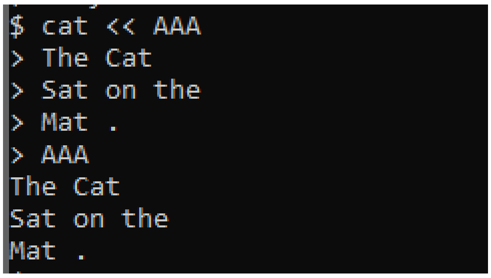
Chapter 14 - Advanced Shell Scripting

Philipp Moritzer - 21170004

Input/output redirection

- Three standard files
  - STDIN (0): keyboard
  - STDOUT (1): monitor
  - STDERR (2): monitor
- Redirection Operator
  - > : redirect STDOUT to a file
  - >> : appends STDOUT to a file
  - < : redirect STDIN to a file
  - |: sends output of a process to input of other process
  - << : Associate input stream until delimiter

```
$ cat << AAA
The Cat
Sat on the
Mat .
AAA # end string
```



```
$ cat << AAA
> The Cat
> Sat on the
> Mat .
> AAA
The Cat
Sat on the
Mat .
```

```
$ find / -name "*.txt" > out.txt 2> err.txt # error output to err.txt, nomral
output to out.txt
```

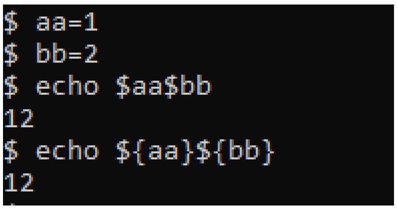
Command Substitution

- Substutites Results
- Back tick

-
  ```
  $ Lines=`wc -l textFile`
  $ a=`expr $a +1`
  ```

- Brace expansion

```
$ Lines="$(wc -l textFile)" # does not work at sh
```



```
$ aa=1
$ bb=2
$ echo $aa$bb
12
$ echo ${aa}${bb}
12
```

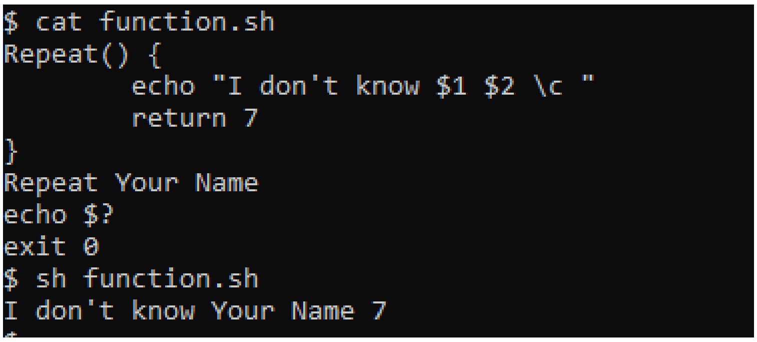
Shell variable

```
$ a=5 # sets a variable to 5
$ export a # makes a a env variable
$ export b=7 # define new env variable
$ echo $a # prints a result (5)
$ set # shows all shell variables
$ env # shows all environment variables
$ set | grep PATH # show location of PATH variable
```

Shell functions

- Function Syntax: name() {commands; }

```
Repeat() {
    echo -n "I don't know $1 $2"
    return 7
}
Repeat Your Name
echo $?
exit 0
```



```
$ cat function.sh
Repeat() {
        echo "I don't know $1 $2 \c "
        return 7
}
Repeat Your Name
echo $?
exit 0
$ sh function.sh
I don't know Your Name 7
```

Scope

- There are two types of scope
    - local & global

```
scope() {
    local lov=1 # local variable
    glov=2 # global variable
    echo local $lov global $glov
}
scope
echo local $lov global $glov
```

Output:
´local 1 global 2´
´local global 2´

Function Libraries

- Include function script file using dot(.)

```
. ./scope.sh # read from scope.sh
scope # function call
```

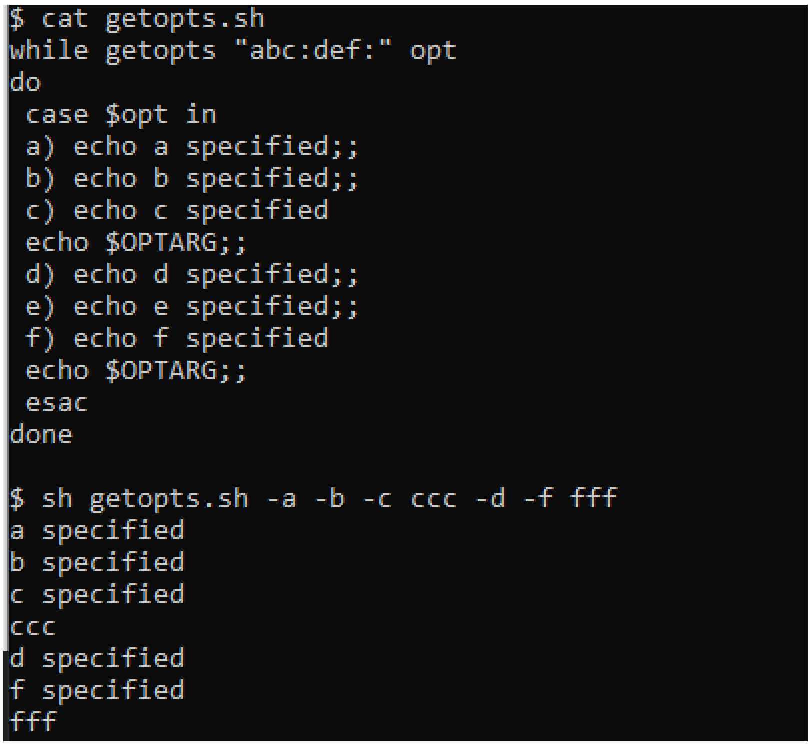
getopts

- A built-in command line parser

```
while getopts "abc:def" opt
do
    case $opt in
        a) echo a specified::
        b) echo b specified::
        c) echo c specified
           echo $OPTARG;;
        d) echo d specified::
        e) echo e specified::
        f) echo f specified
           echo $ OPTARG::
    esac
Done
```

```
$ sh getopts.sh -a -b -c cc -d -e -f ff
```

Output:



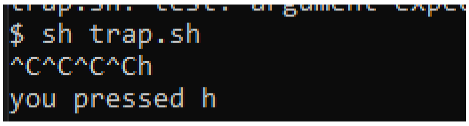
```
$ cat getopts.sh
while getopts "abc:def:" opt
do
 case $opt in
 a) echo a specified;;
 b) echo b specified;;
 c) echo c specified
 echo $OPTARG;;
 d) echo d specified;;
 e) echo e specified;;
 f) echo f specified
 echo $OPTARG;;
 esac
done

$ sh getopts.sh -a -b -c ccc -d -f fff
a specified
b specified
c specified
ccc
d specified
f specified
fff
```

Signals and Traps

```
$ trap '' 2 # single quote, ignore ^c
```

```
trap '' 2
while [ 1 = 1 ]
do
read a
if [ $a = 'a' ]
then
    exit
else
    echo you pressed $a
fi
done
```

Output:



```
trap.sh: test: argument expe
$ sh trap.sh
^C^C^C^Ch
you pressed h
```

File Handling

```
$ If [ -w writeFIle || -x writeFile ] # if file has writeable or executable
permission statement will result in true
```

```
$ If [ -r writeFile && -s writeFile ] # true if file is readable and size is greater than 0
```

Arguments:

- -d: directory
- -e: file exists
- -r: readable
- -w: writeable
- -x: executable
- -s: size is greater than zero