Der Umgang mit Linux

1 Häufige verwendete Befehle der Kommandozeile

In dieser Aufgabe werden Befehle der *Kommandozeile* eingesetzt. Häufig gibt es alternative grafische Werkzeuge, die dasselbe tun.

Aufgabe: Im Netz sind eine Reihe von Anleitungen zur Shell-Programmierung verfügbar. Arbeiten Sie eine einführende Anleitung zur bash Shell durch, bevor Sie diese Aufgabe bearbeiten. (z.B.: `http://de.wikibooks.org/wiki/Linux-Kompendium:_Shellprogrammierung`)

In diesen Boxen finden Sie kleine Aufgaben. Dokumentieren Sie die Ergebnisse dieser Aufgaben so, dass Sie die Aufgaben auf dieser Basis problemlos erneut ausführen können.

Hinweis: Bitte arbeiten Sie auf dem Shared Folder, so dass Sie die Daten auf Ihrem Z Laufwerk finden. Die Daten der USB Festplatten, auf denen die Linux VMs liegen, werden nicht sichert.

1. 1 Hilfe!

Die Programme `info, man, apropos` bieten Hilfe zu (fast) allen Konsolenprogrammen. Noch mehr Information steht im Verzeichnis `/usr/share/doc`.

Hier einige Links zu Linux Befehlssammlungen:
`http://files.fosswire.com/wpu/2007/08/fwunixref.pdf`
`http://www.pc-erfahrung.de/linux/linux-befehle.html`
`http://www.jux-net.info/jux2/docs/sys100/index.html`

1. 2 script: Aktionen aufzeichnen

Mit diesem Befehl werden die Ein- und Ausgaben des Terminals in einer Datei mitprotokolliert.

1. 3 mkdir: Erzeuge Verzeichnis

`mkdir` erzeugt in Ihrem Homeverzeichnis das Verzeichnis `Bs_Prakt`.

```
franzkorf@linux-146z:~> mkdir ~/Bs_Prakt
franzkorf@linux-146z:~>
```

Das Zeichen `~` steht für das Heimatverzeichnis. In diesem Beispiel hätte man auch tippen können: `mkdir /home/bs/Bs_Prakt`.

Beachte: Unix-Filesysteme unterscheiden Groß- und Kleinschreibung.

Aufgabe:

- Zeichnen Sie alles auf, was Sie ab jetzt tun.
- Was ist *Tab-Expansion* und was nützt Ihnen das bei der Arbeit mit der Kommandozeile?
- Was erhalten Sie beim Drücken der Tastenkombination <Alt><.>?

1. 4 cd: Verzeichnis wechseln

```
franzkorf@linux-146z:~> cd ~/Bs_Prakt
franzkorf@linux-146z:~/Bs_Prakt>
```

1. 5 pwd: Wo bin ich in Verzeichnisbaum?

```
franzkorf@linux-146z:~> pwd
/home/franzkorf
franzkorf@linux-146z:~>
```

1. 6 whoami: Wer bin ich?

```
franzkorf@linux-146z:~> whoami
franzkorf
franzkorf@linux-146z:~>
```

1. 7 ls: Verzeichnisinhalt anzeigen

Bitte machen Sie sich mit den Optionen von diesem Befehl vertraut. Führen Sie folgenden Befehl aus:

```
franzkorf@linux-146z:~>
franzkorf@linux-146z:~> ls -l /etc
total 2296
-rw-r--r--  1 root root     15161 2010-07-05 22:01 a2ps.cfg
-rw-r--r--  1 root root      2565 2010-07-05 22:01 a2ps-site.cfg
-rw-r--r--  1 root root        25 2010-07-01 17:27 aclocal_dirlist
drwxr-xr-x  3 root root      4096 2010-07-06 08:13 acpi
-rw-r--r--  1 root root        44 2011-01-20 19:47 adjtime
... usw.
franzkorf@linux-146z:~>
```

Aufgabe:

- Geben Sie das Verzeichnis nach *Erweiterung* sortiert aus.
- Geben Sie das Verzeichnis nach *Modifikationszeit* sortiert aus.
- Kehren Sie für beide Sortiervarianten die *Reihenfolge* um.
- Geben Sie das Verzeichnis *rekursiv*, (d.h. mit allen Unterverzeichnissen) aus.

1. 8 Unix-Verzeichnisstruktur

/bin	Ausführbare Systemprogramme
/boot	Kernel
/dev	Geräte-Pseudodateien
/etc	Konfigurationsdaten
/home	Benutzerdaten
/lib	System-Bibliotheken und Kernelmodule
/lost+found	Fundsachen nach Dateisystemcheck
/media	Externe Datenträger
/mnt	Temporär eingehängte Datenträger
/proc	Pseudo-Dateisystem mit Prozess- (und zur Zeit noch Kernel-informationen – siehe sys)
/root	Heimatverzeichnis des Benutzers root
/sbin	Administrative Systemprogramme
/srv	Dateien für Server-Dienste
/sys	Kernel- und sonstige Systeminformationen
/tmp	Temporäre Dateien (werden beim Herunterfahren des Systems gelöscht)
/usr	Unix System Resources
/usr/bin	Anwendungsprogramme
/usr/lib	Anwendungsbibliotheken
/usr/share	Daten für Anwendungsprogramme
/usr/share/doc	Dokumentation
/usr/local	Dieselbe Unterverzeichnisstruktur noch mal für selbst installierte Programme
/var	Veränderliche Daten
/var/log	System-Protokolldateien

1. 9 less, more, cat: Textdateien anzeigen

```
franzkorf@linux-146z:~/Bs_Prakt> cat ~/.bashrc
# Sample .bashrc for SuSE Linux
# Copyright (c) SuSE GmbH Nuernberg

# There are 3 different types of shells in bash: the login shell, normal shell
# and interactive shell. Login shells read ~/.profile and interactive shells
```

```
# read ~/.bashrc; in our setup, /etc/profile sources ~/.bashrc - thus all
# settings made here will also take effect in a login shell.
#
# NOTE: It is recommended to make language settings in ~/.profile rather than
# here, since multilingual X sessions would not work properly if LANG is over-
# ridden in every subshell.

... usw.

franzkorf@linux-146z:~/Bs_Prakt>
```

Das ist OK für kurze Dateien. Für längere Dateien nimmt man den Pager `less` oder den einfachen Standard-Pager `more`.

1. 10 E/A-Umleitung und Pipes

Man kann die Standardausgabe eines Programms von der Konsole in eine Datei umleiten:

```
ls /etc > my_listing.txt
```

Wenn ich die Ausgabe an eine Datei *anhänngen* will, verwende ich den Operator `>>`:

```
ls /bin >> my_listing.txt
```

Auf dieselbe Weise kann man die Standard*eingabe* umleiten:

```
sort -r < my_listing.txt
```

Mit dem Operator `<<` kann man sogenannte *Here-files* erzeugen:

```
cat <<EOF
> Saemtlicher Text, der hier steht,
> wird ausgegeben, bis eine Zeile kommt,
> in der EOF (oder was auch immer oben angegeben wurde)
> am Beginn der Zeile steht
> EOF
Saemtlicher Text, der hier steht,
wird ausgegeben, bis eine Zeile kommt,
in der EOF (oder was auch immer oben angegeben wurde)
am Beginn der Zeile steht
```

Das verwendet man gerne in Skript-Dateien um längere Texte auszugeben, in Abschnitt 2 finden Sie diese Technik wieder.

Man kann die Standardeingabe eines Programms mit Hilfe des *Pipe-Symbols* `|` mit der Standardausgabe eines anderen Programms verbinden:

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

Aufgabe: Erläutern Sie die in diesem Beispiel verwendeten Optionen von sort.

1. 11 cp: Kopieren, mv: Verschieben, ln: Verlinken, rm: Löschen

Wir erzeugen eine kleine Textdatei:

```
cat << EOF > text1.txt
Hallo, dies
ist etwas
Text!
EOF
```

Mit dem Befehl `cp` erzeugen wir Kopien:

```
cp text1.txt text2.txt
cp text1.txt text3.txt
cp text1.txt text4.txt
```

Mit dem Befehl `mv` verschieben wir Dateien bzw. benennen sie um:

```
mv text4.txt text04.txt
```

Mit dem Befehl `ln -s` erzeugen wir einen *symbolischen Link* auf eine Datei:

```
ln -s text2.txt ltext2.txt
ln -s text3.txt ltext3.txt
```

Mit dem Befehl `rm` löschen Sie eine Datei

```
rm text01.txt
```

Aufgabe:

- Dokumentieren Sie mit `ls -l` das Resultat Ihrer Aktionen
- Editieren Sie `ltext3.txt`: Wie verändert sich `text3.txt`?
- Was passiert, wenn Sie `ltext2.txt` löschen?
- Was passiert, wenn Sie `text3.txt` löschen?

1. 12 Shell-Sonderzeichen

Bei der Angabe von Pfadnamen (Dateinamen) können Sie Sonderzeichen verwenden:

Zeichen	Bedeutung	Beispiel
`*`	beliebiger Text („*“ entspricht dem Zeichen *)	`ls ~/Vorlesg/*/Skript`
`?`	Ein beliebiges Zeichen („\?“ entspricht dem Zeichen ?)	`ls text?.txt`
`[aei]`	Auswahl von Zeichen (hier: a, e, i)	`ls text[23].txt`
`[a-z]`	Bereich von Zeichen (hier: Kleinbuchstaben)	`ls version[1-4].c`

Aufgabe: Demonstrieren Sie die Platzhalterzeichen mit eigenen Beispielen.

1. 13 grep: Nach Text suchen

`grep` ist ein mächtiges Werkzeug zur Suche in Texten:

```
grep EDITOR .bashrc
# Some applications read the EDITOR variable to determine your favourite text
#export EDITOR=/usr/bin/vim
#export EDITOR=/usr/bin/mcedit
```

grep wird gerne in Kombination mit Pipes verwendet:

```
wolfo@waas:shared$ ls /etc/ | grep fs
ffserver.conf
fstab
fstab~
fstab-2009-02-16
fstab-2009-02-24
gnome-vfs-2.0
gnome-vfs-mime-magic
initramfs-tools
login.defs
mke2fs.conf
wolfo@waas:shared$ ls /etc/ | grep fs$
login.defs
wolfo@waas:shared$ ls /etc/ | grep ^fs
fstab
fstab~
fstab-2009-02-16
fstab-2009-02-24
wolfo@waas:shared$ ls -l /etc/ | grep ^fs
wolfo@waas:shared$ ls -l /etc/ | grep "\<fs"
-rw-r--r--  1 root root     1999 16. Sep 19:46 fstab
-rw-r--r--  1 root root     1996 16. Sep 19:43 fstab~
-rw-r--r--  1 root root      521 16. Feb 2009  fstab-2009-02-16
-rw-r--r--  1 root root      687 24. Feb 2009  fstab-2009-02-24
```

In dem oberen Listing sehen Sie ein paar einfache Beispiele von *regulären Ausdrücken*, die grep so mächtig machen.

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

1. 14 ps, pstree: Laufende Prozesse anzeigen

```
franzkorf@linux-146z:~/Bs_Prakt> ps
  PID TTY          TIME CMD
12339 pts/0    00:00:00 bash
24317 pts/0    00:00:00 ps
franzkorf@linux-146z:~/Bs_Prakt> pstree
  ...
franzkorf@linux-146z:~/Bs_Prakt>
```

Probieren Sie die Optionen `ps a` und `ps aux`.

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

1. 15 htop, top: Interaktive Prozessanzeige

Mit `top` oder `htop` können Sie interaktiv die Prozessliste durchstöbern. `info htop` erzählt Ihnen mehr.

Hinweis: Das Programm htop müssen Sie erst installieren:

```
sudo zypper install htop
```

1. 16 Ausführen eigener Programme

Dazu müssen wir erst mal ein eigenes Programm haben. Hier die minimalistische Version:

```
fohl@FOHL-PC:tmp$ cat << EOF > hello.c
> #include<stdio.h>
> int main (void)
> {
>   printf("Hallo BS-Praktikum!\n");
>
>   return (0);
> }
> EOF
```

Wir prüfen, ob die Programmdatei richtig erzeugt wurde:

```
fohl@FOHL-PC:tmp$ cat hello.c
#include<stdio.h>
int main (void)
{
  printf("Hallo BS-Praktikum!\n");

  return (0);
}
```

Nun kompilieren wir das Programm `hello`:

```
fohl@FOHL-PC:tmp$ make hello
cc     hello.c   -o hello
fohl@FOHL-PC:tmp$ ls -l hell*
-rwxr-xr-x 1 fohl fohl 6598 29. Sep 15:27 hello
-rw-r--r-- 1 fohl fohl   88 29. Sep 15:26 hello.c
fohl@FOHL-PC:tmp$ ./hello
Hallo BS-Praktikum!
```

Das Listing zeigte uns durch die Buchstaben `x`, dass `hello` ausführbar ist.

Hinweis: In der Unix-Welt ist aus Sicherheitsgründen das aktuelle Verzeichnis *nicht* im Pfad der ausführbaren Programme!

1. 17 Diagnosetool ldd: Welche Bibliotheken benötigt ein Programm?

```
fohl@FOHL-PC:tmp$ ldd hello
        linux-gate.so.1 =>  (0xb8085000)
        libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb7f03000)
        /lib/ld-linux.so.2 (0xb8086000)
```

1. 18 Diagnosetool strace: Systemaufrufe protokollieren

Das Zauberwerkzeug zum Lokalisieren von Problemen:

```
fohl@FOHL-PC:tmp$ strace ./hello
execve("./hello", ["./hello"], [/* 54 vars */]) = 0
brk(0)                                  = 0x83b4000

... Jede Menge Text ...

write(1, "Hallo BS-Praktikum!\n"..., 20Hallo BS-Praktikum!
) = 20
exit_group(0)                           = ?
```

1. 19 Diagnosetool dmesg: Kernel-Meldungen der aktuellen Sitzung ansehen

Mitunter ist es interessant, die aktuellen Meldungen des Kernels anzusehen. Diese Meldungen schreibt der Kernel auf einem normal konfigurierten System in die Datei `/var/log/messages`. Ein direktes Auslesen z.B. mit `less /var/log/messages` scheitert, weil nur root Leserechte hat. Normaluser verwenden den Befehl `dmesg`, der die Kernelmeldungen der aktuellen Sitzung ausgibt (Tipp: Ausgabe durch `less` pipen). Will man nur die neuesten Meldungen haben, *diese aber fortlaufend*, gibt man ein:

```
dmesg | tail -f
```

Beenden der Anzeige mit <Strg><C>

Aufgabe: Beantworten Sie folgende Fragen:

- Was ist der Unterschied zwischen einer Shell- und einer Umgebungsvariablen (environment variable)?
- Welche Information enthalten die Umgebungsvariablen $HOME, $PATH, $UID und $USER?
- Was bewirkt der Befehl `cd \$HOME`? Gibt es eine einfachere Alternative?
- Welche Funktion hat die TAB-Rechts Taste bei der Eingabe eines nicht vollsttändigen Dateinamens oder eines nicht vollständigen Programmnamens?
- Welche Funktionen haben die Tasten <Pfeil-oben> und <Pfeil-unter>, wenn noch kein Befehl eingegeben wurde?
- Welche Funktion hat der history Befehl?
- Was ist die Funktion der `.bashrc` Datei im Verzeichnis`\$HOME`?
- Modifizieren Sie die Umgebungsvariable `PATH` so, dass ein Programm zuerst im aktuellen Verzeichnis gesucht wird.

2 Vorgänge automatisieren: Shellskripte

Hier ist das Gerüst eines Shellskripts:

```
#!/bin/bash
# The third shell script
# <Your name>
# <Date>

usage()
{
cat <<EOF
  $0  [OPTIONS]
  Asking the user for her or his name and display a greeting
  OPTIONS:
  -h    --help          Display this help
EOF
}
# ----------------------------------------------------------
ask_for_name()
{
    echo "Please enter your name:"
    read user_name
}
# ##########################################################
#                    main

# check for options
## note the blanks after '[' and before ']'
if [ $# -lt 1 ]; then
    # No option, so ask for name
    ask_for_name
    # display greeting
cat <<EOF

####################
  Hello $user_name,
  nice to meet you!
####################

EOF
else
    # at least 1 arg, let's check it
    case $1 in
        "-h" | "--help")     # the only valid arg
        usage
        ;;
        *)                   # anything else is not valid
        echo "Invalid option"
    esac
fi

exit 0
```

Hinweis: Ein gesittetes Programm oder Shellskript gibt bei Aufruf mit den Optionen `-h` oder `--help` einen *Hilfetext* aus.

Es gibt (zumindest in der GNU-Welt) *Kurzoptionen*, die aus *einem Strich* und *einem Buchstaben* bestehen, und *Langoptionen*, die aus *zwei Strichen* und *einem ganzen Wort* bestehen. Die Kurzoptionen können zusammengefasst werden:
`tar -x -z -v -f ~/archiv.tgz` ist dasselbe wie `tar -xzvf ~/archiv.tgz`
Ein ganz besonders gesittetes Programm reagiert auf die Option `--version` mit der Ausgabe von Versionsinformationen

Aufgabe:

- Was tut das oben angegebene Shellskript?
- Wie bekommen Sie heraus, welche Version des C-Compilers `gcc` auf Ihrer virtuellen Maschine installiert ist?
- Schreiben Sie ein Shellskript `splitfix.sh`, das das tut, was folgender Hilfetext verspricht:

```
./splitfix.sh --help
  splitfix.sh [OPTIONS] FILE [FILE ...] Split FILE into
     fixed-size pieces.
     The pieces are 10 lines long,
        if FILE is a text file.
     The pieces are 10 kiB long,
        if FILE is *not* a text file.
     The last piece may be smaller, it contains the rest
        of the original file.
     The output files bear the name of the input file
        with a 4-digit numerical suffix.
     The original file can be reconstructed with
        the command    ``cat FILE.*''

  EXAMPLE:
      splitfix.sh foo.pdf
        splits foo.pdf into the files
        foo.pdf.0000 foo.pdf.0001 etc.

  splitfix.sh [-h | --help] This help text
  splitfix.sh --version     Print version number

  OPTIONS:
  -h
    --help    this help text

  -s SIZE     size of the pieces
                in lines (for text files)
                or in kiBytes (for other files)

  -v
    --verbose print debugging messages
```

Hinweis: Für diese Aufgabe benötigen Sie die Programme

  - `shift`, um bei der Bearbeitung der Eingabeargumente das nächste Argument zum Argument `\$1` zu machen. (Das ist genaugenommen kein Programm, sondern ein eingebauter Shell-Befehl, deshalb finden Sie die Hilfe dazu auch nicht mit `shift --help`, sondern mit `help shift`).
  - `split`, um das Aufteilen der Files tatsächlich zu machen.
  - `file`, um die Eigenschaften der bearbeiteten Files zu ermitteln,
  - `grep`, um die Ausgabe des `file`-Befehls zu untersuchen,

Viel Erfolg und viel Spaß!