Studium neben dem Beruf

FOM Hochschule für Oekonomie & Management
University of Applied Sciences

Betriebssysteme Praxis

SS 2011

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.

Foliensatz C (29.04.2011)
Topic 103: GNU and Unix commands

Überblick LPIC-1, Prüfung 101

Topic 101: System Architecture

- 101.1 Determine and configure hardware settings
- 101.2 Boot the system
- 101.3 Change runlevels and shutdown or reboot system

Topic 102: Linux Installation and Package Management

- 102.1 Design hard disk layout
- 102.2 Install a boot manager
- 102.3 Manage shared libraries
- 102.4 Use Debian package management
- 102.5 Use RPM and YUM package management

Topic 103: GNU and Unix Commands

- 103.1 Work on the command line
- 103.2 Process text streams using filters
- 103.3 Perform basic file management
- 103.4 Use streams, pipes and redirects
- 103.5 Create, monitor and kill processes
- 103.6 Modify process execution priorities
- 103.7 Search text files using regular expressions
- 103.8 Perform basic file editing operations using vi

Überblick LPIC-1, Prüfung 101

Topic 104: Devices, Linux Filesystems, Filesystem Hierarchy Standard

- 104.1 Create partitions and filesystems
- 104.2 Maintain the integrity of filesystems
- 104.3 Control mounting and unmounting of filesystems
- 104.4 Manage disk quotas
- 104.5 Manage file permissions and ownership
- 104.6 Create and change hard and symbolic links
- 104.7 Find system files and place files in the correct location

Quelle: http://www.lpi.org/eng/certification/the_lpic_program/lpic_1/exam_101_detailed_objectives

Topic 103: GNU and Unix Commands

103.1 Work on the command line

Description: Candidates should be able to interact with shells and commands using the command line. The objective assumes the bash shell.

Key Knowledge Areas:

- Use single shell commands and one line command sequences to perform basic tasks on the command line.
- Use and modify the shell environment including defining, referencing and exporting environment variables.
- Use and edit command history.
- Invoke commands inside and outside the defined path.

The following is a partial list of the used files, terms and utilities: . , bash, echo, env, exec, export, pwd, set, unset, man, uname, history

Topic 103: GNU and Unix Commands

103.2 Process text streams using filters

Description: Candidates should should be able to apply filters to text streams.

Key Knowledge Areas

- Send text files and output streams through text utility filters to modify the output using standard UNIX commands found in the GNU textutils package.

The following is a partial list of the used files, terms and utilities:
cat, cut, expand, fmt, head, od, join, nl, paste, pr, sed, sort, split, tail, tr, unexpand, uniq, wc

103.1 Arbeiten in der Shell

Aus Einführung und Übungsblatt 1 bereits bekannt:

- pwd: aktuelles (Arbeits-) Verzeichnis anzeigen
- cd: Verzeichniswechsel
- .. : nächst höheres Verzeichnis
- ls: Verzeichnisinhalt anzeigen
- cp: Datei kopieren
- vi: Text-Editor
- mkdir: Verzeichnis erzeugen
- rmdir: Verzeichnis löschen
- rm: Datei löschen
- rm -r: Verzeichnis rekursiv löschen
- touch: Datei (leer) erzeugen; Zugriffsdatum aktualisieren
- less: Datei anzeigen
- grep: Suchen in Datei
- head, tail: Anfang und Ende einer Datei
- man: Hilfe anzeigen
- dmesg: Systemmeldungen ausgeben
- wc: word count
- shutdown: System runter fahren

103.1: Shell-Variablen (1)

- Die Shell (und auch andere Programme) nutzen **Umgebungsvariablen** (für Optionen, Einstellungen etc.)
- „set“ gibt eine Liste aller in dieser Shell gesetzten Variablen aus

```
$ set
BASH=/bin/bash
BASH_VERSION='3.2.48(1)-release'
COLUMNS=156
COMMAND_MODE=unix2003
DIRSTACK=()
DISPLAY=/tmp/launch-Lujw2L/org.x:0
EUID=501
GROUPS=()
HISTFILE=/home/esser/.bash_history
HISTFILESIZE=500
HISTSIZE=500
HOME=/home/esser
HOSTNAME=macbookpro.fritz.box
...
```

103.1: Shell-Variablen (2)

- Einzelne Variablen geben Sie mit „echo“ und einem Dollar-Zeichen ($) vor dem Variablennamen aus

```
$ echo $SHELL
/bin/bash
$ _
```

- zum Ändern / Setzen schreiben Sie „var=wert“:

```
$ TESTVAR=fom
$ echo $TESTVAR
fom
$ set | grep TEST
TESTVAR=fom
$ _
```

- Sie können Variablen auch **exportieren**:

```
$ export TESTVAR
$ _
```

→ nächste Folie

103.1: Shell-Variablen (3)

- Exportieren?

  Wert einer Variablen gilt nur lokal in der laufenden Shell.

- Exportierte Variablen gelten auch in aus der Shell heraus gestarteten Programmen

```
$ A=eins; B=zwei; export A
$ echo "A=$A B=$B"
A=eins B=zwei
$ bash              # neue Shell starten; das ist ein neues Programm!
$ echo "A=$A B=$B"
A=eins B=
$ exit              # diese zweite Shell verlassen, zurück zur ersten
$ echo "A=$A B=$B"
A=eins B=zwei
```

!

103.1: Shell-Variablen (4)

- Liste aller exportierten Variablen gibt „export“ ohne Argument aus – allerdings in ungewöhnlicher Syntax

```
$ export
declare -x A="1"
declare -x Apple_PubSub_Socket_Render="/tmp/launch-CYfDhh/Render"
declare -x COMMAND_MODE="unix2003"
declare -x DISPLAY="/tmp/launch-Lujw2L/org.x:0"
declare -x HOME="/Users/esser"
declare -x INFOPATH="/sw/share/info:/sw/info:/usr/share/info"
declare -x LOGNAME="esser"
...
```

- (Hintergrund: „declare -x VAR“ exportiert ebenfalls die Variable VAR, ist also dasselbe wie „export VAR“)

103.1: History (1)

- Shell merkt sich die eingegebenen Befehle („History“)
- Komplette Ausgabe mit „history“:

```
$ history
1  df -h
2  ll
3  /opt/seamonkey/seamonkey
4  dmesg|tail
5  ping hgesser.de
6  google-chrome
7  killall kded4
```

- Wie viele Einträge? Normal 500:

```
$ echo $HISTSIZE
500
```

103.1: History (2)

- Neben Ausgabe der kompletten History gibt es auch eine intelligente Suche nach alten Kommandos: [Strg-R]

```
$  # Suche nach dem letzten echo-Aufruf
$ ^R
(reverse-i-search)`ech': echo $HISTFILESIZE
```

- mit [Eingabe] ausführen
- weitere [Strg-R] liefern ältere Treffer
- Außerdem: Mit [Pfeil hoch], [Pfeil runter] durch alte Befehle blättern
- gefundenes Kommando kann übernommen und überarbeitet werden

103.2: Filter für Text-Streams

- Idee beim Filter:
  - Standardeingabe in Standardausgabe verwandeln
  - Ketten aus Filtern zusammen bauen:
  - `prog1 | filter1 | filter2 | filter3` ...
  - mit Eingabedatei:
    `prog1 < eingabe | filter1 | ...`
- cat, cut, expand, fmt, head, od, join, nl, paste, pr, sed, sort, split, tail, tr, unexpand, uniq, wc

103.2: cat

- cat steht für con**cat**enate (aneinanderfügen)
- gibt mehrere Dateien unmittelbar hintereinander aus
- auf Wunsch auch nur eine Datei
  → Mini-Dateibetrachter
- Spezialoptionen:
  - -n (Zeilennummern)
  - -T (Tabs als ^I anzeigen)
  - … und einige weitere (siehe: man cat)

103.2: cut

- cut kann spaltenweise Text ausschneiden – Spalten sind wahlweise definierbar über
  - Zeichenpositionen
  - Trennzeichen (die logische Spalten voneinander trennen)

c: character; zeichenbasiert

d: delimiter; Trennzeichen

```
$ cat test.txt
1234 678901 234
abc def ghijklmn
r2d2 12 99
1 2 3
Langer Testeintrag
```

```
$ cut -c3-8 test.txt
34 678
c def
d2 12
2 3
nger T
```

```
$ cut -d" " -f2,3 test.txt
678901 234
def ghijklmn
12 99
2 3
Testeintrag
```

f: field (Feld)

103.2: fmt

- fmt (**for**mat) bricht Textdateien um

keine Umbrüche

```
$ cat test.txt
Das ist mal ein Beispiel fuer einen Satz. Das ist mal ein Beispiel fue
r einen Satz. Das ist mal ein Beispiel fuer einen Satz. Das ist mal ei
n Beispiel fuer einen Satz. Das ist mal ein Beispiel fuer einen Satz.
Das ist mal ein Beispiel fuer einen Satz. Das ist mal ein Beispiel fue
r einen Satz. Das ist mal ein Beispiel fuer einen Satz. Das ist mal ei
n Beispiel fuer einen Satz. Das ist mal ein Beispiel fuer einen Satz.

$ fmt test.txt
Das ist mal ein Beispiel fuer einen Satz. Das ist mal ein Beispiel
fuer einen Satz. Das ist mal ein Beispiel fuer einen Satz. Das ist
mal ein Beispiel fuer einen Satz. Das ist mal ein Beispiel fuer
einen Satz. Das ist mal ein Beispiel fuer einen Satz. Das ist mal
ein Beispiel fuer einen Satz. Das ist mal ein Beispiel fuer einen
Satz. Das ist mal ein Beispiel fuer einen Satz. Das ist mal ein
Beispiel fuer einen Satz.
```

Zeilen-umbrüche

- Parameter -w75: Breite 75 (width)

103.2: split

- split kann große Dateien in mehrere Dateien mit angegebener Maximalgröße aufteilen
- (cat fügt diese anschließend wieder zusammen)

```
$ split ZM_ePaper_18_11.pdf -b1440k ZM_ePaper_18_11.pdf.
$ ls -l ZM*
-rw-r--r-- 1 esser esser 10551293 2011-04-29 06:58 ZM_ePaper_18_11.pdf
-rw-r--r-- 1 esser esser  1474560 2011-04-29 14:46 ZM_ePaper_18_11.pdf.aa
-rw-r--r-- 1 esser esser  1474560 2011-04-29 14:46 ZM_ePaper_18_11.pdf.ab
-rw-r--r-- 1 esser esser  1474560 2011-04-29 14:46 ZM_ePaper_18_11.pdf.ac
-rw-r--r-- 1 esser esser  1474560 2011-04-29 14:46 ZM_ePaper_18_11.pdf.ad
-rw-r--r-- 1 esser esser  1474560 2011-04-29 14:46 ZM_ePaper_18_11.pdf.ae
-rw-r--r-- 1 esser esser  1474560 2011-04-29 14:46 ZM_ePaper_18_11.pdf.af
-rw-r--r-- 1 esser esser  1474560 2011-04-29 14:46 ZM_ePaper_18_11.pdf.ag
-rw-r--r-- 1 esser esser   229373 2011-04-29 14:46 ZM_ePaper_18_11.pdf.ah
$ cat ZM_ePaper_18_11.pdf.* > ZM_Kopie.pdf
$ ls -l ZM_Kopie.pdf
-rw-r--r-- 1 esser esser 10551293 2011-04-29 14:48 ZM_Kopie.pdf
$ diff ZM_ePaper_18_11.pdf ZM_Kopie.pdf
$ _
```

103.2: sort

- sort ist ein komplexes Sortier-Tool, das
  - Sortierung nach *n*-ter Spalte
  - alphabetische und numerische Sortierung

  unterstützt
- Einfache Beispiele:

```
$ cat test3.txt
13 Autos
5 LKW
24 Fahrraeder
2 Baeume
Wohnung
Haus
Hotel
Strasse
Allee
```

```
$ sort test3.txt
13 Autos
2 Baeume
24 Fahrraeder
5 LKW
Allee
Haus
Hotel
Strasse
Wohnung
```

```
$ sort -n test3.txt
Allee
Haus
Hotel
Strasse
Wohnung
2 Baeume
5 LKW
13 Autos
24 Fahrraeder
```

103.2: uniq

- uniq (**uniq**ue, einmalig) fasst mehrere identische (aufeinander folgende) Zeilen zu einer zusammen; entfernt also Doppler
- Alternative: Beim Sortieren mit sort kann man über die Option -u (**u**nique) direkt Doppler entfernen;
  - statt `sort datei | uniq` also besser `sort -u datei`

103.2: grep

- grep (**g**lobal/**r**egular **e**xpression/**p**rint) zeigt nur die Zeilen einer Datei, die einen Suchbegriff enthalten – oder nicht enthalten (Option -v)

```
$ wc -l /etc/passwd
57 /etc/passwd
$ grep esser /etc/passwd
esser:x:1000:1000:Hans-Georg Esser,,,:/home/esser:/bin/bash
$ grep /bin/bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
esser:x:1000:1000:Hans-Georg Esser,,,:/home/esser:/bin/bash
$ grep -v /bin/bash /etc/passwd | head -n5
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
```

103.2: sed (1/2)

- sed (**S**tream **Ed**itor) führt (u. a.) Suchen-/Ersetzen-Funktionen in einem Text durch

```
$ cat test4.txt
Das Wort ist ein Wort, und mehrere
Woerter sind der Plural von Wort.
Ohne Woerter oder Worte gibt es
keinen Satz - wir sind wortlos.

$ sed 's/Wort/Bild/' test4.txt
Das Bild ist ein Wort, und mehrere
Woerter sind der Plural von Bild.
Ohne Woerter oder Bilde gibt es
keinen Satz - wir sind wortlos.

$ sed 's/Wort/FOM/g' test4.txt
Das FOM ist ein FOM, und mehrere
Woerter sind der Plural von FOM.
Ohne Woerter oder FOMe gibt es
keinen Satz - wir sind wortlos.
```

```
$ sed 's/Wort/FOM/gi' test4.txt
Das FOM ist ein FOM, und mehrere
Woerter sind der Plural von FOM.
Ohne Woerter oder FOMe gibt es
keinen Satz - wir sind FOMlos.
```

s: substitute (**s**/.../.../gi)
g: global (s/.../.../**g**i)
i: ignore case (s/.../.../g**i**)

Die i-Option gibt es nicht in jeder sed-Version!

103.2: sed (2/2)

- sed-Optionen:
  - -i: in-place-editing, verändert die angegebene Datei; am besten mit Angabe eines Suffix für eine Backup-Datei:
    z. B. `sed -i.bak 's/Wort/Bild/g' test4.txt`
    legt erst Sicherheitskopie test4.txt.bak an und verändert dann test4.txt
  - -e: zum Kombinieren mehrerer Ersetzungen; z. B. `sed -e 's/1/eins/g' -e 's/2/zwei/g' test.txt`
  - weitere Optionen → Manpage

103.2: Reguläre Ausdrücke

- Idee: Allgemeinere Suchbegriffe, vergleichbar mit Wildcards (*, ?) bei Dateinamen
- Muster:
  - . – ein beliebiges Zeichen
  - [abcd] – eines der Zeichen a, b, c, d
  - [2-8] – eines der Zeichen 2, 3, 4, 5, 6, 7, 8
  - ^ – Zeilenanfang
  - $ – Zeilenende
  - ? – vorheriger Ausdruck darf vorkommen, muss aber nicht
  - * – vorheriger Ausdruck kann beliebig oft (auch 0 mal) vorkommen

```
$ cat test5.txt
Haus
Die Hotels
Hotels am Wasser
Bau-Haus-Objekt
Diese Zeile nicht
```

```
$ grep 'H.*s' test5.txt
Haus
Die Hotels
Hotels am Wasser
Bau-Haus-Objekt
```

```
$ sed 's/H.*s/HAUS/g' test5.txt
HAUS
Die HAUS
HAUSer
Bau-HAUS-Objekt
Diese Zeile nicht
```

103.2: Reguläre Ausdrücke

- Beispiele für reguläre Ausdrücke
  (live, in der Shell...)