bash

ohjelmointikielenä

Bourne-again shell

Metropolia

Oppimistavoitteet

- Tutustua shell komentotiedostojen perusperiaatteisiin
  - bash komentotulkki
- Tutustua komentotiedostojen ohjelmoinnin perusperiaatteisiin
- Osata vähintäänkin lukea komentotiedoston toiminnallisuus
- Oppia laatimaan yksinkertaisia komentotiedostoja

Sisältö

- komentotulkki
- aliakset
- komentorivin evaluointi
- muuttujat: shell-muuttujat vs. ympäristömuuttujat
- syöttö- ja tulostus
- komentoriviargumentit
- kontrollirakenteet
  - ehdollinen lause (if) ja testilauseke
  - valintalause (case)
  - toisto (while, until, for, select)
  - kontrollin siirto (break, continue, return)
- funktiot
- keskeytyssignaalien käsittely
- taulukot
- automaattisesti suoritettavat komentotiedostot
- kertauskysymyksiä

Opiskelumateriaalia

- bash-komentotulkin yksityiskohdat selviävät esim. lukemalla järjestelmän opastussivua
  - man bash
- Mukavampaa opiskelumateriaalia löytyy myös Internetistä, esim.
  - Bash by example -sarja
    http://www-128.ibm.com/developerworks/linux/library/l-bash.html
    http://www-128.ibm.com/developerworks/linux/library/l-bash2.html
    http://www-128.ibm.com/developerworks/linux/library/l-bash3.html
  - Bash Guide for Beginners
    http://tldp.org/LDP/Bash-Beginners-Guide/html/
  - BASH Programming - Introduction HOW-TO
    http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Komentotulkki (shell)

- On ohjelma, jonka kautta voi käynnistää muita ohjelmia
  - tekstipohjainen
- On nimeltään
  - sh (bourne shell), bash (bourne-again shell), ksh (korn shell)
  - csh (C-shell), tcsh (tenex csh)
  - zsh, ...
- Keskeiset ideat samoja, eroja komennoissa ja syntaksissa
- Komentoja voi antaa interaktiivisesti tai komentoja voi kirjoittaa isomman joukon etukäteen erikseen käynnistettäväksi komentotiedostoksi
- Myös "korkean tason" ohjelmointikieli

Komentotulkki

- Tunnistaa kaikki ohjelmointikielen peruspiirteet
  - muuttujat
  - loogiset ja aritmeettiset lausekkeet
  - viittaukset komentorivillä annettuihin argumentteihin
  - kontrollirakenteet: ehtolause, valintalause, toistolauseet
  - syöttö ja tulostus
  - funktiot
- Muita piirteitä
  - komentojen, käyttäjätunnusten ja tiedostonimien täydennys painettaessa sarkainta
  - history-mekanismi komentojen uudelleensuorittamiseksi
  - komentojen uudelleennimeäminen alias-mekanismilla
  - mahd. saada ilmoituksia esim. tdstoa ylikirjoitettaessa
    - valitsin -i (interactive)

Komentorivin syntaksi

- Komentorivi muodostuu komentosanasta, sen mahdollisista valitsimista ja argumenteista

```
komento [-valitsimet] [arg2] ... [argn]
```

- Komento
  - komentotulkkiin koodatun toiminnon (komennon) tunnus
  - erillisen, järjestelmään kuuluvan ohjelman (komennon) tunnus
  - käyttäjän itsensä laatiman ohjelman (komennon) tunnus
    - käännettyä koodia
    - tekstimuodossa olevia komentoja
    - komentotiedosto
- Valitsin
  - modifioi tai tarkentaa komennon toiminnallisuutta
- Argumentti
  - tiedostonimi, avainsana tms., johon komento kohdistuu

History-mekanismi

- Shell liittää komentoriviin tapahtumanumeron
  - tapa viitata aiemmin suoritettuun komentoon: ! ja "hakukomennot"
  - komennon voi suorittaa uudelleen
  - komentoa tai sen osaa voi käyttää uusissa komennoissa
- Ympäristömuuttuja HISTSIZE
  - arvona tiedostossa `~/.bash_history` pidettävien komentojen lukumäärä
  - oletus 500
- Komennolla history saa luettelon talletetuista komennoista
- Aiemmin annettuja komentoja voi selailla nuolinäppäimin

kehoite

```
$ history
1 emacs eppu
2 cat eppu dippu > puppu
3 mail hakkinen -s 'Se dippu..' < dippu
$ !!                    viimeisin
$ !-1                   toiseksi viimeisin
$ !2                    komento 2
$ !c                    viimeisin c-alkuinen
$ !?dippu?              komennossa ??-merkkien välissä oleva osa
$ !!:gs/dippu/puppu/    korvaus ennen suoritusta
$ ^dippu^zippu          korvaus ennen suoritusta
$ echo !2:2             komennon 2 toinen kntosanaa seuraava sana
$ echo !2:^-$           komennon 2 kaikki kntosanaa seuraavat osat
```


Alias-mekanismi

- Alias-määrittely aiheuttaa komennossa syötetyn osan korvaamisen jollakin toisella ennen komennon suoritusta
  - korvaamisen voi kieltää \ -merkillä
- Tehdyt määritykset voi tarkistaa komennolla alias ja niitä voi purkaa komennolla unalias
- Alias-määritykset eivät periydy aliprosesseille
  - ne ovat prosessin (ko. komentotulkin) paikallisia määrityksiä, eivät ole voimassa uusissa käynnistetyissä ohjelmissa

```
$ alias cp='cp -i'          kysele ylikirjoittamisesta (-i interactive)
$ alias mv='mv -i'
$ alias logout=exit         nyt heittomerkit eivät pakollisia
$ alias lo=logout
$ unalias cp
$ unalias -a                poista kaikki
```

- = -merkin kummallakaan puolella ei saa olla välilyöntiä

- Kun korvaava osa on yksinkertaisissa lainausmerkeissä ', komentotulkki tulkitsee sen riviä evaluoidessaan yhdeksi kokonaisuudeksi
- Komentoriviargumentit liitetään automaattisesti aliaksen perään
  - jos ne on tarve sijoittaa johonkin muuhun kohtaan käskyä, käytä aliaksen sijasta funktiota

```
$ alias lls='ls -l $* | less'
$
$ lls dir1 dir2
```

$ tarkoittaa kaikkia annettuja argumentteja*

argumentit tähän

- ei siis onnistu tyyliin
  - `alias mun_grep='grep $* tdstot'`

Tiedostonimien jokerimerkit

Jokerimerkkien avulla voi viitata useisiin tdstonimiin kerralla

- Kysymysmerkki ? korvaa minkä tahansa yhden merkin
- Tähtimerkki * korvaa minkä tahansa merkkijonon.
- Hakasulut [...] ilmaisee vaihtoehtoiset yksittäiset merkit.

  `$ lpr osa[0-9] osa[12][0-9] osa3[0-5]`

  Voi antaa muodossa [!abc], jolloin kelpuuttaa mitkä tahansa muut kuin luettelut merkit
- Aaltosuluilla {...} merkitään pitempi vaihteleva osa

  `$ cat ../{memo,pr*}.c`
- Tilde ~ on viite käyttäjän omaan kotihakemistoon, `~kjätunnus` on viite toisen käyttäjän kotihakemistoon

Komentorivin evaluointi

Esimerkki

```
$ grep -i matti *.txt | sort -n > tulos.txt
```

- shell
  - evaluoi (jäsentelee) rivin ennenkuin käynnistää yhtäkään ohjelmaa
  - täydentää komentoriviä aakkostetuilla tiedostonimillä, saadaan esim.

```
grep -i matti aa.txt ee.txt tt.txt | sort -n > tulos.txt
```

  - etsii ohjelmat hakupolulta
  - luo putken grep-komennon tulosteista sort-komennon syötteeksi, sekä luo sort-komennon tulosteiden uudelleenohjauksen tulostiedostoon
  - käynnistää ohjelmat erillisiksi (ali)prosesseiksi
    - i.a. uudelleenohjaus tapahtuu prosessien käynnistämisen yhteydessä

Komentorivin evaluointijärjestys

Ennen komentorivin suoritusta komentotulkki

a) korvaa history-viittaukset
b) tulkitsee erottimet \b, \t, \n jne. ja jakaa rivin sanoihin, myös erikoismerkit &, |, ;, >, <, (,), &&, ||, >> ja << ovat sanoja
   - ellei niiden merkitystä kumota lainausmerkein
c) päivittää history-listaa
d) muodostaa käskykokonaisuuksia
   - tutkii lainausmerkkiparit 'ja "
   - korvaa aliakset (ja niiden history-viitteet / aliakset)
   - korvaa muuttujat ja parametrit niiden arvoilla
   - korvaa `...`-merkkien väliset komennot niiden tulosteilla
   - laajentaa jokerimerkit aakkostetuksi tiedostonimien listaksi
   - selvittää uudelleenohjaukset, putket ja taustaprosessit

Ja vasta sitten se alkaa etsimään komentoa / ohjelmaa $PATH-muuttujassa kerrotuista hakemistoista

- Evaluointia voi rajoittaa merkkipareilla '...' "..." ja `...`
  - '...' *erikoismerkkejä ei tulkita UNIXin tyyliin (vahvin)*
  - "..." *vain ` ja \ ja $ tulkitaan UNIXin tyyliin*
  - `...` *vain \ tulkitaan UNIXin tyyliin*

```
$ echo "Tähtiä kuin otavassa ($lkm kpl): *******"
Tähtiä kuin otavassa (7 kpl): *******
$ echo 'Kirosana, kirosana $*#!!'
Kirosana, kirosana $*#!!
```

- Evaluointia ja suoritusta voi seurata asettamalla lipukkeita
  - $ set -o noexec tai set -n *lue komento, älä suorita*
  - $ set -o verbose tai set -v *tulosta käsitellyt rivit*
  - $ set -o xtrace tai set -x *tulosta rivi evaluoinnin jälkeen*

Voit käyttää näitä skriptiesi debuggauksessa!

- Kaikki lipukkeet (~on/off -muuttujat) saa näkyviin komennolla
  $ set -o
- Lipukkeen saa pois päältä komennolla
  $ set +o lipuke

Muuttujat

Muuttujat

- Tunnus alkaa kirjaimella, muut merkit numeroita, kirjaimia tai _
- Tulevat esitellyksi, kun niille annetaan arvo sijoituksella muuttuja=arvo tai set-lauseella (vain lipukemuuttujat)

```
$ nimi=averell                  Huom: ei välilyöntejä
$ kokonimi="Averell Dalton"
$ set -f                        kuten -o noglob
```

- Shell tunnistaa viittauksen tunnuksen edessä olevasta $-merkistä

```
$ prefix=poika
$ echo Terve mieheen, $prefix{mies}
Terve mieheen, poikamies
```

```
$ echo kokonimi
kokonimi
$ echo $kokonimi
Averell Dalton
```

- Muuttujia on kahdenlaisia
  - prosessin paikalliset muuttujat, ns. shell-muuttujat
  - aliprosesseille periytyvät globaalit muuttujat, ns. ympäristömuuttujat (engl. environment variables)

Muuttujat

- Bashissa voi käyttää muuttujien asettamiseksi käskyjä
  - export [-nf] [muuttuja[=arvo]] *ympäristömuuttuja*
  - readonly [-f] [muuttuja ... muuttuja] *arvoa ei voi muuttaa, vakio*
  - local [muuttuja[=arvo]] *funktion paikallinen muuttuja*
  - declare [-frxi] [muuttuja[=arvo]]
  - typeset [-frxi] [muuttuja[=arvo]]
- Valitsimet:
  - -n = no, -f = funktio, -r = readonly, -x = export, ja -i = integer
- Ilman argumentteja komennot näyttävät muuttujien arvot
- Kaikkien muuttujien arvot voi listata komennolla set
- Muuttuja tai funktio määritellään globaaliksi (ympäristömuuttujaksi)
  - komennolla export (~viedä ulos) tai
  - komentojen declare ja typeset valitsimella -x
- Kun shell tyypillisesti luo uuden prosessin suorittamaan komentoa, ovat ympäristömuuttujat käytettävissä aliprosessissa

Muuttujat

- Myös tavalliset sovellusohjelmat voivat käyttää ja muuttaa ympäristömuuttujia (mutta ei shell-muuttujia)
- Ympäristömuuttujat on tapana kirjoittaa isoilla kirjaimilla

```
$ export PAGER=more
$ PRINTER=U325ps ; export PRINTER
$ function hei() { echo Heipä hei ; }
$ export -f hei
$ set -o allexport              kaikista seur. tulee ympäristömuuttujia
```

- Muuttujan tai funktion voi palauttaa lokaaliksi export-komennon tarkentimella –n
- Muuttujan tai funktion voi poistaa komennolla
  - $ unset [-f] [muuttuja ... muuttuja]

Ympäristömuuttujia

- Globaaleja järjestelmän valmiiksi asettamia ympäristömuuttujia
  - näkyvät kaikille luotaville prosesseille
  - myös tavalliset sovellukset voivat käyttää näitä

$HOST	*koneen tunnus*
$OS	*(operating system) käyttöjärjestelmän tunniste*
$USER	*käyttäjätunnus*
$HOME, ~	*kotihakemiston absoluuttinen polku*
$PWD	*nykyinen työhakemisto*
$PATH	*komentojen ja ohjelmien hakupolku*
$CDPATH	*cd-komennon hakupolku*
$EDITOR	*tekstintoimitin*
$PAGER	*tekstin sivuttaja*
$PRINTER	*oletuskirjoitin*

man 5 environ
man bash

$SHELL	*käytössä oleva komentotulkki*
$SHLVL	*(shell level) kasvaa yhdellä, kun aliprosessi käynnistyy*
$IGNOREEOF	*shellistä ei voi poistua painamalla ctrl-d*
$PS1	*normaali kehote, oletus bashissa $*
$PS2	*toissijainen kehote, oletus >*
$PS3	*select-lauseen käyttämä kehote, oletus +*
$PS4	*jäljityksessä käytettävä kehote (kun asetettu set -o xtrace)*
$REPLY	*read-lauseella luettu kokonainen rivi*
$IFS	*(internal-field separator) kenttien välinen erotin oletus: välilyönti, tab, enter*
$*,$@	*komentorivillä välitetyt argumentit (kaikki)*
$0, $1,$2,..	*komentosana $0 ja muut argumentit $1, $2 jne*
$#	*komentoriviargumenttien lukumäärä*
$$	*komentoa suorittavan prosessin tunnus (pid)*
$!	*viimeeksi suoritetun taustaprosessin tunnus*
$?	*viimeeksi suoritetun komennon paluukoodi*

Shell-muuttujia

- Paikallisia (lipuke)muuttujia, jotka eivät periydy aliprosessille

$noclobber	*estä tulostusvirran ohjaaminen vanhan tdston päälle voi pakottaa: >|, >&| ja >>|*
$noglob	*estä shelliä evaluoimasta merkkejä * ? [] {} ja ~*
$nolinks	*cd komento ei siirry linkkiä pitkin uuteen hakemistoon (set -o physical)*
$notify	*töiden tilamuutoksista ilmoitetaan heti*
$verbose	*näytä suoritettava rivi history-korvausten jälkeen*
$xtrace	*näytä PS4 ja komentorivi evaluoinnin jälkeen*

man bash

Komentotiedostot eli Skriptit

Komentotiedostot

- Komentosarjat voi koota komentotiedostoiksi, skripteiksi
- Komentotdston luominen, suorittaminen ja suorituksen seuraaminen helppoa ja nopeaa
  - ei käännöksiä
  - olemassaolevien komentojen / ohjelmien hyväksikäyttö
  - nopeat prototyypit tai uudet palvelut
- Kerää omat komentotiedostosi hakemistoon ~/bin
  - luo ko. hakemisto ja anna siihen x-oikeus
  - usein jo valmiiksi mukana komentojen hakupolussa (echo $PATH)
  - jos ei ole, täydennä hakupolkua ~/.bash_profile -tiedostossa

```
PATH=$PATH:$HOME/bin
```

Komentotiedoston käynnistys

1) Suoraan nimellä, jos tdstoon on suoritusoikeus(x).

$ chmod u+x tdsto
$ tdsto

Jos hakemisto, jossa skripti majailee ei ole komentojen hakupolussa, annettava muodossa ./tdsto

2) Antamalla se syötteeksi sh-komennolle

Linuxissa sh on symbolinen linkki bash-tulkiin

$ bash tdsto

- Komentotiedostot suoritetaan oletusarvoisesti sh-tulkilla
- Jos käytät jotain muuta kuin bash-tulkkia (erilainen syntaksi!), merkitse se kommentin muodossa komentotiedoston ensimmäiselle riville, esim.
  #!/usr/bin/csh

Aloittaa kommentin

Syöttö ja tulostus

Useista koneista löytyy myös printf

- Tulostus komennolla
  echo [-n] *Ei vaihda tulostusriviä, jos valitsin -n*
- Lukeminen komennolla
  read [-p kehote] [muuttuja ... muuttuja] *-p= prompt*

```
echo -n "Kerro etunimesi: "
read etunimi
read -p "Kerro sukunimesi: " sukunimi
```

- Syötettäessä arvot erotetaan toisistaan muuttujassa IFS kerrotuin merkein
  - oletus: välilyönti, tab tai enter
- Jos välilyönti osa syötettä, käytettävä lainausmerkkejä '...' tai "..."
- Jos muuttujantunnuksia ei anneta erikseen, koko rivi menee muuttujan REPLY arvoksi

Estävät shell-evaluoinnin

- Syötteet voi antaa myös osana komentotiedostoa käyttämällä ns. here document ”-merkintää: **<<loppumerkki**

```
$ cat puh
echo nimi               puh    huone
grep "$*" << Loppu
Aaltonen Anna           4111    B444
Ahonen Anssi            4010    A301
...
Olematon Oskari         4242    C464
Loppu
```

```
$ puh Oskari
nimi            puh    huone
Olematon Oskari         4242    C464
```


Komentoriviargumentit

```
# args-esim arg1 arg2 ... argn
echo Komentosana: $0
echo Argumentit $1 $2 $3 $4 $5 $6 $7 $8 $9
echo "Kaikki (voi olla yli 10):" $*
shift
echo Kaksi ensimmäistä shiftin jälkeen: $1 $2
echo ja kaikki: $*
shift
echo Seuraavan shiftin jälkeen: $1 $2
echo ja kaikki: $*
```

```
$ args-esim eka toka kolmas neljäs
Komentosana: args-esim
Argumentit eka toka kolmas neljäs
Kaikki (voi olla yli 10): eka toka kolmas neljäs
Kaksi ensimmäistä shiftin jälkeen: toka kolmas
ja kaikki: toka kolmas neljäs
Seuraavan shiftin jälkeen: kolmas neljäs
ja kaikki: kolmas neljäs
```


Kontrollirakenteet

- Komentotiedostoissa käytettävissä normaalit ohjelmoinnin peruselementit
  - muuttujat
  - valinta
  - toisto
  - funktiot ja parametrit
  - syöttö ja tulostus
- Shell tarkka syntaksin suhteen, joten *kirjoitusasu huolellisesti* esim.
  - avainsanaa edeltävän komennon lopussa oltava ; ellei avainsanaa laita omalle rivilleen
  - testilausekkeen hakasulkujen kahtapuolen välilyönti
  - sijoitusmerkin = ympärillä ei tyhjämerkkejä

Ehdollinen lause

if testilauseke ; then komennot ; fi
if testilauseke ; then komennot ; else komennot ; fi
if testilauseke ; then komennot ;
elif testilauseke ; then komennot ;
else komennot ; fi

; ei pakollinen tässä

- Jos avainsana aloittaa oman rivin, sen eteen tai edeltävän rivin loppuun ei tarvitse kirjoittaa puolipistettä

Testilauseke

- Ehdollisessa lauseessa ja toistolauseissa tarvittava testi muotoa
  test lauseke tai
  [lauseke] *Huom: välilyönnit hakasulkujen ympärillä!*
- Palauttaa joko arvon 0 (=TRUE) tai 1 (=FALSE)
- Lausekkeessa voi verrata merkkijonoja tai numeerisia arvoja
- Myös komennot palauttavat arvonaan tiedon onnistumisesta, ja niitä voi käyttää testilausekkeena
- Peräkkäisten komentojen paluuarvo on oikeanpuoleisimman komennon paluuarvo ($?)
  grep kalle puh | wc -l

Tiedostonimille ks. man test, help test

[-d tunnus]	onko hakemiston nimi (directory)
[-f tunnus]	onko tiedoston nimi (file)
[-e tunnus]	onko olemassa (exist)
[-r tunnus]	onko lukuoikeus (read)
[-w tunnus]	onko kirjoitusoikeus (write)
[-x tunnus]	onko suoritusoikeus (execute)
[tunnus1 -nt tunnus2]	onko uudempi (newer than)
[tunnus1 -ot tunnus2]	onko vanhempi (older than)

jne.

Merkkijonoille

[-z merkkijono]	onko pituus 0 (ts. arvoa ei asetettu)
[-n merkkijono]	onko pituus > 0
[merkkijono]	onko pituus > 0
[mjono1 = mjono2]	ovatko merkkijonot samoja

Aritmeettisille lausekkeille

< , = , > ovat varattuja!

[lauseke1 -eq lauseke2]	samat (equal)
[lauseke1 -ne lauseke2]	erisuuret (not equal)
[lauseke1 -lt lauseke2]	pienempi kuin (less than)
[lauseke1 -le lauseke2]	pienempi tai yhtäsuuri (less or equal)
[lauseke1 -gt lauseke2]	suurempi kuin (greater than)
[lauseke1 -ge lauseke2]	suurempi tai yhtäsuuri (greater or equal)

- Lausekkeena voi olla myös -l merkkijono, jolloin verrataan merkkijonojen pituuksia

Loogiset lausekkeet

[! lauseke]	boolean NOT
[lauseke -a lauseke]	boolean AND
[lauseke -o lauseke]	boolean OR

Esim: read-lause, if-lause

```
# match
# check if arg1 arg2 arg3 are same

echo -n "word 1: " ; read arg1
read -p "word 2: " arg2
read -p "word 3: " arg3

if [ $arg1 = $arg2 -a $arg1 = $arg3 ] ; then
   echo "Match: words 1, 2 & 3"
elif [ $arg1 = $arg2 ] ; then
   echo "Match: words 1 & 2"
elif [ $arg1 = $arg3 ] ; then
   echo "Match: words 1 & 3"
elif [ $arg2 = $arg3 ] ; then
   echo "Match: words 2 & 3"
else
   echo Did not match
fi
```

```
$ match
word 1: saippuakauppias
word 2: kusti
word 3: saippuakauppias
Match: words 1 & 3
```

```
$ match
word 1: kuka on kusti
word 2: kuka on kusti
word 3: kuka on kusti
match: line 7: [: too many arguments
match: line 9: [: too many arguments
match: line 11: [: too many arguments
match: line 13: [: too many arguments
No match
```


Aritmetiikka

- Muuttujat oletusarvoisesti merkkijonoja
  ```
  $ a=5+5 ; echo $a
  5+5
  ```
- Muuttujasta saa kokonaislukumuuttujan declare-lauseella
  ```
  $ declare -i a
  $ a=5+5 ; echo $a
  10
  ```
  Käytä tätä tapaa
- Aritmeettisessa lausekkeessa muuttujaan voi viitata ilman $-merkkiä
  ```
  $ declare -i z=5
  $ z=$z+5
  $ z=z+5
  ```
- Aritmeettisten lausekkeiden evaluointiin voi käyttää myös shellin sisäistä komentoa let tai aritmeettista lavennusta $[ ]
  ```
  $ let d=5+5 b="21 * 2"
  $ echo $d " " $b
  10 42
  $ x=$[10/3]; echo $x
  3
  ```
  ei välejä =:n ympärille

  blanko, joten lainausmerkit

  vain kokonaislukuaritmetiikka

Aritmeettisia lausekkeita voi käsitellä myös expr-komennolla. Vaatii kyllä totuttelua.

Valintalause

```
case alkio in
    sään. lauseke )    komennot        voi olla "jokeri"lauseke
    ;;
    sään. lauseke )    komennot
    ;;
    sään. lauseke )    komennot
    ;;
 esac
```

- Kontrolli ei valu seuraavaan kohtaan kuten C:ssä ja Javassa
  - Ei tarvita break-lausetta
- Säännöllinen lauseke kuten grep-komennossa

Esim: case-lause

```
# case-testi                Huom: test on varattu sana!
read -p "Anna A, B tai C: " letter

case "$letter" in
  a|A)
  echo Annoit A
  ;;
  b|B)
  echo Annoit B
  ;;
  c|C)
  echo Annoit C
  ;;
  *)
  echo Et antanut A:ta, B:tä tai C:tä
esac
```

Nyt muuttujassa voi olla myös blankkoja

```
$ case-testi
Anna A, B tai C: B
Annoit B
$ case-testi
Anna A, B tai C: x
Et antanut A:ta, B:tä tai C:tä
```


Toisto

```
while testilauseke ; do        jatkumisehto: niin kauan kuin ehto totta
   komennot
done

until testilauseke ; do        lopetusehto: kunnes ehto tulee todeksi
   komennot
done

for indeksimuuttuja ; do       käy läpi kaikki komentoriviargumentit
   komennot
done

for indeksimuuttuja in argumenttilista ; do    käy läpi annetun listan
   komennot
done
```

peräkkäiset numerot voi tuottaa ohjelmalla seq (ks. man seq)

```
$ seq 1 9
1 2 3 4 5 6 7 8 9
```


Esim: while-lause

```
# fac n

if [ $# -ne 1 ] ; then
   echo Usage: fac n
   exit
fi

declare -i index=1      # integer variable
declare -i result=1

while [ $index -le $1 ] ; do
   result=result*index
   index=index+1
done
echo "$1! = " $result
```

Käytäntö: Tarkasta oikea käyttötapa heti!

```
$ fac
Usage: fac n
$ fac 5
5! =  120
```


Esim: until-lause

```
# secret
secretname="averell hakka"
name=noname
echo Try to guess the secret name
echo
until [ "$name" = "$secretname" ] ; do
   read -p "Give your guess : " name
done
echo Very good
```

muuttuja voi sisältää välilyöntejäkin!

```
$ secret
Try to guess the secret name

Give your guess : pekka
Give your guess : kusti
Give your guess : averell hakka
Very good
```

Esim: for-lause

```
# elukoi
for i ; do
   echo $i
done
muu="muutama muu muuli"
for elukka in lehmä sika sonni "$muu" ; do
   echo $elukka
done
```

```
$ elukoi aasi apina "kaksi anacondaa"
aasi
apina
kaksi anacondaa
lehmä
sika
sonni
muutama muu muuli
```


Esim: for-lause

- Nykyisin bash tunnistaa myös C-kielestä tutun for-lauseen kontrolliosan

```
for (( alkutoimet; ehto; lopputoimet )) ; do
   komennot
done
```

huomaa kahdet sulut

```
# toista
for (( c=1; c<=5; c++ ))
do
   echo "Toistokierros $c ..."
done
```

```
$ toista
Toistokierros 1 ...
Toistokierros 2 ...
Toistokierros 3 ...
Tositokierros 4 ...
Toistokierros 5 ...
```



```
select muuttuja ; do
   komennot
done
```

käy läpi kaikki komentoriviargumentit

```
select muuttuja in argumenttilista ; do
   komennot
done
```

käy läpi argumenttilistan

- select-lause yhdistää toiston ja valinnan
- tuottaa automaattisesti komentoriviargumenteista tai argumenttilistasta valintalistan, josta käyttäjä voi valita arvon muuttujalle
- toiston voi katkaista painamalla Ctlr-D, jolloin jatkaa seuraavasta lauseesta
- voi katkaista myös eksplisiittisesti break-lauseella
  - Suositus: lisää argumenttilistaan valinta "lopetus", ja sillä sitten break

Esim: select-lause

```
$ cat valitse
PS3="Valitse hedelmä [1-3]: "
select hede in omena appelsiini mandariini ; do
   echo "Valintasi on $hede"
   break
done
```

Selectissä käytössä Promp String 3

Uusi toistokierros, ellei katkaista breakillä

```
$ valitse
1 omena
2 appelsiini
3 mandariini
Valitse hedelmä [1-3]: 3
Valintasi on mandariini
```


Kontrollin siirto

`break` [n]	*kontrolli silmukan ohi seuraavan lauseeseen ohittaa n done sanaa*
`continue` [n]	*kontrolli takaisin silmukan alkuun palaa sisäkkäisissä silmukoissa n-tasoa*
`return` [n]	*paluu funktiosta, paluuarvon välitys*
`exit` tai `exit`(status)	*lopeta suoritus, paluu kutsujaan paluuarvo 0 = kaikki OK*
`exec` tdsto	*kontrolli toiseen komentotiedostoon, ei palaa enää tähän skriptiin*
`source` tdsto . tdsto	*kontrolli toiseen komentotiedostoon, palaa takaisin (vrt. funktion kutsu)*

Ryhmittely

(komennot) *suorita aliprosessina, muuttujien ja ympäristömuuttujien muutokset eivät voimassa prosessin päättyessä*

{ komennot ; } *koottu lohko, suoritus nyky-ympäristössä*

tarvitaan, ellei loppusulku omalla rivillään

Ryhmittely: Funktiot

```
tunnus () {
   komennot
}
```

oman funktion määrittely

saa puuttua tästä

```
function tunnus () {
   komennot
}
```

oman funktion määrittely

- Funktiota kutsuttaessa ei käytetä sulkuja
- Kutsussa voi antaa positionaalisia parametreja funktion tunnuksen perässä
  - vrt. komentoriviargumentit
  - niihin viitataan funktion lohkossa tunnuksilla $1, $2, jne..
  - paluun jälkeen $1, $2, jne. ennalleen (= alkup. komentoriviargumentit)
- Funktiossa voi esitellä paikallisia muuttujia määreellä local
  - ne ovat käytettävissä vain ko. funktiossa
  - eivät voi sotkeentua globaaleihin muuttujiin

Esim: funktiot

```
# funktiot whoson ja go

whoson ()
{
   date
   echo Users Currently Logged In
   who
}

function go () { cd $HOME/$1 ; PS1="[`pwd`]: " ; }

Kutsuesimerkki
$ go public_html
[/home/hakka/public_html]:
```

Prompt String 1

positionaalinen parametri

kehoite nyt tällainen

Keskeytysten (signaalien) käsittely

- Komentotdston suoritus päättyy, kun
  - tdsto loppuu
  - kohdataan komento exit tai exit(status)
  - käyttäjä painaa ctrl-c
  - ko. prosessi saa kill käskyssä välitetyn signaalin
- Signaalit kertovat poikkeustilanteista, kuten epäonnistuneesta systeemikutsusta, isäprosessin päättymisestä, jne.
- Signaalinumerot saa listattua komentamalla kill -l
- Signaaleihin voi varautua komennolla
  `trap` [komennot] signaalinumero

Signaalinumeroita

1	SIGHUP	katkeaa jostain epätavallisesta syystä (line hangup)
2	SIGINT	painettu ctrl-C (keskeytys)
3	SIGQUIT	painettu quit-näppäintä (ctrl-| tai ctrl-\)
9	SIGKILL	annettu komento kill –KILL (ei voi kieltäytyä)
15	SIGTERM	annettu komento kill (ohjelmallinen lopetus)
20	SIGSTOP	painettu ctrl-z

0 suorita komennot aina, kun komentotdston suoritus loppuu (normaalisti tai exitillä)

Esim: keskeytyksen käsittely

Komentotiedoston paluuarvo

```
# loop
trap 'echo You pressed ctrl-C; exit 1' SIGINT
while true
do
   echo -n "Looping..."
done
```

```
$ loop
Looping...Looping...Looping...Looping...Looping
...Looping...Looping...Looping...Looping...Loop
ing...Looping...Looping...Looping...Looping...L
ooping...Looping...Looping...Looping...Looping.
..Looping...Looping...Looping...Looping...Loopi
ng...Looping...Looping...Looping...Looping...Lo
oping...Looping...You pressed ctrl-C
```



```
# addbanner
script=`basename $0`                          ks. man basename, man dirname
if [ ! -r $HOME/banner ] ; then
 echo $script : need readable $HOME/banner file
 exit 1
fi          prosessinumero                    Poista aina tarpeettomat aputiedostot!
temp=/tmp/$$.$script
echo Temp file name is $$.$script
trap 'exit 1' 1 2 3 15
trap 'rm $temp 2> /dev/null ; echo Temp file removed' 0

for file ; do
  if [ -r $file -a -w $file ] ; then
    cat $HOME/banner $file > $temp
    cp $temp $file
    echo $script : banner added to $file
  else
    echo $script : need r and w permissions \
         for $file                            \ poistaa rivivaihdon merkityksen
  fi
done
```


Taulukot

Taulukot

~ versiosta 2.1 alkaen

- Uudet bash-versiot sallivat myös yksiulotteiset taulukot
- Ei tarvitse esitellä etukäteen, ensimmäinen sijoitus luo taulukon
  `taulu[3]=5`
- Taulukon voi esitellä etukäteenkin, kokoa ei tarvitse antaa
  `declare -a taulu`
- Taulukkoon sijoittaminen indeksoimalla
- Taulukkoon saa jäädä "aukkoja", "puuttuvia lokeroja"
  `taulu[5]=5`
  `taulu[2]=2`
- Alkioon viitattaessa käytettävä aaltosulkuja
  `echo ${taulu[5]}`
- Alkiot ovat merkkijonoja (kuten tavalliset muuttujat)
  `taulu[3]=kolome`
  - ohjelmoija vastaa, että aritmetiikkaa vain numeromerkkijonoilla

- Usealle taulukon alkiolle voi antaa arvot kootusti
  `declare -a taulu=(eka kaksi kolome)` *Pienin indeksi 0*
- Voi määrätä myös mille indeksille sijoitus menee
  `taulu=([1]=eka kaksi [15]=viistöistä)`
- Read-lauseella voi lukea kerralla taulukollisen arvoja
  `read -a taulu`
- Kaikkiin taulukon alkioihin voi viitata yhdellä kertaa merkinnällä
  `${taulu[@]}` tai `${taulu[*]}`
- Kaikki käytössä olevat indeksinumerot saa listaksi merkinnällä
  `${!taulu[@]}`
- Taulukon alkioiden lukumäärän saa merkinnällä
  `${#taulu[@]}`tai `${#taulu[*]}` *{ } –sulut estää tiedostonimeksi evaluoinnin*
- @: alkiot evaloituvat aina kukin omaksi merkkijonokseen
- *: yleensä kuten @. Jos viittaus lainausmerkeissä "`${taulu[#]}`", alkiot evaloituvat yhdeksi merkkijonoksi

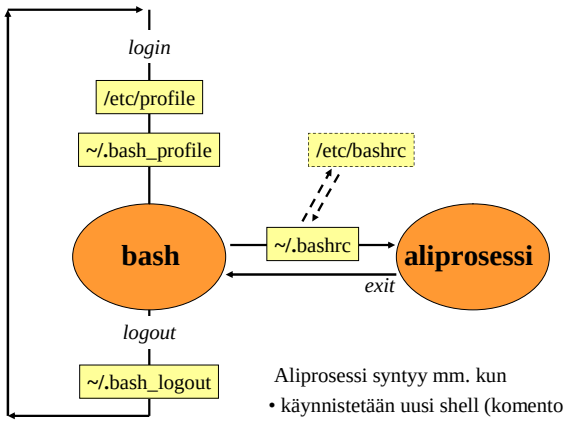

```
# colors
declare -a colors

# Lue yhdellä read-lauseella koko taulukollinen alkioita
echo "Luettele lempivärisi (erottele blankoilla):"
read -a colors

# Tulosta taulukon alkiot, kukin omalle rivilleen
count=${#colors[*]}
index=0
while [ $index -lt $count ] ; do
  echo ${colors[$index]}
  let "index = $index + 1"
done

# Tulosta koko taulukko kerralla
echo ${colors[*]}
```

```
$ colors
Luettele lempivärisi (erottele blankoilla):
sininen punainen oranssi vihreä
sininen
punainen
oranssi
vihreä
sininen punainen oranssi vihreä
```

- Alkion poisto
  `unset $taulu[15]`
- Koko taulukon poisto
  `unset taulu`

Automaattisesti suoritettavat komentotiedostot

Autom. suoritettavat komentotiedostot



Aliprosessi syntyy mm. kun
- käynnistetään uusi shell (komento bash)
- käynnistetään joku sovellus

/etc/profile

- suoritetaan istunnon alkaessa
- kaikille käyttäjille tarkoitetut yhteiset alkutoimet
- pääteasetukset, käyttöympäristön asetuksia
- ympäristömuuttujien oletusarvoja (mm. hakupolut)
- luotavien tiedostojen käyttöoikeudet (umask)
-

~/.bash_profile

- suoritetaan kerran istunnon alkaessa (/etc/profile:n jälkeen)
- henkilökohtaisten alkurutiinien suoritukset
- ympäristömuuttujien asetuksia (myös oletusarvot voi korvata)
  - omat globaalit käyttöympäristön asetukset
  - esim. hakupolun täydennys, oletuskirjoitin

. -alkuiset tdstonimet saa näkyviin komentamalla ls -a

~/.bashrc

rc = run command

- suoritetaan aina uuden prosessin käynnistyessä, ei kuitenkaan automaattisesti istunnon alussa
- lokaalit shell-muuttujien asetukset
- omien funktioiden ja aliasten määrittely

~/.bash_logout

- suoritetaan kerran istunnon päätteeksi
- lopputoimet, muistutukset

~/.pinerc, ~/.emacs, ~/.newsrc jne.

- suoritetaan ko. sovellusta käynnistettäessä

Esim: /etc/profile

```
# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

quota -q 2>&1 | grep -v 193.167.197.17

# Path manipulation
pathmunge () {
   if ! echo $PATH | /bin/egrep -q "(^|:)$1($|:)" ; then
      if [ "$2" = "after" ] ; then
         PATH=$PATH:$1
      else
         PATH=$1:$PATH
      fi
   fi
}
if [ `id -u` = 0 ]; then
   pathmunge /sbin
   pathmunge /usr/sbin
   pathmunge /usr/local/sbin
fi
pathmunge /usr/X11R6/bin after
unset pathmunge
```

Onko jo polussa:
Alussa, keskellä tai, lopussa

jatkuu...


```
#ulimit -S -c 0 > /dev/null 2>&1        # No core files by default

# Hard and soft max. limits for users
#ulimit -d 500000       # RAM: data kB
#ulimit -l 500000       # RAM: locked kB
#ulimit -m 500000       # RAM: memory kB
#ulimit -v 500000       # RAM: virtual kB
#ulimit -s 8192 # stack: kB
#ulimit -t 600          # cpu time in seconds per process
#ulimit -u 42           # processes
#ulimit -f 300000       # file size max, kB

USER="`id -un`"
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi
```

jatkuu...


```
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
```

suoritus tämän prosessin ympäristössä

- Tuossa mainitusta /etc/profile.d hakemistosta käynnistetään lisää komentotdstoja
- Sisältävät sovelluskohtaisia globaaleja aliaksia ja ympäristömuuttujien asetuksia

```
colorls.sh
glib2.sh
java.sh
krb5.sh
lam.sh
lang.sh
less.sh
ulimit.sh
vim.sh
which-2.sh
```


~/.bash_profile

```
#.bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
   . ~/.bashrc
fi

# User specific environment and startup programs goes below

export PATH=$PATH:$HOME/bin
unset USERNAME

export CDPATH=".:..:../.." # Directory search path for cd command

function ls()    { command ls --color=auto -F "$@" ; }
function psg()   { ps auxw | grep "$@" | grep -v grep ; }
function nkill() { kill -9 `psg "$@" | cut -c9-14` ; }
```

koska ei suoriteta oletuksena istunnon alussa

Ettei tulisi rekursioita!

Etsi prosessia nimellä
Tapa nimellä

~/.bashrc

```
# User specific aliases and functions, common to all interactive shells
# Source global definitions
if [ -f /etc/bashrc ]; then
   . /etc/bashrc
fi
```

```
# /etc/bashrc
# System wide functions and aliases.
# Environment stuff goes in /etc/profile

# We want this to get set, even for non-interactive, non-login shells.
if [ "`id -gn`" = "`id -un`" -a `id -u` -gt 99 ]; then
   umask 007
else
   umask 022
fi
```

nämähän eivät periydy aliprosesseille automaattisesti

jatkuu...


```
## Are we an interactive shell?
if [ "$PS1" ]; then
   if [ -x /usr/bin/tput ]; then
     if [ "x`tput kbs`" != "x" ]; then # Can't do this with "dumb" terminal
        stty erase `tput kbs`
     elif [ -x /usr/bin/wc ]; then
       if [ "`tput kbs|wc -c `" -gt 0 ]; # Can't do this "dumb" terminal
       then
         stty erase `tput kbs`
       fi
     fi
   fi
   case $TERM in
   xterm*)
    if [ -e /etc/sysconfig/bash-prompt-xterm ]; then
        PROMPT_COMMAND=/etc/sysconfig/bash-prompt-xterm
    else
        PROMPT_COMMAND='echo -ne
                  "\033]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}\007"'
        fi
        ;;
```

jatkuu...


```
   screen)
        if [ -e /etc/sysconfig/bash-prompt-screen ]; then
          PROMPT_COMMAND=/etc/sysconfig/bash-prompt-screen
        else
          PROMPT_COMMAND='echo -ne
             "\033_${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}\033\\"'
        fi
        ;;
   *)
        [ -e /etc/sysconfig/bash-prompt-default ] &&
     PROMPT_COMMAND=/etc/sysconfig/bash-prompt-default
        ;;
    esac

    # Turn on checkwinsize
    shopt -s checkwinsize
    [ "$PS1" = "\\s-\\v\\$ " ] && PS1="[\u@\h \W]\\$ "

    if [ "x$SHLVL" != "x1" ]; then      # We are not a login shell
       for i in /etc/profile.d/*.sh; do
         if [ -r "$i" ]; then
           . $i
         fi
       done
    fi
fi
```


~/.bash_logout

```
#~./.bash_logout

# Actions when logging out

clear
echo `whoami` logged out at `date`
if [ -r /etc/motd ] ; then cat /etc/motd
```


Kertauskysymyksiä

- Selitä mikä on komentotulkki?
- Miten shell-muuttujat ja ympäristömuuttujat eroavat toisistaan?
- Miten komentoriviargumentteihin viitataan?
- Miten funktiossa viitataan sen parametreihin?
- Mikä on $PATH-muuttuja?
- Mitä komentotiedostoja suoritetaan istunnon aluksi?
- Mitä komentotiedostoja suoritetaan aina ohjelmia (komentoja) käynnistettäessä?