EINFÜHRUNG IN LINUX

DR. MATTHIAS M. HÖLZL

1. Aufbau eines Computer-Systems

Ein Computersystem besteht aus Hardware (dem eigentlichen Rechner) und Software (den Programmen). Zur Hardware zählen der *Prozessor* (CPU, Central Processing Unit oder ALU, Arithmetic Logic Unit), der *Hauptspeicher* (RAM, Random Access Memory), die Massenspeicher (z.B. *Festplatten* (Hard Disks, Winchester Drives), *Floppy Disks* oder *Bandlaufwerke*) sowie Ein- und Ausgabegeräte (Monitor, Tastatur, Maus, Bildschirm, Drucker). Bei der Software unterscheidet man das *Betriebssystem* (Betriebssystemkern, Kernel), die *Systemsoftware* und *Anwendungsprogramme*.

1.1. **Das Betriebssystem.** Das Betriebssystem kann man von zwei Gesichtspunkten aus betrachten: als „idealisierter Computer“ und als „Ressourcenverwalter“.

- Das Betriebssystem als idealisierter Computer: Die Programmierung der Hardware ist kompliziert, es ist leicht dabei Fehler zu machen. Um den Programmierern von Anwendungssoftware die Arbeit zu erleichtern stellt das Betriebssystem eine abstraktere Sicht auf den Computer zur Verfügung.
- Das Betriebssystem als Ressourcenverwalter: Das Betriebssystem teilt die Systemressourcen unter den Anwendern auf. Z.B. kann immer nur ein Programm auf den Drucker zugreifen, alle anderen Druckaufträge müssen warten.

1.2. **Systemsoftware.** Das Betriebssystem stellt eine Schnittstelle für andere Programme zur Verfügung. Damit der Benutzer mit dem Computer interagieren kann, muss ein Programm die Benutzereingaben in eine für das Betriebssystem verständliche Form umwandeln. Dies geschieht entweder durch eine textbasierte Benutzeroberfläche (*Shell*, *Kommandointerpreter*) oder durch eine *graphische Benutzeroberfläche* (GUI, Graphical User Interface).

Auch der Betriebssystemkern nimmt Systemprogramme in Anspruch um gewisse Funktionen auszuführen. Beim Bootvorgang wird zuerst der Betriebssystemkern gestartet; der Kernel ruft dann Systemprogramme

auf, die Funktionen wie Anmeldung der Benutzer oder Aufbau von Netzwerkverbindungen durchführen.

Man fasst oft den Kernel und die Systemsoftware unter dem Begriff „Betriebssystem“ zusammen. Man muss dem Kontext entnehmen, welche Bedeutung gemeint ist.

1.3. **Anwendungssoftware.** Anwendungsprogramme sind diejenigen Programme, mit denen die „eigentliche Arbeit“ geleistet wird, z.B. Textverarbeitungsprogramme, Datenbanken oder Spiele.

2. Linux

Linux ist ein an Unix angelehntes freies Betriebssystem. Wie Unix ist es Multiuser/Multitasking-fähig, d.h. es können mehrere Benutzer gleichzeitig an einem Rechner arbeiten (z.B. über Netzwerkverbindungen) und jeder Benutzer kann mehrere Programme gleichzeitig ausführen.

Im Gegensatz zu anderen Betriebssystemen basiert Unix nur auf wenigen grundlegenden Konzepten:

- Benutzer,
- Dateien,
- Prozesse,
- Pipes und Filter.

2.1. **Benutzer.** Wenn mehrere Benutzer an einem Rechner arbeiten, muss man die Daten der einzelnen Anwender unterscheiden können und unterschiedliche Zugriffsrechte vergeben können. Unter Unix und Linux geschieht das über Benutzer und Gruppen.

Beim Einrichten eines Benutzers durch den Systemverwalter wird ihm ein Name und eine Benutzernummer (UID) zugeordnet, z.B. `hoelzl` und `500`. Außerdem ist jeder Benutzer in mindestens einer *Gruppe*. Zugriffsrechte auf Daten können sowohl für einen Benutzer als auch für eine Gruppe vergeben werden.

2.2. **Das Dateisystem.** Um Daten dauerhaft zu speichern oder zwischen Programmen auszutauschen verwendet man *Dateien*. Eine Datei die Anwendungsdaten wie Texte oder eine Datenbank enthält bezeichnet man als *reguläre Datei* oder nur als Datei. Um Dateien übersichtlich zu organisieren können sie in *Verzeichnissen* (Ordnern) organisiert werden. Alle auf dem Rechner gespeicherten Dateien bilden das *Dateisystem*.

Sowohl Systemprogramme als auch Anwendungsprogramme sind in Dateien gespeichert. Jeder Benutzer hat ein eigenes Verzeichnis, in dem

er seine Daten abspeichern kann, das sog *Home-Verzeichnis* oder *Login-Verzeichnis*.

Neben dem Inhalt einer Datei oder eines Ordners speichert das Betriebssystem noch weitere Informationen:

- Den Namen,
- Erstellungsdatum, Datum der letzten Modifikation oder des letzten Zugriffes,
- Besitzer der Datei (Benutzer und Gruppe): Jede Datei „gehört“ einem Benutzer und einer Gruppe,
- Zugriffsrechte (Benutzer, Gruppe und Andere).

Da Ordner sowohl Dateien als auch andere Verzeichnisse enthalten können, ergibt sich eine baumartige Struktur des Dateisystems. Unter Windows gibt es einen derartigen Verzeichnisbaum für jedes am Computer angeschlossene Laufwerk, unter Linux werden alle Laufwerke in einem einzigen Baum zusammengefasst. Die Wurzel dieses Verzeichnisbaumes nennt man *Root-Verzeichnis*. Der Name des Root-Verzeichnisses ist '`/`'. Den Weg vom Wurzelverzeichnis zu einer Datei nennt man einen *Pfad* zu dieser Datei oder *absoluten Dateinamen*. Die einzelnen Bestandteiled eines Pfades werden durch Schrägstriche '`/`' getrennt. So würde man z.B. den Pfad zu einer Datei '`passwd`' im Verzeichnis '`etc`' als '`/etc/passwd`' angeben. Der Pfad zu einer Datei '`xml.ps`' in einem Verzeichnis '`tc`', das wiederum in einem Verzeichnis '`home`' gespeichert ist, wäre '`/home/tc/xml.ps`'; der Pfad zum Verzeichnis '`tc`' wäre '`/home/tc`'. Oft gibt man bei Verzeichnissen den Pfad auch mit einem nachfolgenden Schrägstrich an: '`/home/tc/`'. Jede Datei kann durch die Angabe eines Pfades eindeutig bestimmt werden.

Da die Angabe eines absoluten Dateinames oft zu umständlich ist, hat jeder Prozess ein *aktuelles Verzeichnis* (working Directory). Gibt man nur einen Dateinamen an, so bezieht sich dieser auf die Datei mit diesem Namen im aktuellen Verzeichnis. Einen Weg, der nicht mit einem Schrägstrich beginnt bezeichnet man als *relativen Dateinamen*. Ein relativer Dateiname wird durch Voranstellen des Pfades des aktuellen Verzeichnises zu einem absoluten Dateinamen ergänzt: Ist das aktuelle Verzeichnis '`/home/tc`', so benennt der relative Dateiname '`xml.ps`' die Datei '`/home/tc/xml.ps`', der (relative) Datiname '`lisp/lmucl/reader.lisp`' bezeichnet die Datei mit dem absoluten Namen '`/home/tc/lisp/lmucl/reader.lisp`'.

Das aktuelle Verzeichnis hat den speziellen Namen '`.`', das übergeordnete Verzeichnis '`..`'. Das Login-Verzeichnis hat den Namen '`~`'.

Außer regulären Dateien und Ordnern kennt Linux noch verschiedene Arten von *speziellen Dateien* (special files):

- Gerätedateien (Device Files): Angeschlossene Geräte, z.B. Drucker oder Festplatten, können wie normale Dateien angesprochen werden. Schreibt man Daten in eine mit einem Drucker verbundene Gerätedatei, so werden diese Daten ausgedruckt.
- Symbolische Links: Diese Dateien stellen einen Verweis auf eine andere Datei dar, so dass eine Datei unter mehreren Namen angesprochen werden kann.
- (Named) Pipes zur Kommunikation zwischen Prozessen.

Für jede Datei lassen sich Zugriffsrechte vergeben. Für den Eigentümer, die Gruppe und alle anderen Benutzer kann getrennt eingestellt werden, ob

- die Datei lesbar ist
- in die Datei geschrieben werden kann
- die Datei ausführbar ist.

Dateiberechtigungen werden durch eine Dreiergruppe der Form `rwx` angegeben. Die Buchstaben stehen für lesen (read), schreiben (write) und ausführen (execute). Ist die entsprechende Berechtigung nicht vorhanden, so ist der Buchstabe durch einen Strich ersetzt: `r--` bedeutet, dass für eine Datei nur Leserechte bestehen. Berechtigungen für Verzeichnisse werden genauso abgekürzt, haben aber unterschiedliche Bedeutung:

Berechtigung	Datei	Verzeichnis
`r`	Inhalt anzeigen	Verzeichnis auflisten
`w`	Inhalt ändern	Dateien löschen
`x`	ausführen	als aktuelles Verzeichnis verwenden

Die häufigsten Berechtigungen für Verzeichnisse sind:

`---` Keine Berechtigung.
`r-x` Verbietet das Löschen oder Anlegen von Dateien, erlaubt aber die Arbeit mit vorhandenen Dateien.
`rwx` Erlaubt alle Operationen mit Dateien.

Ob eine Datei gelöscht werden kann ist also von den Berechtigungen des Verzeichnisses abhängig, in dem sich die Datei befindet, nicht von den Berechtigungen der Datei selber.

Die Berechtigungen für den Eigentümer, die Gruppe und andere Benutzer werden angegeben, indem man die Berechtigungen nacheinander schreibt, z.B. `rwxr-x---` für ein Verzeichnis in dem der Eigentümer die Rechte `rwx`, die Gruppe die Rechte `r-x` und alle anderen Benutzer keine Rechte haben.

2.3. **Prozesse.** Die Programme, die ein Computer unter Linux ausführen kann sind in Dateien auf der Festplatte gespeichert. Ein derartiges Programm ist eine statische Beschreibung der auszuführenden Arbeitsschritte. Wenn der Benutzer ein Programm ausführt wird Platz im Hauptspeicher belegt, das Programm greift auf Dateien zu und gibt Meldungen auf dem Bildschirm aus. Ein derartiges „in Ausführung befindliches“ Programm nennt man einen *Prozess*. Ein Prozess beschreibt also z.B.

- welches Programm ausgeführt wird
- in welchem Zustand sich das Programm gerade befindet, z.B., welche Anweisung gerade ausgeführt wird
- welcher Benutzer das Programm ausführt
- das aktuelle Verzeichnis, das zur Ergänzung von relativen Dateinamen verwendet wird
- die Priorität des Prozesses

Wird ein Programm, z.B. eine Textverarbeitung, von mehreren Benutzern gleichzeitig ausgeführt, so erzeugt das Betriebssystem mehrere Prozesse. Jeder dieser Prozesse hat ein eigenes aktuelles Verzeichnis, einen eigenen Zustand usw.

Jeder Prozess hat eine eindeutige Prozessnummer (PID). Über diese Prozessnummer kann man *Signale* an den Prozess senden, die ihn zu einer bestimmten Aktion veranlassen. Prozesse die keine Eingaben des Benutzers benötigen können auch als *Hintergrundprozesse* gestartet werden. Startet man einen Hintergrundprozess von einer Shell aus, so wartet die Shell nicht bis der Prozess beendet ist sondern gibt gleich eine neue Eingabeaufforderung aus. Die Ausgabe des Prozesses erscheint in der Shell. Einen Hintergrundprozess, der weder Ein- noch Ausgabe benötigt, bezeichnet man als *Dämon*.

2.4. **Pipes und Filter.** Viele Programme in Linux nehmen Daten als Eingabe, verarbeiten sie und erzeugen Ausgabedaten ohne weitere Interaktion mit dem Benutzer. Derartige Programme nennt man *Filter*. Durch Pipes kann man die Eingabe eines Filters mit der Ausgabe eines anderen Filters verknüpfen und dadurch mit einfachen Mitteln relativ komplexe Anweisungen geben. Das Symbol für eine Pipe in der Shell ist ein senkrechter Strich `|`. Der Befehl `find -name '*.tex'` durchsucht das aktuelle Verzeichnis nach Dateien mit der Endung `.tex`; der Befehl `wc -l` zählt die Zeilen seiner Eingabe. Durch

```
find -name '*.tex' | wc -l
```

erhält man also die Anzahl aller Dateien mit Endung `.tex`.