Excercise Sheet 2

Johannes Koch

April 3, 2018

1 Task 1

1.1 Script 1

```
1 #!/bin/bash
2 # Script 1
3 #
4 for FN in $@
5 do
6 chmod 0750 "$FN"
7 done
```

Line 1: tell the shell to use `/bin/bash` as interpreter
Line 2-3: comments
Line 4-5, 7: for loop, executes Line 6 for every given commandline argument
Line 6: change the access permissions for the current argument to `-rwxr-x--- ...`

Execution with filenames or directories as commandline arguments:
The script will change the access permissions for each valid argument or throw an error to stdout.

Execution without any arguments:
Since the for-loop will never execute the script will do nothing.

1.2 Script 2

```
1 #!/bin/bash
2 # Script 2
3 if (( $# < 3 ))
4 then
5 printf "%b" "Error. Not enough arguments.\n" >&2
6 printf "%b" "usage: myscript file1 op file2\n" >&2
7 exit 1
8 elif (( $# > 3 ))
9 then
10 printf "%b" "Error. Too many arguments.\n" >&2
11 printf "%b" "usage: myscript file1 op file2\n" >&2
12 exit 2
13 else
14 printf "%b" "Argument count correct. Proceeding...\n"
15 fi
```

Line 1: tell the shell to use `/bin/bash` as interpreter
Line 2: comment
Line 3-4: if the number of commandline arguments is less than 3, execute Lines 5-7
Line 5: print the string `"Error. ..."` to `stderr` while interpreting escape characters
Line 6: same as Line 5 with other text Line 7: exit with error code 1 (Catchall for general errors)
Line 8-9: if the number of commandline arguments is greater than 3, execute Lines 10-12
Line 10-11: same as Line 5 with other error message
Line 12: exit with error code 2 (Misuse of shell builtins(according to Bash documentation))
Line 13: if none of the above cases occur, execute Line 14
Line 14: prints "Argument count correct. Proceeding..." and a newline to `stdout`
Line 15: end the if clause

Execution with less than 3 commandline arguments:
Lines 5-6 will be printed to `stderr` and the program will exit with error code 1

Execution with more than 3 commandline arguments:
Lines 10-11 will be printed to `stderr` and the program will exit with error code 2

Execution with 3 commandline arguments:
"Argument count correct. Proceeding..." and a newline command will be printed to `stdout`

1.3 Script 3

```
1 #!/bin/bash
2 # Script 3
3 INFILE=$1
4 OUTFILE=$2
5 if [ -e "$INFILE" ]
6 then
7 if [ -w "$OUTFILE" ]
8 then
9 cat "$INFILE" >> "$OUTFILE"
10 else
11 echo "can not write to $OUTFILE"
12 fi
13 else
14 echo "can not read from $INFILE"
15 fi
```

Line 1: tell the shell to use `/bin/bash` as interpreter
Line 2: comment
Line 3: `INFILE` is now the first commandline argument
Line 4: `OUTFILE` is now the second commandline argument
Line 5-6: if the file `INFILE` exists execute Lines 7-12
Line 7-8: if the `OUTFILE` is a writeable file, execute Line 9
Line 9: append the content of `INFILE` to `OUTFILE`
Line 10: if `OUTFILE` is not a writeable file, execute Line 11
Line 11: print "can not write to " and the second argument to `stdout`
Line 12: end the inner if clause
Line 13: if `INFILE` does not exist, execute Line 14
Line 14: prints "can not read from" and the first argument to `stdout`
Line 15: end the outer if clause

Execution with at least 2 arguments, where the first one is an existing file and the second a writeable file:
The content of the first file will be appended to the second file

Execution with at least 1 argument, where the first one is an existing file:
"can not write to " and the second argument or "" will be printed to `stdout`

Execution with any number of arguments, where the first one doesn't exist:
"can not read from " and the first argument or "" will be printed to `stdout`

2 Task 2

```
#!/bin/bash
# This bash script creates a backup of the current directory
# - if the target path already exists, only newer files will be copied
# - if the target path doesn't exist, it will be created

# check the number of arguments
if (( $# != 1 )) ; then
  # trow an error if the argument is invalid
  echo "Error. Invalid number of arguments" >&2
  echo "usage: $0 <path to target directory>" >&2
  exit 1
else
  # check if the directory exists
  WD=`pwd`
  DIRNAME=`basename "$WD"`
  DIR=$1
  if [ -e "$DIR" ] ; then
    # check for each file/dir individually if it is newer
    # for FILE in `ls *` ; do
    for FILE in * ; do
      # check if the file exists
      if [ -e "$DIR/$DIRNAME/$FILE" ] ; then
        # if the file exists, check its age
        if [ "$FILE" -nt "$DIR/$DIRNAME/$FILE" ] ; then
          # overwrite without asking
          \cp -rf "$FILE" "$DIR/$DIRNAME"
        fi
      else
        cp -rf "$FILE" "$DIR/$DIRNAME"
      fi
    done
  else
    # if the dir doesn't exist, make it
    mkdir -p "$DIR" && cp -rf "$WD" "$DIR"
  fi
fi
```