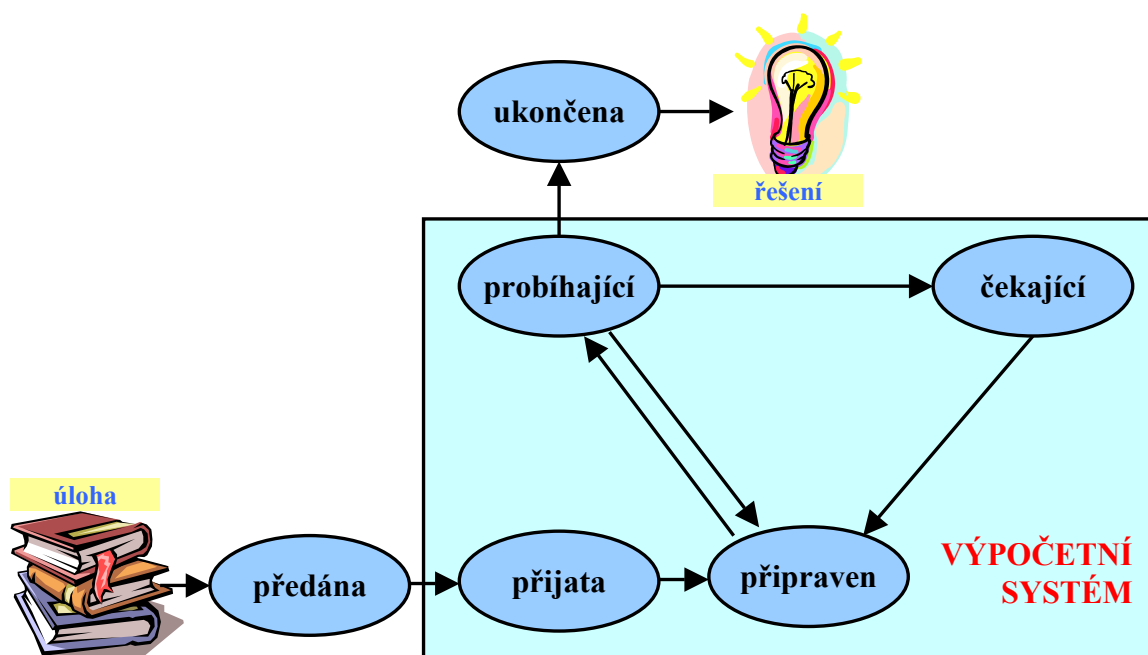
Životní cyklus procesu v operačním systému se skládá z přechodu mezi třemi hlavními stavy procesu:

4. Stav **probíhající** (angl. running) - procesu je přidělen procesor a je vykonáván.
5. Stav **čekající** (angl. waiting) - proces čeká na určitou událost, např. na dokončení vstupně/výstupní operace.
6. Stav **připraven** (angl. ready) - proces je připraven k vykonání a čeká pouze na přidělení procesoru.

Tyto 3 hlavní procesy nestačí pro úplný popis pohybu úlohy v operačním systému. Pro úplnost uvedme ještě další tři stavy úlohy:

7. Stav **předána** (angl. submit) - uživatel předal svou úlohu systému a ten na ni reaguje. Stav mírně archaický používaný u systémů s pomalým zaváděním dat.
8. Stav **přijata** (angl. hold) - úloha je na disku počítače ve vnitřní reprezentaci a očekává přidělení prostředků.
9. Stav **ukončena** (angl. complete) - výpočet úlohy skončil a všechny přidělené prostředky jsou uvolněny k dalšímu použití.

Úplný model výpočtu úlohy výpočetním systémem se zahrnutím stavů procesu i úlohy je uveden na **Obrázek 4.1**.



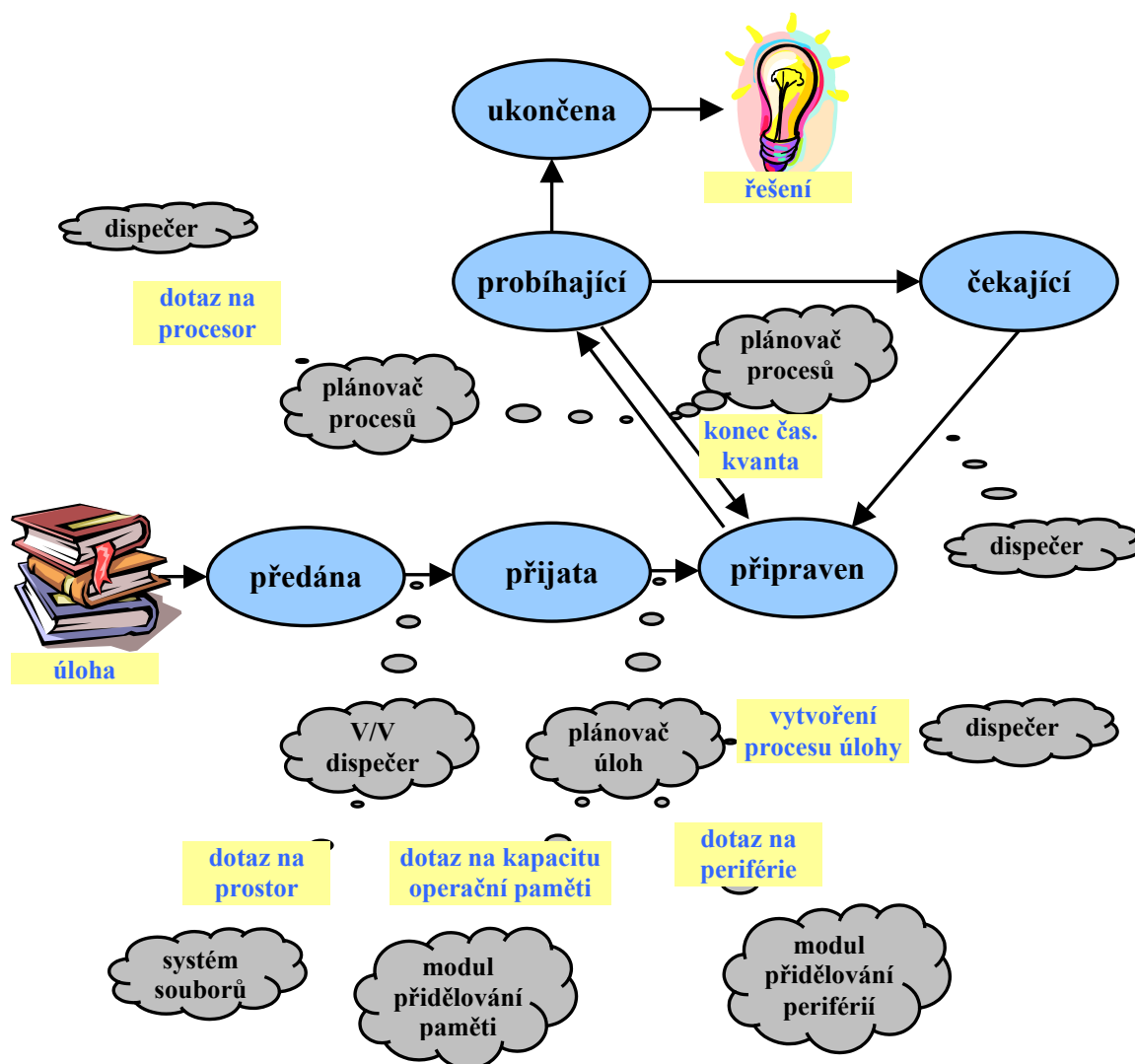
Obrázek 4.1: Model výpočtu úlohy s uvedením stavů.

4.2.2 Vykonyávání procesu

Jak proces prochází jednotlivými stavy ukazuje **Obrázek 4.2**. Elipsy označují jednotlivé stavy procesu, obláčky pak moduly OS, které zajišťují změny stavu. Celý výpočet úlohy se dá popsat slovně takto:

- ▶ uživatel předá úlohu k výpočtu (uvede úlohu do stavu předána);
- ▶ V/V dispečer úlohu za pomoci systému souborů otestuje a určí, zda je schopen vyhradit dostatečné místo na pevném disku; tím úloha přechází do stavu přijata;
- ▶ plánovač úloh se dotáže modulu přidělování paměti na požadovanou kapacitu operační paměti;
- ▶ plánovač úloh se dotáže modulu přidělování periférií na požadované periférie;
- ▶ je volán dispečer, aby vytvořil příslušné záznamy a modul přidělování paměti, aby procesu přidělil dostatečnou operační paměť;
- ▶ úloha je zavedena do paměti a jí odpovídající proces je ve stavu připraven;
- ▶ procesu se ujímá se ho plánovač procesů a ten úloze přidělí procesor, jakmile je to možné;
- ▶ procesor je úloze přidělen pouze na určitou dobu (tzv. časové kvantum). Pokud úloha není během této doby dokončena, dispečer uloží stav registrů a procesu, plánovač procesu procesor odebere a vrátí proces do stavu připraven;
- ▶ je-li úloha ve stavu probíhající a žádá o čtení ze souboru (nebo jinou V/V operaci), modul správy souborů volá modul přidělování periférií, aby zahájil čtení (nebo jinou V/V operaci). Modul přidělování periférií ji zahájí a zároveň požádá plánovač procesů, aby proces převedl do stavu čekající. Je-li V/V operace dokončena, je patřičné přerušení vyhodnoceno jako žádost o navrácení procesu do stavu připraven;
- ▶ pokud je úloha ve stavu probíhající dokončena, modul přidělování periférií jí odebere přidělené periférie, modul přidělování paměti uvolní paměť, která byla úloze alokována a plánovač procesu jí odebere procesor. Tím je výpočet úlohy ukončen.

Pozn. Některé ze zmíněných operací jsou v **Obrázek 4.2** vynechány z důvodu zachování přehlednosti schématu.



Obrázek 4.2: Životní cyklus procesu a úlohy v operačním systému.

4.3 Základy operačního systému UNIX

Vznik OS UNIX je datován do 60. let minulého století. Tehdy existoval projekt MULTICS (**M**ultiplexed **I**nformation and **C**omputing **S**ervice) organizací **AT&T** (American Telephone and Telegraph), **Honeywell**, **General Electric** a **MIT** (Massachusetts Institute of Technology).

4.3.1 Vstup uživatele do systému

login:

je text, který nově přichází uživatel čte na obrazovce terminálu. Aby se přesvědčil, že je terminál stále připojen k počítači, kde běží UNIX, stlačením klávesy Enter se výpis textu "login: " zopakuje. Uživatel může nyní psát na klávesnici svoje jméno, které je počítači známo a které uživatel přidělil správce systému. Po odeslání jména klávesou Enter se na obrazovce objeví text

password:

a UNIX opět čeká na reakci uživatele. Uživatel má obvykle svá data v počítači chráněna před neoprávněným přístupem heslem. Heslo nyní napíše na klávesnici (přitom se psaný text na obrazovce nezobrazuje). Pokud je zápis jména i hesla správný, na obrazovce se před uživatelem objeví zpráva o vstupu do systému, uvítací text atp. Výpis se zastaví a UNIX čeká na vstup od uživatele. Na obrazovce se objeví:

\$

Od tohoto okamžiku je uživatel regulérně přihlášen v systému a může interaktivně využívat všech zdrojů výpočetního systému běžného uživatele.

Po vstupu do systému je uživatel evidován jako obyčejný (znak výzvy pro komunikaci je '\$'), anebo privilegovaný (znak výzvy je '#'). Obvykle má systém jednoho privilegovaného uživatele se jménem 'root' (tzv. superuživatel). Po přihlášení pod tímto jménem má uživatel neomezená přístupová práva ke všem datům a zdrojům výpočetního systému. Obyčejní uživatelé jsou hlídáni systémem a mohou přepisovat nebo rušit jen vlastní data na základě požadavků pro operační systém. Jiná než privilegovaná a neprivilegovaná úroveň není v UNIXu zavedena.

Uživatel se může ze systému odhlásit uzavřením vstupu pro komunikaci v rámci jeho sezení, což je pro UNIX znak **Ctrl-d** (současné stisknutí klávesy **Ctrl** a **d**). Sezení je ukončeno a UNIX na terminál opět vypíše text končící na

login:

4.3.2 Základní komunikace

Uživatelé komunikují s operačním systémem pomocí příkazového řádku. Chceme-li např. znát seznam všech souborů v našem adresáři po přihlášení, můžeme psát příkazový řádek

\$ ls -a

kde znak '\$' je znak výzvy pro zápis příkazového řádku (vypisuje UNIX) a příkazový řádek je text "ls -a". Jméno příkazu je "ls" a část "-a", nazývaná volby příkazu, určuje změnu chování příkazu oproti implicitní formě. Odpovědí systému je seznam všech souborů, které jsou evidovány v adresáři. Jsme-li přihlášení vůbec poprvé, pravděpodobně nám UNIX odpoví:

.

```
..  
.profile
```

přítom každý řádek odpovídá jednomu jménu souboru.

Budeme-li si v následujícím kroku chtít zobrazit obsah souboru se jménem např. ".profile", napíšeme:

```
$ cat .profile
```

a na obrazovce terminálu je zobrazen obsah souboru ".profile" (obsahuje nastavení výchozího stavu uživatelské komunikace se systémem).

V uvedeném příkladu příkazového řádku pro zajištění našeho požadavku vytváří UNIX proces, který provádí příkaz `cat` a manipuluje se souborem ".profile". Oba dva fenomény, soubor i proces, jsou podporovány ve své existenci fenoménem třetím, kterým je jádro (angl. kernel).

Jádro je program, který je zaveden po zapnutí počítače do operační paměti a tam je spuštěn. Je programem, který z počítače na úrovni technického vybavení vytváří virtuální počítač, odstíní uživatele od přímého styku s technickým vybavením, umožňuje se strojem komunikaci ve formě vyššího programovacího jazyka a dokáže zajistit sdílení technických zdrojů několika uživatelům najednou. Pro uchování dat na vnějších paměťových médiích (zejména magnetických discích) zajišťuje přístup ke struktuře dat nazývané systém souborů (angl. file system). Pro uživatele to znamená, že může svá data ukládat do souborů a opět je ze souborů vybírat pro další zpracování. Proces je realizace akce uživatele nebo systému. Je nositelem změn v datové základně konkrétní instalace operačního systému. Procesem je např. provedení kopie souboru na tiskárnu, překlad z úrovně zdrojového textu programu do úrovně strojového kódu, spuštění databáze atd. Jsou to volání jádra (System Calls), pomocí kterých procesy interagují s jádrem. Volání jádra má z hlediska psaní programu tvar volání knihovni funkce; je to např. žádost o zpřístupnění dat z disku, žádost o vytvoření nového procesu atd.

4.3.3 Soubory

Pro UNIX je soubor posloupnost slabik bez jakékoliv další struktury. Rozlišujeme tři hlavní skupiny souborů:

- ▶ obvyčejné soubory,
- ▶ adresáře,
- ▶ speciální soubory.

Každý soubor je ve vnitřní struktuře systému souborů reprezentován i-uzlem (i-node, index node). I-uzel obsahuje všechny atributy souboru a odkazy na datovou část souboru. Atributem souboru je např. informace, zda je soubor adresářem nebo obvyčejným či speciálním souborem, kdo je jeho vlastníkem, jaká přístupová práva pro čtení nebo zápis mají uživatelé k souboru. Výpis hlavních atributů souboru ".profile" v pracovním adresáři nám zajistí příkaz

```
$ ls -l .profile
```

kde odezvou v našem případě bude

```
-rw-r----- 1 petr group 506 May 12 10:37 .profile
```

Každý řádek uvedeného výpisu se vztahuje vždy k jednomu souboru. První znak výpisu na řádku je typ souboru a může být

- ▶ - - obvyčejný soubor,
- ▶ d - adresář,
- ▶ l - nepřímý odkaz,

- ▶ c - znakový speciální soubor,
- ▶ b - blokový speciální soubor.

Ve výpisu atributů souboru následuje dále devět znaků, které určují přístupová práva k souboru, přitom jednotlivé trojice postupně pro vlastníka souboru, skupinu, do které vlastník patří, a pro ostatní uživatele. V každé trojici mohou přístupová práva znamenat

- ▶ r - čtení souboru je dovoleno,
- ▶ w - zápis do souboru je dovolen,
- ▶ x - obsah souboru může být spuštěn jako proveditelný program,
- ▶ - - přístup k souboru je zakázán.

Uživatelé jsou rozděleni do skupin. Příslušnost ke skupině určí správce systému obvykle při registraci uživatele v systému. Uživatelé stejné skupiny jsou si potom bližší z hlediska přístupu k souborům než uživatelé jiných skupin. Soubor ".profile" v našem příkladu může vlastník číst i modifikovat a členové téže skupiny číst. Ostatní práva jsou odebrána.

Po atributu přístupových práv následuje počet odkazů na soubor z různých míst systému souborů a v našem případě je jeden.

Atribut "petr" je jméno vlastníka souboru ".profile" a "group" je skupina souboru ".profile".

Následuje velikost souboru ve slabikách (506 znaků) a datum a čas poslední modifikace souboru. Poslední vypsanou položkou na řádku je jméno souboru.

Obyčejný soubor obsahuje data uživatele. Obyčejným souborem je např. soubor ".profile". Obsah obyčejného souboru můžeme zpřístupnit příkazem cat(1) (podle uvedeného příkladu čl. 2.2) nebo příkazem more(1):

```
$ more .profile
```

kdy je obsah souboru ".profile" zobrazován po stránkách. Další příkaz podobný more(1) je pg(1).

Uvedený způsob výpisu obsahu souboru je korektní, je-li obyčejný soubor souborem textovým, tj., obsahuje-li znaky z dolní části tabulky ASCII, které jsou běžně zobrazitelné na obrazovce terminálu nebo na tiskárně a jsou používány k záznamu běžně čitelného textu (zdrojové texty programů, dokumenty, telefonní seznamy, atd.). Jejich společným znakem je řazení do záznamů, ukončených znakem nového řádku. Oproti tomu soubory binární obsahují znaky z celé tabulky ASCII a jsou obvykle čitelné zvláštními aplikacemi nebo jádrem. Binární soubor je např. program v proveditelné podobě, systémové tabulky atd. Obsah binárního souboru si můžeme prohlížet pomocí příkazu od(1):

```
$ od -c a.out
```

Příkaz od(1) bez použití volby "-c" vypisuje obsah souboru "a.out" v osmičkové soustavě. Volba "-x" mění výpis z osmičkové soustavy na šestnáctkovou, "-d" na desítkovou a "-c" je výpis znakový, což znamená, že všechny běžně zobrazitelné znaky jsou vypisovány svým obsahem a ostatní ve svém ekvivalentu v osmičkové soustavě. Výpis je uskutečňován po 16 znacích na řádku, řádky jsou počítány (v osmičkové soustavě) v levém sloupci. Je-li ve výpisu samostatně na řádku uveden znak "*", znamená to opakování právě uvedeného řádku. Kolik takových znaků je opakováno, zjistíme podle počítadla znaků v levém sloupci výpisu.

Adresář je binární soubor s atributem příznaku adresáře a obsahuje seznam souborů. Po přihlášení je uživateli nastaven určitý adresář, který je označován jako domovský (home directory). Uživatel v průběhu sezení může měnit nastavení na různé adresáře. Adresář, který

má právě nastaven, je označován jako pracovní (current directory). Obsah adresáře je zobrazitelný příkazem `ls(1)`, ale ve skutečnosti jde o rozpis obsahu i-uzlů. Pracovní adresář je binární soubor a jeho obsah lze vypsat příkazem

```
$ od -c .
0000000 355 004 . \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
0000020 # \0 . \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
0000040 0 \0 . p r o f i l e \0 \0 \0 \0 \0
```

Každý řádek výpisu odpovídá jednomu souboru. První dva znaky (za počítadlem) jsou identifikací čísla i-uzlu a dále následuje 14 znaků jména souboru. Pro operační systém je rozhodující číslo i-uzlu, se kterým je spojeno v adresáři jméno souboru.

Výpis adresáře způsobem "`ls -a`" i "`od -c .`" nás upozorňuje na přítomnost dvou souborů se jménem "." a "..". Z výpisu příkazu

```
$ ls -la
total 6
drwxr-xr-x  2 petr group  48 May 12 10:37 .
drwxr-xr-x 16 bin  bin   256 May 12 10:37 ..
-rw-r----- 1 petr group 506 May 12 10:37 .profile
```

vidíme, že se jedná o adresáře. Soubor "." je nadřazeným adresářem a "." je aktuální adresář (ukazuje na i-uzel vypsaného adresáře). Systém souborů tvoří z hlediska adresářů hierarchickou strukturu, která je zajištěna položkami "." a ".." v každém adresáři. První řádek uvedeného výpisu ("total 6") nás informuje o celkovém obsazení souborů adresáře v blocích.

Hierarchie je završena adresářem označovaným "/", nazývaným také jako kořenový adresář (root directory). Kořenový adresář obsahuje podadresáře, které dále mohou obsahovat podadresáře do teoreticky libovolné hloubky. Celý systém souborů je spojen ve strom adresářů. V rámci tohoto stromu má uživatel přidělen adresář, který je mu nastaven po přihlášení do systému a kterým začíná jeho uživatelská oblast dat (jeho uživatelský podstrom). Je to jeho domovský adresář (home directory).

Výpis jména pracovního adresáře získáme příkazem `pwd(1)` (print working directory):

```
$ pwd
/usr/petr
$
```

Výpis `/usr/petr` nás informuje o nastavení na adresář "petr", jehož nadřazeným adresářem je adresář "usr" a tomuto je nadřazen adresář "/" (kořenový adresář). Výpis je označován jako cesta (path) k adresáři (v tomto případě k pracovnímu adresáři).

Změnit svůj pracovní adresář můžeme příkazem `cd` (change directory):

```
$ cd /usr
$ pwd
/usr
$
```

Použije-li uživatel příkaz `cd` bez argumentu, nastavuje si tak domovský adresář.

```
$ cd
$ pwd
/usr/petr
$
```

Změna pracovního adresáře podléhá samozřejmě kontrole oprávnění vstupu do adresáře podle přístupových práv, vstup do adresáře je dovozen příznakem "x" ve výpis atributů adresáře.

Adresář vytvoříme příkazem `mkdir(1)` (make directory) a zrušíme příkazem `rmdir(1)` (remove directory):

```
$ mkdir testy (vytvoření adresáře "testy")
$ ls -l
total 2
drwxr-xr-x 2 petr group 32 May 13 11:27 testy
$ cd testy (nastavení na adresář "testy")
$ mkdir soubory
$ cd (nastavení na domovský adresář)
$ rmdir testy
rmdir: testy: Directory not empty
```

Adresář totiž můžeme zrušit, jen je-li prázdný, takže píšeme:

```
$ rmdir testy/soubory
$ rmdir testy
$
```

Cesta k adresáři může být absolutní ("/usr/petr/testy/soubory") nebo relativní ("testy/soubory") vzhledem k pracovnímu adresáři.

Používat můžeme také jména adresářů "." a "..". Např.:

```
$ pwd
/usr/petr
$ cd ..
$ pwd
/usr
$
```

Adresář obsahuje obyčejné soubory. Základní manipulace se soubory jsou kopie, přejmenování a zrušení souboru. Kopii souboru vytvoříme příkazem cp(1) (copy):

```
$ cp .profile text
```

kde jako první argument příkazu je jméno existujícího souboru (".profile") a druhý argument je jméno nově vytvářeného souboru ("text").

Přejmenovat soubor můžeme příkazem mv(1) (move):

```
$ mv text iniscript
```

kde jako první argument je původní jméno existujícího souboru ("text") a druhý argument jeho nové jméno ("iniscript"). Vzhledem k tomu, že je pro systém rozhodující číslo i-uzlu, je zřejmé, že je v tomto případě provedena změna položky jména v pracovním adresáři.

Soubor rušíme příkazem rm(1) (remove):

```
$ rm iniscript
```

Použijeme-li volbu "-i"

```
$ rm -i iniscript
iniscript: ? _
```

rm(1) očekává potvrzení o rušení (yes=rušíme soubor, cokoliv jiného znamená, že soubor není zrušen). Dále je možné použít volbu "-r", pomocí které můžeme rušit rekurzivně celý podstrom:

```
$ pwd
/usr/petr
$ mkdir testy
$ cd testy
$ cp /usr/petr/.profile iniscript
$ cd
$ rm -r testy
$ ls -l testy
testy not found
$
```


Při použití příkazu "cp /usr/petr/.profile iniscript" používáme absolutní cestu k souboru ".profile" a relativní cestu k souboru "iniscript". Vzhledem k možnosti využít adresáře se jménem "." můžeme příkaz přepsat na

```
$ cp ../.profile iniscript
```

Adresář "." můžeme využít pro referenci pracovního adresáře také v příkazu cp(1), kde na místě druhého argumentu může být uvedeno jméno adresáře a koncové jméno souboru je při kopii převedeno na nově vznikající soubor.

```
$ cp ../profile .
```

okopíruje z nadřazeného adresáře soubor ".profile". Nově vzniklý soubor má pak také jméno ".profile".

Obyčejný soubor vzniká obvykle jako produkt editace (editorem ed(1), ex(1) nebo vi(1), viz Příloha B) nebo některé jiné aplikace (překladače, databáze atd.). Pomocí programu cat(1) lze soubor se textovým obsahem vytvořit také, a sice pomocí mechanismu přesměrování. Každý program spuštěný z příkazového řádku obvykle pracuje interaktivním způsobem, standardně čte z klávesnice terminálu a výsledky své činnosti zobrazuje na obrazovku terminálu. Vstupu z klávesnice říkáme standardní vstup (standard input) a výstupu na obrazovku standardní výstup (standard output). Programy pracují jako filtry, data ze standardního vstupu předávají na standardní výstup. Napíšeme-li příkaz cat(1) bez argumentů, cat(1) očekává vstup dat nikoliv z určeného souboru, ale z klávesnice:

```
$ cat
zkusebni text
zkusebni text
dalsi radek zkusebniho textu
dalsi radek zkusebniho textu
^d$
```

A po řádcích jej předává na standardní výstup. Standardní výstup můžeme přepnout a soubor pomocí znaku '>' takto:

```
$ cat > ztext
zkusebni text
dalsi radek zkusebniho textu
^d$ ls -l ztext
-rw-r--r-- 1 petr group 43 May 13 13:46 ztext
$ cat ztext
zkusebni text
dalsi radek zkusebniho textu
$
```

Soubor "ztext" je nově vytvořen a pokud již dříve existoval, je zkrácen na nulovou délku a naplněn výstupem programu cat. Pokud soubor existoval a chceme-li zachovat jeho obsah, můžeme nový text k obsahu souboru jen připojit tak, že standardní výstup přesměrujeme využitím dvojznaku ">>":

```
$ cat >> ztext
jeste jeden radek zkusebniho textu
^d$ cat ztext
zkusebni text
dalsi radek zkusebniho textu
jeste jeden radek zkusebniho textu
$
```

a naopak, pokud by soubor se jménem "ztext" doposud neexistoval, je vytvořen s nulovou délkou a obsah z klávesnice je k němu připojen.

Příkazem find(1) vyhledáme soubor podle některého jeho atributu, a to v dané části stromu adresářů. Např.

```
$ pwd
/usr/petr
$ find . -name ztext -print
/usr/petr/ztext
...
```

prohledává podstrom začínající pracovním adresářem "." a na standardní výstup vypisuje (na základě volby "-name") cesty všech souborů, jejichž jméno je "ztext". Výpis je zajištěn pomocí volby "-print". Lze totiž požadovat nejen výpis cesty k souboru, ale i provedení určité akce. Např.

```
$ find /usr/petr -name core -exec rm {} \;
```

hledá v podstromu adresáře "/usr/petr" všechny soubory se jménem "core" a ruší je. Určit jiný atribut, podle kterého find(1) vybírá soubory, můžeme použitím jiné volby, např.

```
$ find /usr -user petr -print
```

vyhledá všechny soubory od adresáře "/usr", které jsou ve vlastnictví uživatele "petr". Konečně

```
$ find /usr -user petr -type d -print
```

vypíše všechny adresáře v podstromu "/usr", které vlastní uživatel "petr" (lze použít více voleb současně), a

```
$ find /usr/petr /usr/jan -mtime -7 -print
```

vypíše cestu k souborům podstromů "/usr/petr" a "/usr/jan" (lze stanovit více výchozích adresářů současně), které byly změněny v posledních 7 dnech.

Speciální soubor je přístup k periférii výpočetního systému. Speciální soubory jsou standardně uloženy v podstromu začínajícím adresářem "/dev" a uživateli jsou dostupné podle přístupových práv. Přístup uživatele např. k tiskárně může být pomocí speciálního souboru "lp":

```
$ ls -l /dev/lp
c-w--w--w-- 1 bin bin 6, 0 May 14 09:14 /dev/lp
$
```

kde význam vypsaných atributů speciálního souboru se shoduje s významem atributů obyčejného souboru vyjma místo, kde bývá uvedena velikost v počtu slabik. Zde jsou zobrazena dvě čísla oddělená čárkou (6,0), která nazýváme (po řadě) hlavní číslo speciálního souboru (major number) a vedlejší číslo speciálního souboru (minor number). Hlavní číslo je identifikace typu zařízení (tiskárna, disk ...) a vedlejší určuje specifika konkrétní periferie (např. o kterou periférii v pořadí se jedná a jakým způsobem bude ovládána).

Vzhledem k uvedenému příkladu může uživatel psát data na tiskárnu ze souboru např. "ztext" takto:

```
$ cat ztext > /dev/lp
```

Což mu umožní jednak přesměrování a jednak atribut přístupu pro zápis k souboru "/dev/lp". Přestože je v mnohých instancích zápis na tiskárnu takto povolen, je lépe právo pro zápis ponechat pouze superuživateli, a ostatním uživatelům umožnit tisk na tiskárnu pomocí mechanismu spooling. To znamená, že požadavky na tisk jsou řazeny do fronty typu FIFO (First In, First Out) a postupně odebírány a posílány do speciálního souboru operačním systémem. Tisk přesměrováním dvou uživatelů současně totiž způsobí na tiskárně smíchání obsahů obou souborů. Pomocí mechanismu spooling uživatel tiskne příkazem

```
$ lp ztext
printer-268
```

obsah souboru "ztext". Výpis textu "printer-268" je označení ve frontě požadavků na tisk. V době tisku (nebo jen existence ve frontě) může uživatel použít

```
$ cancel printer-268
```

a tím požadavek z fronty zruší.

Jiný příklad je úschova dat na přenosné médium, jako je např. disketa nebo magnetická páska. Magnetickou pásku využívá ve většině systémů pouze superuživatel pro úschovu dat a v rámci celého systému souborů, takže speciální soubor magnetické pásky může např. vypadat:

```
$ ls -l /dev/rmt
crw-rw---- 1 root root 10, 0 May 15 10:27 /dev/rmt
$
```

kde jsou práva pro zápis a čtení obyčejnému uživateli odebrána.

U diskety se předpokládá lokální úschova dat. Uživatel proto soubor

```
$ ls -l /dev/rfd
crw-rw-rw- 3 bin bin 2, 0 May 15 09:24 /dev/rfd
$
```

využívá pro zápis i čtení např. archivačním programem tar(1). Příkaz

```
$ tar cf /dev/rfd ztext
```

provede archivaci souboru "ztext" z pracovního adresáře na disketu (volba c=create je vytvoření archivace, f=file znamená že následující argument příkazu je určení speciálního souboru). tar(1) je definován při použití voleb bez znaku '-'.
\$ tar tf /dev/rfd

vypíše obsah archivace (table) a

```
$ tar xf /dev/rfd ztext
```

soubor se jménem "ztext" z diskety přesouvá do pracovního adresáře (extract). Není-li uvedeno jméno souboru, přesouván je celý obsah archivace.

Jména speciálních souborů se řídí konvencemi danými výrobcem, nikoliv obecnou normou. Přesto existují zvyklosti pro jména speciálních souborů jako např.:

```
console operátorská konzola,
clock hodiny,
fd disketa,
dsk disk,
kmem operační paměť jádra,
lp tiskárna,
mem operační paměť,
mt klasická magnetická páska,
tty terminál.
```

kde uvedené texty zastupují pouze používaný základ jména, protože např. disketa pro sekvenční přístup (po znacích) mívá předponu "r", tj. "rfd" (stejně tak mg. páska) a dále u více připojených periférií stejného typu je příponou jména pořadí periferie, např. "tty00", "tty01", ...

Přístup k periférii je buď znakový nebo blokový. Typicky blokové zařízení je disk, znakové terminál. Ale i s diskem můžeme pracovat sekvenčně po znacích, proto je i pro něj vytvořen znakový speciální soubor. Znakový speciální soubor má při výpisu "ls -l" atributů v prvním sloupci znak 'c', blokový znak 'b'.

Speciální soubor vytváří privilegovaný uživatel pomocí příkazu `mknod(1)`, kdy respektuje všechny vazby na operační systém a připojené periferie (viz kap.9).

Změna atributů souboru (ať už obvyčejného, adresáře, speciálního) znamená změnu dat v i-uzlu. Uveďme si pro tyto změny několik příkladů.

Pomocí příkazu `chown(1)` můžeme měnit vlastníka souboru. Příkazem

```
$ chown root ztext
```

předáme vlastnictví souboru "ztext" uživateli "root". Opětne navrácení souboru uživateli "petr" ale může uskutečnit pouze uživatel "root". Obdobně lze pomocí příkazu `chgrp(1)` změnit příslušnost souboru ke skupině.

Příkaz `chmod(1)` mění přístupová práva souboru. Změnu můžeme zapsat buďto číselně, např.

```
$ chmod 775 davka
```

což je specifikace v osmičkové soustavě přístupových práv. Souboru se jménem "davka" budou přístupová práva nastavena způsobem

```
rw-rw-r--x
111111101
 7 7 5
```

což odpovídá hodnotě v osmičkové soustavě jednotlivých cifer vždy pro trojici bitů oprávnění. Mnemonický zápis vychází ze zadání vlastníka souboru (u=user), skupiny (g=group), ostatních (o=others) nebo všech (a=all) uživatelů, a označení přístupových práv "r" (r=read) pro čtení, "w" (w=write) pro zápis a "x" (x=execute) pro provádění. Dále je možné vlastníkově, skupině nebo ostatním právo nastavit (=), připojit (+) nebo odebrat(-). Např.

```
$ chmod a=rx,g+w, u+w ztext
```

je nastavení, které odpovídá předchozímu příkladu s numerickým nastavením (nejprve je všem nastaveno pouze právo čtení a provádění a dále je přístup pro zápis přiznán nejprve skupině a pak vlastníkově souboru).

Systém souborů je realizován na magnetickém médiu. Logická struktura magnetického média je svazek (angl. je používán výraz *filesystem*, v překladu systém souborů). Ve struktuře se vyskytuje nám známý výraz i-uzel. i-uzel je jednoznačná identifikace souboru, obsahuje jeho atributy, s číslem i-uzlu je v adresáři spojeno jméno souboru. Číslo i-uzlu je jeho pořadí v rámci oblasti i-uzlů. Velikost i-uzlu je 64 slabik a i-uzel obsahuje tyto informace:

- ▶ vlastníka souboru,
- ▶ typ souboru (obyčejný soubor, adresář ...),
- ▶ přístupová práva,
- ▶ datum a čas poslední manipulace se souborem,
- ▶ počet odkazů na soubor z různých míst stromu adresářů,
- ▶ tabulka datových bloků,
- ▶ velikost souboru.

Položka počet odkazů je možnost přístupu k témuž souboru z několika různých míst systému souborů, např.:

```
$ pwd
/usr/petr
$ cat > sdileny
^d$
```

(vytvoření souboru s nulovou délkou). Za předpokladu, že soubor "sdileny" dosud v pracovním adresáři ("/usr/petr") neexistoval, je z oblasti i-uzlů vybrán první neobsazený, souboru je v adresáři přiděleno odpovídající číslo i-uzlu a současně s alokací položek i-uzlu je nastaven počet odkazů na 1. Příkazem

```
$ ln sdileny ../jan/sdileny
```

vytváříme nový odkaz na tentýž i-uzel (za předpokladu, že existuje adresář "/usr/jan" a jeho přístupová práva jsou pro nás příznivá).

Z obou adresářů je možný přístup k datům téhož souboru. Je-li "/usr/jan" domovský adresář uživatele "jan" a v systému je současně přihlášen uživatel "petr" i "jan", současně ze dvou různých terminálů je možné soubor "/usr/petr/sdileny" alias "/usr/jan/sdileny" číst a současně do něj zapisovat (podle nastavených přístupových práv). Můžeme to prověřit příkazem

```
$ ls -i sdileny
829 sdileny
$ ls -i ../jan/sdileny
829 ../jan/sdileny
$
```

když pomocí volby "-i" požadujeme kromě výpisu jména souboru i číslo odpovídajícího i-uzlu.

Rušíme-li libovolný soubor (příkazem `rm(1)`), systém souborů (pomocí rutin jádra) prohlídí i-uzel a snižuje hodnotu počtu odkazů o 1. Je-li po této dekrementaci hodnota větší než 0, systém rozpojí pouze vazbu jméno souboru - i-uzel. Je-li ale hodnota i-uzlu nulová, i-uzel je uvolněn ze seznamu alokovaných a převeden do seznamu volných, datový obsah souboru je zrušen a soubor i se svou vnitřní strukturou zaniká. Např.:

```
$ rm sdileny
```

zruší odkaz z adresáře "/usr/petr". I-uzel zůstává alokován, s ním všechna data souboru.

```
$ rm ../jan/sdileny
```

zruší poslední odkaz na i-uzel a tím i soubor.

Uvedený způsob práce s více odkazy na jedna data se omezuje na práci s jedním svazkem. Protože by takové zúžení bylo nepraktické, je možné použít odkaz na soubor, který existuje na jiném svazku, k tomu má příkaz `ln(1)` volbu "-s". Odkaz na soubor jiného svazku je nepřímý odkaz a v případě výpisu adresáře pomocí `ls(1)` jej rozpoznáme podle znaku 'l' v prvním sloupci. Nepřímý odkaz je realizován příznakem v i-uzlu a tak, že v datové části je uložena cesta k souboru na jiném svazku. Nepřímé odkazy přinášejí komplikace správci systému, spojí-li svazky do stromu adresářů jinak než obvykle, protože v případě, že svazek není připojen k odpovídajícímu adresáři, nepřímý odkaz nevede nikam (připojování svazků je popsáno v následujícím textu).

Odkaz (přímý i nepřímý) může být vytvořen i na adresář, je to ale dovoleno pouze privilegovanému uživateli.

Přístup k souboru v UNIXu je obecně vícenásobný. Za předpokladu vhodných přístupových práv mohou uživatelé současně číst nebo i zapisovat do jednoho souboru současně. Jádro pouze zajišťuje výlučný přístup k obsahu souboru v daném čase. V případě, že data sdílí několik uživatelů (např. při práci na společném projektu), musí volit vedoucí projektu (v součinnosti se správcem systému) jiné mechanismy pro zajištění konzistence dat. Jedna z možností je zamykání určité části dat souboru proti zápisu (nebo i čtení) po stanovenou dobu. Zamykání souboru je podporováno voláním jádra `fcntl(2)`.

4.3.4 Procesy

Obdobně jako systém souborů, i procesy tvoří hierarchickou strukturu. Je-li nastartován operační systém, tj. je-li spuštěno jádro, vzniká za jeho podpory proces nazývaný swapper (někdy sched). Je to proces, který je identifikován jako proces č. 0. Jeho hlavním smyslem je pracovat pro údržbu správy paměti, ale také, ihned na začátku své činnosti, sám vytváří další proces pomocí odkazu na jádro. Tento další proces je nazýván init a má identifikační číslo 1. Proces, který vznikne odkazem na jádro z jiného procesu, nazýváme procesem synovským (child) vzhledem k procesu, který jej takto vytváří a který je označován jako otec nebo rodič (parent). Žádosti o vznik dalšího procesu říkáme fork (rozvětvení) a proces jej uplatní pomocí volání jádra fork(2). Proces init je otcem dalších procesů. Init udržuje několik úrovní stavu systému, přitom je každá úroveň reprezentována množinou procesů, které init vytváří a dohlíží na jejich stav. Hlavní dvě úrovně jsou označovány jako úroveň jednouživatelská a úroveň víceuživatelská. Na úrovni jednouživatelské vytváří init jen několik dalších procesů, které umožňují vstup do systému pouze jednomu uživateli, a to privilegovanému. Je to úroveň pro údržbu a celkové změny v systému. Úroveň víceuživatelská je standardní úroveň pro běh systému, kdy se interaktivně z terminálů uživatelé přihlašují do systému a pracují v něm.

Aby proces init umožnil uživateli vstup do systému, vytváří proces, jehož standardním vstupem i výstupem je terminál.. Takový proces je nazýván getty a komunikuje s uživatelem v době přihlašování. Procesům getty (jejich počet je dán počtem terminálů) jádro přiděluje pro jednoznačnou identifikaci poslední přidělené číslo procesu zvýšené o 1. Proces getty se v případě správného přihlášení mění na proces sh, který realizuje běžnou komunikaci uživatele se systémem a je nazýván příkazovým interpretem (např. Bourne shell), sh na obrazovce vypisuje znak '\$' (nebo '#') a interpretuje příkazový řádek uživatele.

Příkazový řádek uživatele, např.

```
$ cat ztext
```

je interpretován procesem sh vytvořením nového procesu (voláním jádra fork(2)) se jménem cat. Sh je tedy rodičem cat.

Část procesů běží v nekonečné smyčce, probouzí se přitom k akci na základě časového limitu nebo pokynu jádra. Takovému procesu říkáme démon a příkladem může být právě proces update, který periodicky řídí frontu tiskárny.

Identifikační číslo procesu má zkratku PID (process identification) a je to atribut každého procesu pro jeho jednoznačné odlišení od ostatních procesů jádrem, uživatelem nebo kterýmkoli procesem. PID je celé kladné číslo a je přidělováno nově vznikajícímu procesu podle naposledy použité hodnoty zvýšené o 1. Hodnota PID v instalacích, kde pracuje několik desítek uživatelů současně za nepřetržitého provozu systému několika dnů i týdnů, jistě dosáhne nejvyšší možné hodnoty. Tehdy jádro začíná opět od hodnoty 0. Vynechává přitom ta PID, která jsou přidělena aktivním procesům a měla by tak stejnou hodnotu.

Procesy můžeme rozdělit na systémové (systémem zvýhodňované) a ostatní. Systémový proces je např. init, swapper, do skupiny ostatních patří sh, cat atd. Rozdíl mezi systémovými jiným procesem je zejména ve stanovení jeho dynamické priority, což je číselná hodnota, podle které systém jednomu z procesů přidělí čas procesoru. Víceuživatelský operační systém UNIX zajišťuje sdílení zdrojů výpočetního systému více uživatelům pomocí více procesů, které běží zdánlivě současně. Ve skutečnosti jádro uděluje čas procesoru vždy jen jednomu procesu a jen na omezenou dobu. Vyhoví tak v průběhu velmi krátkého časového intervalu všem procesům, takže se zdá, že procesy běží (uživatelé pracují) současně. Měřítkem přidělení času procesoru jednomu z procesů je dynamická priorita procesu. Je vypočtena pro každý proces z jeho výchozí uživatelské priority, což je celé kladné číslo v intervalu obvykle

0-39 (menší hodnota znamená vyšší prioritu), a z času systému. Dynamická priorita je vypočítávána všem procesům vždy po uplynutí 1 vteřiny, anebo na základě interakce procesu jádrem. Protože interakce z terminálu znamená zpracování vstupu jádrem, zvýhodňovány jsou procesy podporující práci u terminálu. Z vlastnosti zvýhodňování interaktivních procesů jádrem plyne pro UNIX označení interaktivní operační systém. Hranicí v hodnotě výchozí priority je hodnota 20, která je dělicím bodem mezi systémovými a jinými procesy. Žádný obyčejný proces nemůže mít po dobu svého běhu nastavenou dynamickou prioritu na hodnotu nižší, než je 20. Tím je co nejvíce zajištěna průchodnost systému a ochrana proti zahlcení.

Výchozí priorita při spuštění procesu je právě 20. Proces se ale může části své priority vzdát ve prospěch ostatních. Při spouštění procesu např. lze psát

```
$ nice -5 find / -name core -print > fcore
```

kdy před vlastní příkaz `find(1)` přepíšeme příkaz `nice(1)`. Volba "-5" je hodnota, o kolik bude výchozí priorita spouštěného procesu pro `find(1)` nižší. V našem případě bude výchozí priorita procesu `find` 25. Napíšeme-li

```
$ nice find ...
```

výchozí priorita je implicitně snížena o 10 (je tedy 30). Privilegovaný uživatel může prioritu spouštěného procesu i zvyšovat, a to způsobem

```
# nice -10 ...
```

takže výchozí priorita je o 10 zvýšena ($20 - 10 = 10$, výchozí priorita je 10).

Dříve než může být procesu přidělen procesor, musí být proces zaveden do operační paměti. Ve chvíli, kdy je proces vytvořen, je snahou jádra přidělit mu požadovanou paměť. Není-li dostatečně velká požadovaná paměť volná, jádro uplatňuje na procesy v operační paměti algoritmy se snahou odsunout některé procesy mimo paměť a zajistit tak dostatek volné paměti pro nový proces. Orientace jádra zde vychází z posouzení doby procesu již strávené v paměti. Proces nejdéle sídlící v paměti je odsunut. Místo, kam jsou procesy z paměti odsouvány a odkud jsou opět po čase do paměti zavlékány, je disková paměť označovaná jako odkládací oblast. Odkládací oblast je dána parametry jádra a je situována mimo jakýkoliv svazek systému souborů. Pro práci s odkládací oblastí jádro používá proces s `PID=0` označovaný jako `swapper`.

Texty `fork`, `sleep` a `exit` vyjadřují odpovídající volání jádra po řadě pro vytvoření procesu, neaktivní čekání procesu po stanovený interval nebo od okamžiku vzniku určité události a ukončení procesu. Jinak je na běh procesu rozdělen na úroveň uživatelskou a supervizorovou, tj. rozdělení na část, kdy proces interpretuje instrukce programu a kdy pro proces pracuje jádro (proces použil některé volání jádra). Mátoha je stav procesu po žádosti o jeho ukončení, kdy systém ještě nezrušil pro daný proces všechny odpovídající položky v tabulkách jádra a v co nejkratší době tak učiní.

Odkládací oblast může být také využívána uživatelem pro zefektivnění opakovaného spouštění procesu nad tímž programem. Soubor, ve kterém je program v systému souborů uložen, má v tomto případě nastaven tzv. `t-bit`. `T-bit` nastavujeme příkazem

```
$ chmod u+t soubor
```

kde "soubor" je jméno souboru s programem. Jádro po skončení procesu, který interpretuje program soubor okopíruje obsah souboru s programem do odkládací oblasti a toto umístění si poznamená. Při opětovém spuštění programu jádro využívá přímého kontaktu s textem programu v odkládací paměti, takže doba spuštění se tím zkrátí.

Uživatel přihlášený do systému je vlastníkem všech procesů, které interaktivně spustil. V případě, že se nejedná o privilegovaného uživatele, jádro při spouštění zjišťuje přístupová práva k souboru, v němž je program uložen. Nemá-li uživatel přístupová práva pro provádění

(x-bit), program nemůže být spuštěn. Dále je možné, že může program spustit, ale využívá některé volání jádra, na které má oprávnění pouze superuživatel. V tomto případě proces nebude pracovat správně. Je ale běžné, že systém umožní v rámci spuštěného programu privilegovanou akci, a sice tak, že v místě nastaveného bitu proveditelnosti je namísto označení 'x' označení 's' (tzv. nastavení s-bitu), např.:

```
# chmod g,o+s soubor
```

Princip je v tom, že je uživatel v akci regulován programem, který vlastní superuživatel. Nastavením s-bitu může také obyčejný uživatel dát k dispozici kontrolované manipulace se svými daty ostatním uživatelům.

Proces je v době běhu v paměti rozdělen na tři části - segmenty. Textový segment obsahuje instrukce programu a v datovém segmentu jsou datové struktury programu. Pokud není řečeno jinak, UNIX považuje textový segment za část, která není v době běhu procesu měnitelná a je-li vytvořeno několik procesů s odkazem na tentýž program (např. editor vi(1), program getty(8), příkazový interpret sh(1) atd.), běžící procesy sdílí tentýž textový segment. Každý takový nově vzniklý proces proto požaduje přidělení operační paměti pouze pro datový segment a zásobník.

Procesy jsou sdružovány do skupin, které nazýváme rodiny procesů. Každý proces má právo prohlásit se (pomocí volání jádra setpgroup(2)) za vedoucího skupiny (otce rodiny), a tím se osamostatnit od rodiny, do které dosud patřil. Každý shell se po přihlášení uživatele prohlašuje za otce rodiny, osamostatňuje se tak a všichni jeho synové (nezaloží-li si také svoji rodinu) jsou s ním existenčně spjati. Ukončí-li totiž otec rodiny svoji činnost, všichni jeho synové umírají také (pokud nejsou v supervizorovém režimu). V řádkové komunikaci u terminálu může uživatel příkazovým řádkem končícím na znak '&' vytvořit proces, který běží, a uživatel může zadávat další příkazy, tj. vytvářet další procesy, např.

```
$ cc velkyprog.c &  
159  
$
```

(překlad programu v jazyce C), kde vypsané číslo je PID vzniklého procesu. Pokud ale ještě před dokončením takového procesu dá pokyn k odhlášení, ukončí tak činnost nejen procesu pro sh (příkazový interpret), ale také procesu pro cc, syn je násilně přerušen. Syna prohlásíme za samostatného příkazem

```
$ nohup cc velkyprog.c &
```

a po odhlášení běží proces cc až do regulérního ukončení.

Komunikace procesů je na základní úrovni zajištěna zasíláním signálů. Signály jsou číslovány od 1, přitom např. přerušení z klávesnice odpovídá signálu č. 2. Např. proces sh může poslat svému synu, jehož PID zná, signál č. 2 takto

```
$ kill -2 159
```

Jméno příkazu přitom odpovídá dřívějšímu významu použití signálu. Zaslat procesu signál totiž znamenalo proces ukončit. Výchozí reakce procesu na příchod různých signálů jsou dnes stanoveny a nemusí to být vždy ukončení. Proces ale může reakci na příchod signálu měnit, může např. pouze provést určitou akci a pokračovat dál v činnosti, nebo signál ignorovat, anebo provést určitou akci a svoji činnost ukončit. Závisí to na způsobu použití volání jádra signal(2). Proces posílá jinému procesu signál voláním jádra kill(2). Musí ale znát PID procesu, kterému signál posílá.

Komunikace dvou procesů na úrovni přenosu dat je možná pomocí roury, kdy procesy pracují podle schématu

```
PRODUCENT | KONZUMENT
```


tak, že proces PRODUCENT předává svá výstupní data na vstup procesu KONZUMENT.

Schéma odpovídá zápisu spojení dvou procesů v příkazovém řádku. Spojení je provedeno tak, že standardní výstup procesu PRODUCENT je přepnut do části paměti pro rouru a standardní vstup procesu KONZUMENT je na tutéž paměť napojen jako proces, který z roury data čte. Roura pracuje sekvenčně způsobem FIFO, KONZUMENT dostává nejprve data, která PRODUCENT do roury zapsal jako první. Po příkazovém řádku

```
$ ls | wc -l
```

nebude výstup procesu ls (seznam souborů) vypisován na obrazovku terminálu, ale bude zapisován do vytvořené roury a proces wc (příkaz s volbou "-l" počítá řádky standardního vstupu) bude zpracovávat data čtená z roury. Jádro zajistí pozastavení procesu PRODUCENT, je-li roura plná, anebo procesu KONZUMENT, je-li roura prázdná. PRODUCENT uzavírá rouru znakem konce souboru (CTRL-d) a KONZUMENT po přečtení konce souboru také, čímž roura zaniká.

Komunikace procesů pomocí roury je omezena na procesy, které jsou v přímém příbuzenském vztahu. Takto nemohou např. spolupracovat procesy vytvořené různými uživateli. Tento a další problémy při komunikaci procesů řeší část systému nazývaná jako IPC (Interprocess Communication).

V rámci IPC je možné vytvářet a používat pojmenovanou routu, kdy je roura vytvořena trvale a spojena se jménem souboru (zvláštního typu). Dále je v IPC možné, aby dva libovolné procesy spolupracovaly na úrovni předávání zpráv, je možné, aby sdílely data svých datových oblastí, a konečně je využitelný mechanismus Dijkstrových semaforů. IPC ale řeší situaci nejen v rámci jednoho výpočetního systému, ale pomocí schránek také v sítích.

4.3.5 Jádro

Jádro (kernel) je programové vybavení, které pracuje na počítači bez jakékoliv další programové podpory. e velkým rozhraním mezi uživatelem a technickým vybavením výpočetního systému. Zajišťuje realizaci odkazů uživatelů nebo procesů na periferie a vůbec všechny technické zdroje počítače. Dále udržuje a podporuje v činnosti systém souborů a běžící procesy.

Uživatelská úroveň je jádrem podporována přes rozhraní volání jádra. Stručně řečeno, program používá volání funkce, která je voláním jádra, a program tak vstupuje do režimu obsluhy jádrem (proces vstupuje do supervizorové úrovně). Činnost jádra v okamžiku, kdy zajišťuje určitou akci vyžádanou procesem, je jiným procesem nepřerušitelná. Vzhledem k časové prodlevě, po kterou proces v supervizorové úrovni zůstává, není vhodný pro řízení procesů v reálném čase. Snaha vyřešit problém reálného času je dlouholetá, její výsledky se zatím odrážejí v operačním systému HP-UX od firmy HEWLETT PACKARD, kde je jádro upraveno tak, aby doba strávená v supervizorové úrovni procesu byla minimalizována. Jiným příkladem úpravy operačního systému UNIX pro reálný čas je LynxOS rovněž americké firmy Lynx Real-Time Systems. I verze UNIX SYSTEM V.4 má podle dokumentace možnost práce procesů v reálném čase. Praktické zkušenosti ale ukazují, že jde prozatím o úpravu jádra, která nemá reálné využití (interakce ostatních uživatelů se výrazně zpomalí). SVID3 práci v reálném čase už ale zahrnuje.

4.3.6 Uživatelé

Uživatel je poslední entita, která souvisí se základy operačního systému UNIX. V době, kdy je v systému přihlášen, je uživatel reprezentován akcemi, které vykonává pomocí skupiny

procesů. Vzhledem k tomu, že je mu v době přihlášení přidělen nový proces shellu, který sám sebe prohlásí za otce rodiny, odpovídá uživateli rodina procesů tohoto shellu.

Staticky je uživatel reprezentován všemi soubory, u kterých je označen jako vlastník (zejména jsou to data začínající jeho domovským adresářem). Dále je to uživatelova registrace v tabulkách souborů `/etc/passwd` a `/etc/group`. V `/etc/passwd` odpovídá registraci jednoho uživatele 1 řádek souboru. Na řádku jsou uvedeny jednotlivé položky uživatele oddělené znakem `:`. Např.

```
...  
petr:*:236:50:Petr Nevecny, tel 537973:/usr/petr:/bin/sh  
...
```

První položka na řádku je jméno uživatele. Druhá je jeho heslo, které je zde v zakódované podobě. V současných systémech je na místě položky hesla, pokud je přístup heslem chráněn, uveden znak `*` (nebo `x`), a šifrované heslo je uloženo na jiném, běžně nesdělovaném místě (např. v `/etc/secure`). Následující položka je číselná identifikace uživatele, která je v rámci instalace jedinečná (zde 236). Tato hodnota je např. uložena v `i`-uzlu souboru, který uživatel vlastní. Dále následuje identifikace skupiny, do které uživatel patří (50).

Položka identifikace skupiny koresponduje se souborem `/etc/group`, kde je uveden seznam všech skupin, na každém řádku jedna, např.

```
...  
group::50:jan,petr  
...
```

a položky mají po řadě význam: jméno skupiny, heslo (uvedená skupina je bez hesla), číselná identifikace a seznam uživatelů patřících do skupiny a oddělených znakem `,`. Uživatel 236 tedy patří do skupiny se jménem `group`.

Další položka `/etc/passwd` po čísle skupiny je komentář. Následuje cesta k domovskému adresáři, podle ní shell nastavuje uživateli pracovní adresář v době přihlášení. Konečně poslední položkou je cesta k programu, který bude spuštěn jako příkazový interpret pro přihlášení uživatele. Tato položka bývá naplněna buď `/bin/sh` (Bourne shell) nebo `/bin/csh` (C-shell), anebo kteroukoliv jinou aplikací, která je schopna vést dialog s uživatelem u terminálu. Výhodou je, že při ukončení činnosti (ať už regulérního nebo z důvodů havárie programu) je uživatel odhlášen, takže nezůstává po ukončení aplikace v systému a nemůže neodborným zásahem poškodit uložená data. Je-li v souboru `/etc/passwd` poslední položka na řádku vynechána (řádek končí znakem `:`), uživateli je standardně spuštěn program souboru `/bin/sh`.

5 Přenos informací a počítačové sítě

Pod pojmem počítačová síť (dále jen síť) si lze představit skupinu uzlů navzájem propojených komunikačními kanály. Sítě mohou být navzájem propojeny s jinými sítěmi a mohou také obsahovat podsítě. Uzly sítě mohou být:

- ▶ koncová zařízení (například uživatelské počítače, servery),
- ▶ sdílená zařízení (např. tiskárny, scannery),
- ▶ propojovací zařízení (např. přepínače).

Pro klasifikaci sítí lze použít různá kritéria. Nejčastěji používaná jsou:

- ▶ Prostor, který příslušná síť pokrývá (např. lokální síť, rozlehlé síť).
- ▶ Způsob, jakým jsou navzájem propojeny jednotlivé uzly sítě - síťová topologie (např. sběrníková topologie).
- ▶ Způsob vysílání datových signálů - technika řízení přístupu k přenosovému médiu (např. Ethernet).
- ▶ Typ přenosové technologie - typ síťové architektury (např. síť TCP/IP).
- ▶ Typ informace, která je přenášena (síť hlasové, datové, kombinované).
- ▶ Způsob správy uživatelských přístupů do sítě (síť privátní a síť veřejné).
- ▶ Typ fyzických spojů (médiu) použitých v infrastruktuře sítě (např. síť s optickými kabely, síť s koaxiálními kabely, síť s kroucenými dvoulinkami).

Úkolem sítě je zajistit komunikaci mezi uzly pro přenos informace a pro sdílení prostředků. Informace jsou na zdrojovém systému formovány do tzv. zpráv. Zprávy kromě vlastních uživatelských dat obsahují i data režijní (např. adresy), která jsou určena nejen cílovému systému, ale také systémům mezilehlým, přes které je zpráva přenášena. Přenosové technologie mají omezen max. objem dat, která mohou být v jedné zprávě přenesena. Pokud by požadovaná zpráva překročila tento limit, původní zpráva je na vysílací straně rozdělena na potřebný počet částí, z nichž každá je opatřena nutnými režijními daty. Takto vznikají základní datové jednotky určené k přenosu, zvané datové pakety.

5.1 Základy propojování prostředků

5.1.1 Referenční model OSI

V průběhu dvacátého století došlo k prudkému rozvoji výpočetní technologie, zejména vzrostl počet propojení počítačů a počítačových sítí. S tím souvisela potřeba zajistit vzájemnou bezproblémovou propojitelnost různých systémů pro možnost sdílení jejich výpočetních zdrojů a vzájemnou komunikaci. Problém kompatibility je řešitelný jen standardem, popisujícím jednotnou síťovou architekturu pro volně připojitelná koncová zařízení sítě, tzv. otevřené systémy. Za všech standardů se ustálil vrstvý model předložený Mezinárodní standardizační organizací (ISO), která publikovala v roce 1984 mezinárodní normu IS 7498 nazvanou Referenční model OSI (Open Systems Interconnection). Standard řeší nejenom síťová propojení, ale také formáty a organizaci dat (informace) pro možnost poskytování distribuovaných informačních služeb.

Pro zajištění kompatibility sítí je nutné, aby komunikující zařízení splňovala řadu kritérií: mechanická, elektrická, obvodová, protokolová a další. Výše uvedený referenční model byl zaveden pro sjednocení těchto kritérií a sjednocení popisu kompatibility sítí.

Princip vrstvého řízení datové komunikace vychází z myšlenky rozdělení činností při komunikaci nejen v horizontálním směru datového řetězce, ale i ve směru vertikálního členění datových prostředků. Vrstvy referenčního modelu OSI jsou charakterizovány svým účelem – funkcemi, které jsou na dané vrstvě vykonávány. Důležitou vlastností modelu OSI je, že nemá platnost jen pro existující systémy a datové řetězce, ale je možné jej použít i pro nové aplikace. To je zajištěno jeho otevřeností. Každá z níže popsanych vrstev je do značné míry nezávislá a může být i nezávisle rozšiřována. Struktura vlastního referenčního modelu OSI je uvedena na **Obrázek 5.1**.

7	aplikační vrstva
6	prezentační vrstva
5	relační vrstva
4	transportní vrstva
3	síťová vrstva
2	linková vrstva
1	fyzická vrstva

Obrázek 5.1: Sedmivrstvá architektura referenčního modelu OSI.

Fyzická vrstva

Fyzická vrstva popisuje elektrické, mechanické, procedurální a funkční specifikace pro aktivaci, deaktivaci a použití fyzického spojení mezi médiem a komunikujícím zařízením. Jsou to například: úroveň napětí, časování změn, maximální délka média, použité konektory. Některé specifikace ve své definici překrývají i linkovou vrstvu, např. Ethernet.

Linková vrstva

Linková (spojová) vrstva definuje způsob spolehlivého přenosu dat přes fyzické síťové médium. Je zde popsáno fyzické adresování, topologie sítě, nebo způsoby zjišťování chyb v přenosu. Linková vrstva bývá rozdělována do dvou podvrstev:

- ▶ podvrstva logického řízení linky (LLC),
- ▶ podvrstva přístupu na médium (MAC).

Důležité je především fyzické adresování na podvrstvě přístupu k médiu. Fyzické adresy (tzv. MAC adresy) definují, jak jsou identifikována komunikační zařízení na úrovni linkové vrstvy.

Síťová vrstva

Síťová vrstva poskytuje funkce zajišťující směrování. To je nezbytné pro existenci různých datových spojů v jedné síti.

Transportní vrstva

Transportní vrstva implementuje metody spolehlivého přenosu dat. Mezi její nejdůležitější funkce patří zejména:

- ▶ řízení přenosu mezi komunikujícími zařízeními tak, že vysílající zařízení neodesílá více dat, než je schopno přijímající zařízení přijmout,
- ▶ koncentrování dat z více zdrojů do jediného fyzického spoje metodou multiplexu,
- ▶ provoz virtuálních spojů (sestavení, využití a ukončení),
- ▶ zajištění detekce a opravy chyb přenosu.

Relační vrstva

Základním úkolem relační vrstvy je sestavení, provoz a ukončení komunikační relace mezi jednotlivými prvky nadřazené prezentační vrstvy. Komunikační relace je tvořena vyžádáním služby a odezvou na požadavek služby.

Prezentační vrstva

Úkolem prezentační vrstvy je zajištění kódování, konverze a šifrování dat tak, aby byly informace čitelné v jiné aplikační vrstvě (realizované na jiném systému).

Aplikační vrstva

Aplikační vrstva slouží přímo uživatelskému programu. Její funkcí je především identifikace komunikujících partnerů, určení dostupnosti potřebných zdrojů a synchronizace komunikace.

5.1.2 Informační jednotky přenosu dat

V sítích se při přenosu dat používají jisté jednotky, tzn. balíky dat. Jejich názvosloví kolísá a je obvykle zaměňováno. Jedná se především o pojmy: rámec, paket, datagram, segment, zpráva a datová buňka.

Rámec

Rámec je informační jednotka, jejíž zdroj a příjemce leží na linkové vrstvě modelu OSI. Rámec se skládá ze záhlaví a závěru linkové vrstvy, mezi nimiž jsou umístěna data nadřazené vrstvy.

Paket

Paket je informační jednotka, jejíž zdroj a příjemce leží na úrovni síťové vrstvy. Paket je opět ohraničen záhlavím a závěrem, mezi nimiž jsou umístěna data nadřazené vrstvy.

Datagram

Datagram (často zaměňován za paket) je informační jednotka, jejíž zdroj a příjemce leží na úrovni síťové vrstvy stejně jako v případě paketu. Rozdíl mezi paketem a datagramem je v kontextu jejich používání. Pojem datagram se používá tam, kde spojení je již sestaveno, tzn. u pevných okruhů.

Segment

Segment je informační jednotkou na úrovni transportní vrstvy.

Zpráva

Zpráva je označením datové jednotky, kde zdroj i příjemce leží nad síťovou vrstvou.

Datová buňka

Označuje informační jednotku pevné velikosti, kde zdroj i příjemce leží v linkové vrstvě. Datová buňka obsahuje pouze záhlaví a data.

5.2 Typy sítí

Počítačové sítě lze dělit podle kritérií uvedených na začátku kapitoly 5.

5.2.1 Sítě podle rozlehlosti

Lokální síť – LAN (Local Area Network)

Lokální síť (LAN) je síť pokrývající zhruba jednu místnost až jednu budovu¹ (zhruba 5-100 koncových zařízení). LAN spojuje, především osobní počítače, pracovní stanice, servery a ostatní zařízení poskytující sdílené služby (např. síťové tiskárny, scannery, atd.). Vzdálenosti propojení bývají poměrně krátké (řádově desítky metrů). V LAN se používají také propojovací zařízení pro propojení jednotlivých segmentů sítě a pro její spojení s okolím. Jsou to zejména

- ▶ **Rozbočovače** (HUB) - zařízení, kde dochází k větvení datového spoje. Data přicházející z jednoho nebo více směrů jsou předávána do jednoho nebo více jiných směrů. Z topologického hlediska představuje HUB sběrnici zborcenou do jednoho bodu.
- ▶ **Přepínače** (switch) - zařízení, ve kterém se, podobně jako v rozbočovači, schází několik datových spojů. Na rozdíl od rozbočovače dochází v přepínači k automatické volbě směru odcházejících dat. Data se šíří na základě znalosti přilehlých síťových segmentů do jednoho určitého spoje, který vede k cílovému zařízení.
- ▶ **Směrovače** (router) - zařízení, které určuje další uzel, do kterého mají být předána data na cestě směřující do místa jejich určení. Směrovač tedy propojuje minimálně dvě podsítě LAN, přičemž určuje cestu pro každý datový paket, který jím prochází.

Metropolitní síť – MAN (Metropolitan Area Network)

Metropolitní síť (MAN) je síť, která propojuje uživatele určitých geografických oblastí nebo regionů, které jsou zpravidla rozlehlejší než plochy, které pokrývají sítě lokální, ale menší než pokrytí sítí rozlehlých (WAN). Pojem MAN má charakter spíše administrativní a používá se pro označení určité části rozlehlé sítě vymezené územím příslušného města/obce a spadající pod jednu správní instituci. Po stránce technologické mají městské sítě znaky sítí rozlehlých (viz níže). V rámci městských sítí dochází k propojení určitého počtu sítí lokálních přes společný páteří (tzv. backbone) segment.

Rozlehlé sítě – WAN (Wide Area Network)

Rozlehlé sítě (WAN) pokrývají rozsáhlé geografické oblasti. WAN zajišťují propojení lokálních a městských sítí na celostátní i mezinárodní úrovni. Tato propojení jsou realizována určitým počtem mezilehlých síťových uzlů, směrovačů WAN, ke kterým jsou připojeny hraniční směrovače propojovaných lokálních nebo městských sítí. Zásadním rysem sítí WAN je skutečnost, že přenosové kanály (tzv. okruhy) jsou zpravidla ve vlastnictví společností,

¹ Definice rozsáhlosti je velmi individuální, uvedená čísla jsou pouze orientační.

provozujících komunikační služby, a jsou pronajímány dalším organizacím nebo jednotlivým uživatelům.

Technologie přenosů v rozlehlých sítích jsou založeny na dvou principech podporovaných příslušnými protokoly

- ▶ Přepínání přenosových spojů.
- ▶ Přepínání datových paketů, které je implementováno
 - datagramovou službou nebo
 - vytvářením virtuálních spojů.

Pro síť WAN se používá několik vysokorychlostních technologií, z nichž některé nabízejí i možnost nedatových přenosů:

▶ **ISDN** (Integrated Service Digital Network)

ISDN je sada standardů vytvořených pro digitální přenosy přes běžnou telefonní síť, kde přenosovou infrastrukturu tvoří především metalické vodiče. Služby sítě ISDN jsou poskytovány ve dvou úrovních:

- Základní úroveň. Vytváří komunikační rozhraní BRI (Basic Rate Interface), která je určena pro domácnosti a malé lokální sítě. V BRI jsou k dispozici dva přenosové komunikační kanály B pro uživatelská data s max. přenosovou rychlostí 64 kbps a jeden kanál D pro přenos řídicích dat max. přenosovou rychlostí 16 kbps. Kanály B a D jsou virtuální a vlastní přenos se uskutečňuje na jednom časově sdíleném vodiči.
- Primární úroveň. Vytvářející komunikační rozhraní PRI (Primary Rate Interface), určené pro střední a velké lokální sítě. V PRI je k dispozici 30 kanálů B s max. přenosovou rychlostí 64 kbps a jeden D kanál s max. přenosovou rychlostí 64 kbps (přenos uživatelských dat rychlostí až 1,92 Mbps).

▶ **Frame Relay**

Frame Relay je protokol založený na technice přepínání paketů s vytvářením virtuálních okruhů. Poskytuje zpravidla pouze službu pro datové přenosy. Přenosové rychlosti se pohybují v rozmezí 64 kbps až 2 Mbps podle evropských standardů.

▶ **ATM** (Asynchronous Transfer Mode)

Technologie ATM poskytuje přenosové služby vytvářením pevných nebo přepínaných virtuálních okruhů, které přetrvávají mezi komunikujícími uzly po celou dobu potřebnou pro přenos určeného objemu dat. Data jsou fragmentována do paketů (buněk) pevné délky 53 B. Prvních 5 B buňky jsou režijní informace a dalších 48 B je určeno pro uživatelská data. Přenos dat v ATM se uskutečňuje na sdíleném médiu (optický vodič nebo koaxiální kabel) metodou asynchronního časového sdílení. ATM technologie se vyznačuje možností přenášet datové i nedatové informace. Přenosové rychlosti dosahují v ATM 1,5 Mbps až 622 Mbps. ATM technologie se používá nejen ve WAN ale i pro vytvoření páteřního segmentu velkých lokálních sítí.

▶ **Gigabit Ethernet**

Technologie Gigabit Ethernet představuje novou generaci síťových technologií Ethernet doposud používaných převážně v sítích LAN (viz dále). Gigabit Ethernet se vyznačuje vysokými přenosovými rychlostmi (až 1 Gbps) a značnými vzdálenostmi mezi jednotlivými

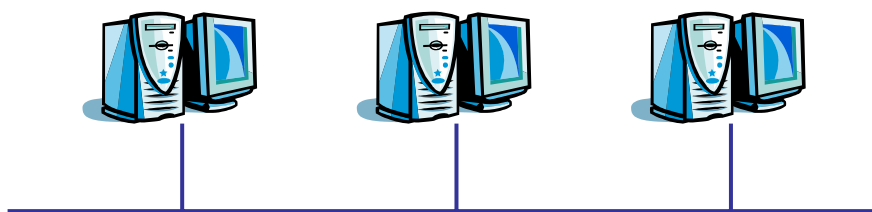
uzly (do 10 km). Výhodou použití Gigabit Ethernetu je to, že síťové komponenty jsou slučitelné s komponenty dalších technologií řady Ethernet, které se obvykle používají v infrastrukturách lokálních sítí. Připojení takových lokálních sítí do rozlehlé sítě technologie Gigabit Ethernet je pak po technické stránce bezproblémové.

5.2.2 Síť podle topologie

Topologie je způsob, jak jsou síťová zařízení v síti organizována. Používají se čtyři základní topologické útvary: sběrnice, strom, kruh, a hvězda. Topologie je tvořena logickým sestavením sítě a skutečné fyzické propojení se může odlišovat.

Sběrníková topologie

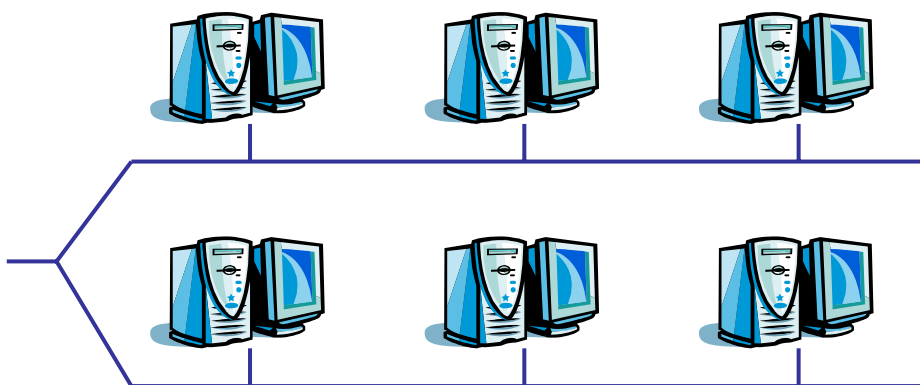
Sběrníková topologie tvoří lineární architekturu, kde se informace z jedné stanice obousměrně přenáší po přenosovém médiu a je současně přijímána všemi ostatními stanicemi. Výhodou je úsporné použití média, nevýhodou je nefunkčnost celé sítě při přerušení sdíleného média na kterémkoliv místě.



Obrázek 5.2: Sběrníková topologie lokální sítě.

Stromová topologie

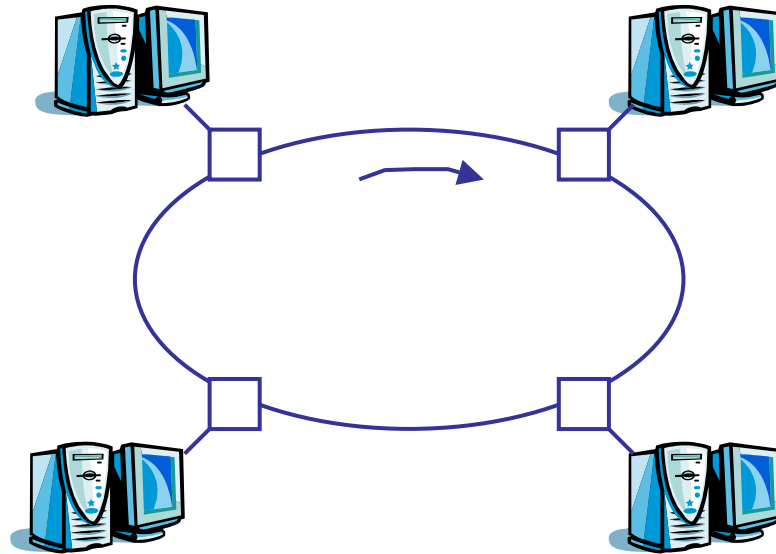
Stromová topologie tvoří násobně lineární architekturu, ve které je také použito sdílené médium jako u sběrníkové technologie. Šíření informace je také obousměrné.



Obrázek 5.3: Stromová topologie lokální sítě.

Kruhová topologie

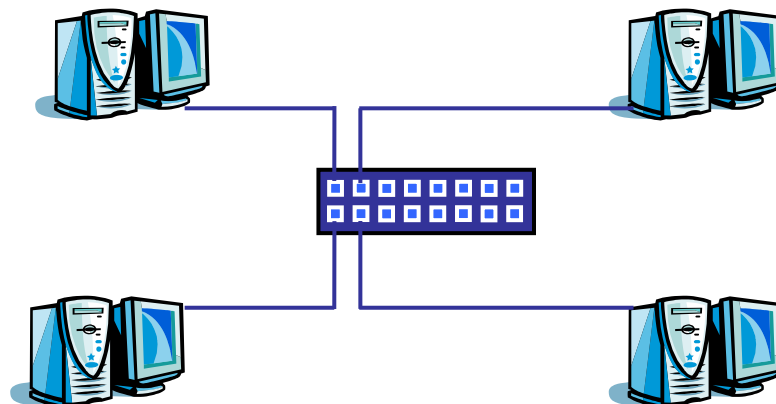
Kruhová topologie je charakteristická smyčkou, kterou tvoří médium. Smyčka je přerušována opakovači, které mají dvojí funkci: připojují uzly k médiu a opakují informaci (přijatou informaci bez zdržení znovu vyšlou dále po směru šíření informace). Důvodem opakování je znovuobnovení informace (např. eliminace útlumu).



Obrázek 5.4: Kruhová topologie lokální sítě.

Hvězdicová topologie

V hvězdicové topologii je každý uzel přímo propojen do centrálního uzlu dvojicí spojů. Tento pár spojů je zpravidla realizován vodičem typu kroucená dvoulinka. Jeden spoj je určen pro příjem signálů, druhý spoj pro vysílání signálů. Centrálním propojovacím bodem může být buď rozbočovač nebo přepínač.



Obrázek 5.5: Hvězdicová topologie lokální sítě.

5.2.3 Síť podle technologie přístupu k médiu

Technologie přístupu k médiu je významnou charakteristikou především u sítí lokálních. Pro lokální síť je typická topologie s násobným přístupem k přenosovému médiu, které je tedy sdíleno všemi připojenými uzly. Techniky řízení přístupu ke sdílenému mediu jsou řešeny několika standardy, které implementují síťová zařízení (rozbočovače, přepínače, síťové adaptory počítačů atd.) použitelná pro určitý typ lokální sítě. Nejznámější a nejčastěji používané standardy jsou:

- ▶ Ethernet nebo IEEE 802.3,
- ▶ Token Ring nebo IEEE 802.5.

Ethernet/IEEE 802.3

Ethernet/IEEE 802.3 technologie jsou v současnosti nejpoužívanější síťové technologie pro LAN. Ethernet/IEEE 802.3 LAN používají přístupovou techniku CSMA/CD (Carrier-Sense Multiply Access with Collision Detection). CSMA/CD technika je založena na náhodném přístupu uzlů k médium v čase, kdy chtějí vysílat. Všechny kolize, které mohou nastat v případech (současné vysílání z více uzlů), jsou vyřešeny postupy CSMA/CD.

Ethernet/IEEE 802.3 LAN (nebo také CSMA/CD LAN) používá především sběrnicovou nebo stromovou topologii. Topologie CSMA/CD sítě je specifikována standardy, které určují kromě také max. rychlost přenosu dat, limity pro délky jednotlivých kabelových segmentů, možný způsob jejich propojení, maximální počet připojených uzlů, typ kabelů, typ kabelových zásuvek a zástrček atd. V současnosti používanými standardy jsou

- ▶ 10Base5, 10Base2 (přenosová rychlost 10 Mbps, sběrnicová topologie, médium koaxiální kabel),
- ▶ 10BaseT (přenosová rychlost 10 Mbps, stromová topologie, médium kroucená dvoulinka),
- ▶ 100BaseTX (přenosová rychlost 100 Mbps - tzv. "Fast Ethernet", médium kroucená dvoulinka, koaxiální kabel, optický kabel),
- ▶ 1000BaseTX/SX/LX (přenosová rychlost 1 Gbps – tzv. "Gigabit Ethernet", médium koaxiální kabel, optický kabel).

Token Ring/IEEE 802.5

Token Ring/IEEE 802.5 technologie se v porovnání s technologií Ethernet používají méně. Pro Token Ring je charakteristická kruhová topologie a metoda řízeného (deterministického) přístupu k přenosovému médium. Metoda používá pro řízené vysílání speciální datový paket zvaný "token" (poukázka), který obíhá v určeném směru mezi uzly sítě uspořádanými do kruhy. Vysílat je oprávněn pouze uzel, který obdržel poukázku. Pokud nemá v úmyslu vysílat žádná data, předá poukázku sousedícímu uzlu po směru komunikace. Je tím zajištěno, že v určitém okamžiku do přenosového média vysílá pouze jeden uzel. Původně byla tato technologie vyvinuta firmou IBM a je využívána ve firemní síťové architektuře SNA.

5.2.4 Sítě podle síťové architektury

Síťová architektura je struktura, která umožňuje výměnu dat mezi počítačovými systémy a splňuje určené požadavky vztahující se k této výměně dat. Pojem síťová architektura může být chápán jako fyzická struktura sítě (topologie) nebo jako logická (abstraktní) struktura popisující jednotlivé vrstvy. V tomto pojetí je síťová architektura nazývána také protokolová architektura.

Protokol je souhrn pravidel, formátů a procedur určující způsob výměny dat mezi dvěma komunikujícími uzly. Protokoly nižších vrstev definují fyzikální standardy, které musí být dodržovány, bitové a bytové pořadí v přenášených paketech, způsoby detekce a korekce chyb, atd. Protokoly vyšších vrstev se zabývají formáty dat, syntaxí zpráv a určují způsob dialogu komunikujících stran. Protokoly jsou implementovány buď hardwarem (např. síťovými adaptéry) nebo softwarem (operačními systémy).

Existuje velké množství síťových architektur, většinou vytvořených a definovaných komerčními subjekty. Mezi nejvýznamnější se řadí:

- ▶ **NetWare** je síťový operační systém firmy Novell, Inc. Zahrnuje řadu protokolů, které často koeexistují na společných fyzických spojích s protokoly jiných architektur, jako jsou TCP/IP nebo AppleTalk.
- ▶ **SNA** (Simple Network Architecture) je síťová architektura firmy IBM Corp.
- ▶ **DECnet** je sada protokolů firmy Digital Equipment Corporation, které tvoří síťovou architekturu DNA (Digital Network Architecture).
- ▶ **AppleTalk** je síťová architektura firmy Apple Computer, Inc. pro počítače typu Macintosh.

Většina dnešních síťových architektur zahrnuje sadu protokolů TCP/IP. TCP/IP je v současnosti nejpoužívanější architekturou pro komunikaci v síti a je standardem Internetu.

5.2.5 Sítě podle typu média

Média slouží k přenosu signálu. Přenosová média jsou podle své fyzikální charakteristiky nejčastěji rozdělena na

- ▶ metalická,
- ▶ optická a
- ▶ ostatní.

Datová komunikace je ve všech případech realizována prostřednictvím elektromagnetických vln. Příkladem metalických médií je kroucená dvoulinka a koaxiální kabel. Příkladem optického média je optický kabel, skládající se ze svazku skleněných nebo plastických vláken). Příkladem ostatních médií je vzduch a vakuum.

Typickými parametry metalických a optických medií je celková přenosová rychlost dat a šířka přenosového pásma, kterou příslušné médium podporuje. Pro ostatní média je základním parametrem také frekvenční rozsah modulovaného signálu, který je nositelem signálu. Používají se tři typy frekvenčních rozsahů:

- ▶ Mikrovlnné frekvence (rozsah asi 2 až 40 GHz) vhodné pro přenosy typu point-to-point. Používají se přesně směřované paprsky vysílaného signálu.
- ▶ Radiové frekvence (rozsah asi 30 MHz až 2 GHz) vhodné pro bezdrátové lokální sítě typu WLAN (Wireless Local Area Network) Používají se všesměrové vysílací antény.
- ▶ Infračervené frekvence (rozsah asi 3×10^{11} až 2×10^{14} Hz) vhodné pro bezdrátové lokální sítě typu WLAN.

Kroucená dvoulinka

Kroucená dvoulinka (twisted pair) je tvořena dvěma spirálovitě zkroucenými izolovanými měděnými vodiči. Tento pár vodičů představuje jeden komunikační spoj. Určitý počet párů vodičů je svázán do kabelu chráněného pevným vnějším obalem. Vzájemné zkroucení jednotlivých párů vodičů minimalizuje jejich vzájemné elektromagnetické interference. Kroucená dvoulinka se vyrábí ve dvou variantách jako stíněná a nestíněná. Nestíněná kroucená dvoulinka UTP (Unshielded Twisted Pair) je levnější než stíněná a je proto běžně používaná v infrastruktuře lokálních počítačových sítí. U stíněné kroucené dvoulinky STP (Shielded Twisted Pair) je každý pár navzájem zkroucených vodičů obalen kovovou síťkou zabraňující interferencí mezi páry vodičů. Protože je tento typ kabelu

mnohem dražší než kabely UTP a také práce s nimi je náročnější, používá se kabeláž STP mnohem méně než kabeláž UTP.

Koaxiální kabel

Koaxiální kabel je tvořen dvěma vodiči. Dutý vnější stínící vodič obaluje vnitřní vodič. Vnitřní vodič oddělený izolací je buď pevný drát nebo lanko. Rovněž vnější vodič může být pevný nebo tvořený sítkou. Celý kabel je ještě obalen pláštěm nebo stíněním.

Koaxiální kabel lze použít v infrastrukturách lokálních počítačových sítí pro různé typy provozu a vzdálenosti. V porovnání s kabely UTP je ovšem kabeláž s koaxiálními kabely mnohem finančně nákladnější. U současných síťových infrastruktur se koaxiálními kabely provádí propojování mezilehlých síťových zařízení, koncové uživatelské stanice se do infrastruktury připojují kabely typu UTP.

Optický kabel

Optický vodič je 2 až 125 μm tenké pružné médium schopné vést optický paprsek. Optický kabel je cylindrický a je složen ze tří soustředných částí. Vnitřní část je jádro, které je obaleno světelně izolační vrstvou a vnějším obalem je plášť chránící funkční část kabelu před mechanickým poškozením, navlhnutím, námrazám apod.

Vnitřní světelně vodivé jádro je tvořeno jedním nebo několika vlákny vyrobenými ze skla nebo průzračného plastu. Materiál vrstvy, která světelné vodiče obklopuje je, světelný izolátor, což znamená, že jeho optické vlastnosti jsou odlišné od optických vlastností jádra.

Optické kabely se používají v infrastrukturách lokálních počítačových sítí a mají oproti metalickým kabelům několik významných předností. Kromě toho se využívají optické kabely také u spojů rozlehlých počítačových sítí (WAN). Mezi výhody optických kabelů v porovnání s vodiči metalickými patří zejména:

- ▶ vyšší přenosová kapacita,
- ▶ menší rozměry a hmotnost,
- ▶ menší přenosový útlum,
- ▶ odolnost proti elektromagnetickému rušení.

5.3 Architektura TCP/IP

Architekturu **TCP/IP** (Transmission Control Protocol/Internet Protocol) vytváří základní sada komunikačních protokolů Internetu². Je používána také v sítích privátních (intranet). Implementace protokolů TCP/IP je základní podmínkou pro možnost přímého připojení počítače nebo jiného zařízení do Internetu.

Sada protokolů architektury TCP/IP má čtyři vrstvy, které lze vztáhnout k referenčnímu modelu OSI (viz **Obrázek 5.6**). Architektura TCP/IP zahrnuje více protokolů, zobrazeny jsou pouze nejdůležitější nebo nejpoužívanější.

² V Internetu se používá zkráceného názvu TCP/IP protokol, ačkoliv se jedná o sadu protokolů, pojmenovanou podle dvou nejdůležitějších částí.

vrstva OSI	protokoly					vrstva TCP/IP
7	SMTP	FTP	TELNET	SNMP	DNS	aplikační vrstva
6	HTTP	IMAP	POP	RTP	RIP	
5						
4	TCP				UDP	transportní vrstva
3	IP				ICMP	sít'ová vrstva
2	protokoly sít'ového rozhraní					sít'ové rozhraní
1						

Obrázek 5.6: Protokoly sady TCP/IP ve vrstvách OSI a vrstvách TCP/IP.

5.3.1 Vrstvy architektury TCP/IP

Vrstva síťového rozhraní

Vrstva síťového rozhraní zajišťuje přístup k přenosovému médium a řízení datového spoje. Vrstva není rámci architektury TCP/IP podrobněji specifikována. Její implementace závisí na přenosové technologii, kterou používá příslušná síť, např. Ethernet pro LAN nebo ATM pro WAN). TCP/IP může být implementována na jakémkoliv typu sítě z hlediska používané přenosové technologie, tzn. na sítích heterogenních.

Síťová vrstva

Síťová vrstva je charakterizována především protokolem IP, bývá proto také nazývána vrstva IP. Důležitým protokolem této vrstvy je také protokol ICMP.

Protokol IP (Internet Protocol) podporuje komunikaci mezi síťovými uzly a zajišťuje na základě schématu IP adresace směrovací cesty pro pakety, které se v kontextu této vrstvy nazývají datagramy. IP protokol také provádí fragmentaci datagramů v případě, že objem dat přesahuje maximální velikost rámce, který může přijmout cílový uzel. Tu stanoví příslušná přenosová technologie (např. Ethernet specifikuje maximální velikost rámce 15000 B). IP protokol provede případně také znovusestavení datagramu na straně příjemce, tzv. defragmentaci přijatých dat.

Protokol **ICMP (Internet Control Message Protocol)** podporuje a doplňuje protokol IP tak, že informuje vysílací uzel o nekorektních situacích, které nastaly v průběhu přenosu datagramu. Jedná se tedy o zprávy režijní, které se v případě korektních přenosů nevysílají.

Transportní vrstva

Transportní vrstva zajišťuje službu předání dat mezi koncovými uživatelskými procesy. Transportní vrstva zahrnuje dva důležité protokoly: TCP protokol a UDP protokol.

Protokol **TCP (Transmission Control Protocol)** poskytuje tzv. spojovanou službu. To znamená, že vytvoří mezi komunikujícími koncovými procesy virtuální komunikační kanál, ve kterém probíhá řízený (potvrzovaný) přesun dat. Je to služba zajišťující spolehlivý přesun dat bez ztrát a duplikací datových segmentů.

Protokol **UDP** poskytuje tzv. nespojovanou službu bez potvrzování příjmu předávaných datagramů. Přesun dat probíhá rychleji než v předchozím případě, korektnost transakce však musí ověřit protokoly vyšší vrstvy (tj. aplikační).

Aplikační vrstva

Aplikační vrstva je nejvyšší vrstvou TCP/IP architektury zajišťující uživatelským aplikacím přístup k službám Internetu. Dále zajišťuje správu relací koncových aplikací (jedná se o relace typu klient – server) a přípravu dat k transakci. Aplikační vrstva TCP/IP sdružuje funkce nejvyšších třech vrstev referenčního modelu OSI. Do aplikační vrstvy je přiřazeno velké množství protokolů určených různým službám Internetu. Některé protokoly poskytují tutéž službu a programátor volí pro aplikaci, kterou vyvíjí, ten protokol, který odpovídá nejvíce jeho záměrům. Určitá skupina aplikačních protokolů jsou protokoly povinné, neboť operační systém, který by je neimplementoval, by nemohl poskytovat ani minimální síťový servis. Další skupinu tvoří protokoly doporučené, které podporují základní síťové služby a poslední skupinou jsou protokoly doporučené, podporující rozšířené síťové služby. Vývoj nových aplikačních protokolů neustává, naopak počet nových aplikačních protokolů se neustále zvyšuje v souvislosti s novými službami, které Internet nabízí (např. protokoly pro podporu multimediálních přenosů v reálném čase).

Mezi nejpoužívanější aplikační protokoly TCP/IP sady patří např. následující protokoly zajišťující uživatelům internetové služby

- ▶ TELNET – protokol virtuálního vzdáleného terminálu,
- ▶ FTP (File Transfer Protocol) – protokol podporující přenos souborů mezi vzdálenými systémy,
- ▶ SMTP (Simple Mail Transport Protocol) – protokol podporující internetovou elektronickou poštu,
- ▶ IMAP (Internet Message Access Protocol) a POP (Post Office Protocol) – protokoly umožňující přístup uživatelům do poštovních schránek ze vzdálených počítačů,
- ▶ HTTP (Hypertext Transfer Protocol) – protokol podporující distribuované informační systémy používající hypertextové dokumenty.

Následující protokoly nezprostředkovávají přímo uživatelské služby, ale spíše služby systémové, vytvářející pro uživatele funkční pracovní prostředí:

- ▶ směrovací protokoly, např. RIP (Routing Information Protocol), OSPF (Open the Shortest Path First), BGP (Border Gateway Protocol) – pro podporu vytváření přenosových cest datagramů,
- ▶ DNS (Domain Name System) - protokol pro podporu mapování doménových jmen uzlů do IP adres a naopak,
- ▶ SNMP (Simple Network Management Protocol) – protokol podporující monitoring, správu a koordinaci TCP/IP sítě,
- ▶ PPP (Point-to-Point Protocol) – protokol umožňující připojení uživatele počítače přes komutovanou telefonní linku do sítě (tzv. "dial – up"),
- ▶ RTP (Real Time Protocol) – protokol podporující přenosy dat v reálném čase.

Protokoly sady TCP/IP jsou vyvíjeny a udržovány nezávisle na typu hardware nebo operačního systému a jsou implementovány na různých systémových platformách. TCP/IP je základním síťovým protokolem operačních systémů UNIX a je implementován ve všech ostatních současných operačních systémech (Microsoft operační systémy, IBM OS/2, Novell, atd.). TCP/IP sítě mohou být provozovány na různých přenosových technologiích jako je Ethernet, IEEE 802.3, Token Ring, FDDI a bezdrátové síťové systémy.

5.3.2 Adresace v architektuře TCP/IP

Adresní schéma implementované v protokolu IP verze 4, který je v současnosti jedním ze základních protokolů Internetu, umožňuje teoreticky jednoznačně identifikovat až 4.294.967.296 síťových uzlů. IP adresa je tvořena 32bitovým binárním slovem identifikujícím odesílatele a příjemce informace, která se ve formě datových paketů přenáší Internetem. IP adresa má dvě části: část označující určitou (pod)síť a část označující určité zařízení (PC, server, směrovač, ...) náležející do té sítě. Správcem IP adresového pole Internetu je organizace NIC (Network Information Center).

Na Internetu (tj. na směrovačích, které předávají paket mezi sebou směrem k cíli) se zkoumá pouze část IP adresy označující síť. Teprve v rámci cílové sítě se paket doručuje podle identifikátoru zařízení, zvaném také lokální adresa. Každá síť připojená k Internetu musí mít svou adresu.

Vzhledem k tomu, že existující sítě různých velikostí, byly pro adresní schéma IP navrženy 4 třídy IP adres, označených A, B, C a D. IP adresy se zpravidla zapisují jako čtveřice dekadických čísel (s hodnotou 0-255), z nichž každé vyjadřuje hodnotu osmice bitů. Čísla jsou navzájem oddělena tečkou, např. 195.178.72.10.

Třída A je určena velmi velkým sítím s velkým počtem síťových zařízení. Pro adresu třídy A platí, že první bit první osmice má hodnotu 0 a dalších 7 bitů je určeno pro identifikaci sítě. Zbýlých 24 bitů lze využít pro identifikaci síťových zařízení. Příklad IP adresy třídy A: 68.122.150.12.

Třída B je určena sítím střední velikosti. Pro adresu třídy B platí, že první dva bity první osmice mají hodnotu 10 a dalších 14 bitů je určeno pro identifikaci sítě. Zbýlých 16 bitů lze využít pro identifikaci síťových zařízení. Příklad IP adresy třídy B: 147.22.250.112.

Třída C je určena pro sítě menší, adres tohoto typu je k dispozici nejvíce. V současnosti je to prakticky jediný typ IP adres, který se organizacím přiděluje. Pokud by k adresaci podnikové sítě nestačila jedna IP adresa, je podniku přidělena skupina více IP adres. Pro adresu třídy C platí, že první 3 bity první osmice mají hodnotu 110 a dalších 21 bitů je určeno pro identifikaci sítě. Zbýlých 8 bitů lze využít pro identifikaci síťových zařízení. Příklad IP adresy třídy C: 197.2.58.12.

Třída D je určena pro skupinovou adresaci a slouží k označení určité skupiny zařízení, která byla vytvořena s určitým záměrem (např. správa sítě, skupinové multimediální přenosy atd.). Jednotlivá zařízení skupiny mohou náležet do různých IP sítí. Takové zařízení má tedy současně IP adresu síťovou i IP adresu skupinovou. Pro adresu třídy D platí, že první 4 bity první osmice mají hodnotu 1110 a dalších 28 bitů je určeno pro identifikaci skupiny. Příklad IP adresy třídy D: 225.2.58.12.

Tabulky rozsahu IP adres v jednotlivých třídách jsou uvedeny v **Tabulka 5.1** v dekadickém vyjádření a v **Tabulka 5.2** v binárním vyjádření.

<i>Třída</i>	<i>1. byte</i>	<i>2. byte</i>	<i>3. byte</i>	<i>4. byte</i>
A	adresa sítě 0xxxxxxx	adresa zařízení xxxxxxx xxxxxxxx xxxxxxxx		
B	adresa sítě 10xxxxxx xxxxxxxx		adresa zařízení xxxxxxx xxxxxxxx	
C	adresa sítě 110xxxxx xxxxxxxx xxxxxxxx			adresa zařízení xxxxxxx

Tabulka 5.1: Tabulka rozsahu IP adres v třídách A, B, C v binárním vyjádření.

<i>Třída</i>	<i>1. byte</i>	<i>2. byte</i>	<i>3. byte</i>	<i>4. byte</i>
A	adresa sítě 1 - 126	adresa zařízení 0.0.1 - 255.255.254		
B	adresa sítě 128.0 - 191.255		adresa zařízení 0.1 - 255. 254	
C	adresa sítě 192.0.0 - 223.255.255			adresa zařízení 1 - 254

Tabulka 5.2: Tabulka rozsahu IP adres v třídách A, B, C v dekadickém vyjádření.

Kromě síťové IP adresy, která je adresou logickou, má každé síťové zařízení svou adresu hardwarovou, danou výrobcem. Tato adresa se označuje někdy jako MAC adresa nebo Ethernet adresa. Je vyjádřena binárním slovem délky 48 bitů a obvykle se zapisuje jako šest dvojic hexadecimálních čísel navzájem oddělených dvojtečkou (např.: 08:00:20:91:DC:83). Hardwarové adresy jsou používány síťovým rozhraním.

IP adresy jsou numerickými identifikátory používanými protokoly sady TCP/IP k určení uzlu sítě. Pro uživatelské aplikace jsou však vhodnějšími identifikátory síťových uzlů alfanumerické řetězce, které mohou mít nějaký význam a jsou lépe zapamatovatelné.

Každé přidělené IP adrese je proto možné přiřadit alfanumerický řetězec, který se stává tzv. **doménovým jménem zařízení**. Doménové jméno zahrnuje jméno zařízení a označení subdomény a domény, ke které náleží. Jednotlivé složky doménového jména jsou odděleny tečkou. Například doménové jméno ant.feec.vutbr.cz je přiřazeno zařízení s IP adresou 147.229.192.10. Jméno zařízení je "ant" a přísluší do domény "feec.vutbr.cz". "vutbr" je označení části domény označené "cz".

Všechna doménová jména Internetu jsou hierarchicky uspořádána do systému **Domain Name System** (DNS). Jeho vrchol tvoří "kořenová" doména, spravovaná NIC. Další úroveň tvoří státní nebo oborové domény (tzv. domény nejvyšší úrovně). Pod nimi jsou domény druhé, třetí, atd. úrovně, které jsou ve správě organizací provozujících lokální počítačové sítě.

Uživatelské programy používají k označení síťového zařízení spíše doménová jména, než číselné IP adresy. Protože ale protokoly nižších vrstev TCP/IP vyžadují pro svou činnost IP adresy a nikoliv doménová jména, musí být doménová jména na IP adresy konvertována. Tento převod provádí funkce implementující protokol DNS, které jsou vestavěny do každé

síťové aplikace. Konverze doménových jmen na IP adresy je tedy pro uživatele aplikace zcela transparentní.

Prostor doménových jmen je administrativně rozdělen do oblastí (oblastí je často podniková nebo univerzitní LAN). Každé zařízení náležející do určité oblasti má svůj záznam v tzv. souboru oblastí, který je databází pro službu DNS. Soubory oblastí jsou udržovány na vyhrazeném počítači zvaném "name server" nebo DNS server. DNS je tedy distribuovaný systém Internetu, protože informační zdroje podporující DNS servis jsou rozptýleny na mnoha "name serverech", odpovídajících za příslušnou oblast.

Současná verze IP protokolu (verze 4) přestává vyhovovat stávajícím požadavkům na síťové služby. Exponenciální rozšiřování Internetu tento problém ještě násobí. Hlavní problémy IPv4 jsou:

- ▶ Pro adresaci používá pouze 32 bitů. Adresová oblast se tedy může časem vyčerpat s nárůstem počtu zařízení, pro která bude požadováno připojení do Internetu.
- ▶ Směrovací tabulky na páteřních směrovačích Internetu se neustále zvětšují, čímž na těchto zařízeních vznikají problémy s negativním dopadem na provoz Internetu. Aby se snížila na těchto směrovačích časová ztráta při datových přenosech, bylo by třeba vyvinout velmi náročné HW a SW technologie.
- ▶ IPv4 neumožňuje, aby aplikace požadovaly speciální služby (tzv. QoS – Quality of Service) pro přenosy časově závislých dat, např. multimediální přenosy.
- ▶ IPv4 neposkytuje možnost, aby aplikace mohly žádat vyšší stupeň zabezpečení přenosu dat a autentizaci.

V důsledku výše uvedených nedostatků současné verze IP protokolu zahájila IETF (Internet Engineering Task Force) vývoj nového standardu. Výsledek byl publikován v dokumentu RFC2460 z r. 1998 popisujícího IP nové generace s oficiálním názvem IP verze 6 (IPv6). Jeho nejvýznamnější rozšíření jsou:

- ▶ Používá větší adresové pole. IPv6 adresa má 128 bitů a je možno adresovat až $3,4 \cdot 10^{38}$ zařízení.
- ▶ Zajišťuje rychlejší zpracování režijních informací při průchodu směrovači.
- ▶ Definuje mechanismy zajišťující autenticitu, důvěrnost a celistvost pro IP pakety.
- ▶ Umožňuje aplikacím označit IP pakety tak, aby jim bylo poskytnuto při přenosech speciální zacházení (např. při multimediálních přenosech v reálném čase).

IPv6 adresa je zapisována jako osm čtyřmístných hexadecimálních čísel oddělených dvojtečkou. Zkrácený zápis umožňuje, aby se v adrese na jednom místě vyskytly dvě dvojtečky vedle sebe (dané místo bude doplněno adekvátním počtem nul).

Protokol IPv6 rozlišuje 3 druhy adres:

- ▶ Unicast - adresa je přiřazena jednomu síťovému rozhraní.
- ▶ Anycast - adresa je přiřazena několika síťovým rozhraním a paket je doručen na libovolné z nich.
- ▶ Multicast - adresa je přiřazena skupině zařízení a paket je doručen na všechna tato zařízení.

V protokolu IPv6 není možné používat libovolné adresy; adresové místo je podle prefixů děleno následovně:

- ▶ 001 - globální unicastové adresy,
- ▶ 1111 1110 10 - lokální unicastové adresy na jednom segmentu,
- ▶ 1111 1110 11 - lokální unicastové adresy v jedné organizaci,
- ▶ 1111 1111 - multicastové adresy,
- ▶ adresa 0::xxxx:xxxx je IPv4 adresa při tunelování IPv4 přes IPv6,
- ▶ adresa 0::FFFF:xxxx:xxxx je IPv4 adresa pro zařízení, která nepodporují IPv6.

Hlavička IPv6 byla oproti hlavičce IPv4 výrazně zjednodušena. Chybějící funkce byly implementovány pomocí zřetěžených hlaviček. Hlavička IPv6 paketu obsahuje jednu položku "next header". To je typ další hlavičky, která následuje. Tato další hlavička obsahuje opět položku "typ hlavičky", což je typ hlavičky, která za touto následuje. V poli "next header" rovněž může být typ protokolu (např. TCP, UDP) – to znamená, že už žádné další hlavičky nejsou a datová oblast se předá ovladači daného protokolu.

5.3.3 Datagramy v architektuře TCP/IP

Datagramy jsou v architektuře TCP/IP opatřovány hlavičkami, které nesou důležité informace nutné k uskutečnění spolehlivého přenosu zprávy od odesílatele k příjemci. Struktura výsledného datagramu je znázorněna na **Obrázek 5.7**.



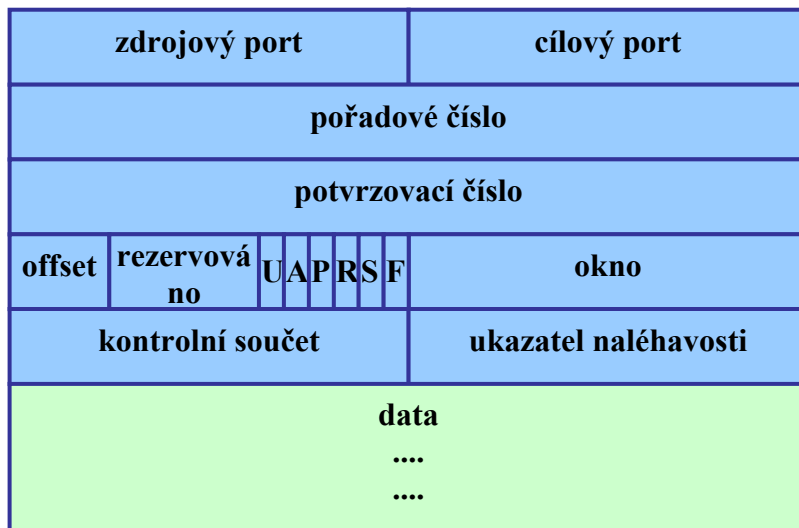
Obrázek 5.7: Struktura datagramu v architektuře TCP/IP.

Protokol TCP zajišťuje rozdělení zprávy do datagramů na straně odesílatele, sestavení zprávy z datagramů na straně příjemce, znovuposlání ztracených nebo poškozených datagramů a seřazení datagramů podle ve správném pořadí.

Podobně jako jiné protokoly, TCP předřadí datagramu hlavičku obsahující informace nutné ke správné komunikaci. Ta může probíhat v tzv. multiplexním režimu, kdy existuje více spojení mezi odesílatelem (zdrojem) a příjemcem (cílem). Odlišení mezi jednotlivými spojeními je dáno pomocí tzv. portů.

Hlavička protokolu TCP (viz. **Obrázek 5.8**) obsahuje nejméně 20 bytů, které jsou použity pro několik parametrů. Nejdůležitější z nich jsou tyto:

- ▶ zdrojový port – číslo portu odesílatele,
- ▶ cílový port – číslo portu příjemce,
- ▶ pořadové číslo – pořadové číslo datagramu (čísluje se v bytech s krokem rovným délce datagramů),
- ▶ potvrzovací číslo – číslo přijatého datagramu (je vyplněno jen u potvrzovacího datagramu vyslaného zpět odesílateli),
- ▶ kontrolní součet – "součet" hodnot bytů v datagramu.



Obrázek 5.8: Datagram s hlavičkou TCP.

6 Internet a jeho služby

V roce 1969 zahájila Agentura pro rozvojové projekty ministerstva obrany USA **DARPA** (Defense Advanced Research Projects Agency) výzkumný projekt, jehož cílem byly nové techniky a technologie pro přenosy dat mezi různými typy sítí. Předmětem výzkumu byl vývoj komunikačních protokolů, které by umožnily síťovým počítačům transparentně vzájemně komunikovat přes neomezený počet datových spojů. Pokusná reálná síť, která byla pro výzkumné záměry zprovozněna, byla pojmenována ARPANET a systém sítí, kterým se výzkum teoreticky zabýval byl označován jako "Internet". Sada protokolů, která byla v rámci projektu vyvinuta, byla pojmenována jako sada TCP/IP po dvou nejvýznamnějších protokolech celé sady, protokolu TCP (Transmission Control Protocol) a protokolu IP (Internet Protocol). Základní požadovanou charakteristikou sítě byla vzájemná nezávislost jednotlivých uzlů. To by ve válečných podmínkách umožňovalo i v případě, že by některé z uzlů byly zničeny, provozovat komunikaci mezi uzly dosud funkčními. Po období „studené války“ byly výsledky výzkumu poskytnuty civilní sféře a další vývoj komunikačních protokolů se stal středem zájmu univerzitních výzkumných pracovišť.

V roce 1986 iniciovala organizace National Science Foundation (NSF) vývoj páteřní sítě NFSNET, která by byla oporou stále se rozšiřujícího ARPANETu. Od roku 1989 se již nehovoří o ARPANETu, ale o Internetu a tato síť se postupně rozšiřovala přes všechny světadíly počínajíc Evropou. Postupně se také vymaňovala z akademické sféry a stále více se komercializovala. V současnosti pokrývá Internet celý civilizovaný svět a počet jeho uživatelů neustále vzrůstá. Vliv Internetu zasahuje do mnohých oblastí lidských aktivit. Internet se stal nejen mocným komunikačním prostředkem, ale také zdrojem nevyčerpatelného množství informací nejrůznějšího charakteru. Požadavky na stále nové typy služeb, které by Internet mohl poskytovat, jsou hybnou silou pro rozvoj jeho technologií.

I když Internet nemá žádného vlastníka, jsou jeho administrativa a rozvoj zajišťovány činností organizace Internet Architecture Board (IAB), ustavené v roce 1983. V současnosti IAB zahrnuje několik složek, z nichž Internet Engineering Task Force (IETF) je pověřena rozvojem protokolů sady TCP/IP a jejich následnou standardizací. IETF soustřeďuje asi 50 pracovních skupin, řešících konkrétní úkoly. Výsledkem práce pracovních skupin jsou nové nebo aktualizované TCP/IP standardy, zvané "RFC dokumenty". "RFC" (Request for Comments) je historický název dokumentů, připravených k připomínkovému řízení před jejich dokončením z doby, kdy probíhal výzkumný projekt ARPANET. RFC dokumenty, kterých je nyní více než 2000, jsou opatřeny číselným identifikátorem a jsou volně k dispozici na Internetu.

Další složky působící nezastupitelně v administrativě Internetu jsou:

- ▶ Internet Assigned Numbers Authority (IANA), která je pověřena centrálním přidělováním nejrůznějších číselných i jiných identifikátorů sloužících technologickým i správním účelům v rámci Internetu.
- ▶ RFC Editor, která je pověřena publikováním a správou RFC dokumentů.
- ▶ Internet Registry (IR), provádějící centrální údržbu DNS (Domain Name System) distribuované databáze.
- ▶ Network Information Center (NIC), což je celá řada center rozptýlených na Internetu poskytujících rady, pomoc, informace, přístup k dokumentaci a další služby uživatelům Internetu.

6.1 Historické mezníky Internetu

1969

V roce 1969 vznikl v DARPA výzkumný projekt, jehož cílem byl vývoj technologií pro přenos dat mezi různými typy sítí. Předmětem výzkumu byl vývoj komunikačních protokolů, které by umožnily počítačům vzájemně komunikovat přes neomezený počet datových spojů. První počítačová síť se skládala ze 4 propojených uzlů a byla pojmenována ARPANET:

Uzel 1: [University of California Los Angeles](#) (UCLA),

Uzel 2: Stanford Research Institute (SRI, dnes [SRI International](#)),

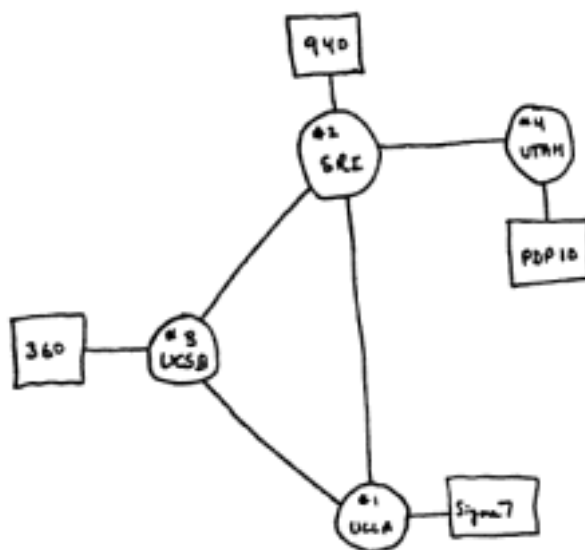
Uzel 3: [University of California Santa Barbara](#) (UCSB),

Uzel 4: [University of Utah](#).

Uzly byly vybaveny zařízením firmy [Bolt Beranek and Newman, Inc.](#) (BBN, minipočítače Honeywell DDP-516 s 12 kB paměti), propojení zajistila společnost [AT&T](#) (telefonní linky s přenosovou rychlostí max. 50 kb/s).

1971

V roce 1971 už síť ARPANET obsahuje 15 uzlů s 23 počítači: UCLA, SRI, UCSB, University of Utah, BBN, [Massachusetts Institute of Technology](#) (MIT), [RAND](#), SDC, [Harvard University](#), [Lincoln Laboratory](#) na MIT, [Stanford University](#), [Upper Iowa University](#), [Case Western Reserve University](#), [Carnegie Mellon University](#), [Ames Research Center](#) v NASA.



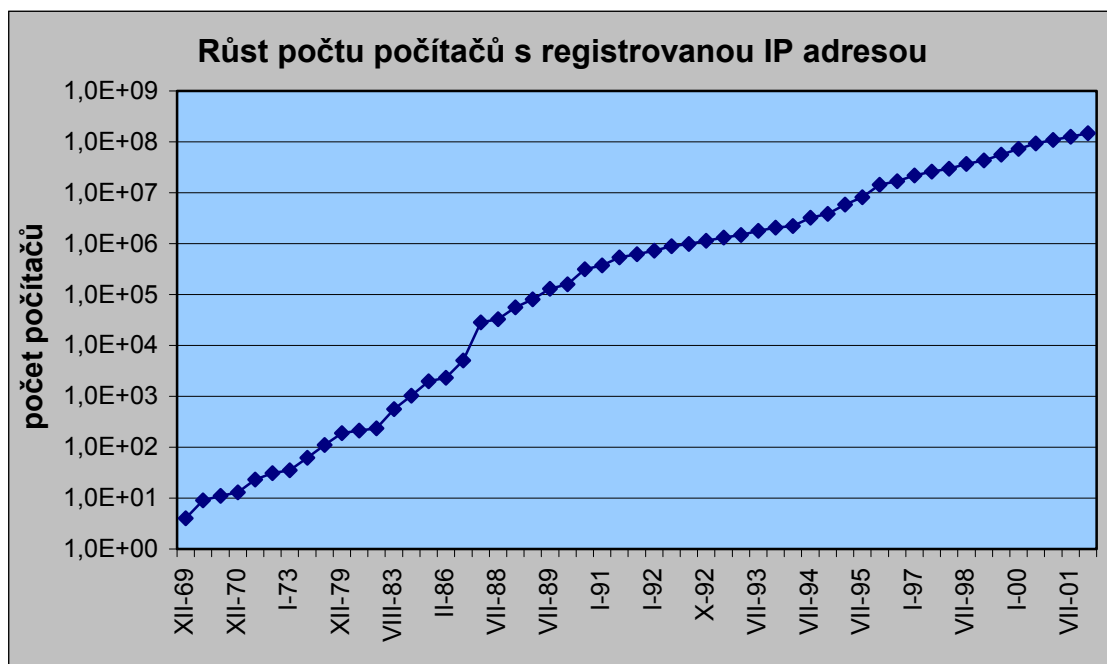
Obrázek 6.1: Originální náčrt sítě ARPANET se 4 uzly.

1973

Bylo uskutečněno první mezinárodní připojení k síti ARPANET z University College of London za pomoci NORSAR (Norsko).

Sada protokolů, která byla v rámci projektu vyvinuta, byla pojmenována jako sada TCP/IP. Základní požadovanou charakteristikou sítě byla vzájemná nezávislost jednotlivých uzlů. Postupně byly výsledky výzkumu poskytnuty civilní sféře a další vývoj komunikačních protokolů se stal středem zájmu především univerzitních výzkumných pracovišť.

V roce 1986 iniciovala organizace National Science Foundation (NSF) vývoj páteční sítě NFSNET, která byla oporou rozšiřujícího se ARPANETu. V roce 1989 se již ujal název Internet a tato síť se postupně rozšiřovala, nejprve do Evropy. Postupně se také stále více komercializovala. V současnosti pokrývá Internet celý svět a počet jeho uživatelů zhruba exponenciálně roste. Ke dni 1.12.1969 byly registrovány pouze 4 počítače, dnes (ke dni 1.1.2002) je to více než 147 mil. počítačů (viz **Graf 6.1**). Další podrobnosti a aktualizované počty počítačů jsou dostupné v dokumentu [Hobbes' Internet Timeline v5.6](#).



Graf 6.1: Počet počítačů s registrovanou IP adresou připojených k Internetu v logaritmickém měřítku.

6.2 Principy Internetu

Internet představuje globální informační systém, který:

- ▶ je logicky spojený v prostoru jedinečných adres, daných protokolem IP nebo jeho následným rozšířením eventuálně následným pokračováním,
- ▶ je schopný podporovat komunikaci prostřednictvím sady protokolů TCP/IP nebo jejím následným rozšířením eventuálně následným pokračováním,
- ▶ zajišťuje, používá nebo zpřístupňuje veřejné nebo privátní uživatelské služby založené právě na existenci komunikační infrastruktury.

Primární síťová architektura používaná Internetu je TCP/IP. Ta také tvoří základ Internetu daný:

- ▶ adresovým systémem podporovaným protokolem IP, který používá numerické adresy (např. 195.178.72.100) a umožňuje, aby jeden uzel měl přiděleno více takových adres,
- ▶ systémem pro transport paketů podporovaný protokolem TCP,
- ▶ systémem doménových jmen podporovaným protokolem DNS, umožňujícím používat pro označení uzlů doménová jména,

- ▶ adresací informačních zdrojů distribuovaných systémů Internetu pomocí URL (Uniform Resource Locator) zahrnující specifikaci metody přístupu k informačnímu zdroji, identifikaci uzlu a místa uložení informační zdroje, a
- ▶ přístup k informačním zdrojům prostřednictvím aplikací podporujících protokol HTTP.

URL je adresa souboru (obecně informační jednotky) nebo služby přístupné prostřednictvím Internetu. Typ tohoto informačního zdroje určuje aplikační protokol, pomocí něhož je zdroj dostupný. Přístupová metoda, která je součástí URL, informuje Web prohlížeč, jaký aplikační protokol má použít pro přístup k požadovanému informačnímu zdroji. URL zahrnuje jak specifikaci přístupové metody ke zdroji, tak umístění vlastního zdroje.

Nejčastější přístupová metoda je použití protokolu HTTP, který umožňuje přístup nejen k webovým stránkám. Formát URL je následující (včetně nepovinných částí umístěných do hranatých závorek):

```
metoda://adresa_serveru[:port]/cesta/[soubor]
```

Přístupová metoda je většinou označena zkratkou příslušného přístupového protokolu:

- ▶ http,
- ▶ https,
- ▶ ftp,
- ▶ mailto,
- ▶ gopher.

Adresa serveru může být zadána textově (např. www.feec.vutbr.cz) v případě, že je na straně web klienta použit protokol DNS, nebo přímo numerická IP adresa (např. 147.229.192.10). U adresy serveru může být specifikováno číslo portu, na kterém bude probíhat komunikace (např. 8080).

Cesta určuje umístění souboru v souborovém systému serveru, ke kterému klient přistupuje. Soubor sám nemusí být specifikován. Například při použití protokolu HTTP je webový server většinou nastaven tak, že automaticky hledá v příslušném adresáři soubor [index.html](#) nebo [welcome.html](#).

Příkladem URL webových stránek je

```
http://www.feec.vutbr.cz/~provaznik
```

a URL serveru ftp pak

```
ftp://brumla.feec.vutbr.cz.
```

6.3 Základní služby Internetu

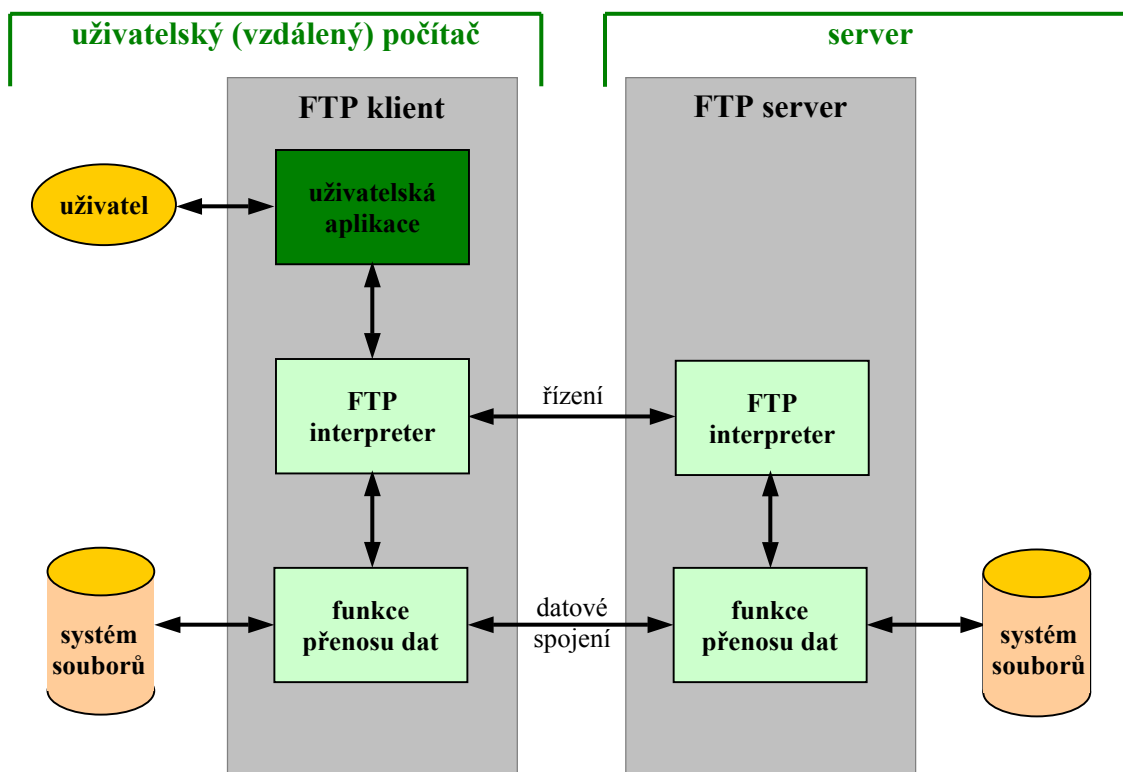
- ▶ Přenos souborů přes Internet (**služba FTP**). Podporováno aplikačním protokolem FTP (File Transfer Protocol), který umožňuje stažení, nahrání, vytvoření, smazání souboru a řadu dalších souborových operací na vzdáleném počítači.
- ▶ Elektronická pošta (**služba E-mail**). Podporováno sadou protokolů (SMTP, MIME, POP, IMAP), které umožňuje vytvořit E-mailovou zprávu, odeslat a přijmout zprávu, vzdálený přístup k E-mailové schránce.
- ▶ Vzdálený přístup k hostitelskému počítači (**služba TELNET**, resp. **SSH**). Podporováno protokoly TELNET a SSH (Secure Shell), které umožňují uživateli

připojit se k hostitelskému stroji z jakéhokoliv počítače Internetu a vytvořit tak virtuální terminál.

- WWW (**služba World Wide Web**, zkráceně **web**). Podporováno protokolem HTTP, který umožňuje přenos hypertextových dokumentů včetně multimediálních dat.
- Diskusní systém (**služba NetNews, též Usenet**). Podporováno protokolem NNTP, který umožňuje šíření příspěvků účastníků moderovaných nebo nemoderovaných diskusních skupin.

6.3.1 Přenos souborů přes Internet

Soubory slouží jako základní prostředek k uložení dat v počítači. V síti počítačů (a také v Internetu) je třeba zajistit vzdálený přístup k souborům tak, aby uživatelé mohli sdílet společná data. Základní operace potřebné ke sdílení jsou: stažení souboru (download), nahrání souboru (upload), odstranění souboru, vytvoření nového souboru. V Internetu je třeba navíc zajistit přenos souborů mezi počítači různých platform s různými operačními systémy. K tomu slouží protokol FTP (File Transfer Protocol) náležící do protokolové sady TCP/IP. Přístup je založen na principu klient/server.



Obrázek 6.2: Princip přenosu souborů v Internetu s využitím služby FTP.

6.3.2 Elektronická pošta

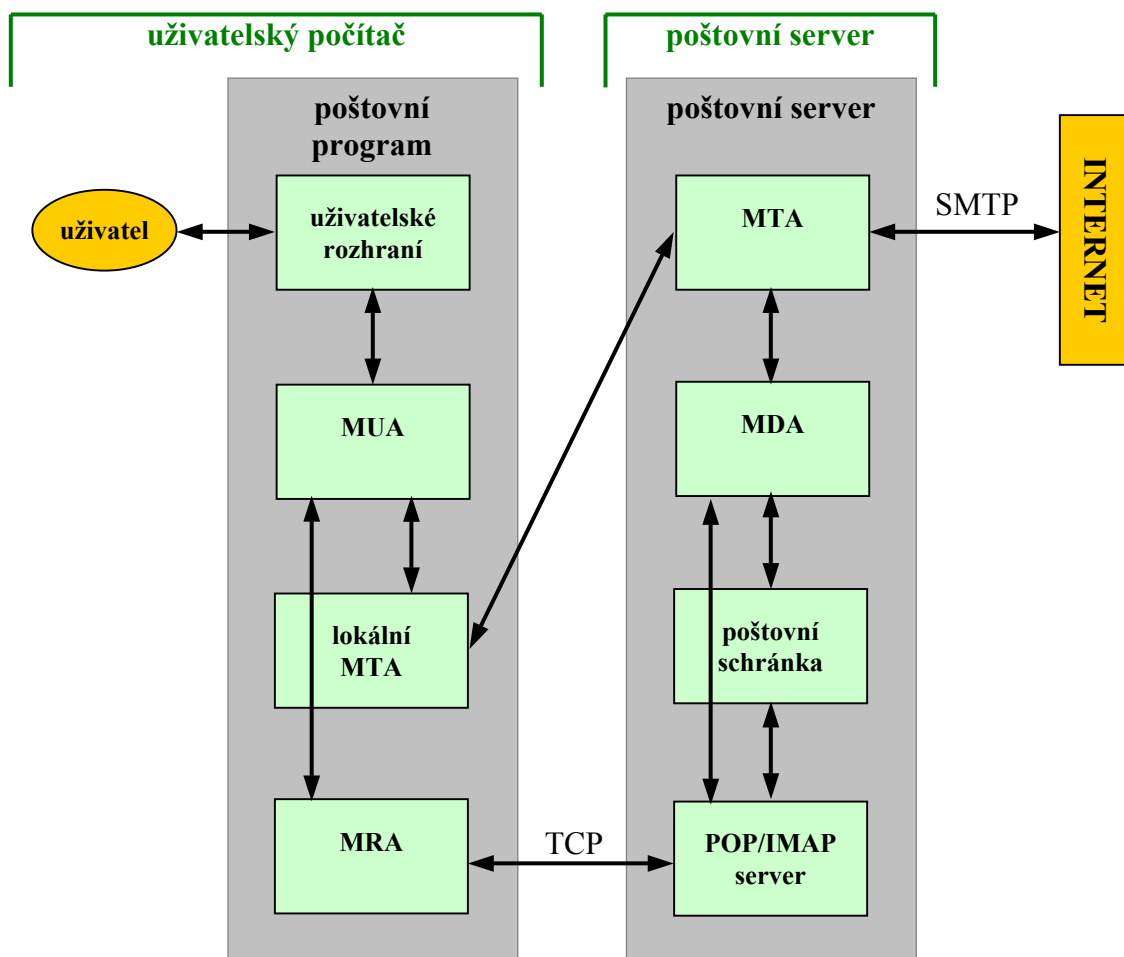
Systém elektronické pošty Internetu zahrnuje několik logických modulů. Nejvýznamnější z nich jsou:

- Mail Transfer Agent (MTA) – program typu mail server, který je provozován na počítači, jehož úkolem je zajišťovat poštovní transakce. Tento program sám nedoručuje zprávy do jednotlivých poštovních schránek, pouze zprávy přebírá a

zjišťuje podle jejich obálek, zda jsou přijatelné (tj. zda jejich adresát má zřízení na počítači uživatelský účet). Převzaté zprávy předá modul MTA modulu MDA, který je ukládá do určených poštovních schránek.

- ▶ Mail Delivery Agent (MDA) – programový modul, který od modulu MTA přebírá zprávy a ukládá je do poštovních schránek adresátů nebo je předává pro konečné doručení dalšímu modulu MTA.
- ▶ Mail User Agent (MUA) – program, který zprostředkuje uživateli odeslání poštovní zprávy a vyzvednutí došlých zpráv z jeho poštovní schránky. Psaní a čtení zpráv bývá zpravidla součástí tohoto programu, nicméně se mohou tyto úkony provádět samostatně pomocí textového editoru.
- ▶ Mail Retrieval Agent (MRA) – uživatelský program nebo modul uživatelského programu, který zprostředkuje přístup do jeho poštovní schránky, umístěné na vzdáleném mail serveru. Modul MRA potom předá zprávy umístěné ve vzdálené schránce modulu MUA.

Model systému elektronické pošty Internetu je zobrazen na **Obrázek 6.3**.



Obrázek 6.3: Logické moduly elektronické pošty.

Jednotlivé moduly elektronické pošty mají k dispozici několik protokolů. Jsou to zejména:

- ▶ protokol SMTP (Simple Mail Transfer Protocol),

- ▶ protokol MIME (Multi-Purpose Internet Mail Extension),
- ▶ protokol POP (Post Office Protocol), a
- ▶ IMAP (Internet Message Access Protocol).

Modul MTA

Modul MTA zajistí, že poštovní zprávy určené k přenosu mají správný formát, a že jsou odeslány směrem k adresátovi. Moduly MTA implementují protokol SMTP (Simple Message Transfer Protocol), pomocí něhož si předávají zprávy. Protokol SMTP je založen na principu klient-server. Modul MTA se stává SMTP klientem v případě, že má zprávu k odeslání a pokud je vyzván jiným MTA k převzetí zprávy, stává se SMTP serverem. Součástí protokolu SMTP je připojení tzv. obálky ke zprávě. Obálka je tedy záležitostí MTA programu na rozdíl od tzv. záhlaví zprávy, které v programu MUA již během vytváření zprávy. Informace nezbytné pro korektní doručení zprávy se do obálky vkládají automaticky programem MTA.

Převzaté poštovní zprávy mohou být zpracovány několika způsoby v závislosti na adrese příjemce uvedené v obálce:

- ▶ pokud jméno příjemce odpovídá označení jedné z existujících poštovních schránek, zpráva je předána modulu MDA, který zprávu do té schránky uloží,
- ▶ pokud je adresa příjemce neplatná, chybná zpráva je vrácena odeslateli,
- ▶ pokud je adresa příjemce platná, ale není lokální, předá se zpráva dalšímu poštovnímu serveru, který může zajistit její doručení do adresátovy schránky.

Možnost využívat službu elektronické pošty nevyžaduje vždy instalaci lokálního MTA modulu. Poštovní zprávy určené k odeslání mohou být předány prostřednictvím modulu MUA odesílatele (pomocí protokolu SMTP) do MTA modulu poskytovatele internetových služeb (ISP), který zprávu dále předá MTA modulu adresáta. Moduly MUA jsou implementovány nejrůznějšími klientskými aplikačními programy (např. Outlook Express pro OS MS Windows nebo PINE pro OS Unix). Nejpoužívanější implementace MTA modulů pro OS Unix je program sendmail.

Modul MDA

Každý MTA modul spolupracuje s moduly MDA, které ukládají zprávy do příslušných poštovních schránek jednotlivých uživatelů. Poštovní schránky jsou součástí souborového systému počítače, který provozuje poštovní služby. MDA moduly jsou dvojího typu:

- ▶ MDA moduly pro lokální doručování zpráv, které ukládají poštovní zprávy do lokálních schránek.
- ▶ MDA moduly pro vzdálené doručování zpráv, které podle pokynů uložených v určitých konfiguračních souborech předávají poštovní zprávy dalším modulům MTA.

Modul MUA

Moduly MUA zprostředkovávají uživateli přímé čtení doručených poštovních zpráv uložených v lokální poštovní schránce a také odeslání nově vytvořené poštovní zprávy na uvedenou adresu. Pokud se jedná o poštovní schránku umístěnou na vzdáleném mail serveru, spolupracuje MUA modul s MRA modulem za pomoci protokolů POP nebo IMAP.

Některé typy modulů MUA umožňují dále:

- ▶ filtrovat příchozí poštovní zprávy podle nastavených pravidel,

- ▶ zjišťovat periodicky stav poštovní schránky,
- ▶ interpretovat netextový obsah zprávy nebo k ní připojených souborů,
- ▶ odesílat odpověď na zpáteční adresu nebo odesílat přijatou zprávu na další specifikované adresy,
- ▶ připojit k odesílané zprávě soubory libovolného typu (text, HTML dokumenty, ...),
- ▶ změnit základní aktuální nastavení elektronické poštovní služby (např. změnit poštovní server, způsob zobrazení zprávy, typ použitého kódování odesílaných zpráv apod.),
- ▶ manipulovat s lokálními i vzdálenými poštovními schránkami,
- ▶ vytvářet adresář elektronických adres,
- ▶ atd.

Modul MRA

Modul MRA umožňuje přístup ke zprávám uloženým na vzdáleném poštovním serveru a jejich doručení na lokální MUA modul na uživatelském počítači. Modul MRA má k dispozici dva protokoly z protokolové sady TCP/IP. Jsou to:

- ▶ Protokol IMAP (Internet Message Access Protocol), který kromě přenosu zpráv ze vzdálené poštovní schránky umožňuje i přístup do dalších schránek, ve kterých uchovává uživatel již dříve přijaté zprávy, dále prohlížení záhlaví zpráv, manipulaci se zprávami ve vzdálených schránkách, atd.
- ▶ Protokol POP3 (Post Office Protocol) umožňující pouze přenos zpráv ze vzdálené poštovní schránky do lokálního MUA modulu uživatele.

Protokol SMTP

Protokol SMTP patří stejně jako FTP a TELNET k základním aplikačním protokolům protokolové sady TCP/IP. Protokol SMTP zajišťuje mechanismus pro předávání poštovních zpráv mezi jednotlivými hostitelskými počítači na principu klient-server. V takové transakci je klientem počítač, který iniciuje relaci za účelem předání zprávy a serverem je počítač, který zprávu přebírá. Protokol SMTP dále umožňuje vytváření uživatelských poštovních skupin, odesílání přímých odpovědí na zprávu nebo její zaslání (postoupení) dalšímu adresátovi. Protokol nespecifikuje vlastní způsob vytváření zprávy, avšak její formát musí odpovídat specifikaci podle dokumentu RFC822. Jakmile je zpráva odesílatelem napsána, protokol SMTP ji převezme a pomocí transportního protokolu TCP naváže spojení s hostitelským počítačem adresáta (příjemce). Během následné SMTP transakce je zpráva předána do počítače adresáta a tam je uložena do příslušné poštovní schránky.

Podle RFC822 se jeví poštovní zpráva jako "obálka" s "obsahem". Obálka obsahuje systémové informace nutné pro uskutečnění předání a převzetí zprávy. Obsah zprávy je objekt, který má být doručen příjemci. Standardní obsah zahrnuje záhlaví zprávy, které má pevně stanovený formát, a vlastní obsah zprávy (tělo zprávy). Záhlaví tvoří určená klíčová slova následovaná příslušnými argumenty. Příklad obsahu poštovní zprávy je:

```
Date: Wed, 26 Jul 2000 13:14:15
From: provazni@seznam.cz
To: provazni@feec.vutbr.cz
Subject: Test - ignorujte
```

```
Test E-mail - ignorujte?
```

První čtyři řádky zprávy tvoří záhlaví, poslední řádek je pak tělo zprávy.

Obecná elektronická adresa má formát: `uživatelské_jméno@poštovní_server`. Část adresy před znakem `@` obvykle představuje adresátovo uživatelské jméno v operačním systému poštovního serveru.

Protokol MIME

Protokol SMTP je určen pro doručování jednoduchých textových zpráv. V dnešní době se stále více vyskytují zprávy, jejichž obsahem jsou různé datové formáty včetně hlasových a obrazových. Takové požadavky splňuje protokol MIME (Multi Purpose Internet Mail Extension).

Specifikace podle protokolu MIME rozšiřuje záhlaví poštovní zprávy o informace, které se vztahují k obsahu zprávy. Specifikace MIME zahrnuje pět definičních polí:

- ▶ `MIME-version` - verze protokolu (má hodnotu 1.0),
- ▶ `Content-type` - typ dat obsažených v těle zprávy (Text, Multipart, Message, Image, Video, Audio, Application),
- ▶ `Content-transfer-encoding` - kódovací metoda použitá při přenosu dat (7bit, 8bit, binary, base64, ...),
- ▶ `Content-id` - jednoznačný identifikátor zprávy použitelný pro případný odkaz na zprávu,
- ▶ `Content-description` - text popisující obsah objektu, který obsahuje tělo zprávy (používá se v případě, že objekt je nezobrazitelný, např. zvuková data).

Protokol MIME specifikují podrobně standardy RFC2045 až RFC2049 a implementují jej všechny novější verze klientských poštovních aplikací a samozřejmě také implementace poštovních serverů.

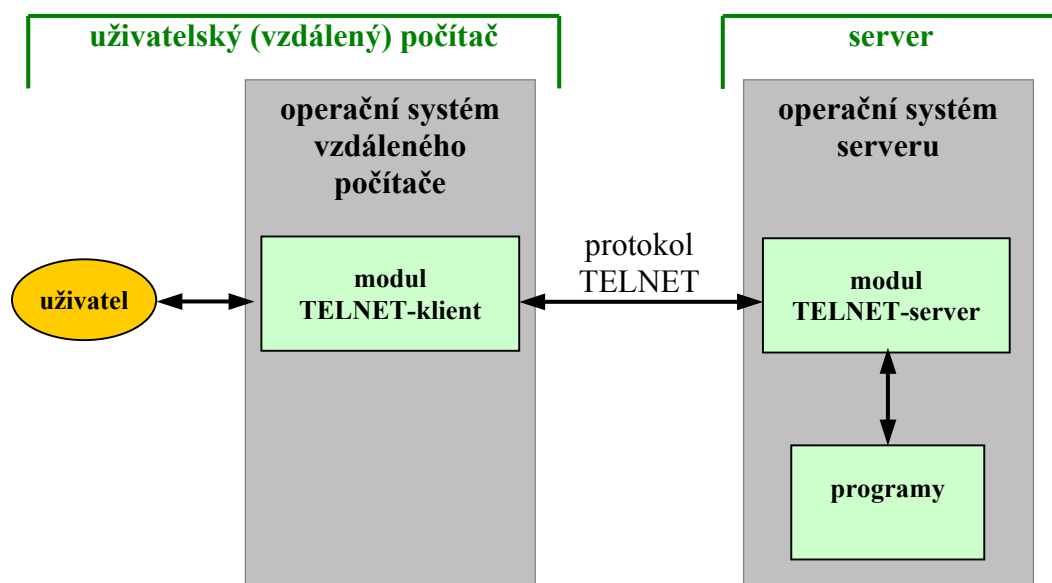
6.3.3 Vzdálený přístup k hostitelskému počítači

TELNET je služba umožňující připojení uživatele ke vzdálenému počítači-serveru. Uživatel musí mít na serveru zřízen svůj účet, opravňující jej využívat jeho výpočetní zdroje.

Služba TELNET je založena na protokolu TELNET, který je stejně jako FTP, jedním ze základních aplikačních protokolů sady TCP/IP. Protokol TELNET zajišťuje uživateli pracujícím na terminálu nebo počítači třídy PC možnost připojit se ke vzdálenému počítači, přičemž na jeho lokálním počítači vznikne stejné pracovní prostředí, jako by byl k hostitelskému počítači připojen přímo. Implementace protokolu TELNET zahrnuje dvě části:

- ▶ Aplikační program-klient TELNET, což je modul vytvářející interaktivní komunikační rozhraní a konvertující jednotlivé znaky skutečného (lokálního) terminálu do síťového standardu a naopak.
- ▶ TELNET server, což je program spuštěný na hostitelském počítači. Je prostředníkem mezi vzdáleným terminálem a programy a utilitami vlastního systému, které uživatel během relace spouští ze svého síťového terminálu. Tento vzdálený terminál se pak jeví systémovým utilitám a programům jako terminál lokální.

Přenos dat mezi klientským a serverovým modulem se uskutečňuje prostřednictvím transportního protokolu TCP. Princip modelu TELNET klient-server je znázorněn na **Obrázek 6.4**.



Obrázek 6.4: Princip připojení ke vzdálenému počítači pomocí služby TELNET.

Bezpečné (tj. odolné proti odposlechu) připojení ke vzdálenému počítači poskytuje protokol **SSH** (Secure Shell). Komunikace protokolem SSH je zabezpečena ve dvou rovinách:

- ▶ obě strany spojení klient-server jsou autentizovány za použití digitální certifikace,
- ▶ hesla jsou chráněna šifrou.

Pro spojení a autentizaci používá SSH kryptografický systém s veřejnými klíči RSA a pro šifrování přenášených dat kryptografické systémy se soukromými klíči Blowfish, DES a IDEA.

Protokol SSH chrání před:

- ▶ IP spoofing - počítač odesílá pakety, které předstírají, že odcházejí z jiného uzlu,
- ▶ IP source routing - počítač přijímá pakety, které předstírají, že přicházejí z jiného uzlu,
- ▶ DNS spoofing - útočník falšuje záznam databáze DNS serveru,
- ▶ odposlechem dat (hesla apod.) prováděného z jiného uzlu sítě,
- ▶ manipulací s daty prováděnou neoprávněnými osobami.

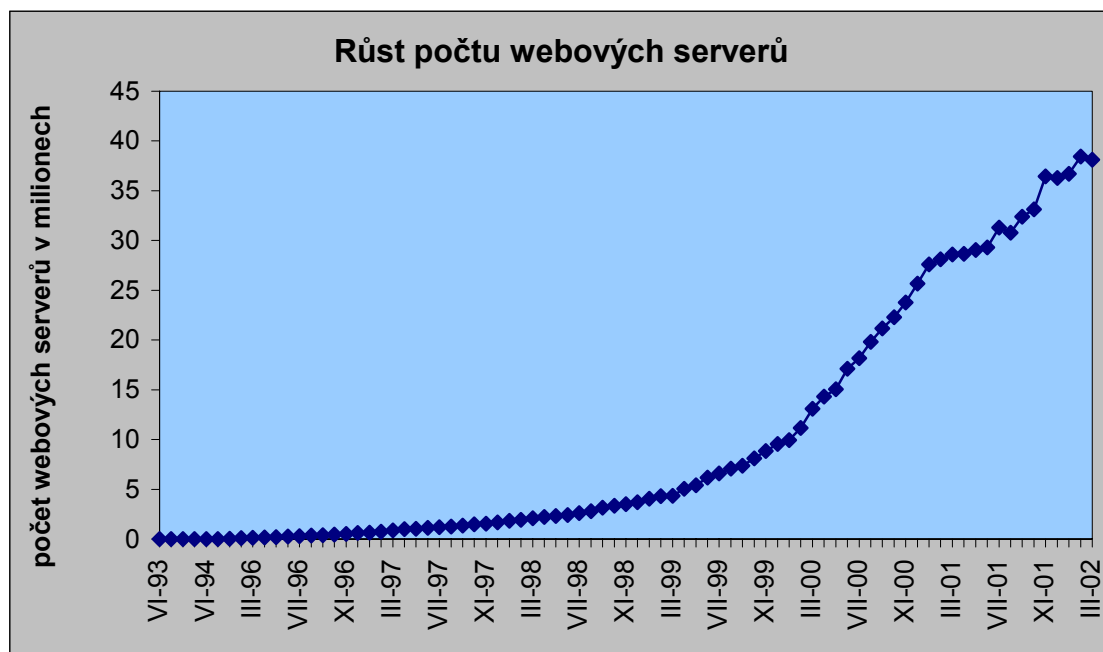
6.4 Služba Web

Služba World Wide Web (Web nebo WWW nebo W3) je systém internetových serverů, které uchovávají hypertextové dokumenty a umožňují k nim přístup. Hypertext je způsob organizace informační jednotky. Obsahuje odkazy na další související informační jednotky, čímž se tato informační jednotka stává nelineárním informačním zdrojem. Systém WWW je podporován aplikačním protokolem HTTP (Hypertext Transfer Protocol), příslušejícím do protokolové sady TCP/IP a založeném na principu relace klient-server. WWW klient často zajišťuje rozhraní i pro další aplikační protokoly internetových služeb (TELNET, FTP, SMTP), čímž vytváří pohodlné uživatelské prostředí pro přístup k různým službám Internetu.

Činnost systému WWW je založena na hypertextovém přístupu k informačním zdrojům, tzn. přes odkazy obsažené v hypertextových dokumentech. Odkazy mohou odkazovat na různé zdroje, jako např. další hypertextové dokumenty, prosté textové dokumenty, obrazové statické nebo animované zdroje nebo zdroje zvukové a video. Proto bývá WWW systém označován jako systém multimediální.

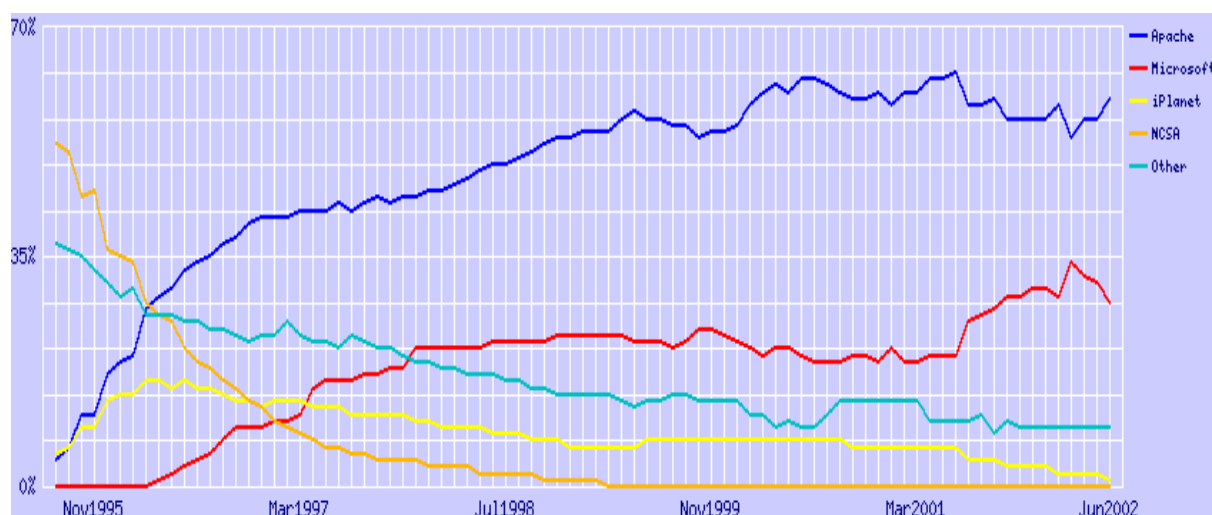
Pro vytváření hypertextových dokumentů je určen zejména jazyk HTML (Hypertext Markup Language), který kromě vytvoření odkazů umožňuje také specifikovat typ písma použitého v dokumentu, grafickou strukturu dokumentu včetně umístění obrazových objektů apod. HTML dokument je tedy předpisem pro WWW klienta, jak má příslušný informační zdroj graficky (nebo zvukově) interpretovat. Dalšími prostředky pro tvorbu webových stránek je dnes Java, JavaScript, CGI, a další.

Služba Web vznikla v roce 1993 a v dnešní době je druhou nejoblíbenější službou Internetu (po elektronické poště). Rozšiřování webu je ohromující a stále má exponenciální charakter. To lze dobře vidět na růstu počtu webových serverů viz **Graf 6.2**. Ke dni 1.6.1993 činil počet serverů pouhých 130, dnes (ke dni 1.3.2002) činí počet serverů více než 38 milionů. Další podrobnosti a aktualizované počty jsou dostupné v dokumentu [Hobbess' Internet Timeline v5.6](#).



Graf 6.2: Počet webových serverů na Internetu.

Z analýzy typu webových serverů ([Apache](#), [Apache Software Foundation](#), [Microsoft](#), [Microsoft Corporation, Inc.](#), [iPlanet](#), nyní součástí [SUN](#) Microsystems, [NCSA](#), [National Center for Supercomputing Applications](#), a [ostatní](#)) používaných v období 11/1995 až 06/2002 lze vysledovat jednoznačný příklon k volně šířitelnému software Apache a k produktům fy Microsoft viz **Graf 6.3**.



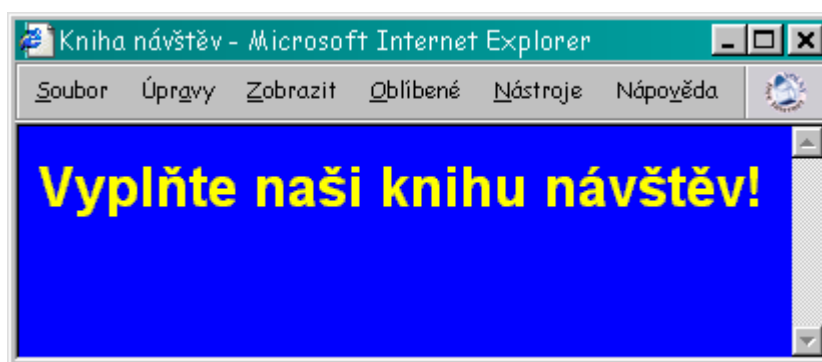
Graf 6.3: Vývoj podílu webových serverů různých výrobců.

6.4.1 Nástroje pro tvorbu webových stránek

I když webové stránky mohou být v jakémkoliv formátu, standardem je jazyk **HTML** (Hypertext Markup Language). Jazyk HTML slouží pro vytváření formátovaného textu, který může obsahovat obrázky, zvuky, animace, video sekvence a odkazy na další webové stránky, respektive její části. Jazyk HTML je podrobně diskutován v kapitole 7.

Příklad velmi jednoduché stránky v napsané jazyce HTML je:

```
<HTML>
<HEAD>
<TITLE>Kniha návštěv</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
CHARSET=iso-8859-2">
</HEAD>
<BODY BGCOLOR="Blue" TEXT="Yellow">
<H1>Vyplňte naši knihu návštěv!</H1>
</BODY>
</HTML>
```



Obrázek 6.5: Příklad jednoduché stránky vytvořené v jazyce HTML.

Tvorba HTML stránek často spočívá v převzetí dříve vytvořených stránek a úpravě HTML kódu. Výše uvedený příklad je uveden ještě jednou (*modře*), nyní ale s komentářem (*červeně*).

```
<HTML>
    označení začátku textu v jazyce HTML
<HEAD>
```


začátek hlavičky

```
<TITLE>Kniha návštěv</title>
```

titulek zobrazený v horní liště prohlížeče

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
CHARSET=iso-8859-2">
```

meta informace o tom, že obsah stránky je v jazyce HTML v kódové stránce iso-8859-2 (středoevropská latinka)

```
</HEAD>
```

konec hlavičky

```
<BODY BGCOLOR="Blue" TEXT="Yellow">
```

začátek těla stránky s parametry barvy pozadí (modrá) a barvy textu (žlutá)

```
<H1>Vyplňte naši knihu návštěv!</H1>
```

nadpis úrovně 1

```
</BODY>
```

konec těla stránky

```
</HTML>
```

konec textu v jazyce HTML

V roce 1996 bylo zavedeno rozšíření jazyka HTML pojmenované **CSS** (Cascading Style Sheets). CSS umožňují přiřadit jednotlivým značkám jazyka HTML určité charakteristiky (např. písmo, barvu, řádkování, apod.). Díky tomu je možné odděleně uchovat nastavení platné pro všechny stránky, které k sobě logicky patří (např. z jednoho webového site). Změna nastavení se pak projeví automaticky ve všech stránkách.

Pro tvorbu webových stránek se také používá mladší jazyk **XML** (Extensible Markup Language). XML je metajazyk, který dovoluje definovat vlastní značky. Díky tomu je poměrně volný a flexibilní.

Některé požadavky uživatelů nebo tvůrců webových stránek nemohou být splněny použitím statických stránek (t.j. neměnných, uložených v souborech). V takovém případě se používají tzv. dynamické stránky, které jsou generovány jako výstup spuštěného programu. Nejčastěji se používají nástroje **CGI** (Common Gateway Interface), **JavaScript**, a **PHP**.

CGI

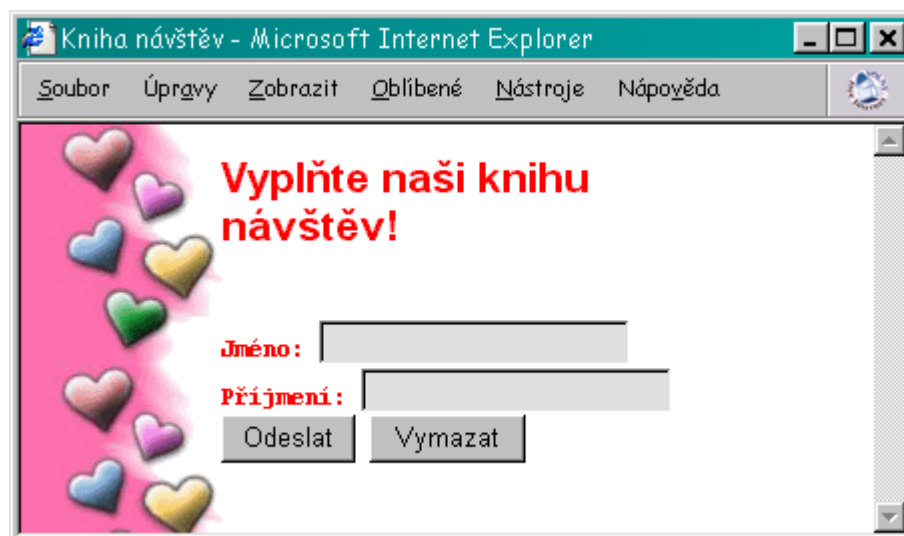
Rozhraní CGI slouží k vytváření skriptů, které tvoří interaktivní uživatelem ovládané aplikace. CGI umožňuje webovému serveru komunikovat s dalšími programy, které na jsou na serveru spuštěny. Pomocí CGI může server vyvolat externí program a předat mu uživatel definovaná data. Příkladem je přenos dat vyplněné uživatelem ve formuláři na webové stránce. Program pak data zpracuje a výstup může směřovat zpět prohlížeči. Například je takto vygenerována nová webová stránka obsahující potvrzení o správnosti dat vložených do formuláře.

Jako příklad použití rozhraní CGI vytvoříme stránku pro knihu návštěv. Uživatel zadá své jméno a příjmení do formuláře a kliknutím na tlačítko "Odeslat" data odešle na server. Obsah stránky (*modře*) i s komentářem (*červeně*) lze zapsat takto:

```
<HTML>
<HEAD>
<TITLE>Kniha návštěv</title>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
CHARSET=iso-8859-2">
</HEAD>
<BODY TEXT="Red" BACKGROUND="background.gif" LEFTMARGIN="100">
```


začátek těla stránky s parametry barvy textu (červená), obrázku na pozadí (background.gif) a odsazení zleva (100 pixelů)

```
<H2>Vyplňte naši knihu návštěv!</H2>
<PRE>
    začátek předformátovaného textu
    <FORM NAME="Formular" METHOD="GET" ACTION="/cgi-bin/guestbook.pl">
        začátek formuláře s parametry jméno (Formular), metoda zaslání dat (GET - viz protokol HTTP níže), data budou zpracována skriptem guestbook.pl v adresáři cgi-bin
        <b>Jméno:</b> <INPUT TYPE="text" NAME="FirstName">
        první textová položka formuláře
        <b>Příjmení:</b> <INPUT TYPE="text" NAME="LastName">
        druhá textová položka formuláře
        <INPUT TYPE="SUBMIT" VALUE="Odeslat"> <INPUT TYPE="RESET"
        VALUE="Vymazat">
        tlačítko Odeslat (při kliknutí provede akci SUBMIT - odeslání dat) a tlačítko Vymazat (při kliknutí provede akci RESET - vymazání vyplněných textových polí)
    </FORM>
    začátek formuláře
</PRE>
konec předformátovaného textu
</BODY>
</HTML>
```



Obrázek 6.6: Příklad stránky s formulářem s použitím skriptu CGI.

Data z formuláře se předají jako parametry do CGI programu (v našem příkladě jsou to parametry `FirstName` a `LastName`). Předání se uskutečňuje pomocí URL nebo těla požadavku. V případě použití metody GET se parametry předají jakou součástí URL. Prohlížeč, která provádí transakci, pak vysílá protokolem HTTP řádek:

```
GET /cgi-bin.guestbook.pl?FirstName=Ivo&LastName=Provaznik HTTP/1.0
```

K oddělení parametrů je použit znak `&`. Webový server pak předává dvojice `parametr=hodnota` CGI programu pomocí proměnných operačního systému.

JavaScript

JavaScript je jednoduchý objektově orientovaný skriptovací jazyk. Univerzální jádro jazyka je součástí většiny prohlížečů. Jedná se o jazyk, jehož kód je vložen přímo do webových stránek a je vykonáván na straně klienta.

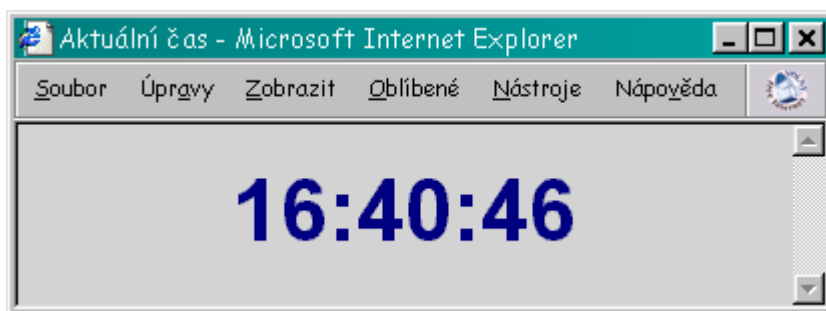
Popis jazyka JavaScript přesahuje rámec textu. Pro ilustraci uvádíme jednoduchý příklad, ve kterém je definována stránka obsahující skript pro průběžné zobrazení aktuálního času.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<title> Aktuální čas </title>
<style type="text/css">
<!--
    body {background-color: lightgrey;}
    #time {font-family: Helvetica, Arial, sans-serif; font-weight:
    700; font-size: 32pt; color: navy;}
//-->
</style>
</head>
<body>
<script language="JavaScript" type="text/javascript">
<!--
    function buildtime() {
        var time = new Date();
        var h,m,s;

        h = time.getHours();
        h = (h<10) ? "0"+h : h;
        m = time.getMinutes();
        m = (m<10) ? "0"+m : m;
        s = time.getSeconds();
        s = (s<10) ? "0"+s : s;
        return(h + ":" + m + ":" + s);}

    function refresh() {
        document.elements.time = document.getElementById("time");
        document.elements.time.innerHTML = buildtime();}
//-->

</script>
<table align="center">
<tr><td valign="middle">
    <script language="JavaScript" type="text/javascript">
    <!--
        document.write("<div id=\"time\">",buildtime(),"</div>");
        document.elements = new Object();
        setInterval('refresh()',100);
    //-->
    </script>
</td></tr>
</table>
</body>
</html>
```



Obrázek 6.7: Ukázka grafického výstupu programu v jazyce JavaScript.

6.4.2 Protokol HTTP

Protokol HTTP (Hypertext Transfer Protocol) definuje způsob, jakým spolu komunikují klient WWW a server WWW. HTTP tak tvoří páteř systému WWW.

Veškerá komunikace prostřednictvím protokolu HTTP má jednotný obecný formát. Na straně klienta (uživatel pracující s prohlížečem) vzniká požadavek, na který odpovídá server (počítač s webovými stránkami). Každý klientský požadavek a odpověď serveru mají tři části:

- ▶ řádek požadavku nebo odpovědi,
- ▶ hlavička,
- ▶ tělo.

Klient iniciuje komunikaci následujícím způsobem:

1. Klient kontaktuje server. Pak posílá serveru **žádost o stránku** pomocí příkazu *metoda*, za kterým následuje adresa stránky a číslo verze protokolu (viz dále 6.4.3).
Například:

```
GET /index.html HTTP/1.1
```

Zde je použita metoda `GET`, kterou žádáme o stránku `index.html` protokolem verze `1.1`.

2. Poté klient nepovinně posílá **hlavičkové informace**, kterými server informuje o své konfiguraci a o typech dokumentů, které přijímá. Jednotlivé hlavičkové informace se posílají řádek po řádku, na každém je uvedeno jméno hlavičky a její hodnota.
Například:

```
User-Agent: Mozilla/4.05 (WinNT; I)
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
...
```

Uvedené hlavičky uvádějí jméno a verzi klienta a některé druhy přijímaných dokumentů. Konec hlaviček klient signalizuje prázdným řádkem.

3. Po odeslání požadavku a hlaviček může klient posílat další **data**. Ty se používají zejména při práci s CGI programy.

Server na požadavek klienta reaguje následujícím způsobem.

1. Server odpovídá **stavovým řádkem**, který obsahuje tři položky: verzi protokolu HTTP, stavový kód a popis. Verze protokolu HTTP je označením verze protokolu, který server použije k odpovědi. Stavový kód je třímístné číslo indikující výsledek zpracování požadavku serveru. Popis je text vysvětlující stavový kód. Například:

```
HTTP/1.1 200 OK
```

Uvedený stavový řádek říká, že server používá protokol verze 1.1. Stavový kód 200 znamená, že požadavek klienta byl úspěšně splněn a za hlavičkami bude následovat požadovaná stránka. Totéž potvrzuje popisný text OK.

2. Za stavovým řádkem posílá server **hlavičkové informace** o sobě a požadované stránce. Například:

```
Date: Fri, 20 Sep 2002 08:17:32 GMT
Server: Apache/1.5
Last-modified: Tue, 19 Sep 2002 22:10:00 GMT
Content-type: text/html
Content-length: 5248
```

Konec hlaviček klient opět signalizuje prázdným řádkem.

3. Pokud je možné požadavek splnit, následují požadovaná data. Data jsou tvořena obsahem souboru stránky nebo výstupem CGI programu. Pokud není možné požadavek splnit, obvyklá data obsahují text vysvětlující, proč byl požadavek neúspěšný.

Od verze HTTP 1.1 je obvyklé, že navázané spojení není po splnění prvního požadavku zrušeno, ale server čeká, zda stejný klient nevznese další požadavek. Tím se šetří režie nutná k opakovanému navázání spojení v případě, že stránka obsahuje další vnořené dokumenty (obrázky, rámce, apod.).

Protokol HTTP je tzv. bezstavový, tzn. neudrží informace o ukončeném spojení. Každé nové spojení tedy začíná zcela od začátku. Výhodou je, že server může obsluhovat velké množství klientů a není zatížen velkou režii uchovávání a aktualizace informací o každém z nich.

6.4.3 Žádosti klienta v protokolu HTTP

Požadavek klienta je rozdělen do tří částí. První řádek vždy obsahuje příkaz protokolu HTTP zvaný **metoda**, dále URI (Uniform Resource Identifier) identifikující soubor nebo prostředek, který klient požaduje, a verzi protokolu HTTP. Druhou část požadavku tvoří **hlavičkové informace**, které obsahují údaje o klientovi a o datech, které serveru posílá. Třetí částí požadavku je **tělo entity** - data která klient posílá serveru.

Poznámka. URI je obecný termín označující všechny platné formáty adresačních schémat, které se v systému WWW používají. Nejčastěji používaným schématem je URL (Uniform Resource Locator).

Protokol HTTP má definovány tři metody: GET, HEAD a POST.

Metoda GET

Metoda GET je nejčastěji používanou metodou žádosti o informace umístěné na serveru dané URI adresou. Datová část požadavku GET je vždy prázdná. Metoda GET je (zjednodušeně) požadavkem "poskytni soubor se jménem ...". Například klient posílá žádost ve tvaru:

```
GET /index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/2.02Gold (WinNT; I)
Host: www.feec.vutbr.cz
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
```

Řádek `Connection: Keep-Alive` požaduje po serveru zachování spojení po odpovědi na tento požadavek. Server může například odpovědět:

```
HTTP/1.0 200 Document follows
```

```
Date: Fri, 20 Sep 2002 08:17:32 GMT
Server: Apache/1.5
Last-modified: Tue, 19 Sep 2002 22:10:00 GMT
Content-type: text/html
Content-length: 5248
{tělo stránky}
```

Metoda GET

Metoda HEAD pracuje obdobně jako metoda GET s tím rozdílem, že server v datové části odpovědi nezasílá žádnou informaci. Metodou HEAD klient získá pouze hlavičkové informace o souboru nebo o prostředku (stejně jako u metody GET).

Tato metoda se používá v případě, mají-li být klientem získány pouze informace o stránce, nikoliv však stránka sama. Například se tak zjišťuje:

- ▶ čas změny stránky (pro práci s vyrovnávací pamětí na straně klienta),
- ▶ velikost stránky (pro odhad přenosového času),
- ▶ typ dokumentu (klient bude číst pouze stránky určitého formátu), apod.

Komunikace metodou HEAD vypadá například takto:

```
HEAD /index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/2.02Gold (WinNT; I)
Host: www.feec.vutbr.cz
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
```

Server může například odpovědět:

```
HTTP/1.0 200 Document follows
Date: Fri, 20 Sep 2002 08:17:32 GMT
Server: Apache/1.5
Last-modified: Tue, 19 Sep 2002 22:10:00 GMT
Content-type: text/html
Content-length: 5248
```

Metoda POST

Metoda post umožňuje klientovi poslat na server data. Data jsou předána programu, který je zpracuje a k němuž má server přístup (například skript CGI). Metoda POST se používá zejména pro

- ▶ zasílání formulářových dat,
- ▶ síťové služby,
- ▶ volání programů s řádkovým rozhraním,
- ▶ databázové operace, apod.

Klientský požadavek metodou POST zaslání formulářových dat (měsíc a den) pro zpracování skriptem CGI `datum.pl` může vypadat takto:

```
POST /cgi-bin/datum.pl HTTP/1.0
User-Agent: Mozilla/2.02Gold (WinNT; I)
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Host: www.feec.vutbr.cz
Content-type: application/x-www-form-urlencoded
Content-length: 18
{prázdný řádek}
month=july&date=22
```

7 Jazyk HTML

Jazyk HTML (Hypertext Markup Language) je prostředkem služby WWW pro popis webové stránky (dále jen stránky). HTML se vyznačuje následujícími principy:

- Popis stránek je vždy a pouze v textovém formátu. Veškerá binární data (obrázky, video sekvence, programy, atd.) jsou umístěna v jiných souborech a na stránce je na tyto soubory vytvořen odkaz.
- Značky (příkazy) jazyka jsou spolu se svými parametry uzavřeny do ostrých závorek "<" a ">".
- Značky jsou buď párové nebo nepárové. Párové značky se vztahují pouze na část stránky, jejíž text je jimi vymezen. Nepárové značky se vztahují na celou stránku.
- Párové značky jsou totožné, pouze ukončující značka začíná znakem "/" Příkladem je <BODY> a </BODY>.

HTML je jazyk a má samozřejmě svou syntaxi definovanou normou (standardem). Pozoruhodnost a unikátnost HTML spočívá v tom, že je sice exaktně definován, ale současně je velice flexibilní z hlediska jeho interpretace. Pokud se tvůrce dopustí i velkého množství chyb, není ještě zpravidla zabráněno zobrazení stránky.

Počet značek jazyka HTML je značně velký (počet parametrů u některých značek taktéž). Bohaté a profesionálně vypadající stránky však lze vytvořit s desítkou značek.

Pro úplnost dodejme, že soubory obsahující popis stránek mají příponu `htm` nebo `html`.

7.1 Základní webová stránka

Soubor HTML vždy obsahuje popis stránky nebo její části. Stránkou je míněna plocha, která se zobrazí v okně prohlížeče a její vzhled může být velice variabilní. I když to dnešní prohlížeče striktně nevyžadují, má stránka obsahovat některé důležité značky. Lze definovat takzvanou základní stránku, tvořená minimální kostrou. Ve správné stránce (z hlediska standardu jazyka HTML verze 4.0) by neměly chybět následující části (nebo značky). Pod uvedením částí nebo značek je vždy příklad.

- označení typu dokumentu, verze použitého HTML
`<!DOCTYPE HTML PUBLIC"-//W3C//DTD HTML 4.0//EN">`
- začátek textu ve formátu HTML
`<HTML>`
- začátek hlavičky
`<HEAD>`
- jméno stránky zobrazované v titulku okna prohlížeče
`<TITLE>Jméno dokumentu</TITLE>`
- konec hlavičky
`</HEAD>`
- začátek těla (zobrazovaného obsahu stránky)
`<BODY>`
- vlastní obsah

.....

- konec těla

</BODY>

- konec textu ve formátu HTML

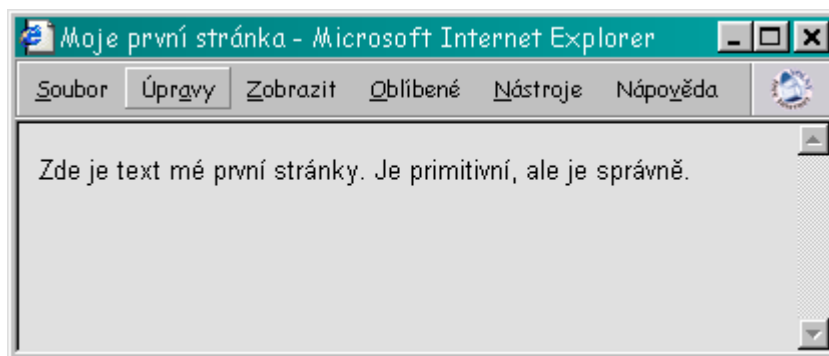
</HTML>

Celá základní stránka může mít tvar uvedený v následujícím příkladu.

Příklad 7.1:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Moje první stránka</TITLE>
</HEAD>
<BODY>
Zde je text mé první stránky. Je primitivní, ale je správně.
</BODY>
</HTML>
```

Stránka je pak v prohlížeči zobrazena například jako na **Obrázek 7.1**.



Obrázek 7.1: Příklad zobrazení základní stránky. Zobrazení se může lišit podle typu prohlížeče, jeho verze, a nastavení prohlížeče.

Při vytváření i té nejjednodušší stránky je nutné znát několik málo základních pravidel syntaxe jazyka:

- HTML nerozlišuje malé a velké znaky. To znamená, že například značka `<HEAD>` může být napsána také `<head>`.
- HTML striktně nevyžaduje určité rozmístění značek na řádcích. Celou stránku je dokonce možné napsat nestrukturovaně na jediný řádek. V zájmu zachování přehlednosti je ovšem lépe zachovat alespoň nějakou strukturu.
- HTML ignoruje znak ENTER (přechod na nový řádek), duplicitní mezery a tabelátory, a to i v zobrazovaném textu. Například text

```
<BODY>
Zde je text mé první stránky. Je primitivní, ale je správně.
</BODY>
```

bude zobrazen naprosto stejně jako text

```
<BODY>
Zde je text mé první stránky.
Je primitivní, ale je správně.
</BODY>
```

7.2 Základy HTML

7.2.1 Tělo stránky

Tělo stránky je definováno značkou `<BODY>`, která plní dvě funkce: vymezuje text, který má být zobrazen, a pomocí svých parametrů definuje jeho vlastnosti. Syntaxe značky `<BODY>` je následující:

```
<BODY BACKGROUND="URL" TEXT="barva" BGCOLOR="barva" LINK="barva"
VLINK="barva" ALINK="barva" LEFTMARGIN="nn" TOPMARGIN="nn">
```

Parametry značky `<BODY>` jsou nepovinné. Jejich definování ale nastavuje důležité vlastnosti zobrazení příslušné stránky. Vynecháme-li parametry, je nastavení dané nastavením prohlížeče.

Parametry značky body definují

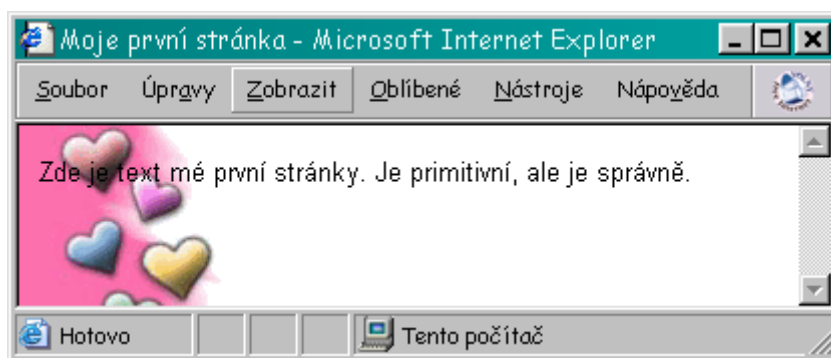
- ▶ obrázek na pozadí,
- ▶ implicitní barvy elementů na stránce a
- ▶ vzdálenosti levého a horního okraje stránky.

Obrázek na pozadí

Obrázek na pozadí je definován parametrem `BACKGROUND`. Za názvem parametru následuje jméno souboru s obrázkem, který zaplní pozadí stránky. Obrázek musí být uložený v datovém typu, který umí prohlížeč zobrazit. Nejpoužívanější formáty jsou JPG (JPEG) a GIF. Příklady použití parametru `BACKGROUND`:

```
<BODY BACKGROUND="pozadi.gif">
<BODY BACKGROUND="adresar/pozadi.gif">
<BODY BACKGROUND="http://www.feec.vutbr.cz/UBMI/pozadi.gif">
```

Příklady postupně ukazují, jak je použit obrázek umístěný v aktuálním adresáři, v podadresáři `adresar`, a na webovém serveru `www.feec.vutbr.cz`.



Obrázek 7.2: Příklad zobrazení základní stránky s obrázkem na pozadí.

Implicitní barvy elementů na stránce

V případě, že není použit obrázek na pozadí, lze stránku jednoduše vyplnit zvolenou barvou. K tomu slouží parametr `BGCOLOR`:

Barva je určena definicí úrovně jasu jednotlivých barevných složek (**R**GB - **č**ervená-**z**elená-**m**odrá). Úrovně jsou dány hodnotou 0-255 v hexadecimálním tvaru. Černá je tedy `000000`, bílá pak `FFFFFF`.

Další parametry definují barvu textu a textových odkazů. `TEXT` určuje barvu textu (implicitně černá), `LINK` barvu nenavštíveného odkazu (implicitně modrá `0000FF`), `VLINK`

barvu navštíveného odkazu (implicitně světle fialová 800080), a ALINK barva aktuálního odkazu (implicitně červená FF0000). U ALINK jde o barvu odkazu v okamžiku, kdy na něj uživatel klepne myší.

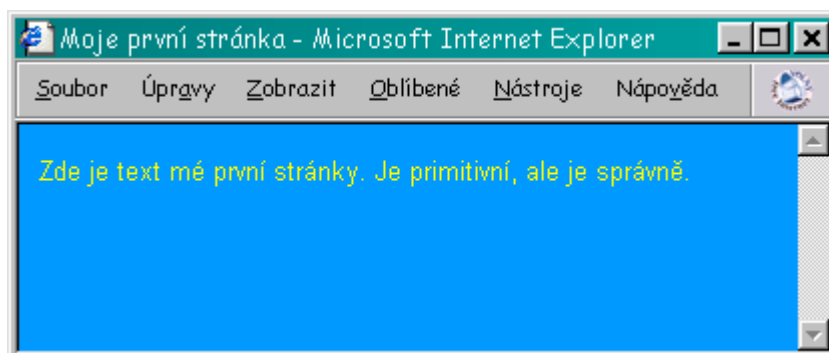
Vybrané barvy jsou pojmenované, tzn. lze použít jejich pojmenování namísto hexadecimálního kódu.

Pojmenování	Jméno barvy	Barva	Kód
Black	černá		000000
Aqua	světle modrá		00FFFF
Blue	modrá		0000FF
Fuchsia	purpurová		FF00FF
Gray	šedá		808080
Green	zelená		008000
Lime	světle zelená		00FF00
Navy	tmavě modrá		000080
Olive	olivová		808000
Purple	fialová		800080
Maroon	tmavě červená		800000
Red	červená		FF0000
Silver	stříbrná		C0C0C0
Teal	modrozelená		008080
White	bílá		FFFFFF
Yellow	žlutá		FFFF00

Tabulka 7.1: Pojmenované barvy jazyka HTML.

Příklad použití parametrů určujících barvy:

```
<BODY BACKGROUND="#0099FF">
<BODY TEXT="Yellow">
```



Obrázek 7.3: Příklad zobrazení základní stránky s jednolitým pozadím barvy 0099FF a textem barvy FFFF00.

Vzdálenosti okrajů stránky

Okraje stránky lze nastavit pomocí parametrů `LEFTMARGIN` a `TOPMARGIN`. Parametry udávají, kolik pixelů od levého či horního okraje začíná zobrazení obsahu stránky. Prázdný prostor je vyplněn obrázkem nebo barvou pozadí.

```
<BODY LEFTMARGIN="25">
```

7.2.2 Formátování textu

Formátování textu v HTML stránkách je dvojího druhu:

- ▶ logické (relativní) - formátování podle významu textu,
- ▶ fyzické (absolutní) - specifické formátování.

Logické formátování umožňuje oddělit obsahovou stránku od vzhledové, zaručuje vzhledovou konzistenci dokumentu a je flexibilní. Fyzické formátování zajišťuje specifické zobrazení textu bez ohledu na nastavení prohlížeče.

K logickému formátování se například řadí značky

- ▶ `<Hn>` - nadpisy,
- ▶ `<DFN>` - definiční text,
- ▶ `<CODE>` - programový kód.

K fyzickému formátování patří například značky

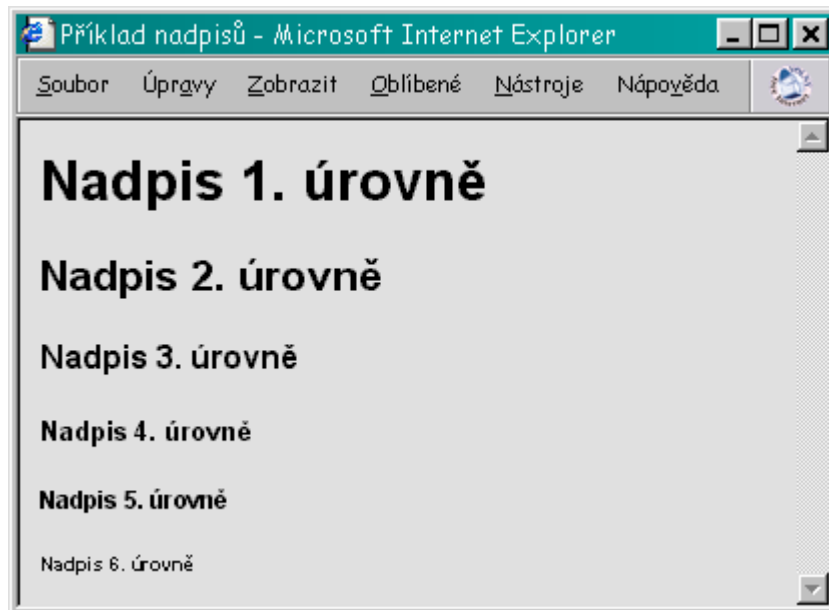
- ▶ `` - tučný text,
- ▶ `<I>` - kurzíva.

Formátovací značky definované v HTML jsou: `TT`, `I`, `B`, `U`, `STRIKE`, `BIG`, `SMALL`, `SUP`, `SUB`, `STRONG`, `DFN`, `CODE`, `SAMP`, `KBD`, `VAR`, `CITE`, `FONT`.

Nadpisy

Nadpisy jsou v běžném textu nejčastěji využívaným nástrojem formátování textu. Jde o značky ve tvaru `<Hn>`, kde $n=1..6$ označuje úroveň nadpisu. Syntaxe nadpisových značek je zřejmá z příkladu:

```
<H1>Nadpis 1. úrovně</H1>
<H2>Nadpis 2. úrovně</H1>
...
```



Obrázek 7.4: Příklad zobrazení stránky se všemi úrovněmi nadpisů.

Odstavce

Značka odstavce `<P>` (z angl. paragraph) je prvním nástrojem pro oddělení částí textu (jak již bylo řečeno, HTML ignoruje ENTER, duplicitní mezery a tabelátory).

Obsah odstavce může být zarovnán, a to čtyřmi způsoby. Syntaxe značky je pak

```
<P ALIGN="zarovnání">
```

kde `zarovnání` je

- `LEFT` - zarovná text doleva,
- `RIGHT` - zarovná text doprava,
- `CENTER` - text vycentruje,
- `JUSTIFY` - zarovná text do bloku.

Šíře odstavce je daná aktuální velikostí okna prohlížeče. Například text

```
<P ALIGN="CENTER">
```

Okraje stránky lze nastavit pomocí parametrů `LEFTMARGIN` a `TOPMARGIN`. Parametry udávají, kolik pixelů od levého či horního okraje začíná zobrazení obsahu stránky.

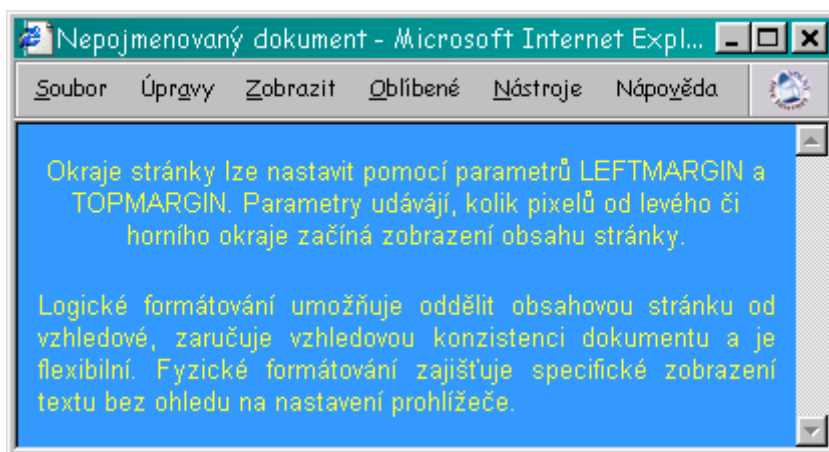
```
</P>
```

```
<P ALIGN="JUSTIFY">
```

Logické formátování umožňuje oddělit obsahovou stránku od vzhledové, zaručuje vzhledovou konzistenci dokumentu a je flexibilní. Fyzické formátování zajišťuje specifické zobrazení textu bez ohledu na nastavení prohlížeče.

```
</P>
```

bude zobrazen tak, jak je vidět na **Obrázek 7.5**.



Obrázek 7.5: Příklad použití odstavců se různým zarovnáním.

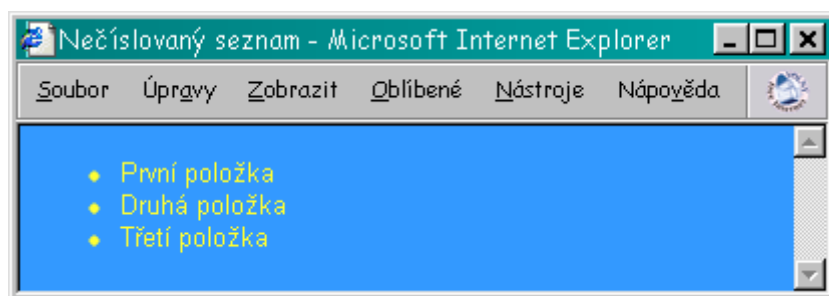
Zalomení řádků

Je-li nutné v odstavci odskočit na nový řádek, lze použít nepárové značky `
` (z angl. break).

Seznamy

Nejjednodušší seznam definovaný v jazyce HTML je **nečíslovaný seznam** položek pomocí značky `` (z angl. unnumbered list). Celý seznam je vymezen značkami ``, jednotlivé položky pak značkami `` (z angl. list item). Například:

```
<UL>
  <LI>První položka</LI>
  <LI>Druhá položka</LI>
  <LI>Třetí položka</LI>
</UL>
```



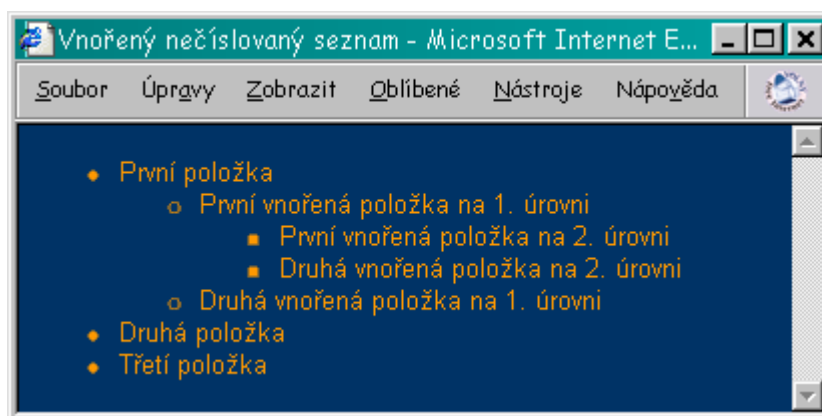
Obrázek 7.6: Příklad nečíslovaného seznamu.

Položku seznamu lze též automaticky číslovat. Zápis **číslovaného seznamu** vypadá totožně, pouze značka `` je zaměněna značkou ``.

Oba typy seznamů lze použít i ve vnořené podobě. Například nečíslovaný vnořený seznam:

```
<UL>
  <LI>První položka</LI>
  <UL>
    <LI>První vnořená položka na 1. úrovni</LI>
    <UL>
      <LI>První vnořená položka na 2. úrovni</LI>
      <LI>Druhá vnořená položka na 2. úrovni</LI>
    </UL>
    <LI>Druhá vnořená položka na 1. úrovni</LI>
  </UL>
  <LI>Druhá položka</LI>
```

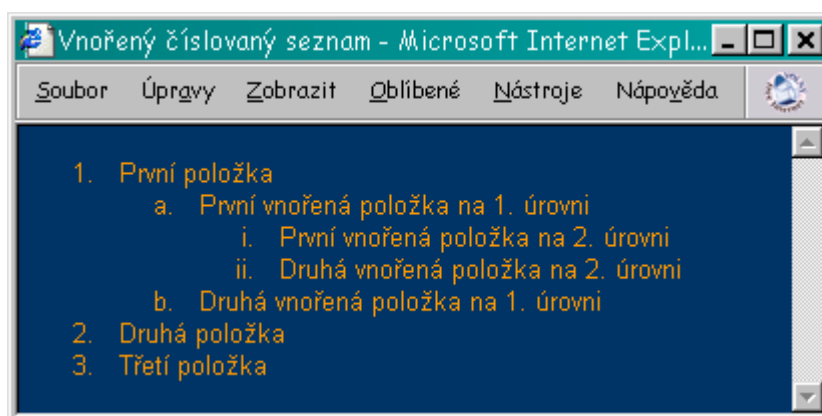
```
<LI>Třetí položka</LI>
</UL>
```



Obrázek 7.7: Příklad vnořeného nečíslovaného seznamu.

A dále číselovaný vnořený seznam:

```
<OL>
<LI>První položka</LI>
<OL TYPE="a">
<LI>První vnořená položka na 1. úrovni</LI>
<OL TYPE="i">
<LI>První vnořená položka na 2. úrovni</LI>
<LI>Druhá vnořená položka na 2. úrovni</LI>
</OL>
<LI>Druhá vnořená položka na 1. úrovni</LI>
</OL>
<LI>Druhá položka</LI>
<LI>Třetí položka</LI>
</OL>
```



Obrázek 7.8: Příklad vnořeného číselovaného seznamu s různým typem číslování na různých úrovních.

Jak je vidět z příkladu, u číselovaného seznamu lze volit různé způsoby číslování. Syntaxe značky je pak

```
<OL TYPE="typ">
```

kde *typ* je

A - pro číslování typu A, B, C, ...

a - pro číslování typu a, b, c, ...

I - pro číslování typu I, II, III, ...

i - pro číslování typu i, ii, iii, ...

1 - pro číslování typu 1, 2, 3, ...

7.2.3 Odkazy

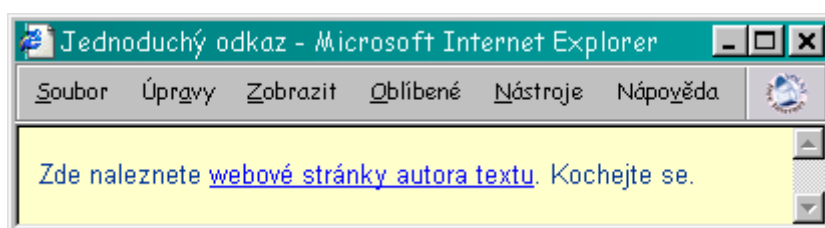
Odkaz je základním nástrojem tvorby hypertextových multimediálních stránek. K vytvoření odkazu se používá značka `<A>` (z angl. anchor), která má syntaxi

```
<A HREF="URL" TITLE="popisek" TARGET="cíl">Zobrazený text</A>
```

Použití všech parametrů není povinné (úplná syntaxe obsahuje 17 parametrů!). Nejjednodušší tvar pro odkaz na jinou HTML stránku je například

```
Zde naleznete <A  
HREF="http://www.feec.vutbr.cz/~provaznik/index.html">webové stránky  
autora textu</A>. Kochejte se.
```

Prohlížeč zobrazí text například takto



Obrázek 7.9: Příklad odkazu na externí stránku.

Význam ostatních parametrů je popsán v následujícím textu.

TITLE - uvádí alternativní název odkazu, který se objeví při podržení kurzoru myši nad odkazem.

TARGET - označuje cíl, kam bude odkazovaná stránka načtena. Cíl bývá využíván především u rámců. Vyhrazené hodnoty parametru jsou tyto:

- ▶ `_self` - odkazovaná stránka je načtena do rámce, ze kterého byl odkaz vyvolán.
- ▶ `_parent` - odkazovaná stránka je načtena do nadřazeného okna.
- ▶ `_blank` - odkazovaná stránka je načtena do nového prázdného okna.
- ▶ `jméno_okna` - odkazovaná stránka je načtena do okna se jménem `jméno_okna`.

7.2.4 Obrázky

Pro umísťování obrázků na webovou stránku se používá značka `` (z angl. image). Základní syntaxe značky je

```
<IMG SRC="URL">
```

Úplná syntaxe je pak

```
<IMG SRC="URL" ALIGN="zarovnání" ALT="text" BORDER="pixelů"  
HEIGHT="pixelů" WIDTH="pixelů" HSPACE="pixelů" VSPACE="pixelů"  
DYNsrc="URL" LOOP="n" START="fileopen"|"mouseover">
```

Parametr `SRC` určuje zdroj, ve kterém je umístěn obrázek. Například

```
<IMG SRC="lokální_obrazek.gif">  
<IMG SRC="cesta/jiny_obrazek.jpg">  
<IMG SRC="http://www.server.org/dalsi_obrazek.gif">
```

Parametr `ALIGN` může nabývat těchto hodnot

LEFT - zarovnání obrázku vlevo, text obtéká bezprostředně vpravo,

RIGHT - zarovnání obrázku vpravo, text obtéká bezprostředně vlevo,

ABSBOTTOM - zarovnání spodního okraje obrázku se spodním krajem písmen sahajících pod základnu textového řádku,

ABSMIDDLE - zarovnání středu obrázku se středem textového řádku,

BASELINE - zarovnání spodního okraje obrázku se základnou textového řádku,

BOTTOM - stejné jako BASELINE,

MIDDLE - zarovnání středu obrázku se základnou textového řádku,

TEXTTOP - zarovnání horního okraje obrázku s nejvyšším znakem v textovém řádku,

TOP - zarovnání horního okraje obrázku s nejvyšším elementem v textovém řádku.

Parametr **ALT** definuje alternativní text, který se zobrazí na místě obrázku v případě

- ▶ načítání obrázku,
- ▶ umístění kurzoru nad obrázek (a setrvání),
- ▶ vypnutí načítání obrázků (volba prohlížeče).

Parametr **BORDER** definuje velikost (šíři) rámečku obklopujícího obrázek. Je-li použito **BORDER="0"**, pak není rámeček zobrazen.

Parametry **HEIGHT** a **WIDTH** určují výšku, resp. šířku, obrázku. Pokud se hodnoty zadávané v pixelech liší od skutečných rozměrů obrázku, dochází k jeho převzorkování (a např. zmenšení). Tyto parametry není nutné používat, protože prohlížeč při načtení obrázku rozměry zjistí. Při tvorbě profesionálních stránek se naopak tyto parametry používají s hodnotami rovnými skutečným rozměrům obrázku. Při načítání je pak nejprve vyhrazeno obdélníkové místo obrázku, ten je pak do této plochy postupně zobrazován. Nedochází tak k "poskakování" obsahu stránky.

Parametry **HSPACE** a **VSPACE** určují velikost volného prostoru po stranách obrázku v horizontálním, resp. vertikálním, směru.

Parametr **DYN SRC** je náhradou za parametr **SRC** v případě, že na místo obrázku chceme zobrazit videoklip (například soubor .avi).

Parametr **LOOP** se používá pro definování počtu opakování přehrání videoklipu (kde **n** je počet).

Parametr **START** určuje, kdy bude videoklip spuštěn:

fileopen - při otevření stránky,

mouseover - při přejetí klipu kurzorem.

Níže je uveden příklad vloženého obrázku zarovnaného vpravo. Původně velký obrázek (814x1024 pixelů) je upraven definováním výšky 100 pixelů. Prohlížeč automaticky dopočítá druhý rozměr tak, aby byl zachován poměr velikostí stran. Obrázek má alternativní text, má definována prázdná místa okolo z důvodu zachování odstupu od textu. Má rámeček.

```
<BODY BGCOLOR="#FFCC66" TEXT="#003333">  
<IMG SRC="sara2.jpg" ALIGN="RIGHT" HEIGHT="100" BORDER="3" HSPACE="10"  
VSPACE="3" ALT="Malá Sára">
```

<P ALIGN=JUSTIFY>BIOSIGNAL 2000 international Euroconference is a medium-sized scientific meeting giving a forum to top experts, young scientists and experienced researchers involved in biomedical engineering and related fields. Four general areas are covered: 1. measurement and interpretation of physiologic signals, 2. medical imaging and image analysis, 3. physiologic modelling and simulation, 4. computer based clinical decision making.</P>

Zobrazení tabulky v prohlížeči je uvedeno na **Obrázek 7.10**.



Obrázek 7.10: Příklad vloženého obrázku.

7.2.5 Tabulky

Jazyk HTML poskytuje poměrně bohaté nástroje pro tvorbu tabulek. Tabulky jsou ve webových stránkách používány nejenom k přehlednému maticovému zobrazení informací, ale také k členění různě formátovaného textu (např. sloupcová sazba), rozmíst'ování obrázků, apod. Základním nástrojem je párová značka `TABLE`, ke které se váží další pomocné značky (zejména `TR`, `TD`, `TH`, a doplňkové `THEAD`, `TFOOT`, `COLGROUP`, `CAPTION`).

Vlastní tabulka

Značka je `TABLE` definována takto:

```
<TABLE ALIGN="zarovnání" WIDTH="n" HEIGHT="n" BACKGROUND="URL"
BGCOLOR="barva" BORDER="n" BORDERCOLOR="barva" BORDERCOLORDARK="barva"
BORDERCOLORLIGHT="barva" CELLPADDING="n" CELLSPACING="n" COLS="n"
FRAME="typ" RULES="typ" VALIGN="zarovnání">
```

Parametry jsou sice nepovinné, ale jejich použití umožní tabulku velmi atraktivně zformátovat. Jednotlivé parametry jsou vysvětleny níže.

ALIGN - horizontální zarovnání tabulky. Parametr nabývá hodnot `LEFT` (vlevo - implicitní), `CENTER` (na střed) a `RIGHT` (vpravo).

WIDTH - určuje šířku tabulky v pixelech (např. `WIDTH="100"`) nebo v procentech (`WIDTH="90%"`) šíře okna prohlížeče.

HEIGHT - určuje výšku tabulky v pixelech (např. `HEIGHT="100"`) nebo v procentech (`HEIGHT="90%"`) výšky okna prohlížeče.

BACKGROUND - vložení obrázku na pozadí tabulky (podobně jako vložení obrázku na pozadí celé stránky).

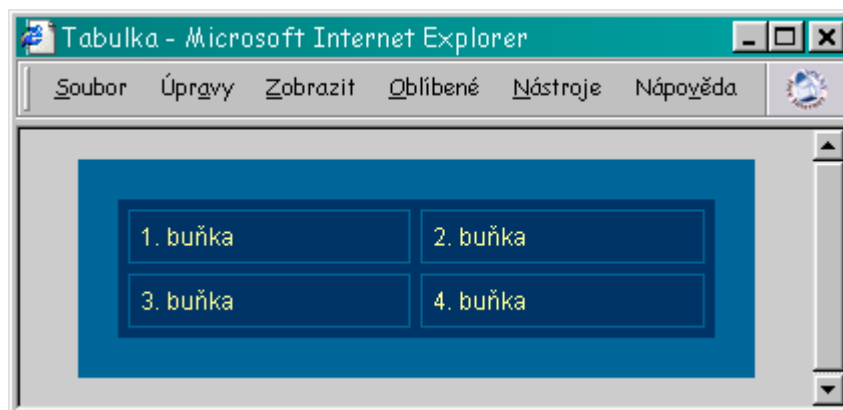
BGCOLOR - barva pozadí tabulky. Je-li tento parametr použit současně s parametrem **BACKGROUND**, má **BACKGROUND** přednost. Naopak, barva pozadí může být překryta barvou pozadí jednotlivých buněk.

BORDER - tloušťka rámečku tabulky v pixelech. Rámeček není implicitně jednolitý, ale napodobuje plastické stínování.

BORDERCOLOR - určuje jednotnou barvu rámečku včetně stínování. Příklad tohoto typu obarvení je

```
<TABLE ALIGN="CENTER" WIDTH="90%" BORDER="20" BGCOLOR="#003366"
BORDERCOLOR="#006699" CELLPADDING="5" CELLSPACING="5">
```

Zobrazení tabulky v prohlížeči je uvedeno na **Obrázek 7.11**.



Obrázek 7.11: Příklad tabulky s jednolitým rámečkem.

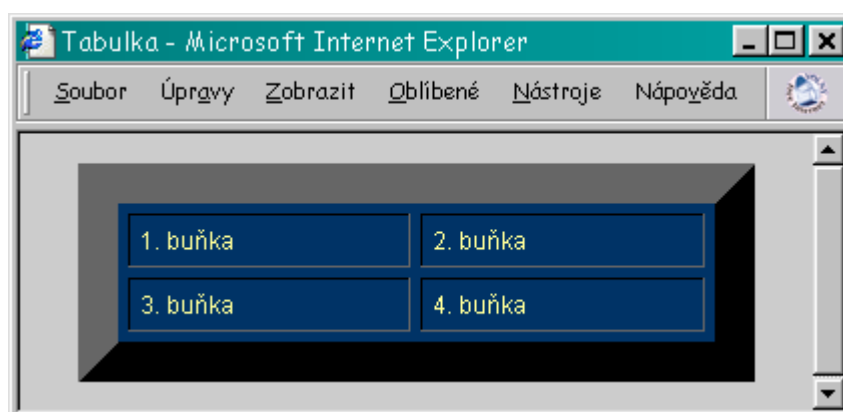
BORDERCOLORDARK - určuje barvu spodní a pravé strany rámečku.

BORDERCOLORLIGHT - určuje barvu horní a levé strany rámečku.

Příklad tohoto typu obarvení je

```
<TABLE ALIGN="CENTER" WIDTH="90%" BORDER="20" BGCOLOR="#003366"
BORDERCOLORDARK="#000000" BORDERCOLORLIGHT="#666666" CELLPADDING="5"
CELLSPACING="5">
```

Zobrazení tabulky v prohlížeči je uvedeno na **Obrázek 7.12**.



Obrázek 7.12: Příklad tabulky s plastickým rámečkem.

CELLPADDING - určuje vzdálenost mezi okraji buňky a jejím obsahem v pixelech. Implicitní hodnota je 1.

CELLSPACING - určuje vzdálenosti mezi buňkami v pixelech. Implicitní hodnota je 2.

COLS - udává počet sloupců tabulky. Parametr není povinný, protože počet sloupců vyplývá z definice jednotlivých buněk (viz níže). Uvedení parametru však urychlí prvotní vykreslení tabulky bez pozdějšího "poskakování" textu.

FRAME - určuje vzhled rámečku. Může nabývat těchto hodnot

- ▶ **VOID** - bez vnějšího rámečku,
- ▶ **ABOVE** - rámeček pouze nahoře,
- ▶ **BELOW** - rámeček pouze dole,
- ▶ **HSIDES** - rámeček pouze nahoře a dole,
- ▶ **VSIDES** - rámeček pouze vpravo a vlevo,
- ▶ **RHS** - rámeček pouze vpravo,
- ▶ **LHS** - rámeček pouze vlevo.

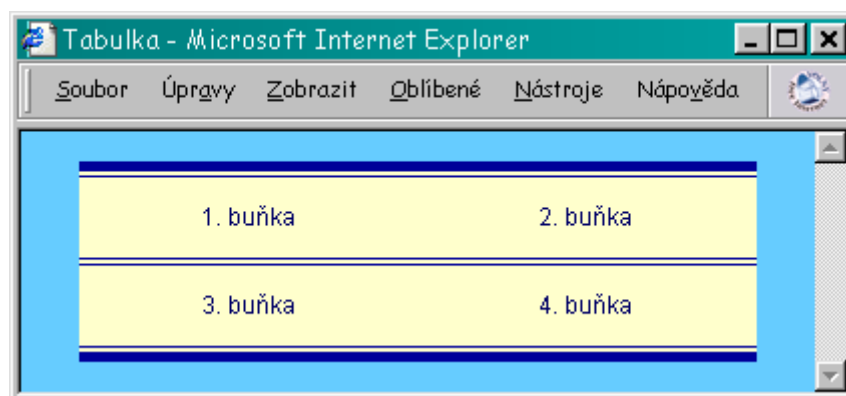
RULES - určuje vnitřní rámečky (linky oddělující jednotlivé řádky a sloupce). Může nabývat těchto hodnot

- ▶ **NONE** - bez vnitřních rámečků,
- ▶ **GROUPS** - vodorovné linky mezi hlavičkou, tělem a patičkou tabulky,
- ▶ **ROWS** - linky mezi všemi řádky tabulky,
- ▶ **COLS** - linky mezi všemi sloupci tabulky,
- ▶ **ALL** - všechny vnitřní linky.

Jako příklad formátované tabulky uveďme

```
<TABLE ALIGN="CENTER" WIDTH="90%" HEIGHT="100" BGCOLOR="#FFFFCC"
BORDER="5" BORDERCOLOR="#000099" CELLSPACING="2" CELLPADDING="2"
COLS="2" FRAME="HSIDES" RULES="ROWS">
```

Zobrazení tabulky v prohlížeči je uvedeno na **Obrázek 7.13**.



Obrázek 7.13: Příklad formátované tabulky.

Řádky tabulky

Řádky tabulky se vytvářejí párovou značkou **TR**. Počet řádků není v tabulce předem definován, je určen až počtem výskytů značky **TR**. Syntaxe **TR** je

```
<TR ALIGN="zarovnání" BGCOLOR="barva" BORDERCOLOR="barva"
BORDERCOLORDARK="barva" BORDERCOLORLIGHT="barva" VALIGN="zarovnání">
vlatní obsah řádku
</TR>
```

ALIGN - horizontální zarovnání textu v řádku. Parametr nabývá hodnot **LEFT** (vlevo - implicitní), **CENTER** (na střed) a **RIGHT** (vpravo).

BORDERCOLOR - určuje jednotnou barvu vnitřního rámečku včetně stínování.

`BORDERCOLORDARK` - určuje barvu spodní a pravé strany vnitřního rámečku.

`BORDERCOLORLIGHT` - určuje barvu horní a levé strany vnitřního rámečku.

`VALIGN` - určuje vertikální zarovnání textu v buňkách (může být přepsáno určením zarovnání v definicích buněk). Může nabývat těchto hodnot

- ▶ `TOP` - na vrchní okraj buňky,
- ▶ `MIDDLE` - na střed (implicitně),
- ▶ `BOTTOM` - na spodní okraj buňky.

Příklad formátování řádku tabulky je

```
<TABLE ALIGN="CENTER" WIDTH="90%" HEIGHT="100" BGCOLOR="#FFFFCC"
BORDER="5" BORDERCOLOR="#000099" CELSPACING="2" CELLPADDING="2"
COLS="2">
<TR ALIGN="LEFT" BGCOLOR="#FF6600" BORDERCOLOR="#66FFFF"
VALIGN="BOTTOM">
...
</TR>
```

Zobrazení tabulky v prohlížeči je uvedeno na **Obrázek 7.14**.



Obrázek 7.14: Příklad formátování řádku tabulky.

Buňky tabulky

Buňky tabulky jsou definovány párovou značkou `TD`. Tato značka má velké množství parametrů, uvedeme jen ty nejdůležitější. Syntaxe značky je

```
<TD ALIGN="zarovnání" WIDTH="n" BACKGROUND="URL" BORDERCOLOR="barva"
NOWRAP COLSPAN="n" ROWSPAN="n">
obsah buňky
</TD>
```

`ALIGN` - horizontální zarovnání tabulky. Parametr nabývá hodnot `LEFT` (vlevo - implicitní), `CENTER` (na střed) a `RIGHT` (vpravo).

`WIDTH` - určuje šířku buňky v pixelech (např. `WIDTH="100"`) nebo v procentech (`WIDTH="90%"`) šíře okna tabulky.

`HEIGHT` - určuje výšku buňky v pixelech (např. `HEIGHT="100"`) nebo v procentech (`HEIGHT="90%"`) výšky tabulky.

`BACKGROUND` - vložení obrázku na pozadí buňky (podobně jako vložení obrázku na pozadí celé stránky nebo tabulky).

`BGCOLOR` - barva pozadí buňky. Je-li tento parametr použit současně s parametrem `BACKGROUND`, má `BACKGROUND` přednost.

BORDER - tloušťka rámečku buňky v pixelech. Rámeček není implicitně jednolitý, ale napodobuje plastické stínování.

BORDERCOLOR - určuje jednotnou barvu rámečku buňky včetně stínování.

NOWRAP - zakázání zalamování textu uvnitř buňky.

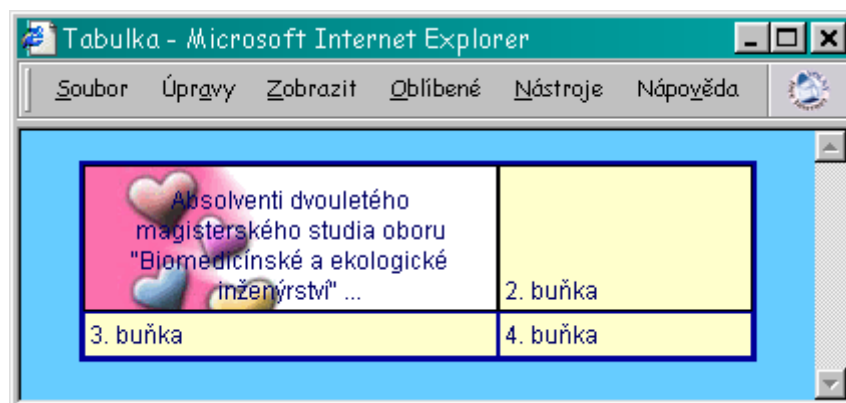
COLSPAN - spojí více buněk ležících v jednom řádku.

ROWSPAN - spojí více buněk ležících v jednom sloupci.

Příklad kompletního zápisu jednoduché tabulky v kódu HTML s formátováním jednotlivé buňky je

```
<TABLE ALIGN="CENTER" WIDTH="90%" HEIGHT="100" BGCOLOR="#FFFFCC"
BORDER="2" BORDERCOLOR="#000099" CELSPACING="0" CELLPADDING="2">
  <TR ALIGN="LEFT" BGCOLOR="#FF6600" BORDERCOLOR="#000000"
  VALIGN="BOTTOM">
    <TD ALIGN="CENTER" WIDTH="62%" BACKGROUND="fig.gif"
    BORDERCOLOR="#000000">Absolventi dvouletého magisterského studia
    oboru "Biomedicínské a ekologické inženýrství" ...</TD>
    <TD WIDTH="38%" BGCOLOR="#FFFFCC">2. buňka</TD>
  </TR>
  <TR ALIGN="LEFT">
    <TD WIDTH="62%">3. buňka</TD>
    <TD WIDTH="38%">4. buňka</TD>
  </TR>
</TABLE>
```

Zobrazení tabulky v prohlížeči je uvedeno na **Obrázek 7.15**.




Obrázek 7.15: Příklad formátování buňky tabulky.

Slučování buněk pomocí parametrů **COLSPAN** a **ROWSPAN** není z popisu zcela zřejmé. Následující příklad ukazuje, jak lze obojí slučování použít.

```
<TABLE ALIGN="CENTER" WIDTH="90%" BORDER="2" BORDERCOLOR="BLACK"
CELLSPACING="0" CELLPADDING="2">
  <TR>
    <TD ALIGN="CENTER" COLSPAN="2" BACKGROUND="fig.gif"
    BORDERCOLOR="BLACK">Studenti dvouletého magisterského studia oboru
    "Biomedicínské a ekologické inženýrství"</TD>
  </TR>
  <TR ALIGN="CENTER" BGCOLOR="#FFFFCC">
    <TD>1. ročník</TD>
    <TD ROWSPAN="2">SEZNAM JMEN</TD>
  </TR>
  <TR ALIGN="CENTER" BGCOLOR="#FFFFCC">
    <TD>2. ročník</TD>
  </TR>
```

</TABLE>

Příklad je dokumentován na **Obrázek 7.16**.

 Studenti dvouletého magisterského studia oboru "Biomedicínské a ekologické inženýrství"	
1. ročník	seznam jmen
2. ročník	


Obrázek 7.16: Příklad slučování buněk tabulky ve vertikálním i horizontálním směru.

Hlavička tabulky

Hlavička tabulky je definována párovou značkou **TH**. Tato značka je modifikací značky **TD** (má tytéž parametry). Jde vlastně o buňky zvláštního (většinou prvního) řádku tabulky, s implicitně nastaveným tučným textem a zarovnáním textu na střed. Definice parametrů je již uvedena u značky **TD**. Použití hlavičky je vidět z následujícího příkladu.

```
<TABLE ALIGN="CENTER" WIDTH="90%" BORDER="2" BORDERCOLOR="BLACK"
CELLSPACING="0" CELLPADDING="2">
  <TR>
    <TD ALIGN="CENTER" COLSPAN="2" BACKGROUND="fig.gif"
      BORDERCOLOR="BLACK">Studenti dvouletého magisterského studia oboru
      "Biomedicínské a ekologické inženýrství"</TD>
  </TR>
  <TR ALIGN="CENTER" BGCOLOR="#FFFFCC">
    <TH>ročník studia</TH>
    <TH>jména studentů</TH>
  </TR>
  <TR ALIGN="CENTER" BGCOLOR="#FFFFCC">
    <TD>1.</TD>
    <TD ROWSPAN="2">Karel Vopršálek<BR>Josef Vopička</TD>
  </TR>
  <TR ALIGN="CENTER" BGCOLOR="#FFFFCC">
    <TD>2.</TD>
  </TR>
</TABLE>
```

Příklad je dokumentován na **Obrázek 7.17**.

 Studenti dvouletého magisterského studia oboru "Biomedicínské a ekologické inženýrství"	
ročník studia	jména studentů
1.	Karel Vopršálek
2.	Josef Vopička

Obrázek 7.17: Příklad použití hlavičky tabulky.

8 Dodatky

8.1 Seznam zkratek

ALU	Arithmetic-Logic Unit
ANSI	American National Standard Institute
ASCII	The American Standard Code for Information Interchange
ATA	AT Attachment
ATAPI	AT Attachment Packet Interface
BCD	Binary Coded Decimal
BGP	Border Gateway Protocol
BER	Bit Error Ratio
CGI	Common Gateway Interface
CHS	Cylinder Head Sector
CPU	Central Processor Unit
CRC	Cyclic Redundance Checking
CSS	Cascading Style Sheets
DMA	Direct Memory Access
DNS	Domain Name System
DTD	Document Type Definition
EIDE	Enhanced Integrated Drive Electronics
FDD	Floppy Disk Drive
FTP	File Transfer Protocol
FSB	Front Side Bus
HDD	Hard Disk Drive
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDE	Integrated Drive Electronics
IMAP	Internet Mail Access Protocol
IP	Internet Protocol
IrDA	Infrared Data Association
ISO	International Organisation for Standardization

LBA	Logical Block Address
LLC	Logic Link Control
LSB	Least Significant Bit
MAC	Medium Access Control
MIME	Multi-Purpose Internet Mail Extension
MSB	Most Significant Bit
NaN	Not a Number
NRZI	Non Return Zero Invert
OSI	Open Systems Interconnection
OSPF	Open the Shortest Path First
POP	Post Office Protocol
PPP	Point-to-Point Protocol
RAM	Random Access Memory
RIP	Routing Information Protocol
ROM	Read Only Memory
RTP	Real Time Protocol
SCSI	Small Computer System Interface
S.M.A.R.T.	Self-Monitoring, Analysis, and Reporting Technology
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SPI	SCSI Parallel Interface
SPP	Standard Parallel Port
SSH	Secure Shell
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal System Bus
XML	Extensible Markup Language

Seznam použité literatury

- [1] Jirovský V.: Vademecum správce sítě. Grada 2001.
- [2] Dostálek L.: Velký průvodce protokoly TCP/IP. Bezpečnost. Computer Press 2001.
- [3] Spainhour S., Eckstein R.: Webmaster v kostce. Computer Press 1999.
- [4] Garfinkel S., Spafford G.: Bezpečnost v UNIXu a Internetu v praxi. Computer Press 1998.
- [5] Grand M.: JAVA. Referenční příručka jazyka. Computer Press 1998.
- [6] Minasi M.: Velký průvodce hardwarem PC. Grada 1998.
- [7] Brodský J., Skočovský L.: Operační systém UNIX a jazyk C. SNTL 1989.
- [8] Hlavenka J., Sedlář R., Kučera M., Schneider Z., Mach J.: Vytváříme WWW stránky a spravujeme moderní web site. Computer Press 2001.
- [9] Castro E.: HTML 4 pro World Wide Web. SoftPress 2001.
- [10] Pužmanová R.: Moderní komunikační sítě od A do Z. Computer Press 1998.
- [11] Naik D. C.: Internet standardy a protokoly. Computer Press 1999.
- [12] IEEE Computer Society: IEEE Standard for Binary Floating-Point Arithmetic, IEEE Standard 754-1985.