Linux-harjoitus 9

Linuxin mukana tulevat komentotulkit (mm. bash, tcsh, ksh, jne…) sisältävät ohjelmointikielen, joka on varsin tehokas ja ilmaisuvoimainen. Tähän yhdistettynä unix-maailmasta tutut tehokkaat apuohjelmat, voi sanoa, ettei Linuxissa jonkun pienen ohjelmointiongelman ratkaisemiseen tarvita mitään muuta ulkoista ohjelmointikieltä.

Hieman linkkejä:

http://cs.stadia.fi/~kuivanen/linux/kom.php, lyhyt ohje komentoriviohjelmointiin.

http://ameba.lpt.fi/~kettueer/unix-perusteet/komentotiedostot.html, toinen ohje komentoriviohjelmointiin (Eero Kettunen).

http://www.ling.helsinki.fi/~mlounela/unixk/sisalto.html, Hieman yleisluontoisempi ohje, sivuaa myös komentoriviohjelmointia.

http://koti.welho.com/jkajaste/bash.html, Jaakko Kajasteen laajempi opas komentoriviohjelmointiin.

Lisää löytää vaikkapa antamalla Googlelle hakusanoiksi esim. ”shell ohjelmointi”.

Ohjelmien suoritus unix-pohjaisissa käyttöympäristöissä

Unix poikkeaa tässä Windows-maailmasta ratkaisevasti. Kun Windows-maailmassa etsitään suoritettavaa ohjelmaa ensiksi työhakemistosta ja mennään vasta sitten hakemaan ns. hakemistopolusta (PATH), unix-pohjaisissa käyttöjärjestelmissä mennään suoraan hakemaan ohjelmaa hakemistopolusta. Tällä estetään ns. ohjelman kaappaukset.

Myöskään tiedoston nimen pääte ei määrää sitä, minkä tyyppinen ohjelma on kyseessä. Jos unixissa laittaa ohjelman (skriptin) nimeksi *ohjelma.sh*, on kirjoitettava näkyviin koko ohjelman nimi, viimeistä kirjainta myöten.

Skriptille tulee aina ennen suoritusta antaa suoritusoikeudet. Suoritusoikeus annetaan seuraavalla komennolla:

```
$ chmod u+x ohjelma.sh
```

Esimerkki antaa omistajalle (u – user) lisää (+) suoritusoikeuden (eXecute)

Miten skripti käynnistetään unixissa?

Jos työhakemisto on polussa, voi skriptin käynnistää aivan samoin kuin Windowsin komentoriviltä, eli näin:

```
$ ohjelma.sh
```

Jos työhakemisto ei ole polussa, pitää ohjelman sijainti kertoa. Yksinkertaisimmillaan se voidaan sitoa työhakemistoon näin:

```
$ ./ohjelma.sh
```

Piste viittaa aina työhakemistoon. Näin ollen yllä olevassa esimerkissä kerrotaan, että ”suorita työhakemistossa oleva *ohjelma.sh*-tiedosto”

Muitakin tapoja suorittaa skripti on, esimerkiksi antamalla se parametrinä komentotulkille.

Mitkä hakemistot ovat polussa (PATH)?

Perinteisesti bin-nimiset hakemistot ovat tarkoitettu ohjelmille (*/bin*, */usr/bin*, */usr/local/bin*, jne). Myös käyttäjän kotihakemiston alle tehty *bin*-hakemisto on olemassa valmiina polussa, vaikkei itse hakemistoa vielä olekaan. Näin ollen jos haluat tehdä sellaiseen paikkaan skriptisi, josta voit suorittaa ne missä vain, tee kotihakemistoosi hakemisto *bin*, jonne talletat kaikki skriptisi. Seuraavat tehtävätkin voit toteuttaa siellä.

Ensimmäinen skripti

Kaikki itseään kunnioittavat ohjelmointioppaat aloittavat seuraavalla esimerkillä. Kirjoita se haluamallasi editorilla ja anna sille nimeksi vaikkapa *hello*:

```
#!/bin/sh
echo ”Hello world”
```

Anna suoritusoikeudet sille ja testaa se.

Ensimmäinen rivi määrittää käytettävän komentotulkin. *Sh* on Unixien perus-komentotulkki, Bourne Shell. Sitä käytetään yleisimmin, koska se on mukana kaikissa unixeissa. Perus-linux-komentotulkit, kuten *bash* ja *zsh*, perustuvat tähän komentotulkkiin ja niiden komentokieli on jokseenkin samanlainen. C-komentotulkkihaaran kieli on taas näistä poikkeavaa. Tähän haaraan ei tällä kertaa puututa. Yleensäkin suurin osa komentoriviohjelmointioppaista keskittyy ns. Bourne-haaran kieliin.

Tehtäviä

1. Lisää aiemmin esillä olleeseen hello-skriptiin kahden rivin väliin rivi, jolle kirjoitat komennon *clear*. Testaa skriptiä ja katso, miten tulos muuttui.

2. Kirjoita seuraava skripti ja mieti, mitä se tekee

   ```
   #!/bin/sh
   echo ”Moi, mikä sinun nimesi on?”
   read nimi
   echo ”Hei, minusta $nimi on kaunis nimi.”
   ```

3. Vaihda jälkimmäiseen echo-lauseeseen kaksinkertaisten lainausmerkkien tilalle yksinkertaiset (heittomerkki – ’). Kokeile nyt samaa skriptiä. Mitä havaitsit?

4. Lisää samaan skriptiin seuraava rivi loppuun:

   ```
   echo ”Tiesitkö, että tänään on vuoden `date +%j`.s päivä?”
   ```

Tuo ”hipsu”-merkki löytyy näppäimistöltä kysymysmerkin vierestä. Saatat tarvita välilyöntinäppäimen painallusta merkin jälkeen. Mitä tulostui?

5. Seuraava skripti kertoo nimensä sekä parametrinsä:

```
#!/bin/sh
echo ”Tämän skriptin nimi on $0”
echo ”Ensimmäinen parametri on $1 ja toinen $2”
echo ”Annoit yhteensä $# parametriä, jotka olivat: $*”
```

Kokeile skriptiä antamalla komennon perään muutama parametri, vaikkapa näin:

```
$ param.sh eka toka kolmas neljas
```

Numerot 0 – 9 ovat siis varattu komentorivin parametreille. Normaalisti tieto välitetäänkin skripteille komentorivin parametreillä eikä millään read-lauseella.

6. Mitä seuraava tekee:

```
#!/bin/sh
if [ $1 -gt 10 ]
then
   echo ”parametri oli suurempi kuin 10”
else
   echo ”parametri oli 10 tai pienempi”
fi
```

7. Entä seuraava:

```
#!/bin/sh
summa=0
for luku in 1 2 3 4 5 6
do
  summa=`expr $summa + $luku`
done
echo "Lukujen 1 - 6 summa on $summa."
```

8. Miten saisit yo. skriptin lukemaan luvut parametreinä?

9. Tee skripti, joka ilmoittaa sille parametrinä annettujen kahden luvun tulon.

10. Kuinka saat edellisen skriptin tarkistamaan, annettiinko sille jokin muu määrä kuin kaksi parametriä?

Awk

Awk on eräs Unix-maailman skriptikielistä. Sen nimi tulee todellisuudessa kolmesta nimestä: Aho, Weinberger, Kerningnhan. Se on ns. normaaleihin ohjelmointikieliin tottuneelle varsin erilainen kieli, mutta se on erittäin vahva työkalu sarakemuotoisen datan käsittelyyn. Siksi sillä on varsin paljon käyttötarkoituksia Unix/Linux-ympäristöissä. Gnu-versioon awk:sta löytää parhaan manuaalin osoitteesta *http://www.gnu.org/software/gawk/manual/html_node/index.html* . Toinen vastaava on *http://www.cs.uu.nl/docs/vakken/st/nawk/nawk_toc.html*. Suomeksi awk:sta kirjoittaa lyhyesti Jukka Korpela osoitteessa *http://www.cs.tut.fi/~jkorpela/unix/5.11.html*.

Awk on varsin monipuolinen työkalu. Tässä dokumentissa esitellään muutama sen ominaisuuksista. Ominaisuudet käydään läpi esimerkkien avulla.

Awk-esimerkkejä

Awk on sarakeorientoituneen datan käsittelyyn tarkoitettu työkalu. Esimerkiksi

```
ls -l | awk '{print $9" "$5}'
```

tulostaa tiedostolistauksesta tiedoston koon sekä tiedoston nimen. $-merkillä ja luvulla siis viitataan sarakkeeseen. Sarakelaskuri alkaa edellisen perusteella 1:stä.

Jos kenttäerotin on jotain muuta kuin välilyönti taikka tabulaattori, sen voi kertoa -F-optiolla, esim. näin:

```
awk -F: '{print $1}' /etc/passwd
```

Tämä tulostaa ensimmäisen kentän */etc/passwd* -tiedostosta.

Olkoon tekstitiedosto, jossa on talletettuna seuraavat tiedot:

```
Kalle      100
Elmeri     250
Kalle       90
Juuso      120
Uuno        90
Juuso      250
```

Kirjoitetaan seuraavat asiat tiedostoon nimeltään *laske*:

```
BEGIN      { print ”Frekvenssit:”}
           { sum[$1] += $2 }
END        { for (name in sum)
             print name, sum[name] }
```

Sen jälkeen kokeillaan komennolla

```
awk -f laske nimet
```

mitä tapahtuu. Huomaa optio -f ja mitkä parametrien roolit ovat. Mitä komennon suoritus tekee?

Seuraava awk-esimerkki etsii pisimmän rivin pituuden tiedostosta *data*:

```
awk '{ if (length($0) > max) max = length($0) }
 END { print max }' data
```

Olkoon seuraavanlainen tiedosto:

```
Uuno   100 97 58
Lissu  84 72 93
Ulla   72 92 89
Elmeri 99 34 44
```

Seuraava operaatio laskee kunkin henkilön kohdalla olevien lukujen keskiarvon:

```
awk '{ sum = $2 + $3 + $4 ; avg = sum / 3 print $1, avg }' grades
```

kun tiedoston nimi on *grades*.

Awk-tehtäviä

1. Toteuta edellisessä kappaleessa esitetyt esimerkit.

2. Toteuta nimi-esimerkin koodi siten, että yhteissummien sijasta lasketaan nimien esiintymismäärät taulukossa.

3. Tulosta */etc/passwd*-tiedoston user-id-numerot ja tulosta ne numerojärjestyksessä.

4. Mitä seuraava awk-ohjelma tekee?

```
   {
     for (i = 1; i <= NF; i++)
         freq[$i]++
}
END {
     for (word in freq)
         printf "%s\t%d\n", word, freq[word]
}
```

5. Laske vastaavalla operaatiolla tiedoston rivien lukumäärä.Seuraava awk-operaatio laskee, montako marraskuussa modifioitua tiedostoa on tiedostolistauksessa:

```
ls -l | awk '$5 == "Nov" { sum++ } END { print sum }'
```

Toteuta nyt edellisen perusteella awk-ohjelma, joka laskee marraskuussa (taikka jossain muussa kokeilun kannalta sopivassa kuussa) muokattujen tiedostojen koot yhteen.

6. Kuinka lasket awk:n avulla tiedostojen koot yhteen?

7. Haluat tietää, montako yli 10 000 tavun tiedostoa hakemitossasi on. Laadi awk-ohjelma, jolla lasket niiden kokonaistavumäärän ja lukumäärän.

8. Luo jokin pieni tekstitiedosto. Tee sen jälkeen awk-ohjelma, joka laskee kaikkien yli 50-merkkisten rivien lukumäärän tiedostosta.

9. Tutustu awk:n valmiiksimääriteltyihin muuttujiin. Mikä muuttuja kertoo tiedoston rivien lukumäärän?