Labor Client/Server-Programmierung: Grundlegende Linux-Befehle

Im Folgenden sind die wichtigsten Befehle zur Eingabe an der Eingabeaufforderung („Terminal“, „Konsole“) aufgeführt, die im Rahmen der Aufgaben benötigt werden.

1. Befehlseingabe

- Befehle werden nach Eingabe mit der Entertaste ausgeführt.
- Parameter sind durch spitze Klammern gekennzeichnet: `<parameter>`
- Optionale Parameter sind zusätzlich durch eckige Klammern gekennezeichnet: `[<optional>]`

History-Funktion innerhalb desselben Terminals

- Bereits eingegebene Befehle können durch die Pfeil-oben Taste durchgeblättert werden
- Mit Strg-r kann die History nach Befehlen durchsucht werden
- Mit Alt-. können Parameter vorangegangener Befehle durchgeblättert werden

2. Dokumentation (man pages)

Zu C-Funktionen und Linux-Kommandos sind man-pages installiert. Diese werden aufgerufen mit:

`man <name>` (z.B. `man read`)

Sind für ein Name mehrere Einträge vorhanden, so kann durch Angabe einer „Section“ angegeben werden, was gesucht wird:

`man <section> <name>` (z.B. `man 2 accept`)

wichtige Sections

- Section 1: Systemkommandos (für Kommandozeile)
- Section 2: System calls
- Section 3: Aufrufe von C-Bibliotheken

Man-pages können durch Eingabe von einem Schrägstrich (/) und eines Suchbegriffes durchsucht werden. Das Weitersuchen nach dem gleichen Begriff erfolgt durch Eingabe von „n“.

Man-pages werden mit q beendet.

3. Dateien und Verzeichnisse

Vezeichnisse in Linux werden durch „/“ getrennt (forward-slash im Gegensatz zum back-slash „\“ bei Windows). Es werden keine Laufwerksbuchstaben wie bei Windows (`C:\Windows`) verwendet.

Datei- und Verzeichnisnamen sollten **keine Leerzeichen** enthalten. Leerzeichen in Datei- oder Verzeichnisnamen müssen durch einen vorangestellen back-slash gekennzeichnet werden.

Home Verzeichnis eines Benutzers (entspricht „Eigene Dateien“)

- liegt unter `/home/<user>`, z.B. `/home/student`
- Abgekürzte Darstellung: `~`

Aktuelles Verzeichnis

- wird an der Eingabeaufforderung angezeigt. Z.B.: `user@xubuntu:~/aufgabe1 →~/aufgabe1` bedeutet Verzeichnis `aufgabe1` unter dem Home-Verzeichnis `~`
- Anzeige des absoluten Pfads (ohne ~) mit dem Befehl `pwd`
- Das aktuelle Verzeichnis wird als Punkt „`.`“ angegeben. Zum Beispiel als Ziel für Kopieraktionen.

Befehle

- **`ls`**
  Anzeigen des Verzeichnisinhalts
- **`ls -l`**
  Anzeigen des Verzeichnisinhalts, **ausführlich**
- **`cd <directory>`**
  Wechsel in ein Verzeichnis, z.B. `cd aufgabe1`
- **`cd ..`**
  Wechsel in das übergeordnete Verzeichnis
- **`mkdir <directory>`**
  Erstellen eines Verzeichnisses, z.B. `mkdir aufgabe1`
- **`rm <file>`**
  Löschen einer Datei
- **`rm -r <directory>`**
  Löschen eines Verzeichnis, inkl. Untervezeichnissen und enthaltenen Dateien.
- **`cp <source> <destination>`**
  kopieren von Dateien
- **`cp -r <source> <destination>`**
  Kopieren von Verzeichnissen

4. Ausführen von Programmen

- **Systemweit installierte** Programme werden durch Aufruf des Programmnamens gestartet, z.B. `pstree`
- Ausführbare Dateien **im aktuellen Verzeichnis** werden durch voranstellen von „`./`“ gestartet. Z.b. Ausführen des Programms „signals“ im aktuellen Verzeichnis: `./signals`
- Es kann auch der relative oder absolute Pfad ausgehend vom aktuellen Verzeichnis angegben werden. Z.b. `/home/student/aufgabe1/signal` oder `aufgabe1/signal` (wenn man sich in `home/student/` befindet)
- Die Ausgabe kann durch Angabe des pipe-Symbols („|“ = Alt Gr „>“, beim Y) in andere Prozesse umgeleitet werden

- Umleitung in **less**, um größere Mengen Text zu betrachten: z.B. `pstree | less` (scrollen mit oben/unten, beenden von less mit q)
- Umleitung in **grep**, um Textzeilen zu suchen: z.B. `pstree | grep signal` (zeigt nur Textzeilen, die den Text „signal“ enthalten)

5. Nachinstallieren von Programmen

Programme sind in packages organisiert, diese können nachinstalliert werden mit:

```
sudo apt-get install <package>
```

Beispiel: nachinstallieren von des Editors gedit: `sudo apt-get install gedit`

Der Befehl sudo führt den folgenden Befehl mit root-Rechten aus. Es muss das Benutzerpasswort eingegeben werden.

6. Download von Dateien

Mit wget können Dateien einfach über die URL als Datei ins aktuelle Verzeichnis heruntergeladen werden.

Beispiel: `wget jochenkoegel.de/dhbw/code/signals.c`

7. Kompilieren von C-Quellcode

Der Compiler heißt gcc und lässt sich von der Kommandozeile aufrufen.

```
gcc -o <binary> <source.c>
```

`<binary>` benennt die ausführbare Datei, die der Compiler erstellt

`<source.c>` ist die C-Quelldatei, die der Compiler verwenden soll

Beispiel: `gcc -o signals signals.c`

8. Prozesssteuerung

- **`ps`**
  Gibt die Liste aktiver Prozesse aus. Ohne weitere Parameter werden nur die von der aktuellen Konsole gestarteten Prozesse ausgegeben. Mit `ps –e` werden alle Prozesse ausgegeben (erweitertes Format mit `-f/-F`). Mit `ps –ef | less` wird die Ausgabe in das Programm `less` umgeleitet und kann durchgeblättert werden.
- **`top`**
  Übersicht über laufende Prozesse, sortiert nach CPU/Speicherverbrauch
- **`gnome-system-monitor`**
  Grafische Version von top
- **`pstree`**
  (pstree -lcp | less)
  Ausgeben der Prozesshierarchie
- **`strace <programm>`**
  Verfolgung von Systemaufrufen des mit strace gestarteten Programms

- **`pgrep <name>`**
  Gibt alle PIDs der Prozesse mit Namen <name> aus
- **Strg-C**
  in Konsole des laufenden Prozesses

  Sendet das Signal SIGINT an den Prozess. Dies unterbricht den Prozess, woraufhin sich die meisten Programme geregelt beenden.
- **Strg-Z**
  in Konsole des laufenden Prozesses

  Versetzt den Prozess in den Zustand suspended (angehalten)
- **&**

  Angehängt an den Namen eines Programms führt die dazu, dass das Programm im Hintergrund ausgeführt wird. z.B. `gedit newfile.txt &`
- **`fg`**

  Versetzt einen angehaltenen Prozess in der aktuellen Konsole wieder in den Zustand ready und setzt ihn in den Vordergrund
- **`bg`**

  Versetzt einen angehaltenen Prozess in der aktuellen Konsole wieder in den Zustand ready und setzt ihn in den Hintergrund
- **`kill <pid>`**

  Sendet ein Signal an den Prozess mit der Prozess-ID <pid>. Ohne Angabe des Signals wird das Signal SIGTERM gesendet.
- **`killall <name>`**

  Sendet allen Prozessen mit dem Namensteil <name> ein Signal, normalerweise SIGTERM.