academiejaar 2014-2015

1e jaar Bachelor in de Informatica

Scriptingtalen

bash
shell variabelen

prof. dr. Peter Dawyndt

Peter.Dawyndt@UGent.be

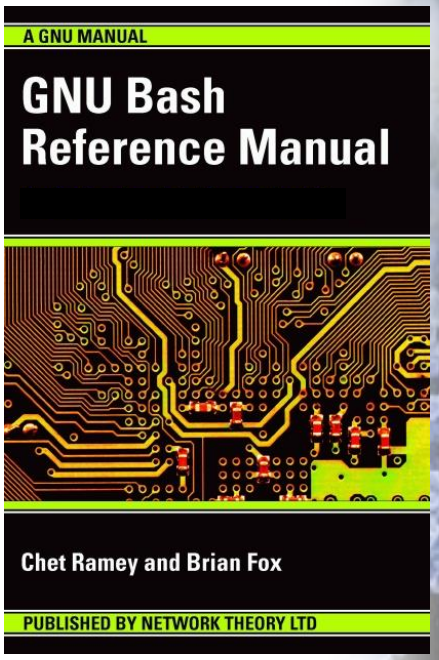
UNIVERSITEIT GENT

Referentiemateriaal

- GNU Bash Reference Manual (versie 4.2, Dec 2010)
  - online beschikbaar op Minerva
  - naslagwerk voor
    - omgevingsvariabelen
    - ingebouwde commando's
  - beschrijving van niet-behandelde features van **`bash`** shell

- SHELLdorado

Heiner's SHELLdorado



UNIVERSITEIT GENT

Shell scripts uitvoeren

- scriptbestand uitvoerbaar maken
  - script uitgevoerd door kindproces met kopie van huidige shell
  - weinig controle door welke type shell het script wordt uitgevoerd

UNIVERSITEIT GENT

```
$ chmod u+x script
$ script
```

Shell scripts uitvoeren

- scriptbestand uitvoerbaar maken
  - script uitgevoerd door kindproces met kopie van huidige shell
  - weinig controle door welke type shell het script wordt uitgevoerd
- uitvoeren commando **`/bin/bash`** met script als parameter
  - afgekort tot **`bash`** als zoekpad de **`/bin`** directory bevat

```
$ /bin/bash script
```

UNIVERSITEIT GENT

Shell scripts uitvoeren

- scriptbestand uitvoerbaar maken
  - script uitgevoerd door kindproces met kopie van huidige shell
  - weinig controle door welke type shell het script wordt uitgevoerd
- uitvoeren commando **`/bin/bash`** met script als parameter
  - afgekort tot **`bash`** als zoekpad de **`/bin`** directory bevat
- hash-bang (*shebang*) regel
  - meest gebruikte manier van uitvoeren
  - huidige shell laat script uitvoeren door opgegeven interpreter
  - uitvoerbaar scriptbestand kan als commando gestart worden

```
#!/bin/bash
script-commando's
```

shell interpreteert deel na **`#!`** als absolute padnaam van *interpreter* die huidige script moet uitvoeren

UNIVERSITEIT GENT

Shell variabelen

- variabele is benoemde plaats in hoofdgeheugen
  - verwijzing naar geheugenlocatie kan op basis van naam in plaats van adres
- regels voor benoemen van variabelen
  - variabele kan bestaan uit
    - cijfers
    - letters
    - onderstrepingstekens (underscore; _)
  - eerste karakter mag geen cijfer zijn

UNIVERSITEIT GENT

Shell variabelen

- hoofdgeheugen als opslagplaats is lezen/schrijven
  - inhoud van variabele kan worden uitgelezen
  - nieuwe inhoud kan naar variabele worden geschreven
  - Bourne Again shell beschouwt inhoud variabele altijd als string
  - theoretisch geen beperking op lengte inhoud van variabele
- soorten shell variabelen
  - omgevingsvariabelen (*environment variables*)
  - gebruikersvariabelen (*user-defined variables*)

UNIVERSITEIT GENT

Omgevingsvariabelen

- worden gebruikt om
  - context waarin shell wordt uitgevoerd aan te passen
  - context waarin commando's worden uitgevoerd aan te passen
- waarde van omgevingsvariabelen
  - initialisatie bij uitvoeren van **`/etc/profile`** bestand tijdens loginprocedure (systeembeheerder)
  - aangepaste initialisatie bij uitvoeren van **`~/.profile`** bestand tijdens loginprocedure (eindgebruiker)
  - kopie doorgegeven aan elk commando en elke subshell die als kindproces van shell wordt opgestart

Omgevingsvariabelen

- **`CDPATH`** zoekpad van **`cd`** commando
  - » directorynamen worden één voor één doorzocht om directory te vinden die als parameter werd doorgegeven aan **`cd`** commando
  - » huidige directory wordt doorzocht als variabele niet is ingesteld
- **`EDITOR`** standaard editor
  - » bijvoorbeeld gebruikt door emailprogramma
- **`HOME`** home directory van gebruiker
- **`IFS`** lijst van veldscheidingstekens
- **`MAIL`** mailboxbestand van gebruiker
- **`MAILCHECK`** hoe vaak moet shell mailbox controleren
  - » frequentie uitgedrukt in seconden
  - » gebruiker wordt verwittigd bij nieuw binnengekomen email

UNIVERSITEIT GENT

lezen/schrijven

Omgevingsvariabelen

- **`PATH`** zoekpad van gebruiker
  - » directories die door shell doorzocht worden om extern commando of programma te vinden
- **`PS1`** eerste shellprompt die op commandolijn verschijnt
  - » standaard ingesteld op **$**
- **`PS2`** tweede shellprompt
  - » verschijnt op tweede lijn van commando als shell denkt dat commando nog niet volledig is ingegeven
  - » standaard ingesteld op **>**
- **`PWD`** naam huidige directory
- **`TERM`** type terminal waarop gebruiker aan het werken is

UNIVERSITEIT GENT

lezen/schrijven

Omgevingsvariabelen

- `0` naam van commando
- `1-9` waarde van commandolijnargumenten 1-9
- `*` waarde van alle commandolijnargumenten
- `@` waarde van alle commandolijnargumenten
  - elke argument wordt afzonderlijk tss dubbele aanhalingstekens geplaatst als `$@` zelf tussen dubbele aanhalingstekens wordt geplaatst (`"$@"`)
- `#` aantal commandolijnargumenten
- `$` procesidentifier (PID) van huidige proces
- `PPID` procesidentifier van het ouderproces
- `?` exit status van laatst uitgevoerde proces
- `!` procesidentifier (PID) van laatste achtergrondproces

UNIVERSITEIT GENT

enkel-lezen

Gebruikersvariabelen

- gebruikt als tijdelijke opslagplaats in shell script
  - waarde kan wijzigen tijdens uitvoeren shell script
  - ingesteld als lezen/schrijven of als enkel-lezen
  - moeten niet gedeclareerd of geïnitialiseerd worden
    - niet-geïnitialiseerde shell variabele wordt standaard ingesteld op lege string (*null string*)

UNIVERSITEIT GENT

Weergeven shell variabelen

- commando **`set`**
  - indien gebruikt zonder argumenten toont **`set`** waarde van
    - omgevingsvariabelen
    - gebruikersvariabelen
  - kan gebruikt worden om waarde van enkel-lezen omgevingsvariabelen aan te passen (*zie verder*)
- commando's **`env`** (System V) en **`printenv`** (BSD)
  - tonen waarde van
    - omgevingsvariabelen (minder compleet)

UNIVERSITEIT GENT

Shell variabelen instellen

- syntaxis

```
naam1=waarde1 [naam2=waarde2 … naamN=waardeN]
```

- betekenis
  - kent ***waardeX*** toe aan variabele met naam ***naamX***
- opmerkingen
  - geen spaties toegelaten voor en achter gelijkteken
  - waarden die spaties bevatten moeten worden ingesloten tussen aanhalingstekens
    - verschil tussen enkele en dubbele aanhalingstekens (*zie verder*)
  - toekennen lege string is toegelaten

UNIVERSITEIT GENT

Shell variabelen uitlezen

- syntaxis

```
$naam
```

- opmerkingen
  - commando **echo** kan gebruikt worden om waarde uit te schrijven
  - gebruik accolades indien einde van naam van variabele niet meteen duidelijk is

```
$ SUB="www"
$ SUBDIR="www_tmp"
$ cp /staff/fwet/pdawyndt/$SUBDIR/project .
$ cp /staff/fwet/pdawyndt/${SUBDIR}/project .
$ cp /staff/fwet/pdawyndt/${SUB}DIR/project .
$
```

UNIVERSITEIT GENT

Shell variabelen gebruiken

```
$ naam=Gaston
$ echo $naam
Gaston
$ naam=Gaston Lagaffe
bash: Lagaffe: command not found
$ naam="Gaston Lagaffe"
$ echo $naam
Gaston Lagaffe
$ naam=Gaston*
$ echo $naam
Gaston.Lagaffe.grap Gaston.Lagaffe.strip Gaston.tex
$ echo "$naam"
Gaston*
$ echo "De naam $naam klinkt vertrouwd!"
De naam Gaston* klinkt vertrouwd!
```

shell probeert tweede token uit te voeren als commando

dubbele aanhalingstekens laten substitutie van variabelen toe, maar geen bestandsnaamexpansie

UNIVERSITEIT GENT

Shell variabelen gebruiken

```
$ echo \$naam
$naam
$ echo '$naam'
$naam
$
```

backslash kan gebruikt worden om de speciale betekenis van één enkel karakter ongedaan te maken

enkele aanhalingstekens maken speciale betekenis van elk karakter ongedaan (escape sequence)

UNIVERSITEIT GENT

Shell variabelen gebruiken

```
$ commando=pwd
$ $commando
/staff/fwet/pdawyndt
$ commando=hallo
$ $commando
bash: hallo: command not found
$
```

UNIVERSITEIT GENT

Variabelen exporteren

- syntaxis

```
export namenlijst
```

- betekenis
  - exporteert namen en kopieert huidige waarden in *namenlijst* naar elke subshell die hierna wordt uitgevoerd
- opmerkingen
  - namen in *namenlijst* gescheiden door spaties
  - waarde van omgevingsvariabelen wordt altijd doorgegeven
  - standaardbereik van gebruikersvariabele is huidige shell
  - commando **`export`** geeft waarden door aan kindprocessen
  - hou rekening met feit dat shell script wordt uitgevoerd door nieuwe shell (*kindproces*)

Variabelen exporteren

- syntaxis

```
export namenlijst
```

kopie van variabele **naam** wordt doorgegeven aan alle commando's en alle subshells die vervolgens worden opgestart

```
$ naam="Gaston Lagaffe"
$ export naam
$
```

UNIVERSITEIT GENT

Variabelen exporteren

- syntaxis

```
export namenlijst
```

```
$ cat toon_naam
#!/bin/bash
echo $naam
exit 0
$ naam="Gaston Lagaffe"
$ toon_naam

$
```

UNIVERSITEIT GENT

Variabelen exporteren

- syntaxis

```
export namenlijst
```

```
$ cat toon_naam
#!/bin/bash
echo $naam
exit 0
$ naam="Gaston Lagaffe"
$ export naam
$ toon_naam
Gaston Lagaffe
$ echo $?
0
$
```

UNIVERSITEIT GENT

Variabelen exporteren

```
$ cat export_demo
#!/bin/bash
naam="Gaston Lagaffe"
export naam
toon_naam_en_wijzig
toon_naam
exit 0
$ cat toon_naam_en_wijzig
#!/bin/bash
echo $naam
naam="Leon Prunelle"
echo $naam
exit 0
$ export_demo
Gaston Lagaffe
Leon Prunelle
Gaston Lagaffe
$
```

```
/bin/bash toon_naam_en_wijzig
naam:Leon Prunelle
```

```
/bin/bash toon_naam_en_wijzig
naam:Gaston Lagaffe
```

UNIVERSITEIT GENT

Variabelen vrijgeven

- syntaxis

```
unset namenlijst
```

- betekenis
  - geheugen ingenomen door variabelen uit *namenlijst* wordt terug vrijgegeven
- opmerkingen
  - namen in *namenlijst* gescheiden door spaties
  - wanneer geheugenruimte voor shell variabele wordt gereserveerd, blijft die normaalgezien behouden zolang shell waarin initialisatie gebeurde actief is
  - commando **`"tekst="`** maakt variabele leeg, maar geeft geheugenruimte niet terug vrij

UNIVERSITEIT GENT

Variabelen vrijgeven

```
$ naam=Gaston plaats=Brussel
$ echo "$naam $plaats"
Gaston Brussel
$ unset naam
$ echo "$naam"

$ echo "$plaats"
Brussel
$ unset naam plaats
$ plaats=
$ echo "$plaats"

$
```

UNIVERSITEIT GENT

Symbolische constanten

- syntaxis

```
readonly namenlijst
```

- betekenis
  - waarde van variabelen uit *namenlijst* wordt "bevroren"
- opmerkingen
  - goed programmeerprincipe om letterlijke constanten in programmacode te vervangen door symbolische constanten (benoemde constanten)
  - weinig gebruikt in shell scripts
  - commando **`readonly`** zonder argumenten geeft lijst met namen en waarden van symbolische constanten

Symbolische constanten

- syntaxis

```
readonly namenlijst
```

```
$ naam=Gaston
$ plaats=Brussel
$ readonly naam plaats
$ echo "$naam $plaats"
Gaston Brussel
$ plaats=Parijs
bash: plaats: readonly variable
$ naam=Leon
bash: naam: readonly variable
$
```

UNIVERSITEIT GENT

Lezen uit stdin

- syntaxis

```
read namenlijst
```

- betekenis
  - leest één enkele regel uit stdin en kent velden uit die regel toe aan variabelen in *namenlijst*
- opmerkingen
  - gebruikt om interactieve shell scripts te schrijven
  - geeft **`true`** terug als een regel wordt uitgelezen, en **`false`** als **`EOF`** bereikt wordt
  - velden gescheiden door delimiter uit omgevingsvariabele **`IFS`**
    - **`IFS`** is standaard ingesteld op witruimte (spaties, tabs en newlines)
    - velden in volgorde toegekend aan variabelen

UNIVERSITEIT GENT

Lezen uit stdin

- syntaxis

```
read namenlijst
```

- betekenis
  - leest één enkele regel uit stdin en kent velden uit die regel toe aan variabelen in *namenlijst*
- opmerkingen
  - gebruikt om interactieve shell scripts te schrijven
  - geeft **`true`** terug als een regel wordt uitgelezen, en **`false`** als **`EOF`** bereikt wordt
  - velden gescheiden door delimiter uit omgevingsvariabele **`IFS`**
    - #velden > #variabelen
      ⇒ resterende velden toegekend aan laatste variabele
    - #velden < #variabelen
      ⇒ lege string toegekend aan resterende variabelen

UNIVERSITEIT GENT

Lezen uit stdin

```
$ cat read_demo
#!/bin/bash
echo -e "Geef invoer: \c"
read lijn
echo "Je gaf: $lijn"
echo -e "Geef nog een regel: \c"
read woord1 woord2 woord3
echo "Het eerste woord is: $woord1"
echo "Het tweede woord is: $woord2"
echo "Het derde woord is: $woord3"
exit 0
$ read_demo
Geef invoer: UNIX rules the network computer world!
Je gaf: UNIX rules the network computer world!
Geef nog een regel: UNIX rules the network computer world!
Het eerste woord is: UNIX
Het tweede woord is: rules
Het derde woord is: the network computer world!
$
```

speciaal karakter dat verhindert dat **echo** naar een nieuwe regel springt op einde van uitvoer

newline op einde van regel wordt niet toegekend aan variabele

UNIVERSITEIT GENT

Speciale echo-karakters

- **`\b`** backspace
- **`\c`** drukt regel af zonder cursor op volgende regel te plaatsen
- **`\f`** form feed
- **`\n`** newline (plaatst cursor op volgende regel)
- **`\r`** carriage return
- **`\t`** horizontale tab
- **`\v`** verticale tab
- **`\\`** backslash (heft speciale betekenis van backslash op)
- **`\o`*`n`*** letterteken met octale waarde ***`n`*** uit ASCII tabel

op sommige UNIX systemen moet je **`\\`** gebruiken in plaats van **`\`**

UNIVERSITEIT GENT

Argumenten doorgeven

- shell scripts gebruiken positionele argumenten
  - in tegenstelling tot benoemde argumenten die door sommige programmeertalen gebruikt worden
  - verwijzen naar eerste negen argumenten: **`$1-$9`**
    - totaal aantal argumenten: **`$#`**
    - *n* > **`$#`** ⇒ **`$`*`n`*`==""`**
  - verwijzen naar alle argumenten: **`$*`** of **`$@`**
    - **`$@`** plaatst alle individuele argumenten binnen aanhalingstekens als het wordt gebruikt als **`"$@"`**
  - verwijzen naar naam van script: **`$0`**

UNIVERSITEIT GENT

Argumenten doorgeven

```
$ cat cmdargs_demo
#!/bin/bash
echo "De naam van het commando is: $0."
echo "Het aantal argumenten is: $#."
echo "Dit zijn de argumenten: $1 $2 $3 $4 $5 $6 $7 $8 $9."
echo "Andere manier om argumenten weer te geven: $@."
echo "Nog een alternatieve manier is: $*."
exit 0
$ cmdargs_demo a b c d e f g h i j
De naam van het commando is: cmdargs_demo.
Het aantal argumenten is: 10.
Dit zijn de argumenten: a b c d e f g h i.
Andere manier om argumenten weer te geven: a b c d e f g h i j.
Nog een alternatieve manier is: a b c d e f g h i j.
$ cmdargs_demo een twee 3 vier 5 6
De naam van het commando is: cmdargs_demo.
Het aantal argumenten is: 6.
Dit zijn de argumenten: een twee 3 vier 5 6.
Andere manier om argumenten weer te geven: een twee 3 vier 5 6.
Nog een alternatieve manier is: een twee 3 vier 5 6.
$
```

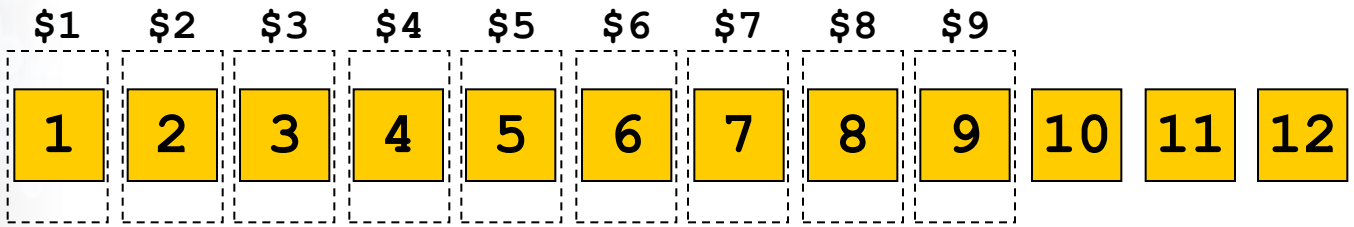
UNIVERSITEIT GENT

Argumenten doorschuiven

- syntaxis

```
shift N
```

- betekenis
  - schuift commandolijnargumenten *N* posities naar links
- opmerkingen
  - standaardwaarde van *N* is 1
  - argumenten die links wegvallen zijn niet langer toegankelijk



UNIVERSITEIT GENT

Argumenten doorschuiven

```
$ cat shift_demo
#!/bin/bash
echo "Naam van het commando: $0."
echo "Doorgegeven argumenten: $@."
echo "Eerste drie argumenten: $1 $2 $3."
shift
echo "Naam van het commando: $0."
echo "Doorgegeven argumenten: $@."
echo "Eerste drie argumenten: $1 $2 $3."
shift 3
echo "Naam van het commando: $0."
echo "Doorgegeven argumenten: $@."
echo "Eerste drie argumenten: $1 $2 $3."
exit 0
$ shift_demo 1 2 3 4 5 6 7 8 9 10 11 12
Naam van het commando: shift_demo.
Doorgegeven argumenten: 1 2 3 4 5 6 7 8 9 10 11 12.
Eerste drie argumenten: 1 2 3.
Naam van het commando: shift_demo.
Doorgegeven argumenten: 2 3 4 5 6 7 8 9 10 11 12.
Eerste drie argumenten: 2 3 4.
Naam van het commando: shift_demo.
Doorgegeven argumenten: 5 6 7 8 9 10 11 12.
Eerste drie argumenten: 5 6 7.
$
```

UNIVERSITEIT GENT

Argumenten overschrijven

- syntaxis

```
set [opties] [argumentenlijst]
```

- betekenis
  - stelt waarde van positionele argumenten in op die van argumenten uit *argumentenlijst*
- opmerkingen
  - **`set`** commando en commandosubstitutie vaak samen gebruikt
    - laat toe om commando-output automatisch in velden te splitsen
    - ontbrekende argumenten opgevuld met lege string
  - vaak gebruikt met optie **`--`**
    - geeft aan dat argumenten die volgen (en eventueel starten met een -) niet moeten beschouwd worden als opties van het **`set`** commando, maar als argumenten ervan

Argumenten overschrijven

- syntaxis

```
set [opties] [argumentenlijst]
```

```
$ date
Mon Mar 10 10:01:58 CET 2014
$ set `date`
$ echo "$@"
Mon Mar 10 10:02:04 CET 2014
$ echo "$2 $3, $6"
Mar 10, 2014
$
```

UNIVERSITEIT GENT

Argumenten overschrijven

```
$ cat set_demo
#!/bin/bash
bestand="$1"
set `ls -il $bestand`
inode="$1"
bytes="$6"
echo -e "Naam\tInode\tBytes"
echo
echo -e "$bestand\t$inode\t$bytes"
exit 0
$ set_demo set_demo
Naam            Inode          Bytes

set_demo        10153548       137
$
```

Wat gebeurt er als het bestand dat wordt doorgegeven aan **set_demo** niet bestaat ?

UNIVERSITEIT GENT

Commando xargs

- syntaxis

```
xargs commando
```

- betekenis
  - **xargs** leest stdin, en roept daarna *commando* op met ingelezen waarden als parameters
- opmerkingen
  - alternatief voor commandosubstitutie

```
$ ls | xargs echo
hfdst1.tex hfdst2.tex hfdst3.tex hfdst4.tex
$ echo `ls`
hfdst1.tex hfdst2.tex hfdst3.tex hfdst4.tex
```

UNIVERSITEIT GENT

Commando xargs

- syntaxis

```
xargs commando
```

- betekenis
  - **xargs** leest stdin, en roept daarna *commando* op met ingelezen waarden als parameters
- opmerkingen
  - alternatief voor commandosubstitutie

```
$ find . -name "*.tex" | xargs grep "\chapter"
...
$ grep "\chapter" `find . -name "*.tex"`
```

UNIVERSITEIT GENT

Commando xargs

- syntaxis

```
xargs commando
```

- betekenis
  - **xargs** leest stdin, en roept daarna *commando* op met ingelezen waarden als parameters
- opmerkingen
  - alternatief voor commandosubstitutie
  - alternatief voor problemen met *exec-buffer*

```
$ rm *
sh: arg list too long
$ ls | xargs rm
```

als parameterlijst te lang wordt, zal commando **rm** zoveel keren worden aangeroepen als nodig

UNIVERSITEIT GENT

Commentaar en headers

- maak er een gewoonte van om commentaar in te sluiten tijdens het schrijven van shell scripts
  - beschrijf functie van reeks commando's
  - beschrijf functie van variabele
- voeg aan elk shell script een hoofding (header) toe
  - informatieve header bevat oa. volgende informatie
    - naam van bestand dat script bevat
    - naam van auteur
    - datum aanmaken script
    - datum laatste wijziging
    - doel van script (in één of twee regels)
    - korte beschrijving van algoritme dat gebruikt werd om het gestelde probleem op te lossen

UNIVERSITEIT GENT

Commentaar en headers

- syntaxis

```
# commentaar
```

- betekenis
  - commentaar start met hekje (**#**) en eindigt met newline

```
#bestandsnaam:          set_demo
#auteur:                Peter Dawyndt
#aangemaakt:            10-03-2011
#laatste wijziging:     10-03-2014
#doel:                  illustreert hoe het set commando werkt
#korte omschrijving:    script wordt uitgevoerd met bestandsnaam als
                        enige commandolijnargument, onthoudt naam,
                        voert set commando uit om output van ls -il
                        toe te kennen aan positionele argumenten ($1
                        tot $9), en toont bestandsnaam, inode nummer
                        en grootte in bytes van het bestand.
```

UNIVERSITEIT GENT

Vragen of opmerkingen ?

UNIVERSITEIT GENT

The sky is the limit…

"If you don't know where you are going,
any road will get you there."

— Lewis Carroll

UNIVERSITEIT GENT