Johdatus komentoriviohjelmointiin

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.html, lyhyt ohje komentoriviohjelmointiin.
http://www.ling.helsinki.fi/~mlounela/unixk/sisalto.html, Hieman yleisluontoisempi ohje, sivuaa myös komentoriviohjelmointia.
http://users.tkk.fi/~mkousa/shell-doc/, Mika Kousan kometotulkkien ohjelmointia käsittelevä sivusto.
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:

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

   Mitä se tekee?

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. Toteuta seuraava skripti. Ole tarkkana kirjoitusasun kanssa!

   ```
   #!/bin/sh
   echo ”Mietin yhtä lukua, yritä arvata se:”
   read arvaus
   luku=`expr $arvaus + 1`
   echo ”Ajattelin lukua $luku. Hävisit niukasti!”
   ```

6. Edellisten tehtävien perusteella: tee skripti, joka kysyy käyttäjältä kahta lukua ja kertoo niiden summan. Huom. *expr* ei osaa käsitellä kuin kokonaislukuja.

7. Jos muutat edellisen tehtävän laskemaan summan sijasta tulon, mitä tapahtuu?

8. 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.

9. Mitä seuraava tekee:

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

10. 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."
    ```

11. Miten saisit yo. skriptin lukemaan luvut parametreinä?