Přednáška 1

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

- **Informace a materiály k předmětu**

  http://edux.felk.cvut.cz

- **Přednášky**
  - Út 7:30-9:00 v T2:D3-209, Jan Trdlička, KN:E-324, trdlicka@fel.cvut.cz
  - Pá 7:30-9:00 v T2:D3-209, Jan Zajíc, KN:E-333, zajic@fel.cvut.cz
- **Cvičení**
  - Halové laboratoře T2:E1-7 a T2:E1-8 (Dejvice)
  - OS Solaris 10 (sparc)

Hodnocení studentů I

- **Testy během semestru**
  - 3 x 15 min. test za 10 bodů (4., 6. a 8. cvičení)
  - 1 x 60 min. test za 30 bodů (11. cvičení)
  - Celkem lze získat 60 bodů.

- **Na zápočet je nutné minimum 30 bodů!**

- **Zkouškový test** za 40 bodů (**nutné správně odpovědět na základní otázky**).

Hodnocení studentů II

- **Klasifikace**
  - pokud získáte v semestru ≥ 30 bodů, můžete získat známku bez zkouškového testu

55 – 60 bodů	A (výborně)
49 – 54 bodů	B (velmi dobře)
43 – 48 bodů	C (dobře)
37 – 42 bodů	D (uspokojivě)
30 – 36 bodů	E (dostatečně)

Hodnocení studentů III

- **Klasifikace**
  - pokud chcete získat lepší známku, pak musíte absolvovat zkouškový test a získáte známku podle následující tabulky

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)

Obsah přednášek I

- **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. Síťové rozhraní.
  9. Grafické rozhraní. Secure shell.

Obsah přednášek II

- **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.

Literatura

- **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

Některé distribuce OS Unix

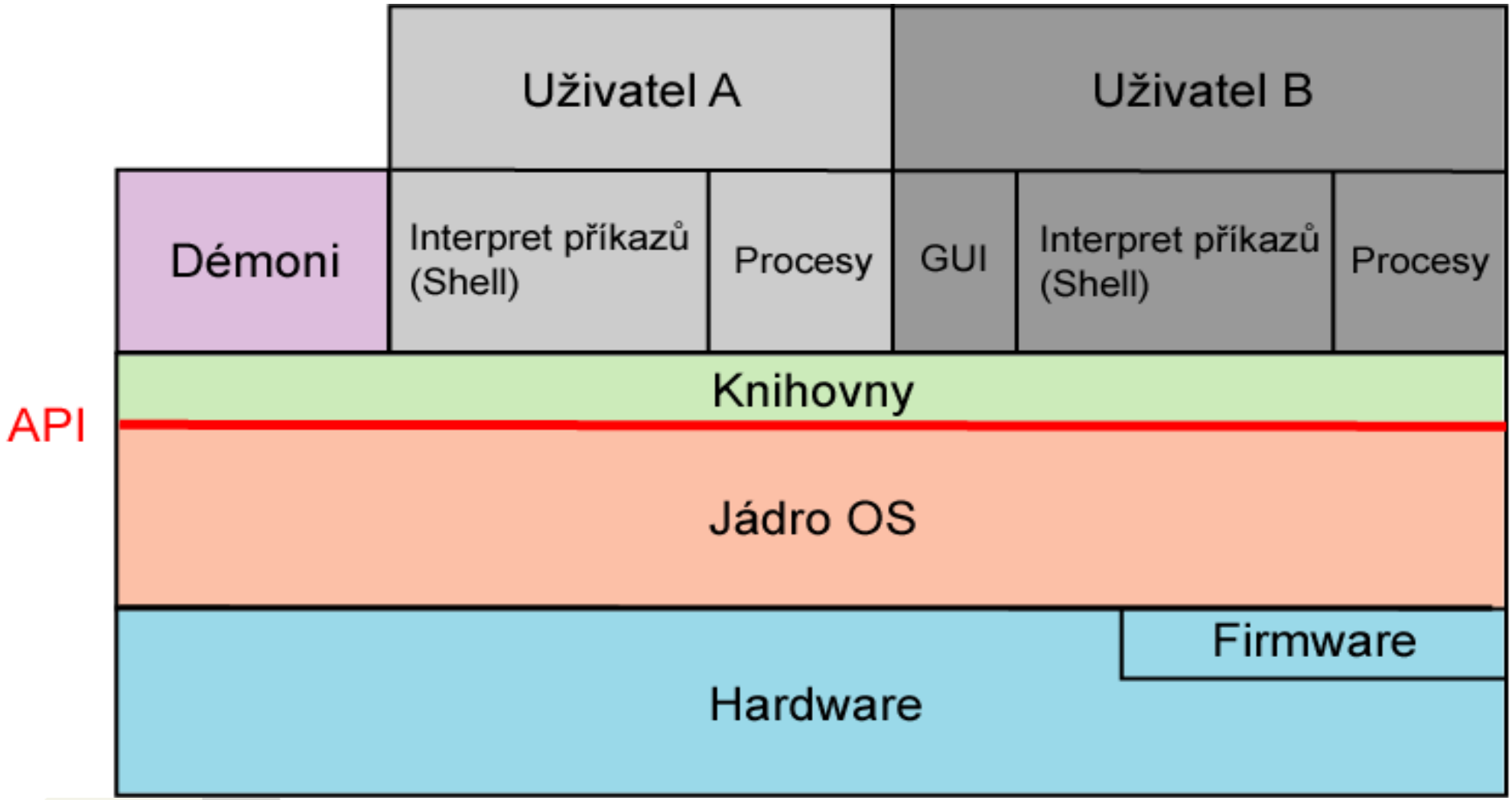
- **OS Linux**
  - Red Hat Enterprise Linux
  - Fedora Core
  - Mandriva Linux (Mandrake Linux)
  - Debian GNU/Linux
  - Ubuntu
  - Gentoo Linux
- **OS Solaris**
  - Solaris 10
  - Open Solaris

- 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,…

Vlastnosti OS Unix I

- **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 multiprocesorových systémů (SMP)**
  - použití vláken v jádře a jejich plánování na různých CPU
- **Unifikované prostředí**
  - přenositelnost mezi platformami (90% jádra v jazyce C)

Vlastnosti OS Unix II

- **Interaktivní přístup s možností vytváření dávek příkazů**
  - shell jako rozhraní uživatele a interpret řídícího 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,…)

Interpret příkazů - shell

- 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 + řídící 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

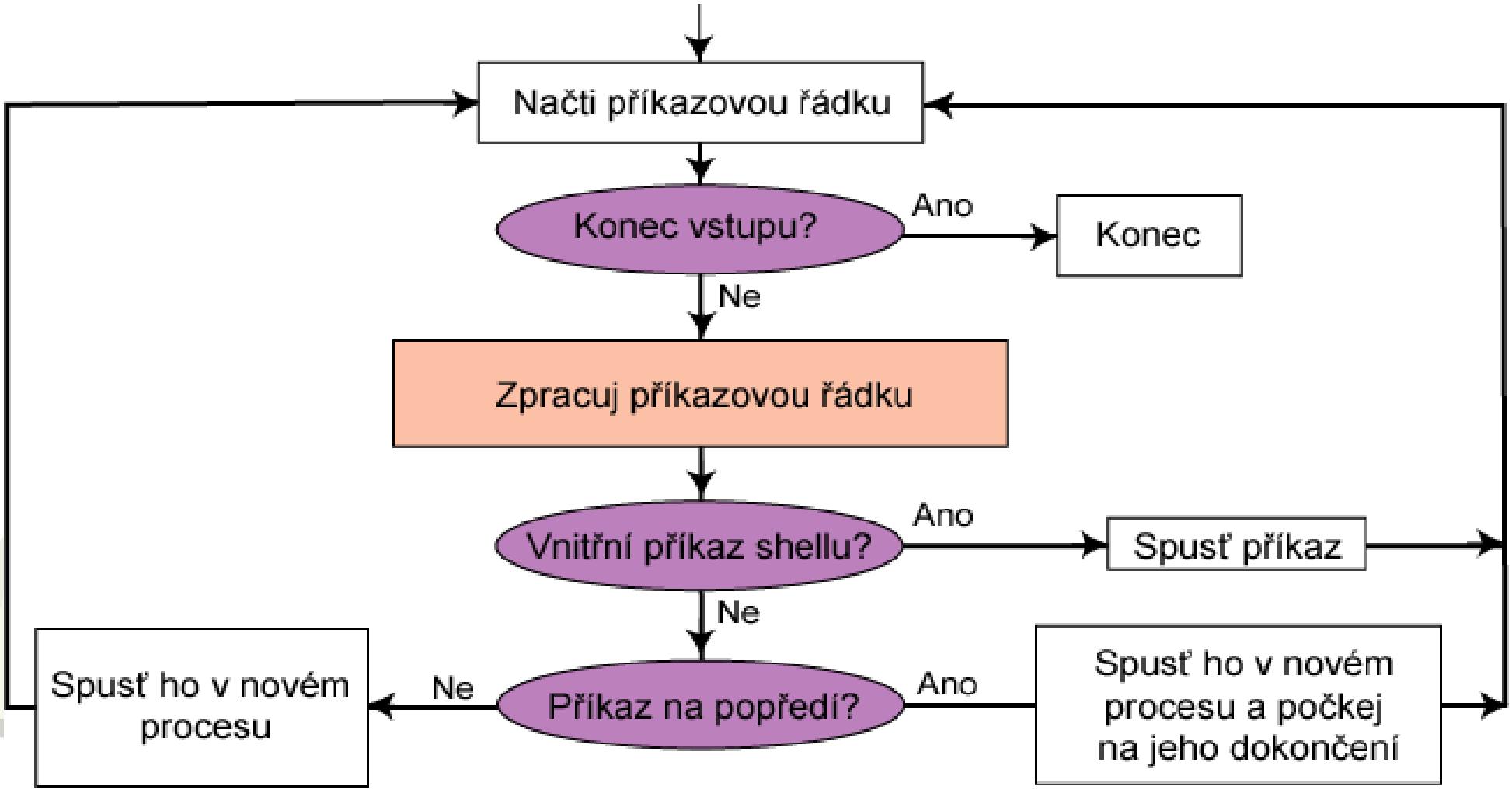
- **Skupina C shellů**

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



Syntaxe příkazové řádky

- **Přiřazení hodnoty proměnné**

  `<výzva>` **`<jméno proměnné>=<hodnota>`**

  `<výzva>`

  je vypsána interpretem před interaktivním čtením příkazu (nikoliv u dávky),
  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

Syntaxe příkazové řádky

- **Formát jednoduchého příkazu**

  `<výzva>` **`<jméno příkazu> <argumenty>`**

  `<jméno příkazu>`

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

  `<argumenty>`

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

  přepínače 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.

Zpracování příkazové řádky

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

- jednoduchý příkaz (simple command) =
  - posloupnost přiřazení hodnot proměnným oddělených mezerami nebo tabelátory
  - 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 standarním 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 (zkratek příkazů)
   - znaku tilda (~)
   - příkazů

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

   - aritmetických výrazů $((výraz))
   - parametrů a proměnných ($1, $HOME,…)
5. **Rozdělení na slova**

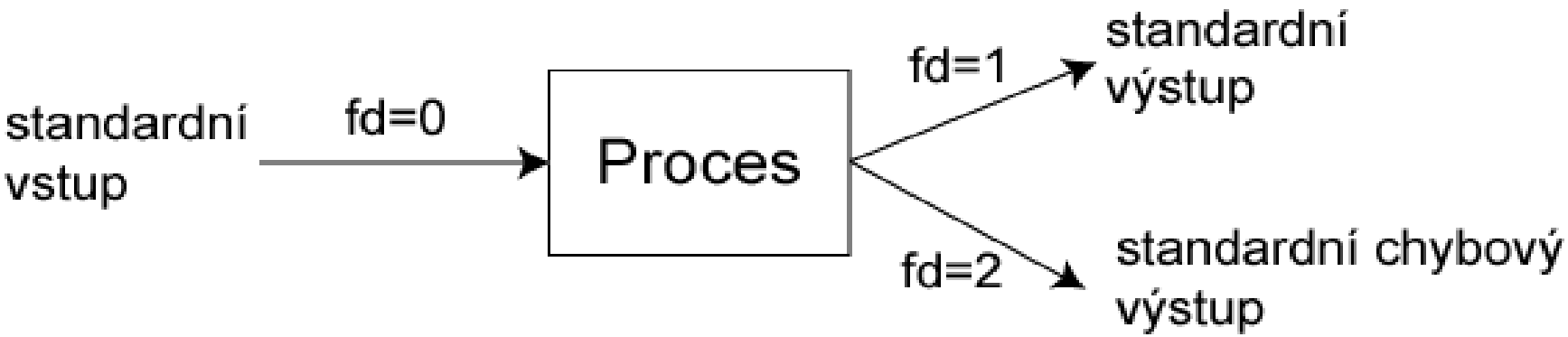
newline mezera TAB	oddělovače slov (lze změnit pomocí proměnné IFS)

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.

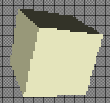
Zpracování příkazové řádky

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

- Při vícenásobném přesměrování se přesměrování vyhodnocují zleva doprava.

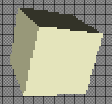


Zpracování příkazové řádky

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

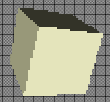
Volání příkazu

- **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



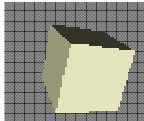
Příklady

```
$ JMENO=Honza
$ echo $JMENO
Honza

$ jmeno="Petr Jitka Tomas"
$ echo $jmeno
Petr Jitka Tomas

$ A=xxx
$ export A
```



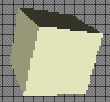
Příklady

```
$ ls
$ ls -lia
$ ls -lia /etc

$ ls -i  *
$ rm -i  *

$ cut -d: -f1 /etc/passwd

$ find /etc -type f -name K[0-1][0-1]*
```



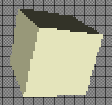
Příklady

```
$ echo *
f1 f2 f3 f31 f32 f33 f34 f35
$ echo \*
*
$ echo '*' "*"
* *
$ echo "Adresar $PWD obsahuje `ls|wc -l` souboru"
Adresar  /home/k336/trdlicka  obsahuje      8  souboru

$ echo "Hodnota \$HOME je $HOME"
Hodnota  $HOME  je  /home/k336/trdlicka
```



Příklady

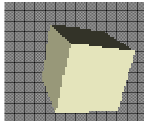
```
$ echo *
a1 b c f1 f2 f3 f31 f32 f33 f34 f35
$ echo f*
f1 f2 f3 f31 f32 f33 f34 f35
$ echo f?
f1 f2 f3
$ echo [abc]*
a1 b c
$ echo [!a]*
b c f1 f2 f3 f31 f32 f33 f34 f35
$ echo e*
e*
```



Příklady

```
$ date > v.txt
$ finger >> v.txt
$ cat v.txt
Sat Sep 30 17:23:11 MEST 2006
Login    Name          TTY    Idle   When         Where
skvorj   Jiri Skvor    pts/2  3:30   Sat 13:46    skvor.felk.cvut.cz
trdlicka Jan Trdlicka  pts/19        Sat 16:56    r5bx111.net.upc.cz

$ find / 1> v.txt  2> /dev/null &
$ find / 1> v.txt  2>&1 &
$ mail honza < zprava
```

```
$ cat > s.txt << KONEC
> Dobry den.
> Jak se mate?
> Ja dobre. KONEC
> KONEC

$ cat s.txt
Dobry den.
Jak se mate?
Ja dobre. KONEC

$ ls > /dev/null 2>&1
```



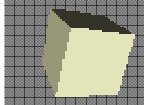
Příklady

```
$ find / 1> v.txt  2> /dev/null &
```

```
$ date ; sleep 2 ;date
Sat Sep 30 17:45:54 MEST 2006
Sat Sep 30 17:45:56 MEST 2006
```

```
$ finger | tail +2 | wc -l
     7
```

```
$grep XYZ f.txt  &&  lp f.txt
$grep XYZ f.txt  ||  echo "XYZ nenalezeno"
```



Příklady

```
$ cd; pwd; (cd /etc; pwd); pwd
/home/k336/trdlicka
/etc
/home/k336/trdlicka
```

- Příklady

```
$ A=`pwd`; echo $A ; cd `echo /etc`; pwd ; cd $A ; pwd
/home/k336/trdlicka/tmp/uos
/etc
/home/k336/trdlicka/tmp/uos
```

Náhrada proměnné

$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