Bash-scriptning

Linuxadministration I 1DV417

Script

bash

csh
ksh

Exekvering av script

bash exSimple

#!/bin/bash

Specialtecken/substitution

Specialtecken: Citattecken (”), grav accent (`), akut accent (´)

Variabelsubstitution — $username

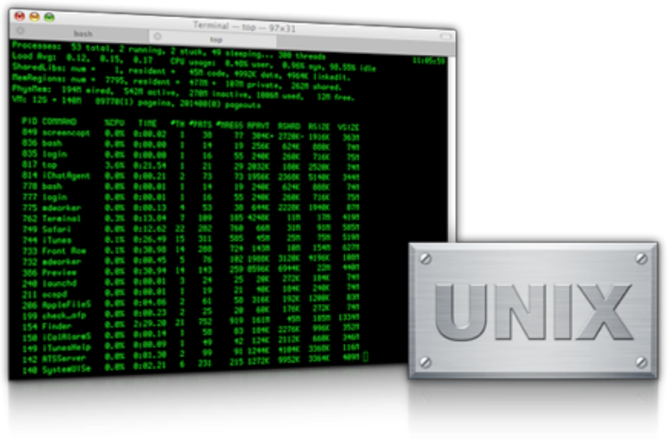
Kommandosubstitution — $(ls) alt. `ls`

Felvända fnuttar fungerar ej — ´ls´

Utskrift till konsolen

```
echo Hello world!
```

```
echo -n Hello world!
```



Variabler

Lokala variabler

```
username=kalle
```

Miljövariabler

```
export username=kalle
declare -x username = kalle
```

Variabler forts.

```
number=1
```

```
echo $number
```

Variabler forts.

declare för att skapa integer-variabler

```
declare -i counter=1
```

let

let counter+=1

+, -, *, /, %,=,+=

Array

```
declare -a users
```

```
declare -a users=(kalle olle
nisse)
```

```
${users[0]}
```

Array forts.

```
users[0]=peter
```

```
${users[*]}
${users[@]}
```

declare

Skapa skrivskyddade variabler, växeln –r
Skapa heltalsvariabler (integer), växeln –i
Skapa en array, växeln –a
Lista funktionsnamn och definitioner, växeln –f
Lista endast funktionsnamn, växeln -F
Exportera variabler till skalet, växeln –x

```
declare -r user=nisse
declare -i number=1
```

declare forts. / unset

```
unset number
```

Inparametrar

```
./script.sh Detta är ett test.
```

$0 = script.sh
$1 = Detta
$2 = är
...
$# = 4
$@ = [Detta][är][ett][test.]

read

```
read username
```

read variable

Sparar inläsningen i variabeln variable

read first last

Sparar första ordet i variabeln first och resterande i variabeln last. Det är möjligt att lägga till fler variabler för att separera inläsningen.

read –a array

Sparar inläsningen till arrayen array med ett ord i varje element

read –p ”Please enter your username: ” username

Visar meddelandet ”Please enter your username” och sparar inläsningen i variabeln username

Upprepningar

```
for username in ${users[*]}
do
    echo $username
done
```

```
while [villkor]
do
    // kod som ska upprepas
done
```

```
for username in kalle olle nisse
do
    echo $username
done
```

```
until [villkor]
do
    // kod som ska upprepas
done
```

if / else

```
if [villkor]
then
    // kod som ska utföras om villkoret är sant
else
    // kod som ska utföras om villkoret är falskt
fi
```

```
if [villkor1]
then
    // kod som ska utföras om villkor1 är sant
elif [villkor2]
then
    // kod som ska utföras om villkor2 är sant
fi
```

case

```
case [variabel] in
värde1)
    // kod som ska köras om variabelns värde motsvarar värde1     ;;
värde2)
    // kod som ska köras om variabelns värde motsvarar värde2     ;;
*)
    // kod som ska köras om inte något av värdena  i case-satsen matchar.
;;
esac
```

Villkor

BASH skiljer på villkor för

- Numeriska värden
- Strängar

Numeriska värden

- Utvärderas med hjälp av det inbyggda kommandot let
- För att utvärdera numeriska värden innesluts testet med dubbla parenteser
- Giltiga operatorer för numeriska villkor innefattar bland annat
  <, >, <=, >=, ==, !=, &&, ||

```
(( 10 != 11 ))
```

Strängar

- För att utvärdera strängar innesluts testet med []
- OBS! det måste vara ett blanksteg före respektive efter villkoret
- Utvärderas med hjälp av det inbyggda kommandot test

```
[ kalle == nisse ]
```

Villkor forts.

För att testa om

två strängar är lika: [sträng1 == sträng2]
två strängar inte är lika: [sträng1 != sträng2]
strängen är null: [sträng]
strängens längd är 0: [-z sträng]
strängens längd är inte 0: [-n sträng]

Heltalstest på strängar

tal1 är lika med tal2: [tal1 –eq tal2]
tal1 är inte lika med tal2: [tal1 –ne tal2]
tal1 är större än tal2: [tal1 –gt tal2]
tal1 är större eller lika med tal2: [tal1 –ge tal2]
tal1 är mindre än tal2: [tal1 –lt tal2]
tal1 är mindre eller lika med tal2: [tal1 –le tal2]

Villkor för filer och kataloger

Det finns speciella villkorstest för filer och kataloger

För att undersöka om:

Filen finns. Växel –e
Det är en vanlig fil. Växel –f
Det är en katalog. Växel –d
Det är en symbolisk länk. Växel –L
Skriptet har läsrättigheter till filen. Växel –r
Skriptet har skrivrättigheter till filen. Växel –w
Skriptet har körrättigheter till filen. Växel –x
Filen är en named pipe. Växel -p

```
[ -e /etc/passwd ]
[ -f /etc/group ]
[ -d /etc ]
[ -L /root/labadminhome ]
```

Returvärden från script

```
exit 1
```

Funktioner

```
function meanvalue
{
    // funktionskod
}
```

```
meanvalue
```

Parametrar till funktioner

```
meanvalue 34 89 76
```

Inläsning från fil

```
#!/bin/bash
while read –a column
do
    echo ”Användarnamn: ${column[0]}”
    echo ”Förnamn: ${column[1]}”
    echo ”Efternamn ${column[2]}”
done
```

```
./readfile < usernames.txt
```