Grundlagen der Technischen Informatik
Wintersemester 23/24

ertis embedded real-time systems ulm university institute

Laborprojekt – Versuch 1

Lernziele

- Verständnis der Kommandozeile
- Eigenständiges Arbeiten mit der Linux-Kommandozeile

Einführung in die Kommandozeile unter Linux

Gedacht für Anfänger zum *Überleben* in der Kommandozeile Zum Umgang mit der Kommandozeile von Linux existieren im Internet zahllose Tutorials, wie z.B. ein Englisches unter `http://linuxcommand.org/lc3_learning_the_shell.php`. Für Anfänger soll hier vermittelt werden, wie die Kommandozeile grundsätzlich funktioniert, andererseits soll man schnell bei den Kommands sein, die man zum Überleben braucht.

Das Grundprinzip Das Grundprinzip von Kommandzeilen ist wie folgt: Man tippt ein paar Zeichen ein und drückt dann die Eingabe-Taste. Die Eingabe wird daraufhin in Wörter zerlegt, wobei das erste Wort das Programm sein muss, das aufgerufen werden soll bzw./ wird. Alle weiteren Wörter werden diesem Programm als Parameter übergeben. Der Trick der Kommandozeile ist also eher: Welche Kommandos gibt es und was bedeuten weitere Wörter für dieses Kommando?

Ein einfaches Beispiel: Tippt man

```
echo huhu
```

dann wird das Programm `echo` aufgerufen. Dieses Programm gibt seine Parameter einfach wieder aus: Es erscheint `huhu` auf dem Bildschirm.

Noch ein Beispiel:

```
rm beispiel.txt
```

löscht die bezeichnete Datei `beispiel.txt`. `rm` ist das Programm remove (auf `rm` abgekürzt, damit man nicht so viel tippen muss), das seinen Parameter, das Wort `beispiel.txt`, als Dateinamen versteht.

Bei der Eingabe des Kommandos gibt es Editierhilfen, von denen man anfangs aber nicht viele braucht: Die Pfeiltasten Rechts-Links und die Backspace-Taste (über der Eingabetaste) um das Zeichen vor dem Cursor zu löschen reichen am Anfang aus. Viel Tipparbeit sparen einem die Pfeiltasten Rauf-Runter, die alte, bereits ausgeführte Kommandos wieder abrufen. Letzteres ist je nach verwendeter Umgebung im Detail unterschiedlich – einfach selbst ausprobieren!

Dateinamen bzw. Pfade Viele Programme erwarten Namen von Dateien oder Verzeichnissen als Parameter. Wichtig für die Kommandozeile ist es daher, dass man versteht, wie man Dateien auf der Kommandozeile spezifiziert, weil dies für alle Kommandos identisch passiert.

Alle Dateien sind in Verzeichnissen einsortiert, die wiederum einen Baum aufspannen. Der Windows-Explorer stellt genau einen solchen Verzeichnisbaum graphisch dar, um mit der Maus darin zu navigieren. Unter Linux heißt das oberste Verzeichnis / (nur der Schrägstrich). Darunter haben Verzeichnisse Namen und dürfen auch wieder Unterverzeichnisse haben. `/home/xy13` ist also im Wurzelverzeichnis / das Unterverzeichnis `home` und darin das Unterverzeichnis `xy13`. Einzelne Dateien hängt man einfach hinten an einen solchen Pfad an: `/home/xy13/meine_datei.txt` spezifiziert im genannten Verzeichnis die Datei `meine_datei.txt`. Der Schrägstrich / ganz vorne ist der Name des Wurzelverzeichnisses, an späteren Stellen im Pfad dient er nur der Trennung der Verzeichnis- und Dateinamen.

Welches Unterverzeichnis wofür gut ist, ist im Prinzip frei wählbar, nur Konventionen und Erfahrung helfen hier weiter. Obiges Beispiel `/home/xy13` hätte gute Chancen das Heimatverzeichnis des Users *xy13* zu sein. Glücklicherweise braucht man anfangs die meisten Konventionen gar nicht zu kennen, sondern macht einfach im eigenen Home-Verzeichnis was man will.

Kommandozeilen unterstützen die Arbeit mit Verzeichnissen mit einem sogenannten *current directory*, also einem aktuellen Verzeichnis. Man kann dieses wechseln, indem man das Kommando `cd`, kurz für *change directory* benutzt. Mit

```
cd /home/xy13
```

sagt man also, dass man sein `/home/xy13` als aktuelles Verzeichnis haben will. Beginnt man einen Verzeichnis- oder Dateinamen *nicht* mit /, wird also automatisch das aktuelle Verzeichnis angenommen wird. Ist also das aktuelle Verzeichnis `/home/xy13`, dann kann man die Datei `meine_datei.txt` einfach genau so angeben. Der Mechanismus um das aktuelle Verzeichnis spart viel Tipparbeit. Möchte man explizit auf das aktuelle Verzeichnis verweisen, so verwendet man statt dem / am Anfang des Pfades einen Punkt `.`, d.h. `./meine_datei.txt`

verweist explizit auf die eigene Datei im aktuellen Verzeichnis. Im Fachjargon spricht man von *absolutem Pfad*, wenn man eine Datei oder ein Verzeichnis mit einem / vorne benennt, oder *relativem Pfad*, wenn man vom current directory aus startet.

Ein weiterer Shortcut ist die Tilde `~`, die sie im Normalfall in Ihr Home-Verzeichnis bringt. Als Beispiel würde `~/meine_datei.txt` auf die eigene Datei verweisen, die sich direkt in Ihrem Home-Verzeichnis befindet. Ebenso können Sie mit

```
cd ~
```

jederzeit schnell in Ihr Home-Verzeichnis wechseln.

Noch ein paar weiterführende Tipps zu Dateien und Verzeichnissen:

- Die Tab-Taste ist üblicherweise mit einer *Dateinamen-Vervollständigung* belegt. Tippt man an einem Dateinamen, ist noch nicht fertig, und drückt dann auf die Tab-Taste, dann sucht der Mechanismus den halb eingetippten Dateinamen im Verzeichnis und vervollständigt den Namen. Bleiben mehrere Möglichkeiten, dann passiert meist irgend etwas, was einem weiterhilft, evtl. auch erst beim zweiten Druck auf die Tab-Taste.

- Das Zeichen `*` steht in einem Dateinamen für beliebige andere Zeichen. Tippt man also z.B. `rm meine_d*.txt`, dann wird ziemlich sicher die Datei `meine_datei.txt` gelöscht, aber beispielsweise auch eine Datei mit dem Namen `meine_daten.txt`!

- Jedes Verzeichnis hat immer einen Eintrag mit dem Namen `..` (zwei Punkte), der jeweils auf das übergeordnete Verzeichnis (*parent directory*) zeigt. Mit

  ```
  cd ..
  ```

  wechselt man beispielsweise in das jeweils übergeordnete Verzeichnis. Aber man kann das auch mitten in einer Pfadangabe nutzen:

  ```
  rm ../anderes_subdir/hallo.txt
  ```

  löscht man beispielsweise die Datei `hallo.txt` im einem Verzeichnis `anderes_subdir`, das *parallel* zum aktuellen Verzeichnis im selben übergeordneten Verzeichnis liegt.

- Das zu startende Kommando, also das erste Wort einer Kommandozeile, darf nicht nur bekannte Kommandos aufrufen. Vielmehr wird als erstes Wort ein beliebiger Dateiname akzeptiert, auch mit Verzeichnisangaben. Ein eigenes Programm im aktuellen Verzeichnis kann man z.B. mit

  ```
  ./meinProgramm meinParameter
  ```

  gestartet werden, wobei es als Parameter `meinParameter` übergeben bekommt. Programme können aber auch in anderen Verzeichnissen gestartet werden:

```
/home/foo/fooProgramm barParameter
```

startet das Programm `fooProgramm` im Home-Verzeichnis eines Users *foo* und übergibt auch einen entsprechenden Parameter. Was Ihnen vielleicht schon aufgefallen ist: Standard-Programme und deren entsprechende Kommandos wie `cd` können Sie immer und überall nutzen, ohne deren Pfad anzugeben. Diese werden in der Umgebung der Kommandozeile gesetzt und quasi für Sie *gefunden*. Wie sich dies erweitern lässt etc. sprengt dann allerdings etwas den Rahmen dieser Kurzeinführung.

Optionen Viele Kommandos kennen auch Optionen, mit denen man deren Verhalten modifizieren kann. Unter Linux/Unix gilt die Konvention, dass Optionen auf der Kommandozeile Wörter sind, die mit einem Minuszeichen anfangen. Beispiel:

```
rm -i *.txt
```

Hier wird beim Löschen von Dateien mit dem Kommando `rm` durch die Option `-i` erreicht, dass vor dem Löschen jeder einzelnen Datei – die in diesem Fall auf `.txt` endet, zurückgefragt wird, ob diese wirklich gelöscht werden soll.

Zwei weitere nützliche Optionen als Beispiele:

```
cp -r Quellverzeichnis Zielverzeichnis
```

Die Option `-r` (engl. *recursive*) kopiert `Quellverzeichnis` *mit allen Unterverzeichnissen und Dateien darin* nach `Zielverzeichnis`.

```
rm -r Verzeichnis
```

Hier löscht die Option `-r` das Verzeichnis *mit allen Unterverzeichnissen und Dateien darin*. Hinweis: Ohne `-r` akzeptiert `rm` nur Dateien zum Löschen und bezieht keine (Unter)Verzeichnisse ein.

Viele Programme können sich (sehr kurz) selbst erklären, indem man sie mit der Option `-h` oder `--help` aufruft. Längere Erklärungen erhält man meist, indem man das Handbuch mittels des Kommandos `man` (engl. *manual*) gefolgt vom eigentlichen Kommando benutzt:

```
man rm
```

Hier wird beispielsweise eine vollständige Nutzeranleitung des Kommandos `rm` gezeigt.

Wichtige erste Kommandos

ls Das Kommando `ls`, für List, zeigt an, welche Dateien es gibt. Ohne Parameter wird der Inhalt des aktuellen Verzeichnisses angezeigt:

```
ls
```

Durch den Parameter / wird der Inhalt des Wurzelverzeichnisses angezeigt:

```
ls /
```

Die Option `-l` gibt deutlich mehr Informationen pro Datei aus:

```
ls -l
```

cp Das Kommando `cp`, für copy, dient zum Kopieren von Dateien.

```
cp original.jpg copy.jpg
```

Hier wird eine Kopie der Datei `original.jpg` als Datei `copy.jpg` erstellt.

```
cp *.jpg ./images/
```

Hier werden alle Bilddateien (`*.jpg`) in ein Unterverzeichnis `images` des aktuellen Verzeichnisses kopiert.

mv Das Kommando `mv`, für move, ist ähnlich dem Kommando `cp`. Dateien oder Verzeichnisse werden jedoch nicht kopiert, sondern verschoben. Das Kommando kann auch gut zum Umbenennen von Dateien oder Verzeichnissen verwendet werden.

rm Das Kommando `rm`, für remove, dient dem Löschen von Dateien (und ggf. Verzeichnissen): Alle Dateien des aktuellen Verzeichnisses würden (Vorsicht!) gelöscht mit:

```
rm *
```

Alle Dateien mit Endung `.jpg` würden im aktuellen Verzeichnis wie folgt gelöscht:

```
rm *.jpg
```

Wie bereits erwähnt sorgt die Option `-r` dafür, dass im angegebenen Verzeichnis alle Dateien und Unterverzeichnisse bzw. sogar das Verzeichnis selbst gelöscht wird. Nutzen Sie das Kommando `rm` mit Bedacht und ggf. die Option `-i`, bei der sie jedes einzelne Löschen bestätigen müssen.

cd Das Kommando `cd`, für change directory, wird zum Wechseln des aktuellen Verzeichnisses benutzt. Beispiele finden sich bereits in der vorherigen Einführung.

mkdir Das Kommando `mkdir`, für make directory, legt ein neues (Unter)Verzeichnis an. Ein Verzeichnis `tmpDir` würde man im aktuellen Verzeichnis wie folgt anlegen:

```
mkdir tmpDir
```

Man kann jedoch auch in einem beliebigen Verzeichnis ein neues Unterverzeichnis anlegen:

```
mkdir /home/foo/bar
```

Hier wird im Home-Verzeichnis des Users *foo* das Verzeichnis `bar` angelegt.

rmdir Das Kommando `rmdir`, für remove directory, ist quasi die Umkehrung zu `mkdir`. `rmdir` löscht jedoch nur leere Verzeichnisse! Wie man ein Verzeichnis leer bekommt, siehe `rm`.

Editieren Zum Bearbeiten/Erstellen etc. von Dateien, benötigt man einen Editor. Wenn es einmal schnell gehen soll, lassen sich einfache Editoren wie `vim` oder `nano` auch direkt von der Kommandozeile starten. Komfortablere Editoren oder *IDEs* sind für moderne Entwicklung – Vorsicht Glaubenskrieg anrollend – unerlässlich und lassen sich meist komfortabler als Programm über die graphische Benutzeroberfläche starten.

exit/logout Die Kommandos `exit` bzw. `logout` schließen Ihre aktuelle Kommandozeile bzw. loggen Sie aus einem System aus. Achtung: Nutzen Sie einen Rechner z.B. im Pool und möchten sich bei kurzen Pausen nicht immer ausloggen, können Sie den Bildschirm auch kurzzeitig sperren und so vor fremdem Zugriff schützen. Wenn Sie die Kommandozeile in einer modernen Linux-Distribution mit einem entsprechenden graphischen Desktop Environment nutzen, müssen Sie das Sperren mit Tastenkürzeln (z.B. CTRL+ALT+L bzw. Super+L) oder durch entsprechende graphische Buttons realisieren.

Wie finde ich ein Programm bzw. Kommando für mein Problem? Jetzt sind wir beim eigentlichen Problem der Kommandozeile: Als Anfänger fehlt einem oft die Erfahrung, welche Programme und deren entsprechende Kommandos es gibt. Was tut man also als Anfänger, wenn man das passende Kommando noch nicht kennt? Die mit Abstand praktikabelste Methode im Laborprojekt: Mitstudierende fragen, die schon Erfahrung im Umgang mit der Kommandozeile haben oder – leider – Selbststudium.

Aufgaben

Aufgabe 1: Einfach mal Loslegen

Loggen Sie sich an einem Rechner im Praktikumsraum genau so ein, wie auf einem Rechner im Linux-Pool O27/133. Zuerst den Accountnamen eingeben, dann rechts unten den gewünschten Windows-Manager auswählen, zuletzt Ihr Passwort eingeben. Als graphisches Desktop Environment oder Window-Manager empfehlen wir *Cinnamon* – vor allem falls Sie Microsoft Windows

gewöhnt sind und/oder Linux noch gar nicht kennen. Ansonsten haben Sie natürlich die freie Auswahl. Öffnen Sie eine Kommendozeile oder *Shell*. In Cinnamon geht das, indem man den Knopf links unten klickt und dann senkrecht darüber das relativ schwarze Icon mit >_ darin.

Kopieren Sie in der Shell den Ordner `versuch1` komplett mit allen Dateien und Unterverzeichnissen in den Ordner `gdti/versuch1` in Ihrem Home-Verzeichnis:

```
cp -r /tiprak/gdti/versuch1 ~/gdti/versuch1
```

Wechseln Sie in das gerade erstellt Verzeichnis, damit sie bei den Dateiangaben ohne Verzeichnisangaben auskommen:

```
cd gdti/versuch1
```

Zur Übung befindet sich in dem Verzeichnis eine Datei, die Sie einfach löschen sollen. Lassen Sie sich alle Dateien in diesem Verzeichnis mit Hilfe des `ls` Kommandos anzeigen und finden Sie die Datei, deren Name schon sagt, dass sie weg will. Löschen Sie diese mittels des Kommandos `rm`.

Zum Erstellen von (Text-)Dateien, also auch Java- oder VHDL-Quelltexten, benutzt man sogenannte Editoren. Notepad oder Notepad++ kennen Sie eventuell von Microsoft Windows, viele IDEs haben auch einen Editor integriert. In einem Text-Fenster kann man aber auch Editieren. Wir möchten Ihnen keinen Editor vorschreiben, empfehlen Ihnen aber für das Laborpojekt den Editor *gedit*, da er für Sie auch ein automatischen Syntax-Highlighting bei Dateien mit Dateiendung `.vhdl` bietet.

Für den Fall, dass Sie `gedit` nutzen möchten, startien Sie auf der Kommandozeile mit:

```
gedit meine_erste_Datei.txt
```

Es geht nun ein eigenes Editor-Fenster auf. (Ersetzen Sie das Kommando ggf. durch den Editor Ihrer Wahl – wir möchten Glaubenskriege vermeiden.)

Schreiben Sie drei Zeilen beliebigen Text in die Datei `meine_erste_Datei.txt` und duplizieren Sie diese mittels Auswählen & Einsetzen (engl. cut/copy & paste), so dass sechs Zeilen in der Datei enthalen sind – suchen Sie ggf. danach,

wie sich dies in Ihrem Editor realisieren lässt.

Abnahme Erklären Sie, wie eine Kommandozeile grundsätzlich funktioniert. Zeigen Sie das genaue von Ihnen verwendete `rm`-Kommando zum Löschen der Datei sowie Ihre Datei `meine_erste_datei.txt`.

Aufgabe 2: Inhalt von Dateien anzeigen

Wie können Sie sich den Inhalt der Datei `sortiere_mich.txt` anzeigen lassen? Finden Sie dies im Selbststudium heraus.

Abnahme Zeigen Sie wenigstens drei Möglichkeiten, sich den Inhalt von Dateien anzeigen zu lassen.

Aufgabe 3: `grep`, `wc` und `sort`

Schauen Sie sich die man-pages der Kommandos `grep`, `wc` und `sort` an und probieren Sie diese aus.

Abnahme

- Wieviele Bytes, Worte und Zeilen hat die Datei `sortiere_mich.txt`?
- Suchen Sie alle Zeilen, die das Wort *absicht* enthalten, unabhängig von Groß- und Kleinschreibung!
- Bringen Sie den Inhalt der Datei `sortiere_mich.txt` so sortiert auf den Bildschirm, dass die Zeilen nach den Nummern in den Zeilen sortiert sind. Zeigen Sie die von Ihnen für `sort` verwendeten Optionen.

Knobelaufgabe Versuch 1: `head` und `tail`

Wenden Sie die Kommandos `head` und `tail` auf die Datei `sortiert.txt` wie folgt an:

- Lassen Sie nur die ersten fünf Zeilen (Zeilen 1–5) anzeigen.
- Lassen Sie nur die letzten fünf Zeilen (Zeilen 16–20) anzeigen.
- Lassen Sie alle Zeilen außer die ersten Fünf anzeigen, d.h., Zeilen 6–20.
- Lassen Sie alle Zeilen außer die letzten Fünf anzeigen, d.h., Zeilen 1–15.

Ihre verwendeten Kommandos dürfen nicht darauf angewiesen sein zu wissen, wieviele Zeilen in der Datei sind! Lösungen wie `tail -n 15` für die dritte Teilaufgabe funktionieren dabei nur, weil die Datei genau 20 Zeilen enthält, und man im Kopf die 5 Zeilen subtrahiert hat, die man nicht sehen will.

Abnahme Führen Sie Ihre vier verwendeten Kommandos vor!