Shellit ja komentojonot

2 Yleistä

2.1 Alkusanat

Luennoilla 3, 4, 5 ja 6 on tarkoitus käsitellä kolmea shelliä (Bourne shell, Korn shelliä ja C shelliä), komentojonojen kirjoittamista em. shelleissä sekä joitakin yleisiä Unix komentoja. Korn ja C shelliä käsittelevissä luvuissa esitellään vain sellaiset ominaisuudet, jotka eroavat Bourne shellin ominaisuuksista.

2.2 Historiaa

Kolme yleisintä UNIX systeemeissä käytettyä shelliä ovat: Bourne shell (sh) (AT&T shell), C shell (csh) (Berkeley shell) ja Korn shell (ksh) , joka on Bourne shellin laajennus. Interaktiivisessa kaytössä nämä kolme shelliä ovat pitkälti samankaltaisia. Syntaksiset erot tulevat vastaan kun kirjoitetaan komentojonoja.

Ensimmäinen merkittävä "standardi" Unix shell oli V7 Unixissa (AT&T V7) esitelty Bourne shell. Se on nimetty kehittäjänsä Stephen Bournen mukaan ja esiteltiin vuonna 1979.

Bourne shelliä käytetetään systeemien ylläpidossa, koska se löytyy lähes jokaisesta UNIX järjestelmästä ja lisäksi se on syntaksiltaan yksinkertainen. Bourne shellin oletuskehoite (prompti) on $ (dollari symboli).

C shell on kehitetty Berkeleyssä Kalifornian yliopistossa 1970 luvun lopussa ja se julkaistiin osana 2BSD UNIXia. C shellin on pääosin kirjoittanut Bill Joy ja siihen on Bourne shelliin verrattuna lisätty ominaisuuksia esimerkiksi komentorivihistoria, aliakset, sisäänrakennettu aritmetiikka, tiedostonimien täydennys sekä töidenhallinta. C shellin syntaksi perustuu C kielen syntaksiin ja siksi se on ohjelmoijien suosiossa. C shellin oletuskehoite on % (prosentti)

Korn shell on laajennus Bourne shellistä ja sen on kehittänyt David Korn AT&T:ssä 1980 luvun puolivälissä. Korn shell julkaistiin 1986 ja se tuli virallisesti osaksi SVR4 UNIXia vuonna 1988. Korn shell toimii UNIXin lisäksi myös muissa systeemeissä (esim. OS/2, VMS ja DOS).

```
sh = Bourne shell
ksh = Korn shell
csh = C shell
tcsh = TC shell
bash = Bourne again shell
zsh =  Z shell
rsh = Restricted shell tai remote shell
```

2.3 Tehtävät

Shell (=komentotulkki, kuori) on ohjelma (sisältäen myös "ohjelmointikielen") , joka toimii käyttäjän ja UNIXin kernelin välissä. Shellin tehtävänä on lukea käyttäjän antamia komentoja joko päätteeltä tai tiedostosta ja "tulkata" ne kernelin ymmärtämään muotoon.

Käyttäjän kirjautuessa UNIX koneeseen käynnistyy login-shell, joka jää odottamaan käyttäjän syötettä. Interaktiivisen shellin tehtävänä on (a) parsia komentorivi, (b) huolehtia jokerimerkkien tulkinnasta, uudelleensuuntauksesta jne. (c) etsiä komennot ja suorittaa löytyneet komennot.

2.4 Ominaisuuksia

- Sisäisten ja ulkoisten komentojen suoritus
- Komentojen ketjutus ja putkitus mahdollista
- Ympäristömuuttujat
- Syötteen ja tulostuksen uudelleensuuntaus mahdollista
- Jokerimerkkien käyttö mahdollista tiedostojen nimissä
- Komentojen tausta-ajo mahdollista
- Komentojonot (=shellskriptit)

3 Bourne shell

Käynistyessään Bourne shell etsii systeemitiedostoa `/etc/profile` ja suorittaa siellä olevat komennot. Seuraavaksi haetaan käyttäjän kotihakemistosta alustustiedostoa nimeltä `.profile` ja suoritetaan siellä olevat komennot.

`/etc/profile` ja `.profile` tiedostot suoritetaan vain kerran sisäänkirjoittautumisen yhteydessä.

3.1 Määritelmiä

Tyhjä (blank)
Tarkoittaa sarkainta (tab) tai välilyöntiä (space)

Metamerkki (metacharacter)
Jokin seuraavista merkeistä:
; & ( ) | < > \ $ * rivinvaihto välilyönti tabulaattori

Nimi (identifier)
Sarja kirjaimia, numeroita tai alaviivoja. Nimiä käytetään esimerkiksi funktioden tai parametrien nimeämiseen

Sana (word)
Merkkijono (ei tyhjä), joka voi sisältää lainausmerkittömän metamerkin.

Komento (command)
Shellin syntaksin mukainen jono merkkejä. Shell suorittaa komennon joko sisäisesti tai kutsumalla tarvittavia ohjelmia.

Sisäinen komento (special command)
Shell suorittaa sisäiset komennot käynnistämättä uutta prosessia.

#
Kommenttimerkki

Parametri (parameter)
Nimi, numero tai jokin seuraavista merkeistä !, $*$, @, #, ?, -, $ ja _ tai näiden yhdistelmä.

Palautus arvo (exit status)
Komennon lopetustila nolla mikäli komento päättyy normaalisti tai 128+virhetila jos komento päättyy epänormaalisti

Muuttuja (variable)
Parametri.

Ympäristömuuttuja (environment variable)
Parametri joka on voimassa lokaalin shellin ulkopuolella.

Bourne shellin ympäristömuuttujat:

PATH	Polku, josta komentoja haetaan.
HOME	Kotihakemisto.
IFS	Sisäinen kenttäerotin, normaalisti tyhjä, tabulaattori tai rivinvaihto.
LOGNAME	Käyttäjätunnus.
MAIL	Mikäli MAIL muuttujan arvo on mail-tiedoston nimi ja MAILPATH muuttujaa ei ole asetettu, shell ilmoittaa käyttäjälle mailin saapumisesta asetettuun tiedostoon.
MAILCHECK	Ilmoittaa kuinka usein shell tarkistaa mailin saapumisen. Oletusarvo 600 sekuntia.
MAILPATH	Kaksoispisteillä erotettu lista tiedostonimiä.
PWD	Nykyinen työhakemisto.
PS1	Ensisijainen kehote (prompt), oletuksena $ merkki.
PS2	Toissijainen kehote, oletuksena > merkki.
SHELL	Shellin polku. Shell Alustaa muuttujat PATH, PS1, PS2, MAILCHECK ja IFS. Login ohjelma alustaa muuttujat HOME ja MAIL

Putki (pipeline)
Sarja komentoja (yksi tai useampia) erotettuna '|' merkillä. Putkessa edellisen komennon tulosteet ohjataan seuraavan komennon syötteeksi. Jokainen komento suoritetaan omassa prosessissa, shell odottaa että edellisen komennon suoritus loppuu ennenkuin seuraava aloitetaan. Putken palautusarvo on viimeisen komennon palautusarvo.
Esim.

```
ls -la | more
```

Lista (list)
Lista on sarja yhdestä tai useammasta putkesta erotettuna jollain seuraavista merkeistä ; , &, &&, ||. Lista voidaan päättää joko ; tai & merkkiin.

; Merkki aiheuttaa putkien synkronisen suorituksen. ';' merkki voidaan korvata yhdella tai useammalla rivinvaihtomerkillä.

```
ls -1; pwd; date
```

& Merkki aiheuttaa putkien asynkronisen suorituksen.

```
komento1 & komento2
```

Tarkoittaa sitä, että `komento1` suoritetaan tausta-ajona.

&& Jäljessä oleva lista suoritetaan vain jos edeltävän putken suoritus palauttaa nollan.

```
cc prog1.c -o prog1 && prog1
```

|| Jäljessä oleva lista suoritetaan vain jos edeltävän putken suoritus palauttaa nollasta poikkeavan arvon.

```
cc prog1.c -o prog1 || prog2
```

3.2 Komentorivin parsinta ja komentojen suoritus

Shell lukee komentorivin ja parsii sen jakamalla syötteen sanoiksi, jotka ovat erotettu välilyönnillä tai tabulaattorilla. Komentorivi päättyy rivinvaihtoon. Shell tarkistaa onko ensimmäinen sana komento. Komento on joko sisäinen komento, funktio tai ulkoinen ohjelma. Funktiot sijaitsevat shellin muistissa. Sisäiset komennot ovat shellohjelman itsensä sisältämiä komentoja ja ne shell suorittaa käynnistämättä uutta shelliä. Muussa tapauksessa shell etsii komentoa PATH-muuttujassa määritellyistä hakemistoista. Jos komento löytyy shell käynnistää lapsishellin fork-systeemikutsulla ja suorittaa komennon. Shell itse odottaa kunnes ohjelman suoritus lakkaa ja tarvittaessa ilmoittaa suoritetun ohjelman statuksen.

3.3 Standardi laitteet

Unixssa prosessiin liitty kolme standardilaitetta:

Tiedostokuvaaja	**Laitenimi**	**Tarkoitus**
0	stdin	syöte
1	stdout	tulostus
2	stderr	virhe

3.4 Syötteen ja tulostuksen uudelleensuuntaus

Shellissä on mahdollista uudelleensuunnata komentojen tulostus tai vastaavasti komentojen syöte voidaan ottaa suoraan esim. tiedostosta. Uudelleensuuntauksessa käytetään seuraavia merkkejä: $<, <<, >, >>$

$<$ Määrää syötteen otettavaksi merkkiä seuraavasta tiedostosta.

```
read line < /etc/passwd
```

<< Määrää syötteen otettavaksi nykyisestä lähteestä kunnes vastaan tulee haluttu merkkijono
```
cat << LABEL
OHJE: ohjelma parametrit.
LABEL
```

> Suuntaa tulosteen merkin perässä olevaan tiedostoon. Mikäli tiedostoa ei ole se luodaan. Varoitus! Olemassa olevan tiedoston sisältö menetetään.
```
find / -name luento.tex > foundit
```

>> Suuntaa tulosteen annetun tiedoston loppuun. Mikäli tiedostoa ei ole olemassa se luodaan.
```
cat file1 >> file2
```

2 > Suuntaa virhetulostuksen haluttuun paikkaan.
```
grep test file 2> /dev/null
```

1 >&2 Suuntaa tulostuksen samaan paikkaan kuin virheen.
```
echo ‘‘komento tarvii agrumentteja’’ 1>&2
```

2 >&1 Suuntaa virheet samaan paikkaan kuin tulostuksen.
```
cc ohjelma.c > loki 2>&1
```

3.5 Paikkasidonnaiset parametrit (positional parameters)

Paikkasidonnaiset parametrit ovat erityisasemassa olevia shellin sisäisiä muuttujia, joita kaytetään erityisesti välitettäessä parametrejä komentojonoille. Nimensä mukaisesti namä paremetrit ovat riippuvaisia paikastaan komentorivillä. (Bourne) shell sallii yhdeksän samanaikaisesti käytössä olevaa parametriä. Paikkasidonnaiset parametrit voidaan asettaa tai tyhjentää `set` komennolla.

Paikkasidonnaiset parametrit

`$0` On nykyisen komentojonon nimi.
```
echo $0
```

`$1-$9` Parametrit 1 - 9.
```
set a s d f g h j
echo $1 $2 $3
```

`$#` Tarkoittaa parametrien lukumäärä.
```
echo $#
```

`$*` Tarkoittaa kaikkia parametrejä
```
echo $*
```

`$@` Sama kuin $*, paitsi lainausmerkeissä
```
echo $@
```

`“$*”` Tarkoittaa “$1 $2 $3 … $9”
```
set a s d f g h ‘‘w e’’
for i in ‘‘$*’’; do echo $i; done
```

`“$@”` Tarkoittaa “$1” “$2” “$3” … “$9”
```
for i in ‘‘$@’’; do echo $i; done
```

Muita erityisasemassa olevia muuttujia

`$$` Shellin prosessinumero (PID).

`$-` Nykyisen shellin optiot.

`$?` Viimeiseksi suoritetun komennon palautusarvo.

`$!` Viimeiseksi taustalle asetetun prosessin PID.

3.6 Merkkien suojaus (quoting)

Mikäli shellin erikoismerkkejä halutaan käyttää sellaisenaan on ne suojattava lainausmerkeillä. Suojaukseen on kolme tapaa: yksinkertaiset heittomerkit (`' '`), lainausmerkit (`" "`) sekä kenoviiva (\). Suojauksen vaativia erikoismerkkejä ovat: ; (komentoerotin), & (tausta-ajo), ( ) (komentojen ryhmitys), { } (komentojen ryhmitys), | (putki), < (syötteen uudelleensuuntaus), > (tulosteen uudelleensuuntaus), rivivaihto, väli/tabulaattori, $ (muuttujan korvausmerkki), * [ ] ? (tiedostonimikorvauksen erikoismerkit).

Kenoviivaa käytetään suojaamaan yksittäistä merkkiä. Kenoviiva on suojattu yksikertaisten heittomerkkien sisällä. Kenoviiva suojaa $, ‘, ja \ merkit lainausmerkkien sisällä.

Yksinkertaiset heittomerkit suojaavat kaikkia erikoismerkkejä. Mikäli haluaa tulostaa yksinkertaisen heittomerkin on se suojattava kenoviivalla tai lainausmerkillä. Heittomerkille pitää loytyä pari eli ne on suljettava.

Lainausmerkit sallivat komentosuorituksen ja muuttujien korvauksen, mutta suojaavat muut erikoismerkit. Lainausmerkille pitää loytyä pari eli ne on suljettava.

3.7 Korvaukset

3.7.1 Tiedostonimikorvaus

Kun shell suorittaa (evaluoi) komentoriviä se korvaa eräät erityismerkit siten, että käyttäjän syöte vastaa tiettyä joukkoa tiedostoja tai hakemistoja.

*	Tarkoittaa nollaa tai useampaa merkkiä. `ls -al a*`
?	Tarkoittaa tasan yhtä merkkiä. `ls -la abc?`
[abc]	Jokin merkeistä a, b tai c. `ls -al [ah]*`
[a-z]	Jokin merkki väliltä a ... z. `ls -al [a-f]*`
[!a-z]	Jokin muu merkki kuin a ... z. `ls -al [!a-f]*`
\	Muuttaa erikoismerkin tavalliseksi.

3.7.2 Parametrien korvaus

$parametri

Korvataan parametrin arvolla.

```
LOG=/tmp/loki.txt; echo $LOG
```

${parametri}

Sama kuin edellä mutta kaarisulut pitää olla siksi, että parametrin perässä on kirjain, numero tai alaviiva, joka ei ole osana parametriä.

```
BLOG=${LOG}.bak; echo $BLOG;
```

${parametri:-sana}

Mikäli parametri on asetettu ja sisältää tekstiä, korvataan parametrin arvolla. Muuten korvataan sana.

```
echo $EDITOR
EDITOR=${EDITOR:-/usr/bin/emacs}; echo $EDITOR
EDITOR=/bin/vi
EDITOR=${EDITOR:-/usr/bin/emacs}; echo $EDITOR
```

${parametri:=sana}

Mikäli parametriä ei ole asetettu tai se on tyhjä asetetaan sen arvoksi sana ja sitten korvataan parametrin arvolla. Paikkasidonnaisia parametrejä ei voi asettaa tällä tavoin.

```
LOG=; echo ${LOG:=/tmp/tmp.log}
```

${parametri:?sana}

Mikäli parametri on asetettu ja on epätyhjä, korvataan sen arvolla. Muussa tapauksessa tulostetaan sana. Mikäli sana puuttuu tulostetaan standardiviesti (parameter null or not set)

```
LOG=file.txt;echo ${LOG:?''LOG not set''}
LOG=; ${LOG:?}
```

${parametri:+sana}

Mikäli parametri on asetettu ja on epätyhjä korvataan sana, muuten ei korvata mitään.

```
TEST=foo; echo ${TEST:+''Is set''}; echo $TEST
```

3.7.3 Komentojen suoritus

Komennon tulostusta voidaan käyttää osana sanaa laittamalla komento ‘ ‘-merkkien sisään.

```
LOGFILE=backup_‘date '+%d%a%m%y'‘.log
```

3.8 Bourne shellin sisäiset komennot

. file
Suorittaa komennot tiedostosta file ja palaa takaisin. Komennot suoritetaan nykyisessä shellissä. Käytetään esimerkiksi shellin alustustiedostojen lukemisessa. Palautusarvo on viimeisen komennon palautusarvo.

:
Tyhjä komento, palautusarvo nolla.

break
Poistuminen sisemmästä silmukasta.

continue
Hyppy sisemmän silmukan alkuun.

cd
Hakemiston vaihtaminen. Ilman parametreja vaihdetaan käyttäjän kotihakemistoon.

echo [**args**]
Tulostaa argumentit.

eval komento
Argumenttien evaluointi ja suoritus komentotulkissa. Shell tutkii komentorivin kahteen kertaan ennen suoritusta.

exec komento
Argumentin suoritus shellin tilalla. Shellistä poistutaan kun komennon suoritus päättyy. Mikäli kyseessä on login shell aiheuttaa uloskirjoittautumisen.

exit [**n**]
Poistutaan shellistä palautusarvolla n.

export [**var**]
Tekee muuttujan tunnetuksi myös alishelleille.

hash
Ylläpitää sisäistä taulukkoa (hash table) joka mahdollistaa komentojen nopeamman löytymisen.

kill [**-signal prosessi**]
Kill lähettää signaalin prosessille. Prosessi voidaan antaa joko PID numerona tai työnumerona (job).

getopts
Käytetään komentojonoissa komentorivioptioiden tarkistamiseen ja parsimiseen.

login [**user**]
Sisään kirjoittautuminen käyttäjätunnuksella user.

newgrp [**arg**]
Käyttäjäryhmän vaihto.

pwd
Tulostaa nykyisen työhakemiston.

read [**var**]
Lukee rivin syötteestä ja laittaa sen muuttujan var arvoksi.

readonly [**var**]
Asettaa muuttujan var vakioksi.

return [**n**]
Paluu funktiosta arvolla n.

set
Parametrien asettaminen.

shift [**n**]
Paikkasidonnaisten parametrien siirto.

times
Tulostaa nykyisen shellin prosessien kertyneet suoritusajat.

trap [**arg**] [**n**]
Kun shell vastaanottaa signaalin n suoritetaan arg.

type [**command**]
Tulostaa komennon tyypin.

ulimit
Asettaa ja tulostaa prosessien resurssirajoitukset.

umask
Tiedostojen oikeuksien luontimaski.

unset
Muuttujan tai funktion poisto.

wait [**pid#n**]
Odottaa taustaprosessin n valmistumista ja tulostaa paluuarvon.

3.9 Yhdistelmäkomennot (compound commands)

(lista)
Suluissa oleva lista suoritetaan erillisessä ympäristössä. Sulun ja listan välissä pitää olla tyhjä merkki.

```
( ls; pwd; date; ) > file
```

{ lista; }
Suoritetaan lista mutta ei erillisessä ympäristössä.

```
{ ls; pwd; date; }
```

case sana **in** [[;] [(] **hahmo** [| **hahmo**] ... **) lista ;;**] ... **; esac**
Tutkitaan vastaako jokin hahmo sanaa ja kun vastaavuus löytyy niin suoritetaan hahmoa vastaava lista. Hahmoon sovelletaan samoja periaatteita kuin tiedostonimikorvauksessa.

for parametri [**in** sana ...] **; do** lista **; done**
Asetetaan parametrin arvoksi sana ja suoritetaan lista. Mikäli sana puuttuu asetetaan parametrin arvoksi paikkasidonnaiset parametrit. Suoritus päättyy kun sana, lista tai paikkasidonnaiset parametrit loppuvat.

parametri () { lista **; }**
Määrittelee parametri-nimisen funktion. Funtiot toimivat komentojonojen tapaan eli niiden avulla voidaan suorittaa joukko komentoja kirjoittamalla vain yksi komento. Funktioiden käyttö on kuitenkin tehokkaampaa kuin komentojonojen, koska määrittelyn jälkeen funktio on osa shellin muistia.

if lista **; then** lista **;** [**elif** lista **; then** lista **;**] ... [**else** lista **;**] **fi**
Aluksi suoritetaan **if** lista ja mikäli sen palautusarvo on nolla suoritetaan ensimmäinen **then** lista. Muuten tutkitaan **elif**, jos sellainen löytyy. Mikäli mikään **elif** ei toteudu suoritetaan **else**, jos sellainen löytyy. Mikäli **else** osaa ei ole tai **then** osa on suoritettu **if**:n palautusarvo on nolla.

until lista **; do** lista **; done**
Suoritetaan **until** lista, mikäli viimeisen komennon palautusarvo on nollasta poikkeava suoritetaan **do** lista. Kun viimeinen komento **until** listassa palauttaa arvon nolla niin poistutaan silmukasta.

while lista **; do** lista **; done**
Suoritetaan **while** lista. Mikäli listan viimeisen komennon palautusarvo on nolla suoritetaan **do** lista ja suoritetaan **while** lista uudestaan. Silmukka päättyy kun **while** listan viimeinen komento palauttaa nollasta poikkeavan arvon.

4 Korn Shell (ksh)

4.1 Ominaisuuksia ja alustustiedostot

Korn shellin omainaisuuksia ovat mm: editoitava historia, aliakset, funktiot, sisäänrakennettu aritmetiikka, töidenhallinta ja debuggaus. Bourne shell on lähes täysin ylöspäin yhteensopiva Korn shellin kanssa, joten vanhemmat Bourne shellin mukaiset komentojonot toimivat Korn shellissä.

Samoin kuin Bourne shell myös ksh suorittaa systeemitiedoston `/etc/profile` ja sen jälkeen käyttäjän kotihakemistosta `.profile` tiedoston. Näiden tiedostojen lisäksi ksh tutkii jos muuttuja ENV asetettu ja suorittaa muuttujassa määritellyn tiedoston. ENV muuttuja voidaan asettaa esimerkiksi `.profile` tiedostossa. Useasti ENV muuttujaan on asetettu tiedosto `.kshrc`. ENV muuttujassa asetettu tiedosto suoritetaan jokaiselle ksh alishellille.

4.2 Aliakset

Aliakset ovat shellin tai käyttäjän määrittelemiä lyhenteitä komennoille. Aliaksen nimessä voidaan käyttää kirjaimia ja numeroita. Shellin määrittelemät aliakset voidaan määritellä uudelleen tai poistaa. Aliakset saadaan voimaan myös alishelleissä määrittelemällä ne ENV muuttujan määräämässä tiedostossa. Ksh ei tue argumenttien välittämistä aliaksille. Määritellyt aliakset voidaan listata komennolla `alias`.

```
alias ls='ls -la'
```

4.3 Töidenhallinta

Töidenhallintaa käytetään etualalla ja taustalla ajettavien töiden suorittamiseen. Ksh:ssa on käytössä seuraavat komennot, joilla töitä voidaan hallita.

jobs
Listaa kaikki keskeneräiset prosessit. Hakasuluissa ilmoitetaan työn numero.

jobs -l
Sama kuin `jobs`, mutta ilmoitetaan myös työn PID.

^Z
Pysäyttää nykyisen työn.

fg %n
Asettaa työn numero `n` suorituksen etualalle.

bg %n
Asettaa työn numero `n` suorituksen taustalle.

wait %n
Odottaa työn numero `n` valmistumista.

kill %n
Lopettaa työn numero `n` suorituksen.

4.4 Ympäristömuuttujat

Bourne shellin osuudessa lueteltujen ympäristömuuttujien lisäksi ksh käyttää seuraavia muuttujia.

CDPATH, COLUMNS, EDITOR, ENV, ERRNO, FCEDIT, FPATH, HISTEDIT, HISTFILE, LINENO, LINES, OLDPWD, PPID, PS3, PS4, RANDOM, REPLY, TMOUT ja VISUAL.

4.5 Paikkasidonnaiset parametrit

Ksh:ssa ovat samat paikkasidonnaiset parametrit kuin Bourne shellissä, mutta parametrien määrää ei ole rajoitettu yhdeksään. Esimerkiksi parametriin numero 10 voidaan viitata `${10}`.

4.6 Lista

`|&` Aiheuttaa edeltävän putken tai komennon asynkronisen suorittamisen siten, että parent shelliin muodostuu kaksisuuntainen putki. Komennon tulostus/syöte voidaan lukea/kirjoittaa parent shellistä komennolla `read/print` käyttäen optiota `-p`.

4.7 Korvaukset

4.7.1 "Mato"merkin korvaus (~ tilde)

Mikäli sana alkaa (ei lainausmerkeissä olevalla) ~ merkillä, tarkistetaan vastaako merkkien ~ ja / välissä oleva osa jotain käyttäjätunnusta /etc/passwd tiedostossa. Mikäli vastaavuus löytyy ~ ja käyttäjätunnus korvataan käyttäjän kotihakemistolla. Pelkkä ~ tai ~ / korvataan `HOME` parametrin arvolla. Jos ~ merkkiä seuraa joko + tai − niin pari korvataan parametrien `PWD` tai `OLDPWD` arvoilla. Edellä esitellyt korvaukset tehdään myös silloin kun parametri alkaa ~ merkillä.

4.7.2 Parametrien korvaus

${#parametri}

Mikäli parametri on * tai @ korvataan paikkasidonnaisten parametrien määrällä. Muuten korvataan parametrin pituudella

```
echo $SHELL; echo ${#SHELL}
echo ${#*}
```

${#parametri[*]}

Korvataan taulukon alkioiden määrällä.

```
print ${#x[*]}
```

${parametri#hahmo}

Mikäli shellin hahmo vastaa parametrin alkua korvaus tehdään seuraavasti: parametrin alusta poistetaan lyhin hahmoa vastaava osa ja korvaus suoritetaan jäljelle jäävällä parametrin osalla.

```
echo $SHELL; echo ${SHELL#*/}
```

${parametri##hahmo}

Samoin kuin edellä, mutta korvattaessa parametrin alusta poistetaan pisin hahmoa vastaava osa

```
echo $SHELL; echo ${SHELL##*/}
```

${parametri%hahmo}

Mikäli shellin hahmo vastaa parametrin loppua korvaus tehdaan seuraavasti: parametrin lopusta poistetaan lyhin hahmoa vastaava osa ja korvaus suoritetaan jäljelle jäävällä parametrin osalla.

```
echo $SHELL; echo ${SHELL%/*}
```

${parametri%%hahmo}

Samoin kuin edellä mutta korvattaessa parametrin lopusta poistetaan pisin hahmoa vastaava osa

```
echo $SHELL; echo ${SHELL%%b*}
```

4.7.3 Komentojen suoritus

Ksh:ssa komennon tulostus voidaan laittaa muuttujaan samoin kuin Bourne shellissäkin eli laittamalla komento ‘ ‘-merkkien sisään. Toinen tapa tehdä sama asia on laittaa komento sulkuihin ja sulkujen eteen $ merkki ($(komento)).

```
CUR_DIR=$(pwd)
```

4.8 Taulukot

Ksh:ssa voidaan määritellä myös yksiulotteisia taulukoita. Taulukon alkioiden maksimimäärä on 1024 alkioita. Alkiot voivat olla sanoja tai kokonaislukuja. Taulukon alkioiden indeksointi alkaa numerosta 0. Taulukon alkioita voidaan alustaa halutussa järjestyksessä. Taulukko voidaan määritellä joko kayttäen `set -A` komentoa esimerkiksi seuraavasti:

```
set -A x 1 2 3 4 5 6
```

Määrittelee `x` nimisen taulukon ja asettaa siihen alkiot 1,2,3,4,5 ja 6

tai taulukko voidaan alustaa suoraan:

```
x[0]=1
x[1]=2
x[2]=3
```

Taulukon arvoja voidaan tulostaa `print` komennolla seuraavasti:

```
print ${x[0]}
```

Tulostaa taulukon `x` alkion 0.

```
print ${x[*]}
```

Tulostaa taulukon `x` kaikki alkiot

```
print ${#x[*]}
```

Tulostaa taulukon alkioiden lukumäärän.

4.9 Ksh:n sisäisiä komentoja

alias
Listaa tai asettaa komentojen aliaksia.

bg
Asettaa annetun työ taustasuoritukseen.

fc
Käytetään komentorivihistorian editointiin. Ellei editoria ole määritetty käytetään FCEDIT muuttujassa määritettyä editoria, jos muuttujaa ei ole määritetty käytetaan `/bin/ed` editoria.

fg
Asettaa annetun työn edustasuoritukseen.

jobs
Listaa suorituksessa olevat työt.

print
Korvaa echo-komennon.

test
Merkkijonojen, lukujen jne. testaus.

whence
Tulostaa tietoja komennosta