Haladó bash

Jegyzetben a 8-9.-10. fejezet (97.-től 125. oldalig). http://stegerjozsef.web.elte.hu/teaching/szamalap.pdf (http://stegerjozsef.web.elte.hu/teaching/szamalap.pdf)

Egyéb források:

Linux parancsok: http://www.letix.hu/ (http://www.letix.hu/)

Szövegszerkesztés linuxban

Terminálban használható szövegszekesztőből több is létezik, melyek közül az egyik leghatékonyabb és legelterjettebb a `Vim`, de mivel használata hosszú gyakorlást igényel, így mi nem ezt használjuk az órán.

Egy egyszerű szövegszerkesztő az **`mcedit`**. Indítása egyszerűen az `'mcedit'` paranccsal történik.

- Fájlok mentése: **F2**
- Kilépés: **F10**
- Fájl megnyitás: Terminálban kiadott **`mcedit FÁJLNÉV`** paranccsal.

Ablakos szövegszerkesztők

A linuxban többféle ablakos szövegszerkesztő van alapértelmezésben telepítve. Az egyik legelterjettebb a **`gedit`** nevű program.

Mindenki indítsa el a gedit nevű programot:

- Menü -> Accessories -> gedit
- Kiadjuk a terminálban: **`gedit`**

Hasonlóan működik, mint a Windowsban a jegyzettömb.

Párhuzamos programkezelés

A linuxban lehetőség van a programok párhuzamos kezelésére, akár 1 terminál ablakon belül is. Az épp futó alkalmazást háttérbe lehet küldeni, és az így visszakapott promt után más parancsok is kiadhatók.

Példa:

Mindenki indítsa a gnuplotot: **`gnuplot`**

Ezután adjuk ki a **`CRTL+Z`** billenytű kombinációval felfügesztés parancsát, majd **`bg`** beütésével háttérbe küldhető a program.

```
gnuplot
'CTRL+z'
bg
```

Futó programok kezelése

A **`jobs`** paranccsal ellenőrizhető, milyen futó programok vannak.

Minden program melett van egy szám (ID), amivel hivatkozható a program:

Program kilövése: **`kill %ID`** paranccsal

```
kill %2
```

Program előrehozása: **`fg %ID`** paranccsal

```
fg %1
```

Program háttérben történe indítása

Egy programot terminálból úgy is el lehet indítani, hogy a promt megmaradjon, azaz a program háttérben indul el (olyan, mintha **CTRL+z** és **gb** utasítást adtunk volna ki. Ezt a program neve után írt **&** karakterrel lehet elérni.

Példa:

```
gedit cica.txt &
```

Különleges karakterek bash-ban 1 ("Joker karakterek")

- **`?`** : 1 darab "akarámit" helyettesít

  ```
  "sinusa?atok.dat"
  # ez lehet "sinusadatok.dat"
  ```

- **`*`** : Akármilyen hosszú szövegrészt helyettesít

  ```
  "sin*.dat"
  # Minden fájl aminek ".dat" a vége, és "sin" az eleje
  ```

Különleges karakterek bash-ban 2

Több parancs kiadása

A **`"&&"`** vagy **`";"`** jellel lehet több parancsot kiadni egy sorban. **`;`** utáni parancs mindenképp lefut, a **`&&`** pedig csak ha minden rendben volt.

```
ls -al && cp valami.tex ize.tex && cd /latex
```

Különleges karakterek bash-ban 3

Parancs kimentének feldolgozása

A **|** jellel át lehet adni egy parancs kimentét egy másik parancsnak:

Példa:

```
ls | sort -r
```

Megj.: A **`sort`** parancs azt csinálja, amit a neve mutat, azaz sorba rendez. A `sort -r` visszafelé rendez (reverse).

Különleges karakterek bash-ban 4

Változók bashban

- **`$`** : Segítségével változó neve és értéke között tudunk különbséget tenni. Ha `$valtozo`-t írunk, az a változónk értékét fogja visszaadni.
  - Példa: `cica=10;echo $cica` ---> Eredménye: **10**
- **`\\`** : Bourne Again Shell escape-karaktere. Többek között ezzel a jellel lehet elérni, hogy az azt követő speciális karakter elveszítse módosító hatását.
  - Példa: `cica=1;echo \$cica` ---> Eredménye: **$cica**

Különleges karakterek bash-ban 5

"Stringkezelés"

- **'** : Megakadályozza a héjat (shell), hogy a $-t a változók jelzésének tekintse
  - Példa: `echo '$cica'` ---> Eredménye: **$cica**
- **"** : Az idézőjelek arra valók, hogy egyes karakterek vagy lefoglalt szavak speciális jellegét feloldják.
  - Példa: `echo "Ez egy szöveg"` ---> Eredménye: **Ez egy szöveg**
- **`...`** vagy **${...}** : Operátor, mely a két jel között lévő utasítást végrehajtja, és a kimenetét adja vissza úgy, hogy az új sor karakter helyére szóközt rak. Ez bash scripteknél lesz hasznos.

Fájlok keresése:

Fájlokat a **`find`** paranccsal lehet keresni.

```
find ~ -name "*.jpg"
# az összes jpg kiterjesztésű kép keresése a home-ban
find ~ -iname "*.DAT"
# kis és nagybetű különbség nincs
find / -perm 777
#777-es joggal rendelkező állomány keresése a gyökérben
find ~/latex -size +1024M
#1024 Mb-nál nagyobb fájlok keresése a /latex mappában
```

Szöveg keresése bash-ban

Szöveget keresni a **grep** parancs segítségével lehet. **Alapértelmezésben nem azonos a kis/NAGY betű.**

```
grep equation ./latex/latex.tex
# "equation" szó keresése a latex.tex-ben
grep equation *
# "equation" szó keresése minden fájlban a mappán belül
grep -i EquAtion * # Kis/NAGY betű nem számít
grep -n equation * # kiírja a sor számát
grep -l equation * # Fájlnevet ad vissza
```

Kimenet átirányítása

Lehetőség van a kimenetet a terminál ablak heylett pl. fájlba irányítani:

```
ls -l > file_lista.txt
Létrejött a file_lista.txt
(ha létezett már a fájl, akkor ezzel felülírodott)
```

Megjegyzés: Fájlból bemenetet csinálni a **"<"** jel tud.

Hozzáírás fájlhoz:

Hozzáírni a **>>** paranccsal lehet egy meglevő fájlhoz:

```
echo "Kiscica,ezt hozzáírom a fájlhoz" >> file_lista.txt

tail file_lista.txt
```

Egyéb Hasznos parancsok 1:

- **passwd**: Jelszó csere. Ekkor kéri a régi jelszót, majd 2-szer az újat a rendszer! Ezeken a gépeken nem használható!
- **file akarmi.tex** : megvizsgálja az akarmi.tex típusát
- **uniq** : Több sorból álló szövegben az ismétlődő sorokkal kezd valamit
- **tree** : kiírja a könyvtárstruktúrát
- **du** : Az aktuális könyvtár fájljainak méretét adja meg
- **df** : Szabad terület számítása, partíciónként

Egyéb Hasznos parancsok 2:

- **`touch file`** : létrehoz egy "file nevű" üres állományt
- **`cut`** : Bemenetként (stdin), vagy paraméterként megadott fájl minden sorának egy megadott részét vágja ki
- **`tr`** : karakterek lecserélése, változtatása adott karaktersorban
  - **`echo vegyes | tr a-z A-Z`** : a vegyes szóban a kis betűket nagyra cseréli.
  - **`echo egyesek | tr -d e`** : az egyesek szóból kitörli az e betűket

Egyéb Hasznos parancsok 3:

- **`diff -u file1 file2 > eredmeny`** : Összehasonlítja a fájlok tartalmát, a különbséget pedig az "eredmeny"-be írja
- **`tar`** : Ki-be tömörítő
  - **`tar -cvfz file.tar.gz /eleresi/ut/amit/mentunk`** : az elérési útvonal alatt levő adat gzip-be tömörítése
  - **`tar -xvfz file.tar.gz`**: Kitömöríti a gz-t, majd a tar-t, és egy /file könyvtárba teszi

Egyéb Hasznos parancsok 4:

- **`login`** : Bejelentkezés
- **`logout`** : Kijelentkezés
- **`who`** : Bejelentkezett felhasználók kiíratása
- **`ps`** : Futó folyamatok kiírása
- **`killall command`** : az összes fehasználó (amihez van jogunk) által futtatott "command" nevű folyamat leállítása
- **`shutdown -P now`** : Kikapcsolás (-P=Poweroff), azonnal (percben lehet megadni az időt)

Haladó szint

Nem kötelező tananyag

Irodalom (angol és magyar nyelven):

Bash programozás:

- http://okt.kmf.uz.ua/dw/lib/exe/fetch.php?media=tt-admin_oi:raffaigaboristvan_bash_programozas.pdf (http://okt.kmf.uz.ua/dw/lib/exe/fetch.php?media=tt-admin_oi:raffaigaboristvan_bash_programozas.pdf)
- http://www.tldp.org/pub/Linux/docs/HOWTO/translations/hu/Bash-Prog-Intro-HOWTO-hu.txt (http://www.tldp.org/pub/Linux/docs/HOWTO/translations/hu/Bash-Prog-Intro-HOWTO-hu.txt)
- http://math.iit.edu/~mccomic/595/unixhelp/ (http://math.iit.edu/~mccomic/595/unixhelp/)
- https://www.youtube.com/watch?v=xtS2NiABf54&list=PLtK75qxsQaMIlFCcFZpTBLnaCJ0I0uiaY (https://www.youtube.com/watch?v=xtS2NiABf54&list=PLtK75qxsQaMIlFCcFZpTBLnaCJ0I0uiaY)
- http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf (http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf)
- http://www.tldp.org/LDP/abs/abs-guide.pdf (http://www.tldp.org/LDP/abs/abs-guide.pdf)

Bash programozás

A parancsértelmező által ismert parancsok fűzhetőek össze egy "programba" vagy más néven shell script-be.

- A program első sorában célszerű minden esetben megadni a futtató programot (shell-t), jelen esetben a bash-t. Tehát a fájl kezdete legyen (bash esetén): **`#!/bin/bash`**
- Más futtató program (parancsértelmező) is megadható a program első sorában (például python).
- Futtatási jog nélkül nem mindig futtatható, ezért célszerű kiadni: **`chmod +x programnev`**

- Program futtatásához a parancs(ok)

```
bash programnev.sh

(./programnev.sh)
```

Más interpreterekhez:

```
python akarmi.py #python script

octave valami.m #octave script
```

Példák:

Hozzunk létre egy **hello.sh** fájlt, és írjuk bele:

```
gedit hello.sh &
```

```
#!/bin/bash
echo Hello World!
```

Mentés után futtassuk le:

```
(chmod u+x hello.sh)
bash ./hello.sh
```

Hasznos dolgok a szrikptekhez:

- Változók: Lehetséges változókat eltárolni és később hivatkozni rájuk: `$izé` az izé nevű változó.

```
szam=20
echo $szam
```

- Egész számok generálása a **`seq`** paranccsal lehetséges (pl. 10-től 100-ig)

```
seq ELSŐ NÖVEKMÉNY UTOLSÓ
seq 10 2 100 # 10-től kettesével 100-ig megy
seq 100 -5 10 # 100-tól 10-ig 5-ösével
```

Bash mint számológép

Próbáljuk ki:

```
a=2
b=5
echo $a*$b
```

- A **`let`** parancs: Ez a parancs képes aritmetikai műveleteket végezni: ("b++" jelentése, a b-t 1-gyel megnöveli)

```
a=1234; let "a=a+1"; echo "$a"
a=2; b=5; let c=$a*$b; echo $c
let 'b = a' "(a += 3) + $((a = 1)), b++"
```

Egyéb példák:

A következő 3 példához használjuk a **test.sh** -t állományt:

```
gedit test.sh &

chmod u+x test.sh
```

A következő 3 példát másoljuk bele (külön-külön), mentsük el a fájlt, majd teszteljük le szkripteket a következő parancs kiadásával:

```
bash test.sh
```

- **While** szerkezet: Ez egy olyan ciklus, ami először megvizsgál egy feltételt, majd végrehajta, ha igaz

```
#!/bin/bash
    COUNTER=0
    while [ $COUNTER -lt 10 ]; do
        echo a szamlalo erteke: $COUNTER
        let COUNTER=COUNTER+1
    done
```

COUNTER változót 1-gyel növeli addig, amíg el nem éri a 10-et.

- **Until** szerkezet:

```
#!/bin/bash
    COUNTER=20
    until [ $COUNTER -lt 10 ]; do
        echo a szamlalo erteke: $COUNTER
        let COUNTER-=1 # COUNTER-=1 jelentése: COUNTER=COUNTER-1
    done
```

COUNTER változót 20-as kezdőértékről 1-gyel csökkenti, amíg el nem éri a 10-et.

- **For** ciklus (ez egy olyan ciklus, ami addig fut, amíg igaz az állítás) Ezt használjuk általában mindig.

Szintatktika: **for "Változó megadás" "Állítás" "Utasítás"**

```
#!/bin/bash
    for i in `seq 1 10`;do
    echo $i
    done
```

Az **i** változó felveszi 1-től 10-ig az értékeket, majd ki is írja a képernyőre

A programok írhatóak 1 folytonos sorba is, ahol az eddigi új sort ";" vagy "&&" jel váltja fel:

- while szerkezet:

```
SZAM=0; while [ $SZAM -lt 10 ]; do echo érték: $SZAM ;let SZAM=SZAM+1;don
e
```

- until szekezet:

```
COUNT=20; until [ $COUNT -lt 10 ]; do echo érték:$COUNT; let COUNT-=1;don
e
```

Példák for ciklusra

- for ciklus:

```
for i in `seq 1 10`;do echo $i; done
for i in `seq 2 5 100`;do echo $i; done
for i in $(ls);do echo fájlok: $i; done
```

Gawk

Jegyzetben az 4. fejezet (53-től 67. oldalig). http://stegerjozsef.web.elte.hu/teaching/szamalap.pdf (http://stegerjozsef.web.elte.hu/teaching/szamalap.pdf)

Az **awk** (vagy **gawk**, azaz GnuAWK) egy olyan programozási nyelv, amit szöveges állományok feldolgozására terveztek. Egy tipikus awk programot (szkript) egy interpreter olvas be és hajt végre. A szkript végrehajtása során a feldolgozott szöveges állomány(ok)at (tartalmának változatlanul hagyásával) másféle kimenetté formálja át. A bemeneti adatok változatlanul hagyása biztosítja, hogy ugyanazon adatokra (például szövegfájlok) többféle awk program is futtatható egymás után, vagyis ugyanazon fájl(ok)ból többféle adat is kinyerhető legyen.

Irodalom:

GAWK:

- http://www.ms.sapientia.ro/~lszabo/unix_linux_hejprogramozas/eloadas2010/10_awk.pdf (http://www.ms.sapientia.ro/~lszabo/unix_linux_hejprogramozas/eloadas2010/10_awk.pdf)
- http://www.grymoire.com/Unix/Awk.html (http://www.grymoire.com/Unix/Awk.html)
- https://blyx.com/public/docs/programacion/Awk_Language_Programming.pdf (https://blyx.com/public/docs/programacion/Awk_Language_Programming.pdf)
- https://www.gnu.org/software/gawk/manual/gawk.pdf (https://www.gnu.org/software/gawk/manual/gawk.pdf)

Töltsük le a két fájlt:

wget itl7.elte.hu/~iracz/Oktatas/SzA/millenium.dat

wget itl7.elte.hu/~iracz/Oktatas/SzA/millenium_new.dat

Nézzük is meg őket:

```
more millenium.dat
more millenium_new.dat
```

Mi az eltérés köztük?

Írassuk ki a két fájlt gawk-val!

```
awk millenium.dat

awk millenium_new.dat
```

A "`print`" parancs

A **`print`** paranccsal lehet kiíratni az awk-kal bármit.

Példa: (Csak az első oszlop kiíratása)

```
awk '{print $1}' millenium.dat

awk '{print $1}' millenium_new.dat
```

Most mi a hiba?

Elő- és utófeldolgozók

A **`BEGIN`** előfeldolgozó a lényegi programot megelőzően fut le. Itt lehet megadni például a bemeneti fájl szerkezetét (pl. mi az oszlopelválasztó karakter). Az **`END`** utófeldolgozó a lényegi program után fut le. Itt adható meg, hogy pl. mentse el egy fájlban adott stílusban az eredményt a szkript.

Különleges vezérlő karakterek

- RS: Record separator (Sorok elválasztója)
- FS: Field separator (Oszlopok elválasztója)
- NR: Number of record (Sorok száma)
- NF: Number of field (Oszlopok száma)

Speciális helyettesítő karakterek:

- `[a-z]` a,b,c,...z
- `[0-9]` 0,1,2,3,...9
- `[:alnum:]` betű vagy szám
- `[:alpha:]` betű
- `[:digit:]` szám

Példa:

```
awk 'BEGIN{FS=";"}{print $1}' millenium_new.dat

awk 'BEGIN{FS=";"}{print $1,$2*$3,$8}' millenium_new.dat
```

Példa programok összekapcsolására:

```
awk '{print $8,$3,$4,$5}' millenium.dat | sort > adat.dat

more adat.dat

gnuplot

splot "adat.dat" u 2:3:4 lw 5
plot "adat.dat" u 1:2 w p ps 7
```