

Přednáška 1

Úvod. Historie OS Unix. Architektura OS Unix. Interpret příkazů – SHELL. Zpracování příkazové řádky. Speciální znaky. Zkratky příkazů.

- **Klasifikace**

90 – 100 bodů	A (výborně)
80 – 89 bodů	B (velmi dobře)
70 – 79 bodů	C (dobře)
60 – 69 bodů	D (uspokojivě)
50 – 59 bodů	E (dostatečně)
< 50 bodů	F (nedostatečně)

- **OS Unix**
 1. Historie. Architektura. Zpracování příkazové řádky.
 2. Systém souborů. Nástroje pro práci se systémem souborů.
 3. Filtry.
 4. Regulární výrazy. Filtry grep, sed a awk.
 5. Identita uživatelů, procesů a souborů. Přístupová práva.
 6. Procesy a vlákna. Signály.
 7. Proměnné. Návratový kód. Příkazy pro větvení výpočtu. Cykly. Parametry skriptu. Vstup a výstup.
 8. Sítové rozhraní.
 9. Grafické rozhraní. Secure shell.

- **MS Windows XP**
 10. Historie. Architektura. Grafické a znakové rozhraní.
 11. Systém souborů. Nástroje pro práci se systémem souborů.
 12. Identita uživatelů, procesů a souborů. Přístupová práva.
 13. Síťové rozhraní. Bezpečnost.

- **OS Unix**
 - [1] Rozumíme Unixu, Jon Lasser, Computer Press, ISBN 80-7226-706-X, 2002.
 - [2] <http://chemi.muni.cz/~n19n/vyuka/>
 - [3] http://8help.osu.edu/wks/unix_course/
 - [4] <http://www.abclinuxu.cz/>
 - [5] Manuálové stránky Unixu.
- **MS Windows XP**
 - [1] Mistrovství v MS Windows XP, Ed Bott, Carl Siechert, Computer Press, ISBN 80-7226-693-4, 2002.
 - [2] MS Windows Resource Kit Site: <http://www.microsoft.com>

Historie OS Unix

konec 60. let	AT&T vyvíjí MULTICS
1969	AT&T Bell Labs - začátek OS Unix
začátek 70.let	AT&T vývoj OS Unix
kolem 1975	University of California at Berkley - UNIX (BSD)
začátek 80. let	Komerční zájem o OS Unix, DARPA
konec 80.let	Návrh standardů (POSIX, XPG, SVID), SVR4 UNIX
1991	Linus B. Torvalds vytváří jádro OS Linux

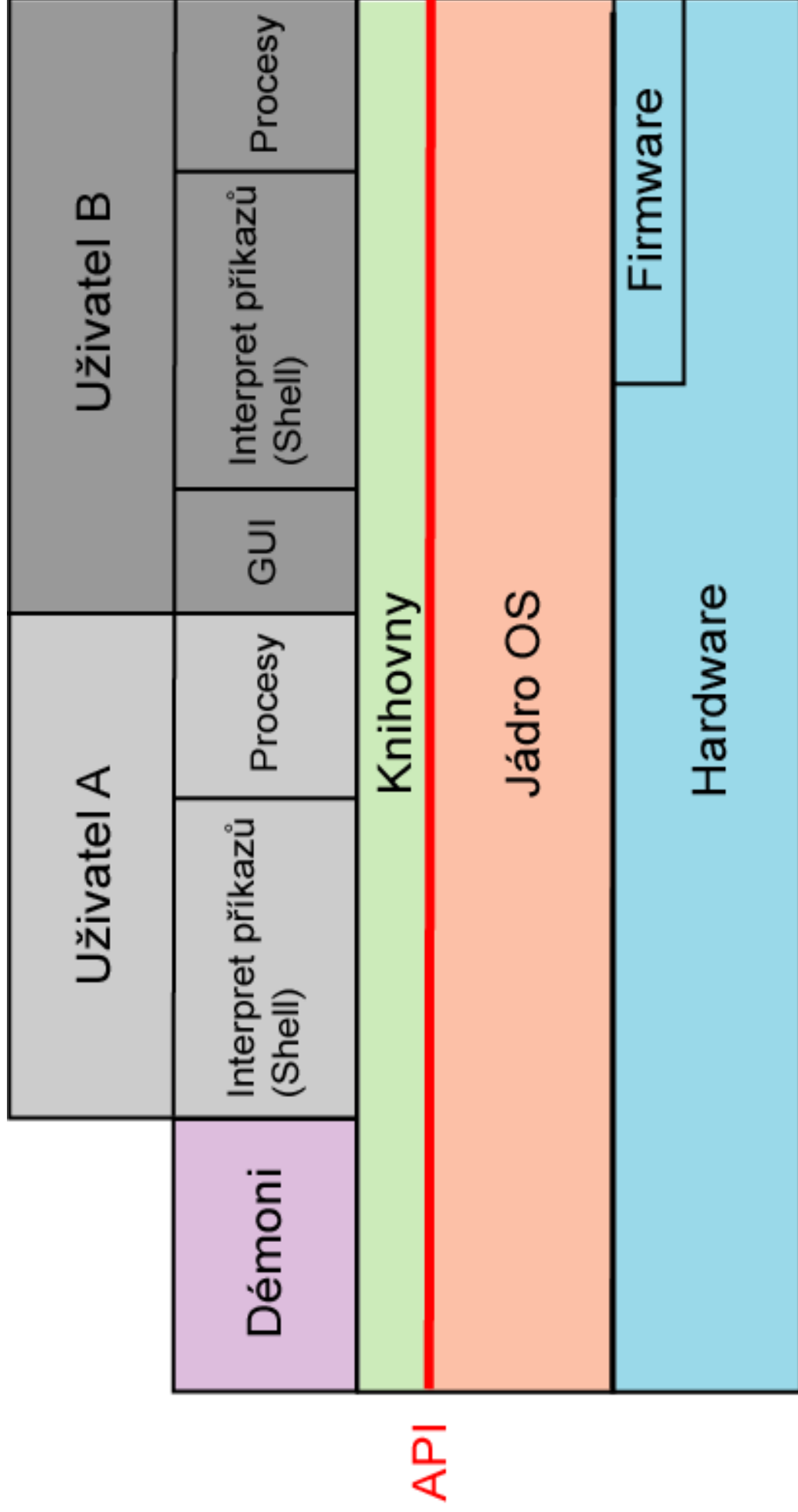
- **OS Linux**
 - Red Hat Enterprise Linux
 - Fedora Core
 - Mandriva Linux (Mandrake Linux)
 - Debian GNU/Linux
 - Ubuntu
 - Gentoo Linux
- **OS Solaris**
 - Solaris 10
 - Open Solaris

- Bootovatelné CD/DVD obsahující příslušnou distribuci OS.
- OS může být spuštěn bez nutnosti instalace na disk.
- Seznam OS nabízejících LiveCD viz. Wikipedia.
 - OS Linux (Ubuntu, Puppy Linux, Knoppix,...)
 - OS BSD (FreeSBIE, NetBSD,...)
 - OS Solaris (BeleniX,...)
 - OS Apple Macintosh (BootCD,...)

Jak se připravovat na cvičení

- Vzdálené připojení na školní servery přes SSH
- Live CD/DVD s OS Unix
- VMWare + OS Unix
- OS Unix (Linux, Solaris, ...)

Architektura OS Unix



Doplnit části jádra, kernel a user mod,...

- **Víceúlohový (multitasking, time-sharing)**
 - běh více úloh (procesů) ve sdílení času
 - ochrana paměti, plánování procesů
- **Vícevláknový (multithreading)**
 - proces se může skládat z několika současně běžících úloh (vláken)
 - přechod od plánování procesů na plánování vláken (thread)
- **Víceuživatelský (multi-user)**
 - možnost současné práce více uživatelů
 - identifikace a vzájemná ochrana uživatelů
- **Podpora multiprocessorových systémů (SMP)**
 - použití vláken v jádře a jejich plánování na různých CPU
- **Unifikované prostředí**
 - přenositelnost mezi platformami (90% jádra v jazyce C)

- **Interaktivní přístup s možností vytváření dávek příkazů**
 - shell jako rozhraní uživatele a interpret řídicího jazyka
- **Přesměrování a řetězení vstupu a výstupu příkazů**
 - vše je soubor (i periferie, nyní i procesy)
- **Hierarchický systém souborů**
 - odpadá potřeba rezervovat místo pro vytvářené soubory
- **Podpora práce v síti**
 - nejprve komunikace mezi dvěma počítači (uucp, mail)
 - později protokoly TCP/IP, NFS, internet a další
- **Grafické prostředí**
 - virtuální grafický terminál X-Window
 - různá grafická uživatelská rozhraní (GUI) nad X (CDE, GNOME, KDE,...)

- Rozhraní mezi uživatelem a jádrem OS
- **Nastavení prostředí**
 - v interpretu můžeme definovat proměnné, které řídí chování vašeho unixového sezení
- **Interaktivní režim**
 - analýza příkazové řádky (nalezení příkazu, substituce,...)
 - spuštění příkazu (binárního programu nebo skriptu)
- **Dávkový režim**
 - interpret provádí příkazy uložené ve skriptu (soubor)
 - skript = příkazy Unixu + řídicí struktury (např. podmíněné příkazy, cykly, ...)

- **Skupina Bourne shellů**

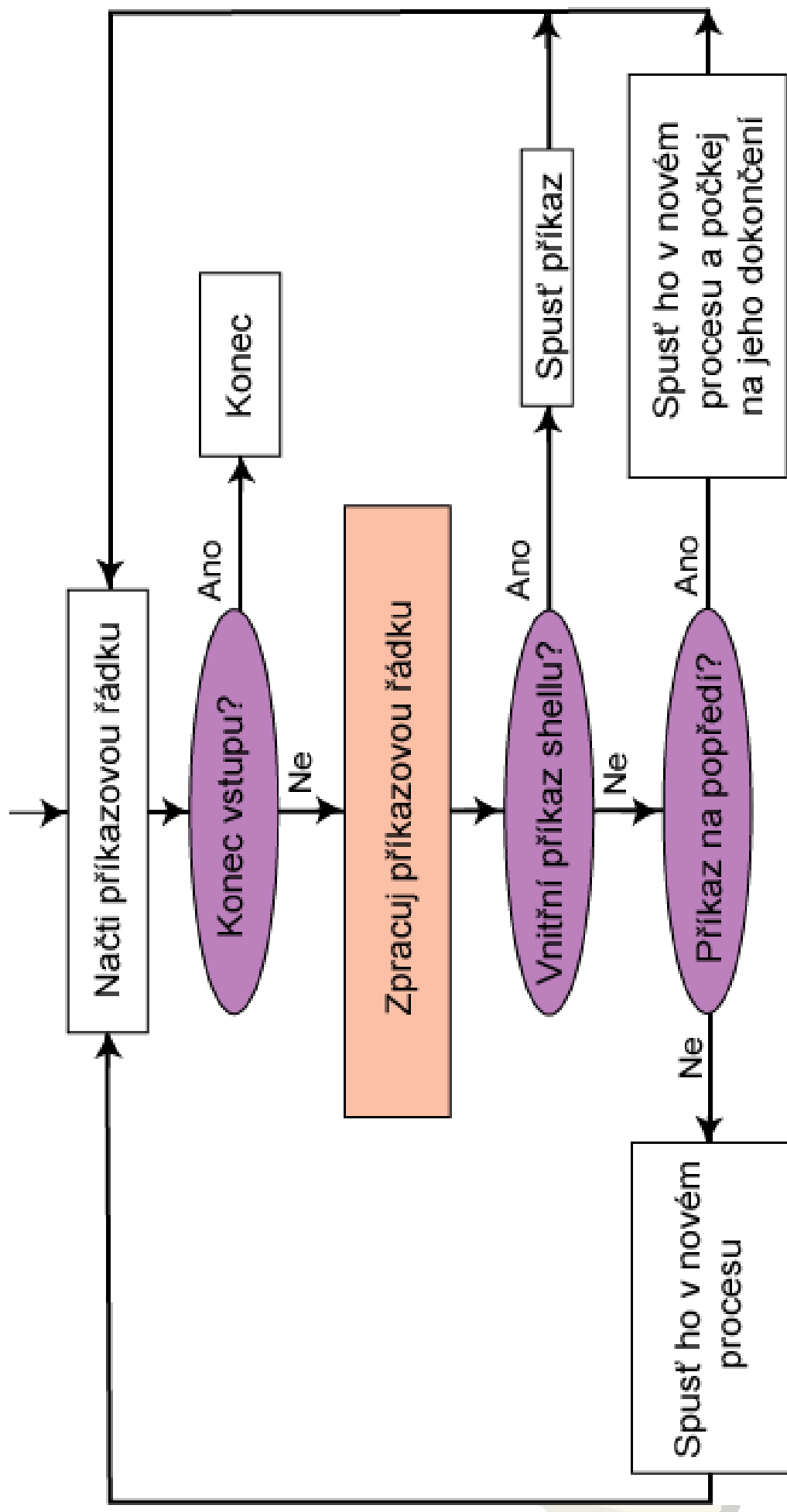
Jméno	Program	Vlastnosti
Bourne shell	/bin/sh	základní
Korn shell	/bin/ksh	historie příkazů, editace příkazové řádky, práce s úlohami, přejmenování příkazů, syntaxe skriptů jako u sh + rozšíření
Bourne again shell	/bin/bash	podobné jako ksh, lepší uživatelské rozhraní, syntaxe skriptů jako u sh + rozšíření
POSIX shell	/bin/sh	drobné rozšíření oproti ksh

- **Skupina C shellů**

Jméno	Program	Vlastnosti
C shell	/bin /csh	uživatelské rozhraní podobné jako ksh, syntaxe skriptů podobná jazyku C
Toronto C shell	/bin/tcsh	podobné jako csh, lepší uživatelské rozhraní

- Podrobné informace o konkrétním interpretu lze najít v unixovém manuálu (např. **man bash**).
- V tomto předmětu se budeme věnovat skupině Bourne shellů.

Zpracování příkazu interpretem



- **Přřazení hodnoty proměnné**

<výzva> <jméno proměnné>=<hodnota>

<výzva>

je vysána interpretem před interaktivním čtením příkazu (nikoliv u dávk), může být změněna nastavením proměnné PS1

<jméno proměnné>

jméno proměnné je identifikátor

mezi jménem proměnné, znakem = a hodnotou nesmí být mezery
příkaz přiřadí příslušné proměnné danou hodnotu

<hodnota>

standardně textový řetězec

pokud obsahuje mezery, je třeba uzavřít do uvozovek

- **Formát jednoduchého příkazu**

<výzva> <jméno příkazu> <argumenty>

<jméno příkazu>

může být pouze jméno nebo cesta (relativní/absolutní) a jméno

<argumenty>

obvykle nejprve prepínače (uvozené znakem -), pak jména souborů

prepínače obvykle jednoznakové

někdy se dají sdružovat, jindy se musí psát zvlášť

v programu dostupné přes proměnné \$1, \$2, ...

1. Detekce znaků rušících speciální význam znaků

	ruší speciální význam následujícího znaku
' ...	všechny znaky uzavřené mezi apostrofy ztrácejí speciální význam (kromě apostrofu)
" ...	mezi uvozovkami ztrácí speciální význam všechny znaky kromě: ruší význam následujícího znaku ' příkaz ' náhrada příkazů \$ náhrada obsahu proměnné

– Shell tyto znaky odstraní při interpretaci řádky.

2. **Odstranění komentářů (#)**
3. **Postupné rozdělení příkazové řádky na jednoduché příkazy**

- jednoduchý příkaz (simple command) =
 - posloupnost přiřazení hodnot proměnným oddělených mezerami nebo tabulátory
 - jméno příkazu následované jednotlivými argumenty
- roura (pipeline) = posloupnost jednoho nebo více příkazů oddělených operátorem |
- seznam příkazů (list) = posloupnost jedné nebo více rour oddělených operátory ; && || a ukončených operátory ; & <newline>
- složený příkaz (compound command) =
(list) { list ; } ((výraz)) [[výraz]] for while until if case

Zpracování příkazové řádky

- Každý příkaz vrací **návratový kód** (0 = úspěšné provedení, 1, ..., 255 = chyba)

příkaz &	asynchronní provádění příkazu shell nečeká na jeho dokončení, příkaz se provádí „na pozadí“)
příkaz1 ; příkaz2	sekvenční provádění příkazů, nejdříve se provedou příkazy před a pak příkazy za středníkem
(příkaz1 ; příkaz2)	podshell, příkazy jsou spuštěny v nové instanci shellu

Zpracování příkazové řádky

příkaz1 příkaz2	roura, příkazy se startují zleva a běží paralelně, standardní výstup předchozího je standardním vstupem následujícího příkazu, návratový kód roury je návratovým kódem posledního příkazu
příkaz1 && příkaz2	sekvenční provádění příkazů, příkaz „za“ se provede pouze tehdy, vrátí-li příkaz „před“ nulový návratový kód (skončí bez chyby)
příkaz1 příkaz2	sekvenční provádění příkazů, příkaz „za“ se provede pouze tehdy, vrátí-li příkaz „před“ nenulový návratový kód (skončí s chybou)

4. Náhrada

- aliasů (zkratok příkazů)
- znaku tilda (~)
- příkazů

<code>` příkaz `</code>	příkaz mezi opačnými apostrofy je proveden a nahrazen (včetně těchto apostrofů) svým std. výstupem
<code>\$(příkaz)</code>	to samé, pouze jiná syntaxe (mimo sh)

- aritmetických výrazů `$((výraz))`
- parametrů a proměnných (`$1`, `$HOME`,...)

5. Rozdělení na slova

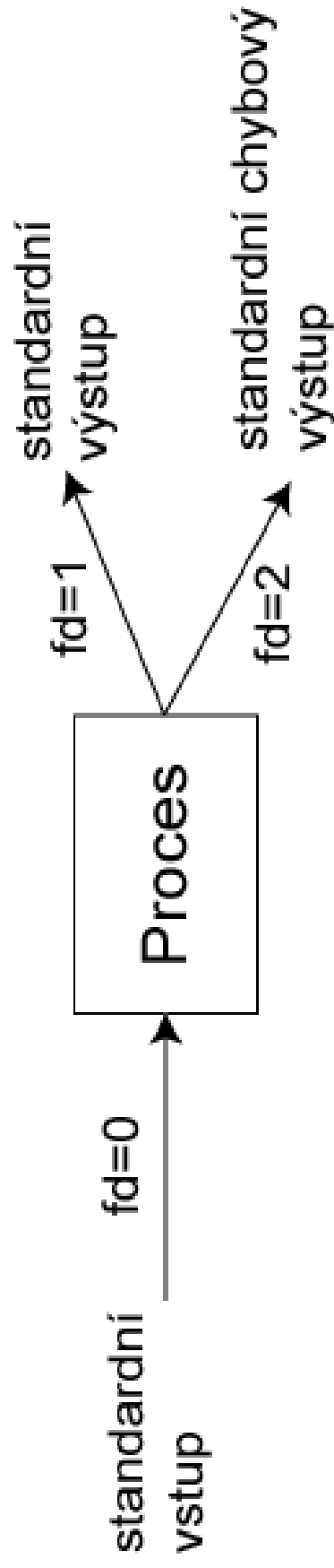
<code>newline mezera TAB</code>	oddělovače slov (Ize změnit pomocí proměnné IFS)
---------------------------------	--

6. Náhrada jmen souborů

*	odpovídá libovolnému řetězci kromě tečky na začátku a / kdekoliv
?	odpovídá jednomu libovolnému znaku kromě tečky na začátku a / kdekoliv
[abc] [a-z]	odpovídá jednomu znaku z uvedených znaků resp. z uvedeného intervalu
[!abc] [!a-z]	odpovídá jednomu znaku mimo uvedených znaků
~	odpovídá domovskému adresáři (kromě sh)
~uživatel	odpovídá domovskému adresáři daného uživatele (kromě sh)

- Znaky . na začátku jména a / se musí explicitně uvádět.
- Pokud ničemu neodpovídá, pak text zůstává nezměněn.
- Nahrazování *, ? a [] lze potlačit příkazem **set -f** a opět povolit příkazem **set +f** (nedoporučuje se).

7. Přesměrování vstupu/výstupu



- Procesy přistupují k souborům pomocí tzv. deskriptorů souborů (0,1,2,...).
- **Každý proces má při spuštění standardně otevřeny tyto deskriptory:**
 - 0 – standardní vstup
 - 1 – standardní výstup
 - 2 – standardní chybový výstup
- Nový proces standardně dědí deskriptory souborů od svého rodiče.
- Pomocí speciálních znaků lze v shellu předefinovat jednotlivé deskriptory.

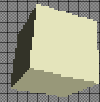
Zpracování příkazové řádky

příkaz < soubor	soubor bude otevřen a nastaven jako std. vstup příkazu
příkaz > soubor	soubor bude otevřen jako std. výstup z příkazu pokud soubor neexistuje, bude otevřen pokud existuje, bude přepsán (lze potlačit nastavením parametru noclobber v shellu – mimo sh)
příkaz >> soubor	soubor bude otevřen jako std. výstup z příkazu pokud soubor neexistuje, bude otevřen pokud existuje, bude výstup připojen na konec
příkaz << řetězec	shell čte vstup až do řádky začínající daným řetězcem (tzv. „here-document“) načtený text se stane std. vstupem příkazu

Zpracování příkazové řádky

příkaz <code>2> soubor</code>	soubor bude otevřen jako std. chybový výstup z příkazu pokud soubor neexistuje, bude otevřen pokud existuje, bude přepsán
příkaz <code>>&n</code>	std. výstup bude zapsán do souboru určeného deskriptorem n
příkaz <code>m>&n</code>	deskriptor n se přiřadí do deskriptoru m výstup příkazu do souboru určeného deskriptorem m se přesměruje do souboru určeného deskriptorem n
příkaz <code>> soubor 2>&1</code>	Std. výstup i std. chybový výstup bude zapsán do souboru.

- Při vícenásobném přesměrování se přesměrování vyhodnocují zleva doprava.



8. **Nastavení parametrů**
9. **Přiřazení hodnot proměnným nebo volání příkazů**



- **Hledání příkazu**
 - absolutní/relativní cesta k příkazu
 - funkce
 - vnitřní příkaz interpretu
 - pokud je příkaz zadán pouze jménem a není to funkce ani vnitřní příkaz, pak se hledá první výskyt spustitelného programu v adresářích definovaných v proměnné **PATH** zleva doprava

\$proměnná

dosadí se obsah proměnné

- Příklady proměnných

\$HOME

domovský adresář

\$PWD

aktuální adresář

\$PS1

definice promptu

\$0

jméno příkazu

\$1,..,\$9

poziční parametr

\$#

počet pozičních parametrů na příkazové řádce

\$\$

identifikační číslo procesu (PID) dané instance shellu

\$?

návratový kód právě ukončeného procesu