Úvod do UNIXu

Libor Forst

- Úvod, charakteristika
- Historie, principy
- Systém souborů, organizace, příkazy
- Procesy, životní cyklus, komunikace
- Shell: koncepce, příkazy
- Zpracování textu (ed, grep, sed, vi, awk)

Literatura

- L.Forst: Shell v příkladech aneb aby váš UNIX skvěle shell; Matfyzpress 2010
  `www.yq.cz/SvP`

- The Single UNIX® Specification, Version 3 (POSIX), The Open Group Base Specifications Issue 7, IEEE Std 1003.1-2008
  `www.opengroup.org/onlinepubs/9699919799`

- manuálové stránky

Literatura (základy)

- J. Brodský, L. Skočovský: Operační systém UNIX a jazyk C; SNTL 1989
- L. Petrlík: Jemný úvod do systému UNIX; Kopp 1995
- M. Sova: UNIX V - úvod do operačního systému; Grada 1993
- M. Brandejs: UNIX - LINUX - praktický průvodce; Grada 1993; ISBN 80-7169-170-4
- G. Todino, J. Strang, J. Peek: Learning the UNIX Operating System; O'Reilly & Associates 2002; ISBN 0-596-00261-0
- A. Robbins: UNIX in a nutshell; O'Reilly & Associates 2006; ISBN 978-0-596-10029-2
- L. Lamb: Learning the vi Editor; O'Reilly & Associates 1990; ISBN 0-937175-67-6

Literatura (programování)

- M. Jelen: UNIX V - programování v systému; Grada 1993; ISBN 80-85623-16-1
- C. Newham, B. Rosenblatt: Learning the bash Shell; O‘Reilly & Associates 2005; ISBN 0-596-00965-8
- D. Dougherty: sed & awk; O‘Reilly & Associates 1997; ISBN 978-1-565-92225-9
- A. Robbins, N. Beebe: Classic Shell Scripting; O'Reilly & Associates Inc., 2005; ISBN 978-0-596-00595-5
- C. Albing, J. Vossen, C. Newham: bash Cookbook; O'Reilly & Associates Inc., 2007; ISBN 978-0-596-52678-8
- E. Quigley: UNIX Shells by Example; Pearson Education Inc. (Prentice-Hall), 2005; ISBN 0-13-147572-X
- S. Kochan, P. Wood: Unix Shell Programming; SAMS, 2003; ISBN 0-672-32390-3

Literatura (principy)

- M.J.Bach: The Design of the UNIX Operating System; Prentice-Hall 1986
- L.Skočovský: Principy a problémy operačního systému UNIX; Science, 1993; ISBN 80-901475-0-X
- L.Skočovský: UNIX, POSIX, Plan9; L. Skočovský, Brno, 1998; ISBN 80-902612-0-5

- M.Welsh, L.Kaufmann: Používáme LINUX; ComputerPress 1997 (O'Reilly); ISBN 80-7226-001-4

- E. Raymond: The Art of UNIX Programming; Addison Wesley; 2004; ISBN 0131429019

Konvence

- Pevná část příkazu (neproporcionálním fontem)

  - píše se tak, jak je zapsána:

    man [**-k**] [*section*] *topic*

- Proměnlivá část příkazu (kurzívou)

  - doplní se požadovaný text (slovo, číslo apod.):

    man [**-k**] [*section*] *topic*

- Volitelná část příkazu:

  man [**-k**] [*section*] *topic*

- Výběr z více variant:

  {**BEGIN** | **END** | */regexp/* | *cond* | } **{** *cmds* **}**

Historie UNIXu

- 1925 - **Bell Laboratories** - výzkum v komunikacích
- 60. léta - s General Electric a MIT vývoj OS **Multics** (MULTIplexed Information and Computing System)
- 1969 - Bell Labs opouští projekt, **Ken Thompson** píše asembler, základní OS a systém souborů pro PDP-7
- 1970 - Multi-cs => **Uni-x** (snad **Brian Kernighan**)
- 1971 - Thompson žádá nový počítač PDP-11 pro další vývoj - zamítnuto
- Thompson předstírá vývoj systému automatizované kanceláře - počítač přidělen => zpracování textů
- 1973 - UNIX přepsán do jazyka C vytvořeného za tím účelem **Dennisem Ritchiem**

Divergence UNIXu

- pol. 70. let - uvolňování UNIXu na univerzity: především University of California **Berkeley**
- 1979 - v Berkeley přepisují UNIX pro 32bitový VAX **BSD Unix** (Berkeley System Distribution) verze 3.0; dnes verze 4.4
- Bell Labs přecházejí pod **AT&T** a pokračují ve vývoji verze **III** až **V.4** - tzv. **SVR4**
- UNIX uvolněn i pro komerci: Microsoft a SCO vyvíjejí pro Intel **XENIX**
- vznikají UNIX International, OSF (Open Software Foundation), X/OPEN,...

Varianty UNIXu

- SUN: **Sun OS, Solaris**
- Silicon Graphics: **Irix**
- DEC: **Ultrix**, **Digital Unix**
- IBM: **AIX**
- HP: **HP-UX**
- Siemens Nixdorf: **SINIX**
- Novell: **UNIXware**
- SCO: **SCO** Unix

- FreeBSD, NetBSD, OpenBSD,...
- Linux

Standardy UNIXu

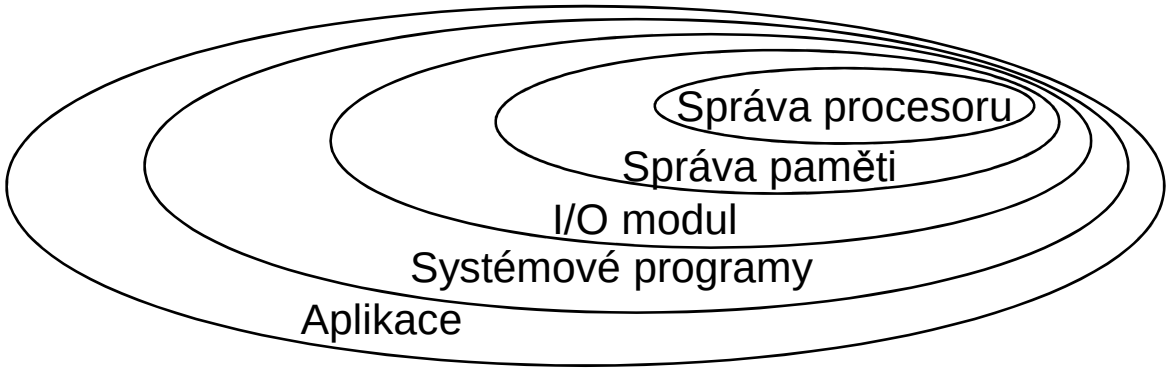
- SVID (System V Interface Definition)
  - "fialová kniha", kterou AT&T vydala poprvé v roce 1985 jako standard, jehož splnění je nutnou podmínkou pro použití obchodního názvu UNIX
- POSIX (Portable Operating System based on UNIX)
  - série standardů organizace IEEE značená P1003.xx, postupně je přejímá vrcholový nadnárodní orgán ISO
- XPG (X/Open Portability Guide)
  - doporučení konsorcia X/Open, které bylo založeno v r. 1984 předními výrobci platforem
- Single UNIX Specification
  - standard organizace Open Group, vzniklé v roce 1996 sloučením X/Open a OSF
  - Version 2 (**UNIX98**), Version 3
  - splnění je nutné pro užití obchodního názvu UNIX

Charakteristika UNIXu

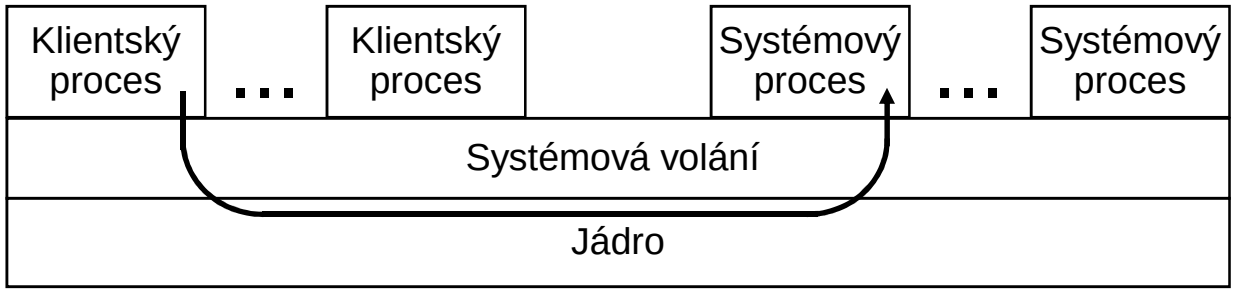
- poučení ale nezatížení minulostí
- nekomerční prostředí
- otevřený operační systém
- systém souborů
- uživatelé, skupiny
- procesy, komunikace
- interpret příkazů, grafické prostředí
- utility, jazyk C
- přenositelnost, modifikovatelnost
- síťová podpora
- volně šiřitelný SW (např. GNU)
- příkaz **`man`**

Modely OS

Klasický OS



UNIX



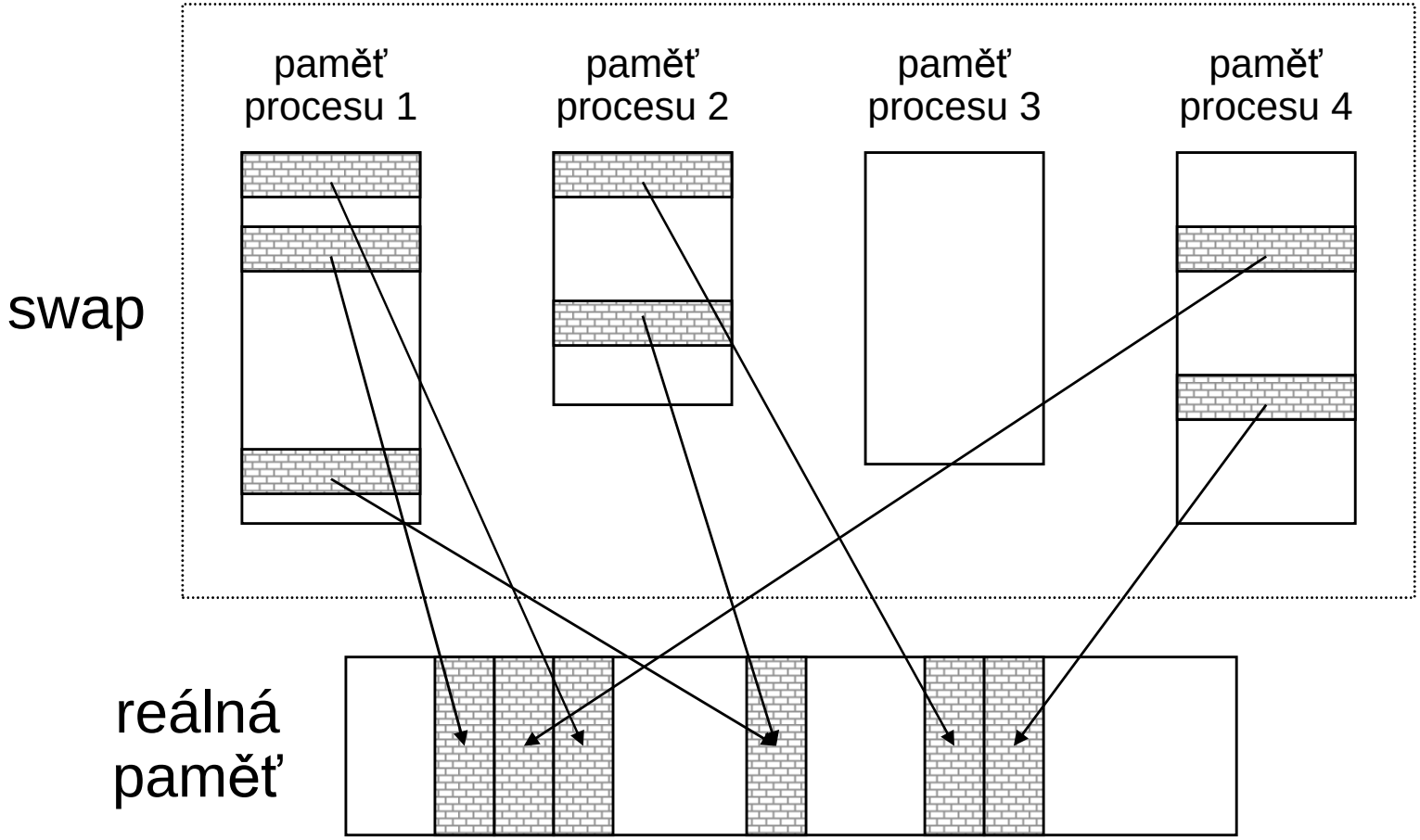
Funkce jádra OS

- Řízení provádění úloh (vytváření, ukončení, suspendování, komunikace, přístup k periferiím,...)
- Správa systému souborů (organizace disku, vytváření a mazání souborů, práva, udržování konzistence,...)
- Správa paměti (přidělování, uvolňování, ochrana, odkládání dočasně nepoužívané paměti - *swapping* resp. *paging*,...)
- Plánování procesů pro sdílení času CPU (plánovací algoritmus, přidělování časových kvant, priority,...)

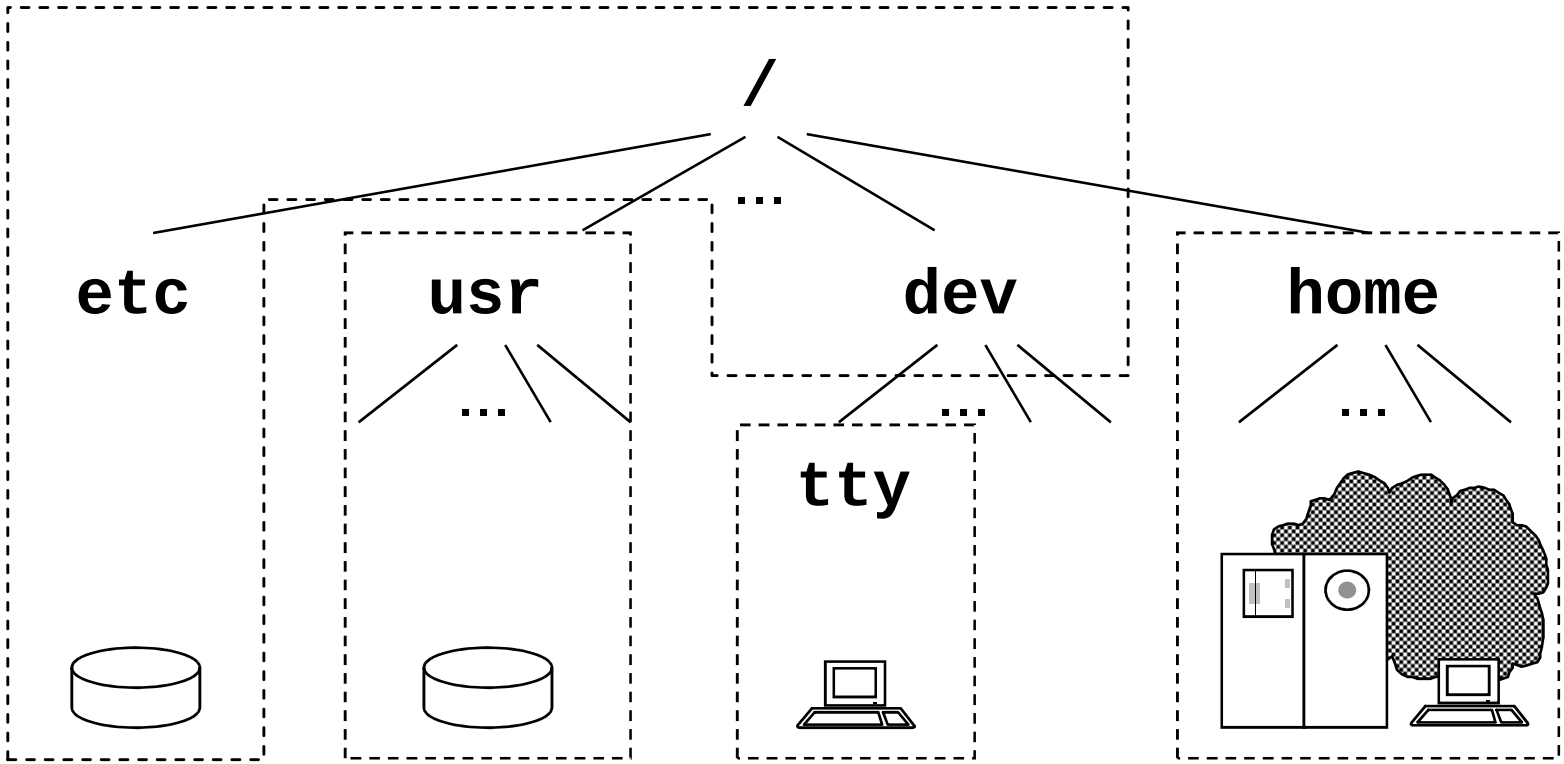
HW požadavky

- Možnost běhu procesu ve dvou režimech:
  - uživatelský (*user mode*): omezený přístup k paměti, instrukcím,...
  - privilegovaný režim (*kernel mode*)
- Hierarchické ošetření přerušení
  - vnější: HW (disky, periferie, ...)
  - vnitřní: událost CPU (adresace, dělení nulou, ...)
  - softwarové: použití speciální instrukce
- Správa paměti - oddělení virtuálního a skutečného adresního prostoru

Virtuální paměť



Jednotný hierarchický systém souborů



Strom adresářů

- **`/bin`** - základní systémové příkazy
- **`/dev`** - speciální soubory (zařízení, *devices*)
- **`/etc`** - konfigurační soubory
- **`/lib`** - základní systémové knihovny
- **`/tmp`** - veřejný adresář pro dočasné soubory
- **`/home`** - kořen domovských adresářů *
- **`/usr/man`** - manuálové stránky *
- **`/usr/spool`** - *spool* (tisk, pošta, ...)
- **`/usr/local`** - lokální instalace *
- **`/usr/include`** - hlavičkové soubory jazyka C

* na některých systémech se může umístění lišit

Proces, komunikace

- Proces
  - zjednodušeně:
    běžící uživatelský nebo systémový program
  - vzniká duplikací rodičovského procesu
  - výpis procesů: příkaz **`ps`**

- Komunikace
  - při startu otec předává data synovi, naopak nelze!
  - roura - tok dat od producenta ke konzumentu:
    `ls | more`
  - další prostředky (např. sdílená paměť)

Interpret příkazů (*shell*)

- základní program pro ovládání UNIXu
- nezávislá komponenta systému: více shellů
- formát příkazů:

  příkaz -přepínače operandy př. **`ls -l /etc`**

- metaznaky, např.:

  `ls *.c > "vypis *.c"`

- příkazy:
  - interní: např. **`echo`**, **`cd`**, **`pwd`**
  - externí: soubory uložené na disku (cesta: **`PATH`**)

Jazyk shellu

- shell interpretuje vlastní programovací jazyk
  - řídící konstrukce (např. **`for`**, **`if`**)
  - proměnné

    `PATH=/bin:/usr/bin:$HOME/bin`

- jazyk řídí textové substituce (*textový procesor*)
- programování přímo na příkazové řádce
- shell-skript - soubor s programem pro shell

  `sh test.sh; ./test.sh`

Příkaz **man**

- Volání:

  man [**-k**] [*section*] *topic*

- Sekce manuálových stránek:
  - **1** - obecné uživatelské příkazy
  - **2** - služby jádra systému (*syscalls*)
  - **3** - knihovní funkce (jazyka C)
  - **4** - zařízení a ovladače zařízení
  - **5** - formáty (konfiguračních) souborů
  - **6** - triviální aplikační programy
  - **7** - různé
  - **8** - administrátorské příkazy a programy

Seznam uživatelů (**`/etc/passwd`**)

```
forst:DxyAF1eG:1004:11:Libor Forst:/u/forst:/bin/sh
```

Význam jednotlivých polí:

- uživatelské (*login*) jméno
- zakódované heslo (dnes např. v **`/etc/shadow`**...)
- číslo (*UID*); superuživatel (*root*) má UID 0
- číslo (*GID*) primární skupiny uživatele
- plné jméno (s příp. komentářem)
- domovský adresář
- login-shell

Seznam skupin (**`/etc/group`**)

```
users::11:operator,novak
```

Význam jednotlivých polí:

- jméno skupiny
- *nepoužito*
- číslo skupiny (*GID*)
- seznam členů skupiny

Ve skupině jsou navíc i všichni uživatelé, kteří ji mají uvedenu jako svoji primární skupinu.

Uživatelská relace

Po přihlášení k systému (lokálně n. vzdáleně - např. pomocí **`ssh`**, **`putty.exe`**) se uživateli spustí jeho *login-shell*. Tím se zahájí jeho uživatelská relace (*session).*

- výpis informací o systému: **`uname`** [**`-amnrsv`**]
- ukončení session: **`logout`**
- změna uživatele (login-shellu): **`login`** *user*
- start shellu nového uživatele: **`su`** [**`-`**] [*user*]
- ukončení shellu: **`exit`**
- zjištění identity uživatele: **`id`**, **`whoami`**, **`who am i`**
- výpis nalogovaných uživatelů: **`who`**, **`w`**
- výpis logu relací: **`last`**

Komunikace mezi uživateli

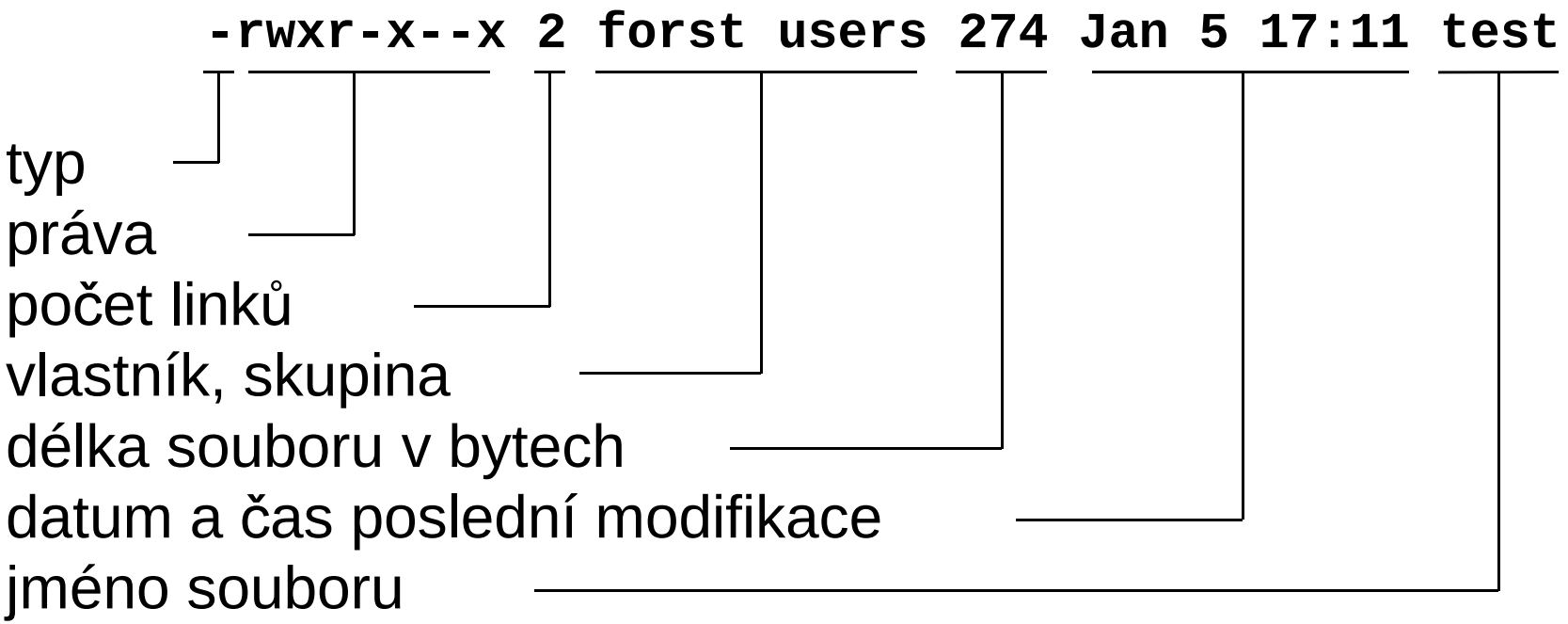
- on-line (zprávy):
  - zaslání: **`write`** *user*
  - potlačení příjmu: **`mesg`** [**`y`** | **`n`**]
- on-line (rozhovor):
  - příkaz: **`talk`** *user*[@*host*]
- off-line: e-mail
  - příjem: **`mail`**
  - posílání: **`mail`** [**`-v`**] [**`-s`***subject*] *email...*
  - zpráva o příjmu: **`biff`** [**`y`** | **`n`**]
  - přesměrování dopisů: **`$HOME/.forward`**

```
forst@ms.mff.cuni.cz
"| /usr/local/bin/filter"
```

Systém souborů

- hierarchický systém
- jednotný přístup k zařízením, adresářům aj.
- diskové svazky, přístup k síťovým diskům
- konzistence, synchronizace (**`sync`**, **`fsck`**)
- ochrana souborů (přístupová práva)
  - pravidla pro jména (délka, znaková sada, case senzitivita)
- cesty (absolutní, relativní, **`.`**, **`..`**)
- formát textových souborů (**<LF>**)

Příkaz `ls`



volby: dlouhý výpis (**`l`**), výpis do 1 sloupce (**`1`**), psát i skryté (**`aA`**), třídit podle času (**`t`**), třídit pozpátku (**`r`**), značit typ souboru (**`F`**), vypisovat rekurzivně (**`R`**), nevypisovat obsah adresářů (**`d`**), sledovat linky (**`L`**)

Typy souborů

- Typy souborů ve výpisu **`ls`**:
  - **`-`** obyčejný soubor (*regular file*): posloupnost bytů
  - **`d`** adresář *(directory)*: sada binárních záznamů o souborech a podadresářích
  - **`b`** blokový speciální soubor, zařízení *(block device)*
  - **`c`** znakové (raw) zařízení *(character device)*
  - **`l`** symbolický link
  - **`p`** pojmenovaná roura *(pipe)*
  - **`s`** *socket*

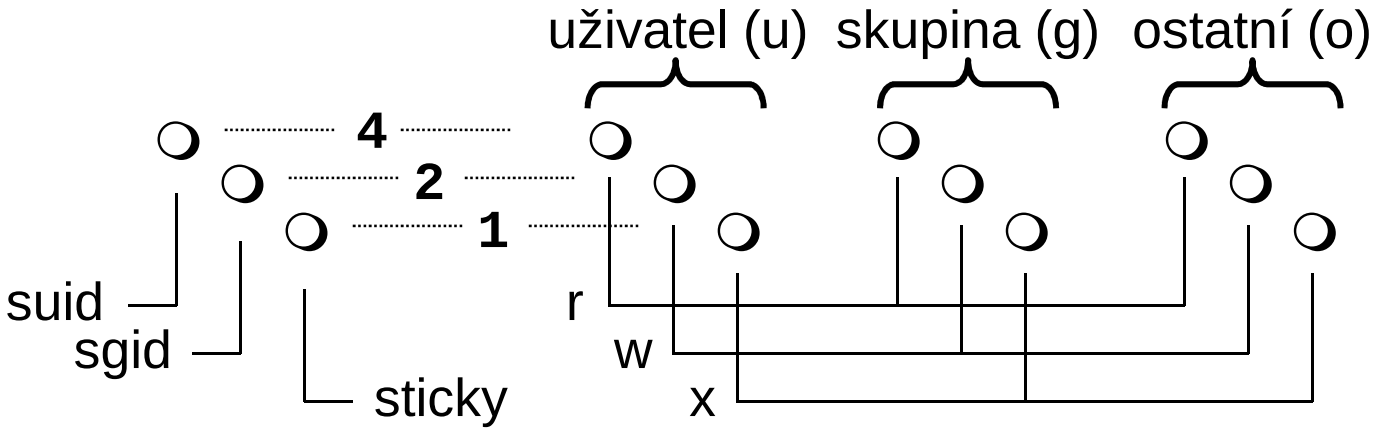
- Rozpoznání typu: příkaz **`file`**

Přístupová práva (file modes)

- tři kategorie vlastníků: user (**u**), group (**g**), others (**o**); platí vždy nejspeciálnější kategorie, v níž je uživatel
- tři práva: prohlížení (read: **r**), modifikace (write: **w**), provádění souboru / práce s adresářem (execute: **x**)

- setUID, setGID (**s**) pro proveditelné soubory: běh pod propůjčenou identitou vlastníka (uživatele / skupiny)
- setGID pro adresář: nové soubory budou mít stejnou skupinu jako adresář (default na řadě systémů)
- sticky bit (**t**) pro adresáře: mazat a přejmenovávat soubory smějí jen vlastníci souborů a root (př. `/tmp`)

Změna přístupových práv



- změna práv (smí pouze uživatel-vlastník a root):
  `chmod` [**`-R`**] **`og-w,+x`** *file...*
  `chmod` [**`-R`**] **`751`** *file...*
- změna vlastníka (smí pouze root): **`chown`**, **`chgrp`**
- defaultní maska: **`umask`** *mask_complement*
- shell s novou defaultní skupinou: **`newgrp`** *group*

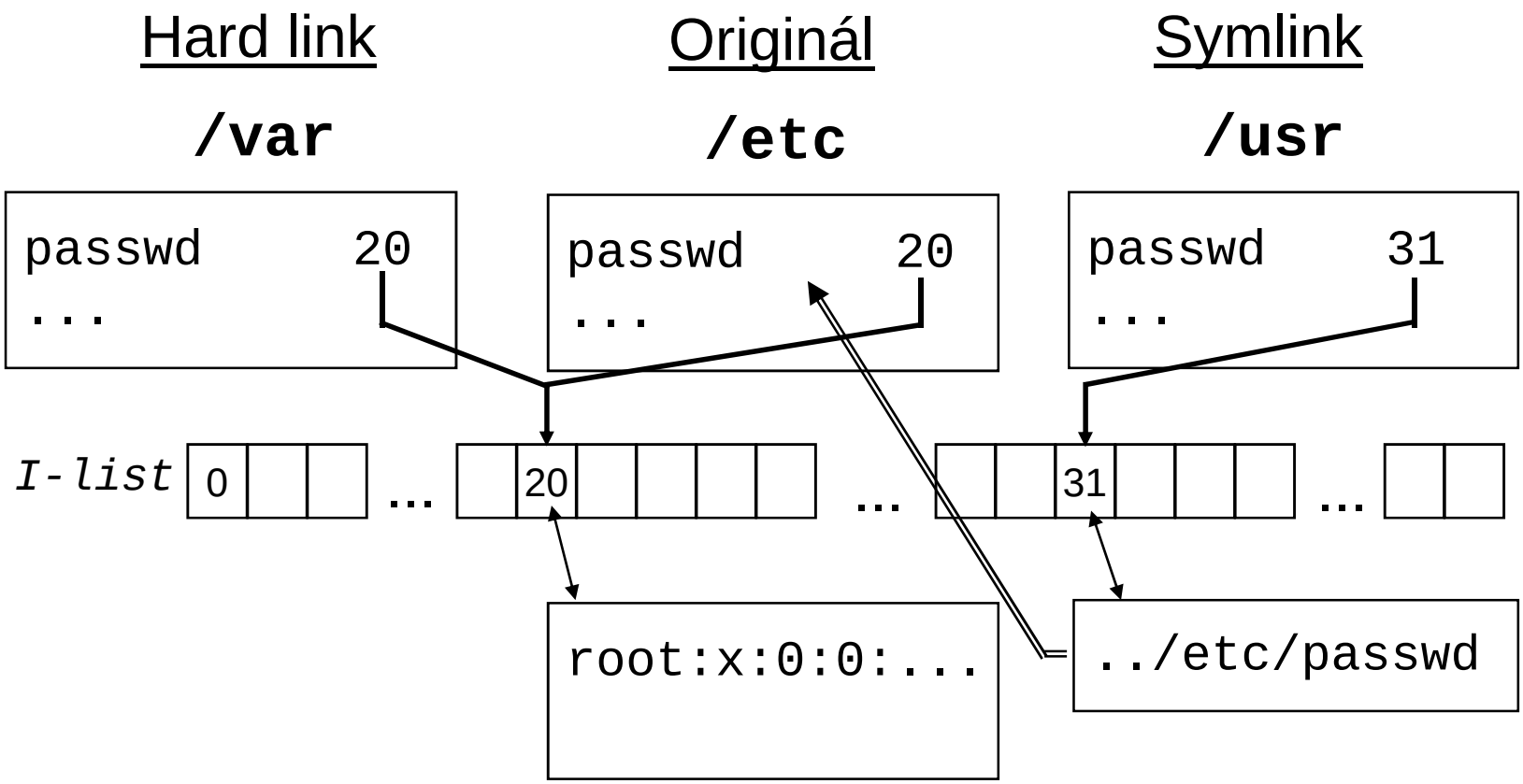
Organizace disku

- Fyzická: *sektor*, *stopa (track)*, *válec (cylindr)*, *povrch*
- Logická: *oddíl (partition)* (odpovídá block/raw device)
  - zobrazení: příkaz **`df`** (display filesystems)
  - konfigurační soubor **`/etc/fstab`**
- Systémová: *filesystem*
  - boot blok
  - superblok(y)
  - i-list (pole i-nodů)
  - datové bloky
- Obraz systému souborů v paměti (**`sync`**, **`fsck`**)

Index node

- Každý soubor v systému souborů má právě jeden i-node, který obsahuje:
  - počet linků
  - ID vlastníka (uživatele a skupiny)
  - přístupová práva
  - typ souboru
  - velikost souboru
  - čas
    - poslední modifikace souboru
    - posledního přístupu k souboru
    - poslední modifikace i-nodu
  - odkazy na datové bloky
- Výpis seznamu souborů s čísly i-nodů: **`ls -i`**
- Výpis informace z i-nodu (není v normě): **`stat`**

Linky



`ln /etc/passwd passwd`

`ln -s ../etc/passwd passwd`
(zacyklení: `Too many symlinks`)

Adresace datových bloků

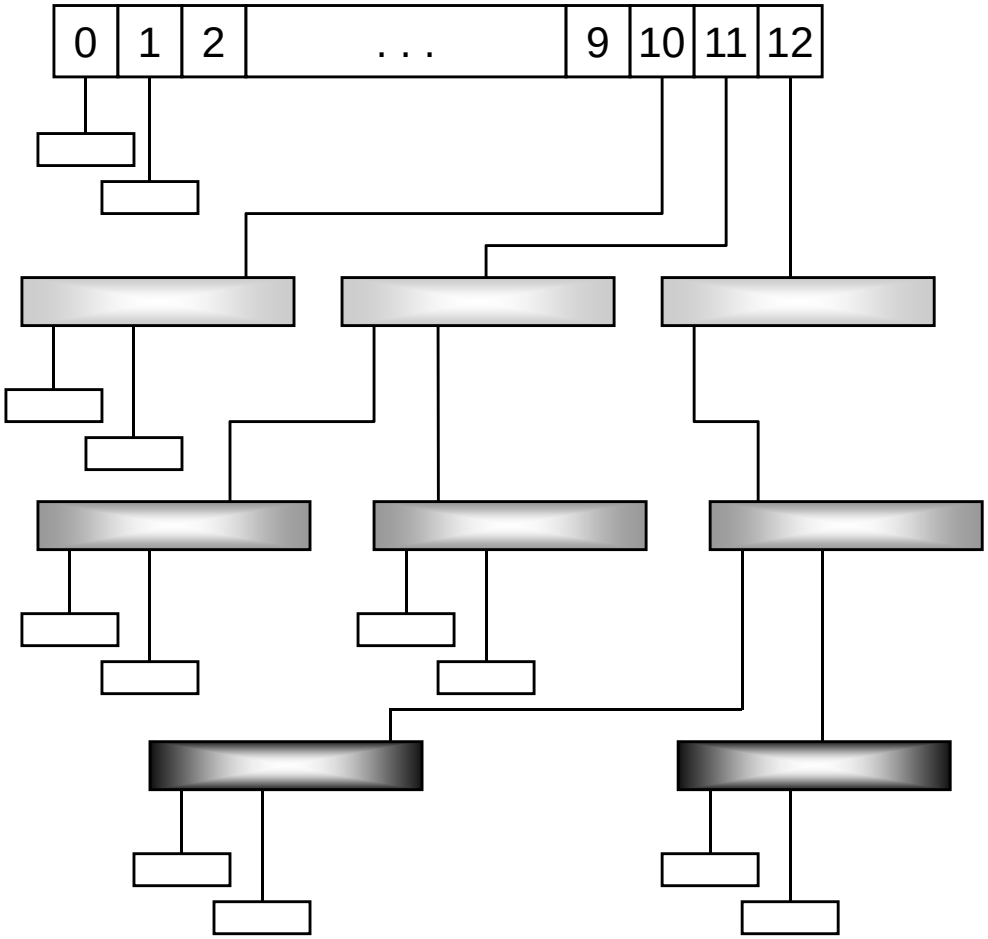
Tabulka přímých odkazů v I-node

Blok nepřímých odkazů 1.řádu

Blok nepřímých odkazů 2.řádu

Blok nepřímých odkazů 3.řádu

Datové bloky



Obecné příkazy

- kopírování souboru: **`cp`** `[-prR]`
- přesun (přejmenování) souboru: **`mv`**
- smazání souboru: **`rm`** `[-rfi]`
- změna data a času: **`touch`** `[{ -t`*čas*`| -r`*soubor* `}]`
- změna aktuálního adresáře: **`cd`**
- cesta k aktuálnímu adresáři: **`pwd`** `[-P]`
- vytvoření adresáře: **`mkdir`** `[-p] [-m`*mode* `]`
- zrušení adresáře: **`rmdir`**

- není undelete !

Výpis souboru

- výpis (zřetězení) souborů: **`cat`** [*files*]
- výpis souborů po stránkách: **`more`**, **`pg`**, **`less`**
- výpis začátku souboru: **`head`** [**`-n`** *n*] [*files*]
- výpis konce souboru: **`tail`** [{**`-n`**|**`-c`**} [**`+`**]*n*] [**`-f`**] [*files*]
- výpis souboru pro tisk: **`pr`**
- výpis souboru s číslováním řádek: **`nl`**
- počet bytů, slov a řádek: **`wc`** [**`-cwl`**]
- kopírování na výstup a do souboru: **`tee`** [**`-a`**] *file*

- výpis binárního souboru: **`od`** [**`-t`***fmt*] [**`-j`***off*] [**`-N`***len*]
- výpis řetězců: **`strings`**

Příkaz **more**

- Volání:

  more [**-***n*] { **+***line* | **+/***regexp* | } [*files*]

- Příkazy (* - může předcházet prefix počtu *k*):
  - mezera, **d** ... další stránka, půl stránky (*)
  - Enter ... další řádka (* - *k* nastaví default)
  - **s**, **f**, **b** ... přeskoč *k* řádek, stránek, stránek zpět (*)
  - */regexp*, **n** ... hledej *k*-tý výskyt řetězce (*)
  - **'** ... návrat na začátek hledání
  - **!***cmd*, **v** ... start shellu, editoru
  - **=**, **h** ... výpis pozice, helpu
  - **:n**, **:p** ... přechod na další soubor

Tisk

	SUSv3	System V	BSD
• tisk:	**`lp`** [*file*]	**`lp`** [*file*]	**`lpr`** [*file*]
	`-d` *printer*	**`-P`** *printer*	**`-d`** *printer*
• výpis stavu tisku:	**`lpstat`** *job*		**`lpq`** *job*
			`-d` *printer*
• zrušení tisku:	**`cancel`** *job*		**`lprm`** *job*
			`-d` *printer*

- popis „tiskáren“: **`/etc/printcap`**
- implicitní tiskárna: proměnná **`PRINTER`**
- spool-oblast: **`/var/spool/*`**
- formátování tisku: **`pr`**, **`mpage`**

Zpracování textu

- porovnávání souborů resp. adresářů:

  `diff` [**`-bBi`**] { **`-e`** | **`-C`***n* | **`-rqs`** } *file1 file2*

  `comm` [**`-123`**] *file1 file2* *(musí být setříděné)*

- výběr polí z řádek souboru (nemění pořadí polí):

  `cut` [**`-s`**] { **`-c`***list* | **`-f`***list* **`-d`***char* } [*files*]

- spojení souborů „po sloupcích“ resp. řádek souboru:

  `paste` [[**`-s`**] **`-d`***chars*] [*files*]

- rozdělení souboru po řádcích n. blocích:

  `split` [{ **`-l`***lines* | **`-b`***bytes*[{**`k`**|**`m`**}] }] [*file* [*name*]]

- konverze znaků:

  `tr` [**`-cds`**] *table1* [*table2*] př.: `tr 'A-Z\n' 'a-z:'`

Příkaz **`sort`**

- Volání:

  `sort` [**`-s`**] [**`-k`***beg*[**`,`***end*][*mod*]] [**`-t`***d*] [**`-ucm`**] [*files*]
- Setřídí soubory na výstup resp. do souboru (**`-o`** *file*)
- Zadání třídícího pole:
  - *beg* ... pozice prvního znaku, *end* ... pozice posledního
  - tvar: *field*[**`.`***char*] ... číslování od 1
- Modifikátory: **`b`** (bez mezer), **`f`** (ignorecase),
  `n` (čísla), **`r`** (opačně)
- Přepínače: **`t`** (oddělovač pole, default: posloupnost mezer),
  `u` (vyluč stejné klíče), **`m`** (merge only),
  `c` (check only), **`s`** (stable - není v normě)
- Pozor na lokální nastavení (`LC_ALL=C`)
- Podobný příkaz: **`uniq`** (netřídí, umí např. vypsat počty)

Příkaz **`find`**

- Volání: **`find`** *cesta... podmínka... akce*
- Podmínky:
  - **`name`**, **`path`**, **`size`**, **`type`**, **`links`**, **`inum`**, **`fstype`**
  - **`user`**, **`group`**, **`perm`**
  - **`atime`**, **`ctime`**, **`mtime`**, **`newer`**
  - hloubka vnoření ve stromě
  - negace (**`!`**), **`-o`**, **`-a`**, závorky
  - číselné hodnoty: *n*, **`+`***n*, **`-`***n*
- Akce:
  - **`print`** (typicky default)
  - **`exec`**; umístění jména: **`{}`**, konec příkazu: středník
- Příklad:
  ```
  find / -name *core -mtime +7 -exec rm {} ";"
  ```
- Varianty: **`which`**, **`whereis`**

Příkaz **dd**

- Provádí kopírování a konverzi dat
- Název a syntaxe parametrů odvozena od JCL příkazu DD (Data Definition) systému IBM 360
- Parametry:
  - **`if=`***file* - vstup (impl. standardní vstup)
  - **`of=`***file* - výstup (impl. standardní výstup)
  - **`bs=`***expr* - velikost bloku (*n*[**k**][**x***n*[**k**]]...)
  - **`count=`***n* - počet bloků
  - **`skip=`***n* - posun od začátku (seek)
  - **`conv=`***c*[**`,`***c*]... - konverze
- Konverze ASCII/EBCDIC, pevná délka řádky/LF
- Př.:

```
dd if=soubor bs=8 count=1
```

Příkaz **`join`**

- Provádí databázový *join* - slití souborů podle rovnosti záznamů v dané klíčové položce
- Přepínače:
  - **`t`** *c* - oddělovač polí
  - {**`1`**|**`2`**} *f* - číslo klíčového pole v souboru 1 resp. 2
  - **`a`** *n* - ze souboru *n* se berou i nespárované řádky
  - **`v`** *n* - ze souboru *n* se berou jen nespárované řádky
  - **`e`** *str* - náhrada za chybějící pole
  - **`o`** *list* - přesný tvar výstupu
- Formát popisu výstupu:
  - seznam polí oddělených čárkami nebo mezerami, příp. zapsaný do více parametrů
  - tvar pole: *n*`.`*f* resp. **`0`**
- Default: první pole je klíč, výpis všech polí po řadě, oddělovačem je posloupnost bílých znaků

Příkaz **`xargs`**

- volání: **`xargs`** *cmd*
  - zavolá příkaz *cmd*, jako argumenty doplní obsah standardního vstupu
  - př.: `xargs rm < soubory_ke_smazani`

- volání: **`xargs`** { **`-L`***lines* | **`-n`***words* } *cmd*
  - opakuje příkaz, jako argumenty doplní vždy text z *lines* řádek standardního vstupu resp. každých *words* slov standardního vstupu

- volání: **`xargs -I`***fn cmd*
  - opakuje příkaz pro každou řádku standardního vstupu, její text doplní do příkazu na místa označená *fn*
  - př.: `ls *.c | xargs -I{} cp -p {} {}.bak`

Archivace

- archivace adresářů: **`tar`** {**`c`** | **`t`** | **`x`**} [**`f`** *file*] [*files*]
  - př.: **`tar cf - . | ssh host tar xf -`**
  - distribuce SW balíků
- v normě nahrazen příkazem **`pax`**
- komprese souborů
  - historický standard (**`.Z`**): **`compress`**
  - GNU (**`.gz`**): **`gzip`**, **`gunzip`**
- systémová záloha: **`backup`**, **`dump`**, **`restore`**
- zálohování po síti: **`rdump`**, **`rrestore`**

Řádkové editory

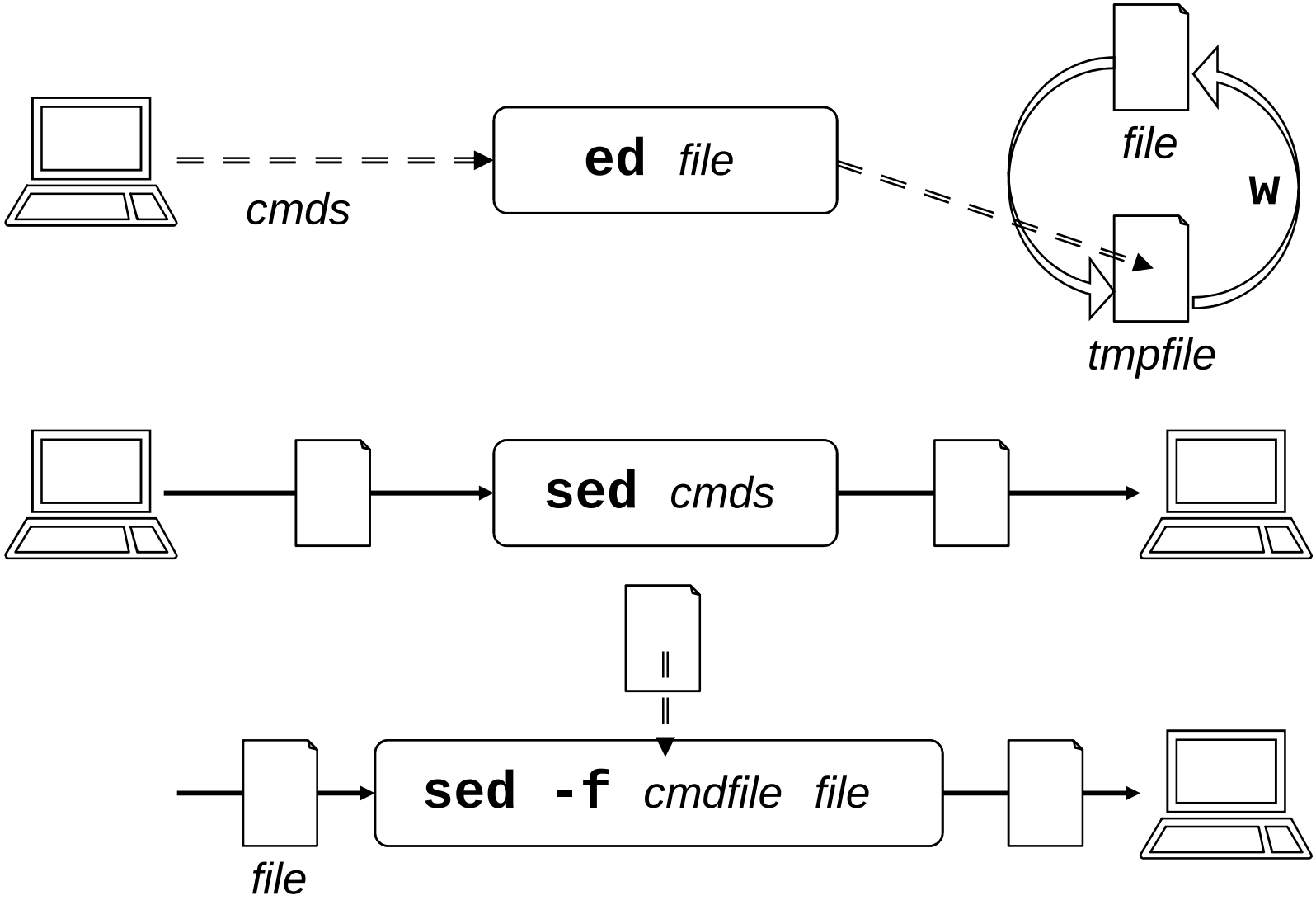
ed - editor dostupný často i v diagnostickém režimu
- edituje kopii souboru, opravy je nutno zapsat zpět
- příkazy se berou ze standardního vstupu
- dávková editace (**ed**-skripty)

- volání: **ed** *soubor*

sed - stream editor
- edituje vstupní proud, výsledek píše na výstup
- editovací příkazy jsou součástí volání

- volání: **sed** *příkazy* [*soubor ...*]
  nebo: **sed -f** *příkazový_soubor* [*soubor ...*]

Schéma práce **ed** a **sed**



Formát příkazu, adresa řádku (**ed**)

- Syntaxe příkazů:

  [*adresa*[**,***adresa*]]*příkaz*[*parametry*]

- V každém okamžiku je jedna řádka aktuální
  (na začátku poslední, dále poslední řádka minulého příkazu)
- Formáty zápisu adresy a jejich význam:

.	aktuální řádka (obvyklý default)
±[*n*]	řádka relativní k aktuální řádce
n	řádka s absolutním číslem *n* (číslováno od 1)
$	poslední řádka souboru
/pat/	následující řádka obsahující vzorek
?pat?	předcházející řádka obsahující vzorek
'x	řádka označená značkou (písmenem) *x*
adr±[*n*]	řádka relativní k řádce s adresou *adr*

Základní regulární výrazy (**ed**, **sed**, **vi**)

Způsob definování řetězců v řadě utilit. Metaznaky:

- **.** … jakýkoliv znak
- **[***list***]**, **[^***list***]** … jakýkoliv znak z výčtu, z doplňku výčtu
  př.: `[a-zA-Z0-9_]`, `[^ ]`, `[]^-]`
- **[[:***třída***:]]** … jakýkoliv znak z třídy
  př.: `[[:alnum:]]`, `[[:xdigit:]]`
- **^**, **$** … začátek a konec řádky (na začátku/konci regexpu)
- *****c … metaznak použitý jako znak (např.: `\.` je tečka)
- *exp****** … libovolné opakování posledního podvýrazu
  př.: `a*`, `[0-9][0-9]*`
- *exp***\{***n***\}**, *exp***\{***m***,**[*n*]**\}** … opakování *n*krát, *m*-*n*krát
- **\(**, **\)**, *****n … uzávorkování části vzoru, zpětná reference
  př.: `\(ab\)*`, `A\(.\)\1A`

Poziční příkazy editoru **ed**

Příkazy s aktuální řádkou jako implicitní adresou,
příkazy označené * nemohou pracovat s blokem:

print, num, list … tisk, s čísly, včetně řídících znaků

delete … mazání řádek

append*, change, insert* … vkládání řádek (ukončení: tečka)

př.:
```
0a
novy radek 1
novy radek 2
.
```

move, to … přesun, kopírování řádek

př.:
```
/begin/,/end/ t $
```

mark* (**k***x*) … nastavení značky *x* (písmeno)

join … spojení řádek (maže LF, impl. +1)

substitute … náhrada řetězců

Příkaz **substitute** (**ed**)

Formát:

s/*pattern***/***replacement***/**{**g**|*n*}

První znak za názvem příkazu definuje oddělovač

př.: `s/\/$//` nebo `s=/$==`

Vzor je regulární výraz, náhrada je text s metaznaky:

- **&** … celý původní text pokrytý regexpem

  př.: `s/.*/(&)/`

- *n* … zpětná reference (pomalá!)

  př.: `s/\(.*\) \(.*\)/\2 \1/`

Globální nahrazování hledá další výskyt regexpu až za místem, které se naposledy modifikovalo:

př.: `s=/\./=/=g` ... nenahradí „`/././`“

Hvězdička „absorbuje“ maximální vyhovující řetězec:

př.: `s/\(.*\)-/\1/` ... smaže poslední mínus

lépe: `s/-$//`

Globální příkazy editoru **ed**

Příkazy s implicitní adresou „celý soubor“:

global, invert (**v**) … provedení příkazu na vybraných řádkách
g/*pattern*/*cmd* [**\<LF>***cmd*]

write (**w** [*file*]) … uložení (pod stejným jménem)

(v případě udání rozsahu se zapíšou jen dané řádky!)
W *file* … připsání do souboru
w!*cmd* … zápis do roury

Příkazy s implicitní adresou „poslední řádka souboru“:

read (**r** [*file*]) … vložení textu souboru

= … výpis čísla řádky

Nepoziční příkazy editoru **ed**

Příkazy bez adresy:

undo	… zrušení poslední opravy
edit (**e** [*file*])	… (znovu-)otevření souboru
file (**f** *file*)	… změna jména editovaného souboru
quit	… ukončení editace
help	… nápověda k poslední chybě

Příklady použití příkazu **`global`**

- **`g/integer/s//longint/g`**

  „prodlouží“ program

- **`g/procedure/i\`**
  `{ begin of procedure }\`
  `.`

  před procedurami odřádkuje (s komentářem)

- **`g/^Chapter/ . W index\`**
  `/./ W index`

  napíše seznam kapitol

- **`g/.*/ m 0`**

  napíše soubor pozpátku

Příkaz **`grep`**

- Původ názvu: **`g/`***re*`/p`
- Varianty:
  - **`egrep`** (**`-E`**, *extended - rozšířené* regulární výrazy)
  - **`fgrep`** (**`-F`**, *fixed* - pouze pevné řetězce)
- Přepínače:

  `-c`(*count*), **`-l`**(*listfiles*), **`-n`**(*number*), **`-q`**(*quiet*)

  `-i`(*ignorecase*) , **`-x`**(*exact*), **`-v`**(*invert*)

  `-e` *expression*, **`-f`** *filename*
- Rozšíření:

  `-w`(*word*) , **`-H`**(*head*)

  `-`*lines* ... počet vypsaných řádek před a po nalezené
- Rychlá implementace regexpů!

Filtr **sed**

- stream editor
- edituje vstup (typicky výstup jiného programu)
- výsledek editace (a/nebo příkazů tisku) vypisuje
- volání:

  sed [**-n**] { [**-e**] *cmd* | **-f** *script* } [*file*]
- příkazy analogické jako v **ed**u
- oddělují se středníkem nebo koncem řádky
- provádějí se v pořadí zápisu
- příkaz nesmí končit mezerou
- příklad:

```
hostname | sed 's/\..*//'
```

Formát příkazu, adresa řádku (**sed**)

- Syntaxe příkazů:

  [*adresa*[**,***adresa*]] *příkaz*[*parametry*]

- Neexistuje institut aktuální řádky, pokud není adresa uvedena, příkaz platí pro každou řádku
- Formáty zápisu adresy a jejich význam:

n	řádka s číslem *n* (číslováno od 1)
$	poslední řádka
/pat/	každá řádka obsahující vzorek

- Doplněk adresního rozsahu: *adresa* **!** *příkaz*...
- Složený příkaz: *adresa* **{**

  příkazy...

  }
- Komentář: *# komentář*...

Příkazy editoru **sed** (I)

- příkazy **ed**u:
  - **p**, **d**, **s**, **w**, **q**
  - **a**, **c**, **i**

    příkaz i nové řádky kromě poslední se ukončují „\“:

```
sed '3a\
ctvrta\
pata'
```

- parametry příkazu **substitute**
  - **p** ... řádka se po modifikaci vypíše na výstup
  - **w** *file* ... řádka se po modifikaci vypíše do souboru
- konverze znaků
  - **y/***intable***/***outtable***/**

    funkce analogická příkazu **tr**

Příkazy editoru **sed** (II)

- řízení toku
  - **n**(ext) … konec práce s řádkou, načtení další řádky
  - `:`*label* … definice návěští
  - **b**(ranch)[*label*] … skok na návěští (na konec)
  - **t**(est) [*label*] … podmíněný skok

    (skočí, pokud od posledního načtení řádky nebo vykonání příkazu `test` byla provedena substituce)

    př.:

```
:loop
s:/\./:/:g
t loop
```

    ... vypustí z cesty všechny sekvence „`/./`“

Příkazy editoru **sed** (III)

- více řádek v pracovním prostoru (oddělovač: **\n**)
  - **N**(ext) … připojení další řádky ze vstupu
  - **P**(rint) … tisk první řádky z prostoru
  - **D**(elete) … vymazání první řádky z prostoru

- odkládací prostor (*hold space*)
  - **h**, **H**(old) … kopie (append) do odkládacího prostoru
  - **g**, **G**(et) … kopie (append) do pracovního prostoru
  - **x**(change) … záměna obsahu prostorů

Příklady použití příkazu **sed** (I)

- **`sed  /record/,/end/d  program.pas`**

  vypíše program bez definic rekordů

- **`sed '/procedure/i\`**
  `{ begin of procedure }' program.pas`

  vypíše před procedurami komentář

- **`sed '1p;$p' program.pas`**

  vypíše zduplikovaně první a poslední řádku

- **`sed -n '4,6!p' program.pas`**

  vypíše soubor bez druhých tří řádek

Příklady použití příkazu **sed** (II)

- **`sed 's/:.*//;s/^/User: /' /etc/passwd`**

  výsledek: `User: forst`

- **`ls *.c | sed 's/\(.*\).c/cp -p & \1.bak/'`**

  výsledek: `cp -p test.c test.bak`

- **`echo ab | sed 's/a/b/;s/b/a/'`**

  výsledek: `ab`

  správně: **`y/ab/ba/`**

  nebo: **`s/a/\`**
  `/g;s/b/a/g;s/\n/b/g`

- **`sed 's/.*:\(.*\) \(.*\):.*/\2 \1/' /etc/passwd`**

  výsledek: `Burns:/home/frank Frank`

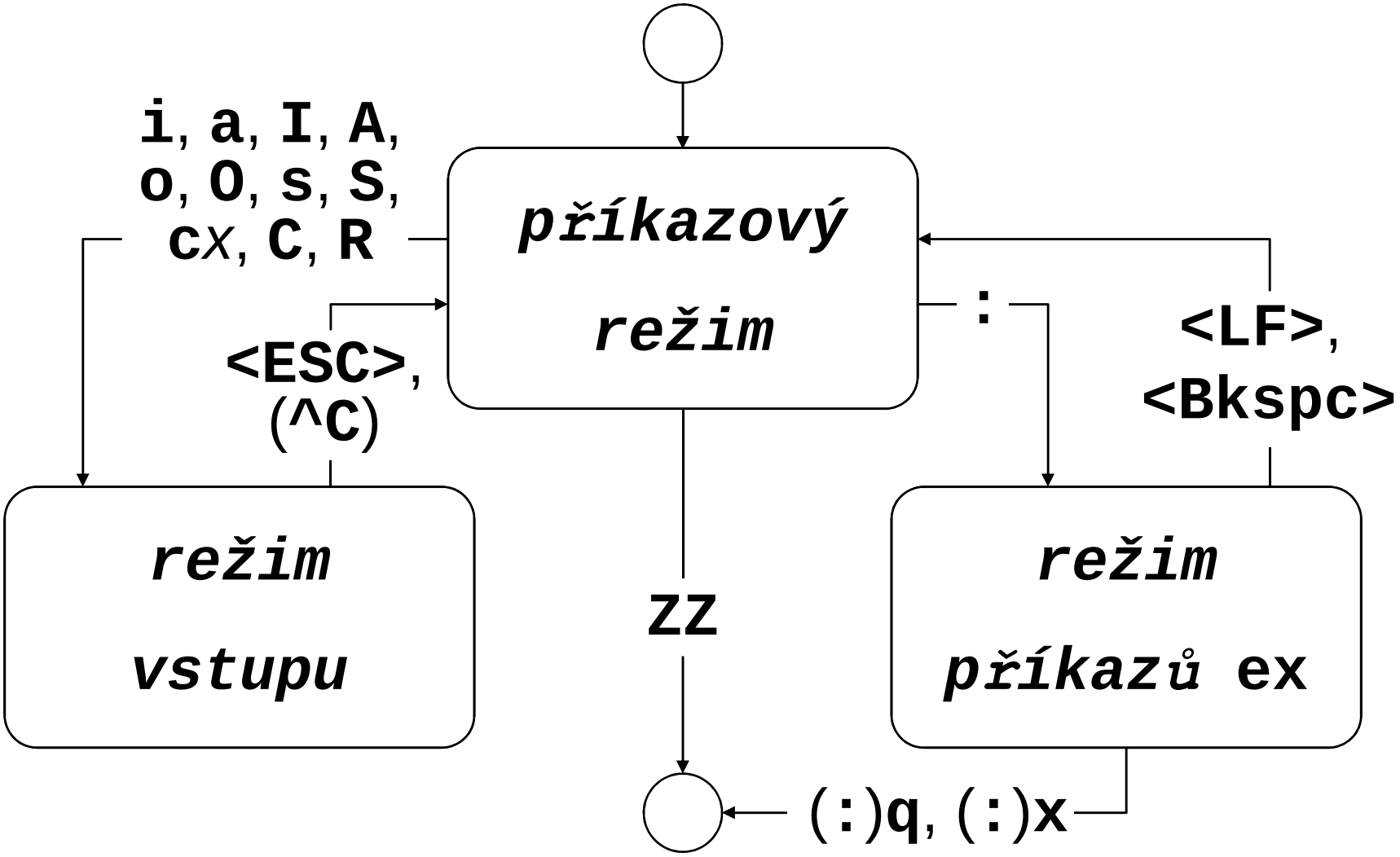
  správně: **`s/.*:\(.*\) \([^:]*\):.*/\2 \1/`**

Editor **`vi`**

- visual editor
- geneze: **`ed`** ⇨ **`ex`** ⇨ **`vi`**
- celoobrazovkový editor
- dostupný na všech UNIXech
- široká paleta příkazů
- malé nezbytné minimum příkazů
- editace kopie souboru
- volání:

  `vi` [**`-rR`**] {**`+`**[*line*] | **`+/`***pattern* } [*files*]

Režimy práce **vi**



Základní příkazy **`vi`**

- **`vi`** *soubor* … vyvolání editoru
- **`i`** … zahájení vkládání textu
- *vkládaný text*
- **`<ESC>`** … ukončení vkládání textu
- **`h`**, **`j`**, **`k`**, **`l`** … pohyb po textu
- */vzorek* … hledání vzorku
- **`x`**, **`dd`** … mazání znaku, řádky
- **`A`** … vkládání na konec řádky
- **`J`** … spojení řádek
- **`ZZ`**, **`:x`** … ukončení editace
- **`:q!`** … zrušení editace

Příkazy pro pohyb (I)

Před příkazy může předcházet opakovací faktor *k*

- **h** (**<BKSPC>**), **j**, **k**, **l** (**<SPACE>**) … o *k* míst (⇦, ⇩, ⇧, ⇨)
- **w**, **b**, **e**, **W**, **B**, **E** … o *k* slov (vpřed, vzad, na konec resp. s ignorováním interpunkce)
- **(**, **)**, **{**, **[[** … na začátek (následující) věty, §, sekce
- **+** (**<LF>**), **-** … začátek následující (předchozí) řádky
- **$**, **0**, **^** … konec řádky, začátek, první nemezerový znak
- **f***x*, **F***x*, **t***x*, **T***x*, **;**, **,** … znak *x* na řádce (dopředu, dozadu), znak před *x*, znak za *x*, opakuj, opakuj v opačném směru
- */regexp*, **?***regexp*, **/**, **?**, **n**, **N** … hledání vzoru dopředu, dozadu, opakuj vzor, opakuj hledání, opakuj obráceně
- **^F**, **^B**, **^D**, **^U** … stránka dopředu, dozadu, půl stránky

Příkazy pro pohyb (II)

Příkazy předchází absolutní hodnota *k*:

- *k*`|` … *k*-tá pozice na řádce
- [*k*]`H` … posun na *k*-tou řádku na obrazovce [1]
- [*k*]`L` … posun na *k*-tou řádku od konce obrazovky [1]
- `M` … posun na prostřední řádku na obrazovce
- [*k*]`G` … posun na *k*-tou řádku souboru [poslední]

Práce se značkou *x* (malé písmeno):

- `'`*x* … posun na pozici označenou značkou *x*
- `''` … posun na poslední označenou pozici
- `'`*x* … posun na začátek řádky se značkou *x*
- `''` … posun na začátek naposledy označené řádky

(označení se provede příkazem `m`*x*)

Vkládání textu, opravy

Před příkazy může předcházet opakovací faktor *k*

- **i**, **a**, **I**, **A** … vkládání před (za) kurzor, řádku
- **o**, **O** … vkládání do nové řádky pod (nad) aktuální (open)
- **~** … změna (malé/velké) písmena pod kurzorem *
- **r***x* … přepis znaku pod kurzorem znakem *x* *
- **R** … zahájení režimu vstupu v přepisovacím módu
- **c***m* … náhrada textu od kurzoru do pozice dané příkazem pro pohyb *m*
- **cc**, **C** … náhrada celé řádky resp. do konce řádky
- **s**, **S** … smaž znak (řádku) a přejdi do režimu vstupu

Příkazy označené * nepřepínají do režimu vstupu.

Mazání, práce s buffery

Před příkazy může předcházet opakovací faktor *k*

- **x**, **X** ... mazání znaku pod (před) kurzorem
- **d***m* … mazání textu od kurzoru do pozice dané příkazem pro pohyb *m*
- **dd**, **D** … mazání celé řádky resp. do konce řádky

Smazaný text se uloží do očíslovaného bufferu.

- **p**, **P** … vložení bufferu za (před) kurzor (příp. řádku)
- ***"n*p**, ***"n*P** … vložení *n*-tého posledního bufferu
- ***"x*p**, ***"x*P** … vložení bufferu *x* (*x* je malé písmeno)

Vložení textu do (pojmenovaného) bufferu:

- [***"x***]**y***m* … vložení textu po pozici danou příkazem *m*
- [***"x***]**yy**, [***"x***]**Y** … vložení řádky

Další příkazy **vi**

- **.** … opakování posledního editačního příkazu
- **u** … zrušení efektu posledního editačního příkazu
- **U** … obnovení řádky do původního stavu
- **J** … slepení řádky s následující
- **%** … skok na odpovídající **)**, **]** nebo **}** (nikoliv **>**)
- **^L** … obnovení obrazovky
- **z<LF>**, **z.**, **z-** … scrollování, aktuální řádka se octne na začátku (uprostřed, na konci) obrazovky
- **^E**, **^Y** … scrollování o řádku
- **^G** … vypsání informace o poloze v editovaném souboru
- **!***m cmd*, **!!***cmd* … použití bloku textu jako vstup a jeho nahrazení výstupem příkazu *cmd*
- **<***m*, **>***m* … indentace
- @*x* … provedení příkazů uložených v bufferu *x*
- **^W**, **^V** … (režim vstupu) smazání slova, vstup řídícího znaku

ex - rozšíření příkazů (I)

- adresy mohou být odděleny středníkem - aktuální se stává první řádka místo poslední
- rozšíření příkazu **`substitute`**
  - parametr **`c`** … nahrazování s potvrzováním (**`y<LF>`**)
  - metaznak **`~`** v regexpu … předchozí výraz
  - sekvence **`\<`** a **`\>`** v regexpu … začátek a konec slova
  - sekvence **`\u`**, **`\l`**, **`\U`** a **`\L`** v řetězci náhrady
    ... převod malá/velká (platí na celé slovo)
- nové příkazy
  - **`co`** (kopíruj, alias příkazu **`t`**)
  - **`j`**(oin)[**`!`**] … spojení řádek, po **`.`** přidává dvě mezery, po **`)`** žádnou, jinak jednu (**`!`** ... bez mezer)
  - **`ya`**(nk)[*x*], **`pu`**(t)[*x*] … práce s (pojmenovanými) buffery

ex - rozšíření příkazů (II)

- **sh**, **!***cmd* … spuštění shellu, příkazu
- **so**(urce) … provedení souboru
- **w!**, **w>>** … zápis do read-only souboru, na konec souboru
- **x**, **wq** … uložení souboru a ukončení editace
- **q!** … ukončení editace bez uložení změn
- **n**[**!**] … editace dalšího souboru (bez uložení změn)
  Pojmenované buffery, poslední regexp a editační příkaz zůstávají zachovány.
- **e**[**!**] [*file*] … editace jiného souboru (**%** je symbol pro aktuální jméno souboru, # pro poslední použité jméno)
- **ab** *word string*, **una** … zkratka
- **map**[**!**] {*char* | *#n*} *string*, **unm** … mapování znaku resp. funkční klávesy (pro režim vstupu); řídící znaky přes **^V**

Nastavení editoru **`vi`**

Nastavování příkazem **`set`**, výpis **`set all`**

- **`autoindent`**, **`ai`** ... odsazování nových řádek [**`noai`**]
- **`directory=`***dir*, **`dir`** ... pracovní adresář [**`=/tmp`**]
- **`ignorecase`**, **`ic`** ... ignorecase při hledání [**`noic`**]
- **`number`**, **`nu`** ... čísla řádek [**`nonu`**]
- **`shell=`***path*, **`sh`** ... cesta k shellu [**`=/bin/sh`**]
- **`showmatch`**, **`sm`** ... hledání závorek [**`nosm`**]
- **`tabstop=`***n*, **`ts`** ... velikost tabelátoru [**`=8`**]
- **`wrapscan`**, **`ws`** ... hledání přes konec souboru [**`ws`**]
- **`wrapmargin=`***n*, **`wm`** ... pravý okraj pro zalamování [**`=0`**]

Předvolby **ex** a **vi**

Před spuštěním editoru se provedou **ex**-příkazy uložené v:

- proměnné **EXINIT**
- domovském adresáři ve scriptu **.exrc**
- aktuálním adresáři ve scriptu **.exrc**

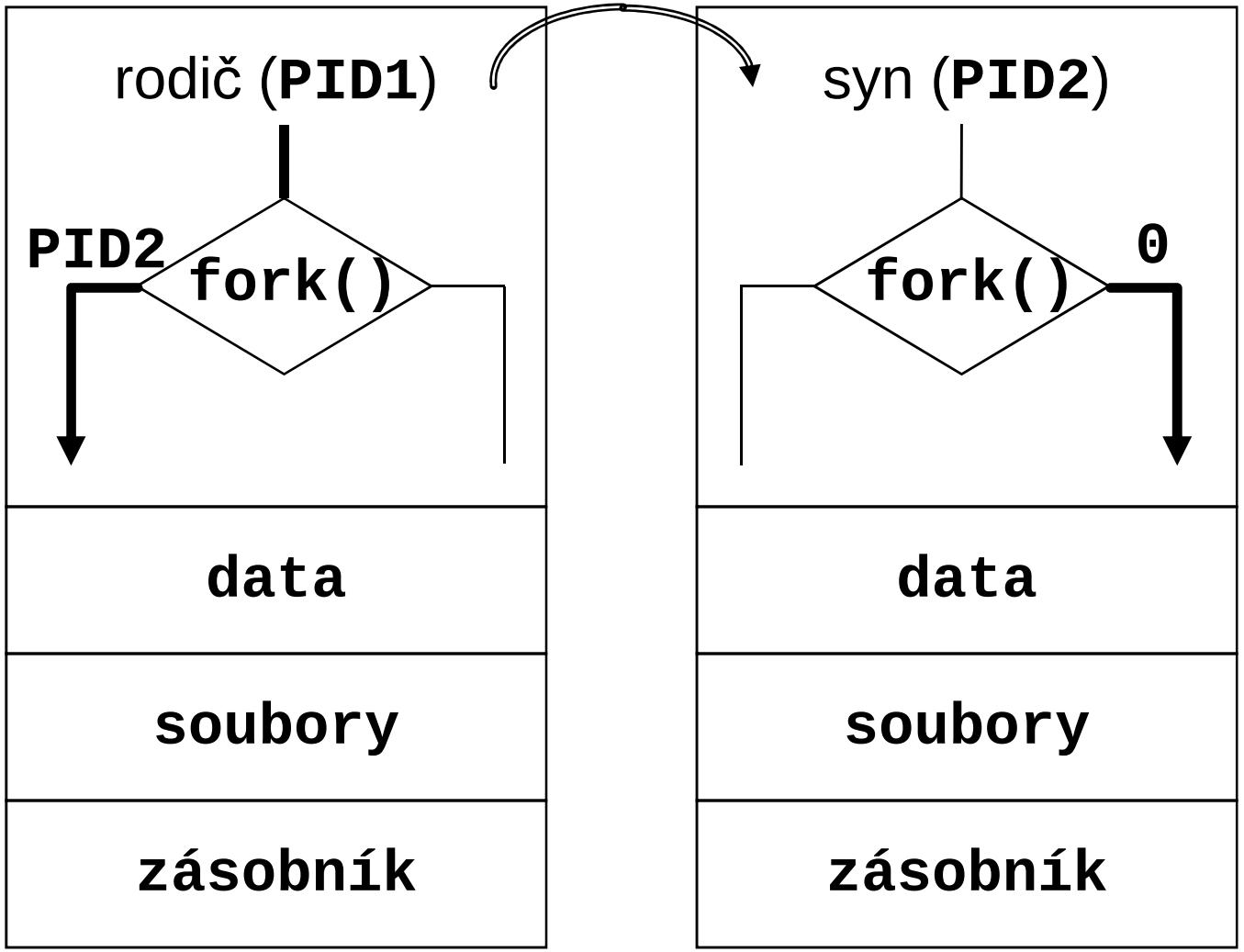
  pokud je nastavena volba **exrc** (implicitně vypnuta)

Příkazy se zapisují bez úvodní dvojtečky (jako v **ex**).

Proces

- prováděný program ... (nejméně jeden) proces
- plánování procesů - priorita
- příkaz **ps**
- PID
- rodičovský proces ⇨ synovský proces
- kontext procesu
  - paměť, soubory, systémové proměnné,...
- komunikace
  - signály, roury, sockety, sdílená paměť,...
- návratová hodnota (0..255)
- běh na popředí, na pozadí, daemon

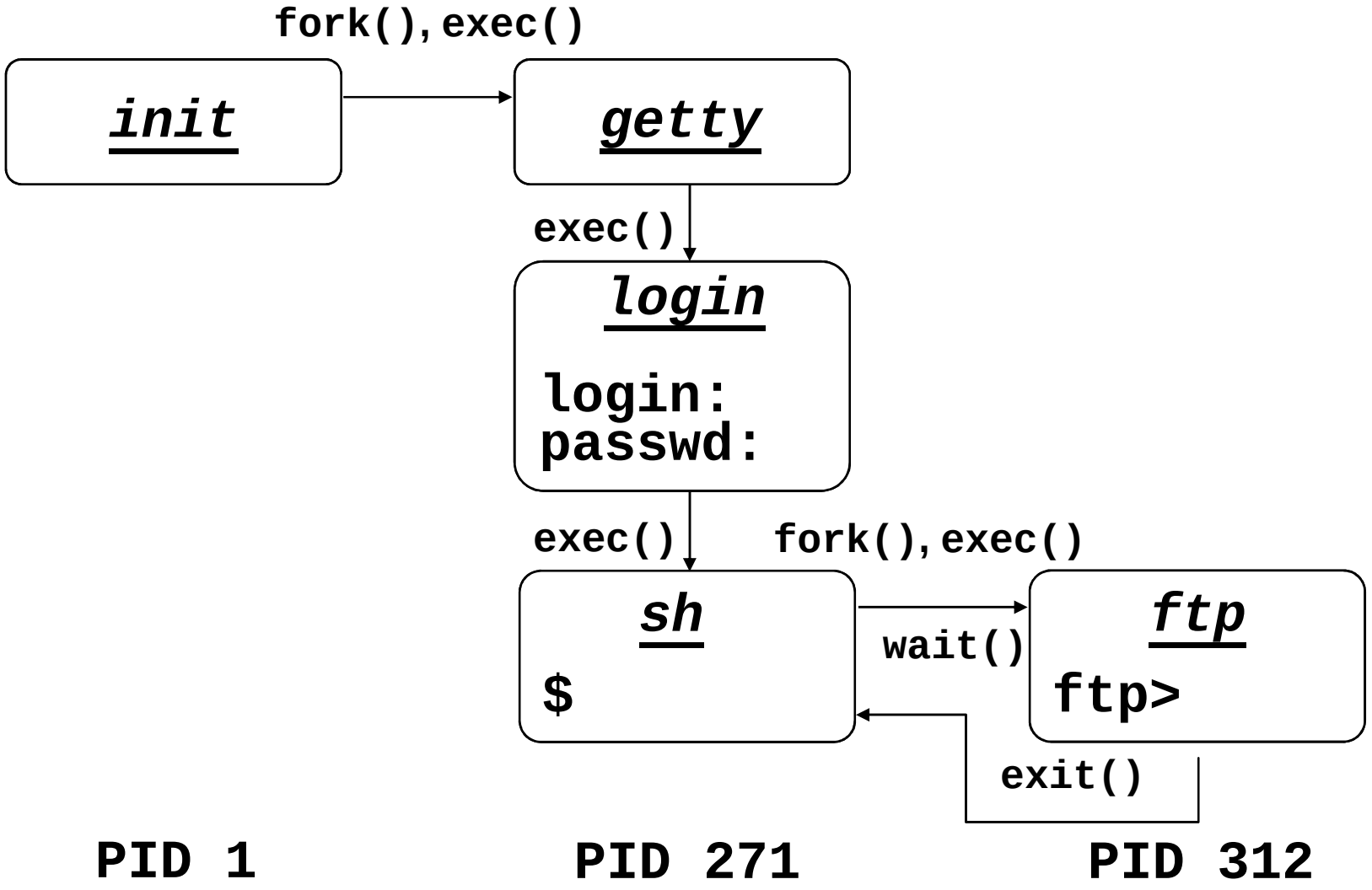
Vznik procesu



Funkce na řízení procesů

- **`fork()`** … vytváří kopii rodičovského procesu; je třeba ošetřit chybu „`Cannot fork`“
- **`exec()`** … překryje adresní prostor procesu zadaným programem
- **`wait()`** … (rodičovský proces) čeká na skončení potomků
- **`exit()`** … ukončí proces a předá rodičovskému procesu návratovou hodnotu

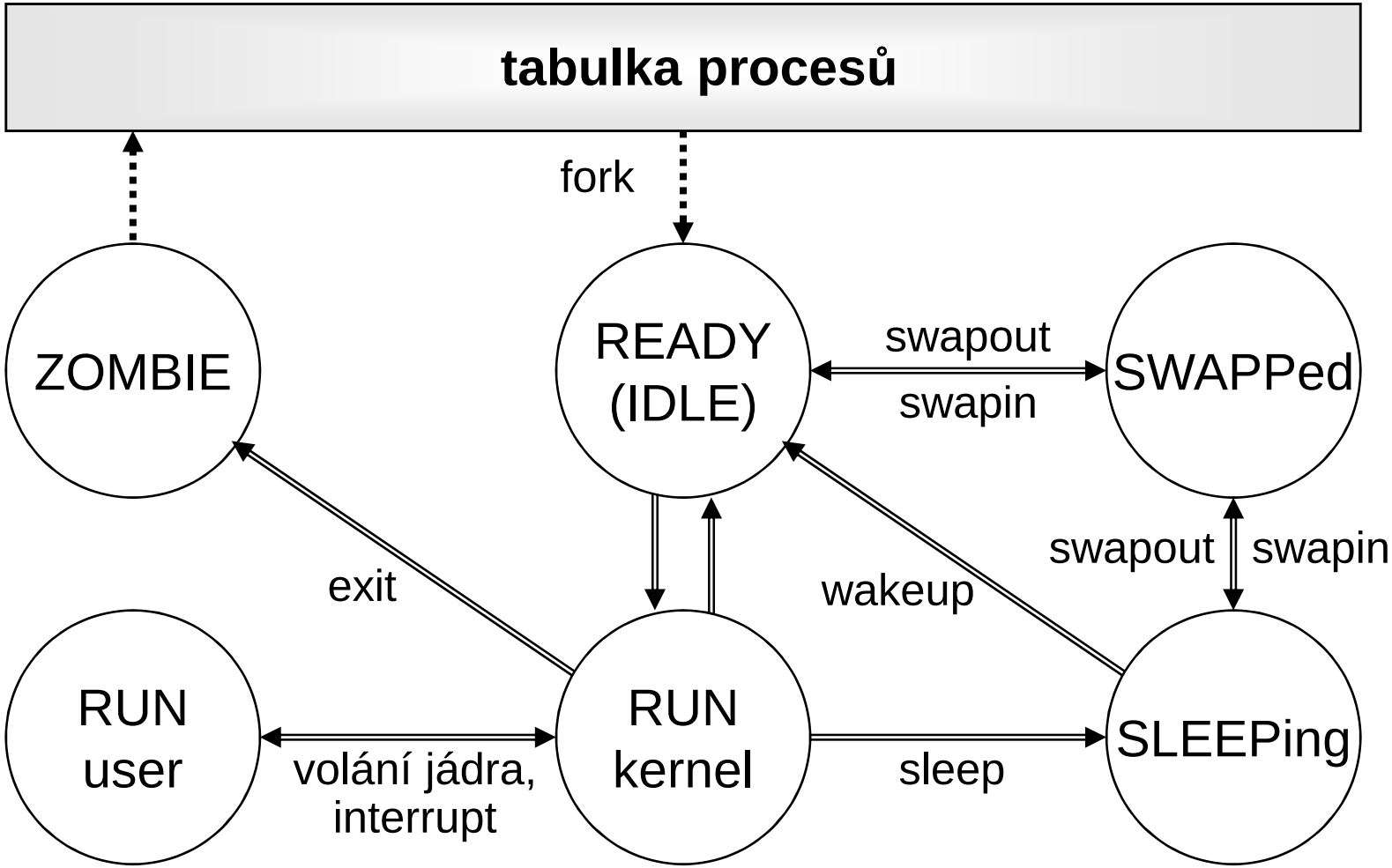
Uživatelská relace



Kontext procesu

- z hlediska uživatele
  - kód, data, zásobník
  - otevřené soubory
  - systémové proměnné (*environment*)
- z hlediska systému
  - obecné registry, programový čítač, stavový registr procesoru, ukazatel do zásobníku, registry pro operace v pohyblivé řádové čárce, registry mapování paměti
  - paměť, kterou proces dosud adresoval v uživatelském režimu
  - paměť v prostoru jádra, která je s daným procesem spojena (např. systémový zásobník procesu)

Stavy procesu



Priorita procesu

- Jeden z faktorů používaných pro plánování procesů
- Kladné číslo (čím vyšší, tím je proces „hodnější“)
- Synovský proces dědí prioritu od otce
- Při startu je možné stanovit jinou prioritu

  `nice -n` *incr cmd*
- Inkrement obvykle povolen v rozsahu -20 až +20
- Pouze root může zadávat záporné hodnoty
- Procesu lze změnit prioritu

  `renice -n` *incr PID...*

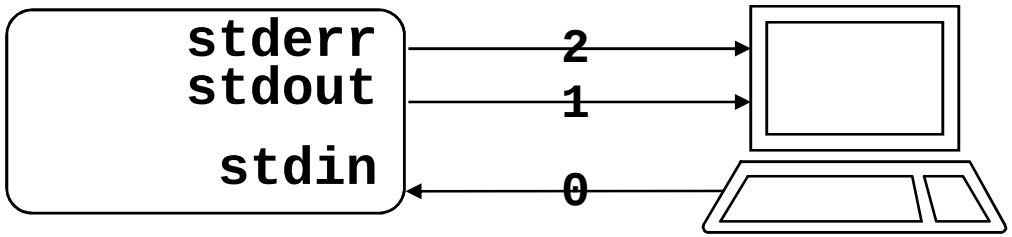
Příkaz **ps**

- **PID**, **TTY**, **STAT**, **TIME** a **COMMAND** vlastních procesů

	System V	BSD	POSIX
• výběr procesů:	**-e** (every)	**-a** (all users) **-x** (no tty)	**-A** (All)
	-p *PIDs* **-t** *ttys* **-U** *users* **-G** *grps*		
• obsah výpisu:	**-l** (long), ...	**-l** (long), ...	
	-o*key,* … (pouze vyjmenované sloupce)		
	-O*key,* … (sloupce navíc)		
• třídění: *(PD program* **top**)	**-r** (cpu) **-m** (mem)		

Proces a I/O

- přístup ke vstupním a výstupním souborům přes tzv. *file-descriptory*
  - 0 - standardní vstup (**`stdin`**)
  - 1 - standardní výstup (**`stdout`**)
  - 2 - standardní chybový výstup (**`stderr`**)
  - … - další otevírané soubory



Komunikace mezi procesy

- zasílání signálů
  - asynchronní řízení
  - informace typu: nastala událost *N*
- vstup/výstup přes roury
- System V Interprocess Communication
  - semafory
  - zasílání zpráv
  - sdílená paměť
- BSD Sockets
  - zasílání zpráv, vytváření proudů
  - v rámci jednoho systému (typ souboru **s**) nebo mezi klientem a servrem po síti

Obsluha signálů

- zaslání signálu:
  - příkaz **`kill`** [**`-`***signal*] *PID*
  - funkce **`kill`**

- ošetření signálu:
  - příkaz **`trap`** [*command*] *signal ...*
  - funkce **`signal`**, **`sigaction`**
  - standardní handlery: **`SIG_IGN`**, **`SIG_DFL`**, **`SIG_ERR`**
  - nemaskovatelné signály: **`KILL`**, **`STOP`**

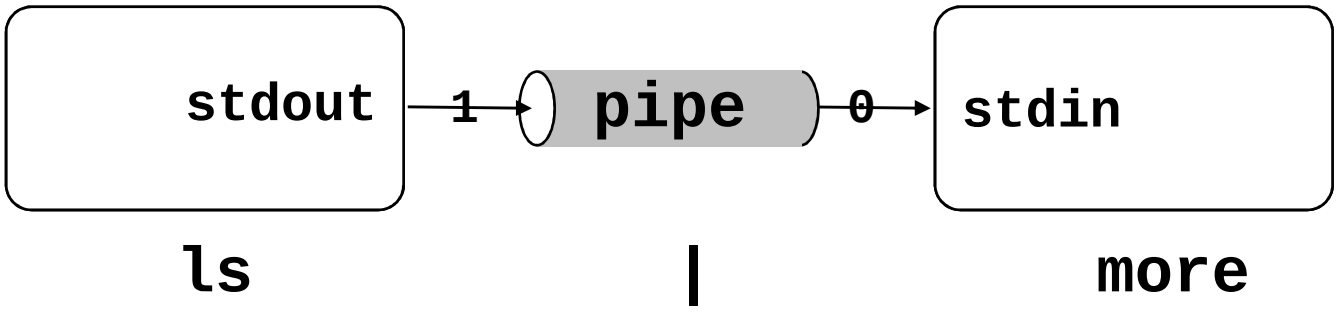
- výpis signálů: **`kill -l`**

Nejdůležitější signály

HUP(**1**)		restart programu
INT(**2**), **QUIT**(**3**)		přerušení uživatelem (**^C**, **^**)
ILL(**4**)		chybná instrukce
ABRT(**6**)		volání funkce **abort**
FPE(**8**)		aritmetická chyba
KILL(**9**)	(nemaskovatelné)	ukončení procesu
SEGV(**11**)		chyba adresace
SYS(**12**)		chybné volání systému
ALRM(**14**)		přerušení od časovače
TERM(**15**)	(maskovatelné)	ukončení procesu (**kill**)
STOP(**17**), **TSTP**(**18**), **CONT**(**19**)		zastavení a spuštění procesu
CHLD(**20**)		ukončení syna
USR1(**30**), **USR2**(**31**)		uživatelské signály

Roury (*pipes*)

- v shellu - spojení vstupu a výstupu dvou procesů



- v programu:
  - roura s externím příkazem: **`popen`**, **`pclose`**
  - roura mezi (sub)procesy: **`pipe`**
- trvalé (pojmenované) roury
  - začleněny do systému souborů, typ **`p`**
  - vytvářejí funkce/příkazy **`mknod`** resp. **`mkfifo`**

System V IPC

- Každé instanci prostředku je přiděleno ID
- Semafory:
  - zobecnění *P* a *V* operací [Dijkstra, Dekker]
  - ošetření *dead-lock*u, havárie procesu
  - funkce: **`semget`**, **`semop`**, **`semctl`**
- Zasílání zpráv:
  - systém vytvoří komunikační kanál
  - funkce: **`msgget`**, **`msgsnd`**, **`msgrcv`**, **`msgctl`**
- Sdílená paměť:
  - systém přidá procesu do tabulky žádanou oblast
  - funkce: **`shmget`**, **`shmat`**, **`shmdt`**, **`shmctl`**

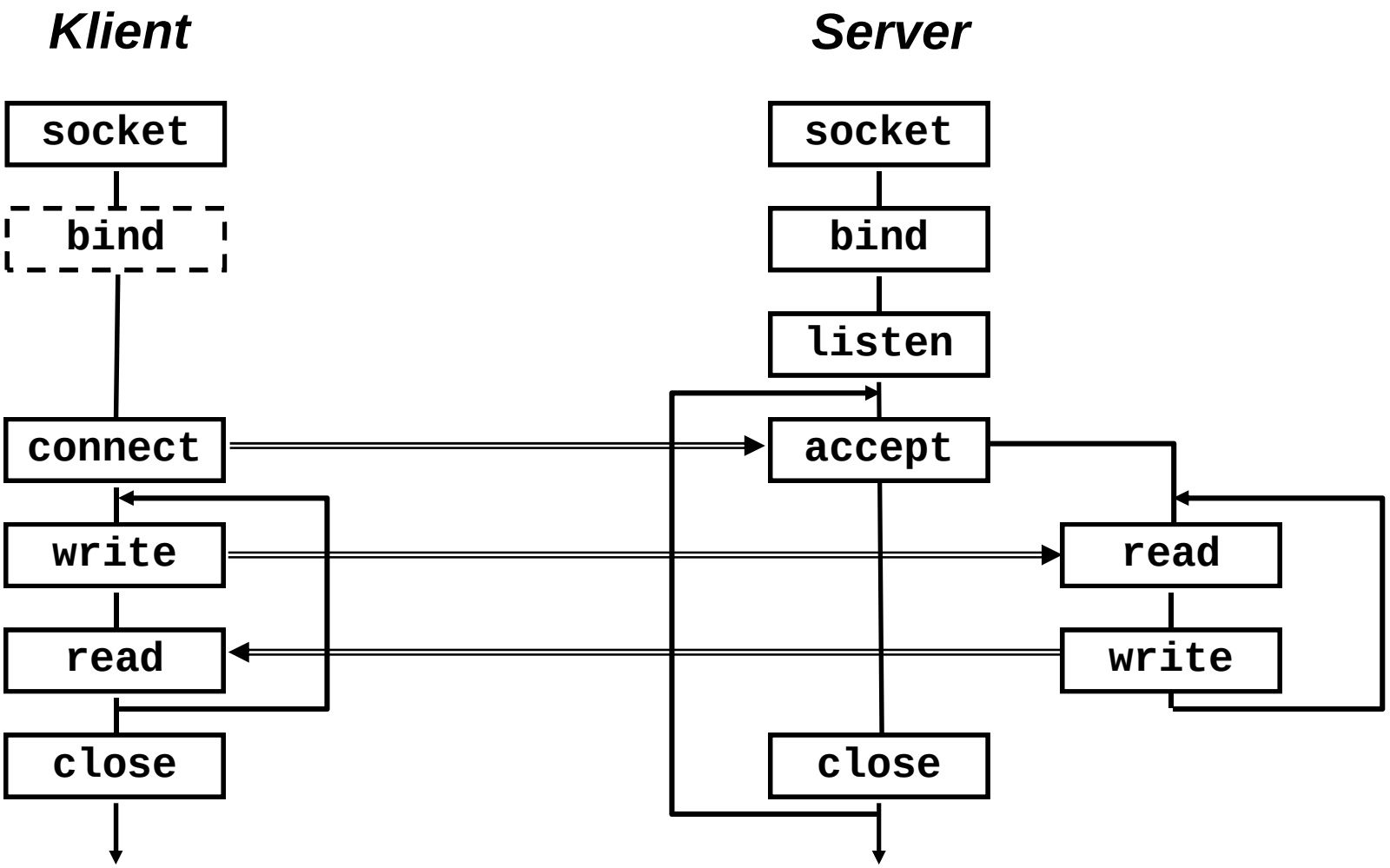
BSD Sockets

Socket - jeden konec kanálu pro klient-server komunikaci

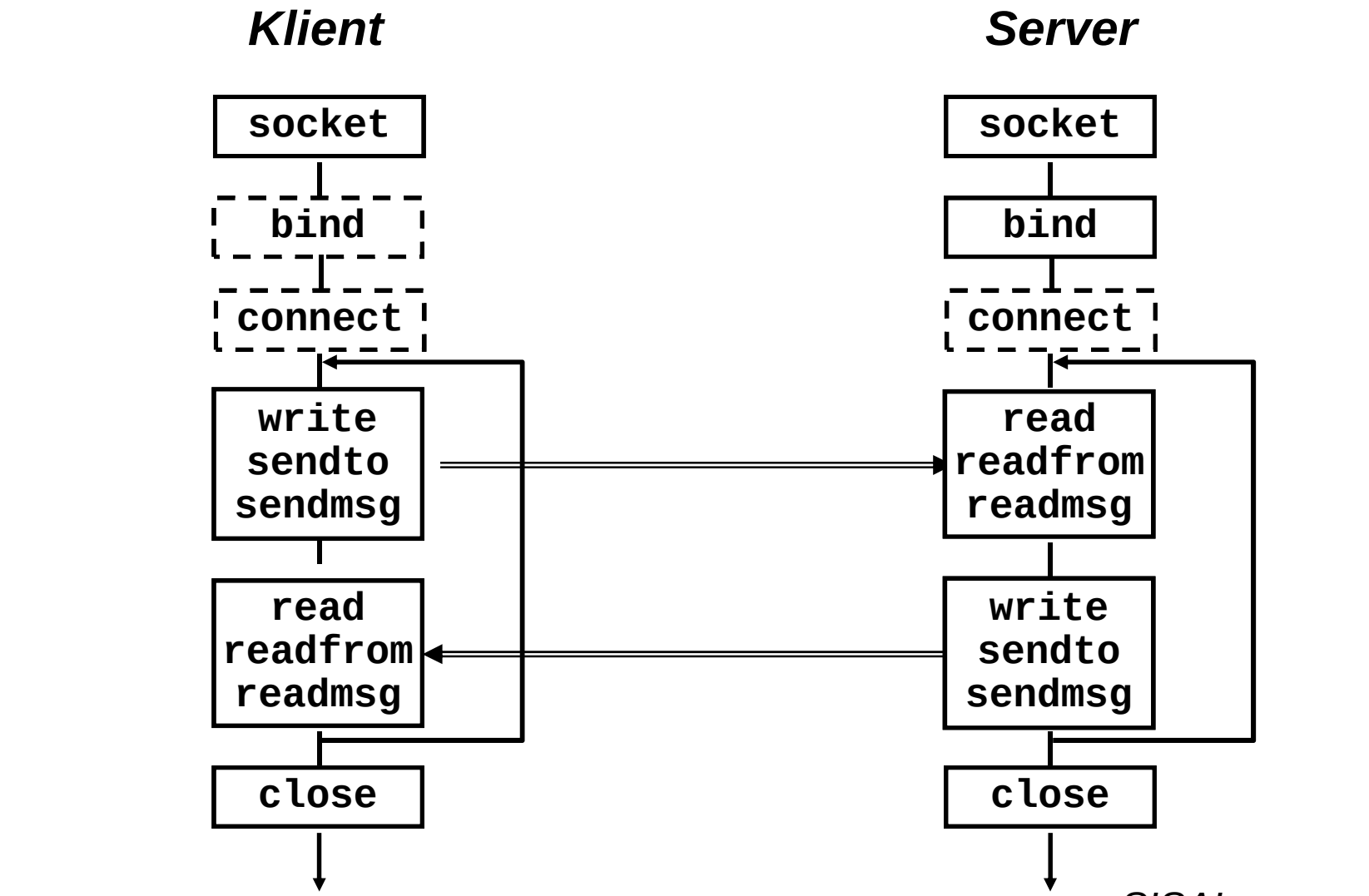
Systémové funkce:

- **`socket`** vytváří deskriptor podle
  - domény (*address family*): **`AF_UNIX`**, **`AF_ INET`**
  - typu: virtuální okruh (*stream*), *datagram*
- **`bind`** přiřazuje vlastní adresu:
  - UNIX: jméno v souborovém systému (typ **`s`**)
  - INET: IP adresa + port
- **`listen`** zahájí příjem zpráv (mj. stanoví délku fronty)
- **`accept`** (server) přijímá požadavek na kanál od klienta
- **`connect`** (klient) navazuje spojení se servrem

Model TCP aplikace



Model UDP aplikace



Start síťových daemonů

- **přímý start**
  - ve startovacích skriptech
  - intenzivně využívané služby, se složitou inicializací

- **nepřímý start** (on demand)
  - provádí daemon **`inetd`**
  - konfigurace v **`/etc/inetd.conf`**:

```
bootps dgram  udp wait   root   /etc/bootpd bootpd
tftp   dgram  udp wait   nobody /etc/tftpd  tftpd /tftpboot
whois  stream tcp nowait nobody /etc/whoisd whoisd
```

  - překonfigurování: **`kill -HUP`** *PID*
  - server komunikuje přes filedeskriptory 0/1

Terminál

- uživatel využívá služeb systému prostřednictvím *terminálu* - buďto skutečného nebo *pseudoterminálu*
- vlastnosti v **`/etc/termcap`** resp. **`/etc/terminfo`**
- typ terminálu v proměnné **`TERM`**
- (re)inicializace terminálu příkazem **`tset`**
- změna vlastností příkazem **`stty`**
  (např. **`stty erase`** *char*)
- přístup na vlastní terminál přes zařízení **`/dev/tty`**

Řídící znaky

- některé lze předefinovat, některé závisí na shellu
  ⇨ nutný soulad terminálu a nastavení **`TERM`**
- typické sekvence:

  Ctrl+H - backspace

  Ctrl+S - pozastavení výpisu

  Ctrl+Q - pokračování výpisu

  Ctrl+C - ukončení procesu (**`SIGINT`**)

  Ctrl+\ - dtto s dumpem (**`SIGQUIT`**)

  Ctrl+D - konec vstupního souboru

  Ctrl+Z - suspendování procesu (**`SIGTSTP`**)
  další spuštění: **`fg`** resp. **`bg`**

Shell

- základní program pro komunikaci s UNIXem
- nezávislá komponenta systému
  - Bourne shell, C shell, Korn shell
- čte řádky a provádí příkazy
  - vlastní příkazy
  - programy uložené v souborovém systému
- textový preprocesor
  - metaznaky
  - proměnné
- programovací jazyk & jeho interpret
  - skripty

Základní vestavěné příkazy shellu

: *comment*	- prázdný příkaz
echo [**-n**] *text*	- výpis textu (s/bez odřádkování)
printf *fmt arg...*	- výpis formátovaného textu
pwd	- výpis cesty k aktuálnímu adresáři
cd [*dir*]	- změna adresáře (vlastnost shellu)
exit [*rc*]	- ukončení shellu s návratovým kódem
set {**+**|**-**}*opt...*	- nastavení přepínačů shellu
ulimit [*limit*]	- nastavení uživatelských limitů
umask [*mask*]	- nastavení defaultního módu souborů

Formátovací direktivy **`printf`**

- Obecný tvar: **%**[*flags*][*width*][*.precision*]*type*
  - **%c** ... výpis jednoho znaku
  - **%s** ... výpis řetězce
  - **%u**, **%d**, **%o**, **%x** ... výpis čísla (unsign., dek., okt., hex.)
  - **%e**, **%f**, **%g** ... výpis reálného čísla
  - **%%** ... výpis procenta
- Modifikátory:
  - **%**[**-**] *width* [**.***len*] **s** ... zarovnání vlevo, max. délka
  - **%**[**+**][**0**]*width fmt-spec* ... vynucení znaménka, ved. nul
  - **%***width* [**.***precision*] *fmt-spec* ... přesnost reálných čísel
- Prakticky identické formátovací direktivy se používají i pro příkaz v **`awk`** a funkci v jazyce C

Metaznaky

- znaky se speciálním významem (např. *****, **>**)
- speciální význam se ruší „quotingem“:
  - zapsáním „\“ před metaznak (tzv. *escape-sekvence*)
  - uzavřením do apostrofů (ruší význam všech metaznaků)
  - uzavřením do uvozovek (neruší význam **$**, `` ` ``, **"** a ****)
- platí i pro speciální význam znaků:

  <LF> ... namísto odeslání příkazu jen pokračovací řádka

  mezera ... několik slov jako jeden parametr
- pozor zvláště u složitějších příkazů (např. `sed "s/ [0-9]*/ #/" ...`)

- komentář: … *#komentář*

Expanzní znaky

Řetězec expanzních znaků se nahradí seznamem všech jmen souborů, které mu vyhovují.

`*`	- zastupuje libovolnou posloupnost znaků
`?`	- zastupuje libovolný znak
`[a-f0-9]`	- zastupuje znak ze seznamu
`[!a-z]`	- zastupuje znak z doplňku seznamu

Bílé znaky se do seznamu zapisují uvozené znakem `\`.
Pro znaky **`!`**, **`]`**, **`-`** platí stejná pravidla jako u regexpů.

Expanzi provádí shell !
Expanze nezahrnuje úvodní tečku ve jméně souboru, nepřekračuje hranici adresáře.

Proměnné v shellu

name=*value*	- nastavení hodnoty
name=*value cmd*	- nastavení pouze pro příkaz *cmd*
$*name*, **${***name***}**	- použití hodnoty (textová substituce)
${#*name***}**	- substituce délky hodnoty

Identifikátor - alfanumerické znaky, case senzitivní.

Proměnné mají pouze textovou hodnotu.

Substituce nenastavené proměnné - prázdný řetězec.

Výpis hodnoty proměnné: **`set`**, **`echo "$`***name***`"`**

Proměnné: lokální vs. *environmentové*.

Synovskému procesu (subshell, roura) se předávají jen *exportované* proměnné (příkazem **`export`** *variable*).

Syn nemůže modifikovat proměnné otce!

Environmentové proměnné

IFS - oddělovač polí (Internal Field Separator), implicitně: IFS=<mezera><tab><LF>

PS1, **PS2** - prompt, prompt na pokračovací řádce

PATH - cesta: adresáře se spustitelnými soubory (aktuální adresář není implicitní!)

CDPATH - cesta pro příkaz **cd**

TERM - typ terminálu

SHELL - prováděný shell

LOGNAME - jméno uživatele

HOME - domovský adresář

MAIL - soubor s poštou

Podmíněná substituce proměnných

zápis	hodnota, je-li proměnná *name*	
	definována	nedefinována
${*name***:-***value***}**	**$***name*	*value*
${*name***:=***value***}**	**$***name*	*value* +nastavení *name*=*value*
${*name***:+***value***}**	*value*	“”
${*name***:?***value***}**	**$***name*	“” +**echo** *value* a **exit**

Příkazové soubory - skripty

- „přímé“ volání (práva **`+rx`**):

  script params

- volání přes shell (práva **`+r`**):

  sh [*options*] *script params*

- vložené volání (běží ve stejném procesu shellu, nikoliv jako nová instance):

  . *script*

- první řádek může obsahovat interpret a optiony:

  #**!***abs_path_to_interpreter* [*options*]

- startovací skripty (spouštějí se jako vložené volání):

  `/etc/profile`, **`.profile`**

Poziční parametry, speciální proměnné

$*n* - *n*-tý parametr (skriptu), *n* <= 9

$# - počet parametrů (skriptu)

$0 - název skriptu

shift [*n*] - posun pozičních parametrů (`$2` ⇒ `$1`)

set [**--**] *text* - nastavení nových pozičních parametrů

př.: `set -- a + b` ⇒ `$1=a, $2=+, $3=b, $#=3`

`IFS=:; set $PATH` ⇒ `$1=/bin, ...`

$* - všechny poziční parametry jako text

$@** - dtto, ale "**$@" je "**$1**" "**$2**" **...**

$? - návratový kód posledního příkazu

$$ - PID běžícího shellu

$! - PID posledního procesu na pozadí

Přesměrování vstupu příkazu

zápis	přesměrování vstupu příkazu ...

cmd **<** *file*	... ze souboru *file*
cmd **<<** *str*	... ze vstupu shellu (textu shellscriptu); vstup se chová jako text v uvozovkách př.: `ed xxx << END` `${cislo_radky}d` ← *here document* `END`

cmd **<<** `\`*str*	dtto; text se chová jako v apostrofech
	př.: `ed xxx << \END`
	`1,$d`
	`END`

cmd **<<-** *str*	dtto; text je možno odsazovat
	př.: `ed xxx <<- END`
	`1,$d`
	`END`

Přesměrování výstupu příkazu

zápis	přesměrování ...
cmd **>** *file*	standardního výstupu do souboru *file*
cmd **2>** *file*	chybového výstupu do souboru *file* př.: `rm xxx 2> /dev/null`
cmd **>>** *file*	standardního výstupu na konec souboru
cmd **2>>** *file*	chybového výstupu na konec souboru
cmd **2>&1**	chybového výstupu do standardního, pozor na pořadí přesměrování: - `grep xxx file > log 2>&1` oba výstupy do jdou souboru `log` - `grep xxx file 2>&1 > log` výstup do souboru `log`, chyby na výstup

Kombinování příkazů

- *cmd1* **|** [<LF>] *cmd2*
  - roura (*pipe*) mezi příkazy
    př.: `ls -l *.c | wc -l`
- *cmd1*`;` *cmd2*
  - sekvence příkazů
- *cmd1* **||** [<LF>] *cmd2*, *cmd1* **&&** [<LF>] *cmd2*
  - podmíněná sekvence příkazů
    př.: `rm aa && echo Soubor aa smazan`
- **{** *cmd1*`;` *cmd2*`;`**}**
  - skupina příkazů
- **(***cmd1*`;` *cmd2***)**
  - provedení příkazů v podprocesu
    př.: `(cd wrk; rm *)`

Příkaz **`read`**

- Příkaz **`read`** *var* načte řádku ze vstupu do proměnné
- Nastavuje návratový kód (úspěch se dá testovat)
- Pokud má příkaz více argumentů, přiřazuje postupně do jednotlivých proměnných pole vstupní řádky (do poslední proměnné zbytek); oddělovače polí udává proměnná **`IFS`**; sousední bílé znaky se slučují; pro čtení *as-is* lze nastavit **`IFS=''`**
- Znak \ ve vstupu se interpretuje jako quoting (zruší funkci oddělovače polí, ale i konce řádky!); dá se potlačit přepínačem **`-r`**
- Při spuštění z příkazové řádky čte z terminálu, ale lze jej přesměrovat (**`read`** *var* **`<`** *file*), naopak lze vynutit čtení z terminálu (**`read`** *var* **`</dev/tty`**)

Příklady použití **`read`**

- `echo -n "Napiš číslo: "; read x`
  ... přečte odpověď

- `IFS=: read user x x x name x < /etc/passwd`
  ... načte login a jméno (prvního) uživatele

- `LHOST=ss1000.ms.mff.cuni.cz`
  `echo $LHOST | cut -f1 -d. | read SHOST`
  ... neudělá nic (SHOST se nastaví v „synovi“)

- `echo $LHOST | cut -f1 -d. > /tmp/x.$$ read SHOST < /tmp/x.$$`
  `rm /tmp/x.$$`

Použití výstupu příkazu

...`*cmd*`... - vložení výstupu příkazu *cmd* do textu řádky

- př.: `SHOST=`echo $LHOST | cut -f1 -d.``

  ⇓

  `SHOST=ss1000`

- příkaz běží jako podproces téže instance shellu
- maže se poslední LF
- pozor na vnořené použití
  - nutno „escapovat“ vnitřní apostrofy (a backslashe)
  - řešení: postupně ukládat výsledek do proměnných
  - od **ksh** výše lze použít ...**$(***cmd***)**...
- př.:
```
rm `cat soubory`
vi `grep –l '^\\.\\\\"' man8/*.8`
```

Řídící struktury

```
if příkaz
then  příkazy
[elif  příkaz
 then  příkazy]
[else  příkazy]
fi
```

```
case slovo in
vzor1 | vzor2 )
   příkazy;;
*)
   příkazy;;
esac
```

```
{while|until} příkaz
do
  příkazy
done
```

```
for var [ in text ]
do
  příkazy
done
```

Příkaz **`test`**

- volání: **`test`** *podmínka* nebo **`[`** *podmínka* **`]`**
- v případě pravdivé podmínky vrací **`0`**
- pozor na nenastavené proměnné, mezery apod.:

  správně: **`[ -n = "$x" ]`**

  špatně: **`[ -n = $x ]`**, **`[-n="$x"]`**
- logické operace (mají nepodmíněné vyhodnocování):
  - konjunkce: *cond1* **`-a`** *cond2*
  - disjunkce: *cond1* **`-o`** *cond2*
  - negace: **`!`** *cond*
  - závorky: **`(`** *cond* **`)`**

    pozor - v shellu je nutno zrušit metavýznam

Operátory příkazu **`test`**

`-e` *file*	- soubor *file* existuje
`-f` *file*	- soubor *file* je obyčejný soubor
`-d` *file*	- soubor *file* je adresář
`-L` *file*	- soubor *file* je symbolický link
`-r` *file*	- uživatel má k souboru *file* právo **`r`**
`-w` *file*	- uživatel má k souboru *file* právo **`w`**
`-x` *file*	- uživatel má k souboru *file* právo **`x`**
`-s` *file*	- soubor *file* má nenulovou délku
`-z` *str*	- řetězec *str* je prázdný
`-n` *str*	- řetězec *str* je neprázdný
str1 **`=`** *str2*	- rovnost řetězců (opravdu rovnost: **`$x = a*`** !)
str1 **`!=`** *str2*	- nerovnost řetězců
int1 **`-eq`** *int2*	- rovnost čísel (též **`-ne`**, **`-lt`**, **`-le`**, **`-gt`**, **`-ge`**)

Příkaz **`expr`**

- volání: **`expr`** *opndA op opndB ...*
- vypíše výsledek a vrací návratovou hodnotu
- shell nemá sám implementovanou aritmetiku!
- logické operátory: **`=`**, **`<`**, **`>`**, **`<=`**, **`>=`**, **`!=`**
- aritmetické operátory: **`+`**, **`-`**, **`*`**, **`/`**, **`%`**
- řetězcové operátory (v normě je pouze „**`:`“**):
  - *string* **`:`** *regexp* (automatické ukotvení na začátek!)
  - **`match`** *string regexp*
  - **`substr`** *string pos len*
  - **`length`** *string*
  - **`index`** *string chars*
- pozor na metaznaky

Řídící struktury - `if`

Příklad:

```
if [ -d tmp ]; then
    echo adresar existuje
elif mkdir tmp; then
    echo adresar vytvoren
else
    echo adresar nejde vytvorit
fi
```

Poznámky:

- Testovaným příkazem může být i roura.
- Výsledek příkazu může být znegován: `if !` *cmd*
- Je třeba ošetřit, pokud příkaz něco vypisuje:

```
if echo "$x" | grep ... > /dev/null
```

Řídící struktury - **`case`**

Příklad:

```
case $1 in
-h | -\? ) echo "Navod: …"; exit;;
'' | *[!0-9]* )
    echo "Nebylo zadano cislo"; exit;;
* ) CISLO=$1;;
esac
```

Poznámky:

- V návěštích se používají *wildcardy*, ale bez zvláštního významu tečky a lomítka, shell je neexpanduje.
- Pořadí návěští je významné (někdy pomocí něj lze nahradit negaci nebo regexpy).
- V návěští lze použít (např. i testovanou) proměnnou.

Řídící struktury - **while**, **until**

Příklad:

```
while read line; do
    case $line in
    \#* ) continue;;
    *   ) $line;;
    esac
done < script
```

Příklad:

```
i=1; until mkdir /tmp/$i; do
    i=`expr $i + 1`
done
```

Příklad:

```
while [ $# -gt 0 ]; do
    case $1 in
    -n  ) N=$2; shift 2;;
    -n* ) N=`echo $1 | cut -c3-`; shift;;
    *   ) break;;
    esac
done
```

Řídící struktury - `for`

Příklad:

```
list=MFF,FF,FaF,FTVS
for file in *; do
    case ,$list, in
    *,$file,* ) cp $file ${file}_bak;;
```

Poznámky:

- Cyklus od 1 do *n* (`seq` není v normě):

```
for i in `seq 1 $n`; do
i=1; while [ $i -le $n ]; do i=`expr $i + 1`
i=:; while [ ${#i} -le $n ]; do i=:$i
```

- Cyklus `for` se nehodí pro čtení souboru:

```
for line in `cat soubor`
```

Příklad: čtení vstupního souboru

- ```
  n=0
  while read x < file; do
     n=`expr $n + 1`
  done
  ```
  ... čte stále první řádku
- ```
  n=0
  cat file | while read x; do
     n=`expr $n + 1`
  done
  ```
  ... proměnná **n** se nastaví pouze v synovi
- ```
  n=0
  while read x; do
     n=`expr $n + 1`
  done < file
  ```
  ```

Příklad: čtení výstupu roury

- ```
  n=0
  find ... | ( while read x; do
      printf "Mam smazat $x? (a/[n]) "
      read z
      case $z in
      '' | n* | N* ) continue;;
      esac
      rm $x; n=`expr $n + 1`
    done
    echo Smazano $n souboru
  )
  ```
  ... proměnná **z** se čte také ze souboru
- `read z < /dev/tty`
- `{ … read z <&3 … } 3<&0`
  ```

Funkce

Definice funkce *name*:

*name***(){**

statements

}

- běží ve stejném procesu
- proměnné jsou globální, může je měnit!
- volání + parametry stejné jako při volání příkazu
- funkce přistupuje k parametrům jako **$#**, **$1** atp.
  (jsou lokální, nemění je volajícímu!)
- návratovou hodnotou je návratová hodnota posledního příkazu, lze nastavit: **`return`** *val*
- priorita: funkce, interní příkazy, externí programy
  interní příkaz lze vyvolat pomocí **`command`** *cmd*
- funkce se nedědí do subshellů

Postup zpracování řádky

Postupuje se zleva doprava v následujících krocích:

1. rozdělení řádky na atomy (po operátorech)
2. detekce řídících struktur a operátorů
3. detekce operátorů přesměrování
   a definic proměnných
4. substituce proměnných a vložených příkazů

5*. rozdělení výsledku substitucí na pole podle **`$IFS`**

6*. náhrada expanzních znaků

7. zrušení quotingu metaznaků

*Kroky 5 a 6 se neprovádějí při přiřazení proměnných.

Vícenásobné čtení řádky

`eval` *arg* - příkaz sestaví svoje (už jednou zpracované) argumenty do řádky, znova ji zpracuje a provede výsledný příkaz

- př.: `read login x uid x < /etc/passwd`
  `eval UID$uid=$login`
  ⇓
  `UID0=root`
  - nepřímé proměnné (lze nahradit pole)
- př.: `eval echo \$$#`
  - vypsání hodnoty posledního parametru

Řízení procesů

cmd **`&`** - provedení na pozadí

`wait` - čekání na skončení procesu na pozadí

... počínaje **`csh`** je dokonalejší správa (**`jobs`**,...)

`exec` *cmd* - volání funkce `exec( )` s příkazem *cmd*
(shell se změní v zavolaný program)

... počínaje **`ksh`** lze použít pro přesměrování deskriptorů aktuálního shellu (např. **`exec 3<&0`**)

Ošetření signálů v shellu

- Nastavení handleru: **`trap`** [*cmd*] *sig...*
  - parametr *sig* je číslo/jméno signálu nebo `0`/`EXIT`
  - příkaz *cmd* (*handler*) se provádí v rámci shellu

- Synovský proces nemá možnost ošetřit signály zamaskované otcem.

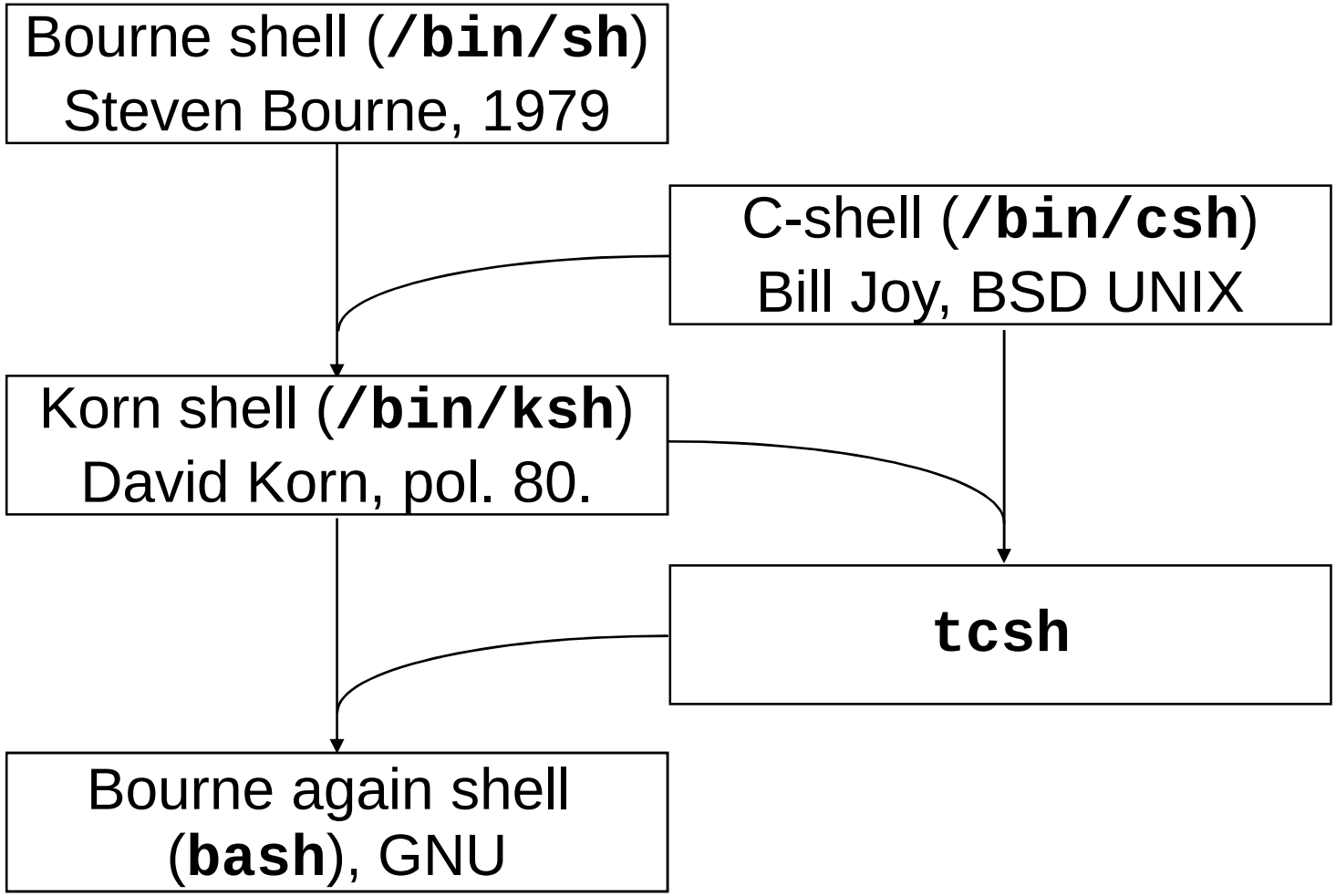
- Zamaskování signálů: **`trap ""`** *sig...*
- Návrat implicitního ošetření: **`trap`** *sig...*

Přepínače shellu

Přepínače se dají zadat
- na příkazové řádce při spuštění shellu
- na první řádce shell-scriptu
- příkazem **`set`**

Nejdůležitější přepínače:
`-a` ... všechny proměnné jsou exportovány
`-C` ... přesměrování nepřepíše existující soubory
`-e` ... chyba v příkazu způsobí ukončení shellu
`-f` ... potlačení expanzních znaků
`-n` ... příkazy jsou pouze vypsány a neprovádí se
`-u` ... expanze nenastavené proměnné je chyba
`-v` ... vstupní řádky se vypisují na chybový výstup
`-x` ... příkazy se před provedením vypisují

Vývoj shellů



C-shell

Zásadní odlišnosti:

- **`.login`**, **`.cshrc`** … startup script
- **`set`** *var*`=`*str*, **`env`**, **`setenv`**, **`@`** *var expr* … proměnné
- **`foreach`**, výrazy a příkazy C
- **`>&`**, **`>>&`**, **`|&`** … přesměrování chybového výstupu
- **`$<`** … přímý vstup z terminálu

Novinky přejaté nebo modifikované:

- **`~`**[*user*] … domovský adresář
- **`<ESC>`** … kompletace jmen souborů
- **`history`**, **`!`**[[**`-`**]*n*], **`!`**[[**`?`**]*str*] … historie příkazů
- **`alias`** *name str* … přejmenovávání příkazů
- **`pushd`**, **`popd`** … příkaz **`cd`** se zásobníkem

Korn shell

- **`cd`** *old new*, **`cd -`** … náhrada v cestě, undo **`cd`**
- **`VISUAL`**, **`set -o`** *ed* … historie s editací řádku
- `\` resp. **`<Esc><Esc>`** … kompletace jmen
- **`FPATH`** … cesta pro funkce
- **`*()`**, **`+()`**, **`?()`**, **`@()`**, **`!()`** … regulární expanzní znaky
- **`${`***var#pat***`}`**, **`${##}`**, **`${%}`**, **`${%%}`** … **`$`***var* zkrácená o min.(max.) řetězec ze zač.(konce) vyhovující vzoru
- **`[[]]`** … interní **`test`** (**`<`**, **`>`**, **`-nt`**, **`-ot`**, **`-O`**, **`-G`**)
- **`let`** *var***`=`***exp*, **`(())`**… aritmetika
- **`${`***v***`[`***e***`]}`**, **`${#`***v***`[*]}`**, *v***`[`***e***`]=`***s***, **`set -A`** *v str* … pole
- **`select`**, **`getopts`**, **`typeset`**

Zpracování přepínačů (**getopts**)

```
while getopts :x:y NAME; do
    case $NAME in
    x )  opt_x=$OPTARG;;
    y )  opt_y=1;;
    \? ) echo "Spatny prepinac $OPTARG";;
    : )  echo "Chybi hodnota $OPTARG";;
    esac
done
shift `expr $OPTIND - 1`
```

Práce s časem

- spuštění programu s měřením času:

  `time` *command*

- pozastavení běhu:

  `sleep` *seconds*

- výpis aktuálního (nebo jiného*) data a času:

  `date` [**`+`***format*]

  Formát (shodný s funkcí **`strftime`**): text s **`%`**-direktivami

  - **`aAbB`** ... krátké/dlouhé jméno dne/měsíce
  - **`dmyYHMS`** ... datum a čas číselně
  - **`uUVjC`** ... číslo dne v týdnu, týdne, dne v roce, století
  - **`cxX`** ... „normální“ tvar data a času
  - **`s`** ... sekundy od počátku „letopočtu“ (1.1.1970) *

Synchronizace

- Pokud dva procesy sdílejí nějaký zdroj, je nutné současný přístup ke *kritickým sekcím* programů ošetřit zámkem.
- Synchronizace přes soubor: program testuje, zda existuje *lock* soubor - pokud ano, je zdroj zamčen, program čeká ve smyčce (`sleep` !), a když zmizí, sám vytvoří nový.
- Test zamčení zámku a jeho nastavení musí být z hlediska operačního systému *nepřerušitelná* dvojice operací, např. vyhovuje `mkdir` nebo přesměrování (>) při zapnutém `-C`.
- Po opuštění kritické sekce se soubor musí smaže, je třeba ošetřit i nepřirozené případy (`trap` !). Pro případ kontroly po havárii je dobré zámek označit číslem procesu.

Neinteraktivní zpracování

- Spuštění příkazu se zablokovaným signálem **`HUP`** a **`QUIT`** a výstupem do **`$HOME/nohup.out`**

  `nohup` *command*

- Spuštění příkazu v určený čas (uživateli musí být povoleno v souborech **`at.allow`** resp. **`at.deny`**, výstup jde uživateli mailem):

  `at` {**`-t`** *mmddHHMM* | *time* [**`+`***incr*] } *command*

  Příkaz umožňuje vypisovat (**`-l`**) a mazat (**`-r`**) joby.

- Pravidelné spouštění pomocí démona **`cron`**:

  `crontab` [**`-l`**]

  Příklad záznamu:

  ```
  0 1 * * 1-2,5 /usr/sbin/backup
  ```

Filtr **awk**

- Aho, Weinberger, Kernighan
- jazyk podobný C s několika rozdíly:
  - znak LF je významový
  - snazší práce s řetězci
  - je interpretovaný
- dialekty: **awk**, **nawk**, **gawk**
- volání:

  awk [*opt*] {**-f** *script* | *pgm*} {*params* | *file* | **-**}...
- filtr zpracovává postupně záznamy (řádky) zadaných souborů a provádí na nich příkazy z awk-skriptu
- př.: `ls -l |`

```
awk '/^-/ { s += $5 } END { print s }'
```

Vzory a akce (**awk**)

- Program (awk-skript) se skládá z větví ve tvaru

  vzor **{** *akce* **}**

- Typy vzorů:

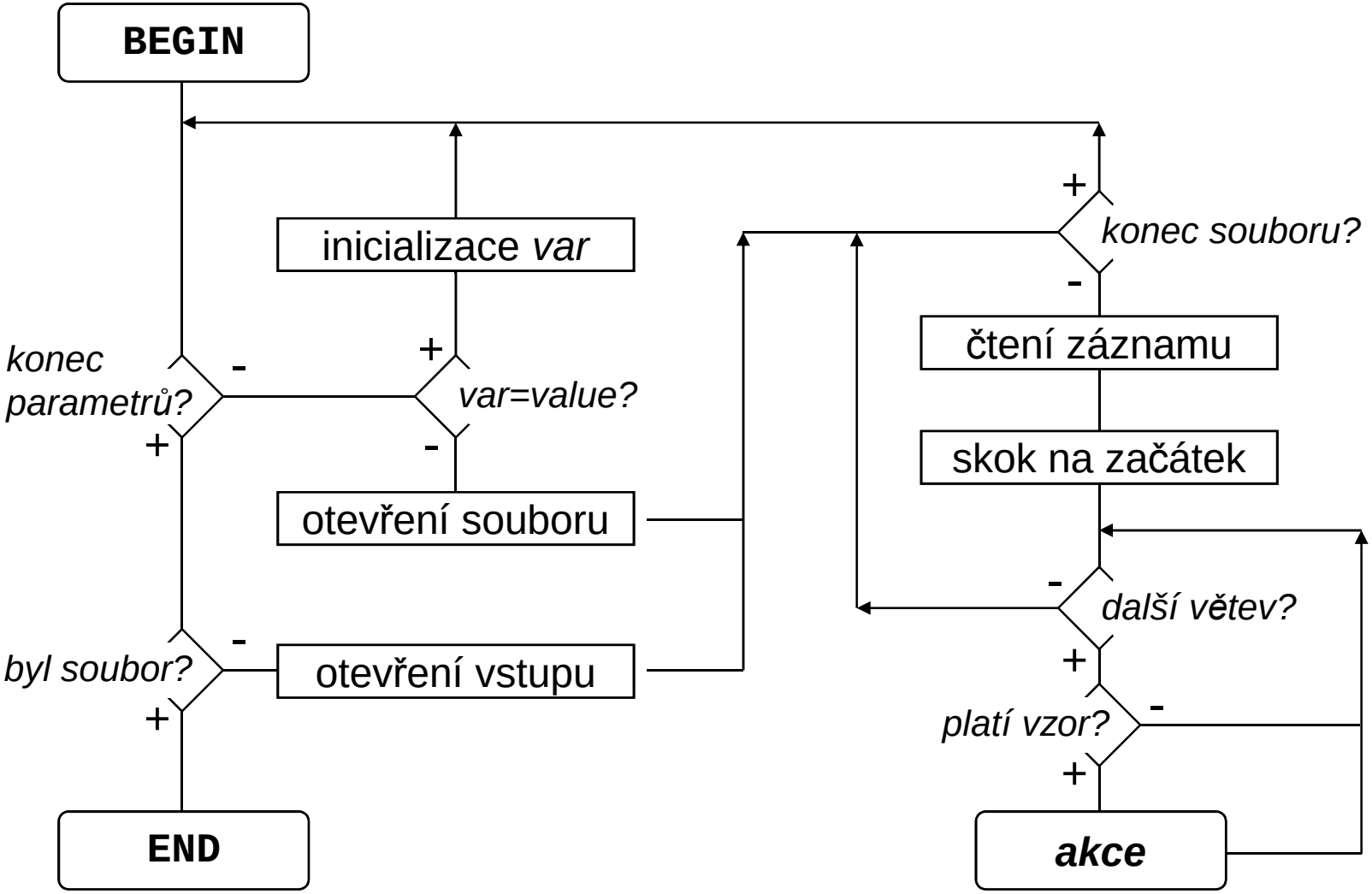
BEGIN	provede se jednou, na začátku práce
END	provede se jednou, na konci práce
/regexp/	provádí se, pokud řádka vyhovuje
expression	provádí se, pokud platí podmínka
vzor1,*vzor2*	začne se provádět, když platí *vzor1*, přestane se provádět, když platí *vzor2*

- Implicitní vzor: proveď vždy
- Implicitní akce: opiš řádku

Příklad programu **awk**

```
BEGIN   { procedur=0; radek=0 }
/procedure/ { procedur++; print;
        radek=1; hloubka=0; next }
! radek { next }
        { radek++ }
/begin/ { hloubka++ }
/end/   { hloubka-- }
/end/ && ! hloubka {
        print "Radek:", radek; radek=0 }
END     { print "Procedur: " procedur }
```

Diagram běhu programu **awk**



Rozšířené regulární výrazy (**awk**)

Nové (a změněné) metaznaky
- *exp***+**, *exp***?** … opakování (>0, <=1)
- *exp1***|***exp2***|***exp3* … varianty
- **(**, **)** … uzávorkování výrazů

Ujasnění významu metaznaků
- **^**, **$** … začátek a konec testovaného řetězce

Oproti základním regulárním výrazům chybějí
- **\<**, **\>**, **\{**, **\}**, **\(**, **\)**, *****n*

Regulární výraz musí být zapsán jako literál (není možné testovat s výrazem uloženým v proměnné)!

Záznamy (**awk**)

- Záznamem je typicky řádka
- Oddělovač záznamů je uložen v proměnné **RS** a je možné ho změnit za jiný znak: **RS="***char***"**
  - např. pro HTML: **RS="<"**
- Oddělovačem může být i prázdná řádka: **RS=""**
- Změna se projeví až u následujícího záznamu
- Číslo záznamu: proměnná **NR**
- Oddělovač záznamů na výstupu (řetězec, který ukončuje příkaz **print**): **ORS=***string*

Pole záznamu (**awk**)

- Vstupní záznam se automaticky rozdělí na pole
- Počet polí: proměnná **`NF`**
- Na jednotlivá pole je možné se odkazovat jako **`$`***číslo*
- Číslo lze zadat jako výraz, např. **`$(NF - 1)`**
- Pozor na rozdíl mezi **`NF`** a **`$NF`** !
- Na celý záznam je možné se odkazovat jako **`$0`**
- Pole záznamu je možné měnit, důsledkem je ztráta přesného tvaru záznamu (zmizí oddělovače)!

Oddělovač polí (**awk**)

- Oddělovač polí je uložen v proměnné **FS**
- Může být inicializován při volání přepínačem **-F***sep*
- Oddělovač může být zadán jako
  - mezera, pak je oddělovačem posloupnost bílých znaků
  - nemezerový znak, pak je oddělovačem každý znak
  - (**nawk**) regulární výraz, např. řádek a==b
    - má tři pole, pokud **FS="="**
    - má dvě pole, pokud **FS="=="** nebo **FS="=+"**
- Změna platí až od dalšího záznamu *
- Oddělovač parametrů příkazu **print**: **OFS=***sep*

Základní syntaxe **awk**

- Jazyk **awk** je **řádkově orientovaný**
- Příkazy se oddělují středníkem nebo koncem řádky, příkaz musí být (až na výjimky) na jedné řádce
- Má-li příkaz pokračovat na další řádce, musí předcházející řádka končit zpětným lomítkem
- Výjimky:
  - za podmínkou **`if`** a **`while`**
  - za čárkou, za otevírací složenou závorkou
  - za operátorem **`&&`** a **`||`**

- Komentář: text na řádce počínaje znakem #

Konstanty, proměnné (**awk**)

- Konstanty
  - běžné aritmetické konstanty
  - řetězce se omezují uvozovkami
  - *escape* sekvence: **\b**, **\f**, **\n**, **\r**, **\t**, *ooo*, **\x***xx*

- Proměnné
  - mají pouze textovou hodnotu
  - v aritmetickém kontextu se text převede na číslo
  - jsou inicializovány
  - asociativní pole (indexem je řetězec): *var***[***item***]**
  - (**nawk**) speciální *member* operátor: *item* **in** *var*

Výrazy (**awk**)

- aritmetické operátory:
  - běžné C-operátory: **+**, **-**, *****, **/**, **%** (modulo)
  - umocnění: **^**
  - přiřazovací operátory, in(de)krement: **=**, **+=**, ... , **++**, **--**
- operátor zřetězení: mezera (!)
  - př.: `"File: " FILENAME " opened"`
- relační a logické operátory (výsledek je **1**/**0**):
  - běžné C-operátory: **<**, **>**, **<=**, **>=**, **==**, **!=**, **!**, **||**, **&&**
  - operátor *match* (shoda s regulárním výrazem zadaným literálem, nikoliv proměnnou) a jeho negace: **~**, **!~**
    např. test, zda 2. pole začíná tečkou: `$2 ~ /^\./`
- (**nawk**) podmíněný výraz: *cond* **?** *then* **:** *else*

Základní příkazy (**awk**)

- **{***cmd1***;***cmd2***}** ... složený příkaz
- **if(***cond***)***cmd* [**;else** *cmd*] ... podmíněný příkaz
- **while(***cond***)***cmd* ... příkaz cyklu
- **do** *cmd***;while(***cond***)**... příkaz cyklu
- **for(***init***;***cond***;***step***)***cmd* ... příkaz cyklu (výraz *step* se vyhodnotí na konci každé iterace)
- **for(***var* **in** *array***)***cmd* ... příkaz cyklu (opakování těla pro každý index, v náhodném pořadí!)
- **break**, **continue** ... konec cyklu, další iterace cyklu
- **next** ... konec zpracování záznamu
- **exit** ... konec programu (skok na větev END)

Výstupní příkazy (**awk**)

- **`print`**
  tisk celého záznamu ukončeného **`ORS`** (default: LF)
- **`print`** *`str1, str2, ...`*
  tisk řetězců oddělených **`OFS`** (mezera), ukončený **`ORS`**
- **`printf`** *`fmt, par1, par2, ...`*
  formátovaný tisk
- **`print,printf >`** *filename*
  výpis do souboru (max. 10 otevřených souborů !)
- **`print,printf >>`** *filename*
  append do souboru

př.:
```
printf "%s::%d:\n",
       grp, gid >> "/etc/group"
```

Knihovní funkce (**awk**)

- matematické funkce: **`int`**, **`exp`**, **`log`**, **`sqrt`**
- (**nawk**): **`sin`**, **`cos`**, **`atan2`**, **`rand`**, **`srand`**
- řetězcové funkce:
  - **`index(`***s*,*t*`)`** ... vrací pozici *t* v *s* nebo **0**
  - **`length(`***s*`)`** ... vrací délku řetězce *s*
  - **`split(`***s*,*var*,*sep*`)`** ... rozdělí *s* na slova oddělená separátorem *sep* a přiřadí je do prvků pole *var*, vrací počet; př.: `split("194.50.16.1",ip,".")`
  - **`sprintf(`***fmt*,*...*`)`** ... vrací formátovaný text jako řetězec
  - **`substr(`***s*,*pos*[,*len*]`)`** ... vrací podřetězec od pozice *pos*
- (**nawk**): **`match`**, **`close`**, **`sub`**, **`gsub`**
- (**gawk**): **`tolower`**, **`toupper`**, **`strftime`**

Vlastní funkce (**nawk**)

- **`function`** *name*`(`*parameter-list*`){`
  statements
  `}`
- **`return`** *expression*

- definují se na úrovni klauzulí
- nezáleží na pořadí
- vlastní „knihovna“ funkcí: **`awk -f`** *lib* **`-f`** *script ...*
- proměnné jsou globální, parametry lokální
- používají se ve výrazech
- parametry není nutno zadávat všechny

Konfigurace programu v **awk**

- Předání parametrů přes **echo**:

  př.:
```
echo $LOW $HIGH | awk '
    NR == 1 { low=$1; high=$2;
       FS=":"; next }
    ...' - /etc/passwd
```

- Expanze hodnot proměnných pomocí shellu:

  př.: `awk /"$RE"/`

- Nastavování proměnných z příkazové řádky:

  př.: `awk var=value1 file1 var=value2 file2`

- Proměnné environmentu (**nawk**): pole **ENVIRON**

  př.: `file = ENVIRON["HOME"] "/log"`

Vestavěné proměnné (**awk**)

- **RS**, **ORS**, **NR**, **FS**, **OFS**, **NF**
- **FILENAME** - jméno právě zpracovávaného souboru

  př.: `FILENAME == "-" { ... }`

- **FNR** - číslo záznamu uvnitř souboru
- **ARGC**, **ARGV** - počet parametrů, pole parametrů
  - sémantika jako v C
  - v seznamu není awk-skript a přepínače

  př.: `{ ARGV[ARGC++] = "soubor" }`
- **SUBSEP** - oddělovač dimenzí v indexu polí
- **RLENGTH** - délka řetězce nalezeného funkcí **match**

Komunikace se systémem v **awk**

- změna proměnné prostředí: nelze !
  - `PATH=‘awk '{print path}'‘`
  - `eval ‘awk '{printf "PATH=%s;HOME=%s", p, h}'‘`
  - ```
    awk '{print path; print home}' | {
            read PATH; read HOME; ...
       }
    ```
  - ```
    { read PATH; read HOME; } << EOF
    `awk '{print path; print home}'`
    EOF
    ```

- volání příkazu (**nawk**): funkce **`system(`***command***`)`**
  - př.: `system( "rm " filename )`
  - funkce vrací návratovou hodnotu, ale ne výstup !
  - příkaz běží v subshellu !

Příkaz **`getline`**, roura (**`nawk`**)

- **`getline`** [*var*] [**`<`**{**`"-"`** | *filename*}]

  načtení řádky z právě čteného souboru, ze standardního vstupu resp. ze souboru *filename* do polí **`$0`**, **`$1`**, ... resp. do proměnné *var*

  př.: `getline < "/etc/hosts"`

- *command* **`| getline`**

  čtení výstupu příkazu (*roura*)

  př.: `"pwd" | getline dir`

- **`print |`** *command*

  výstup do roury

  př.: `printf "Job %d ended", id | "mail " adm`

  Max. počet otevřených rour: 1 !

Jazyk C - soubory

`*.c`, **`*.cpp`**	zdrojové soubory
`*.h`	hlavičkové (*header*) soubory
`*.o`	přeložené moduly (*object-moduly*)
`a.out`	implicitní jméno výsledku kompilace
`/usr/include`	kořen systémových headerů
`/lib/lib*.a`, **`.so`**	systémové linkovací knihovny

Jazyk C - kompilátor

Volání: **`cc`** [*options*] *file...*

Nejdůležitější volby:

`-o`*filename*	výstupní jméno
`-c`	pouze překlad (nelinkovat)
`-E`	pouze preprocesor (nepřekládat)
`-O`*level*	nastavení úrovně optimalizace
`-g`*level*	nastavení úrovně ladicích informací
`-D`*macro*	definuj makro pro preprocesor
`-U`*macro*	oddefinuj makro pro preprocesor
`-I`*path*	umístění **`#include`** souborů (headerů)
`-l`*lib*	linkuj s knihovnou **`lib`***lib***`.a`**
`-L`*path*	cesta pro knihovny (**`-l`***lib*)

Předdefinovaná makra

Kromě standardních maker (**__DATE__**, **__FILE__**, **__LINE__**, **__cplusplus**, apod.) jsou v UNIXu zavedena další makra jako

unix	je vždy definováno v prostředí UNIXu
mips, **i386**,...	hardwarová architektura
__osf__,...	klon operačního systému
SunOS	verze operačního systému
_POSIX_SOURCE, **_XOPEN_SOURCE**, **_ANSI_C_SOURCE**	překlad podle příslušné normy

Výpis maker: **cc -dM -E** *file*

Program **`make`**

- generátor příkazů
- správa SW projektů
- příklad (soubor **`makefile`** n. **`Makefile`**):

```
program: main.o util.o
	cc -o program main.o util.o
main.o: main.c program.h
	cc -c main.c
util.o: util.c program.h
	cc -c util.c
```

- překlad potřebných souborů a slinkování programu:

```
make [program]
```

Syntaxe vstupního souboru (**`make`**)

- popis závislostí cíle: *targets* **:** [*files*]
- prováděné příkazy: **<Tab>***command*
- komentář: *#comment*
- pokračovací řádek: *line-beginning*\
  line-continuation

Makra (**make**)

- definice makra:

  name **=** *string*
- nedefinovaná makra jsou prázdná
- nezáleží na pořadí definic
- nelze předefinovat
- definice na příkazové řádce:

  make *target name*=*string*
- použití makra:

  $*name*, **${***name***}** nebo **(***name***)**
- systémové proměnné jsou makry

Systémová administrace

- Základní úkoly:
  - instalace (OS, SW balíky)
  - konfigurace (systémy souborů, uživatelé, služby, ...)
  - zálohování systému
  - sledování systému (`syslog`, `cron`,...)

- V principu jsou administrátorské činnosti na různých typech UNIXových systémů podobné, ale speciální prostředky pro jejich vykonávání je podstatně liší, a to i u systémů od stejných výrobců

Start systému

- Jako první proces se startuje **`init`**, jenž pak řídí další činnost systému.
- BSD systémy:
  - skript **`/etc/rc`** („run control“)
  - skripty volané z **`/etc/rc`** (např. **`/etc/rc.local`**)
  - konfigurace **`/etc/rc.conf`**
- Systém V:
  - start skriptů řídí úroveň běhu a konfigurační soubor **`/etc/inittab`**
  - skripty jsou soustředěny do adresářů **`/etc/rc#.d`**
- V současnosti obvykle nějaká kombinace

Úrovně běhu, **`inittab`**

- Volí se při startu systému, nebo voláním **`init`** *level*

- V detailech se mohou lišit, typicky ale
  - 0 ... znamená zastavení systému
  - 1 ... znamená single-user režim
  - 3 ... znamená plnohodnotný režim běhu

- Konfigurační soubor **`inittab`**:

```
l3:3:wait:/sbin/rc default
```

Startovací skripty

- Klasický způsob:
  - pro runlevel # v adresáři **`/etc/rc#.d`**
  - jména: **`S##`*`služba`*** resp. **`K##`*`služba`***
  - pořadí dáno číslem
  - skript volá jiný skript z **`/etc/init.d`**
    s parametrem **`start`** resp. **`stop`**

- Současné systémy používají rozličné variace, pořadí startu obvykle určuje systém sám podle vyznačených závislostí

The End