Iterasjon i shellprogrammering

Løkkestrukturer i Bash *

- `while`
  - Går *så lenge* en løkkebetingelse er *true*
- `until`
  - Går *inntil* en løkkebetingelse er *true*
- `for in`
  - Itererer over hvert *element* i en angitt *liste*

*: I tillegg finnes det en `for` - løkke i Bash med syntaks som ligner på løkker i f.eks. Java og C, se lærebokens avsnitt 7.6.2

while og until

```
while condition
do
    commands
done
```

```
until condition
do
    commands
done
```

- *condition* håndteres likt som i `if` – setningen
- Gjør ofte tallregning i løkken som endrer betingelsen
- Løkker kan også styres med disse to kommandoene:

`break`	Avbryt og hopp ut av løkken
`continue`	Avbryt og gå til toppen av løkken igjen

`while` – eksempel: Sum av naturlige tall

```
#!/bin/bash

[ $# -ne 1 ] && \
{ echo "usage $0 number"; exit 1; }

i=0
sum=0
while [ $i -lt $1 ]
do
  ((i++))
  ((sum+=i))
done
echo 1+2+3+...+$1 = $sum
exit 0
```

`until` – eksempel: Hacker-detektoren

```
#!/bin/bash

[ $# -ne 1 ] &&\
{ echo "usage $0 name"; exit 1; }

until who | grep $1 > /dev/null
do
  echo "Waiting..."
  sleep 30
done

echo "`date`: Hacker alert! $1 has logged on!"
```

`shift`

- Kan brukes for å gå gjennom alle *parametrene* til et shellprogram med en løkke
- Et kall på `shift` flytter alle innparametre til scriptet "et hakk til venstre":
  - `$1` overskrives med `$2`
  - `$2` overskrives med `$3`
  - Etc...

shift – eksempel

```
#!/bin/bash

while [ "$1" -o "$2" -o "$3" ]
do
   echo $1 $2 $3
   shift; shift; shift
done
```

`for in` – løkke

```
for var in list
do
    command(s)
done
```

- *`list`* er en liste med verdier, adskilt med whitespace
- Listen kan angis eksplisitt eller med jokernotasjon, lages med en Linux-kommando eller hentes fra en variabel
- Løkken går én gang per element i listen
- I hver iterasjon lagres *nåværende* verdi i listen i *`var`*

Enkelt `for` - eksempel

```
#!/bin/bash

for i in en to tre "fire fem seks"
do
   echo "$i"
done
```

Antall filer i stående katalog *

```
#!/bin/bash

n_files=0

for file in `ls`
do
  n_files=$((n_files+1))
done

echo $n_files
```

*: Scriptet gir samme resultat som f.eks `ls | wc -l`

Summen av antall ord på flere filer

```
#!/bin/bash
total=0
for filename in $@
do
  if [ -r $filename ]
  then
    words=$(wc -w $filename | cut -f1 -d' ')
    ((total+=words))
  fi
done
echo $total
```

Alle tekstfiler i stående katalog med skrivetilgang

```
#!/bin/bash

for filename in *.txt
do
  if [ -w $filename ]
    then echo $filename
  fi
done
```

Antall forekomster av et ord på en fil

```
#!/bin/bash
[ $# -ne 2 ] &&\
{ echo "usage: $0 file word"; exit 1; }
[ ! -r $1 ] &&\
{ echo "$0: $1 is not a readable file"; exit 1; }

count=0
for word in `cat $1`
do
  if [ $word == $2 ]
  then
    ((count++))
  fi
done
echo "Occurrences of '$2' in file \"$1\": $count"
```

Bruk av `seq` sammen med løkker

```
seq [first] [step] last
```

- Returnerer liste av tall fra og med `first` (default 1) til og med `last` , med steg på `step` (default 1)
- Eksempel:

```
#!/bin/bash
for number in `seq 1 2 20`
do
   echo $number
done
```

Et litt større eksempel

- Parametre skal være en eller flere *kataloger*
- Scriptet sjekker først at antall parametre er riktig
- Deretter, for hver innparameter:
  - Sjekker at parameteren er en gyldig katalog
  - Flytter til denne katalogen
  - Sorterer alle tekst-filer i katalogen
  - Lagrer hver sortert fil med et filnavn som er likt navnet på usortert fil, men med strengen `_sorted` lagt til
- Kode: dirsort