

Programozás Alapjai

Linux alapismeretek

Dr. Gergely Tamás

Dr. Jász Judit

Szegedi Tudományegyetem
Informatikai Intézet
Szoftverfejlesztés Tanszék

2023

(v0914)



1

Linux

- Alapfogalmak

- Linux parancsok
- Linux shell

- Felhasználók
- Hálózat



1

Linux

Alapfogalmak

- Linux parancsok
- Linux shell

- Felhasználók
- Hálózat



Linux alapismeretek

Miért?

- Egy informatikus számítógépekkel dolgozik, így nem engedheti meg magának, hogy ne tudja kezelni azt.
- Konkrétabban: a gyakorlaton programokat kell majd írni gépek segítségével, és ezeket tudni kell kezelni.
- Minél több rendszert ismer valaki, annál átfogóbb képet kap, és könnyebben ismerkedik újakkal. (Van Windows is másik kurzuson.)



Miért a Linux operációs rendszer?

- Sok helyen nagyon elterjedt:
 - 2017. november 1. óta a Top 500 mainframe 100%-án linux alapú rendszer fut.*
 - Beágyazott rendszerekben (SetTopBox-ok, TV-k, routerek, Raspberry, Arduino) elterjedt.
 - Az Android is Linux (és Java) alapokra épül.
- A Unix alapok miatt szorosan összefonódik a C nyelvvel.
- A parancssor-orientáltság miatt egyszerűbben lehet programokat írni (kisebb a belépési küszöb).
- A root nem „csak” egy adminisztrátor.

*<https://www.top500.org/statistics/details/osfam/1/>

„A Linux csak egy kernel, az operációs rendszer a GNU.”

– Richard M. Stallman

- A GNU/Linux egy POSIX szabványokat követő Unix-szerű operációs rendszer
 - GNU = „**G**NU's **N**ot **U**nix”
 - Többtaszkú, többfelhasználós, virtuális memóriával, védett üzemmóddal, korszerű memóriakezeléssel, megosztott programkönyvtárakkal, demand paging mechanizmussal, széles körű TCP/IP hálózati támogatással, stb.
- És ez mit jelent?
 - Most annyit, hogy egy normálisan használható operációs rendszer . . .
 - . . . de különböző kurzusokon (pl. operációs rendszerek) mindegyik kifejezés el lesz magyarázva.

- Olyan programrendszer, amely közvetítő szerepet tölt be a számítógép hardver erőforrásai és a felhasználó között.
- Főbb funkciói:
 - Programok betöltése és végrehajtása
 - Erőforrások elosztása
 - Input/output műveletek végzése
 - Háttértárakon tárolt adatrendszerek kezelése
 - A felhasználó által kiadott parancsok értelmezése és végrehajtása
 - A működés közben fellépett hibák lekezelése



- Az adatok (szövegek, képek, hangok, programok, stb.) hosszú távú tárolására a háttértárak szolgálnak.
- Tárolási egységek
 - bit** A legkisebb tárolási egység, 0 vagy 1 értékkel.
 - bájt** 8 bitnyi információ $2^8 = 256$ értékkel.
 - fájl** A szorosan egymáshoz tartozó, egy összetett adatot alkotó bájtokat a háttértárakon egy fájlban tároljuk.
 - könyvtár** A fájlokat a háttértárakon könyvtárakban tároljuk.



- A háttértárolón tárolt adatok logikailag egy fa szerkezetben vannak tárolva.
 - Gyökér (root) könyvtár
 - Alkönyvtárak
 - Fájlok
 - Aktuális könyvtár



fájl szerkezet	út vonal
/	/
+ bin	/bin
+- bash	/bin/bash
+- cp	/bin/cp
:	
+ boot	/boot
+ dev	/dev
+ etc	/etc
+ home	/home
+- gertom	/home/gertom
+- .bashrc	/home/gertom/.bashrc
+- gyakorlat	/home/gertom/gyakorlat
:	
+- h531674	/home/h531674
:	
+ lib	/lib
+ root	/root
+ tmp	/tmp
+ usr	/usr
+ var	/var

- Aktuális könyvtár
 - Ebben dolgozunk. Bármely kiadott parancs itt hajtódik végre, a relatív útvonalak innen indulnak. (TIK (N46.246970, E20.142445))
- Relatív útvonal
 - Egy fájl vagy könyvtár nevének megadása az aktuális könyvtárhoz képest. (Dóm (+0.001992, +0.006720))
 - gyakorlat, ..., ../h531674
- Abszolút útvonal
 - Egy fájl vagy könyvtár nevének megadása a gyökérkönyvtárból kiindulva. (Dóm (N46.248962, E20.149165))
 - /home/gertom/gyakorlat, /home, /home/h531674

Aktuális könyvtár abszolút útvonala	Relatív útvonal	Abszolút útvonal
/home/gertom	gyakorlat	/home/gertom/gyakorlat
/home/gertom	..	/home
/home/gertom	../h531674	/home/h531674
/bin	bash	/bin/bash
/bin	..	/
/bin	../etc	/etc

- Speciális jelentésű fájlnevek
 - / A könyvtárhierarchia gyökere.
 - .. A hierarchiában egy szinttel feljebb lévő könyvtár.
 - . Az aktuális könyvtár.
- A .-tal kezdődő nevek rejtett fájlt vagy könyvtárat jeleznek. Ezek bizonyos műveletek esetén rejtve maradnak, azaz nem veszünk róluk tudomást.



- A kabinetes gépeken egy `valaki` nevű felhasználó létezik, aki jelszó nélkül tud bejelentkezni. A home könyvtára a lokális gépen létező `/home.local/valaki`.
- A `mounthome` szkript futtatása után csatolódik fel a központi szerveren lévő saját hallgatói home könyvtár a `/home/hxxxxxx` pontra. Ez az `umounthome` szkript futtatásáig, vagy a rendszer újraindításáig elérhető marad!



1

Linux

Alapfogalmak

Linux parancsok

Linux shell

Felhasználók

Hálózat



Linux parancsok

„De miért?”

- Linux-on ez a „természetes”
- A parancsértelmező nyelve egy olyan programozási nyelvnek is felfogható, aminek a linux parancsok az utasításai
- Távolról, grafikus felület nélkül is használható
- Tudni kell parancssorból programot fordítani, amihez nem árt, ha valaki parancssorból tudja kezelni a fájlrendszert
- Néha egyszerűbb és gyorsabb egy paranccsal elintézni valamit, mint végigkattintgatni a grafikus felület húsz elemén



Linux parancsok

Általános alak

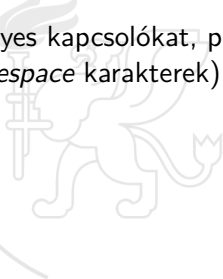
- Egy parancs általános alakja:
 - parancs kapcsolók paraméterek

parancs A parancs (vagy program) neve.

kapcsolók Általában kötőjellel kezdődő paraméter, ami a parancs/program működését befolyásolja.

paraméterek A parancs paramétereit, adatok, amiket fel fog dolgozni.

- Az egyes kapcsolókat, paramétereket szóközzel vagy tabulátorral (*whitespace* karakterek) választjuk el egymástól.



Linux parancsok

A fájlrendszer műveletei

pwd **print working directory** – az aktuális könyvtár lekérdezése

ls **list** – könyvtár tartalmának kilistázása

cd **change directory** – az aktuális könyvtár megváltoztatása

mkdir **make directory** – könyvtár létrehozása

rmdir **remove directory** – könyvtár törlése

mv **move** – fájl/könyvtár mozgatása/átnevezése

cp **copy** – fájl (vagy könyvtár) másolása

rm **remove** – fájl (vagy könyvtár) törlése

ln **link** – fájl (vagy könyvtár) linkelése

chmod **change mode** – fájl (vagy könyvtár) hozzáférési jogainak megváltoztatása

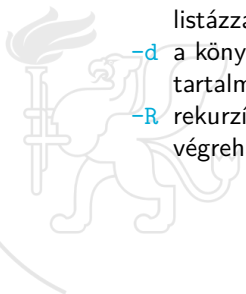
Linux parancsok

pwd, cd

- pwd
 - Kiírja az aktuális könyvtár abszolút elérési útvonalát.
- cd <könyvtár>
 - Ezentúl a <könyvtár> lesz az aktuális könyvtár.
 - Ha nem adunk meg paramétert, akkor a *home* könyvtárunk lesz az aktuális.



- `ls <kapcsolók> <lista>`
 - Kijelentés a listában megadott fájlok tulajdonságait és könyvtárak tartalmát.
 - Ha nincs `<lista>` az olyan, mintha a `.` lett volna a paraméter.
 - Fontosabb kapcsolók
 - `l` részletes lista sok tulajdonsággal
 - `a` a listában szereplő könyvtárak rejtett bejegyzéseit is listázza
 - `d` a könyvtárak tulajdonságait listázza, és nem a tartalmukat
 - `R` rekurzív mód, az alkönyvtárak minden bejegyzésére végrehajtja ugyanezt a parancsot



Linux parancsok

mkdir, rmdir

- `mkdir <kapcsolók> <lista>`
 - Létrehozza a listában megadott könyvtárakat. Hosszabb útvonal esetén csak az utolsó elemet, a többi létezőnek tekinti.
 - Fontosabb kapcsolók
 - p a teljes megadott útvonalat megpróbálja létrehozni
- `rmdir <kapcsolók> <lista>`
 - Törli a listában megadott könyvtárakat. Hosszabb útvonal esetén csak az utolsó elemet.
 - Nemüres könyvtárat nem töröl.
 - Fontosabb kapcsolók
 - p a teljes megadott útvonalat megpróbálja törölni

- `mv <kapcsolók> <lista> <újhely>`
 - Átmozgatja a `<lista>` elemeit az `<újhely>` könyvtárba.
 - Ha az `<újhely>` létező könyvtárat jelöl, akkor ebbe mozgatja a `<lista>` elemeit (fájlokat és könyvtárakat is).
 - Ha az `<újhely>` nem létezik, akkor a `<lista>` egyelemű kell legyen, és átnevezés (is) történik.
 - Fontosabb kapcsolók

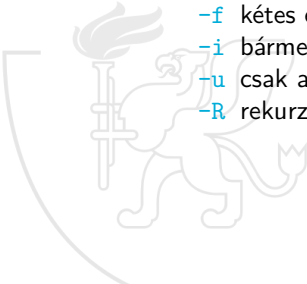
`-f` kétes esetben sem kérdez vissza

`-i` bármely kétes esetben megerősítést vár

`-u` csak a régebbi fájlokat írja felül



- cp <kapcsolók> <lista> <újhely>
 - Átmásolja a <lista> elemeit az <újhely> könyvtárba.
 - Ha az <újhely> létező könyvtárat jelöl, akkor ebbe mozgatja a <lista> elemeit.
 - Ha az <újhely> nem létezik, akkor a <lista> egyelemű kell legyen, és ezen az új néven jön létre a másolat.
 - Fontosabb kapcsolók

- 
- f kétes esetben sem kérdez vissza
 - i bármely kétes esetben megerősítést vár
 - u csak a régebbi fájlokat írja felül
 - R rekurzívan a teljes könyvtárstruktúrát lemásolja

- `rm <kapcsolók> <lista>`
 - Törli a listában megadott fájlokat.
 - A könyvtárakat alapesetben átugorja.
 - Fontosabb kapcsolók
 - f kétes esetben sem kérdez vissza
 - i bármely kétes esetben megerősítést vár
 - R rekurzívan a teljes könyvtárstruktúrát törli

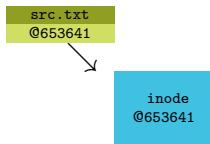
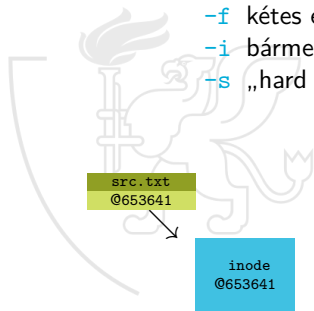


- `ln <kapcsolók> <lista> <újhely>`
 - Létrehoz a <lista> elemeire mutató link(ek)et az <újhely> helyen.
 - Ha az <újhely> létező könyvtárat jelöl, akkor ebben hozza létre a <lista> elemeire mutató linkeket.
 - Ha az <újhely> nem létezik, akkor a <lista> egyelemű kell legyen, és ezen az új néven jön létre a link.
 - Fontosabb kapcsolók

-f kétes esetben sem kérdez vissza

-i bármely kétes esetben megerősítést vár

-s „hard link” helyett „soft link”-(ek)et készít



- `ln <kapcsolók> <lista> <újhely>`
 - Létrehoz a <lista> elemeire mutató link(ek)et az <újhely> helyen.
 - Ha az <újhely> létező könyvtárat jelöl, akkor ebben hozza létre a <lista> elemeire mutató linkeket.
 - Ha az <újhely> nem létezik, akkor a <lista> egyelemű kell legyen, és ezen az új néven jön létre a link.
 - Fontosabb kapcsolók

-f kétes esetben sem kérdez vissza

-i bármely kétes esetben megerősítést vár

-s „hard link” helyett „soft link”-(ek)et készít

```
ln -s src.txt dst.txt
```

src.txt
@653641

inode
@653641

```
ln src.txt dst.txt
```

src.txt
@653641

inode
@653641

- `ln <kapcsolók> <lista> <újhely>`
 - Létrehoz a <lista> elemeire mutató link(ek)et az <újhely> helyen.
 - Ha az <újhely> létező könyvtárat jelöl, akkor ebben hozza létre a <lista> elemeire mutató linkeket.
 - Ha az <újhely> nem létezik, akkor a <lista> egyelemű kell legyen, és ezen az új néven jön létre a link.
 - Fontosabb kapcsolók

-f kétes esetben sem kérdez vissza

-i bármely kétes esetben megerősítést vár

-s „hard link” helyett „soft link”-(ek)et készít

```
ln -s src.txt dst.txt
```

```
ln src.txt dst.txt
```

src.txt
@653641

dst.txt
@src.txt

inode
@653641

src.txt
@653641

dst.txt
@653641

inode
@653641

- `ln <kapcsolók> <lista> <újhely>`
 - Létrehoz a <lista> elemeire mutató link(ek)et az <újhely> helyen.
 - Ha az <újhely> létező könyvtárat jelöl, akkor ebben hozza létre a <lista> elemeire mutató linkeket.
 - Ha az <újhely> nem létezik, akkor a <lista> egyelemű kell legyen, és ezen az új néven jön létre a link.
 - Fontosabb kapcsolók

-f kétes esetben sem kérdez vissza

-i bármely kétes esetben megerősítést vár

-s „hard link” helyett „soft link”-(ek)et készít

```
ln -s src.txt dst.txt
rm src.txt
```

src.txt
@653641



dst.txt
@src.txt

inode
@653641

```
ln src.txt dst.txt
rm src.txt
```

src.txt
@653641



dst.txt
@653641



inode
@653641

- `ln <kapcsolók> <lista> <újhely>`
 - Létrehoz a <lista> elemeire mutató link(ek)et az <újhely> helyen.
 - Ha az <újhely> létező könyvtárat jelöl, akkor ebben hozza létre a <lista> elemeire mutató linkeket.
 - Ha az <újhely> nem létezik, akkor a <lista> egyelemű kell legyen, és ezen az új néven jön létre a link.
 - Fontosabb kapcsolók

-f kétes esetben sem kérdez vissza

-i bármely kétes esetben megerősítést vár

-s „hard link” helyett „soft link”-(ek)et készít

```
ln -s src.txt dst.txt  
rm src.txt
```

```
ln src.txt dst.txt  
rm src.txt
```

dst.txt
@src.txt

inode
@653641

dst.txt
@653641

inode
@653641

- `ln <kapcsolók> <lista> <újhely>`
 - Létrehoz a <lista> elemeire mutató link(ek)et az <újhely> helyen.
 - Ha az <újhely> létező könyvtárat jelöl, akkor ebben hozza létre a <lista> elemeire mutató linkeket.
 - Ha az <újhely> nem létezik, akkor a <lista> egyelemű kell legyen, és ezen az új néven jön létre a link.
 - Fontosabb kapcsolók

`-f` kétes esetben sem kérdez vissza

`-i` bármely kétes esetben megerősítést vár

`-s` „hard link” helyett „soft link”-(ek)et készít

```
ln -s src.txt dst.txt
rm src.txt
```

```
ln src.txt dst.txt
rm src.txt
```

dst.txt
@src.txt

dst.txt
@653641

inode
@653641

- `chmod <kapcsolók> <jogok> <lista>`
 - Megváltoztatja a <lista>-ban található elemek hozzáférési jogait a <jogok> alapján.
 - A <jogok> leírását meg lehet adni ...
 - ... <kinek><hogyan><mit> alakban
 - `<kinek>` : a (All), u (User), g (Group), o (Other)
 - `<hogyan>` : + (megadás), - (megvonás), = (beállítás)
 - `<mit>` : r (Read), w (Write), x (eXecute)
- `pl.: chmod go-w out.txt`
 - ... <user><group><other> alakban három oktális számjeggyel
 - `4` : Read
 - `2` : Write
 - `1` : Execute
 - `pl. chmod 640 out.txt`
- Fontosabb kapcsolók
 - `-R` rekurzívan beállítja a jogokat

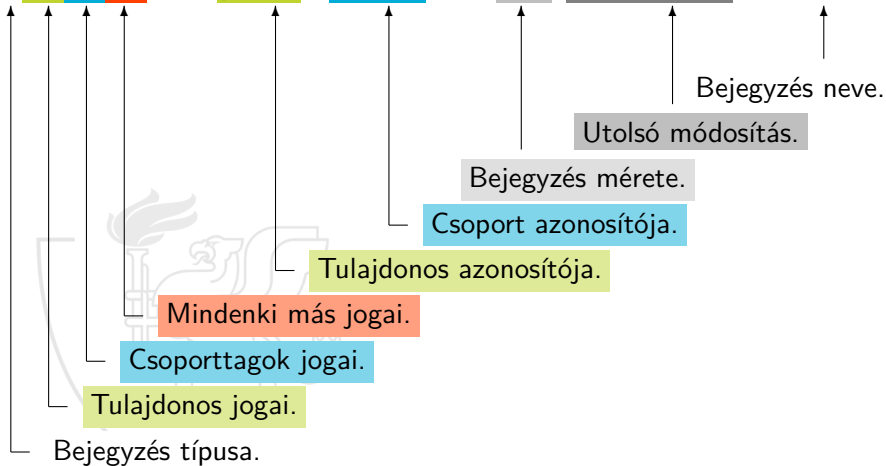
- A linux többfelhasználós rendszer. A fájlokhoz hozzáférést valamilyen módon korlátozni kell, ehhez rendelkezésre áll:
 - 3 osztály
 - Tulajdonos (User), Csoport (Group), Többiek (Others)
 - 3 féle jog
 - Olvasás (Read), Írás (Write), Végrehajtás (eXecute)
- Minden egyes fájlhoz el van tárolva némi plusz információ:
 - Tulajdonos
 - Minden egyes fájl vagy könyvtár pontosan egy felhasználóhoz tartozik.
 - Csoport
 - Minden egyes fájl vagy könyvtár pontosan egy csoportba tartozik.
 - Minden egyes felhasználó több csoportba tartozhat.
 - Utolsó módosítás dátuma
 - A fájlhoz tartozó adat helye (inode)
 - ...

Jogosultságok

```
$ ls -ld out.txt Gyakorlat
```

```
-rw-rw---- 1 gertom inf2000  
drwxr-xr-x 5 gertom inf2000
```

```
19 May 7 12:03 out.txt  
1024 Feb 7 2004 Gyakorlat
```



Linux parancsok

Szövegfájlok kezelése

`cat` – *fájlok teljes megjelenítése*

`more` – *fájlok megjelenítése oldalanként*

`less` – *fájlok megjelenítése oldalanként*

`head` – *fájlok első sorainak megjelenítése*

`tail` – *fájlok utolsó sorainak megjelenítése*

`grep` – *mintára illeszkedő sorok megjelenítése*

`wc` – *sorok, szavak, karakterek számának kiírása*



Linux parancsok

cat, more, less

- `cat <lista>`
 - A `<lista>` elemeinek teljes tartalmát egyszerre írja ki a képernyőre.
- `more <lista>`
 - A `<lista>` elemeinek teljes tartalmát oldalanként írja ki a képernyőre.
- `less <fájl>`
 - A `<fájl>` tartalmát írja ki a képernyőre, lapozható módon.



- `head <kapcsolók> <lista>`

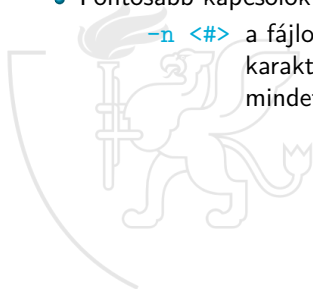
- A `<lista>` elemeinek első sorait írja ki a képernyőre.
- Fontosabb kapcsolók

- `-n <#>` a fájl első `<#>` sorát írja ki

- `tail <kapcsolók> <lista>`

- A `<lista>` elemeinek utolsó sorait írja ki a képernyőre.
- Fontosabb kapcsolók

- `-n <#>` a fájl utolsó `<#>` sorát írja ki, vagy ha `<#>` a „+” karakterrel kezdődik, akkor az `<#>`-ik sortól kezdve mindet



- `grep <kapcsolók> <minta> <lista>`
 - A <lista> elemeinek azon sorait írja ki a képernyőre, amelyekben megtalálható a <minta>.
 - Fontosabb kapcsolók
 - i *case insensitive* keresés, a kis- és nagybetű között nem tesz különbséget
 - n kiírja a sorok számát is
 - v inverz kimenet, a <minta>-t *nem* tartalmazó sorokat írja ki
- `wc <kapcsolók> <lista>`
 - Kiírja a <lista> elemeiben található bájtok/sorok/szavak számát.
 - Fontosabb kapcsolók
 - l a sorok számát írja ki
 - w a szavak számát írja ki
 - c a karakterek számát írja ki

Linux parancsok

Egyéb parancsok

`file` – fájl típus meghatározás

`du` – elfoglalt méret kiszámítása

`echo` – szöveg kiírása

`man` – manual-ok, leírások

`info` – manual-ok, leírások

`passwd` – jelszó váltás

`exit` – kilépés



- `file <fájl>`
 - A <fájl> típusát adja meg.
- `du <kapcsolók> <lista>`
 - Összeszámolja a <lista> elemei által a fájlrendszeren foglalt területet (fájlok méretét illetve könyvtárak tartalmát rekurzívan).
 - Fontosabb kapcsolók
 - s csak a végösszeget írja ki, a <lista> egyes elemei által foglalt méreteket nem
 - m kilobájtok helyett megabájtokban számol
 - h az ember számára könnyen olvasható méretkiírás
- `echo <kapcsolók> <szöveg>`
 - Kiírja a <szöveg>-et a képernyőre.
 - Fontosabb kapcsolók
 - n nem tesz sorvége jelet a kiíratás végére

Linux parancsok

echo, man, info, passwd

- `man <parancs>`
`info <parancs>`
 - Mindkettő a <parancs> részletes leírását mutatja meg, eltérő formában.
- `passwd`
 - A jelszó megváltoztatására szolgál.
 - **A kabinetben nem ezt kell használni, hanem a <http://www.inf.u-szeged.hu/jelszo> oldalról elérhető jelszóváltó űrlapot!**
- `exit`
 - Kilép a shell-ből, zárja a kapcsolatot. (Használható a <ctrl>+d billentyűkombináció is.)

1

Linux

Alapfogalmak

Linux parancsok
Linux shell

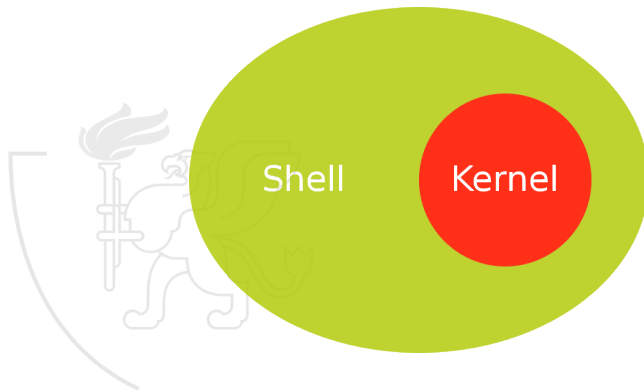
Felhasználók
Hálózat



Shell

Parancsértelmező

- A shell is „csak” egy program, de ez teszi lehetővé, hogy az operációs rendszer magjával, a kernellel kommunikálni tudjunk.
- Egy gyakori ábra:



- Sokféle van belőle
 - sh, bash, csh, dash, ksh, zsh
- Főbb funkciói
 - Kapcsolattartás a kernel és a felhasználó között.
 - Mintaillesztés.
 - Programok indítása.
 - Környezeti változók kezelése.
 - Be- és kimenet kezelés.
- Az eddig látott parancsokat is a shell értelmezi.
- A Shell egy úgynevezett prompt-tal jelzi, ha kész a parancsaink fogadására.
 - Ez általában tartalmazza a felhasználói azonosítót, a gép nevét, az aktuális könyvtárat, de egyéni ízlés szerint megváltoztatható.
 - A példáinkban a promptot a sor eleji \$ jel jelöli.

- A shell a megadott mintát mindig fájlokra és könyvtárakra próbálja illeszteni.
- Paraméterek megadásánál használhatunk joker (wildcard) karaktereket.

? Pontosan egy tetszőleges karaktert helyettesít.

* Akárhány tetszőleges karaktert helyettesít.

[...] A [és] jelek között felsorolt karaktereket helyettesítheti.

\c A c speciális karaktert jelenti, ahol c lehet:

- \, szóköz, ', ", ?, *, [,], ' ,



Mintaillesztés

példák [1/2]

- `alfa.?`
- `start??.xml`
- `*.html`
- `*vector*`
- `[Kk]epek`
- `gcc-[4-6]`
- `Egy_nev*`
- `?s_ [Ee]zek_komb*t?ak?[a-z]okf*k ?ppen`
- `\?x*`

- A Shell a minta helyére beilleszti az összes, a mintára illeszkedő fájl vagy könyvtár nevét.

```
$ ls -l *.occ
-rw-r----- 1 gertom inf2000 12366 jún 20 13:19 504.occ
-rw-r----- 1 gertom inf2000 764  ápr 23 08:03 plex.occ
```

- A szókezdő . nem illeszkedik!

```
$ ls -l ?ba*
ls: ?ba*: Nincs ilyen fájl vagy könyvtár
$ ls -l .ba*
-rw----- 1 gertom inf2000 7075 sze 1 22:23 .bash_history
-rw-r--r-- 1 gertom inf2000 24 sze 25 2001 .bash_logout
-rw-r--r-- 1 gertom inf2000 256 okt 5 2002 .bash_profile
-rw-r--r-- 1 gertom inf2000 135 dec 10 2003 .bashrc
```



Programok indítása, paraméterezése

- Programokat a program (esetleg útvonallal ellátott) nevének beírásával indíthatunk.
- Egy program pontos helyének megkeresése a
 - `type <programnév>`
paranccsal lehetséges.
- A programnak paramétereket adhatunk, melyeket *whitespace* karakterek (szóköz, tabulátor) választanak el egymástól.
 - Az ilyen elválasztó karakterek száma lényegtelen.

```
$ echo Az eredmény ugyanaz
Az eredmény ugyanaz
$ echo   Az       eredmény   ugyanaz
Az eredmény ugyanaz
```

- Mit tehetünk, ha pl. a szóköz is része a paraméternek?

```
$ echo három\ \ \ szokoz
három  szokoz
$ echo "három  szokoz"
három  szokoz
$ echo 'három  szokoz '
három  szokoz
$ echo 'három\ \ \ szokoz '
három\ \ \ szokoz
```

- " (idézőjel) Néhány behelyettesítés működik.

```
$ echo "x\"ab\"y\"  
x"aby"  
$ echo "$HOME"  
/home/gertom
```

- ' (apoztróf) Semmit sem helyettesít, karakterről karakterre másol.

```
$ echo 'x\'ab\'y\'  
x\aby\  
$ echo '$HOME'  
$HOME
```



Linux parancsok

Környezeti változók

set – *beállított változók listázása*

export – *változó érvényességének kiterjesztése*

unset – *változó törlése*

= – *értékadás*



Környezeti változók

- A változók különféle értékeket vehetnek fel, és amikor a shell egy változót talál a parancssorban, az adott változó aktuális értékét behelyettesíti oda.
- Általános alakjuk
 - \$NEV
- Értékadás
 - NEV=<ertek>
- Előre definiált változók
 - \$HOME** A felhasználó saját könyvtára (használható a ~ is).
 - \$PATH** Keresési útvonal, a shell az itt felsorolt könyvtárakban keresi a futtatandó programot.
 - \$PS1** A prompt kinézetét írja le.

Linux parancsok

set, export, unset

- set
 - Kiírja a beállított változókat és értékeiket (meg pár egyéb dolgot).
- export <változó>
 - A <változó> elérhető lesz a shellből elindított más programok számára is.
- unset <változó>
 - Törli a <változó>-t.

```
$ echo $NEV
$ NEV=ertek
$ export NEV
$ echo $NEV
ertek
$ NEV=mas
$ echo $NEV
mas
$ unset NEV
$ echo $NEV
$
```

- Ha egy szövegfájlokkal dolgozó linux programot úgy indítunk el, hogy nem adunk meg neki fájlnevet, akkor az általában automatikusan a *standard input*-ját (bemenet) fogja használni, azaz alapesetben nekünk kell ott helyben begépelnünk a feldolgozandó szöveget.
 - Ez a gépelés a `<ctrl>+d` billentyűkombináció lenyomásáig tart (amit `^d`-vel szokás jelölni).
 - A `^d`-vel vigyázzunk, mert a shell is egy ilyen program!
- Az ilyen programok általában a *standard output*-ra (kimenet) írnak, azaz alapesetben a képernyőn jelenik meg az eredmény.
- Minden programnak van még egy kimenete, a *standard error* (hiba), ahová a hibaüzeneteket küldi. Alapesetben ez is a képernyőn jelenik meg (de független a standard outputtól).

Programok be- és kimenete

Példák

- A következő példákban az alábbi színjelölést használjuk:
 - **standard input** a billentyűzetről
 - **standard output** a terminálra
 - **standard error** a terminálra

```
$ cat
Ha a cat programot paraméter nélkül indítjuk el,
Ha a cat programot paraméter nélkül indítjuk el,
akkor a standard inputot használja, ami jelenleg
akkor a standard inputot használja, ami jelenleg
a billentyűzet.
a billentyűzet.
^d
$
```



Programok kimenete

Standard output átirányítása felülírással

- A program kimenetét a > jellel lehet egy fájlba irányítani.

```
$ cat >kimenet.txt
```

Ha a cat programot paraméter nélkül indítjuk el, akkor a standard inputot használja, ami jelenleg a billentyűzet.

A > hatására a kimenet nem soronként azonnal jelenik meg, hanem a kimenet.txt-be íródik bele.

```
^d
```

```
$ more kimenet.txt
```

Ha a cat programot paraméter nélkül indítjuk el, akkor a standard inputot használja, ami jelenleg a billentyűzet.

A > hatására a kimenet nem soronként azonnal jelenik meg, hanem a kimenet.txt-be íródik bele.

```
$
```

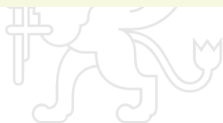


Programok kimenete

Standard output átirányítása hozzáfűzéssel

- A program kimenetét a >> jellel lehet egy fájl eddigi tartalma mögé fűzni.

```
$ cat >kimenet.txt
Az előző példában már feltöltöttük a kimenet.txt-t.
Most elveszik az előbb beleírt szöveg, mert >-t használtunk.
~d
$ cat >>kimenet.txt
Most viszont az előző is megmarad, mert a >> hozzáfűzi
az új szöveget, és nem felülírja az eredetit.
~d
$ more kimenet.txt
Az előző példában már feltöltöttük a kimenet.txt-t.
Most elveszik az előbb beleírt szöveg, mert >-t használtunk.
Most viszont az előző is megmarad, mert a >> hozzáfűzi
az új szöveget, és nem felülírja az eredetit.
$
```



Programok bemenete

Standard input átirányítása

- A program bemenetére a < jellel lehet egy fájl tartalmát ráirányítani.

```
$ echo 'Még egy sor, hogy több legyen.' >>kimenet.txt
$ cat <kimenet.txt
Az előző példában már feltöltöttük a kimenet.txt-t.
Most elveszik az előbb beírt szöveg, mert >-t használtunk.
Most viszont az előző is megmarad, mert a >> hozzáfűzi
az új szöveget, és nem felülírja az eredetit.
Még egy sor, hogy több legyen.
$ head -n 4 <kimenet.txt >k2.txt
$ tail -n 2 <k2.txt >k3.txt
$ cat k3.txt
Most viszont az előző is megmarad, mert a >> hozzáfűzi
az új szöveget, és nem felülírja az eredetit.
$
```



Programok be- és kimenete

Csővezeték (pipe)

- A program kimenetét a | jellel lehet a következő program bemenetére irányítani.

```
$ cat <kimenet.txt
```

Az előző példában már feltöltöttük a kimenet.txt-t.

Most elveszik az előbb beleírt szöveg, mert >-t használtunk.

Most viszont az előző is megmarad, mert a >> hozzáfűzi az új szöveget, és nem felülírja az eredetit.

Még egy sor, hogy több legyen.

```
$ head -n 4 kimenet.txt | tail -n 2
```

Most viszont az előző is megmarad, mert a >> hozzáfűzi az új szöveget, és nem felülírja az eredetit.

```
$ cat <kimenet.txt | grep o | head -n 2 >ki.txt
```

```
$ cat ki.txt
```

Most elveszik az előbb beleírt szöveg, mert >-t használtunk.

Most viszont az előző is megmarad, mert a >> hozzáfűzi

```
$
```



- A linux egyszerre több programot is képes futtatni.
- A programok alaphelyzetben induláskor előtérben kezdenek futni, azaz a standard inputon keresztül fogják a billentyűzetet, így amíg nem végeznek, addig nem kapjuk vissza a prompt-ot.
- Az ilyen programokkal két dolgot lehet csinálni:
 - `<ctrl>+c` Azonnal *megszakítja* a program futását, a végrehajtás befejeződik. (^c)
 - `<ctrl>+z` *Leállítja* a program futását, de az később folytatható. (^z)
- Egy programot el lehet egyből a háttérben indítani, ha a parancssor végére egy & jelet teszünk.
 - Háttérben csak olyan programok futhatnak, amelyeknek a standard inputon nincs szükségük adatra, vagy ezt egy fájlból átirányítással kiküszöböltük.
 - Általában grafikus programokat, tömörítéseket vagy nagyobb fájlműveleteket szokás így indítani.

- Egy *job* nem más, mint a valami miatt egymástól függő programok összessége.
 - Az előtérben egyetlen aktív job futhat.
 - A háttérben tetszőleges számú job futhat vagy várakozhat.
 - A háttérben lévő job-okra a sorszámukkal hivatkozhatunk.
- Egy program egy adott futó vagy várakozó példányát *process*-nek nevezzük.
 - Minden process rendelkezik egy *pid* számmal, ez azonosítja a process-t.
 - Egy job-hoz több process is tartozhat.



Linux parancsok

Job-ok és process-ek

`jobs` – job-ok listázása

`ps` – process-ek listázása

`fg` – előtérben futtatás

`bg` – háttérben futtatás

`kill` – job-ok, process-ek manipulálása



Linux parancsok

jobs, ps, fg, bg, kill

- jobs
 - Kiestázza a felhasználóhoz (munkafolyamathoz) tartozó job-okat.
- ps
 - Kiestázza a rendszer (felhasználók) process-eit.
- fg <#>
 - Az előtérben újraindítja a #-ik job-ot.
- bg <#>
 - A háttérben újraindítja a #-ik job-ot.
- kill <kapcsolók> <#>
 - Megszakítja a <#> azonosítójú process-t, vagy ha a <#> a „%” karakterrel kezdődik, akkor az adott sorszámú job-ot. Ha a sorszám -1, akkor *minden* saját process-t megszakít.
 - Fontosabb kapcsolók
 - s KILL Megszakítás (nem blokkolható).
 - s STOP Leállítás.
 - s CONT Újraindítás.

- Indítsunk egy munkát az előtérben.
 - A `yes` folyamatosan `y` karaktereket ír a standard outputra.
 - A `tr y n` a bemenetben található `y` karaktereket `n`-re cseréli.
 - A `/dev/null` egy olyan fájl, ami mindent „lenyel”.

```
$ yes | tr y n >/dev/null # Elindítjuk a műveletet az előtérben
^z # Leállítjuk a futását
[1]+ Stopped yes | tr y n >/dev/null
$ fg %1 # Újraindítjuk szintén az előtérben
yes | tr y n >/dev/null
^c # Megszakítjuk a futását.
$ yes | tr y n >/dev/null & # Elindítjuk a műveletet a háttérben
[1] 11503
$ jobs # Megnézzük milyen job-jaink futnak
[1]+ Running yes | tr y n >/dev/null &
$ ps # Megnézzük milyen process-eink vannak
  PID TTY          TIME CMD
 11288 pts/5    00:00:00 bash
  11502 pts/5    00:00:26 yes
  11503 pts/5    00:00:04 tr
  11504 pts/5    00:00:00 ps
$ kill %1 # Megszakítjuk az 1-es job-ot
$
[1]+ Félbeszakítva yes | tr y n >/dev/null
$ jobs # Megnézzük milyen job-jaink futnak
$
```

Több program indítása sorban

- Ha több programot sorban egymás után szeretnénk végrehajtatni, akkor soroljuk fel őket egyetlen parancssorban.
- Az egyes parancsok elválasztására többféle jelet használhatunk:
 - ; Egmás eredményeitől függetlenül lesznek sorban végrehajtva.
 - && A másodikat csak akkor hajtja végre, ha az első sikerült (és).
 - || A másodikat csak akkor hajtja végre, ha az első **nem** sikerült (vagy).

```
$ cp out.txt Gyakorlat && rm out.txt
$ rm out.txt ; cp afonya.tex Gyakorlat
$ mv out.txt Gyakorlat || mv in.txt Gyakorlat
```

1

Linux

Alapfogalmak

Linux parancsok
Linux shell

Felhasználók
Hálózat



- Mint arról már volt szó, a linux egy többfelhasználós rendszer, vagyis ugyanazt a rendszert több felhasználó is jogosult használni, akár egyidőben is.
 - A felhasználóknak a fájlrendszerre vonatkozó jogosultságaik vannak (ezeket már láttuk).
 - A felhasználók tudhatnak egymásról.



Linux parancsok

Felhasználók

`finger` – általános felhasználói információk

`who` – bejelentkezett felhasználók

`w` – bejelentkezett felhasználók és munkáik

`last` – utolsó bejelentkezések



Linux parancsok

finger, who, w, last

- `finger <paraméter>`
 - Paraméter nélkül megadja a bejelentkezett felhasználók listáját.
 - Ha a `<paraméter>` ...
 - egy azonosító, akkor kiírja az azonosítóhoz tartozó felhasználó adatait.
 - egy név, akkor kiírja a hasonló nevű felhasználók adatait.
 - `@host` alakú akkor kiírja, hogy a `host` nevű gépen kik vannak bejelentkezve (ha nincs letiltva).
 - `id@host` alakú akkor kiírja a `host` nevű gépen lévő `id` azonosítójú felhasználó adatait (ha nincs letiltva).
- `who`
 - Megadja, hogy a gépre éppen mely felhasználók vannak bejelentkezve.
- `w`
 - Megadja, hogy a gépre éppen mely felhasználók vannak bejelentkezve, és éppen mit futtatnak az előtérben.
- `last`
 - Megadja, hogy a gépre az elmúlt időszakban kik és mikor jelentkeztek be.

1

Linux

Alapfogalmak

Linux parancsok
Linux shell

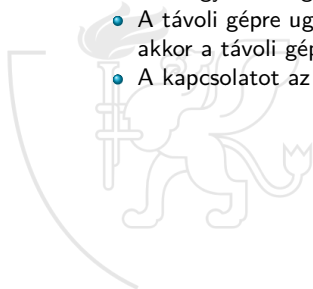
Felhasználók
Hálózat



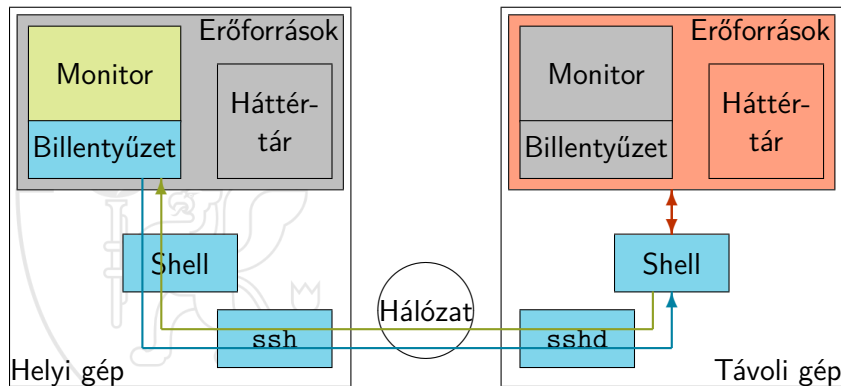
- A világ számítógépeinek nagy része az interneten keresztül TCP/IP protokollal segítségével kapcsolódik egymáshoz, amelyben egy gép azonosítása IP-cím alapján történik.
 - IPv4
 - 32 bites cím, ami 4 darab egybájtos számból áll, melyeket ponttal elválasztva decimális alakban írunk le.
 - Az SZTE gépeinek például 160.114.*.* alakú a címük.
 - Ez maximum kb. 4 milliárd cím, ami már „elfogyott”.
 - IPv6
 - 128 bites cím, ami 8 darab kétbájtos számból áll, melyeket kettősponttal elválasztva hexadecimális alakban írunk le.
 - Például: fe80:0:0:0:e2bc:4eff:fe18:9f3c.
 - Ez egy darabig talán elég lesz (a Balaton minden vízmolekulájára jutna öt ilyen cím, a Föld minden mm³-re kb. 300 millió).
- Ha valakinek 2 ilyen gépre is van azonosítója, akkor az egyikről a hálózaton keresztül elérheti a másik gépet, be tud jelentkezni rá és dolgozni tud rajta.

- Az IP-címek mellett létezik a *domain név* mint azonosító.
- Ez az emberek által könnyebben megjegyezhető formájú `www.inf.u-szeged.hu` alakú hierarchikus név.
 - Legalábbis a kétszintű hierarchia még létezik.
- A két azonosító közötti megfeleltetést a Domain Name Server-ek biztosítják, és a `host` programmal tudjuk ellenőrizni, hogy mely névhez milyen cím tartozik.
 - (Az időnként előforduló „DNS error” nem egy organikus számítógépre utal, hanem arra, hogy a beállított Domain Name Server-en keresztül nem sikerült megtudni a névhez tartozó IP-címet.)

- Ha be vagyunk jelentkezve egy gépre, az ssh program segítségével tudunk onnan egy másikra bejelentkezni.
- Az ssh használata:
 - `ssh <azonosító>@<gépnév>`
 - Az <azonosító> felhasználót bejelentkezteti a <gépnév> gépre.
 - Az <azonosító>@ rész csak akkor kell, ha az a helyi és távoli gépen nem egyezik meg.
 - A távoli gépre ugyanúgy jelszóval kell bejelentkeznünk. Ha ez sikerült, akkor a távoli gépen elindul egy shell.
 - A kapcsolatot az `exit` paranccsal (vagy `^d`-vel) zárhatjuk.

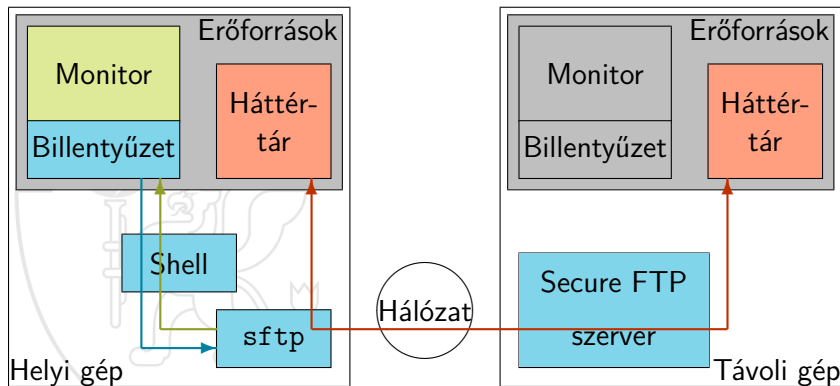


- Sikeres bejelentkezés után a távoli gépen elindul egy shell, ami a helyi géptől kapja a bemenetét, és a helyi gépre küldi a kimenetét.



- Ha be vagyunk jelentkezve egy gépre, az sftp program segítségével tudunk onnan fájlmozgatás céljából egy másikra bejelentkezni.
- Az sftp használata:
 - `sftp <azonosító>@<gépnév>`
 - Az <azonosító> felhasználót bejelentkezteti a <gépnév> gépre fájlmozgatás céljából.
 - Az <azonosító>@ rész csak akkor kell, ha az a helyi és távoli gépen nem egyezik meg.
 - A távoli gépre ugyanúgy jelszóval kell bejelentkeznünk. Ha ez sikerült, akkor létrejön a kapcsolat a távoli géppel, amin keresztül a megfelelő parancsokkal fájlokat mozgathatunk a két gép között.
 - A kapcsolatot az `exit` vagy `bye` paranccsal zárhatjuk.

- Sikeres bejelentkezés után létrejön a kapcsolat a távoli géppel, amin keresztül fájlokat mozgathatunk a két gép között.



SSH és SFTP

Különbségek

ssh -val bejelentkezve a távoli gépre csak azok az erőforrások érhetőek el, amik a távoli gépen elérhetőek, így a két gép között nem tudunk fájlokat mozgatni.

sftp -vel bejelentkezve a távoli gépre annak csak a háttértárolóját tudjuk elérni, más erőforrásait nem, cserébe a helyi gép háttértárolója is a rendelkezésünkre áll, így tudunk a két gép között fájlokat másolni.



- A bejelentkezés hasonlít az `ssh`-hoz, de a távoli gépen nem shell indul.
- Kiadható parancsok:

`lpwd` – Távoli és helyi aktuális könyvtár lekérdezése.

`lls` – Távoli és helyi aktuális könyvtár tartalmának listázása.

`lcd` – Könyvtárváltás távoli és helyi gépen.

`lmkdir` – Könyvtár létrehozás távoli és helyi fájlrendszeren.

`rm` – Távoli fájl törlése.

`rmdir` – Távoli könyvtár törlése.

`get` – Másolás a távoli gépről a helyi gépre.

`put` – Másolás helyi gépről a távoli gépre.

`ascii` – Váltás szöveges fájlok átvitelére.

`binary` – Váltás bináris fájlok átvitelére.

`bye, exit` – Kilépés.

- `scp <kapcsolók> <források> <cél>`
 - Úgy működik, mint a `cp` parancs, csak a forrásként és/vagy célként egy távoli gép könyvtára is megadható. Az útvonalakat ki lehet egészíteni a `<gépnév>`: vagy `<azonosító>@<gépnév>`: előtaggal.
 - Ez relatív útvonal esetén a távoli gépen lévő home könyvtárat jelenti.

```
$ scp Gyakorlat/in.txt gertom@linux.inf.u-szeged.hu:  
$ scp gertom@linux.inf.u-szeged.hu:in.txt vissza.txt
```

