UNIX-käyttöjärjestelmä

Luento 4

Timo Savola

<timo.savola@evtek.fi>

21. huhtikuuta 2006

Osa I

Shell

Lausekkeet

- Komentoriville kirjotettu komento on *lauseke*

```
echo "foo"
echo $USER
MUUTTUJA=1 ls -l
```

- Rivinvaihto erottaa lausekkeet
- Lausekkeet voidaan erottaa myös puolipisteellä

```
ls /bin ; ls /lib
echo "Hakemisto"; LS_BLOCK_SIZE=1 ls -l
touch tiedosto ; stat tiedosto ; rm tiedosto
```

Kommentit

- Komentoriviä ei suoriteta vaan se tulkitaan kommentiksi mikäli se alkaa # -merkillä

```
# Tämä ei ole lauseke
```

- Kommentti voi esiintyä myös lausekkeen perässä

```
mkdir ./work # Luodaan työhakemisto
```

Paluuarvot

- Ohjelmat kertovat onnistumisesta paluuarvolla:
  - 0 Ohjelman toiminnot suoritettiin onnistuneesti
  - 1...127 Ohjelmassa tapahtui jokin virhe
- Lausekkeen arvo on komennon paluuarvo mikäli komento voitiin suorittaa (muussa tapauksessa jokin virhekoodi)
- Erikoismuuttuja $? sisältää edellisen lausekkeen paluuarvon

```
echo $?
```

Lausekkeiden yhdistely

- Komentoja voidaan yhdistellä loogisilla operaattoreilla:
  - `&&` Ja
  - `||` Tai
- `&&` -operaattorilla yhdistetyistä komennoista jälkimmäinen suoritetaan vain jos ensimmäinen onnistui
- `||` -operaattorilla yhdistetyistä komennoista jälkimmäistä ei suoriteta jos ensimmäinen onnistui
- Lausekkeen paluuarvo on viimeisen suoritetun komennon paluuarvo
- Esimerkkejä:

```
mkdir eka && mkdir toka
stat /bin/tar || stat /usr/bin/tar ; echo $?
```

Alishellit

- Suluissa olevat lausekkeet suoritetaan *alishellissä*

```
(lauseke ; toinen-lauseke)
```

- Alishellissä asetetut muuttujat eivät vaikuta ylempään shelliin

```
N=1
(N=2 ; echo $N ; X=10)
echo $N
echo $X
```

- Alishellin paluuarvo on viimeisen lausekkeen paluuarvo

Komentokorvaus

- ‘ -merkkien (“takahipsu”) välissä olevan lausekkeen tulostetta voidaan käyttää toisessa lausekkeessa

```
stat ‘ls‘
```

Syötteen ja tulosteen uudelleenohjaus

- Komentojen syöte ja tuloste on normaalisti liitetty shellin käyttämään terminaaliin
  - Syöte tulee näppäimistöltä
  - Tuloste näkyy näytöllä
- Syöte voidaan lukea tiedostosta:

```
komento < tiedosto
komento 0< tiedosto
```

- Tuloste voidaan kirjoittaa tiedostoon:

```
komento > tiedosto
komento 1> tiedosto
komento 2> tiedosto
```

- Esimerkkejä:

```
ls -l > listaus.txt 2> /dev/null
cat < listaus.txt
```

Putket

- Komentoja voidaan *putkittaa* (pipe) yhdistämällä edellisen tuloste seuraavan syötteeseen:

```
komento1 | komento2
```

- Näin muodostettuja putkia (pipeline) voidaan käyttää lausekkeissa yksittäisten komentojen sijaan
- Ensimmäisen komennon syöte tulee terminaalista ja viimeisen tuloste tulee terminaaliin

- Esimerkkejä:

```
ls -l /bin | less
cat kurssit/*.txt | grep Pekka | sort > pekka.txt
```

Työt

- Shell pitää kirjaa käynnissä olevista *töistä* (komennoista)
- Töitä voi suorittaa etu- ja taka-alalla, pysäyttää ja keskeyttää
- Normaalista komennosta tulee etualalla suoritettava työ
- & -merkkiin päättyvä komento käynnistetään taka-alalle

```
find . -name file.txt &
```

- Töillä on numerot joilla niihin voi viitata
  - Shell kertoo työn numeron ja tilan jos sitä ei suoriteta etualalla

Töiden hallinta

- `jobs` -komento listaa käynnissä olevat työt
- `fg` -komento siirtää työn etualalle

  ```
  fg 1
  ```

- `bg` -komento siirtää työn taka-alalle

  ```
  bg 1
  ```

- Etualalla olevan työn voi pysäyttää painamalla Ctrl-Z
  - Lähettää `STOP`-signaalin
- Etualalla olevan työn voi keskeyttää painamalla Ctrl-C
  - Lähettää `INT`-signaalin
- Sisäänrakennetulla `kill` -komennolla voi lähettää signaaleja taka-alalla oleville tai pysäytetyille töille

  ```
  kill %1
  ```

Osa II

Skriptit

Ei-interaktiivinen shell

- Normaalisti käynnistetty shell on interaktiivinen
  - Tulostaa komentokehotteen
  - Lukee asetustiedostoja ym.
- Shell ei ole interaktiivinen jos syöte ei tule terminaalista

  ```
  /bin/sh < komennot
  echo "ls -l"| /bin/sh
  ```
- Shellille voi antaa komentotiedoston nimen parametrina

  ```
  /bin/sh komennot
  ```

Skriptit

- Tavallisten ohjelmien lisäksi voidaan suorittaa *skriptejä*
  - Tiedosto, joka sisältää jonkin ohjelman ymmärtämiä komentoja
  - Tulkkiohjelma kerrotaan kirjoittamalla ensimmäiselle riville:

    #! *ohjelman-polku*
- Skriptit suoritetaan samalla tavalla kuin normaalit ohjelmat
  - Suojauksen tulee sallia suoritus (`chmod +x`)
- Shell-skripti esitellään kirjoitamalla rivi:

  #! `/bin/sh`
- Tämän jälkeen tulevat rivit (lausekkeet) suoritetaan shellillä
- Shell-skripteille käytetään monesti päätettä “`.sh`”

Parametrit

- Shell-skripteille voidaan antaa komentoriviparametreja
- Niiden arvot voidaan lukea erikoismuuttujista:

  `$1`, `$2`, `$3`, ...
- Parametrien määrä voidaan lukea erikoismuuttujasta:

  `$#`

- Esimerkkiskripti:

```
#! /bin/sh
echo "Kohdehakemiston nimi: $1"
echo "Kohdetiedoston nimi: $2"
echo "Kopioitava tiedosto: $3"
mkdir $1
cp $3 $1/$2
echo "Kopiointikomennon paluuarvo: $?"
```

- Skriptin käynnistys luo shellin lapsiprosessiksi toisen shellin jossa skriptin komennot suoritetaan
- Komentoja sisältävä tiedosto voidaan haluta suorittaa nykyisessä shellissä
  - Muuttujiin ym. tehdyt muutokset halutaan näkymään nykyisessä shellissä
- Tätä varten on olemassa “.” -komento (piste):

```
. tiedosto.sh
```

Osa III

Työkaluja

man

- Näyttää “manuaalisivun” halutusta aiheesta:

  `man` *aihe*

- Manuaalisivut käsittelevät lähinnä järjestelmän komentoja, asetustiedostoja ja ohjelmointirajapintoja
- Yleisimmät komennot on dokumentoitu man-sivuilla
  - Shellin toiminnot on dokumentoitu sivulla “builtins”

- Sivut on jaettu aihepiireihin (section)
  - Samanniminen sivu voi löytyä useammasta aihepiiristä
  - Haluttuun sivuun voidaan viitata yksiselitteisesti kirjoittamalla aihepiirin numero ennen sivua
- Kuvaukset eri aihepiireistä saa komennolla:

  `man 7 man` (näyttää sivun “man” aihepiiristä 7)
- Man-sivuihin viitataan tekstissä usein kirjoittamalla:

  nimi (*numero*)

- `apropos` ja `whatis` -komennoilla voi etsiä man-sivuja

cat

- *Concatenate*
- Tulostaa parametreina saamiensa tiedostojen sisällöt
- Tulostaa syötteensä mikäli parametreja ei annettu
- Esimerkkejä:

```
cat /proc/version
cat tieto1.txt tieto2.txt > tiedot.txt
cat > kirje.txt
```

less (more)

- *Less is more*

- Antaa käyttäjän selata parametrina saamansa tiedoston sisältöä
- Näyttää syötteensä mikäli parametria ei annettu
- Toimintoja:
  - ↓↑ Selaa tiedostoa
  - g Hyppää tiedoston alkuun
  - G Hyppää tiedoston loppuun
  - / Etsi tekstiä
  - q Poistu ohjelmasta

- Esimerkkejä:

```
less /proc/cpuinfo
cat *.txt | less
```

nano (pico)

- Helppokäyttöinen tekstieditori

- Antaa käyttäjän luoda tai muokata tekstitiedostoja
- Parametrina voi antaa olemassaolevan tai uuden tiedostonimen
- Toimintoja:

Ctrl-O Tallenna tiedosto
Ctrl-X Poistu ohjelmasta