Arbeiten mit Linux und an der Kommandozeile

AST, Wintersemester 2010/2011

1 Einleitung

Es wird hier ein kleiner Einblick zum Arbeiten mit der grafischen Oberfläche KDE und an der Kommandozeile unter Unix gegeben. Die Kommandozeile bietet viele nützliche Werkzeuge, um effizient zu arbeiten und komplexe Arbeitsschritte zu automatisieren, erfordert aber etwas Einarbeitungszeit. Hier soll ein erster Einblick gegeben werden.

2 Die grafische Oberfläche

Nachdem man sich an seinem Rechner unter der grafischen Oberfläche angemeldet hat, bekommt man seinen Desktop zu sehen. Unter Linux gibt es eine ganze Palette von unterschiedlichen grafischen Benutzeroberflächen, hier am ZBH ist KDE installiert. Um Programme unter KDE per Maus zu starten, klickt man das Startmenu (links unten der grüne Kreis) an und tippt den Namen eines Programms in das Suchfeld, z.B. "firefox". Alternativ kann man die Programme auch aus den Menus auswählen.

Wichtig für diese und folgende Übungen sind der File Manager Dolphin (sucht nach "file manager" im Startmenu), ein Terminal um die Kommandozeile zu benutzen (sucht nach "terminal" oder "konsole") und ein Editor (sucht nach "kate").

Alternativ könnt ihr Programme auch über ein Tastenkürzel starten: drückt Alt-F2, ihr erhaltet dann ein Eingabefeld wo ihr ein Programm starten könnt. Die Namen der Programme, die für uns heute wichtig sind, sind "dolphin" (der Filemanager), "konsole" (das Terminalprogramm) und "kate" (der Editor).

Startet alle diese Programme einmal über das Menu und einmal über Alt-F2 und macht euch ein wenig mit dem Filemanager vertraut. Wenn ihr z.B. einen USB-Stick an den Rechner anschließt, könnt ihr mit dem Filemanager darauf zugreifen und den Stick auch wieder sicher entfernen.

3 Ändern der Shell auf `bash`

Es gibt unter Unix verschiedene sogenannte "shells" (Kommandozeileninterpreter), die euch eine Eingabeaufforderung präsentieren, eure Befehle entgegennehmen, ausführen und euch die Ausgabe dieser Befehle anzeigen. Im ZBH ist standardmäßig die `tcsh` eingestellt, viel besser ist aber die `bash`, die auch die Standardshell fast aller Linux-Distributionen ist. Wir wechseln also zuerst die shell:

```
chsh -s /bin/bash
```

Ihr müsst euch anschließend ausloggen und wieder einloggen um die Änderung wirksam werden zu lassen.

4 Das Programm `konsole`

In dem Programm `konsole` habt ihr, ähnlich wie in `firefox`, die Möglichkeit mehrere Tabs zu öffnen, und könnt so in einem Fenster an mehreren Eingabeaufforderungen gleichzeitig arbeiten. Es empfiehlt sich das Terminalfenster ausreichend breit und hoch zu machen, um unschöne Zeilenumbrüche zu vermeiden. Denkt daran, dass ihr auch die Tastenkürzel für KDE und `konsole` verwenden könnt. Ein paar wichtige sind: Ctrl-Shift-n (neues Tab in `konsole`), Shift-Linkerpfeil und Shift-Rechterpfeil (ein Tab nach links/rechts springen in `konsole`).

5 Arbeiten von zu Hause mit `ssh` und `scp`

Mit `ssh` kann man über das Netzwerk von einem beliebigen anderen Rechner arbeiten. Unter Windows sind die zu `ssh` und `scp` äquivalenten Programme PuTTY und WinSCP.

```
ssh username@rechnername.zbh.uni-hamburg.de    # innerhalb des ZBHs nur rechnername nöti
scp -r username@rechnername.zbh.uni-hamburg.de:~/linuxintro .
```

Loggt euch mal Probeweise auf dem Nachbarrechner ein oder kopiert ein Verzeichnis über das Netzwerk.

6 Die ersten Schritte

Den Befehl `ls` (Kurzform für "list directory contents") benutzt man, um sich den Inhalt von Verzeichnissen anzeigen zu lassen. Gibt man kein Verzeichnis an, wird das aktuelle Verzeichnis ausgegeben.

```
ls
```

Um ein ausführliches Listing zu bekommen, übergibt man dem Befehl einen sogennanten Kommandozeilenparameter:

```
ls -l
```

Das `-l` steht bei `ls` für "long". Viele Befehle nehmen solche Parameter, mit denen sich das Verhalten des Befehls beeinflussen lässt.

Aber was ist Überhaupt das aktuelle Verzeichnis? Wo sind wir überhaupt? Dies erfährt man mit dem Befehl `pwd` ("print working directory").

```
pwd
```

Damit man nicht immer `pwd` verwenden muss, um zu wissen wo man ist, geben die meisten Shells in der Eingabeaufforderung den aktuellen Pfad an, z.B.:

```
matthies@mondsee:~/linuxintro>
```

Dies bedeutet, ich bin eingeloggt als user "matthies" auf dem Rechner "mondsee" und befinde mich im Verzeichnis " /linuxintro". Das Verzeichnis " " ist unter Unix eine Abkürzung für euer Homeverzeichnis.

Um das Verzeichnis zu wechseln verwendet man den Befehl `cd` ("change directory"). Vorher wollen wir aber noch ein Verzeichnis anlegen, und zwar mit dem Befehl `mkdir` ("make directory"):

```
mkdir linuxintro
```

```
cd linuxintro
ls -la
```

Der Befehl `ls` zeigt standardmäßig Dateien und Verzeichnisse, die mit "." anfangen, nicht an. Mit der Option `-a` (für "all") werden diese ebenfalls angezeigt. Wir sehen hier in dem Verzeichnis zwei Einträge ("." und "..") die jedes Verzeichnis hat. Das Verzeichnis "." ist das aktuelle Verzeichnis, das Verzeichnis ".." ist das übergeordnete.

Mit dem Befehl

```
cd ..
```

wechseln wir in das übergeordnete Verzeichnis. Mit dem Befehl

```
cd
```

kommen wir jederzeit und von überall direkt in unser Homeverzeichnis.

7 Tastenkürzel zum weniger Tippen

An der Kommandozeile könnt ihr durch die Liste der bisher eingegebenen Befehle mit den Pfeil-nach-oben- / Pfeil-nach-unten-Tasten navigieren. Mit Ctrl-r könnt ihr in dieser Liste alter Befehle suchen, drückt dazu Ctrl-r und tippt dann ein paar Buchstaben ein, es wird euch der letzte eingegebene Befehl angezeigt der diese Buchstaben enthält. Drückt ihr dann nochmal Ctrl-r, so wird der vorletzte darauf passende Befehl eingegeben usw. Mit der Tab-Taste wählt ihr einen Befehl aus. Wenn ihr die Suche abbrechen wollt, drückt Ctrl-c. Mit Ctrl-c könnt ihr übrigens auch die Ausführung eines Befehls abbrechen, den ihr von der Kommandozeile aus gestartet habt.

Aber am allerwichtigsten ist die Tab-Taste um halb eingegebene Dateinamen oder Befehle zu vervollständigen, da man nur so viele Buchstaben eingeben muss bis die Eingabe eindeutig ist (vorher bekommt man durch zweimaliges Tab-drücken alle Möglichkeiten angezeigt).

Um das auszuprobieren, geht ihr in das Homeverzeichnis:

```
cd
```

Nun gebt ihr `cd linu` ein, aber ohne die Eingabetaste zu drücken. Drückt ihr nun die Tab-Taste, wird der Name zu "linuxintro" vervollständigt.

8 Organisation des Dateisystems und Pfadangaben

In Unix ist das Dateisystem in Verzeichnisse organisiert, beginnend mit dem "root"-Verzeichnis "`/`" (gesprochen: "slash"). Bei Pfadangaben wie z.B. `/usr/bin/` werden Verzeichnisse durch ein `/` voneinander getrennt, `/` ist also das root-Verzeichnis welches das Verzeichnis `usr` enthält welches wiederum das Verzeichnis `bin` enthält. Eine Unterteilung des Dateisystems nach Laufwerksbuchstaben (`C:`, `D:`, usw.) wie bei einem anderen bekannten Betriebssystem gibt es nicht, alle Festplatten, USB-Sticks, etc. befinden sich in einem großen Verzeichnisbaum.

Eine Pfadangabe, die mit `/` beginnt (z.B. `/usr/local/zbh/bin`) nennt man eine absolute Pfadangabe. Es ist stets eindeutig, auf welches Verzeichnis oder auf welche Datei sich diese Pfadangabe bezieht. Im Gegensatz dazu ist eine Pfadangabe, die nicht mit einem `/` beginnt (z.B. `usr/bin`), eine relative Pfadangabe, die vom aktuellen Verzeichnis aus beginnend interpretiert wird.

Wenn ein Programm gestartet wird, merkt sich das Betriebssystem aus welchem Verzeichnis heraus es gestartet wurde (dies ist nicht das Verzeichnis in dem das Programm selbst gespeichert ist). Dieses Verzeichnis ist dann für das Programm das aktuelle Arbeitsverzeichnis ("current working directory"). Wenn das Programm eine Datei mit einer relativen Pfadangabe schreibt, so wird diese Pfadangabe vom Betriebssystem so interpretiert, als ob sie im aktuellen Arbeitsverzeichnis beginnt. Absolute Pfadangaben sind dagegen immer unabhängig vom aktuellen Arbeitsverzeichnis.

Der Name eures home-Verzeichnisses kann mit "~" (gesprochen: "tilde") abgekürzt werden, home-Verzeichnisse anderer Benutzer werden mit "~username" abgekürzt. Das aktuelle Verzeichnis kann mit "." benannt werden, das übergeordnete Verzeichnis mit "..". Ein paar Beispiele: `~/work` bezeichnet das Verzeichnis oder die Datei work in eurem home-Verzeichnis, `./work/uebung` ist äquivalent zu `work/uebung`, `../work/uebung/` bezeichnet das Verzeichnis das man erhält, wenn man vom aktuellen Verzeichnis aus in das übergeordnete Verzeichnis geht, dann in das Verzeichnis `work`, und dann in das Verzeichnis `uebung`.

Ein paar Beispiele dazu, versucht euch zu überlegen was jeder Befehl macht:

```
pwd
ls ..
cd ..
pwd
cd /
pwd
ls -la
cd ~/linuxintro
ls -l ..
ls -l /usr/bin/../
pwd
```

9 Wie bekommt man Hilfe?

```
man ls
help cd
info gcc
which ls
type ls
file /usr/bin/ls
```

10 `cp`, `mv`, `rm`, `rmdir` (kopieren, verschieben, löschen)

Dateien werden mit `cp` kopiert und `mv` verschoben oder umbenannt. Verzeichnisse können ebenfalls mit `mv` umbenannt werden, zum kopieren benutzt man `cp -r`. Zum Löschen von Dateien benutzt man `rm`, von Verzeichnissen `rm -r` oder von leeren Verzeichnissen `rmdir`.

11 `wget`, `file`, `gzip`, `gunzip`

Wir laden nun mit dem Befehl `wget` eine pdb-Datei herunter. Der Befehl `file` gibt uns allgemeine Informationen über den Inhalt einer Datei.

```
wget http://www.pdb.org/pdb/files/1igt.pdb.gz
wget http://www.pdb.org/pdb/files/1ohq.pdb.gz
wget http://www.pdb.org/pdb/files/2f5a.pdb.gz
ls
file 1igt.pdb.gz
```

Um Netzwerkbandbreite zu sparen sind die pdb-Dateien mit `gzip` gepackt worden, was auch an der Dateiendung `.gz` zu erkennen ist. Wir können die Dateien mit `gunzip` entpacken:

```
ls -lh
gunzip *.pdb.gz    # wieder packen mit: gzip *.pdb
ls -lh
```

Beim Befehl zum entpacken haben wir uns des "wildcard"-Buchstabens "`*`" bedient. Das Muster `*.pdb.gz` wird dabei von der Shell durch die Namen aller Dateien ersetzt, auf die das Muster passt. Der Stern "`*`" passt auf jede beliebig lange (auch Länge Null ist zulässig) Sequenz aus beliebigen Buchstaben.

12 `less`, `cat`, `head`, `tail` (Textdateien anschauen)

```
head 1igt.pdb    # head zeigt die ersten Zeilen (default ist 10)
tail 1igt.pdb    # tail zeigt die letzten Zeilen (default ist 10)
cat 1igt.pdb     # cat gibt eine Datei komplett aus
less 1igt.pdb    # less zeigt die Datei interaktiv an
```

12.1 Bedienung von `less`

Ihr könnt die Pfeil-nach-oben / Pfeil-nach-unten, Page-Up / Page-Down, Home / End Tasten benutzen um durch das Dokument zu navigieren. Das Programm verlasst ihr durch Drücken der Taste "q". Um in dem Dokument zu Suchen, drückt "/" und gebt eure Suchanfrage ein. Mit "n" kommt ihr zum nächsten Suchtreffer, mit "N" zum vorhergehenden. Wenn ihr "h" drückt, erhaltet ihr eine Auflistung aller Befehle.

13 `grep`, `diff`, Befehle verknüpfen (pipes)

Der Befehl `grep` kann benutzt werden, um Dateien nach Textmustern zu durchsuchen. Es werden alle Zeilen die das Suchmuster (hier MODRES bzw. HEADER) enthalten ausgegeben.

```
grep MODRES *.pdb
grep HEADER *.pdb
```

Die Ausgabe eines Programms kann mit Hilfe einer "pipe", dargestellt durch "|", zur Eingabe in ein anderes Programm weitergeleitet werden. Das Weiterleiten zu `less` ist besonders nützlich beim Betrachten von Programmausgaben, die zu lang für das Ausgabefenster sind.

```
grep REMARK *.pdb
grep REMARK *.pdb | less
```

Ebenso kann die Ausgabe aber in eine Datei umgelenkt werden oder an eine bestehende angehängt werden:

```
grep TITLE *.pdb > remarks    # Ausgabe in die Datei remarks umlenken
                              # ACHTUNG: überschreibt die Datei falls vorhanden
grep REMARK *.pdb >> remarks  # Ausgabe an die Datei remarks anhängen
```

Achtung: wird die Ausgabe auf eine bereits bestehende Datei umgelenkt, so geht der alte Dateiinhalt verloren.

Der `grep` Befehl sucht das Suchmuster an beliebiger Stelle innerhalb einer Zeile. Dies ist aber im Falle von pdb-Dateien nicht das gewünschte Verhalten, da die Record-Identifier immer am Zeilenanfang stehen:

```
grep SOURCE *.pdb
grep ^SOURCE *.pdb
```

Das Programm `diff` wird benutzt, um Unterschiede in Textdateien anzeigen zu lassen:

```
grep SOURCE *.pdb > source1
grep ^SOURCE *.pdb > source2
diff -u source1 source2
```

Es sind auch kompliziertere Anfragen mit regulären Ausdrücken möglich:

```
grep -E '^(HEADER|TITLE|SOURCE|EXPDTA)' *.pdf | less
```

Die folgenden beiden Optionen sind nützlich, um lediglich alle Dateien aufzulisten, die ein Muster enthalten bzw. nicht enthalten:

```
grep -l MODRES *.pdb
grep -L MODRES *.pdb
```

14 `find`

Mit dem Befehl `find` können wir nach Dateien suchen, die bestimmte Kriterien erfüllen. In diesem einfachen Beispiel suchen wir alle Dateien in unserem Homeverzeichnis, die auf ".pdb" enden.

```
find ~ -name '*.pdb'
```

15 Shellvariablen

In der Shell gibt es sogennannte "environment variables" (dt.: Umgebungsvariablen), die ihr in Befehlen verwenden könnt.

```
echo $HOME
ls $HOME
echo $HOST
```

Diese Variablen wurden schon vorher gesetzt. Man kann auch Variablen einen Wert zuweisen:

```
a="hallo welt"
echo "$a"
```