BS Praktikumsaufgabe 01

Der Umgang mit Linux

Version 1.0 - Abgabe am 25. Oktober 2016

Alexander Mendel Karl-Fabian Witte

erstellt am 21. Februar 2023

In diesem Praktikum soll der Umgang mit Linux geübt werden. Dafür werden Grundprinzipien und Aufbau der Linuxschnittstellen geübt. Zum Schluss soll die `bash` noch einmal genauer mit einem Skript gelernt werden.

Inhaltsverzeichnis

1 Die Macht der Kommandozeile

Es wird kurz der Umgang mit dem Terminal geübt. Es werden hier für die Aufgeben wichtigen Informationen dokumentiert. Das mitgelaufene Logbuch `typescript`, welches mittels `script` gestartet wurde, ist im Reposetory zu finden (siehe Abschnitt 3). Zudem werden einige fundamentalen Kenntnisse für Linuxnutzer mit auf den Weg gegeben.

1. **Was ist *Tab-Expansion* und was nützt Ihnen das bei der Arbeit mit der Kommandozeile?**
   Die *Tab-Expansion* vervollständigt das angefangene Befehlswort (oder auch Verzeichnis- und Dateinamen) in der Shell-Kommandozeile. Wenn im Terminal <**Tab**> gedrückt wird, wird der angefangene String bis zur Mehrdeutigkeit vervollständigt und beim erneuten Drücken der <**Tab**> werden die Möglichkeiten aufgezeit. Das folgende Beispiel verdeutlich es:

```
$ ls -1
xy.pdf
xy.ps
$ acroread x                <- <Tab> druecken
$ acroread xy.p             <- nochmal <Tab> druecken
xy.pdf xy.ps
$ acroread xy.p
```

2. **Was erhalten Sie beim Drücken der Tastenkombination <Alt><.>?**
   <**Alt**><**.**> gibt das letzte Argument des letzten Befehls wieder. Bein erneuten drücken wird das letzte Argument der vorletzten Befehls an die Coursorposition gesetzt, und so weiter.

3. **Geben Sie das Verzeichnis nach *Erweiterung* sortiert aus.**

```
$ ls -X
```

4. **Geben Sie das Verzeichnis nach *Modifikationszeit* sortiert aus.**

```
$ ls -t
```

5. **Kehren Sie für beide Sortiervarianten die *Reihenfolge* um.**

```
$ ls -Xr
$ ls -tr
```

6. **Geben Sie das Verzeichnis nach *rekursiv* d.h. mit allen Unterverzeichnissen aus.**

```
$ ls -R
```

7. **Erläutern Sie die Operatioen von dem sort-Befehl `ls -l | sort -rnk5`.**

```
$ ls -l | sort -rnk5
------------------------
```

```
-r  : umgekehrte Reihenfolge
-n  : numerische Wertsortierung 0 < 1 < 2 ...
-k5 : die 5te Spalte (hier mit ls -l -> Dateigroesse)
```

Es wird also nach Dateingröße absteigend sortiert.

8. **Dokumentieren Sie mit `ls -l` das Resultat Ihrer Aktion.**

```
bs@linux-8b19:~/Bs_Prak> ls -l
insgesamt 16
lrwxrwxrwx 1 bs users    9 17. Okt 03:36 ltext2.txt -> text2.txt
lrwxrwxrwx 1 bs users    9 17. Okt 03:37 ltext3.txt -> text3.txt
-rw-r--r-- 1 bs users 3976 17. Okt 03:30 my_listing.txt
-rw-r--r-- 1 bs users   28 17. Okt 03:33 text04.txt
-rw-r--r-- 1 bs users   28 17. Okt 03:33 text2.txt
-rw-r--r-- 1 bs users   28 17. Okt 03:33 text3.txt
```

9. **Editieren Sie `ltext3.txt`: Wie verändert sich `text3.txt`?**
Die Änderung ist von `ltext3.txt` ist auch in `text3.txt` gespeichert worden.

10. **Was passiert, wenn Sie `ltext2.txt` löschen?**
Die Beseitigung von `ltext2.txt` hat keine Auswirkung auf text2.txt

11. **Was passiert, wenn Sie `text3.txt` löschen?**
Die Datei `ltext3.txt` ist ein symbolische Link, welcher wie eine Windows gewöhnliche Verknüpfung funtioniert: Er verzweigt nur auf die Originaldatei und enthält sonst keine Information. Da die Originaldatei nicht mehr existiert, funtioniert `ltext3.txt` auch nicht mehr. Anders wäre es mit einem harten Link (`ln text3.txt htext3.txt`). Dieser verweist auf die physikalische Adresse, wie es die Originalsdatei auch nur tut, und ist somit, von außen gesehen, eine synchronisierte Kopie.

12. **Demonstrieren Sie die Platzhalterzeichen mit eigenen Beispielen.**

```
bs@linux-8b19:~/Bs_Prak> ls
ltext3.txt my_listing.txt text04.txt text2.txt
text04.txt text2.txt
bs@linux-8b19:~/Bs_Prak> ls *ext?.txt
ltext3.txt text2.txt
bs@linux-8b19:~/Bs_Prak> ls [ml]*
ltext3.txt my_listing.txt
bs@linux-8b19:~/Bs_Prak> ls [k-n]*
ltext3.txt my_listing.txt
```

13. **Finden Sie heraus, was die Wirkung der Zeichen $, ^ und \< ist. Warum musste der Suchausdruck im letzten Beispiel in Anführungszeichen (*quotes*) gesetzt werden?**

```
$ ls -l /etc/ | grep "\<fs"
# Es werden alle Zeilen ausgegeben, in denen Namen/Wörter vorkommen,
# die mit "fs" beginnen.

$ ls -1 /etc/ | grep ^fs
# Es werden alle Zeilen ausgegeben,  mit "fs" beginnen.

$ ls -l /etc/ | grep fs$
# Es werden alle Zeilen ausgegeben, die mit fs enden.
```

14. **Geben Sie alle Prozesse aus, deren Kommandozeile mit k beginnt.**

```
ps aux | grep ^k
# oder wenn man de Befehle will
# und nicht einen usernamen hat der mit k anfaengt
ps aux | grep "\<fs"
```

2 Vorgänge automatisieren: Shellskripte

Nun soll die Macht eines Shellskriptes kennen gelernt werden. Dafür wird erst ein gegebenes Skript analysiert. Danach soll dann eins selber geschrieben werden, was die Anzahl der Datein in einem Verzeichnis zurück gibt. Dabei wird die `bash` verwendet und gehofft, dass der Interpreter jede Zeile auch übersetzten kann.

1. **Was tut das oben angegebene Shellskript?**
   Es fragt nach den Anwender an seinem Namen und und gibt dann in einem `here file` eine schmucke ASCII-Begrüßung auf dem Terminal aus. Zudem kann man mit dem Flag `-h` oder `-help` eine Kurzanleitung sich ausgeben lassen. Alle weiteren Argumente werden nicht geduldet.

2. **Wie bekommen Sie heraus, welche Version des C-Compilers `gcc` auf Ihrer virtuellen Maschine installiert ist?**
   `gcc` ist ein ganz besonders gesittets Programm:

   ```
   $ gcc --version
   ```

3. **Schreiben Sie ein Shellskript `filecount.sh`, das die Anzahl von Dateien der als Option spezifizierten Typen ausgibt. Mit der Option `-h` oder `-help` soll es diesen Hilfetext ausgeben und damit gleichzeitig erklären, was genau Sie implementieren sollen.**

   ```
   cat<<EOF
     usage :
       $0 [ OPTIONS ] DIRECTORY

         Print the number of files in DIRECTORY
         of types given in OPTIONS

         Default: Print numbers of regular files
         in current directory

     OPTIONS :
     FILE type options:
       -f --regular-file  regular files
       -l --symlink       symbolic links
       -D --device        character and block devices
       -d --directory     directories
       -a --all           Include hidden files
   ```

```
  Other options:
    -h --help          Display this text
    -e --echo          Print selected files
    -v --verbose       print debugging messages
    -V --version       print Versionsnumber
EOF
```

Grundidee:

```
ls -l | egrep ^$TYPE | wc -l
# for echo
ls -l | egrep ^$TYPE | tee ...
```

2.1 filecount.sh - der Entwurf

Die Grundidee wurde uns ja schon gegeben. Beim Aufruf von `ls -l` ist der erste Character in einer Zeile ein Bezeichner `$TYPE` für den `file type`. Diese Zeilen ziehen wir mit `egrep ^$TYPE` heraus und lassen anschließend die Zeilen mit `wc -l` zählen.

Die Optionsflag werden mit Hilfe der Funtion `getopt` herausgefiltert und in einer while-Schleife eingebetteten Fallunterscheidung (`case`) die entsprechenden Optionen in dieser verarbeitet. Dabei ermöglicht `getopt` das Zusammenfassen der sogenannten *short options* ohne diese Strings explizit in der Fallunterscheidung mit jeweils einem Fall zu unterscheiden.

Die `file type opions` werden so behandelt, dass wir jeweils den `file type character` hinten an den zu Anfang leeren `TYPES`- String anhängen. Die Optionen `-help` und `-version` geben einen entsprechenden Text im Terminal aus und beenden das Skript. Für `-verbose` und `-echo` werden Flags gesetzt, die mit If-Blöcken entsprechende Extrafunktionen aufrufen. Beipiel:

```
if [ VERBOSE -eq 1 ]; then
  echo "verbose mode activ"
fi
```

Wir haben uns entschlossen haben, dass die type-flags sich nicht gegenseitig ausschließen und man so eine Komposition an unteschiedlichsten Typen zählen lassen kann, werden wird der String `TYPES` wie folgt nach den Regeln der Regulären Ausdrücke verweden:

```
ls -l | egrep ^[$TYPES] | wc -l
```

Wenn das all-Flag gesetzt wird, wird in zum einen in den `TYPES`-String nicht angehängt, sondern einfach alle Möglichkeiten der *file types* geschrieben. (siehe `info ls`) Zudem, um auch versteckte Datein anzeigen zu lassen, müssen die ls-Option mit `-A` erweitert werden. Somit haben wir für diese Flags auch eine Variable `LSFLAGS` erstellt, um diese bei `-a` entsprechend zu ändern. (`ls -A` is almost-all ist, welche die hidden files darstellt, jedoch das current und parent directory nicht mitauflistet.)

Sind alle Optionen angearbeitet, bleibt Dank `getopt` das nicht-Options- Argument stehen. Dieses fragen wir zunächst ab, ob es überhaupt ein gibt. Wenn keins übergeben wurde, bleibt die Variable `DIRECTORY` auf dem Defaultwert `'.'`. Wenn es ein Argument gibt, so wird getestet, ob es sich auch um ein `directory` handle. Wenn ja, wird dieses in `DIRECTORY` gespeichert. Ansonsten wird eine Fehlermeldung ausgegeben. Es wird nur ein directory akzeptiert, sodass ein weiters Argument auch zum Abbruch führt. Ist nichts in den leeren String `TYPES` bei der Falunterscheidungsschleife geschrieben worden, so wird dieser mit dem default für reguläre Files `char '-'` gefüllt. Zum Schluss wird die eigendliche Funktion mit dem Aufruf `calc` (eigene Funktion dieses Skripts) gestartet.

Wir haben uns entschlossen, die Funktion `tee` anders zu verwenden: als einfaches entfärbendes Rohr, anstatt eines T-Stücks. Die Alternative dazu wird hier einmal vereinfacht dargestellt:

```
if [ $ECHO -eq 1 ]; then                    # echo mode
  ls $LSFLAG | egrep ^[$TYPES] | tee | wc -l
else                                        # no echo mode
  ls $LSFLAG | egrep ^[$TYPES] | wc -l
fi
```

2.2 filecount.sh - der Code

```
1  #!/bin/bash
2  # filecount.sh
3  # Count the number of spacific file types in a directory
4
5  AUTHOR="Alex and Kalle"
6  VERSION="1.1"
7  UPDATE="20.10.2016"
8
9  #initialyse / default
10 # flags
11 VERBOSE=0
12 ECHOMODE=0
```

```
13 # vars
14 TYPES=''
15 LSOPTION="-l"
16 DIRECTORY="."
17
18 # Constants for getopt
19 SHORT_OPT=hVveflDda
20 LONG_OPT="help,version,verbose,echo,regular-file,symlink,device,directory,all"
21
22 #-------------------------------------------------------------------------------
23 usage()
24 {
25 cat<<EOF
26 usage :
27   $0 [ OPTIONS ] DIRECTORY
28
29     Print the number of files in DIRECTORY
30     of types given in OPTIONS
31
32     Default: Print numbers of regular files
33     in current directory
34
35 OPTIONS :
36 FILE type options:
37     -f --regular-file regular files
38     -l --symlink      symbolic links
39     -D --device       character and block devices
40     -d --directory    directories
41     -a --all          Include hidden files
42
43 Other options:
44     -h --help     Display this text
45     -e --echo     Print selected files
46     -v --verbose  print debugging messages
47     -V --version  print Versionsnumber
48 EOF
49 }
50 #-------------------------------------------------------------------------------
51 #sehr gesittetes Programm
52 version()
53 {
54 cat <<EOF
55 $0:
56 Version $VERSION, last update $DATE
57 coded by $AUTHOR
58 EOF
59 }
60 #-------------------------------------------------------------------------------
```

```
61  verbose_mode()
62  {
63      VERBOSE=1
64      echo -e "verbose mode activ: print debugging messages\n";
65  }
66  #--------------------------------------------------------------------------
67  calc()
68  {
69      if [ $ECHOMODE -eq 1 ]; then                              # ECHO MODE START
70          if [ $VERBOSE -ne 0 ]; then
71              echo -e "Choosen files will be printed. \n";
72          fi
73          ls $LSOPTION $DIRECTORY | egrep "^[$TYPES]" | tee      # ECHO MODE END
74      else
75          if [ $VERBOSE -ne 0 ]; then
76              echo -e "Choosen files will be summed. \n";
77          fi
78      fi
79      # Here is the magic
80      ls $LSOPTION $DIRECTORY | egrep "^[$TYPES]" | wc -l
81  }
82  #--------------------------------------------------------------------------
83  # ########################################################################## #
84  #                                  main                                      #
85  # ########################################################################## #
86
87  # define getopt
88  ARGS=$(getopt -o $SHORT_OPT -l $LONG_OPT -n "filecount.sh" -- "$@");
89
90  # execute getopt with eval
91  eval set -- "$ARGS";
92
93  while true; do
94      case "$1" in              # ---------------------------------- CASE beginn
95          "-h" | "--help")                                              # HELP
96              usage
97              exit 0;
98          ;;
99          "-V" | "--version")                                       # VERSION
100             version
101             exit 0;
102         ;;
103         "-v" | "--verbose")                                       # VERBOSE
104             verbose_mode
105             shift
106         ;;
107         "-e" | "--echo")                                              # ECHO
108             ECHOMODE=1
```

```
109             if [ $VERBOSE -ne 0 ]; then
110                 echo -e "echo mode activ.\n";
111             fi
112             shift
113         ;;
114         "-f" | "--regular-file")                                    # REGULAR FILES
115             TYPES=$TYPES'-'
116             if [ $VERBOSE -ne 0 ]; then
117                 echo -e "Option \"-f\" set";
118                 echo -e "\"Regular files\" get counted.\n";
119             fi
120             shift
121         ;;
122         "-l" | "--symlink")                                          # SymLink
123             TYPES=$TYPES'l'
124             if [ $VERBOSE -ne 0 ]; then
125                 echo -e "Option \"-l\" set";
126                 echo -e " \"symbolic links\" get counted.\n";
127             fi
128             shift
129         ;;
130         "-D" | "--device")                                           # DEVICES
131             TYPES=$TYPES"cb" #set character and block devices
132             if [ $VERBOSE -ne 0 ]; then
133                 echo -e "Option \"-D\" set";
134                 echo -e "\"Character and block devices\" get counted.\n";
135             fi
136             shift
137         ;;
138         "-d" | "--directory")                                      # DIRECTORIES
139             TYPES=$TYPES'd'
140             if [ $VERBOSE -ne 0 ]; then
141                 echo -e "Option \"-d\" set";
142                 echo -e "\" Directories\" get countet. \n";
143             fi
144             shift
145         ;;
146         "-a" | "--all")                                                # ALL
147             LSOPTION="-lA"        # almost all flags for all except .. .
148             TYPES="-bcCdDlMnpPs?" # asterix doesnt work in a box
149             if [ $VERBOSE -ne 0 ]; then
150                 echo -e "Option \"-a\" set"
151                 echo -e "\"All files (including hidden)\" get counted.\n";
152             fi
153             shift;
154         ;;
155         --)                                                         # OPTGET END
156             shift;
```

```
157             break;
158     esac
159 done                                   # ----------------------OPTION CASE end
160
161 if [ ! -z $1  ]; then # is there anything else ?
162 # anything else should be directory
163   if [ -d $1 ]; then    # test if it is an valid
164       DIRECTORY=$1
165   else
166     echo -e "\"$1\" is not a driectory";
167     echo -e "type \"$0 --help\" for more information\n";
168     exit 1;
169   fi
170   if [ $2 ]; then    # is there an other argument
171     echo -e "too many arguments"
172     echo -e "type \"$0 --help\" for more information\n"
173     exit 1;
174   fi
175 fi
176
177 if [ -z $TYPES ]; then  # is the length of TYPE 0 (empty)?
178     TYPES='-';
179 fi
180
181 # This is just for debug purpose
182 if [ $VERBOSE -ne 0 ]; then
183     echo -e "Arguments for calc are:";
184     echo -e "DIRECTORY: $DIRECTORY";
185     echo -e "VERBOSE:  $VERBOSE   TYPES:    $TYPES"
186     echo -e "ECHOMODE: $ECHOMODE  LSOPTION: $LSOPTION \n"
187 fi
188 calc                                                    # START THE Calculation
189 exit 0
```

Listing 1: filecount.sh : das komplette script

3 Quellcode

In dem angehängten Verzeichnis ist das Shellskript `filecount.sh` und die die Logdatei `typescript`. Der komplette Quellcode zu den Aufgaben sowie zu diesem Dokument kann im Repository unter folgender URL betrachtet werden:
https://bitbucket.org/bibutNkafawi/a01_der_umgang_mit_linux