

Shell scripting

NLLGG Lezing
9 februari 2002

Marcel Nijenhof
marceln@xs4all.nl

Introductie

- De eerste kennismaking met de shell is vaak na de eerste login dan wel het starten van een terminal window vanuit een grafische omgeving. Daar staat dan de shell prompt waar je veel commando's kunt geven.
- Verder wordt de shell vaak gebruikt voor het schrijven van scripten. O.a de opstartscripten voor diverse applicaties zijn vaak in de shell geschreven.

Wat gaan we wel/niet doen

Wel

- Gebruik maken van de posix shell/ksh/bash
- Kijken na het gebruik van variabele
- Kijken na functies om scripten te schrijven
- Kijken na getopt/command line parsing
- mailfile als voorbeeld bekijken

Niet

- Een beschrijving geven van csh (of de klasieke bourne shell)
- Een beschrijving geven van awk, sed, cut, expr

Het gebruik van variabele

- Het zetten van het display via: `export DISPLAY=naam:0`
- Deze waarde kunnen we gebruiken via: `echo $DISPLAY`
- Binnen scripten zelf is het echter niet nodig om `export` te gebruiken. Dit is alleen nodig als de variabele nog in andere programma's gebruikt wordt.
- Je kunt dus gebruik maken van: `NAAM=waarde`
- Het gebruik kan ook via: `${NAAM}`
- Met deze constructie kun je een hoop leuke dingen doen
 - `${NAAM%%STR}`: Verwijder STR aan het eind
 - `${NAAM##STR}`: Verwijder STR aan het begin
 - `${NAAM[nr]}`: Arrays van variabele
 - `${NAAM:-STR}`: Als naam niet gezet is gebruik STR
 -

.....: `bash(1)`, `posix-sh(1)`, `ksh(1)`

Rekenen, externe commando's, redirectie, pipes

- Binnen de shell kun je rekenen via: `$((...))`
- Externe commando's uitvoeren kan via
 - 'commando'
 - `$(commando)`
- Met behulp van redirectie `>` of `<` kun je lezen van of
- schrijven na een file

- Met behulp van pipes `|` kun je de output van het eerste commando gebruiken als invoer voor het volgende
- Voorbeelden:
 - `nr=$(($nr + 1))`
 - `HOSTNAME=$(hostname)`
 - `USER=$(id -un)`
 - `ls | sort`

○ `sort < /etc/passwd > /tmp/passwd`

Testen

- Een programma heeft een exit status. Voor een normaal programma geldt dat de exit status "0" is als het gelukt is.
- Een uitzondering hierop is uiteraard het commando false die altijd een exit status ongelijk "0" geeft.
- Er is een commando test waarmee je diverse waarde kunt

testen

- Bestaat een file
- Kun je de file lezen
- is een bepaalde variabele gezet
- Heeft een bepaalde variabele een bepaalde waarde
- Het commando "test" wordt ook vaak geschreven als: [...
- voorbeelden:
 - [-f /etc/passwd]
 - ["\${VAR}" = waarde]
 - ["\${NR}" -lt 5]

Controle structuren

- test && commando
- test || commando
- if [...]; then ...; elif [...]; then ...; else ...; fi
- while [...]; do ...; done
- for var in list; do ...; done

```
for f in *.JPG
do
    if [ -f ${f} -a ! -f ${f%.JPG}.jpg ]
    then
        mv ${f} ${f%.JPG}.jpg
    fi
done
```

Controle structuren (2)

- case waarde in; STR1) ... ;; STR2) ... ;; esac

```
case ${opt} in
    STR1)
        echo STR1
        ;;
    STR2)
        echo STR2
        ;;
    *)
        echo Onbekent
        ;;
esac
```

functies

- Het is mogelijk om in de shellsript functies te schrijven waarmee een stuk code op meerdere plaatsen kunt uitvoeren. Verder kun je hiermee je code onderverdelen in stukken code.
- De definitie van een functie gaat via een naam gevolgt door () en dan de code tussen { }

```
showerror() {  
    echo "ERROR(${PROG}): $1"  
    echo "ERROR(${PROG}): $1" >&2  
}
```

getopts

getopts is een shell functie om de argumenten van het script op uniformen wijze te verwerken.

- Het maakt niet uit of de optie los staat of er veel achter een "-" teken staan
- Het maakt niet uit of er een spatie staat tussen de optie en het argument

Voorbeeld getopt

```
while getopt ':fa:' opt
do
  case "${opt}" in
    f) FLAG=y;;
    a) ARG_A="${OPTARG}";;
    \?) showerror "${opt}: unknown option";;
  esac
done
shift $(( ${OPTIND} - 1 ))

[ "${FLAG}" = y ] && echo Optie -f gebruikt
[ -n "${ARG_A}" ] && echo ARG_A: "${ARG_A}"
nr=1;
for args in "$@"
do
  echo ARG[$nr]: "$args"
  nr=$((nr+1))
done
```

set

- Met behulp van set kun je het gedrag van de shell aanpassen
 - set -o vi: Vi edit mode
 - set -x: Print de commando's die uitgevoerd worden set -e: exit als er iets fout gaat
 - set -u: Niet gezetten variabele veroorzaken een fout
- De "-u" kan een script wat robuster maken
- De "-e" zorgt er voor dat scripten niet door gaan als er een fout is
- De "-x" zorgt er voor dat je een script makkelijk kunt debuggen

Voorbeeld script

□ mailfile

- mailfile -f file -m foo@bar mailt file na "foo@bar"
- mailfile kan ook een `${HOME}/etc/mailfile.rc` lezen waarin meerdere file en adressen staan
- mailfile kan na een logfile schrijven wat hij doet

<http://www.nllgg.nl/nllgg/bijeenkomsten/20020209/shell/mailfile>