Úvod do UNIXu

Libor Forst

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

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 Praha 1993; ISBN 80-7169-170-4
- G.Todino, J.Strang, J.Peek: Learning the UNIX Operating System; O'Reilly & Associates 1993
- L.Lamb: Learning the vi Editor; O'Reilly & Associates 1986-1990; ISBN 0-937175-67-6

Literatura (systém)

- 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

Literatura (programování)

- M.Jelen: UNIX V - programování v systému; Grada Praha 1993; ISBN 80-85623-16-1
- B.Rosenblatt: Learning the Korn Shell; O'Reilly & Associates 1993; ISBN 1-56592-054-6
- D.Dougherty: sed & awk; O'Reilly & Associates 1990; ISBN 0-937175-59-5
- D.Curry: Using C on the UNIX System; O'Reilly & Associates 1985,7,8; ISBN 0-937175-23-4
- A.Oram, S.Talbott: Managing Projects with make; O'Reilly & Associates 1986,91; ISBN 0-937175-90-0

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,...

Současné UNIXy

- 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

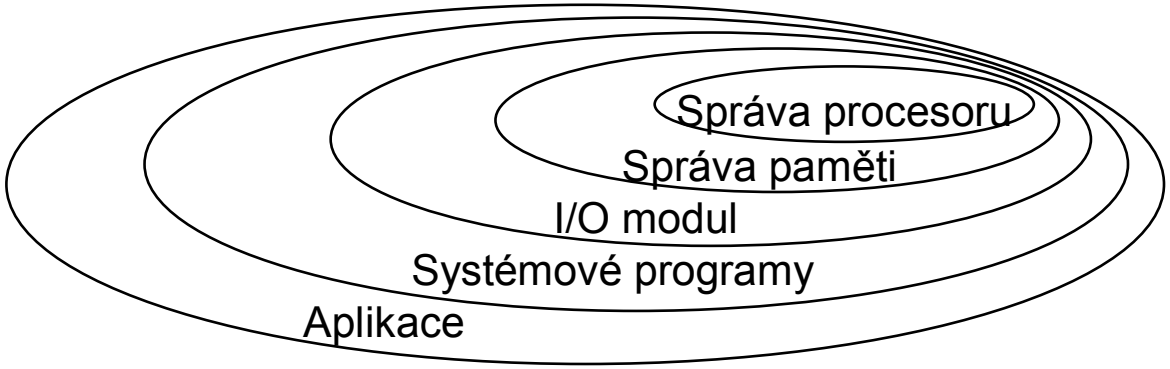
- SVID (System V Interface Definition)
  - "fialová kniha", kterou AT&T vydala poprvé v roce 1985 jako standard, jehož splnění je nutnou podmínkou pro použití obchodního názvu UNIX
- POSIX (Portable Operating System based on UNIX)
  - série standardů organizace IEEE značená P1003.xx, postupně je přejímá 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
  - dnes Version 2 (**UNIX98**)
  - splnění je nutné pro užití obchodního názvu UNIX

Charakteristika UNIXu

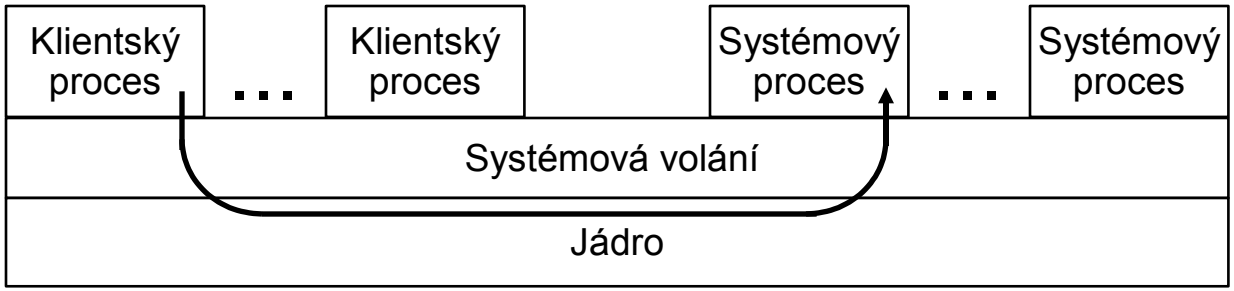
- poučení ale nezatížení minulostí
- nekomerční prostředí
- otevřený operační systém
- systém souborů
- uživatel, skupina
- proces, komunikace
- interpret příkazů, grafické prostředí
- utility, jazyk C
- přenositelnost, modifikovatelnost
- síťová podpora
- volně šiřitelný SW (např. GNU)
- příkaz **`man`**

Model OS UNIX

Klasický OS



UNIX



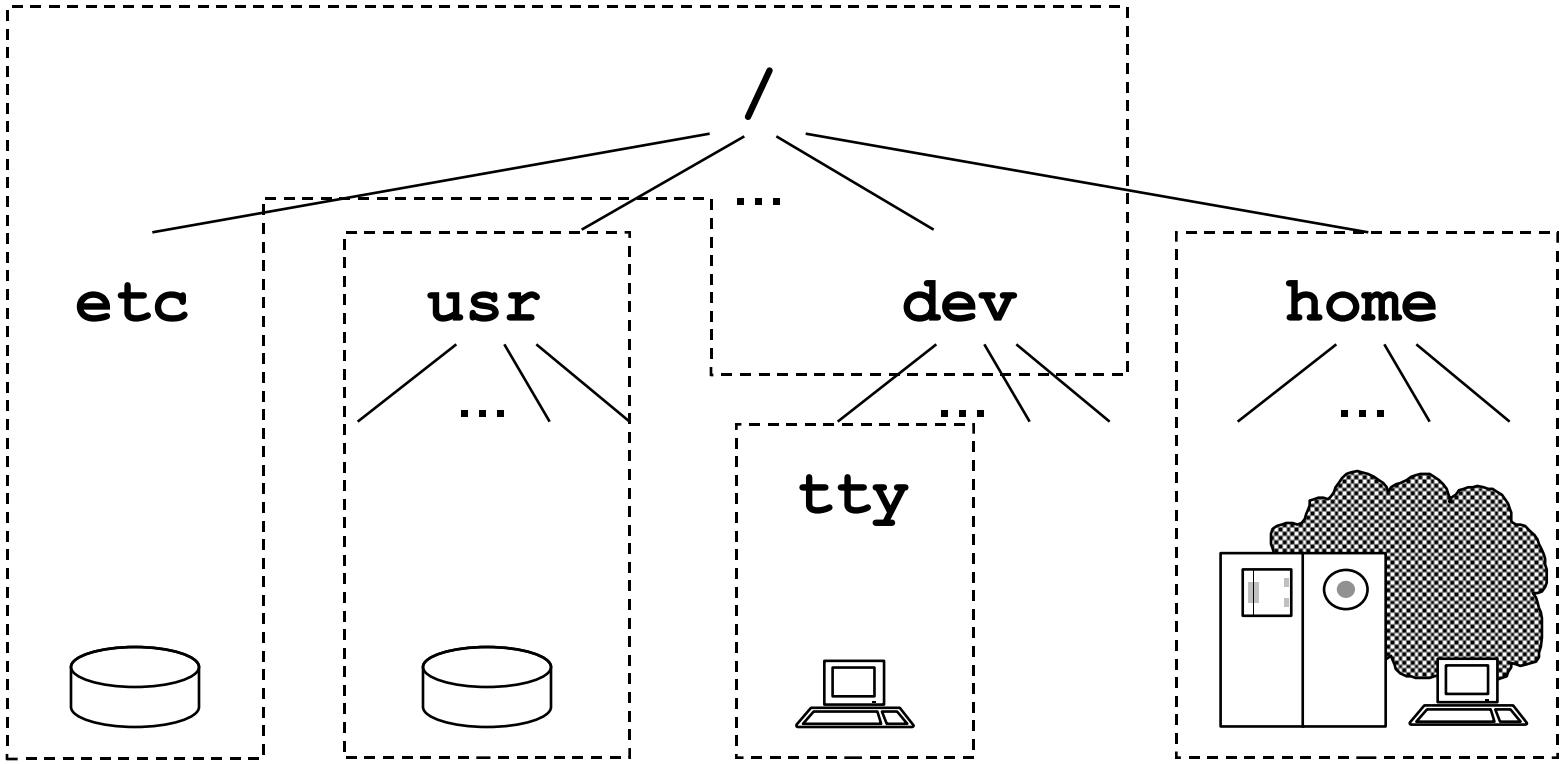
Funkce jádra OS

- Řízení provádění procesů (vytváření, ukončení, suspendování, komunikace, přístup k periferiím,...)
- Správa systému souborů (alokace, uvolňování, ochrana, udržování konzistence,...)
- Přidělování paměti, ochrana paměti, 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 a výjimek, např.:
  - HW chyby
  - časovač
  - disky
  - síť
  - terminály
  - SW přerušení
- Správa paměti - oddělení virtuálního a skutečného adresního prostoru

Jednotný hierarchický systém souborů



Strom adresářů

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

Proces, komunikace

- Proces
  - zjednodušeně:
    běžící uživatelský nebo systémový program
  - vzniká duplikací rodičovského procesu

- 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 komunikaci s UNIXem
- nezávislá komponenta systému: více shellů
- formát příkazů:

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

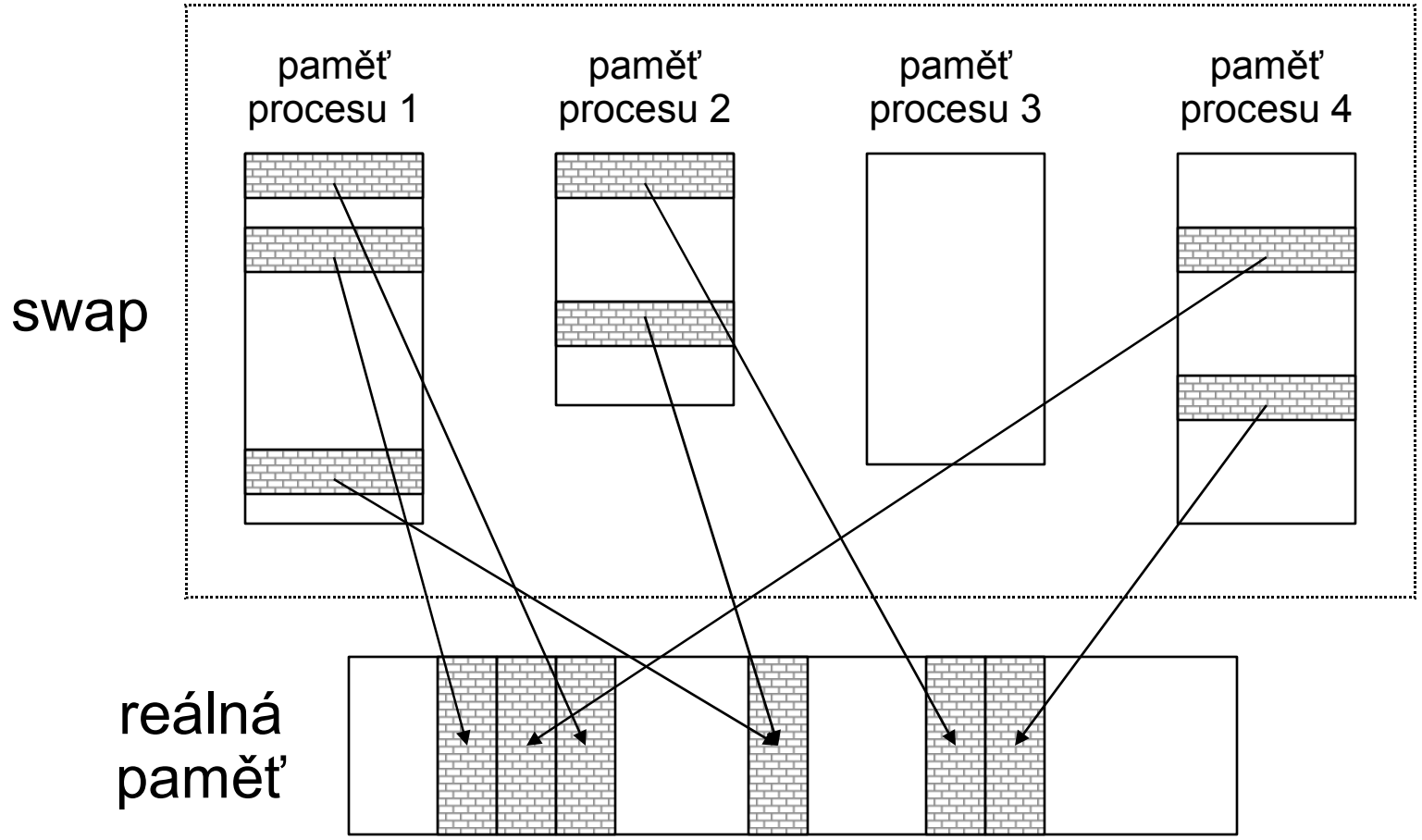
- metaznaky, např.:

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

- vestavěné příkazy: **`echo`**, **`cd`**, **`pwd`**
- umožňuje přímé programování; skripty

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

Virtuální paměť



Příkaz **man**

- Volání:

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

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

Seznam uživatelů (`/etc/passwd`)

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

Význam jednotlivých polí:

- uživatelské jméno
- zakódované heslo (nově v **`/etc/shadow`**...)
- číslo (*UID*); superuživatel (*root*) má UID 0
- číslo (*GID*) primární skupiny
- plné jméno
- 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 se uživateli spustí jeho shell.
Tím se zahájí jeho uživatelská relace (*session).*

- ukončení session: **`logout`**
- změna uživatele (login-shell): **`login`** *user*
- dočasná změna 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

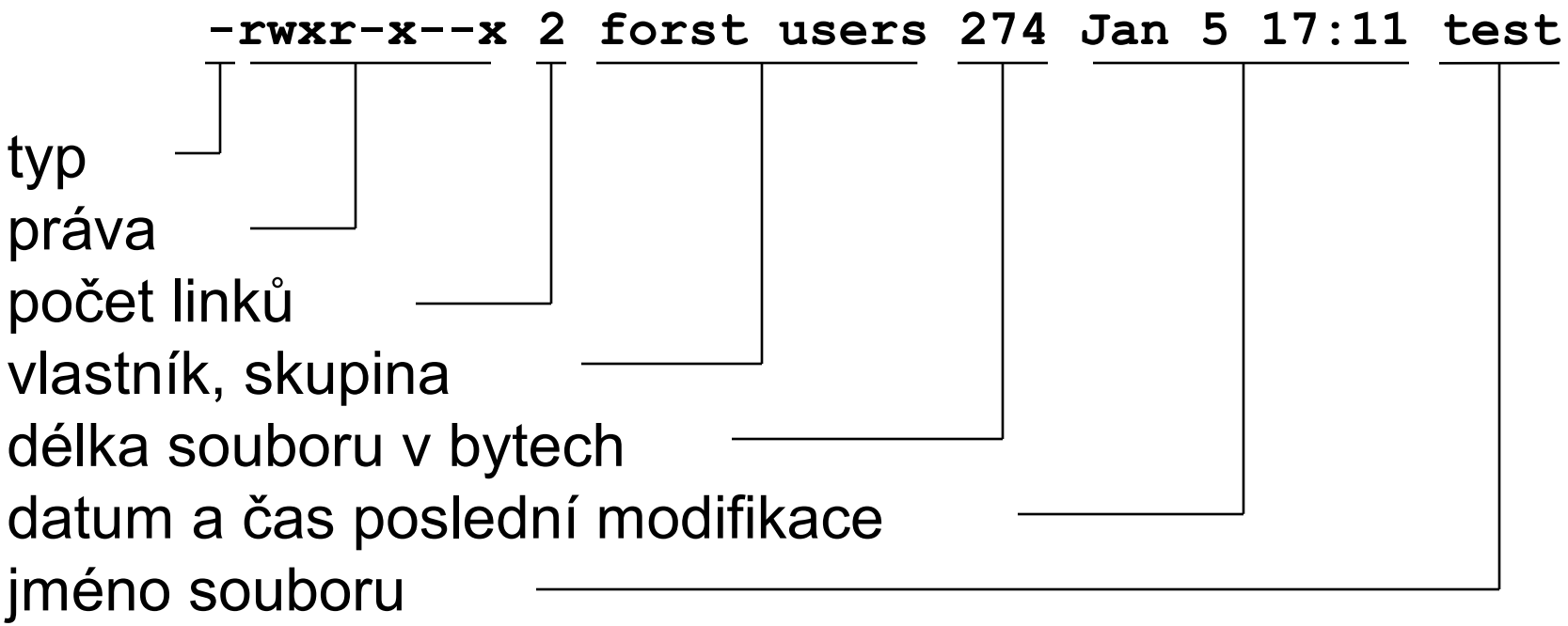
- on-line zprávy:
  - zaslání: **`write`** *user*
  - potlačení příjmu: **`mesg`** [**`y`** | **`n`**]
- on-line rozhovor:
  - příkaz: **`talk`** *user*[**`@`***host*]
- off-line zprávy:
  - posílání: **`mail`** [**`-v`**] [**`-s`***subject*] *email...*
  - příjem: **`mail`**
  - zpráva o příjmu: **`biff`**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

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

Systém souborů

- hierarchický systém
- jednotný přístup k zařízením, adresářům
- diskové svazky, síťové disky
- synchronizace (**`sync`**, **`fsck`**)
- ochrana souborů
- jména (znaková sada, délka)
- cesty (absolutní, relativní, `.`, `..`)
- textové soubory (**`<LF>`**)

Příkaz `ls`



volby: dlouhý výpis (**`l`**), krátký výpis do 1 sloupce (**`1`**), psát i skryté (**`a`**), vypsat/potlačit skupiny (**`g`**), třídit podle času (**`t`**), třídit pozpátku (**`r`**), značit typ souboru (**`F`**), rekurze (**`R`**) resp. jen adresáře (**`d`**), sledovat linky (**`L`**)

Typy souborů

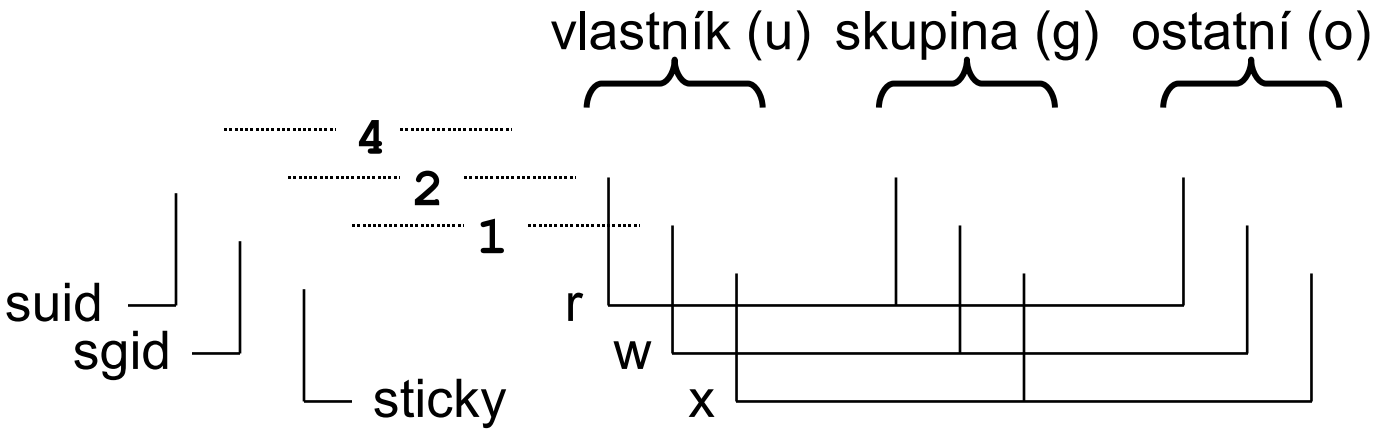
- *plain file*: posloupnost bytů (**`-`**)
- *directory*: seznam jmen souborů (**`d`**)
- speciální zařízení - *device* (**`b`**, **`c`**)
- *symbolické linky* (**`l`**)
- pojmenované roury - *pipe* (**`p`**)
- *sockety* (**`s`**)

- příkaz **`file`**

Přístupová práva

- tři kategorie: vlastník (**`u`**), skupina (**`g`**), ostatní (**`o`**); platí vždy nejspeciálnější kategorie, v níž je uživatel
- tři práva: čtení (**`r`**), zápis (**`w`**), provádění souboru resp. přepnutí se do adresáře (**`x`**)
- setUID, setGID (**`s`**) pro proveditelné soubory: propůjčení identity (skupiny) vlastníka
- setGID pro soubor bez práva spuštění pro skupinu: kontrola zámků při každém přístupu (výpis: **`S`**)
- sticky bit (**`t`**) pro spustitelné soubory: ponechání souboru v paměti
- sticky bit pro adresáře: práva k souborům mají jen vlastníci souborů a nikoli vlastníci adresáře
- setGID pro adresář: nové soubory budou mít stejnou skupinu jako adresář

Změna přístupových práv



- změna práv:
  `chmod` [**`-R`**] **`+w,o=rx`** *file...*
  `chmod` [**`-R`**] **`775`** *file...*
- změna vlastníka: **`chown`**, **`chgrp`**
- defaultní maska: **`umask`** *mask_complement*
- shell s novou defaultní skupinou: **`newgrp`** *group*

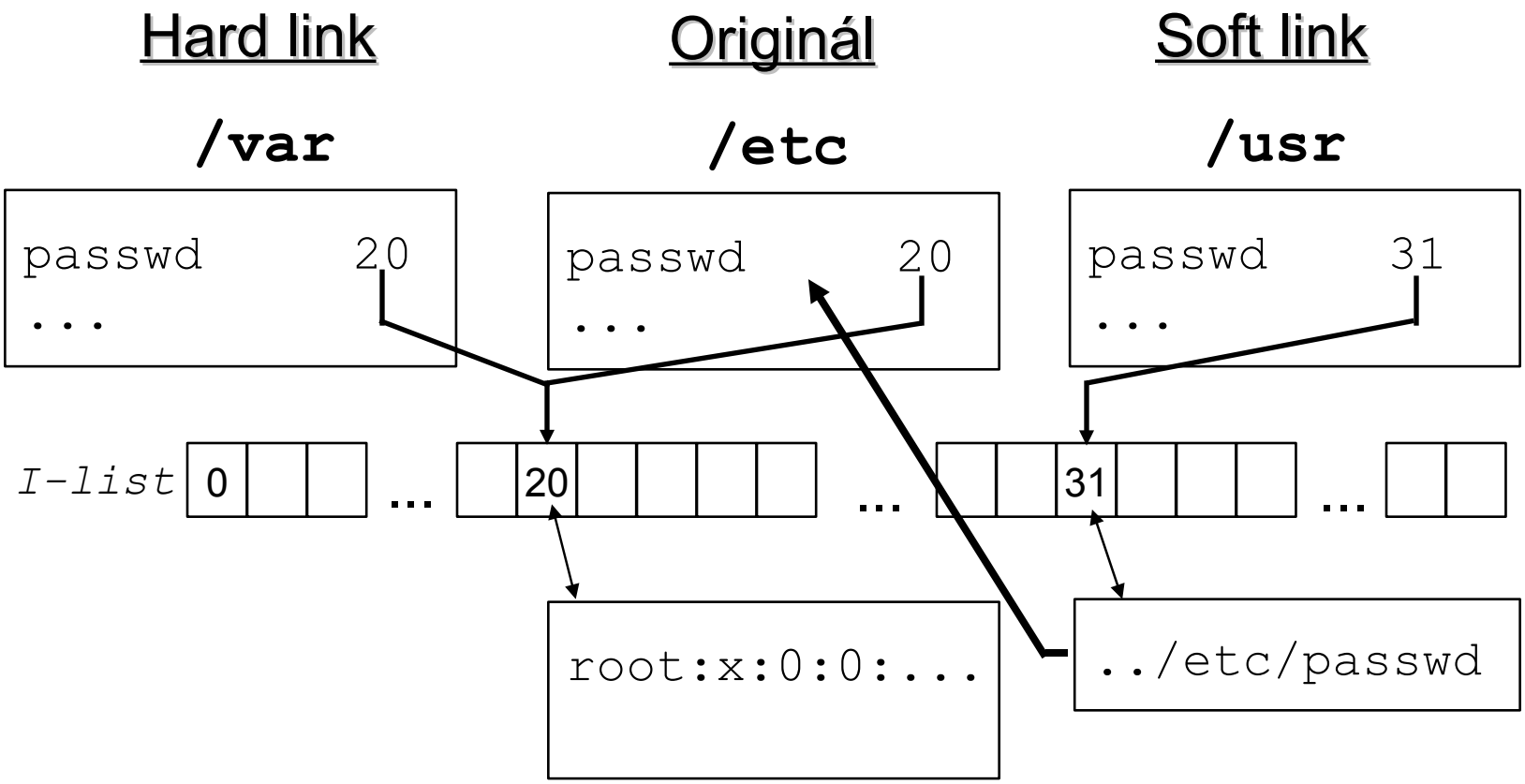
Organizace disku

- Fyzická: *sektor*, *stopa*, *cylindr*, *povrch*
- Logická: *partition* (~device), příkaz **`df`**
- Systémová: *filesystem*
  - boot blok
  - superblok(y)
  - I-list (seznam I-nodů)
  - datové bloky
- Obraz systému souborů v paměti (**`sync`**, **`fsck`**)

Index node

- Každý soubor na disku má právě jeden I-node, který obsahuje:
  - počet linků
  - vlastník, skupina
  - 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`**

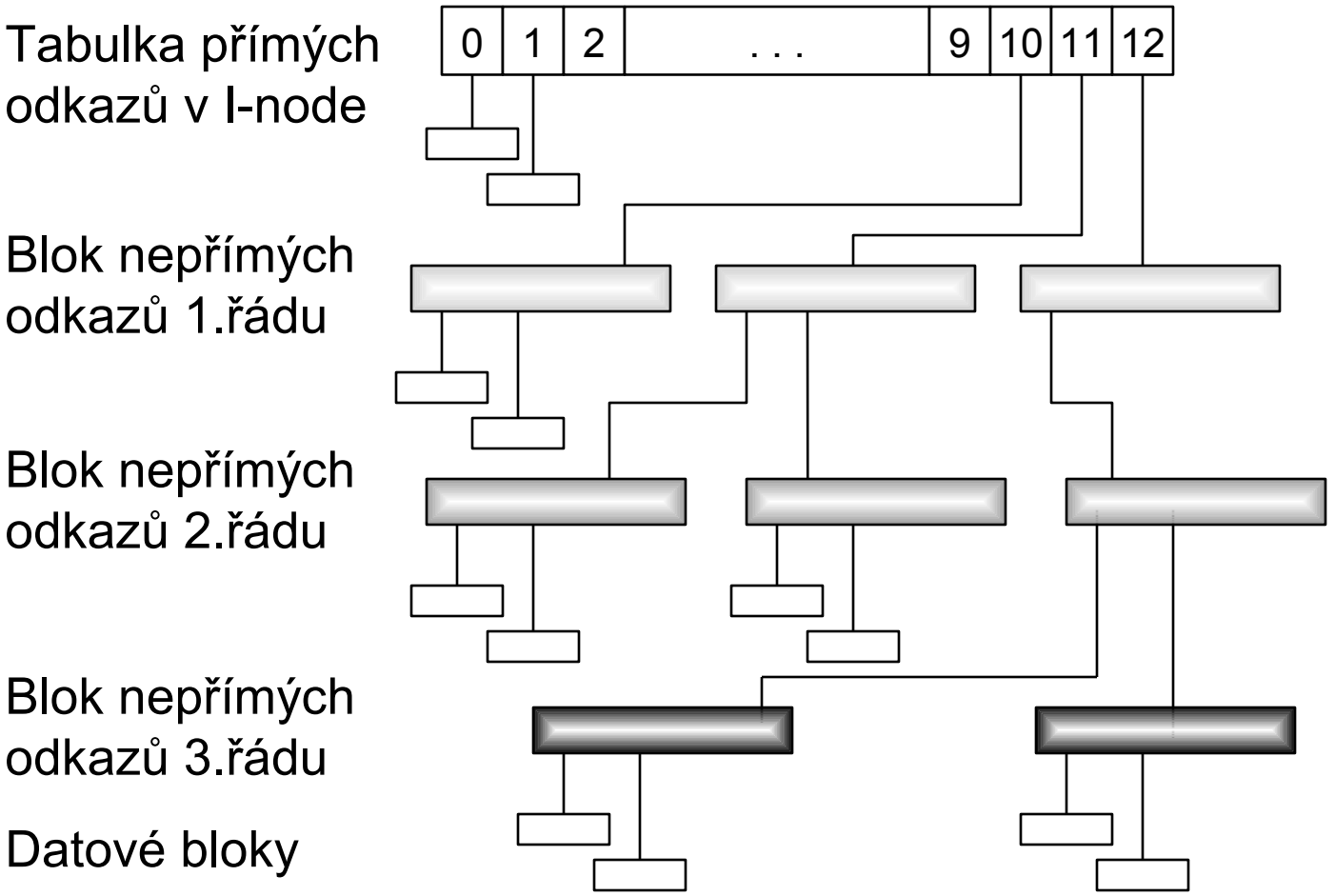
Linky



`ln /etc/passwd passwd`

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

Adresace datových bloků



Obecné příkazy

- kopírování souboru: **`cp`** [`-pr`]
- přesun n. přejmenování souboru: **`mv`**
- smazání souboru: **`rm`** [`-rfi`]
- změna data a času: **`touch`** [{ `-t`*čas* | `-r`*soubor* }]
- změna aktuálního adresáře: **`cd`**
- výpis aktuálního adresáře: **`pwd`**
- vytvoření adresáře: **`mkdir`** [`-p`]
- zrušení adresáře: **`rmdir`**

- není undelete !

Výpis souboru

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

- výpis binárního souboru: **`od`**, **`hd`**, **`strings`**
- výpis s formátem: **`hexdump`** [**`-e`** *formát*]

  formát: [[*repeat*]**/**[*byte_count*]] *"format"* ...

Příkaz **more**

- Volání:

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

- Příkazy (* - může předcházet prefix počtu *k*):
  - mezera, **d** ... další stránka, půl stránky (*)
  - return ... 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

	BSD	SystemV
• tisk:	**lpr** [**-d** *printer*] [*files*]	**lp** [**-P***printer*] [*files*]
• výpis stavu tisku:	**lpq** [**-d** *printer*] *job*	**lpstat** *job*
• zrušení tisku:	**lprm** [**-d** *printer*] *job*	**cancel** *job* [*printer*]

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

Zpracování textu

- porovnávání souborů resp. adresářů:

  `diff` [**`-bwi`**] { **`-e`** | **`-c`***n* | **`-rs`** } *file1 file2*

  `comm` [**`-123`**] *file1 file2*
- výběr polí z řádek souboru (nemění pořadí polí):

  `cut` { **`-c`***list* | **`-f`***list* **`-d`***char* } [*files*]
- spojení souborů „po sloupcích“ resp. řádek souboru:

  `paste` [[**`-s`**] **`-d`***char*] [*files*]
- rozdělení souboru po řádcích:

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

  `tr` [**`-cds`**] *table1* [*table2*] př.: `tr "ABC" "abc"`

Příkaz **`sort`**

- Volání:

  `sort` [**`-bidfnr`**] [**`+`***pos1*[**`-`***pos2*]] [**`-t`***d*] [**`-uc`**] [*files*]

- Zadání třídícího pole:
  - *pos1* ... první znak, *pos2* ... první znak za
  - tvar *field*[**`.`***char*] ... číslování od 0, 0 je default
- Modifikátory: **`b`** (bez mezer), **`f`** (ignorcase), **`n`** (čísla), **`r`** (opačně)
- Přepínače: **`t`** (oddělovač pole), **`u`** (vyluč stejné klíče), **`c`** (jen kontroluje uspořádání)
- Více klíčů (pozice se číslují od 1, *pos2* je poslední znak!):

  `sort -k`*pos1*[**`,`***pos2*][*mod*] **`-k`***...* [*files*]

Příkaz **`find`**

- volání: **`find`** *cesta... podmínka... akce*
- podmínky:
  - **`name`**, **`size`**, **`type`**, **`links`**, **`inum`**, **`fstype`**
  - **`user`**, **`group`**, **`perm`**
  - **`atime`**, **`ctime`**, **`mtime`**, **`newer`**
  - negace (**`!`**), **`-o`**, **`-a`**, závorky
  - hodnoty: *n*, **+***n*, **-***n*
- akce:
  - **`print`**; někdy default
  - **`exec`**; jméno souboru: **`{}`**, ukončení příkazu: středník
- příklad:

```
find / -name core -mtime +7 -exec rm {} ";"
```

- zkratky: **`which`**, **`whereis`**

Příkaz **xargs**

- volání: **xargs** *příkaz*
  - zavolá *příkaz*, jako argumenty doplní text standardního vstupu

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

- volání: **xargs -i** *příkaz*
  - opakuje *příkaz* pro každou řádku standardního vstupu, její text doplní do příkazu na místa označená **{}**
  - př.: `ls -1 *.c | xargs -i cp {} {}.bak`

Archivace

- archivace adresářů: **`tar`** {**`c`**|**`t`**|**`x`**} [**`f`** *`file`*] [*files*]

  - př.: **`tar cf - . | ssh host tar xf -`**
- komprese souborů
  - standardní (**`.Z`**): **`compress`**
  - GNU (**`.gz`**): **`gzip`**, **`gunzip`**
- další příkazy: **`cpio`**, **`dd`**
- systémová záloha: **`backup`**, **`dump`**, **`restore`**
- zálohování po síti: **`rdump`**, **`rrestore`**
- práce s páskou: **`mt`** {**`fsf`** *`n`*|**`bsf`** *`n`*|**`rewind`**}

Regulární výrazy (**ed**, **vi**)

Způsob definování řetězců v řadě příkazů (např.: **ed**, **vi**).

Speciální znaky:

- **.** … jakýkoliv znak mimo **<LF>**
- *****c* … metaznak použitý jako znak (např.: `\.` je tečka)
- **[***list***]**, **[^***list***]** … jakýkoliv znak z výčtu, z doplňku

  př.: `[a-z0-9_]`, `[]^-]`, `[\]\^\-\\]`
- **^**, **$** … začátek a konec řádky
- **\<**, **\>** … začátek a konec slova
- *exp****** … libovolné opakování znaku daného výrazem
- *exp***\{***n***\}**, *exp***\{***m***,***n***\}** … opakování *n*krát, *m*-*n*krát
- **\(**, **\)**, *****n* … část vzoru a její použití (např. v náhradě)

  př.: `A\(.\)\1\{1,\}A`

Editor **`vi`**

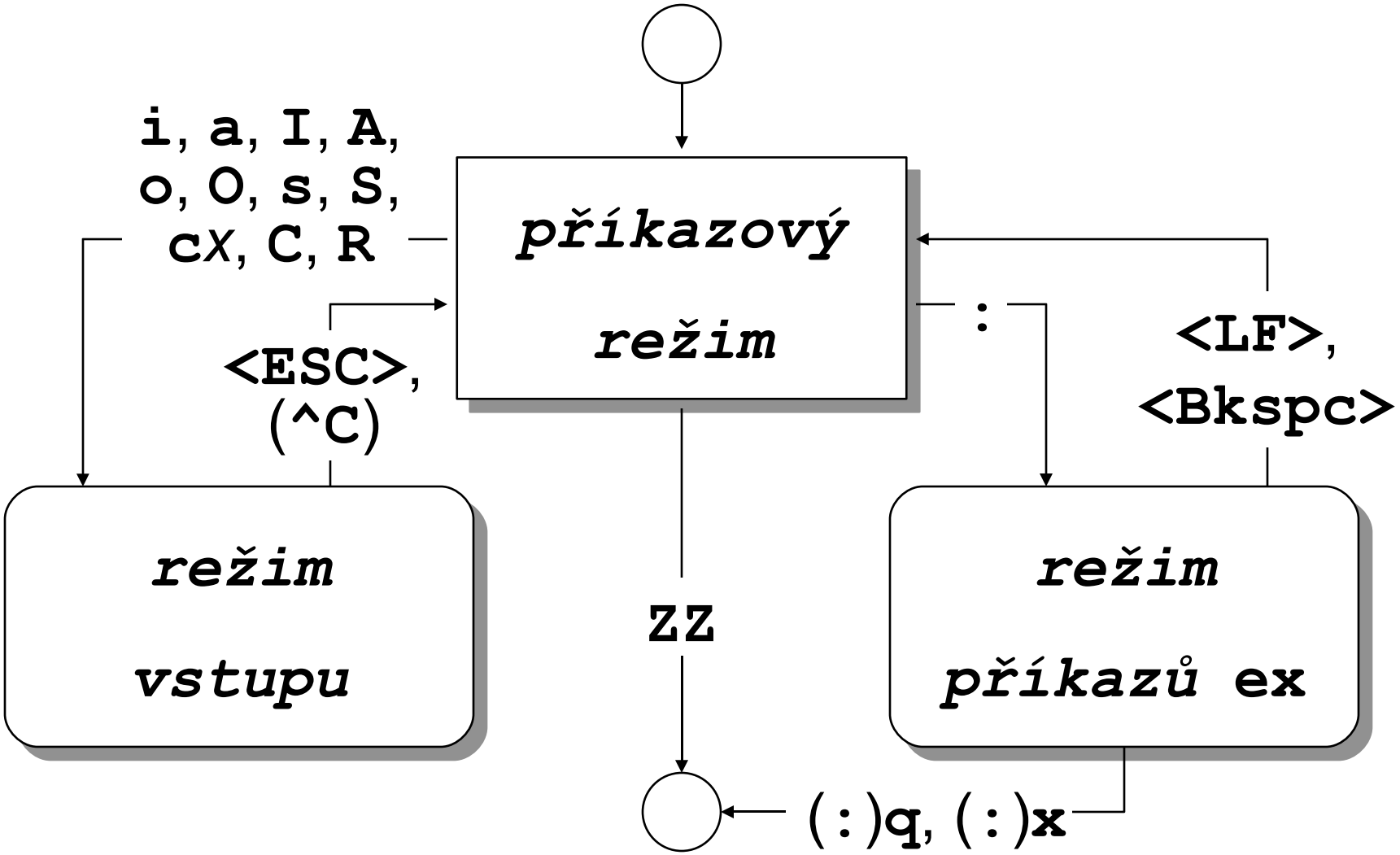
- visual editor
- geneze: **`ed`** **`ex`** **`vi`**
- celoobrazovkový editor
- dostupný na všech UNIXech
- široká paleta příkazů
- malé nezbytné minimum příkazů
- editace kopie souboru
- volání:

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

Základní editace (**`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

Režimy práce **vi**



Příkazy pro pohyb (I)

Před příkazy může předcházet opakovací faktor *k*

- **h** (**<BKSPC>**), **j**, **k**, **l** (**<SPACE>**) … o *k* pozic (, , ,)
- **w**, **b**, **e**, **W**, **B**, **E** … o *k* slov (vpřed, vzad, na konec resp. bez 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 řadky 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)

Jiný způsob 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
- **^L**, **^R** … obnovení obrazovky
- **o** … přechod do celoobrazovkového režimu
- **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 vložení výstupu příkazu do textu
- **@***x* … provedení příkazů uložených v bufferu *x*
- **%** … skok na odpovídající **)**, **]**, **}** nebo **>**
- **^W**, **^V** … (režim vstupu) smaž slovo, ulož speciální znak

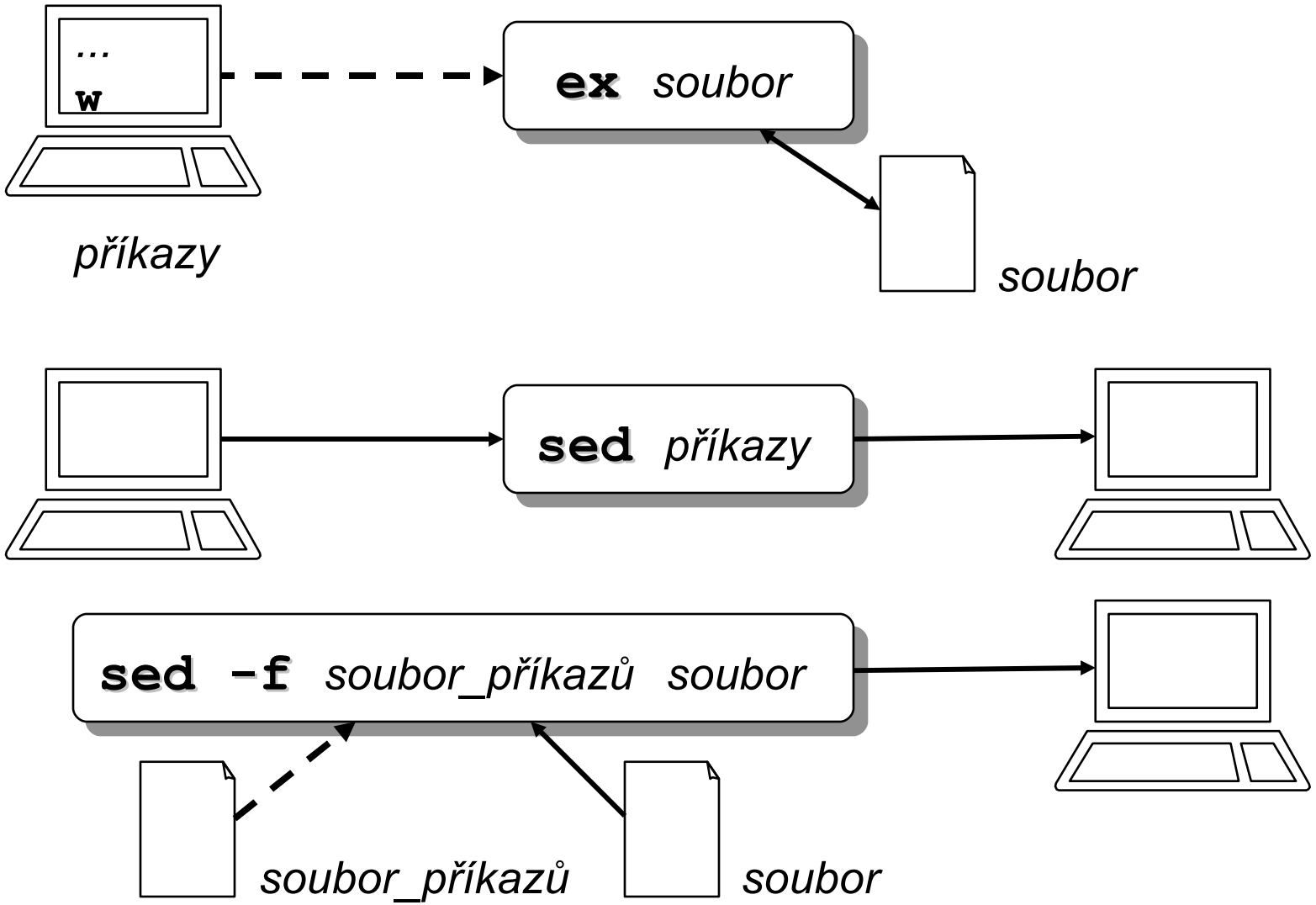
Řádkové editory

ed - základní editor
- dostupný i v diagnostickém režimu
- edituje kopii souboru, opravy je nutno zapsat
- příkazy ze vstupu (**ed**-skripty)
- volání: **ed** *soubor*

ex - rozšíření **ed**u
- součást **vi**

sed - rozšíření **ed**u směrem k programování
- edituje vstupní proud, výsledek vypisuje
- editovací příkazy jsou součástí volání
- volání: **sed** *příkazy* [*soubor ...*]
  nebo: **sed -f** *příkazový_soubor* [*soubor ...*]

Schéma práce **sed** a **ex**



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

- Syntaxe příkazů:

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

- Formát adresy - příkaz se provede na...

	ed	**sed**
(prázdná)	aktuální řádce	každé řádce
.	aktuální řádce	-
n	řádce s číslem *n* (od 1)	řádce s číslem *n*
$	poslední řádce	poslední řádce
±[*n*]	relativně k aktuální řádce	-
/pat/	násl. ř. se vzorkem	každé ř. se vzorkem
?*pat***?**	předch. ř. se vzorkem	-
' *x*	ř. označené značkou *x*	-
adr±[*n*]	relativně k ř. s adresou *adr*	-

Poziční příkazy editoru **ed**

a(ppend), **c**(hange), **i**(nsert)
… vkládání textu (nové řádky následují a končí řádkou se samotnou tečkou)

d(elete), **j**(oin) … mazání, spojování řádek

m(ove)*adr*, **t**(o)*adr* … přesun, kopírování za řádku *adr*

s(ubst)**/**[*pattern*]**/***replacement***/**{**g**|*n*} … náhrada řetězců

g(lobal)**/***regexp***/***cmd* [**\<LF>***cmd*]
… provedení příkazů na všechny řádky se vzorem

v/*regexp***/***cmd* … provedení na řádky neobsahující vzor

p(rint), **n**(um), **l**(ist) … tisk, s čísly, s netisknutelnými znaky
(dají se připojovat za ostatní příkazy)

r*file* … vložení textu souboru

k*x* … nastavení značky *x* (mark)

Nepoziční příkazy editoru **ed**

h(elp) … nápověda k poslední chybě

u(ndo) … zrušení poslední opravy

e(dit) [*file*] … (znovu)otevření souboru

e(dit)[**!**] [*file*] … otevření souboru bez uložení změn

w(rite) [*file*] … uložení (pod jiným jménem)

(v případě udání rozsahu se zapíše jen rozsah)

W(rite) *file* … připsání do souboru *

w(rite)**!***cmd* … zápis do roury *

f(ile) *file* … změna jména editovaného souboru

q(uit) … ukončení editace

Příkazy označené * nejsou zcela přenositelné.

Příklady použití příkazu **`global`**

- **`g/procedure/i`**

  `.`

  před procedurami odřádkuje

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

  „rozšíří“ program

- **`g/var/,/type/-1 m /begin/-1`**

  „spraví“ pořadí deklarací

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

  napíše soubor pozpátku

- **`g/^Chapter/ ++W file`**

  napíše seznam kapitol

Příkaz **grep**

- název: **g/*re*/p**
- varianty:
  - **egrep** (*extended - rozšířené* regulární výrazy)
  - **fgrep** (*fast* - pouze řetězec, ale ne jen jeden)
  - a některé další (např.GNU)
- volby:

  -c(ount), **-l**(istfiles), **-n**(umber)

  -i(gnorecase), **-w**(ord)

  -v ... vypisuje řádky, na nichž vzor nebyl nalezen

  -e *expression*, **-f** *filename*

  -*lines* ... počet vypsaných řádek před a po nalezené (není zcela přenositelné)

ex - rozšíření příkazů (I)

- adresy mohou být odděleny středníkem - první řádka se stává aktuální
- rozšíření příkazu **substitute**
  - **s/***regexp***/***repl***/c** … nahrazení s potvrzováním (**y<LF>**)
  - regexp metaznak **~** ... předchozí regexp
  - repl sekvence **\u**, **\l**, **\U**, **\L** ... převod malá/velká (na celé slovo)
- nové příkazy
  - **co** (kopíruj, alias příkazu **t**)
  - **<***count*, **>***count* … indentace
  - **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!**, **w>>** … zápis do souboru, read-only, append
- **x**, **wq** … zápis a ukončení editace
- **q**, **q!** … ukončení editace (bez uložení změn)
- **n**[**!**], **e**[**!**] [*file*] … editace dalšího souboru (**%** značí aktuální jméno souboru, **#** alternativní jméno), pojmenované buffery, poslední regexp a příkaz se nemažou
- **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**).

Filtr **sed**

- stream editor (*filtr*); velké soubory, programování
- volání:

  sed [**-n**] {[**-e**] *cmd* |**-f** *script*} [*file*]
- příkazy se aplikují na každou řádku
- příkazy se oddělují středníkem nebo koncem řádky
- provádějí se v pořadí zápisu
- příkaz nesmí končit mezerou
- příklad:

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

sed - rozšíření syntaxe

- **#***komentář*
- **#n** … zapnutí přepínače **-n**
- *adresa***!***příkaz* … doplněk adresního prostoru
- *adresa***{**

  příkazy

  } … skupina příkazů

sed - rozšíření příkazů (I)

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

    příkaz i vkládané řádky kromě poslední se ukončují ****:

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

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

    funkce analogická příkazu **tr**

sed - rozšíření příkazů (II)

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

    (pokud od posledního načtení řádky nebo posledního provedení příkazu `test` byla provedena nějaká substituce)

    př.:

```
:loop
s/([^()]*)//
t loop
```

    ... vymaže všechny (i vnořené) spárované závorky

sed - rozšíření příkazů (III)

- víceřádkový prostor (zlom řádek ve vzoru: **\n**)
  - **N**(ext) … připojení další řádky ze vstupu
  - **P**(rint) … tisk první řádky z prostoru
  - **D**(elete) … vymazání první řádky z prostoru

- záložní prostor (*hold space*)
  - **h**, **H**(old) … kopie (append) do *hold space*
  - **g**, **G**(et) … kopie (append) do prostoru vzoru
  - **x**(change) … záměna prostorů

Příklady použití příkazu **sed**

- **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 '3,4!p' program.pas**
  vypíše program bez druhých dvou řádek
- **sed -n '/:\([0-9]*\):\1:/p' /etc/passwd**
  vypíše uživatele se stejným UID a GID

Příklady použití příkazu **substitute**

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

  výsledek: `User: forst`

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

  výsledek: `cp test.c test.bak`

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

  výsledek: `ab`

  správně: **y/ab/ba/**
  s/a/x/g;s/b/a/g;s/x/a/g

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

  výsledek: `Libor Forst:/home/`

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

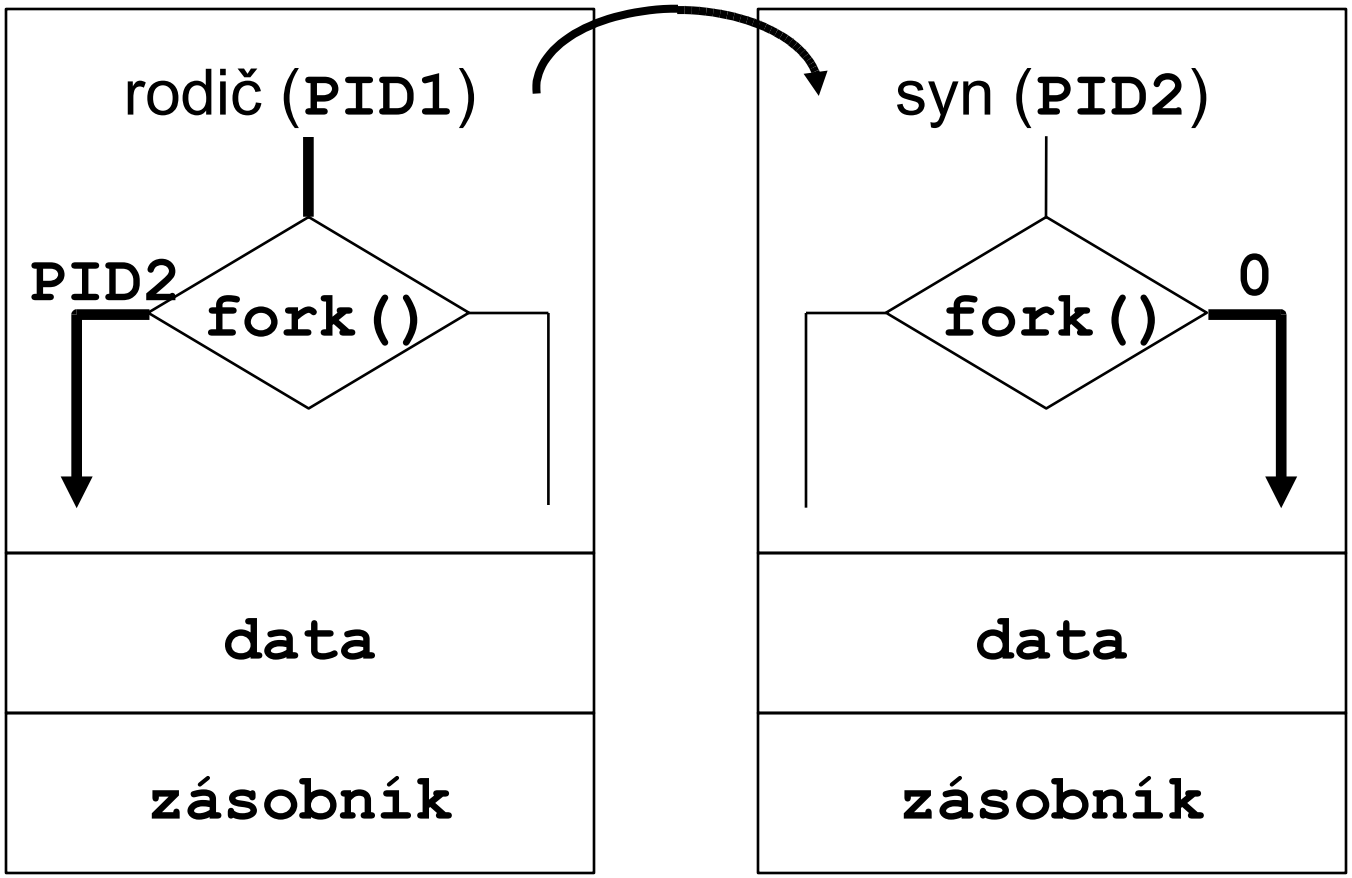
Proces

- prováděný program ... (nejméně jeden) proces
- kontext procesu
  - paměť, soubory, proměnné,...
- rodičovský proces synovský proces
- komunikace
  - signály, roury, sockety, sdílená paměť,...
- PID, příkaz **`ps`**

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
  - pamět, kterou proces dosud adresoval v uživatelském režimu
  - paměť v prostoru jádra, která je s daným procesem spojena (např. systémový zásobník procesu)

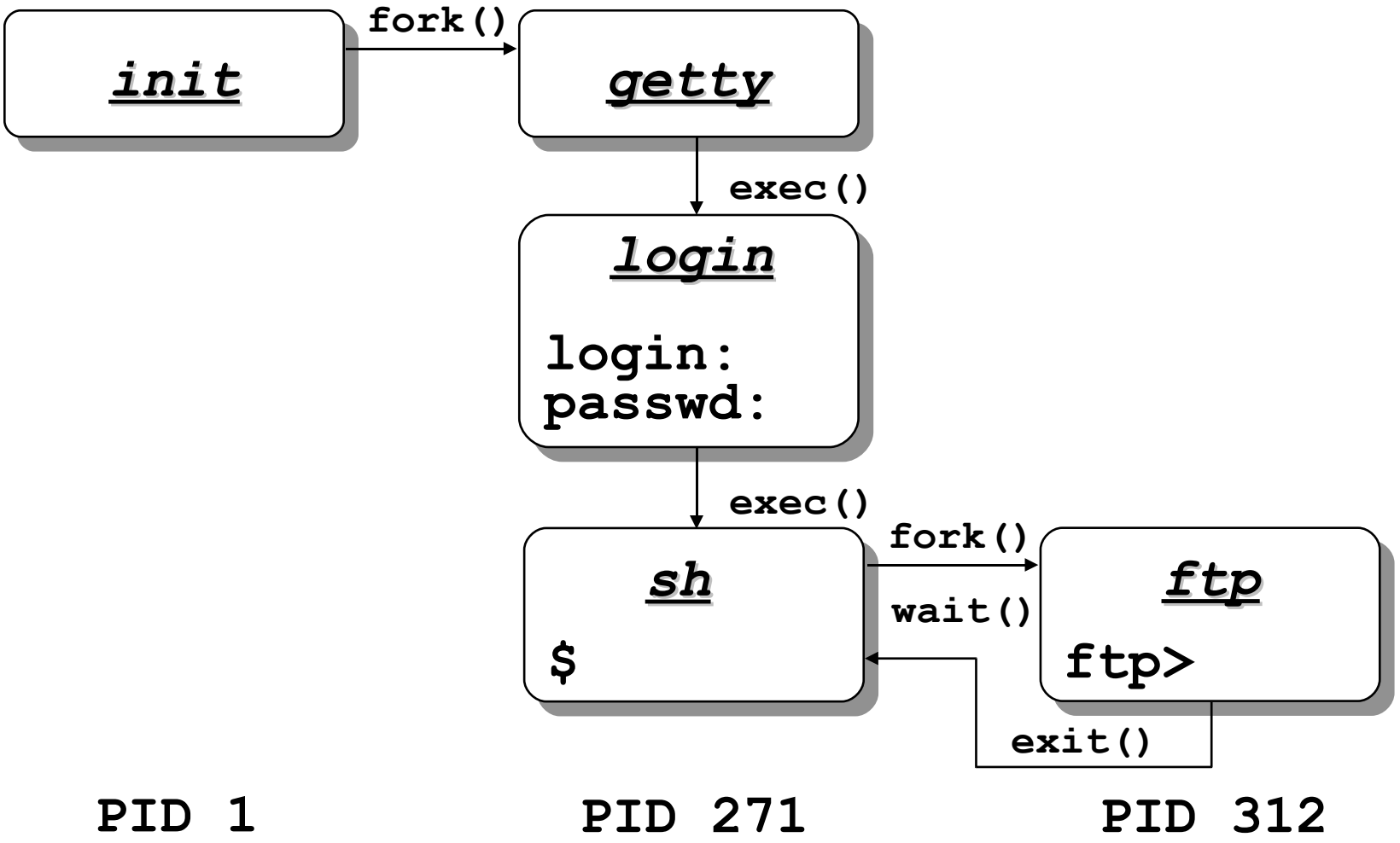
Vznik procesu



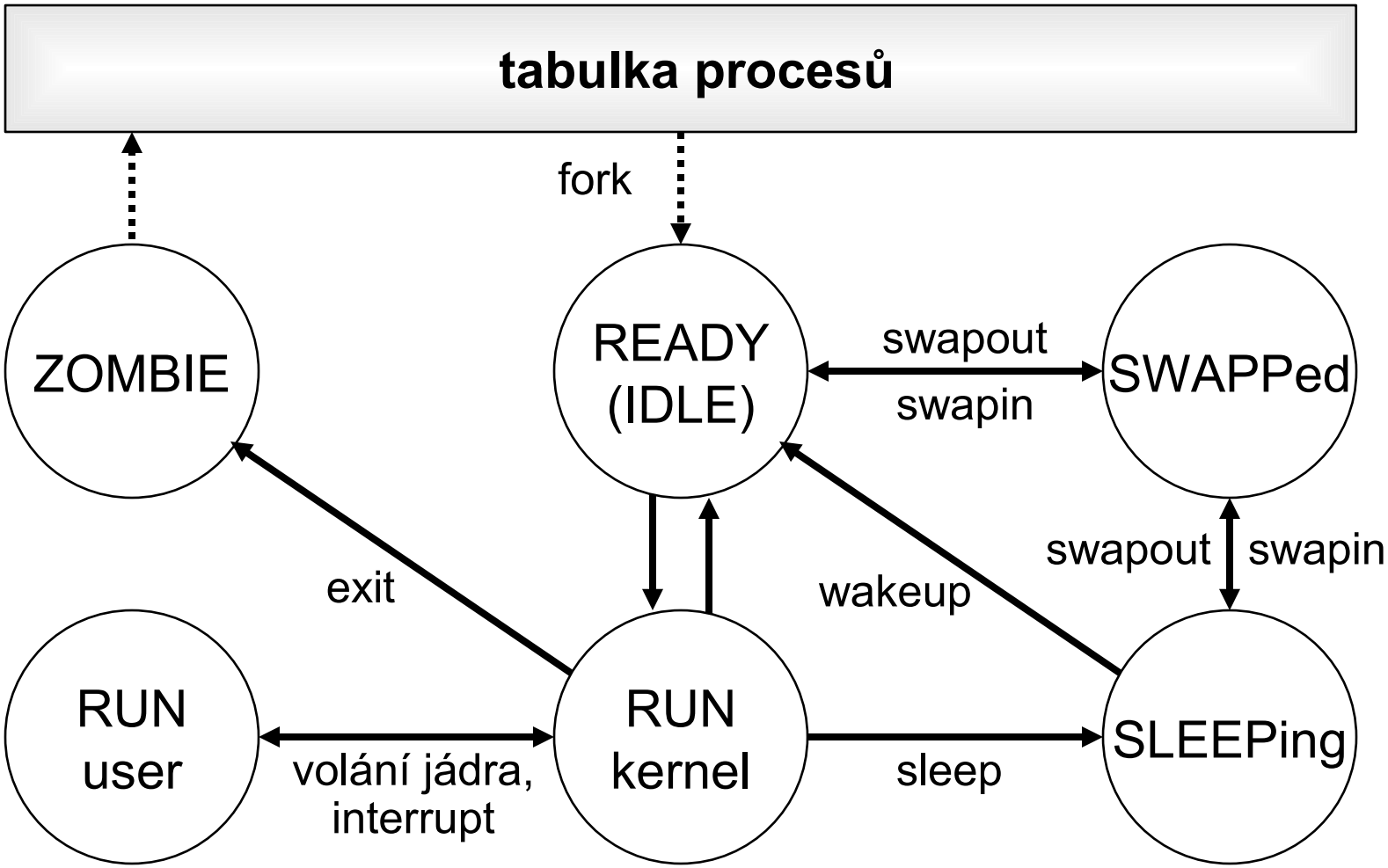
Funkce na vytváření procesů

- **`fork()`** … vytváří kopii rodičovského procesu
- **`exec()`** … překryje adresní prostor procesu zadaným programem
- **`wait()`** … (rodičovský proces) čeká na skončení potomků
- **`exit()`** … ukončí proces a aktivuje rodičovský proces

Uživatelská relace



Stavy procesu



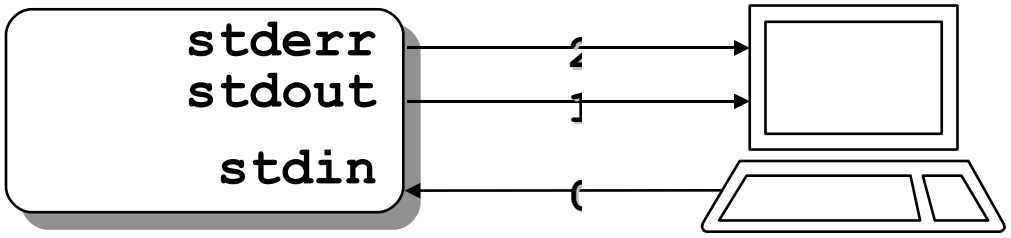
Příkaz **ps**

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

	BSD	SystemV
• cizí procesy:	**-a** (all users) **-x** (no terminal)	**-e** (všechny)
	-p *PID* **-t** *tty*	**-U** *user*
• plný výpis:	**-l** (long) **-u** (usage)	**-l** (long) **-f** (full)
	-o*key*,… (pouze vyjmenované) **-O***key*,… (vyjmenované navíc)	
• třídění:	**-r** (cpu) **-m** (memory)	(*mj. existuje PD program* **top**)

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

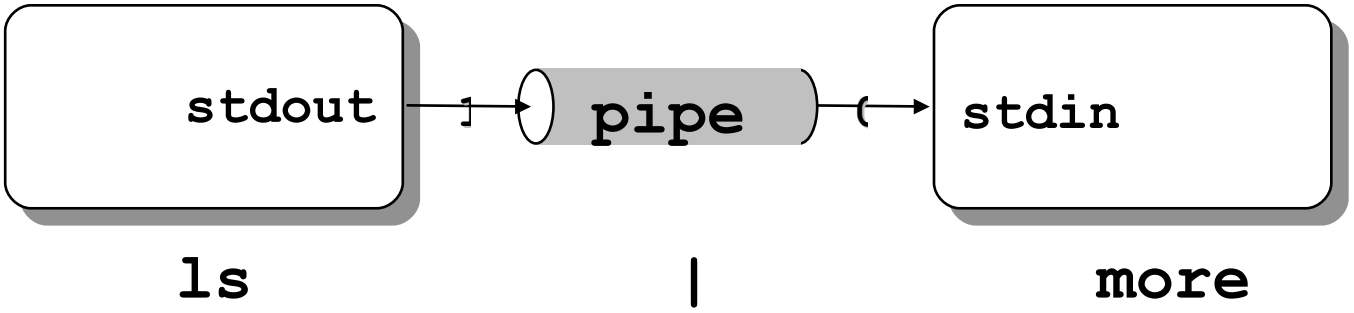


Komunikace mezi procesy

- zasílání signálů
- vstup/výstup přes roury
- System V Interprocess Communication
  - semafory
  - zasílání zpráv
  - sdílená paměť
- BSD Sockets
  - zasílání zpráv, vytváření proudů
  - v rámci jednoho systému i po síti

Roury (*pipes*)

- v shellu - spojení vstupu a výstupu dvou procesů



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

Signály

- zaslání signálu:
  - příkaz **`kill`** [**`-`***signál*] *PID*
  - funkce **`kill`**
- ošetření signálu:
  - shell-příkaz **`trap`** [*příkaz*] *signal ...*
  - funkce **`signal`**
  - 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 (**^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

Synchronizace

- Pokud dva procesy sdílejí nějaký zdroj, je nutné současný přístup ke *kritickým sekcím* programů ošetřit zámkem
- Test zámku a jeho nastavení musí být *nepřerušitelná* dvojice operací provádět v privilegovaném režimu
- Synchronizace přes soubor:
  - program se pokusí vytvořit tzv. *lock soubor* - pokud se to nepodaří, je zdroj zamčen
  - po skončení programu se soubor smaže
  - problém: po havárii soubor nadále existuje
    řešení: do souboru se napíše PID procesu
  - problém: aktivní čekání na uvolnění zdroje

System V Interprocess Communication

- semafory:
  - zobecnění *P* a *V* operací [Dijkstra, Dekker]
  - ošetření *dead-lock*u, havárie procesu
  - funkce: **`semget`**, **`semop`**, **`semctl`**
- zasílání zpráv:
  - systém vytvoří komunikační kanál daného čísla
  - 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, INET: IP adresa, port
- **`listen`** zahájí příjem zpráv (mj. stanoví délku fronty)
- **`accept`** otevírá kanál ke klientovi
- **`connect`** navazuje spojení se servrem

Terminál

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

Řídící znaky

- některé lze předefinovat, některé závisí na shellu konzistence terminálu a shellu
- 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`**

Práce s časem

- spuštění programu s měřením času:

  `time` *command*

- pozastavení běhu:

  `sleep` *seconds*

- výpis aktuálního 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

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

- dtto pravidelné spouštění pomocí démona **`cron`**:

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

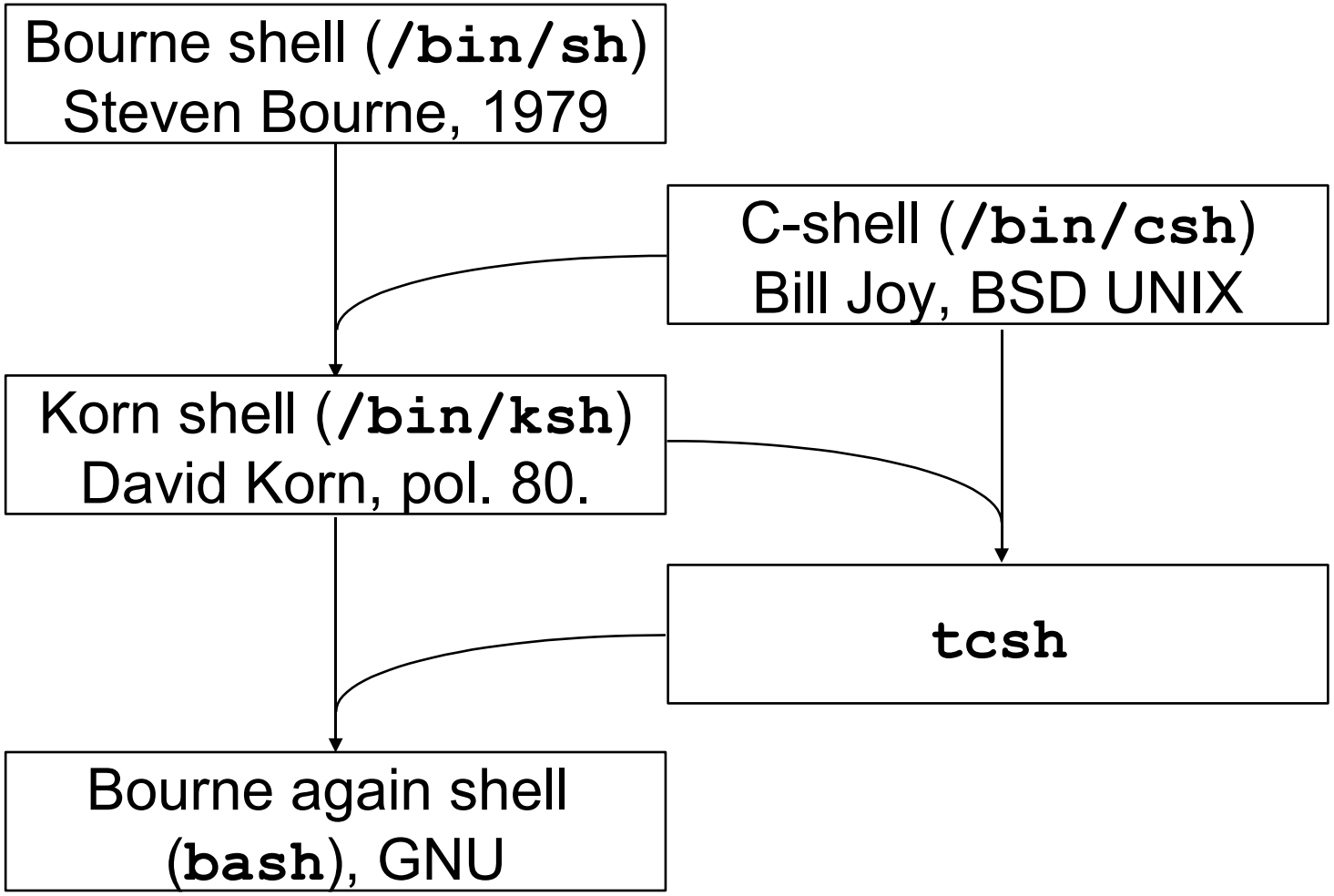
  příklad záznamu:

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

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
  - scripty

Vývoj shellů



Metaznaky

- znaky se speciálním významem (např. *****, **>**)
- speciální význam se ruší (tzv. *escape-sekvence*)
  - znakem ****
  - uzavřením do uvozovek (neruší význam ****, **$** a **`**)
  - uzavřením do apostrofů (ruší i význam ****)
- 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 (sh:`!`)

Bílé znaky a znaky `^`, `]`, `-` se do seznamu zapisují uvozené znakem `\`.

Expanzi provádí shell !
Expanze nezahrnuje úvodní tečku ve jméně souboru.

Základní příkazy shellu

`:` *comment*	- prázdný příkaz
`echo` [**`-n`**] *text*	- výpis řetězce (bez odřádkování)
`cd` [*dir*]	- změna adresáře (vlastnost shellu)
`pwd`	- výpis jména aktuálního adresáře
`exit` [*rc*]	- ukončení shellu s návratovým kódem
`set` *přepínač*	- nastavení přepínačů shellu
`ulimit` *limit*	- nastavení uživatelských limitů
`umask` [*mask*]	- nastavení defaultního módu souborů

Proměnné v shellu

name=*value* - nastavení hodnoty (**`sh`** a **`ksh`**)

name=*value cmd* - dočasné nastavení pro příkaz *cmd*

`$`*name*, **`${`***name***`}`** - použití hodnoty (textová substituce)

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

Proměnné jsou součástí *environmentu*.

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***}**	*value*	“”
${*name***:=***value***}**	**$***name*	*value* +nastavení *name*=*value*
${*name***:?***value***}**	**$***name*	“” **+echo** *value* a **exit**

Příkazové soubory - skripty

- přímé volání (práva **+rx**):

  script parametry

- volání přes shell (práva **+r**):

  sh [*optiony*] *script parametry*

- vložené volání (běží na stejné úrovni shellu):

  . *script*

- první řádek může obsahovat popis interpretu:

  #!*cesta_k_interpretu* [*optiony*]

- startovací skripty (běží na úrovni login-shellu):

  /etc/profile, **.profile**

Práce s parametry, speciální proměnné

`$#` - počet parametrů scriptu

`$0` - název scriptu

`$`*n* - *n*-tý parametr scriptu

`shift` [*n*] - posun číslování parametrů

`set - `*text* - nastavení nových parametrů

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

`$*` - všechny parametry scriptu

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

`$?` - návratový kód posledního příkazu

`$$` - PID tohoto shellu

`$!` - PID posledního backgroundového jobu

Přesměrování vstupu příkazu

zápis	přesměrování ...
cmd < *file*	vstupu ze souboru *file*

cmd << *str*	vstupu ze vstupu shellu (textu shellscriptu)
	vstup se chová jako text v uvozovkách
	př.:
	`ed xxx << END`
	`${cislo_radky}d`
	`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 $soubor > log 2>&1` přesměruje oba výstupy do souboru `log` - `grep xxx $soubor 2>&1 > log` výstup do souboru `log`, chyby na výstup

Kombinování příkazů

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

Příkaz **`read`**

- Příkazem **`read`** *var* načteme řádku ze vstupu do proměnné *var*
- Příkaz nastavuje návratový kód (dá se testovat)
- Pokud má příkaz více argumentů, čte postupně do jednotlivých proměnných pole vstupní řádky (do poslední proměnné zbytek); oddělovač polí udává hodnota proměnné **`IFS`**
- 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`**)
- Pokud vstupní řádka končí znakem **`\`**, načte se i následující a spojí se (zapne/potlačí option **`-e`**/**`-r`**)

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`
  ... nic (`SHOST` se nastaví v „synovi“)

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

Náhrada výstupu příkazu

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

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

  ⇓

  `SHOST=pocitac`

- vložený příkaz běží ve stejné úrovni shellu
- pozor na vnořené použití
  - nutno „escapovat“ vnitřní apostrofy
  - od **`ksh`** výše lze použít ...**`$(`***cmd***`)`**...

Vícenásobné čtení řádky

`eval` *arg* - argumenty se zpracují, spojí se mezerami, výsledek se znovu načte a provede

př.:
```
name=prvni
eval $name=retezec
```
⇓
```
prvni=retezec
```
- přiřadí do proměnné daného jména

př.:
```
eval echo \$$#
```
- vypíše hodnotu posledního parametru

Řídící struktury

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

```
case text 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.:

  špatně: **`[ -z = $x ]`**, **`[-z="$x"]`**

  správně: **`[ -z = "$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`**

`-f` *file*	- soubor *file* existuje
`-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
str1 **`!=`** *str2*	- nerovnost řetězců
int1 **`-eq`** *int2*	- rovnost čísel (**`-ne`**, **`-lt`**, **`-le`**, **`-gt`**, **`-ge`**)

Příkaz **expr**

- volání: **expr** *opndA op opndB*
- vypíše výsledek a vrací návratovou hodnotu
- shell nemá sám aritmetiku, práci s řetězci!
- logické operátory: **=**, **<**, **>**, **<=**, **>=**, **!=**
- aritmetické operátory: **+**, **-**, *****, **/**, **%**
- řetězcové operátory:
  - *string* **:** *regexp* resp. **match** *string regexp*
  - **substr** *string pos len*
  - **length** *string*
  - **index** *string chars*
- pozor na metaznaky

Řídící struktury - **`if`**

Syntaxe:

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

Příklad:

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

Řídící struktury - **case**

Syntaxe:

case *text* **in**
vzor1 **|** *vzor2* **)**
  příkazy;;
***)**
  příkazy;;
esac

Příklad:

```
case $1 in
-h | -\? | '' )
   echo "Navod: ..."
   exit;;
*.Z ) uncompress $1;;
* )   compress $1;;
esac
```

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

Syntaxe:

{**`while`** | **`until`**} *příkaz*
`do`
 příkazy
 `break`
 `continue`
`done` [**<** *file*] [**>** *file*]

Příklad:

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

Řídící struktury - **`for`**

Syntaxe:

`for` *var* **`in`** *text*
`do`
  příkazy
 `break`
 `continue`
`done`

Příklad:

```
lan=cz
for x in *; do
   case $x in
   *.$lan )
      cp $x ${x}_bak;;
   esac
done
```

Čtení vstupního souboru

- `while read x < file; do ...`
  ... čte stále první řádku
- `cat file | while read x; do`
  `  y=`expr $y + 1``
  `done`
  ... proměnná **y** se nastaví pouze v synovi
- `cat file | ( while read x; do`
  `  echo -n "Mam smazat $x? (a/[n]) "`
  `  read z`
  `  case $z in`
  `  a* | A* ) rm $x;;`
  `  esac`
  `  y=`expr $y + 1``
  `done; echo $y )`
  ... proměnná **z** se čte také ze souboru
- ... `read z < /dev/tty`

Řídící příkazy

cmd **&** - provedení na pozadí

wait - čekání na skončení procesu na pozadí

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

exec *cmd* - ukončí shell a provede příkaz

...počínaje **ksh** je možno přiřadit soubory deskriptorům bežícího exempláře shellu

Funkce

Definice funkce *name*:

name **() {**

statements

}

- volání+parametry stejné jako při volání příkazu
- uvnitř funkce jsou přístupné pomocí **`$#`**, **`$1`** atp.
- návratovou hodnotou je návratová hodnota posledního příkazu, lze nastavit: **`return`** *val*
- priorita: funkce, interní příkazy, programy
  interní příkaz lze vyvolat pomocí **`command`** *cmd*
- funkce se nedědí do subshellů

Ošetření signálů v shellu

- Interní příkaz: **`trap`** [*command*] *sig...*

- Příkaz *command* se provádí v rámci shellu
- Synovský proces nemá možnost ošetřit signály zamaskované otcem

- Zamaskování signálů: **`trap`** *"" sig...*
- Defaultní ošetření: **`trap`** *sig...*

Postup zpracování řádky

Postupuje se zleva doprava v následujících krocích:

1. rozdělení řádky na atomy
2. zpracování řídících operátorů
3. zpracování operátorů přesměrování

4-5. náhrada proměnných
náhrada vložených příkazů

6. definice proměnných
7. rozdělení na pole
8. náhrada expanzních znaků
9. zrušení escape-sekvencí

Volby (*options*) shellu

Volby se dají zadat

- z příkazové řádky při spuštění shellu
- na první řádce shell-scriptu
- příkazem **`set`**

Nedůležitější volby:

`-a` ... všechny proměnné jsou exportovány
`-e` ... chyba v příkazu způsobí ukončení shellu
`-f` ... zákaz expandování znaků
`-n` ... příkazy jsou pouze vypsány a neprovádí se
`-t` ... provede se pouze jeden příkaz
`-v` ... vypisují se vstupní řádky shellu
`-x` ... příkazy se před provedením vypisují

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
- **`{`***varA*`,`*varB*`}`** … výčet variant pro expanzi
- **`<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 opravováním 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`**

Filtr **awk**

- Aho, Weinberger, Kernighan
- jazyk podobný C, s výhradou:
  - interpret
  - práce s řetězci
- verze: **awk**, **nawk**, **gawk**
- volání:

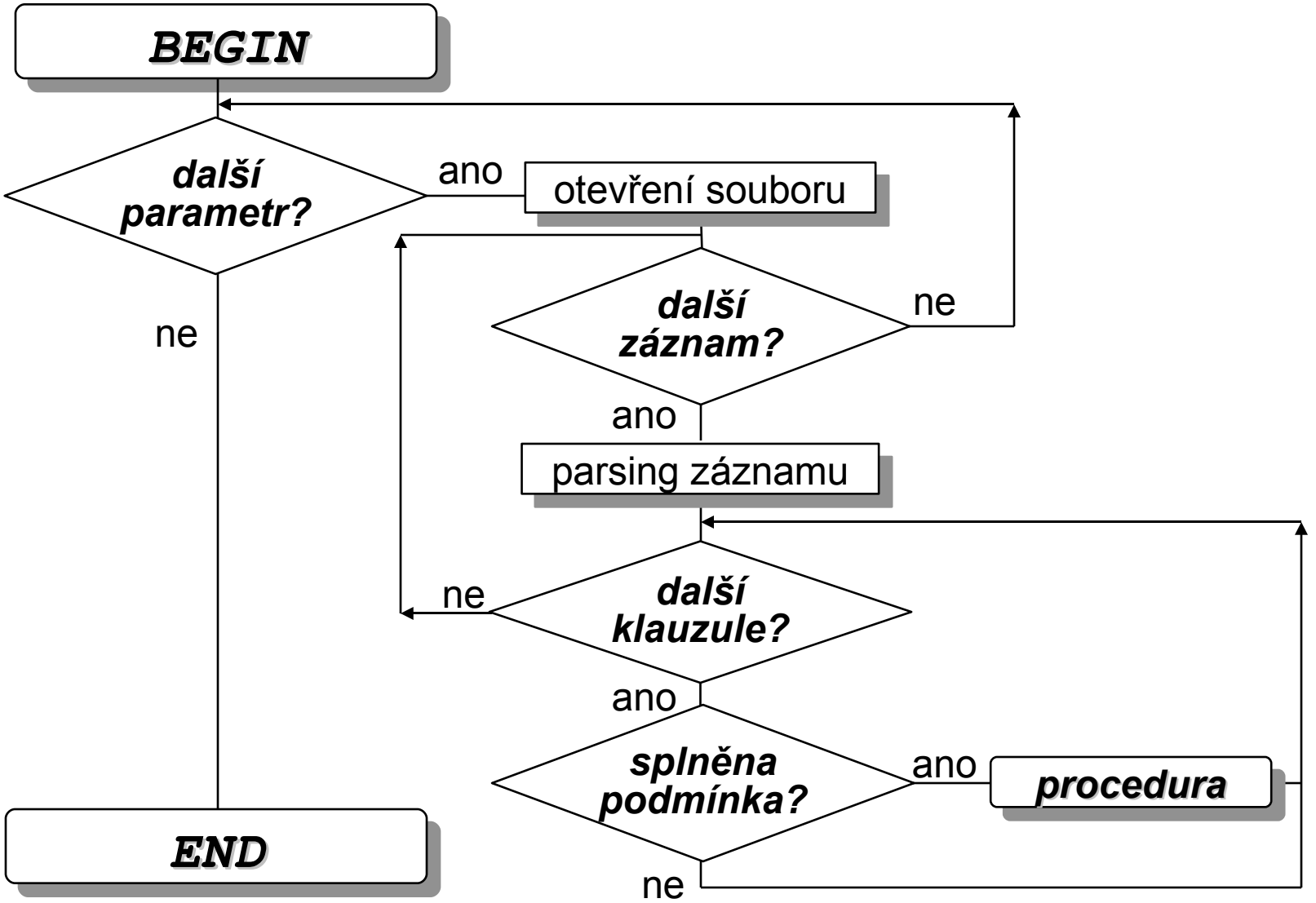
  awk [*opt*] {**-f** *script* | *pgm*} {*params* | *file* | **-**}...
- zpracovává postupně záznamy (řádky) zadaných souborů a provádí na ně příkazy awk-skriptu
- program (awk-skript) se skládá z klauzulí

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

Příklad programu **awk**

```
BEGIN    { procedur=0; lev=0 }
/procedure/ {
            print; lev=1
            radek=1; procedur++; next }
          { if( radek > 0 ) radek++ }
/begin/  { if( lev > 0 ) lev++ }
/end/    { lev-- }
lev == 1 && radek > 0
          { print "Radek:" radek; lev=0 }
END       { print "Procedur: ", procedur }
```

Diagram běhu programu (**awk**)



Regulární výrazy (**awk**)

Oproti regulárním výrazům v **ed**u chybí

- **\<**, **\>**, **\{**, **\}**, **\(**, **\)**, *****n*

Mění se mírně význam metaznaků

- **^**, **$** … začátek a konec záznamu/pole

Objevují se nové metaznaky

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

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

Záznamy (**awk**)

- Záznamem je typicky řádka
- Oddělovač záznamů je řetězec, změna: **RS=***string*
  - 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**)

- Oddělovač polí zadán při volání volbou **-F***regexp*, změna oddělovače: **FS=***regexp*
- Oddělovač je regulární výraz, default: **"[\t\n]+"**
  - 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*
- Počet polí: proměnná **NF**

Základní syntaxe **awk**

- Jazyk **awk** je **řádkově orientovaný**
- Příkazy se oddělují středníkem nebo koncem řádky, příkaz musí být na jedné řádce
- Má-li příkaz pokračovat na další řádce, musí předcházející řádka končit zpětným lomítkem (výjimka: za podmínkou např. příkazu **if** může následovat nová řádka)
- Komentář: text na řádce počínaje znakem **#**

Konstanty, proměnné, pole(**awk**)

- Konstanty
  - běžné aritmetické konstanty
  - řetězce se vyznačují uvozovkami
  - *escape* sekvence: **\b**, **\f**, **\n**, **\r**, **\t**, *****ooo*, **\x***xx*
- Proměnné
  - mají aritmetickou i řetězcovou hodnotu
  - jsou inicializovány
  - asociativní pole (indexem je řetězec): *var*`[`*item*`]`
  - (**nawk**) speciální *member* operátor: *item* **in** *var*
- Pole záznamu: **$***n*
  - např.: **`druhy=$2; n=5; paty=$n`**
  - pozor: **NF** (počet polí) a **$NF** (poslední pole)

Výrazy (**awk**)

- arimetické 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* **;** *test* **;** *step***)** *cmd* ... příkaz cyklu
- **for(***var* **in** *array***)** *cmd* ... příkaz cyklu
- **break**, **continue** ... ukončení (těla) cyklu
- **next** ... ukončení zpracování záznamu
- **exit** ... ukončení programu (skok na END)

Výstupní příkazy (**awk**)

- **print**
  tisk celé řádky, řádka ukončena **ORS** (defaultně **<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"`

Formátovací konvence

- Obecný tvar: `%`[*flags*][*width*][.*precision*]*type*
  - **`%c`**, **`%%`** ... výpis jednoho znaku, výpis procenta
  - **`%s`** ... výpis řetězce (počet znaků udává *precision*)
  - **`%u`**, **`%d`**, **`%o`**, **`%x`** ... celé číslo (bez znam., dek., okt., hex.)
  - **`%e`**, **`%f`**, **`%g`** ... reálné číslo
- Modifikátory:
  - `%`[**`-`**] *width* [**`.`***len*] **`s`** ... zarovnání vlevo, max. délka
  - `%`[**`+`**][**`0`**]*width fmt-spec* ... vynucení znaménka, ved. nul
  - `%`*width* [**`.`***precision*] *fmt-spec* ... přesnost reálných čísel
- Podobné formátovací konvence se používají i pro příkaz **`hexdump`** a funkci **`printf`** v jazyce C

Funkce v **awk**

- matematické funkce: **int**, **cos**, **sin**, **exp**, **log**, **sqrt**
- (**nawk**): **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 řetězec *s* na slova oddělená separátorem *sep* a přiřadí je do prvků pole *var*, vrací počet např.: `split("194.50.16.1",ip,".")`
  - **sprintf(***fmt*,*...***)** ... vrací formátovaný text jako řetězec
  - **substr(***s*,*p*[,*n*]**)** ... vrací podřetězec *s* délky *n* od pozice *p*
- (**nawk**): **match**, **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 ...*

Komunikace se systémem (**awk**)

- předávání parametrů z příkazové řádky:
  - př.: `awk '{print > "'$DIR'/log"}'`
- nastavování proměnných z příkazové řádky (**nawk**):
  - př.: `awk var=value1 file1 var=value2 file2`
- proměnné environmentu (**nawk**): pole **ENVIRON**
  - př.: `print > ENVIRON["HOME"] "/log"`
- změna proměnné environmentu: nelze
  - např.: `eval 'awk '{print "PATH=" path}''`
- volání příkazu: funkce **system(***command***)**
  - př.: `system( "rm " filename )`
  - funkce vrací návratovou hodnotu, ale ne výstup
  - příkaz pracuje v subshellu!

Konfigurace programu (**awk**)

- Příklad předání více parametrů přes **echo**:

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

- Příklad použití konfiguračního souboru:

```
awk -F: '/^-CFG:/ { cfg[$2]=$3; next }
{ # pro ostatni radky
}' cfg /etc/passwd
```

Soubor **cfg** obsahuje řádky tvaru

```
-CFG:low:12
```

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

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

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

C - konvence

`*.c`	jména zdrojových souborů programů v C
`*.h`	jména hlavičkových souborů (*header*ů)
`*.o`	přeložené moduly (*object-moduly*)
`a.out`	jméno proveditelného souboru jako výsledek úspěšné kompilace
`/usr/include`	kořen stromu systémových headerů
`/usr/lib/lib*.a`	umístění knihoven object-modulů

Kompilátor

Volání:

`cc` [*options*] *soubor...*

Nejdůležitější volby:

`-o` *soubor*	výstupní jméno
`-c`	pouze překlad (nelinkovat)
`-E`	pouze preprosesor (nepřekládat)
`-O`*level*	nastavení úrovně optimalizace
`-g`*level*	nastavení úrovně ladicích informací
`-D`*jméno*	definuj makro pro preprocesor
`-U`*jméno*	oddefinuj makro pro preprocesor
`-I`*adresář*	umístění **`#include`** souborů
`-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

Program **make**

- generátor příkazů
- správa SW projektů
- příklad (soubor **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*
- pokračování vkládá mezeru
- nedefinovaná makra jsou prázdná
- nezáleží na pořadí definic
- nelze předefinovat
- definice na příkazové řádce:

  `make` *target name*=*string*
- vyvolání makra:

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

The End