

Computergrundlagen Einführung in UNIX

Maria Fyta

Institut für Computerphysik
Universität Stuttgart

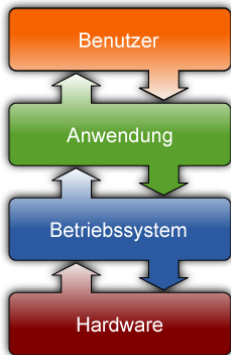
Wintersemester 2012/13



Überblick

- Was ist ein Betriebssystem?
- Architektur von Betriebssystemen (Unix, Linux) und Geschichte
- Unix Systeme - Ein(-aus)loggen - Kennwort
- Unix - Befehle
- Unix Dateisystem
- Verzeichnisse und Dateien
- Links
- Prozesse - Pipelines
- Eingabe und Ausgabe
- Root-Konto
- Betriebssystem-Shell
- Textverarbeitungsprogramme (editors) (vi, emacs)
- Graphikverarbeitungsprogramme (gimp, xmgrace)
- Versionsverwaltung von Dateien (CVS, SVN, Git)

Betriebssysteme



- Vermittler zwischen Benutzer, Programmen und Hardware
- Philosophie der Benutzerschnittstelle
- meist *nicht* graphisch (DOS, UNIX, Cisco IOS)

Betriebssysteme

- Windows: dominiert PC-Markt
- UNICES: (Linux), Mac OS X, IBM AIX, Oracle Solaris, ...
- Supercomputer Top-500 von 2010

Betriebssystem	Installationen	GFlops
Linux	455	27.162.011
other UNICES	23	1.702.295
Windows	5	412.590
andere	17	3.257.787

- UNIX-Systeme dominieren wissenschaftliches Rechnen
- Großrechner am HLRS nutzen verschiedene UNICES
- ... und auch die meisten Internet-Server

Warum?

Windows	Linux & andere UNICES
GUI integraler Teil des Systems	Terminal und/oder X11 + Desktop
Software erwartet oft Admin-Rechte	Benutzer ohne Admin-Rechte, extra root-Account
meist lokale Anwendungen	Anwendungen netzwerk-transparent
graphische Administration	per Terminal administrierbar
fernwartbar mit kommerzieller Software	out-of-the-box fernwartbar
<i>Mein Computer</i>	<i>Unser Computer</i>



Und woher bekomme ich Linux?

- es gibt nicht ein, sondern viele Linuxe — **Distributionen**
- Live-CDs: ausprobieren ohne Installation
- Dual-Boot: Installation parallel mit anderen OS
- Achtung: manchmal lässt Windows keinen Platz mehr — dann **löschen** die Installer es nach *einer* Nachfrage!

Distributionen

- (K)Ubuntu - benutzerfreundlich, mit Gnome/Unity bzw. KDE-GUI
<http://www.ubuntu.com>, <http://www.kubuntu.org>
- Xubuntu: einfacher, aber auch schneller, für Netbooks
<http://www.xubuntu.org>
- OpenSuSE: sehr benutzerfreundlich, einfache Administration
<http://www.opensuse.org>

UNIX-Grundlagen

- mehrere Programme laufen gleichzeitig: Prozesse
- *ein* Programm für *eine* Aufgabe
- Komplexität durch Verknüpfen von Prozessen
- *alles* wird durch Dateien repräsentiert
- es gibt *einen* Verzeichnisbaum
- Benutzer und Gruppen haben Rechte an diesen Dateien

Architektur des Linux Betriebesystemes (I)

- Kernel

Umfasst Gerätetreiber (Grafikkarte, Netzwerkkarte, Festplatten, usw.), Speicherverwaltung, Unterstützung für verschiedene Dateisysteme.

Den Kernel findet man in `/boot/vmlinuz` (binäre Form) und in `/usr/src/linux` (Quelldatei)

- Shells und GUIs

Linux unterstützt 2 verschiedene Befehlszeilen:

1. zeichenorientierte Befehlszeilen (command lines) (z.B. sh oder bash - Bourne shell, csh - C shell)
2. graphische Benutzeroberfläche (graphical interface-GUIs), z.B. KDE oder GNOME window manager.

(Fernzugang meistens durch eine wörtliche Befehlszeile - command line).

Architektur des Linux Betriebesystemes (II)

- Dienstprogramme (system utilities)

- Fast alle Unix Dienstprogramme sind auf Linux portiert (z.B. Befehle wie ls, cp, grep, awk, sed, bc, wc, more, usw.).
 Diese haben sehr spezifische Aufgaben, wie z.B.:
 - ◊ grep findet eine Zeichenkette in eine Datei,
 - ◊ wc zählt die Anzahl der Wörter, Zeilen und bytes in eine Datei)
 Diese Befehle können einfach kombiniert werden statt ein monolithisches Applikationsprogram zu schreiben.
- Die Linux Programme enthalten auch viele nützliche Server Programme, die daemons. Diese unterstützen Fern-Netzwerk und Verwaltungsdienstleistungen (z.B. telnetd und sshd bieten Fern-Einloggen, lpd Druckerdienstleistungen, httpd dient Webseiten, usw.). Ein daemon wird beim Systemstart automatisch hervorgebracht und "wartet" bis ein Ereignis auftritt.

Architektur des Linux Betriebesystemes (III)

- Anwendungen

- Die Linux Distributionen enthalten typischerweise viele standard nützliche Applikationsprogramme. Beispiele sind:
 - ▷ emacs (Dateiaufbereiter-editor),
 - ▷ xv (Bildbetrachter), gcc/g++ (C/C++ Compiler),
 - ▷ xfig (Zeichenpaket),
 - ▷ \LaTeX (leistungsstarke Formatierungssprache) und
 - ▷ soffice (StarOffice ein MS-Office Klon das Word, Excel und Powerpoint Dateien öffnen und schreiben kann).
- Redhat Linux kommt auch mit rpm, den Redhat Package Manager, mit dem man sehr einfach Applikationen installieren und deinstallieren kann.

Unix Systeme: Ein- und Ausloggen (I)

1. Zeichenorientierte Terminale (TTY):

Anmeldung mit telnet oder lokal. Man bekommt das Prompt:

login:

- Hier gibt man den Benutzernamen und drückt die Taste enter/return ↵. (Unix ist schreibungsabhängig). Dann benötigt das System das Kennwort:

login:mfyta

password:

- Man tippt das Kennwort, dann wieder die Taste enter/return ↵. Das Kennwort wird natürlich nicht angezeigt. Wenn man sich vertippt, fängt man wieder von vorne an. Ansonsten, bekommt man ein Shellsymbol:

\$

Mit "exit" oder "logout" oder Ctrl-D kann man sich aus dem zeichenorientierten Unix shell ausloggen.



Unix Systeme: Ein- und Ausloggen (II)

2. Graphische Terminale

Hier bekommt man ein grafisches Prompt für Benutzername und Kennwort. Wenn man sich einlogged kann man einen grafischen window manager sehen, ähnlich wie bei Microsoft Windows. Dort gibt es Menüs oder Iconen für "shell", "xterm", "console", oder "terminal emulator". So wechselt man wieder zu ein shell Prompt. Durch den Menü Optionen "Log out" oder "Exit" kann man sich wieder ausloggen.

- Tipp: Das Kennwort ändern:

Der Unix Befehl ist :

```
$ passwd ↵
```

Das System benötigt das alte Kennwort und danach das neue, das man 2 mal eintippen muss (um Fehler zu verhindern). Es ist wichtig ein sicheres Kennwort zu finden (keine Wörter aus dem Wörterbuch, mindestens 7-8 Zeichen lang, gemischte Zahlen, Buchstaben und Satzzeichen. Keine Zeichen benutzen die man nicht auf allen Tastaturen finden kann. Das Kennwort muss man geheim halten.)

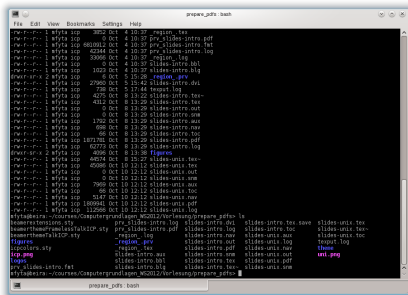
Allgemeines Format der Unix Befehle

Eine Unix Befehlszeile besteht aus dem Namen eines Unix-Befehles (der Befehl ist der Name eines eingebauten -built-in- shell Befehles, ein System utility oder eines Anwendungsprogrammes) durch seine "Argumente" (Optionen und die Ziel-Dateinamen und/oder Ausdrücken). Die allgemeine Syntax für einen Unix Befehl ist:

```
$ command -options targets ←
```

Hier "command"(Befehl) darf ein Verb, "options" (Optionen) ein Adverb und "targets"(Ziele) die direkt Objekte eines Verbes sein. In dem Fall, der Benutzer mehrere Optionen eingeben will, müssen diese nicht immer separat aufgeführt werden. Die Optionen können auch alle gemeinsam nach einem Halbgeviertstrich aufgelistet werden.

Interface: Terminal und Shell



```
poppe_pdfs_bash
ls
-rw-r--r-- 1 fyta scp    802 Oct 4 10:37 region.tex
-rw-r--r-- 1 fyta scp    0 Oct 4 10:37 pry_slides-intro.pdf
-rw-r--r-- 1 fyta scp  481012 Oct 4 10:37 pry_slides-intro.tex
-rw-r--r-- 1 fyta scp  42344 Oct 4 10:37 pry_slides-intro.log
-rw-r--r-- 1 fyta scp  35966 Oct 4 10:37 region.log
-rw-r--r-- 1 fyta scp    0 Oct 4 10:37 slides-intro.tbl
-rw-r--r-- 1 fyta scp  1023 Oct 4 10:37 slides-intro.big
-rw-r--r-- 2 fyta scp    6 Oct 5 15:26 region_psv
-rw-r--r-- 1 fyta scp  27960 Oct 5 15:42 slides-intro.dvi
-rw-r--r-- 1 fyta scp   730 Oct 5 17:44 toopt.log
-rw-r--r-- 1 fyta scp  4275 Oct 8 13:22 slides-intro.tex
-rw-r--r-- 1 fyta scp  412 Oct 8 13:29 slides-intro.tex
-rw-r--r-- 1 fyta scp    0 Oct 8 13:29 slides-intro.out
-rw-r--r-- 1 fyta scp    0 Oct 8 13:29 slides-intro.swe
-rw-r--r-- 1 fyta scp  1792 Oct 8 13:29 slides-intro.aux
-rw-r--r-- 1 fyta scp   596 Oct 8 13:29 slides-intro.nav
-rw-r--r-- 1 fyta scp   66 Oct 8 13:29 slides-intro.toc
-rw-r--r-- 1 fyta scp 187170 Oct 8 13:29 slides-intro.pdf
-rw-r--r-- 1 fyta scp  62773 Oct 8 13:29 slides-intro.log
-rw-r--r-- 2 fyta scp  4096 Oct 8 13:38 figures
-rw-r--r-- 1 fyta scp  4874 Oct 8 15:27 slides-uniti.tex
-rw-r--r-- 1 fyta scp  4598 Oct 10 12:12 slides-uniti.tex
-rw-r--r-- 1 fyta scp    0 Oct 10 12:12 slides-uniti.out
-rw-r--r-- 1 fyta scp    0 Oct 10 12:12 slides-uniti.swe
-rw-r--r-- 1 fyta scp  799 Oct 10 12:12 slides-uniti.aux
-rw-r--r-- 1 fyta scp   66 Oct 10 12:12 slides-uniti.toc
-rw-r--r-- 1 fyta scp  5147 Oct 10 12:12 slides-uniti.nav
-rw-r--r-- 1 fyta scp 18094 Oct 10 12:12 slides-uniti.pdf
-rw-r--r-- 1 fyta scp 112508 Oct 10 12:12 slides-uniti.log
~/figures -course-Computergrundlagen_MCO12/notes/prepare_pdfs_lo
~/figures$ ls
beamertextension sty      pry_slides-intro.log  slides-intro.dvi  slides-intro.tex.swe  slides-uniti.tex
beamertextensionT10CP.sty pry_slides-intro.pdf  slides-intro.log  slides-intro.toc  slides-uniti.tex
beamertextensionT10CP.sty -region_log          slides-intro.nav  slides-uniti.toc  toopt.log
figures                   -region_psv         slides-intro.out  slides-uniti.log  toopt.log
luclocart.sty             slides-intro.aux    slides-intro.pdf  slides-uniti.nav  theme
lisp                       slides-intro.tbl    slides-intro.tex  slides-uniti.pdf  umi.png
pry_slides-intro.fat      slides-intro.big    slides-intro.tex  slides-uniti.swe
~/figures -course-Computergrundlagen_MCO12/notes/prepare_pdfs_lo
```

- Shell in einem Terminal
- Terminal: Tastatur-Eingabe und Zeichen-Ausgabe
- Shell: startet und verwaltet Programme
- funktioniert genauso auch per Netzwerk

Grundlegende Dateisystembefehle

<code>man <program></code>	Hilfe, verlassen mit q
<code>info <program></code>	Ausführliche Hilfe
<code>ls <file>...</code>	Datei(en) auflisten
<code>cp [-r] <src>... <dst></code>	Dateien kopieren (-r: rekursiv)
<code>mv <src>... <dst></code>	Dateien verschieben/umbenennen
<code>rm [-r] <file>...</code>	Dateien löschen
<code>pwd</code>	aktuelles Verzeichnis ausgeben
<code>mkdir <dir>...</code>	Verzeichnis erzeugen
<code>cd <dir>...</code>	das Arbeitsverzeichnis wechseln
<code>cat <file>...</code>	Textdatei auf Terminal ausgeben
<code>less/more <file>...</code>	Textdatei seitenweise anschauen
<code>[u]mount <dev> <dir>...</code>	Ein-/aushängen von Laufwerken

Dateien

- alle Daten werden in *Dateien* gespeichert
- Dateien können in *Verzeichnissen* zusammengefasst werden
- eine Datei wird durch Ihren *Pfad* identifiziert
- besondere Pfade:

.	aktuelles Verzeichnis
..	Übergeordnetes Elternverzeichnis
~	eigenes Benutzerverzeichnis (Home)
~name	Benutzerverzeichnis (Home) des Benutzers name

- Dateien, die mit „.“ beginnen, sind versteckt
- Die Shell (und jedes andere Programm) hat ein aktuelles Arbeitsverzeichnis
- Pfade, die nicht mit „/“ oder „~“ anfangen, sind *relativ* zum Arbeitsverzeichnis

Dateien finden (I)

Mindestens 3 Unix Befehle mit den man Dateien auffinden kann.

1. `find`: Wenn man eine ungefähre Vorstellung der Verzeichnisstruktur hat

```
$ find Verzeichnis -name Zieldatei -print
```

Der Befehl sucht die Zieldatei überall unter dem gegebenen Verzeichnis.
2. `which`: Wenn man ein Anwendungsprogramm durch eine Eingabe des Names im Shell ausführen kann, kann man diesen Befehl benutzen um herauszufinden wo dieses auf der Festplatte gespeichert ist:

```
$ which ls
```

`/bin/ls`
3. `locate`: ist sehr schneller als `find` wenn man Dateien sucht deren Namen einen bestimmten Zuchbegriff haben oder wenn man in einem grossen Dateiraum sucht (z.B. unter `\`).

```
$ locate ".txt"
```

Der Befehl findet alle Dateinamen im Dateisystem die `".txt"` irgendwo in deren vollen Pfade enthalten.

Dateien finden (II)

Noch einige Beispiele

- `$ find /home -name "*.txt" -print 2>/dev/null`
sucht in allen Verzeichnissen Dateien die in ".txt" enden und gibt die passenden Dateien mit deren Pfad aus. Die Ausführungszeichen (`()`) sind nötig um Ausbreitung der Dateinamen zu vermeiden.
`2>/dev/null/` verdrängt Fehlermeldungen (z.B. wenn das Inhalt von Verzeichnissen für die der Benutzer keine Rechte hat nicht lesbar ist).
- `$ find . -name "*.txt" -exec wc -l {} {};`
zählt die Anzahl der Zeilen in jeder Textdatei in und unterhalb des aktuellen Verzeichnisses. `{}` wird durch den Namen jeder gefundene Datei ersetzt und `;` schliesst die Ausführungsklausel (`-exec`).

Dateien finden (III)

- `find` kann auch Dateien nach Typ (z.B. `-type f` für Dateien, `-type d` für Verzeichnisse), nach Berechtigungen (z.B. `-perm o=r` für alle Dateien und Verzeichnissen die by anderen gelesen werden kann, nach Größe (`-size`) finden.
- `locate` speichert alle Dateinamen des Systemes in einem Index der nur einmal am Tag aktualisiert wird. Dies bedeutet dass der Befehl keine Dateien finden kann die kürzlich erstellt wurden. Andererseits findet er Dateien die vor kurzem gelöscht wurden.

Text in Dateien finden - grep

→ grep (General Regular Expression Print)

- \$ grep Optionen Vorlage Dateien

Der Befehl sucht die angegebenen Dateien (oder die Standard Eingabe, wenn keine Dateien benannt werden) für Zeilen die zu eine bestimmte Vorlage passen.

z.B. `$ grep hello *.txt`

durchsucht alle Text-Dateien im aktuellen Verzeichnis für Zeilen die "hello" enthalten.

Einige nutzbare Optionen:

- c: zählt die Anzahl der entsprechenden Zeilen,
- i: Groß-/Kleinschreibung ignorieren,
- v die Zeile die nicht passen ausdrücken,
- n Ausdruck der Zeilennummer vor der entsprechende Zeile.



Weiter mit grep-Beispiele

```
$ grep -vi hello *.txt
```

sucht durch alle Dateien im aktuellen Verzeichnis für Zeilen die keine form des Wortes "hello" (z.B. Hello, HELLO, or hELIO).

Falls man Dateien in einem gesamten Verzeichnisbaum für ein bestimmtes Muster suchen möchte, kann man `grep` mit `find` kombinieren. Dazu benutzt man einfache Anführungszeichen um die Ausgabe von `find` in `grep` zu übergeben.

```
$ grep hello `find . -name "*.txt" -print`
```

Dieser Befehl durchsucht alle Text-Dateien in der Verzeichnisstruktur innerhalb des aktuellen Verzeichnisses für Zeilen die das Wort "hello" enthalten.

Weiter mit grep - Details (I)

Die Muster die `grep` benutzt, sind als reguläre Ausdrücke bekannt. Sowie numerische Ausdrücke werden reguläre Ausdrücke aus grundlegenden Teilausdrücke durch Operatoren kombiniert.

Der wichtigste Ausdruck ist ein regulärer Ausdruck der mit ein einzelnes Zeichen übereinstimmt. Die meisten Zeichen, einschließlich aller Buchstaben und Ziffern sind reguläre Ausdrücke die mit sich selbst übereinstimmen. Jedes anderes Zeichen mit besonderer Bedeutung kann durch “\” angegeben werden.

Weiter mit grep - Details (II)

Eine Liste von '[' und ']' stimmt mit jedes einzelne Zeichen in der Liste zu. Wenn '^' das erste Zeichen der Liste ist, dann passt es auf jedes Zeichen das nicht in der Liste ist. Man kann eine Reihe von Zeichen eingeben durch einen Bindestrich ("-") zwischen dem ersten und letzten Element der Liste. In diesem Sinne, [0-9] passt zu jeder Ziffer und [^a-z] jedes Zeichen das keine Ziffer ist. Die Zeichen '^' und '\$' sind spezielle Zeichen die den Anfang bzw. das Ende einer Zeile bestimmen. Das Zeichen '.' passt mit jedem Zeichen.

```
$ grep ^..[l-z] hello.txt
```

Der Befehl entspricht jede Zeile in hello.txt, die eine dreimäßige Reihenfolge enthält. Diese Zeichenfolge ended mit einem Kleinbuchstaben von l bis z.