Számítógép Architektúrák

5. Gyakorlat

Hozzáférési jogok

- *chmod*
- *umask*
- *chown, chgrp*

chmod (már volt róla szó)

- csak a tulajdonos tudja átállítani ezeket a jogokat

r=4, w=2, x=1 pl:r+x=5

s-setuid bit /root jogosultságot igénylőprogramokhoz (u=rwxs)

- chmod 751 proba.txt
- chmod u+r+w, g-r, o-r-w-x proba.txt
- a=all mindenkinek meg lehet adni a jogosultságot
- chmod u=rwx, g=rx, o=r proba.txt

umask

- létrehozott fájljaink az umask által megadott jogosultságokkal jönnek létre
- azon számok összegét kell megadni, mely jogokat **nem** akarjuk megadni

- umask 033

umask példa

- $ umask u=rwx,g=rwx,o=
- $ mkdir fu
- $ touch bar

- $ ls -l

drwxrwx--- 2 dave dave 512 Sep 1 20:59 fu
-rw-rw---- 1 dave dave 0 Sep 1 20:59 bar

chown, chgrp

- az általunk birtokolt állomány tulajdonosát és csoportját megváltoztathatjuk

- $ chown broda *fájlnév*
- $ chgrp mib2005 fájlnév

- Normál user általában nem használhatja.
- Ha a jogot átadtuk, akkor nem vehetjük vissza, csak az adhatja, vissza, akinek átadtuk (vagy root).

Néhány parancs részletesebben

- **ls** lista készítése a directory-ról.
  -l -- jegyzékeket és állományokat jogosultságaikkal, tulajdonosaikkal stb. írja ki;
  -a -- a rejtett fájlokat is kiírja;
  -F -- jegyzék után"/" szövegállomány után semmit, bináris állomány után *-ot tesz;
  -1 -- a terminálra állománynevenként egy sort ír ki;
  -r -- az állományneveket fordított sorrendben írja ki(alapértelmezés: ABC sorrend)
  -R -- (ls -R ~) kiírja a megadott könyvtár tartalmát az összes alkönyvtárának tartalmával együtt;
  -t -- nem abc sorrendben, hanem időrendben írja ki a fájlokat

Néhány parancs részletesebben

- **ps** – futó folyamatokról ad információt.

  -e -- nem csak a terminálon futó folyamatokról ad listát

  - f --teljes, minden információra kiterjedő listát kér;

  -a -- az összes éppen aktív folyamatunk részletes listáját adja;

  -u -- (**ps -u login-name**) a megadott felhasználó éppen aktív folyamatainak részletes listáját adja;

  - x

Néhány parancs részletesebben

- **kill** PID – processz erőszakos terminálása

kill -9 PID -> nem várja meg, hogy a processz befejeződjön rendesen, így az eredmény elveszhet

- **news** – kiirja a rendszerrel kapcsolatos friss hireket (már, ha van ilyen)

Néhány parancs részletesebben

- **find**: nagyon hasznos, jól paraméterezhető kereső.

$ find ~ -name *.jpg
$ find ~ -name ”*.txt” -atime -7

+7 //hét napnál régebben nyúltak hozzájuk
-7 // hét napnál korábban nyúltak hozzájuk

Programok csoportosítása zárójelezéssel

Akkor előnyös, ha:

- két vagy több processz eredményét akarjuk ugyanarra a csőre kötni
- az operátorok precedenciájának átértékelését akarjuk elérni
- processz szeparálást akarunk elérni

Zárójeltípusok: ( ), { }

Zárójeltípusok példa

Parancs	Hatása
date; who	végrehajtja mindkét parancsot
date; who | wc	elsőt végrehajtja, másodikat wc-vel szűri
(date;who) | wc	a két parancs eredményét wc-vel szűri

Adatfolyam átirányítás (volt…)

- sort -x lista >**&** lista1

átirányítom az stdout-ot és error-t a lista1-be

- sort -X lista > proba 2 > **&1**

az error-ját átirányítja stout-ba

- grep -Q m lista>lista1 2>lista2

lista2 -be irányítja az stdoutját

Parancsbehelyettesítés

- Lényege, hogy a program eredményét nem a kimenetre küldjük, hanem egy másik program paraméterlistájába illesztjük.

A behelyettesítendő parancsot a ` ` aposztrófok közé kell tenni.

$ echo ”Jelenleg `who | wc -l` felhasználó van bejelentkezve”

$ echo ”A HOME jegyzekemben `ls ~/ | wc -w` darab állomány található”

Parancsbehelyettesítés

$ echo `ls -l ~/ | grep "^d...r*"|wc -l`" darab a csoport szamara olvashato jegyzek van a HOME konyvtaramban."

$ echo `ls -l ~/ | grep ".txt$"`

txt kiterjesztésű állományok kiíratása

Shell

1. Parancsértelmező parancsfeldolgozó
2. Programnyelv (script nyelv)

Shell, terminál

- A bejelentkezéstől a kijelentkezésig ezen keresztül kommunikálunk a rendszerrel.

- A shell-eknek különböző fajtáik vannak:
  - Bourne-shell (sh),
  - Bourne Again Shell(bash)
  - csh,
  - tcsh
  - korn (ksh)
  - zsh
- shell-enként a prompt változik
- Kilépés belőlük ^d, exit …

Shell script

- Végrehajtható szöveges állomány, amely UNIX parancsokat tartalmaz.
- Futtatásához szükséges:
  - **r és x jogosultság**
  - #! /bin/sh // a első sorba, ha ”./scripnév” –vel akarjuk indítani
  - ez tudatja a shellel, hogy scriptről van szó
  - ez a program fogja feldolgozni
  - ha nem adjuk meg a shell scriptben, akkor az aktuális shell fogja feldolgozni

- Futtatás: sh scriptnév vagy sh < scriptnév vagy ./sriptnév
- # a sor elejére; jelöli a kommentet.
- Nincs többsoros komment!

1. Példa

- elso.sh:

```
echo ”Hello vilag”
# ez meg egy komment

$ chmod +x elso.sh
$ ./elso.sh
```

Változók

- <u>környezeti változók</u> pl: **echo $SHELL**
  - set – környezeti változók lekérdezése
  - HOME – sajátjegyzék
  - PATH – jegyzéklista a parancsok kereséséhez
  - USER – felhasználónév
  - SHELL – alapértelmezett shell
  - PWD – aktuális jegyzék elérési útja
  - PS1 – prompt karakterlánc elérési útja
  - TERM – terminál típusa

Változók

- pozicionális paraméterek:
  - $1, $2 … , $9 //programnak paraméterként megadott értékek
  - $0 //a program nevét lehet elérni vele
  - $# a parancssori paraméterek száma
  - $$ a futó program folyamatazonosítója
  - $* valamennyi parancssori paraméter egyben

Változók

- felhasználó által létrehozott változók:
  - változó értékadása: x=4
  - ha hivatkozunk rájuk, akkor $ jelet kell elé rakni pl: **$ echo $x**

- **expr**: csak egész számokhoz, ciklusváltozókhoz
- aritmetikai műveletek : +,-,/,*;

$ expr 2 + 3 //kvótázás?

Shell scriptben semlegesiteni kell a muveleti jeleket!

2. Példa

A megadott második paraméterből kivonja az elsőt

$ echo `expr $2 - $1`

Shell változók exportja

Egy létrehozott shell változó mindig ahhoz a shell-hez tartozik, mely létrehozta és a shell gyermekeinek, ha tehát új shell-t indítunk, nem tudjuk a változókat automatikusan elérni, hanem exportálni kell őket az **export** paranccsal.

export

$ x=hello; echo $x; sh; echo $x

itt nem ír ki semmit, mivel egy másik shell-ben vagyunk

$ exit; echo $x

$ x=hello; export x

így már ha másik shell-t indítunk, akkor el fogjuk tudni érni!

read, sleep

- **read**: változó értékének bekérése

```
$ read x
$ echo $x
```

sleep – processz altatása megadott ideig.

```
$ sleep 30 #30 sec-re
```