



# Shell Scripting

**AWK**

**DE HOGESCHOOL  
MET HET NETWERK**

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt  
[www.pxl.be](http://www.pxl.be) - [www.pxl.be/facebook](http://www.pxl.be/facebook)



# Intro AWK

- Eind jaren zeventig ontwikkeld door Alfred **A**ho, Peter **W**einberger en Brian **K**ernighan
- Standaard tool in POSIX-systemen
- Scripttaal
- Bedoeld voor automatisch verwerken van tekst
  - *Patronen zoeken en hiermee gevonden regels verwerken*
- Verschillende varianten
  - GAWK is daar 1 van (GNU AWK)
  - Ook voor windows zijn AWK-varianten beschikbaar



# Intro AWK

- AWK is line oriented
  - `pattern { action }`
  - Elke regel wordt gebruikt als input
  - Een patroon wordt getest op elke regel
    - In geval van een match wordt een actie uitgevoerd op deze regel
    - Indien geen patroon is opgegeven, wordt elke regel geselecteerd



# Secties in AWK

- `awk ' BEGIN { actions } /pattern/ { actions } /pattern/ { actions } END { actions } ' file(s)`
- 2 belangrijke secties: **BEGIN** en **END**

```
BEGIN { print "start" }  
{ print $0 }  
END { print "end" }
```

**print \$0:**     print de volledige regel af  
→ print zonder param print ook de volledige regel af  
→ zie volgende slide



**BEGIN:** acties die voorafgaand aan het lezen van de lijnen worden uitgevoerd

**END:** acties die na het verwerken van de lijnen worden uitgevoerd

# Een AWK-script uitvoeren

## Tekstbestand

```
Eerste lijn tekst  
Tweede lijn tekst  
Derde lijn tekst
```

## AWK-script

```
BEGIN { print "start" }  
{ print $0 }  
END { print "end"}
```

## Uitvoer van het script op het tekstbestand

```
start  
Eerste lijn tekst  
Tweede lijn tekst  
Derde lijn tekst  
end
```



# Een AWK-script uitvoeren

## Tekstbestand

```
Eerste lijn tekst  
Tweede lijn tekst  
Derde lijn tekst
```

## AWK-script

```
{ print $1 $2 $3 }
```

## Uitvoer van het script op het tekstbestand

```
Eerstelijntekst  
Tweedelijntekst  
Derdelijntekst
```

## AWK-script

```
{ print $1,$2,$3 }
```

## Uitvoer van het script op het tekstbestand

```
Eerste lijn tekst  
Tweede lijn tekst  
Derde lijn tekst
```



# Een AWK-script uitvoeren

- Aan de commandline

```
student@ubserv:~$ awk -F: 'BEGIN { print "User\t\tHomedir" } {  
print $1, "\t\t", $6 }' /etc/passwd  
User          Homedir  
root           /root  
daemon        /usr/sbin  
bin            /bin  
sys            /dev
```

- F: duidt aan dat dubbelpunt moet gebruikt worden als Field-seperator
- \$1 duidt aan dat het eerste veld moet afgedrukt worden
- \$6 duidt aan dat het zesde veld moet afgedrukt worden
- \t duidt aan dat er een tab moet tussengevoegd worden  
deze tabs moeten tussen dubbele quotes staan



# Een AWK-script uitvoeren

- In een shell script

```
student@ubserv:~$ cat awk_voorbeeld1.sh
#!/bin/bash --
# awk_voorbeeld 1
# awk in een shell script
# Date 02/12/2024
# Author Veerle Asaert
awk -F: '
BEGIN { print "User\t\tHomedir" }
{ print $1, "\t\t", $6 }
' $1

student@ubserv:~$ chmod u+x awk_voorbeeld1.sh
student@ubserv:~$ ./awk_voorbeeld1.sh /etc/passwd
User                Homedir
root                /root
daemon              /usr/sbin
bin                  /bin
sys                  /dev
```





# Een AWK-script uitvoeren

- In een AWK-script → vorm 1

```
student@ubserv:~$ cat awk_voorbeeld2.awk
# awk_voorbeeld 2
# awk in een awk script
# Date 02/12/2024
# Author Veerle Asaert
BEGIN { print "User\t\tHomedir" }
{ print $1, "\t\t", $6 }
student@ubserv:~$ awk -f awk_voorbeeld2.awk -F: /etc/passwd
User          Homedir
root          /root
daemon        /usr/sbin
bin           /bin
sys           /dev
```

**-f:** duidt aan dat er een file wordt gebruikt met je awk-code in



# Een AWK-script uitvoeren

- In een AWK-script → vorm 2

```
student@ubserv:~$ cat awk_voorbeeld1.awk
#!/usr/bin/awk -f
# awk_voorbeeld 1
# awk in een awk script
# Date 02/12/2024
# Author Veerle Asaert
BEGIN { print "User\t\tHomedir" }
{ print $1, "\t\t", $6 }
student@ubserv:~$ chmod u+x awk_voorbeeld1.awk
student@ubserv:~$ ./awk_voorbeeld1.awk -F: /etc/passwd
User          Homedir
root          /root
daemon        /usr/sbin
bin           /bin
sys           /dev
```

We zien dat de uitlijning met deze tabs geen zuivere output levert  
→ oplossing met behulp van printf  
→ zie volgende slide



# Formatteren van de output

%d Integers  
%f Float  
%s Strings  
%c Character

- printf

```
student@ubserv:~$ awk -F: 'BEGIN { print "Username  
UserID GroupID" } { printf "%20s %6d %6d \n", $1, $3, $4 }'  
/etc/passwd
```

Username	UserID	GroupID
root	0	0
daemon	1	1
bin	2	2
sys	3	3

%-20s

- - links uitgelijnd
- 20 veld van 20 karakters breed
- s een string

```
student@ubserv:~$ awk -F: 'BEGIN { print "Username  
UserID GroupID" } { printf "%-20s %-6d %-6d \n", $1, $3, $4 }'  
/etc/passwd
```

Username	UserID	GroupID
root	0	0
daemon	1	1
bin	2	2
sys	3	3



# Formatteren van de output

%d Integers  
%f Float  
%s Strings  
%c Character

## Tekstbestand

```
Toetsenbord 11.16
Muis 2.98
Muismat 1.36
```

## AWK-script

```
BEGIN { print "Artikel          Prijs" }
{ printf "%-15s %5.2f Euro\n", $1, $2 }
```

## Uitvoer van het script op het tekstbestand

```
Artikel          Prijs
Toetsenbord      11.16 Euro
Muis              2.98 Euro
Muismat          1.36 Euro
```

### Description

Print one position after the decimal  
Two positions after the decimal  
Eight-wide, two positions after the decimal  
Eight-wide, four positions after the decimal  
Eight-wide, two positions after the decimal, zero-filled  
Eight-wide, two positions after the decimal, left-justified  
Printing a much larger number with that same format

### Code

```
printf("%.1f", 10.3456);  
printf("%.2f", 10.3456);  
printf("%8.2f", 10.3456);  
printf("%8.4f", 10.3456);  
printf("%08.2f", 10.3456);  
printf("%-8.2f", 10.3456);  
printf("%-8.2f", 101234567.3456);
```

### Result

```
'10.3'  
'10.35'  
' 10.35'  
' 10.3456'  
'00010.35'  
'10.35 '  
'101234567.35'
```



# Filteren van lijnen met bepaald patroon

- `awk '/pattern/ {action}' file(s)`

```
student@ubserv:~$ awk -F: '/bash/ { print }' /etc/passwd
root:x:0:0:root:/root:/bin/bash
student:x:1000:1000:,,,:/home/student:/bin/bash
```

```
student@ubserv:~$ awk -F: 'BEGIN { print "Username
Homefolder" } /sh$/ { printf "%-20s %-15s \n", $1, $7 }'
/etc/passwd
Username                Homefolder
root                    /bin/bash
student                 /bin/bash
```



# Filteren van lijnen met bepaald patroon

- `awk '/pattern/ {action}' file(s)`

```
student@ubserv:~$ awk -F: \
'BEGIN { print "Username           Homefolder" } \
/sh$/ { printf "%-20s %-15s \n" , $1, "bash" } \
/nologin/ { printf "%-20s %-15s \n" , $1, "/" }' \
/etc/passwd
```

Username	Homefolder
root	bash
daemon	/
bin	/
sys	/



# Filteren van lijnen met bepaald patroon

- `awk '/pattern/ {action}' file(s)`

```
student@ubserv:~$ awk -F: '$3 <= 3 { print }' /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

Operator	Description
<code>x &lt; y</code>	Returns true if x is less than y
<code>x &lt;= y</code>	Returns true if x is less than or equal to y
<code>x == y</code>	Returns true if x is equal to y (for numbers and strings)
<code>x &gt; y</code>	Returns true if x is greater than y
<code>x &gt;= y</code>	Returns true if x is greater than or equal to y
<code>x != y</code>	Returns true if x is not equal to y



# Filteren van lijnen met bepaald patroon

- `awk '/pattern/ {action}' file(s)`

```
student@ubserv:~$ awk -F: '$6 ~ /[cw]$/ { print }' /etc/passwd
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false
```

```
student@ubserv:~$ awk -F: '$6 ~ /^\/home/ { print }' /etc/passwd
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
student:x:1000:1000:Student PXL:/home/student:/bin/bash
testuser:x:1001:1001:,,,:/home/testuser:/bin/bash
```

## Operator

## Description

x ~ y

Returns true if string x matches the regular expression represented by y

x !~ y

Returns true if string x does not match the regular expression represented by y





# Filteren van lijnen met bepaald patroon

- `awk '/pattern/ {action}' file(s)`

```
student@ubserv:~$ awk -F: '$6 ~ /^\/v.*d$/ || $3 == 1000 { print }' /etc/passwd
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
sshd:x:110:65534:./var/run/sshd:/usr/sbin/nologin
student:x:1000:1000:,,,:/home/student:/bin/bash
```

Symbol	Operator	Description
&&	And	Results true when all the expressions are true
	Or	Results true when any of the expressions is true
!	Not	Reverses (negates) the logical expression



# Automatische index

- NR
  - Built-in AWK variabele
  - Lijnnummer
  - Bij END: totaal aantal verwerkte records/lijnen

```
student@ubserv:~$ awk -F: 'NR < 5 { print NR "->" $1 } END { print "Aantal:"NR}'  
/etc/passwd  
1->root  
2->daemon  
3->bin  
4->sys  
Aantal:31
```



# Number Of Fields

- NF

```
student@ubserv:~$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      SFP.localdomain SFP

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0 ip6-localnet
student@ubserv:~$ awk '$1 ~ "^[0-9]+"' { print "Regel " NR " heeft " NF " velden"}'
/etc/hosts
Regel 1 heeft 2 velden
Regel 2 heeft 3 velden
```



# Naam van het bestand

- FILENAME

```
student@ubserv:~$ awk -F: 'END { print "Aantal regels in " FILENAME ":" NR }'  
/etc/hosts  
Aantal regels in /etc/hosts:11
```



# Field Separator

- FS
  - staat los van de variabele IFS van de bash-shell
  - bevat standaard een spatie en een tab

```
student@ubserv:~$ echo $PATH
/home/student/bin:/home/student/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
student@ubserv:~$ echo $PATH | awk -F: '{ print "First path to search for commands: " $1 }'
First path to search for commands: /home/student/bin
student@ubserv:~$ echo $PATH | awk 'BEGIN { FS=":" } { print "First path to search for commands: " $1 }'
First path to search for commands: /home/student/bin
```



# Record Seperator

- RS

## Tekstbestand

```
januari  
79  
  
februari  
132  
  
maart  
783
```

## AWK-script

```
BEGIN { RS="" }  
{ print $1 "-" $2 }  
END { print "Aantal maanden met uitgaven: " NR }
```

`RS=""` heeft ongeveer hetzelfde effect als `RS="\n\n+"`

Verschil: leading newlines in de input worden genegeerd in geval van `RS=""`

## Uitvoer van het script op het tekstbestand

```
januari-79  
februari-132  
maart-783  
Aantal maanden met uitgaven: 3
```



# OUTPUT Field Separator

- OFS

```
student@ubserv:~$ echo $PATH
/home/student/bin:/home/student/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
student@ubserv:~$ echo $PATH | awk -F: '{ print $1,$2,$3,$4 }'
/home/student/bin /home/student/.local/bin /usr/local/sbin /usr/local/bin
student@ubserv:~$ echo $PATH | awk -F: '{ OFS="\n"; print $1,$2,$3,$4 }'
/home/student/bin
/home/student/.local/bin
/usr/local/sbin
/usr/local/bin
```



# OUTPUT Record Seperator

- ORS

```
student@ubserv:~$ awk -F: '$3 < 5 { print $1 }' /etc/passwd
root
daemon
bin
sys
sync
student@ubserv:~$ awk -F: '$3 < 5 { ORS="-";print $1} END {ORS="\n";print ""}'
/etc/passwd
root-daemon-bin-sys-sync-
student@ubserv:~$
```





# Wiskundige bewerkingen

## Tekstbestand

```
Toetsenbord 450 Bfr  
Muis 120 Bfr  
Muismat 55 Bfr
```

## AWK-script

```
BEGIN { print "Artikel          Prijs" }  
{ printf "%-15s %5.2f Euro\n", $1, $2/40.33 }
```

Operator	Description
-----	-----
*	Multiply
/	Divide
%	Mod (returns remainder)
+	Add
-	Subtract
++	Increments value by 1
--	Decrements value by 1
+=	Adds the value

## Uitvoer van het script op het tekstbestand

Artikel	Prijs
Toetsenbord	11.16 Euro
Muis	2.98 Euro
Muismat	1.36 Euro

# String bewerkingen

## Tekstbestand

```
Veerle,Asaert,PXL  
Gert,Van Waeyenberg,PXL
```

## AWK-script

```
BEGIN { FS="," }  
{ print substr($1,1,1) ". " $2 }
```

string

startpos

#chars

## Uitvoer van het script op het tekstbestand

```
V. Asaert  
G. Van Waeyenberg
```

## Function

## Description

length(x)	It returns the length of the argument x. If the argument is not supplied, it finds out the length of the entire line.
substr(s1,s2,s3)	It returns a portion of the string of length s3, starting from position s2 in the string s1.
index(s1,s2)	It returns the position of the string s2 in the string s1. It returns 0 if it is not present.
split(s,a)	It splits the string s into an array a and optionally returns the number of fields. The field separator is specified by FS.
system("cmd")	It runs the Unix command, cmd, and returns its exit status.

# Zoeken en vervangen

## Tekstbestand

```
Veerle,Asaert,PXL  
Gert,Van Waeyenberg,PXL
```

## AWK-script

```
BEGIN { FS="," }  
{print $1,toupper($2)}
```

## Uitvoer van het script op het tekstbestand

```
Veerle ASAERT  
Gert VAN WAEYENBERG
```

Function	Description
<hr/>	<hr/>
<code>toupper(str)</code>	This converts the given string into upper case.
<code>tolower(str)</code>	This converts the given string into lower case.
<code>delete array [element]</code>	This deletes the specified element of the array.
<code>sub(r, s [,t])</code>	This substitutes the first occurrence of the regular expression <code>r</code> by <code>s</code> in the string <code>t</code> . If the string <code>t</code> is not supplied, <code>\$0</code> (entire line/record) is considered. The function returns 1 if successful and 0 otherwise.
<code>gsub(r,s)</code>	This substitutes <code>s</code> in place of <code>r</code> globally in <code>\$0</code> (entire line/record) and returns the number of substitution made.
<code>gsub(r,s,t)</code>	This substitutes <code>s</code> in place of <code>r</code> globally in the string <code>t</code> and returns the number of substitutions made.
<code>match(s,r)</code>	This searches the string <code>s</code> for a substring <code>r</code> . The index of <code>r</code> is returned or zero is returned.

# Zoeken en vervangen

## Tekstbestand

```
Veerle,Asaert,PXL  
Gert,Van Waeyenberg,PXL
```

## AWK-script

```
BEGIN { FS="," }  
{ sub("PXL", " Personeel", $3); print $1,$2 $3 }
```

## Uitvoer van het script op het tekstbestand

```
Veerle Asaert, Personeel  
Gert Van Waeyenberg, Personeel
```

Function	Description
<u>-----</u>	<u>-----</u>
toupper(str)	This converts the given string into upper case.
tolower(str)	This converts the given string into lower case.
delete array [element]	This deletes the specified element of the array.
sub(r, s [,t])	This substitutes the first occurrence of the regular expression r by s in the string t. If the string t is not supplied, \$0 (entire line/record) is considered. The function returns 1 if successful and 0 otherwise.
gsub(r,s)	This substitutes s in place of r globally in \$0 (entire line/record) and returns the number of substitution made.
gsub(r,s,t)	This substitutes s in place of r globally in the string t and returns the number of substitutions made.
match(s,r)	This searches the string s for a substring r. The index of r is returned or zero is returned.

# Werken met variabelen

## Tekstbestand

```
januari 79  
februari 132  
maart 783
```

## AWK-script

```
BEGIN { printf "Totaal\n-----\n"; vmunt="Euro" }  
{ vtotaal+=$2 }  
END { print vtotaal, vmunt }
```

## Uitvoer van het script op het tekstbestand

```
Totaal  
-----  
994 Euro
```



# Werken met variabelen

```
student@ubserv:~$ awk -F: \  
'BEGIN {printf "\nShell gebruikers"} \  
$7 ~ /sh$/ { print $1; vaantal++ } \  
END { print "Aantal: " vaantal "/" NR}' \  
/etc/passwd
```

Shell gebruikers

root

student

Aantal: 2/31



# if-then-else

```
if ()  
{  
  ...  
}  
else  
{  
  ...  
}
```

## Tekstbestand

```
Veerle,Asaert,PXL,v  
Gert,Van Waeyenberg,PXL,m
```

## AWK-script

```
BEGIN { FS="," }  
{ if ($4 == "v")  
  { printf "Mevrouw "  
  }  
  else  
  { printf "Mijnheer "  
  }  
  print $1, $2  
}
```

## Uitvoer van het script op het tekstbestand

```
Mevrouw Veerle Asaert  
Mijnheer Gert Van Waeyenberg
```



# if-then-else

```
if ()  
{  
  ...  
}  
else  
{  
  ...  
}
```

## Tekstbestand

```
Veerle,Asaert,PXL,v  
Gert,Van Waeyenberg,PXL,m
```

## commandline

```
student@ubserv:~$ awk 'BEGIN { FS="," } \  
{ if ($4 == "v") printf "Mevrouw "; else printf "Mijnheer "; print $1, $2 }' \  
personeel
```

## Uitvoer van het script op het tekstbestand

```
Mevrouw Veerle Asaert  
Mijnheer Gert Van Waeyenberg
```





```
for ()
{
...
}
```

# for-loop

Inputfile: /etc/passwd

## AWK-script

```
BEGIN { FS=":" }
{
  if ( $7 ~ "sh$" )
  {
    print $1;
    for (i=2;i<=7;i++)
    {
      print "veld",i, $i
    }
    print ""
  }
}
```

Uitvoer van het script op de inputfile

```
root
veld 2 x
veld 3 0
veld 4 0
veld 5 root
veld 6 /root
veld 7 /bin/bash

student
veld 2 x
veld 3 1000
veld 4 1000
veld 5 Student PXL
veld 6 /home/student
veld 7 /bin/bash
```

```
testuser
```

```
veld 2 x
```

```
veld 3 1000
```



```
while ()
{
...
}
```

# while-loop

## Inputfile

```
student@ubserv:~$ find /etc -name "*.conf" 2> /dev/null > conffiles
```

## commandline

```
student@ubserv:~$ awk '
> BEGIN { FS="/" }
> {
>   i=2
>   while ( i <= NF )
>   {
>     printf "%s", $i
>     if ( i < NF) print ">"
>     i++
>   }
>   print " "
> }
> ' conffiles > conffiles2
```

## Na uitvoer van het script

```
student@ubserv:~$ head conffiles2
etc >
dhcp >
dhclient.conf

etc >
sysctl.d >
10-kernel-hardening.conf

etc >
sysctl.d >
```

i=2 → want regel begint al onmiddellijk met een /

# do-while-loop

```
do
{
...
} while ()
```

## Inputfile

```
student@ubserv:~$ find /etc -name "*.conf" 2> /dev/null > conffiles
```

## commandline

```
student@ubserv:~$ awk '
> BEGIN { FS="/" }
> {
>   i=3
>   print $2
>   do
>   {
>     for (j=2;j<i;j++) printf "  "
>     print "|_", $i;
>     i++
>   } while ( i <= NF )
>   print " "
> }
> ' conffiles > conffiles3
```

## Na uitvoer van het script

```
student@ubserv:~$ head conffiles3
etc
  |_ dhcp
    |_ dhclient.conf

etc
  |_ sysctl.d
    |_ 10-kernel-hardening.conf

etc
  |_ sysctl.d
```

i=3 → is eerste subdir, want regel begint al onmiddellijk met een /  
→ dus pas vanaf eerste subdir wordt er ingesprongen

# getline command

- built-in command
- om input te lezen op een andere manier
- *voor advanced users van awk*
- ook voor input vanaf keyboard:
  - `getline vIngave < "/dev/tty"`



# getline vIngave < "/dev/tty"

AWK-script: toonveld.awk

```
BEGIN {
printf "Wat is je field seperator: "
getline FS < "/dev/tty";
print "FS:", FS
}
{
printf "Welk veld wil je tonen bij de volgende lijn (<=" NF "): "
getline vIngave < "/dev/tty"
if (vIngave <= NF)
{
    print "veld vIngave:", $vIngave
}
else
{
    print "dit veld bestaat niet"
}
}
```



# getline vIngave < "/dev/tty"

```
student@ubserv:~/testdir$ ls -l
total 12
-rw-rw-r-- 1 student student 10 Dec 3 14:11 cijfers
drwxr-xr-x 4 student student 4096 Nov 15 10:54 oefperm
-rw-rw-r-- 1 student student 0 Dec 3 14:11 testfile
-rw-rw-r-- 1 student student 296 Dec 3 14:12 toonveld.awk
student@ubserv:~/testdir$ ls -l | awk -f toonveld.awk
Wat is je field separator:
FS:
Welk veld wil je tonen bij de volgende lijn (<=2): 1
veld vIngave: total
Welk veld wil je tonen bij de volgende lijn (<=9): 1
veld vIngave: -rw-rw-r--
Welk veld wil je tonen bij de volgende lijn (<=9): 3
veld vIngave: student
Welk veld wil je tonen bij de volgende lijn (<=9): 9
veld vIngave: testfile
Welk veld wil je tonen bij de volgende lijn (<=9): 12
dit veld bestaat niet
```



# getline vIngave < "/dev/tty"

```
student@ubserv:~/testdir$ awk -f toonveld.awk /etc/passwd
Wat is je field seperator: :
FS: :
Welk veld wil je tonen bij de volgende lijn (<=7): 1
veld vIngave: root
Welk veld wil je tonen bij de volgende lijn (<=7): 2
veld vIngave: x
Welk veld wil je tonen bij de volgende lijn (<=7): 6
veld vIngave: /bin
Welk veld wil je tonen bij de volgende lijn (<=7): 7
veld vIngave: /usr/sbin/nologin
Welk veld wil je tonen bij de volgende lijn (<=7): 9
dit veld bestaat niet
```



# functions

- Naast de built-in functies kan je ook gebruik maken van user defined functies
- Syntax

```
function function_name(argument1, argument2, ...) {  
    function body  
    [return (...)]  
}
```





# functions

AWK-script: testfunction.awk

```
function aFunction() {  
    print "We printen dit af voorafgaand aan de even lijnen..."  
}  
{  
    if ( NR%2 == 0 )  
    {  
        aFunction()  
    }  
    print $0  
}
```

```
student@ubserv:~/testdir$ cat cijfers  
1  
2  
3  
4  
5
```

```
student@ubserv:~/testdir$ awk -f testfunction.awk cijfers  
1  
We printen dit af voorafgaand aan de even lijnen...  
2  
3  
We printen dit af voorafgaand aan de even lijnen...  
4  
5
```



# functions

## Tekstbestand

```
Toetsenbord 450 Bfr  
Muis 120 Bfr  
Muismat 55 Bfr
```

## AWK-script

```
function Bfr2Euro (bfr) {  
    return (bfr/40.33)  
}  
BEGIN { print "Artikel          Prijs" }  
{ printf "%-15s %5.2f Euro\n", $1, Bfr2Euro($2) }
```

## Uitvoer van het script op het tekstbestand

Artikel	Prijs
Toetsenbord	11.16 Euro
Muis	2.98 Euro
Muismat	1.36 Euro



# Tussenliggende exports

## AWK-script

```
BEGIN { FS=":" }  
{ if ($7 ~ "sh$")  
  { print $0 > "LoginAccounts.txt"  
  }  
  else  
  { printf "%s \n", $0 > "ServiceAccounts.txt"  
  }  
  # print "Regel", NR, "weggeschreven"  
}
```

## LoginAccounts.txt

```
root:x:0:0:root:/root:/bin/bash  
student:x:1000:1000:,,,:/home/student:/bin/bash
```

## ServiceAccounts.txt

```
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

# ARRAYS

## Tekstbestand

```
Eerste lijn  
Tweede lijn  
Derde lijn  
Vierde lijn
```

## AWK-script

```
{ lines [NR] = $0 }  
END {  
    for(i=NR;i>0;i--)  
        print "Regel", i, "van", NR, ":", lines[i]  
}
```

## Output

```
Regel 4 van 4 : Vierde lijn  
Regel 3 van 4 : Derde lijn  
Regel 2 van 4 : Tweede lijn  
Regel 1 van 4 : Eerste lijn
```



# ARRAYS

## Tekstbestand

```
Veerle 19  
Ben 8  
Gert 19  
Kris 11  
Yves 9
```

## AWK-script

```
{  
    if ($2 < 10)  
    {  
        score["gebuisd"]++;  
    } else {  
        score["geslaagd"]++;  
    }  
}  
END {  
    printf "Aantal geslaagd: %d \nAantal gebuisd:  
%d\n", score["geslaagd"], score["gebuisd"]  
}
```

## Output

```
Aantal geslaagd: 3  
Aantal gebuisd: 2
```

