Sormet skripteihin

ITKA203 Käyttöjärjestelmät -kurssin Demo 4 kesällä 2007. Paavo Nieminen / Jyväskylän yliopiston Tietotekniikan laitos.

1 Mistä tässä harjoitteessa on kyse

Tässä julkaistaan Bourne Shellin pikaohje, joka tulee tentissä liitteeksi (mukaelleen aiempien tenttien C-shell -vastaavaa). Skripteistä on vapaaehtoinen yhden pisteen arvoinen palautustehtävä.

Tavoitteet:

- demossa 1 nähty interaktiivisen shellin käyttö laajennetaan skriptien tekemiseen
- näet ja jopa jonkin verran teet skriptiohjelmointia, mikä toivottavasti valaisee mahdollisuuksia ja käyttötarkoituksia moiselle.

Kesäkurssilla 2007 demo 4:n yhteydessä oli myös toinen osio:

- Harjoitustyöohje on nyt julkaistu; sitä saattoi käydä läpi demossa. Omasta harjoitustyöstä tulee yksi piste tenttiin, jos sen saa aikarajaan mennessä palautettua.

2 Skriptit

Tuolla on lisätietoa; hyvän näköinen intro:

- http://steve-parker.org/sh/sh.shtml

Tuolla on myös; kattavan oloinen saitti:

- http://www.shelldorado.com/

Siellä on esimerkkejä ym. Ja Googlella löytyy lisää.

2.1 Esim: Bourne Shellin rakenteita esimerkkikoodina

Tämän kertaiset esimerkkikoodit ovat paketissa demo4.zip kurssin kotisivulla. Siis jalavassa saat paketin esim. näin:

```
wget http://www.cc.jyu.fi/~nieminen/kj07kesa/demo4.zip
unzip demo4.zip
```

Skripti `shrakenteita.sh` demonstroi ohjelmointirakenteita Bourne Shellissä, ja se on samalla pieni esimerkki **ympäristömuuttujasta** (*environment variable*).

2.2 Esim: apuohjelma find

Usein skripteissä ei kannata yrittääkään tehdä kaikkea shellin rajallisilla ohjelmointiominaisuuksilla, vaan kannattaa käyttää apuohjelmia. Täytyy tosin muistaa myös, että kaikkia apuohjelmia ei ole asennettu kaikkialle, ja niiden eri versiot voivat erota toisistaan esim. argumenttien muodon suhteen. Yhteensopivien ja alustariippumattomien skriptien tekeminen on pidemmän päälle taito- ja tietolaji. Katsotaan kuitenkin kahta kätevää työkalua: `find` ja `grep`. Yhdessä nämä ohjelmat tarjoavat samanlaista toiminnallisuutta kuin esim. Windowsin etsintätyökalu (se, jossa on defaulttina sellainen söpö kultakarvainen koira-animaatio, joka hyppii ja hymyilee ja heiluttaa häntää sekä painajaismuistojeni mukaan myös läähättää). Tietysti näitä komentorivityökaluja voi liittää skripteihin, jolloin mahdollistuu paljon automaattista tietojenkäsittelyä (ATK), mikä on eri asia kuin Animaation Tuijottelu, vaikkapa Koiran (ATK). Ehkä edellämainittu oli viimeinen Asenteellisuudessaan Tuomittava Kommenttini tällä ATK-kurssilla.

Find etsii tiedostoja argumenttina annettujen ehtojen mukaisesti. Se on asennettu useisiin unixeihin. Tässä henkilökohtaiseen tietoturvaan liittyvä esimerkki:

```
#!/bin/sh
find $HOME -maxdepth 1 -type d -\! -name ".*" | \
```

```
while read a
do
  chmod go-rwx $a
done
```

Tämän skriptin `find` -komento etsii kotihakemistossa (`$HOME`) olevat hakemistot (`-type d`), joiden nimi ei ala pisteellä (`-\! -name ".*"`, jättää siis piilotiedostot huomioimatta), ja syöttää tulosteensa putkella sh-shellin `while` -toistorakenteelle. Jokaisen findin löytämän hakemiston kohdalla poistetaan hakemistolta kaikki oikeudet kaikilta muilta kuin käyttäjältä itseltään (`chmod go-rwx`). Find rajoittaa tiedostojen etsimisen kotihakemiston sisältämiin hakemistoihin menemättä alihakemistoihin (`-maxdepth 1`). Tämä nyt oli vain yksi esimerkki...

2.3 Esim: apuohjelma grep

Grep on toinen kätevä, aika usein saatavilla oleva apuohjelma. Sillä etsitään tiedostoista rivejä, joilla on tietynlaista tekstiä.

Aion selata Käyttöjärjestelmät -kurssin harjoitustyöt läpi skriptillä, (tai jollain vastaavalla) joka sisältää seuraavan rivin (tai jotain vastaavaa):

```
# Tulostetaan opiskelijan tuottamat vastausrivit:
grep -n -B2 -A3 "HARKKA:" $1
```

Eli grepillä voi etsiä tekstitiedostoista rivejä, joilla on merkkijono. Hyödylliseksi grepin käyttö muuttuu siinä vaiheessa, kun ymmärretään, että sillä voi etsiä täsmällisen merkkijonon sijasta sanarunkoja. Esimerkiksi rivit, joilla on `HAR`, sitten mitä tahansa merkkejä ja ainakin yksi kaksoispiste:

```
grep "HAR.*:" fname.txt
```

Tai itse asiassa mitä tahansa ns. säännöllisten lausekkeiden mukaisia merkkijonovastaavuuksia:

```
grep "[Hh]a[r]*joitus[t T]*yö[A-Za-zöääÖÄÅ]*:" fname.txt
```

Em. komento löytäisi rivit, joilla on esimerkiksi jokin seuraavista merkkijonoista:

```
Harjoitustyö:
harjoitustyö:
Hajoitustyö:
harrrrrrjoitus yöllä:
Harjoitus työläs:
...
```

[reunahuomautus: Säännölliset lausekkeet (*regular expressions*, "*RegEx*") kannattaa jossain vaiheessa opetella, jos ei vielä ole tulleet vastaan. Niitä käytetään mm. Javan merkkijonoluokkien `match` -metodeissa ja ylipäätään aika monessa paikassa. Ne muuttuvat vielä tehokkaammiksi siinä vaiheessa, kun oppii käyttämään ns. ryhmiä (groups) joiden avulla voi korvata regexpin osien sisältöjä uudella tekstillä, toisillaan tai vaihtaa niitä keskenään, tai ylipäätään ottaa niitä irti tietorakenteisiin käsittelyä varten. Säännöllisten lausekkeiden teoriaa ja koneiston toteutusta meillä sisältyy kurssiin nimeltä Automaatit ja kieliopit. Niiden käyttötaito kuuluu joka tapauksessa IT-yleissivistykseen.]

2.4 Palautustehtävä

Yhden pisteen arvoisena palautustehtävänä voit muokata demo1:n lisukemateriaalissa ollutta grafiikkaskriptiä seuraavin tavoin:

- skriptin pitää tarkistaa, onko kukin väliaikainen tiedosto jo olemassa eikä tehdä sitä uudelleen enää, jos se on.
- skriptin pitää tarkistaa, onko komento muotoa `./mun.sh --removetemps` ja siinä tapauksessa poistaa kaikki väliaikaiset kuvatiedostot eikä tehdä mitään muuta.
- skriptiä pitää voida käyttää muodossa `./mun.sh --times 4` eli ensimmäinen argumentti `--times` ja toinen argumentti numero, jolloin skriptin pitää tehdä kuvasta useita erilaisia kopioita numeroparametrissa kerrottu määrä. Riittää, että tehdään vain otsikkorannusta eri nimisiä kopioita; muiden väliaikaisten tiedostojen päälle voi kirjoittaa samalle nimelle jokaisella kierroksella. Ne on siis generoitava uudelleen niin että plasmaefekti ja bitit ovat satunnaiset ja erilaiset joka kerta.
  Tulostiedostot on nimettävä `otsikkorantu_N.jpg` missä `N` on numero 1, 2, ... viimeisenä parametrinä annettu luku.

Rajoituksia:

- Väärin syötettyjä argumentteja ei tarvitse nyt tarkistaa, vaikka oikeasti skriptin kuuluisi kaikki sellainen tehdä; muuten sen kanssa joutuisi pitkällä tähtäimellä vaikeuksiin! Koodia ja huomioitavaa tulisi kuitenkin liikaa tämän kurssin tavoitteisiin nähden. Meillä on ajoittain (vai satunnaisin väliajoin?) luennoitava jatkokurssi shell-ohjelmoinnista. Asiaa voi myös opiskella lisää itsenäisesti, kuten mitä tahansa...
- Ei tarvitse tehdä mitään ihmeellisiä muutoksia mun höperöön esimerkkiskriptiin, eli esim. väliaikaiset tiedostot käsitellään nyt työskentelyhakemistossa. Oikeasti ne pitäisi laittaa `/tmp/` -nimiseen hakemistoon (ja ne pitäisi nimetä niin että yhdenaikaisesta suorituksesta ei tule synkronointiongelmaa eli samaa skriptiä pitäisi voida ajaa eri käyttäjien toimesta yhtäaikaa... eli ei pidä joutua vahingossa käyttämään väliaikaisia tiedostoja jaettuna resurssina) ja täytyisi

huolehtia, että ne aina poistetaan esim. jos skripti lopetetaan kesken antamalla sille signaali... Nyt ei tarvitse näistä huolehtia... “Oikeissa töissä” kuitenkin sitten tehdään asiat kunnolla, eikö niin... :-)

Näissä tulee tutuksi Bourne Shellin perusrakenteet. Niihin on pikabriiffi tämän demomateriaalin lopussa, ohjausta saa tilaisuuksissa tarvittaessa, ja netti on pullollaan materiaalia skriptiohjelmoinnista.

Palauta mail-ohjelmalla jalavasta siten kuin aiemmatkin harkat; sähköpostin otsikoksi tällä kertaa “neljas vastaukseni” (huomaa `a`-kirjain, ei ääkkönen!). +1 piste ekaan tenttiyritykseen, jos palautus on tehty ennen kesän ensimmäistä tenttikertaa.

3 Harjoitustyö

Tämän osion ohjeet ovat harjoitustyöohjeessa.

Palautustehtävä: Palauta harkkasi työohjeessa ilmoitetulla tavalla 26.7.2007 klo 23:59 mennessä (ennen kesän ensimmäistä tenttipäivää), niin saat +1 pistettä tenttiin.

4 Liite: Yhden sivun sh-luntti

Alla on tentinkin liitteenä jaettava lunttilappu, jossa esimerkkien kautta muistutetaan joistakin Bourne Shellin piirteistä. **HUOM:** Tarkoituksella tenttiluntissa ei tulla mainitsemaan sitä, että skriptin ensimmäisellä rivillä pitää kertoa, millä nimenomaisella shellillä se ajetaan... Se **pitää kuitenkin vastauksessa olla**, oikealla syntaksilla ja siitä paikasta, josta yleensä Bourne Shell -ohjelma löytyy eli `#!/bin/sh`. Siitä tietää, onko aihetta yhtään opeteltu, mietitty ja tehty (tai edes *nähty* tämä demopaperi tai skripteistä kertonut luento...) vai yritetäänkö vaan sommitella palikoita paikoilleen kylmiltään.

```
Joitakin ohjelmointirakenteita:
    muuttuja=57               # tarkka syntaksista: ei välilyöntejä!
    muuttuja="eki"; muuttuja="$muuttuja$muuttuja"

    if EHTO                   # Myös:  if EHTO; then ...; \
    then                      #        elif EHTO; then ...; fi
      ...                     #
    fi                        #

    for muuttuja in LISTA     # muttujan "esittelyssä" /asetuksessa ei $
    do
      ... jotakin $muuttuja jotakin ... # muuttujan käytössä $muuttuja
    done

    while EHTO; do ... ; ... ; done     # käskyerotin rivinvaihto tai ;

    # Tee jotain syötteen kaikille riveille:
    while read rivi; do echo "luin: $rivi" ; done < rivit.txt

    case $hanen_nimensa in
      kirsimarja)
          echo "Hei Kippe" ;;
      eskomatias)
          echo "Moi E.M." ;;
    esac

    Aliohjelmat mahdollisia&hyödyllisiä, mutta ei käsitellä ITKA203:lla.

Joidenkin ehtojen käyttöä (Välilyönnit merkityksellisiä! "[" on itse
asiassa komento ja loppuosa on sen argumenttilista!):
    [ -d TIEDOSTONIMI ]       # tosi, jos on hakemisto
    [ -f TIEDOSTONIMI ]       # tosi, jos on tavallinen tiedosto
    [ ! -f TIEDOSTONIMI ]     # tosi, jos ei olemassa tai ei tavallinen
    [ "$muuttuja" -le "57" ]  # tosi, jos muuttuja <= 57 (myös lt,gt,ge)

Argumenttien käyttö:   (yli 9 arg mahd., mutta ei käsitellä ITKA203:lla)
    echo "Tämän skriptin nimi on $0. Eka argumentti $1, neljäs $4"

Joitakin sisäänrakennettuja toimintoja:
    exit VIRHEKOODI              # koodi 0 tarkoittaa onnistumista
    cd                           # vaihtaa skriptin työhakemistoa
    echo                         # kaiuttaa tekstiä
    let muuttuja=$muuttuja+3     # (perusaritmetiikkaa)

Erikoismerkkejä (tuttuja interaktiivisesta käytöstä; toimii skripteissä):
    <   >   >>   |   `KOMENTO`
```