

# *Programozás alapjai 2*

## *UNIX dióhéjban*

Szeberényi Imre

BME IIT

<szebi@iit.bme.hu>



MUEGYETEM 1782

# *Legfontosabb tulajdonságai*

---

- többfelhasználós (multiuser)
- időosztásos (time sharing)
- hardware független
- nyílt rendszer
  - gyártófüggetlen protokollokon, eljárásokon, és szabványokon alapul.
  - nem kötődik egyetlen gyártóhoz sem

# *Legfontosabb tulajdonságai/2*

---

- szorosan kötődik az Internet technológiákhoz
- mikroszámítógéptől mainframe-ig és szuperszámítógépig minden hardware platformra implementáltak
- sok segédprogram, melyek lehetővé teszik komplex feladatok megoldását is
- I/O eszközök egységes kezelése
- hierarchikus állományrendszer

# *A UNIX rövid története*

- A UNIX nem új, de mindig megújul (több, mint 50 éves)
- Bell Laboratories (1968-1974)  
Saját célra szoftver fejlesztői környezet. (Denis Richie, Ken Thompson, PDP-7)
- V6, V7 (C nyelv); Egyetemek, kutatóintézetek (sok ötlet, felhasználók igényei szerint, inkompatibilitás).



# *A UNIX rövid története/2*

---

- két fő irányzat:
  - AT&T SystemV,
  - BSD
- minden nagy gyártó saját implementációval rendelkezett (SVR4, OSF/1, BSD, Linux)
- szabványosítási törekvések (Posix, X/Open portability guide)
- legnépszerűbb PC-n futó változatai: Linux, FreeBSD, SCO, Solaris

# *Bejelentkezés*

---

- Felhasználó azonosítás grafikus, vagy alfanumerikus felületen keresztül:
  - login
  - password
- Jelszó változtatás: `passwd`
- Kedvelt ssh kliens a `putty`
  - fontos a helyes beállítás:
    - keyboard
    - char set (translation)

# *Bejelentkezés/2*

- Bejelentkezés után normál esetben a login v. home katalógusban elindul a parancsértelmező (shell).
- Ez értelmezi a felhasználói parancsokat és indítja a további processzeket (processz = futó program).
- Több shell alakult ki, melyek elsősorban programozói szempontból különböznek.
- A parancsértelmező kezelése kényelmetlen, ha nincs rendesen beállítva a környezet.
- Nagyon fontos a terminál típusának beállítása, annak összhangja a terminál emulátorral. (set term=, TERM=)

# *A UNIX állományrendszere*

- Az állományrendszer hierarchikus, fa
- Legfontosabb tulajdonságok:
  - egyetlen gyökér (root) van a rendszerben, a kötetek (adathordozók) számától függetlenül
  - Fájlok legfontosabb csoportja:
    - egyszerű (plain) (jele: -)
    - katalógus (directory) (jele: d)
    - periféria (device) (jele: c vagy b)
  - további csoportok:
    - socket (jele: s)
    - named pipe (jele: p)
    - szimbolikus link (jele: l)



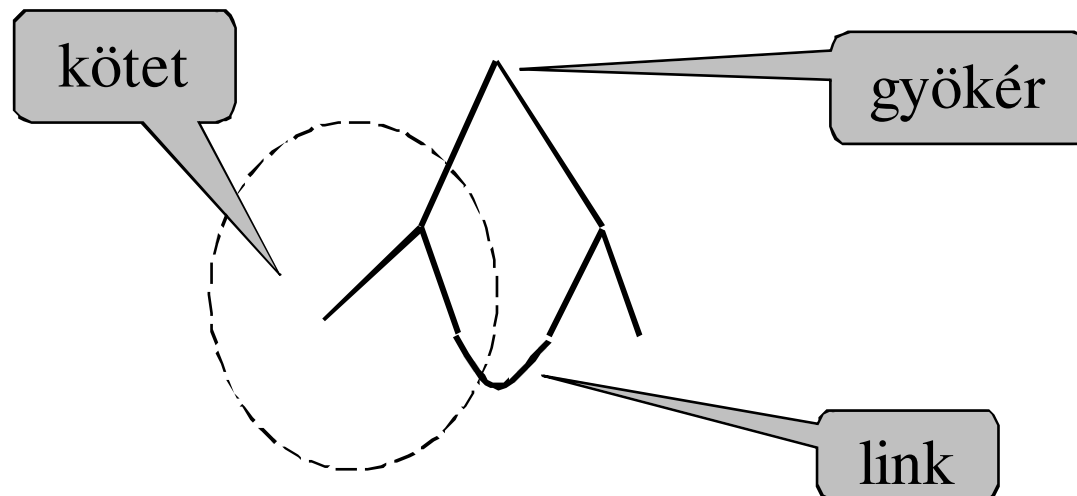
# *A UNIX állományrendszere/2*

---

- Katalógusok, perifériák is fájlként látszanak, általában teljesen azonos módon kezelhetők az egyszerű állományokkal (ugyanazok a rendszerhívások: open, close, read, write, stb.)
- Állományokat sem tartalmuk, sem nevük alapján nem kell megkülönböztetni (azaz: nincs külön szöveg fájl vagy bináris fájl, nem a fájl neve vagy kiterjesztése határozza meg a fájl funkcióját)

# A UNIX állományrendszere/3

- A fa struktúrában keresztkapcsolatok (link) hozhatók létre. A link két változata:
  - Hard link
  - Szimbolikus link
- A fájlok elnevezési szabályai rugalmasak



# *Védelmi rendszer*

---

- Többfelhasználós rendszer, védeni kell:
  - fájlok adatait
  - processzek adatait
- A védelem kiterjed:
  - olvasásra
  - írásra
  - végrehajtásra

# Védelmi rendszer/2

---

- A bejelentkezéskor felhasználó azonosítás történik és meghatározásra kerül az adott felhasználó
  - UID-ja (felhasználó azonosító)
  - GID-je (csoport azonosító)  
(A felhasználók csoportokba oszthatók. Egy felhasználó több csoportba is tartozhat)
- A felhasználó által létrehozott minden processz ezt örökli.

# Védelmi rendszer/3

---

- Minden állománynak van:
  - tulajdonosa (rendszerint aki létrehozta)
  - felhasználói csoportja
  - legalább 3x3-as védelmi kódja
- Ez utóbbi megadja, hogy a
  - tulajdonos
  - csoporttárs
  - bárki másmilyen műveletet végezhet az állományon.

# Védelmi rendszer/4

- Állományokon végezhető műveletek:
  - olvasás (r)
  - írás (w)
  - végrehajtás/keresés (x)
- Az ls parancs a védelmi kódot betűkkel jeleníti meg pl:

```
-rwxr-xr-- 1 joska tanulo ....
```

# Védelmi rendszer/5

- Védelmi szempontból a katalógusok nem különböznek a fájlaktól. A katalógust egy cédulának kell tekinteni, amire a fájl neve van felírva. Egy fájl védelme nem függ az azt tartalmazó katalógus védelmétől.

Pl:

drwxrwxrwx	2	joska . . . . munka
-r--r--r--	1	joska . . . . levelem

# Védelmi rendszer/6

- Az előzőek alapján a fájl módosításához nem kell írási jog a befogadó katalógusra.

Pl:

dr-xr-xr-x	2	joska . . . . munka
-rw-rw-rw-	1	joska . . . . levelem



# *chmod parancs*

---

`chmod okt_szám file1 file2 ...`

`chmod oug+ -=rwxstougX file ...`

Pl:

`chmod 640 uborka`

`chmod -w korte`

`chmod g+w alma`

`chmod oug=rw`

# *Fájlkezeléssel kapcs. parancsok*

- cd (change dir) katalógus váltás
- pwd (print w. dir.) munkakatalógus kiírása
- mkdir (make dir.) katalógus létrehozása
- rmdir (remove dir.) katalógus törlése
- ls (list) katalógus lista
- cp (copy) fájl(ok) másolása
- rm (remove) fájl(ok) törlése
- mv (move) fájl(ok) áthelyezése
- ln (link) fájl hivatkozás készítés
- ln -s szimbolikus hivatkozás készítése

## *Fájlkezeléssel kapcs. parancsok/2*

- `chmod` (ch. mode)      védelmi kód változtatás
- `chown/chgrp`              tulajdonos megváltoztatás
- `mount/umount`              kötet csatlakoztatás

Sok fájlkezeléssel kapcsolatos parancs `-r` (`-R`) kapcsolóra rekurzív lesz (pl. `cp`)

# *On-line manual*

---

man

man man

man -k

man 1 cp, vagy man -s 1 cp

GNU:

info

# Shell

---

- Parancsértelmező (shell) értelmezi a felhasználói parancsokat. Ez független attól, hogy grafikus, vagy alfanumerikus felületen dolgozunk.
- Több shell alakult ki, melyek programozói szempontból és a kényelmi szolgáltatásokban különböznek.

# *Shell legfontosabb tulajdonságai*

---

- Parancsértelmező és programozási nyelv.
- Egyszerűen kialakítható ún. shell szkript vagy parancs fájl, amivel a parancskészlet bővíthető.
- Egyszerű szintaxis a standard input-output átirányítására.
- Csővezeték segítségével komplex feladatok megoldása a meglévő segédprogramokkal.
- Egyénileg konfigurálható, testre szabható.

# *Shell-ek beállítása*

---

- Több jól definiált ponton speciális parancsállományok futtatására van lehetőség:
  - login
  - start
  - logout
  - távoli vagy nem interaktív

# Shell-ek beállítása/2

shell	login	start	logout	rsh, nem interakt.
<b>sh</b>	/etc/profile \$HOME/.profile			
<b>ksh</b>	/etc/profile \$HOME/.profile			
<b>csh</b>	/etc/csh.cshrc /etc/csh.login ~/.cshrc ~/.login	/etc/csh.cshrc ~/.cshrc	/etc/csh.logout ~/.logout	/etc/csh.cshrc ~/.cshrc
<b>tcsh</b>	/etc/csh.cshrc /etc/csh.login ~/.tcshrc    ~/.cshrc ~/.login	/etc/csh.cshrc ~/.tcshrc    ~/.cshrc	/etc/csh.logout ~/.logout	/etc/csh.cshrc ~/.cshrc
<b>bash</b>	/etc/profile ~/.bash_profile   ~/.bash_login    ~/.profile	~/.bashrc	~/.bash_logout	\$BASH_ENV ~/.bashrc



# *Parancsok általános felép.*

## **parancs kapcsolók argumentumok .....**

- Kapcsolók: parancs működését, eredményét módosító paraméterek. Többnyire "-"-szal kezdődnek. (ls -l).
- Általános szabály, hogy ha a fájlnev helyén – áll, akkor az a szabványos be/kimenetet jelöli. (pl: cmp – f1).

# *Parancs és parancssor*

---

- Egy parancssor több parancsot is tartalmazhat.
  - **prog1 ; prog2**
  - **prog1 || prog2**
  - **prog1 && prog2**
- Sőt egy parancssor több sorból is állhat. (folytatósor)
- Egy parancs is lehet többsoros (pl: for)

# *Shell I/O átirányítás*

---

- `prog > file`
- `prog >> file`
- `prog < file`
- `prog << VEGE`  
ezt a prog megkapja a standard bemenetén  
a VEGE végjelig. Ez a „here” dokumentum.  
VEGE
- `prog1 | prog2`

# *Shell parancshelyettesítés*

---

- Lehetővé teszi, hogy egy program kimenete a másik program indítási paramétere legyen:

```
prog1 `prog2`  
more `grep -l alma *.txt`
```

# *Shell állománynév helyettesítés*

---

- FONTOS: állománykezelő szinten nincs kiterjesztés, ezért a . (pont) a név része.
- prog \* - tetszőleges számú bármi
- prog ? - 1 db bármi
- prog [abc] - a, vagy b, vagy c
- prog [a-z] - egy az a-z intervallumból
- prog ~/file - home\_kat/file
- prog ~user\_nev/file - user\_home\_kat/file

# *Shell állománynév helyettesítés/2*

---

- Nincs korlátozás a helyettesítő karakterek alkalmazására.

```
ls A?[0-9]*z
```

```
ls ~/**A**
```

- Ha a programra szeretnénk bízni a kezelést, le kell „takarni”.

```
ls "**A**"
```

# Speciális jelentés megszüntetése

Szinte minden jel speciális:

\* ? [ ] < > | & \$ { } ; ( ) ' " ` \

	'	"	`	\	\$	*
'	t	n	n	n	n	n
"	n	t	i	i	i	n
`	i	i	t	i	i	i

**t** terminális

**n** nem értelmeződik speciálisan

**i** értelmeződik

# *Reguláris kifejezések*

---

- Szövegfeldolgozás jellemző feladat a programfejlesztői környezetekben.
- A szövegre illeszkedő minta egységes megadására találták ki.
- Segítségével bonyolult minták és illesztési szabályok adhatók meg.
- Sajnos nem teljesen egységes, mert van egy bővített változata is.



## *Reguláris kifejezések/2*

- c önmagát a c karaktert jelenti.
- \c "c", ha c nem számjegy és nem (, ), {, }.
- ^ sor elejére illeszt.
- \$ sor végére illeszt.
- . egy db tetsz. kar. (kivéve újsorjel).
- [abc] "a", "b", vagy "c"
- [^abc] bármi ami nem "a", "b", vagy "c"

## Reguláris kifejezések/3

$r^*$   $r$  atom nullszor vagy sokszor ismétlődhet

$r_1r_2$   $r_1$  és  $r_2$  atom egymás után illesztendő úgy,  
hogy  $r_1$  a lehető leghosszabban illeszkedjék.

$r^+$   $r$  atom egyszer vagy sokszor ismétlődhet

$r^?$   $r$  atom 0-szor vagy 1-szer ismétlődhet

$r_1|r_2$   $r_1$  vagy  $r_2$  atom közül csak az egyiknek  
kell illeszkednie

Ez utóbbi 3 csak egrep, awk, lex, perl esetében

## *Reguláris kifejezések/4*

$\backslash(r\backslash)$   $r$  reguláris kifejezésre később hivatkozni lehet a  $\backslash n$  alakban, ahol  $n$  egy számjegy.

$\backslash n$  hivatkozás az  $n$ .  $\backslash(r\backslash)$  alakú reguláris kif.-re

$r\{m,n\}$   $r$  atom minimum  $m$ -szer, de maximum  $n$ -szer ismétlődhet

$r\{m\}$   $r$  atom pontosan  $m$ -szer ismétlődhet

$r\{m,\}$   $r$  atom legalább  $m$ -szer ismétlődhet

Ezek csak ed, sed és perl esetében.

## *Reguláris kifejezés példák*

- Az **a1ma** azt jelenti, hogy az alma minta a soron belül bárhol előfordulhat.
- A **^a1ma** előírja, hogy az alma mintának a sor elején kell előfordulnia.
- A **^[mh]?a1ma** azt jelenti, hogy sor elején alma vagy malma vagy halma mintának kell előfordulnia.

## *Reguláris kifejezés példák/2*

- A `^[^mh]alma` azokra a sorokra illeszkedik, amelyen nem malma, vagy halma sorozattal kezdődnek, de ?alma mintával kezdődnek.
- A `^\([abc]\)\1` azokra a sorokra illeszkedik, amelyek vagy aa, bb vagy cc kezdetűek. (\1 szerepe itt az, hogy az [abc] szabad paraméterek közül az aktuálisan illeszkedőt kijelölje.

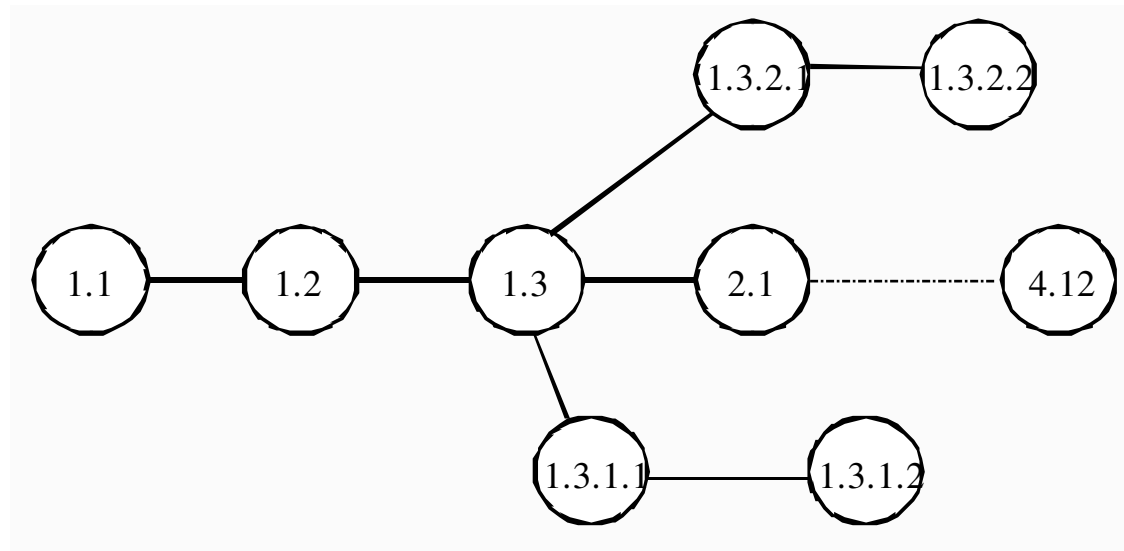
# *Verziókezelő rendszerek*

---

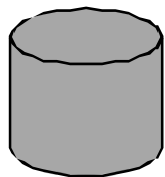
- Együtt kezelik különböző verziókat. Nincs szükség változatos nevű állományokra.
- Vannak operációs rendszerek, melyek ehhez támogatást adnak, de....
- Segítik a verziók adminisztrálását
  - SCCS
  - RCS
  - CVS
  - SVN
  - GIT, HG, GITHUB, GITLAB ...

# Verziókezelő rendszerek/2

Jellemző életrajz: R.L.b.1



Távoli tároló:



Lokális másolatok:

