Linux, Knoppix, Mac OS X, Open Source: Vorteile von Unix et al. in Chemie & Biologie

Teil 2: Shells & Files & Folders

Röbbe Wünschiers

Nachdem Ihnen die Einführung im November-Heft der CLB Appetit auf Unix und seine Derivate (Linux, Mac OS X, CygWin) gemacht hat, wollen wir heute anfangen, mit dem System zu arbeiten. Wie bereits in Teil 1 erwähnt, werden wir uns nur mit der Shell (auch Konsole oder Terminal genannt) beschäftigen. Mit einem Fenster also, das wie ein DOS Fenster aussieht. Warum? Nun, nur hier können wir die wahre Stärke von Unix ausspielen. Die graphische Oberfläche von Unix, das X-Windows System mit den Desktopsystemen KDE oder Gnome, gleicht dagegen weitgehend der Windows Welt. Also, auf geht's...

Einloggen und los geht's...

Unix ist ein sicheres System, d.h. als erstes muss man sich als berechtigter Benutzer zu erkennen geben. Das heißt Sie benötigen einen Benutzernamen und ein Passwort. Falls Sie mit Mac OS X, CygWin oder Knoppix arbeiten, dann entfällt dieser Schritt, da Sie schon beim System angemeldet sind. Dann müssen Sie einfach, wie in Teil I beschrieben, den Terminal öffnen.

Terminal-1

```
01    login as: Freddy
02    Freddy@134.95.189.1's password:
03    Last login: Mon Mar 17 14:04:
      07 2003 from 134.95.189.37
04    [Freddy@Nukleus Freddy]$ PS1="$ "
05    $ date
06    Wed Dec 10 21:56:20 CET 2003
07    $ pwd
08    Freddy
09    $
```

In Terminal 1 meldet sich der Benutzer namens *Freddy* an. Nach Eingabe des Passworts erscheint, abhängig vom Unix System, eine Nachricht. In Terminal 1, Zeile 3 ist der Nachricht zu entnehmen, wann und von welchem Rechner Freddy sich zum letzten Mal eingeloggt hat. Dann erscheint die Eingabezeile 4. Diese sieht, wiederum je nach Unix System, unterschiedlich aus. Hier besagt sie, dass der Benutzer *Feddy* jetzt am Rechner namens *Nukleus* angemeldet ist (lies: Freddy at Nukleus) und sich im Verzeichnis *Freddy* befindet. Wie gesagt, bei Ihnen wird dies anders aussehen. Nun geben Sie das folgende Kommando ein: `PS1="$ "` (vergessen Sie nicht das Leerzeichen) und drücken Sie ENTER. Nun ist der Text vor dem Dollarzeichen verschwunden. *PS1* ist eine Variable. Sie enthält den

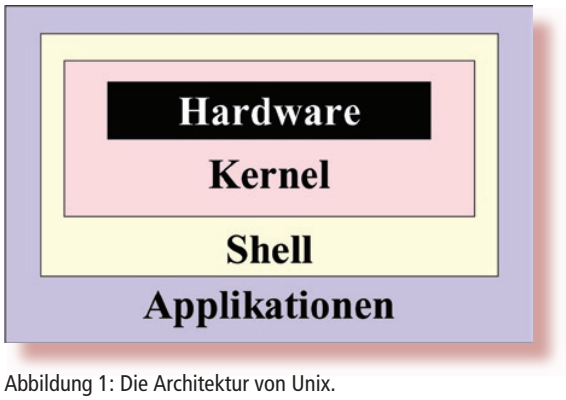


Abbildung 1: Die Architektur von Unix.

Code für das Eingabezeichen. Mit dem Gleichzeichen weisen wir der Variablen einen neuen Wert zu, und zwar ein Dollar- und ein Leerzeichen. Text wird in der Regel in Anführungszeichen gesetzt. Probieren Sie andere Eingabezeichen aus! In Zeile 5 in Terminal 1 wenden wir das aus Teil 1 bekannte Kommando `date` an. In Zeile 6 erscheint das Ergebnis und wir landen in Zeile 7. Das Kommando `date` ist ein Shell Programm, das wir ausführen indem wir es durch Eingabe des Kommandos aufrufen und ENTER drücken. Das Ergebnis liefert das Programm auf den Bildschirm. Lassen Sie uns ein weiteres Kommando ausführen: `pwd` (print working directory). Der Aufruf des Befehls ist in Zeile 7 und die Ausgabe in Zeile 8 in Terminal 1 gezeigt. Wir befinden uns also in einem Verzeichnis namens *Freddy*. Bevor es weiter geht noch ein paar Informationen zur Shell und zum Dateisystem.

Die Shell

Der Terminal ist unsere Tür zu Unix. Abbildung 1 zeigt die grundsätzliche Architektur von Unix. Der Kernel (Kern) des Unix Betriebssystems kontrolliert die Hardware (Festplatte, Arbeitsspeicher, Tastatur, Bildschirm, usw.). Die neueste Hardware (Graphikkarten, Wireless LAN, usw.) werden also, wenn überhaupt, dann nur vom neuesten Kernel unterstützt. Unter www.linux.de kann der neueste Kernel heruntergeladen werden. Die derzeit stabile Version ist 2.4.23. Auf die Installation eines neuen Kernels kann ich hier aber nicht eingehen. Der Kernel wiederum wird von der Shell (Schale, Muschel) kontrolliert und übergibt Nachrichten an die Shell. Schließlich gibt es all' die kleinen und großen Programme (Applikationen), die zum Teil über die Shell, zum Teil direkt mit dem Kernel kommunizieren. Der Kernel enthält auch die Gerätetreiber für die Hardware.

Die Shell ist unsere Welt. Doch was ist die Shell genauer. Wir wissen bereits, dass sie uns über den Terminal bzw. die Konsole zugänglich ist. Wie wir in Terminal 1 gesehen haben, führen wir in der Shell Kommandos aus. Die Shell wird deswegen auch als Kommando Interpreter bezeichnet. Es können aber auch, wie wir später noch sehen werden, mehrere Kommandos zusammen in einer Datei abgespeichert und gemeinsam ausgeführt. Das nennen wir ein Programm. Darum ist die Shell auch eine Programmiersprache. Kennen Sie sich mit DOS aus? Die Shell entspricht der *command.com* und der Programmiersprache *Batch*. Nun gibt es in Unix nicht nur eine Shell, sondern eine Vielzahl von Shells. Die erste Shell wurde 1978 von Steve Bourne von den Bell Laboratories entwickelt. Diese Shell heißt Bourne Shell und hat das Kürzel `sh`. Es gibt noch eine Menge weiterer Shells wie die C Shell (`csh`), die Korn Shell (`ksh`) oder die Bash Shell (`bash`). Letztere ist die Shell mit der wir arbeiten werden. Ich habe die Shell Kürzel wie Befehle notiert – tatsächlich können wir die Shells einfach durch Eingabe ihres Kürzels aktivieren (Abbildung 2).

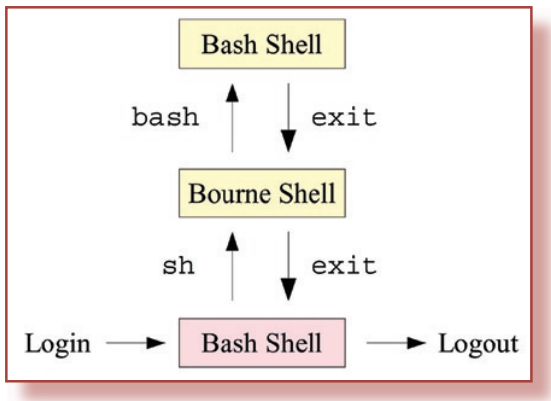


Abbildung 2: Muscheltausch. Nach dem Login landet man in der standardmäßig eingestellten Shell, hier die Bash Shell. Mit den Befehlen `sh` oder `bash` kann man weitere Shell-Ebenen öffnen. Mit dem Kommando `exit` gelangt man wieder zur vorhergehenden Shell. Nur von der Shell die man Initial geöffnet hat, kann man sich mit dem Befehl `logout` wieder ausloggen.

Terminal-2

```
01    $ echo $0
02    bash
03    $ echo $SHELL
04    /bin/bash
05    $ cat /etc/shells
06    # List of acceptable shells
      for chpass(1).
07    # Ftpd will not allow users to
      connect who are not
08    # using one of these shells.
09
10    /bin/bash
11    /bin/csh
12    /bin/sh
13    /bin/tcsh
14    /bin/zsh
15    $ sh
16    sh-2.05a$ echo $0
17    sh
18    sh-2.05a$ exit
19    exit
20    $
```

In Terminal 2 lernen wir etwas mehr über die Shell kennen. Bei den meisten Unix Systemen enthält die Variable *$0* den Namen der aktuell genutzten Shell und die Variable *$SHELL* den Pfad zu dieser Shell (weiter unten werden wir sehen was der Pfad ist). Mit dem Kommando `echo` können wir uns den Inhalt einer Variablen anzeigen lassen. Also, probieren wir es aus. Mit `echo $0` in Zeile 1 lassen wir uns die genutzte Shell anzeigen: in meinem Fall ist dies die Bash Shell. Den Pfad lassen wir uns mit `echo $SHELL` anzeigen. In Zeile 5 nutzen wir den Befehl `cat` (concatenate) um uns den Inhalt einer Datei anzeigen zulassen, welche die Pfade von allen installierten Shells enthält. Diese Datei liegt in dem Ordner */etc* und heißt *shells*. Der vollständige Befehl lautet also: `cat /etc/shells`. Die Ausgabe dieses Befehls umspannt die Zeilen 6-14. Auf einige Kommentare folgen in den Zeilen 10-14 die installierten Shells. Die Kommentare sind optional – es kann also sein, dass sie bei Ihrer Ausgabe fehlen. Wir können die Shell einfach durch Eingabe ihres Namens aktivieren (Abbildung 2). Dies tun wir in Zeile 15. Wir sehen auch, dass sich das Eingabezeichen in `sh-2.05a$` ändert. In Zeile 18 verlassen wir die Bourne Shell wieder mit dem Kommando `exit`.

Dies war eine kleine Exkursion in die Welt der Shells. Nun wissen Sie wie man die aktuelle Shell feststellt und wie man die Shell wechselt. Sofern installiert, können Sie also immer in die von mir favorisierte Bash Shell wechseln.

Das Dateisystem

Die Verzeichnisse (Ordner, Folder, Directories) und Dateien (Files) geben jedem Betriebssystem seine Struktur (Abbildung 3). Merken sollte man sich vor allem zwei Verzeichnisse: das Root- und das Home-Verzeichnis. Dem Root-Verzeichnis „/“ entspricht in der Windows Welt das „*C:*“ Verzeichnis. Das Home-Verzeichnis ist das Verzeichnis in dem Sie sich automatisch nach dem Einloggen befinden. Bei Freddy ist dies *Freddy* und der Pfad lautet */home/Freddy* (in Mac OS X lautet der Pfad */Users/Freddy*). Sie sehen, der von Windows bekannte Backslash (\) ist bei Unix

Verzeichnis	Inhalt
/bin	Essentielle System Programme
/boot	Kernel und wichtige boot-Dateien
/dev	Geräte Datein wie Drucker oder USB
/etc	Systemweite Konfigurationsdateien
/home	Home Verzeichnisse der User
/lib	Wichtige System Dateien
/lost&found	Unzuordbare Dateien nach Systemabstürzen
/mnt	Externe Medien wie Laufwerke
/proc	System Informationen
/root	Home Verzeichnis des System Administrators
/sbin	Administrative Programme
/tmp	Temporäre Dateien
/usr	Statische Datein wie Programme
/var	Variable Dateien wie Log-Dateien

Abbildung 3: Dateisystem. Inhalt der wichtigsten Unix-Verzeichnisse.

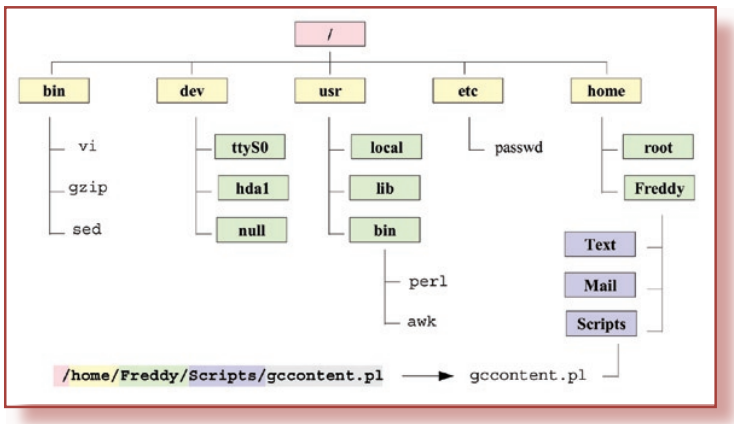


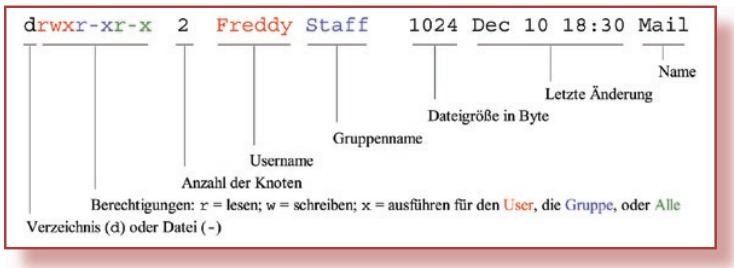
Abbildung 4: Dateisystem. Die höchste Ebene bildet das Root-Verzeichnis (/). Von der nächsten Ebene ist nur eine Auswahl von Verzeichnissen gezeigt (siehe auch Abb. 6). Das Home-Verzeichnis des Benutzers *Freddy* liegt im Verzeichnis *home*, das wiederum im Root-Verzeichnis liegt. Im Home-Verzeichnis von Freddy existieren die 3 Verzeichnisse *text*, *mail* und *scripts*. Der Pfad zu der Datei *gccontent.pl* lautet: */home/Freddy/scripts/gccontent.pl*.

ein Slash (/). Gemäß Abbildung 4 gibt der Pfad den Weg vom Root-Verzeichnis zur Zieldatei oder dem Zielordner an.

Terminal-3

```
01    $ pwd
02    /home/Freddy
03    $ ls
04    Mail   Scripts  Text  info.txt
05    $ ls -l
06    total 8
07    drwxr-xr-x   2   Freddy Staff
      1024 Dec 10 18:30 Mail
08    drwxr-xr-x   2   Freddy Staff
      1024 Dec 10 18:30 Scripts
09    drwxr-xr-x   2   Freddy Staff
      1024 Dec 10 18:30 Text
10    -rw-r--r--   1   Freddy Staff
      18 Dec 10 18:31 info.txt
11    $ cd Text
12    $ ls
13    $ pwd
14    /Users/rw/Freddy/Text
15    $ cd ..
16    $ pwd
17    /Users/rw/Freddy$
18    $
```

Abbildung 5: Berechtigungen. Unix ist ein multi-user Betriebssystem. Daher wird jeder Benutzer einer Gruppe zugeordnet. Die Zugangsberechtigungen können individuell verändert werden. Auf diese Weise kann man den Zugang zu Dateien und Verzeichnissen reglementieren.



In Zeile 1 in Terminal 3 prüfen wir zunächst in welchem Verzeichnis wir uns befinden. Dazu nutzen wir wieder den Befehl `pwd`. Wir, bzw. Freddy, befindet sich im Home-Verzeichnis. Was befindet sich alles im Home-Verzeichnis? Um dies anzuzeigen verwenden wir in Zeile 3 den Befehl `ls` (list). Aber wie können wir erkennen, ob es sich um Dateien oder Verzeichnisse handelt? Mit dem Befehl `ls -l` (list as list) können wir uns weitere Details anzeigen lassen. Der Anhang `-l` wird als Option oder Parameter bezeichnet. Solche Optionen bestehen fast immer aus einem einem einzelnen Buchstaben und spezifizieren die Art und Weise, wie ein Befehl ausgeführt wird. Oft gibt es eine Vielzahl von Optionen für einen Befehl, die dann einfach aneinander gereiht werden. So listet `ls -la` auch sonst versteckte Dateien und Verzeichnisse auf (`a` = all). Die Ausgabe ist in den Zeilen 6-10 in Terminal 3 wiedergegeben. Abbildung 3 zeigt die Bedeutung der einzelnen Informationen zu jeder Datei. Besonders interessant sind die Zugriffsrechte (Abbildung 5). Die Datei *info.txt* (Zeile 10) kann nur von dem Benutzer Freddy verändert werden (`w` = write). Dagegen dürfen alle anderen Benutzer die Datei lesen (`r` = read). Außerdem lernen wir, das der Benutzer Freddy zu der Gruppe Staff gehört. In Unix gehört jeder Benutzer einer Gruppe an. Dies liegt daran, dass Unix ein *multi-user* Betriebssystem ist. Mehrere Benutzer können gleichzeitig in das System eingeloggt sein. Über die Gruppenzugehörigkeit kann der Zugang zu bestimmten Systemresourcen oder Dateien geregelt werden. Zeile 9 in Terminal 3 zeigt uns, das *Text* ein Verzeichnis ist. Mit dem Befehl `cd` (change directory) wechseln wir in Zeile 11 in dieses Verzeichnis und listen in Zeile 12 mit dem Kommando `ls` dessen Inhalt auf; es ist leer. In Zeile 14 überzeugen wir uns mit dem Befehl `pwd` davon, dass wir in das Verzeichnis Text gewechselt sind. Mit `cd ..` (beachten Sie das Leerzeichen) gelangen wir wieder in das übergeordnete Verzeichnis, also wieder zurück ins Home-Verzeichnis von Freddy. Dies ist nochmal in Abbildung 6 verdeutlicht. Nun werden wir sehen, wie man Verzeichnisse und Dateien anlegt.

Terminal-4

```
01    $ ls
02    Mail  Scripts  Text  info.txt
03    $ mkdir Test
04    $ ls
05    Mail   Scripts   Test   Text
      info.txt
06    $ cd Test
07    $ touch neuer.file
08    $ ls
09    neuer.file
10    $ cd ..
11    $ rm Test
12    rm: Test: is a directory
13    $ rm -r Test
14    $ ls
15    Mail  Scripts  Text  info.txt
16    $
```

Der entscheidende Befehl zur Erstellung eines neuen Verzeichnisses heißt `mkdir` (make directory). Wir wenden ihn in Zeile 3 in Terminal 4 an, um das Verzeichnis *Test* zu erstellen. Wir wechseln dann in Zeile 6 in das neu erstellte Verzeichnis und verwenden in Zeile 7 den Befehl `touch` um eine Datei zu erzeugen. Dies ist ein kleiner Trick, der nur für Übungszwecke sinnvoll ist. `touch` wird eigentlich dazu verwendet eine Datei zu berühren und dadurch ihr Datum auf das aktuelle Datum zu setzten. Wenn die Datei noch nicht existiert, dann wird sie erstellt. Die Datei ist dann leer und belegt 0 Byte (testen Sie das mit `ls -l`). Wir wechseln wieder in das Home-Verzeichnis und löschen in Zeile 11 das neu erstellte Verzeichnis, samt Datei, mit dem Kommando `rm` (remove). Upps – wir erhalten eine Fehlermeldung. `rm` kann keine Verzeichnisse mit Inhalt löschen. Und selbst ein leeres Verzeichnis hat noch einen Inhalt: nämlich die beiden Spezialverzeichnisse `.` und `..` (Abbildung 6). Daher verwenden wir den Befehl `rm` mit der Option `-r` (recursively). Wie Zeile 15 zeigt, hat es diesmal geklappt.

History & Auto-Completion

Die Bash Shell bietet einige nette Funktionen, die uns das Leben erleichtern. Dazu gehören die History und Auto-Completion Funktionen. Drücken Sie ein paar Mal [↑]. Sie sehen die letzten Befehl, die Sie verwendet haben. Das ist die History Funktion. Mit [↓] können Sie vorwärts scrollen. Durch drücken von [ENTER] führen Sie den angezeigten Befehl aus. Die Pfeiltasten [→] und [←] und [BackSp] können verwendet werden, um den angezeigte Befehl vorher zu editieren.

Die Auto-Completion Funktion wird mit [TAB] bedient. Wenn Sie einen Befehl oder einen Verzeichnis- oder Dateinamen noch nicht vollständig ausgeschrieben haben, dann können sie durch einmaliges oder zweifaches drücken (in schneller Folge) von [TAB] den Befehls- oder Dateinamen vervollständigen lassen. Gibt es nur eine Möglichkeit der Vervollständigung, dann reicht ein einmaliges drücken von [TAB]. Ansonsten werden nach zweifachem drücken von [TAB] alle offen stehenden Möglichkeiten angezeigt.

Logout

Nach dem Arbeiten mit der Shell sollten Sie sich immer ausloggen. So wird verhindert, dass jemand anders an Ihren Daten herum spielt. Das entsprechend Kommando lautet `logout`. Generell sollten Sie einen Computer nach getaner Arbeit nie einfach abschalten, sondern immer „herunterfahren“. Alle Prozesse werden dann ordnungsgemäß beendet. Andernfalls kann es zu Fehlern in der Datenstruktur kommen. In dieser Beziehung ist Unix anfälliger als Windows. Von der Shell aus können Sie den Computer mit dem Befehl `halt` herunterfahren.

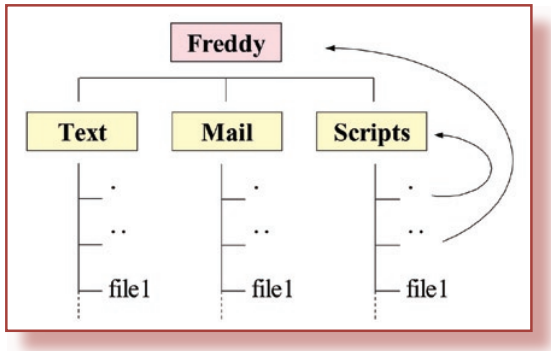


Abbildung 6: Besondere Verzeichnisse. Das Verzeichnis .. verweist auf das übergeordnete Verzeichnis, während das Verzeichnis . auf sich selbst verweist. Diese Spezialverzeichnisse werden automatisch angelegt und können nicht gelöscht werden.

In dieser Ausgabe behandelte Befehle

`pwd`	aktives Verzeichnis anzeigen
`date`	Zeigt das aktuelle Datum an
`ls`	Verzeichnisinhalt ausgeben
`echo`	Ausgabe von Text oder Variabelinhalten
`logout`	logout
`halt`	Computer herunterfahren
`cat`	Dateiinhalt ausgeben
`mkdir`	Verzeichnis erstellen
`rm`	Verzeichnis oder Datei löschen
`cd`	Verzeichnis wechseln
`touch`	Datei anlegen oder Datum ändern
`bash`	Aktiviert die Bash Shell
`sh`	Aktiviert die Bourne Shell
`exit`	Aktive Shell verlassen

In der nächsten Ausgabe werden Sie u.a. sehen wie man Hilfe zu Befehlen erhalten kann, wie Files kopiert und verschoben werden und was Wildcards sind.

Konventionen

Wenn immer im Text Befehle oder erforderliche Eingaben vorkommen, so werden sie in `Courier` gedruckt. Ebenso werden die Beispielkonsolen (Terminals) und Programme in `Courier` gedruckt und grün bzw. rot hinterlegt. Im Terminal beginnt eine Zeile mit dem Dollarzeichen „$“. Davor steht je nach Konfiguration und System entweder der Benutzername oder sonst was. In den Terminals im Text wird dies fortgelassen und nur das Dollarzeichen gedruckt. Jede Zeile ist durch eine Zeilennummer gekennzeichnet, auf die im Text Bezug genommen werden kann. Tastaturbefehle werden in KAPITÄLCHEN gedruckt bzw. durch Tastensymbole wiedergegeben. Manchmal ist es notwendig, ein Leerzeichen eindeutig zu kennzeichnen um Fehlern vorzubeugen. In diesen Fällen steht das offene Quadrat ☐ für ein Leerzeichen.