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

Teil 5: Kleine Helferlein

Röbbe Wünschiers

Bevor wir uns im nächsten Teil ausführlich mit regulären Ausdrücken beschäftigen, möchte ich Ihnen in dieses Mal, quasi zum Luft holen, kleine nützliche Helferlein vorstellen.

Kalender

Es ist Zeit für die Planung des Sommerurlaubs. Auch dabei kann Ihnen Unix/Linux ein klein wenig helfen. Wie in Terminal 1 gezeigt, können Sie sich mit dem Befehl `cal` einen Kalender des aktuellen Monats anzeigen lassen.

Termial 1

```
$ cal
      March 2004
 S  M Tu  W Th  F  S
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
$
```

Der Befehl kennt nur wenige Optionen. `cal` zeigt den aktuellen Monat des aktuellen Jahres an. `cal 2005` zeigt einen Kalender für das kommende Jahr an. `cal -j` (*julian dates*) zeigt ebenfalls einen Kalender des aktuellen Jahres an, allerdings sind die Tage vom 1. Januar an durch gezählt. Mit `cal 7 2008` können Sie sich den Juli im Jahre 2008 ansehen. Frohe Ferien.

Shell-Prompt

In Teil 2 der Unix/Linux Serie (CLB 12/2003) haben Sie bereits die Grundlagen für den Umgang mit der Bash-Shell kennen gelernt. Dabei haben wir auch über das Eingabezeichen (*shell prompt*) gesprochen. Das ist jenes Zeichen, oder Zeichenkette, das am Anfang jeder Zeile steht. In den Beispielterminalen in dieser Ausgabe ist dies immer das Dollarzeichen. Viel häufiger besteht das Eingabezeichen aber aus einer Zeichenkette die neben dem Dollarzeichen ihren Benutzernamen und den Rechnernamen, in den Sie eingeloggt sind, enthält; etwa: `[Freddy@Nukleus]$`. Ich habe Ihnen bereits in Teil 2 dieser Serie gezeigt, dass Sie das Eingabezeichen selbst einstellen können. Dazu mussten wir den Inhalt der Shell-Variablen *PS1* verändern. Generell ist zu sagen, dass die Funktionalität der Bash-Shell, sowie vieler anderer Programme, durch den Zustand so genannter Shell-Variablen gesteuert wird. Lassen Sie sich zunächst den aktuellen Inhalt von *PS1* anzeigen. Dies erreichen Sie mit dem Befehl `echo $PS1`. Den `echo` Befehl haben wir bereits in Teil 3 (CLB 01/2004) verwendet um die Funktion von Wildcards zu testen. Den Inhalt von Shell-Variablen zeigt `echo` an, wenn Sie dem Variablennamen ein Dollarzeichen voranstellen. Nach der Ausführung des eben beschriebenen Befehls sehen Sie wahrscheinlich etwas von der Art: `[\u@\h]$` . Die Backslash- (\) Buchstaben-Kombinationen bedeuten: \u = Benutzer (*user*) und \h = Rechnername (*host*). Des weiteren gibt es: \t = aktuelle Zeit (*time*), \d = aktuelles Datum (*date*), \w = Pfad zum aktuellen Verzeichnis (*working directory*), \W = aktuelles Verzeichnis (*working directory*) und \$ = das $-Zeichen für normale und das #-Zeichen für den Systemadministrator (*root*). Natürlich können Sie diese Zeichenketten mit gewöhnlichem Text verknüpfen, etwa: `Hallo \u, es ist \t Uhr. \$` .

Terminal 2

```
$ PS1="Hallo \u, es ist \t
 Uhr. \$"
Hallo rw, es ist 09:31:44
 Uhr. $ echo $PS1
Hallo \u, es ist \t Uhr. \$
Hallo rw, es ist 09:32:00
 Uhr. $ PS1="\$"
$ PS1="\[\033[34m\]\n\w
\n\[\033[0m\]\$"

~
$ cd GenoMap

~/GenoMap
$
```

Terminal 2 verdeutlicht die Vorgehensweise zur Veränderung des Eingabezeichens (Shell-Prompt). Bitte beachten Sie, dass die Zeilen 1, 2, 4 und 5 aus Platzgründen umgebrochen sind. Auf Ihrem Monitor soll alles in einer Zeile stehen. In Zeile 1 verknüpfen wir Text mit den oben genannten Zeichenkombinationen und kreieren so ein sehr langes Shell-Prompt. *PS1* ist die Variable der wir mittels des Gleichheitszeichens (=) einen Wert (Hallo \u, es ist \t Uhr. \$) zuweisen, den wir in Anführungszeichen (“) setzten müssen. Bitte beachten Sie, das dass Gleichheitszeichen nicht von Leerzeichen umgeben sein darf. In Zeile 2 sehen wir den Effekt unseres neuen Prompts. Mit `echo $PS1` fragen wir den Inhalt von *PS1* nochmal ab. Beachten Sie: Bei der Abfrage, nicht aber bei der Zuweisung, wird der Variablen *PS1* (und allen anderen Variablen) das Dollarzeichen vorangestellt. In Zeile 4 stellen wir mit dem Befehl `PS1="\$"` wieder das ursprüngliche Shell-Prompt her. Der Grund, weshalb ich Ihnen all dies zeige kommt in Zeile 5. Hier definieren wir ein Eingabezeichen, das blau gefärbt den Pfad zum aktuellen Verzeichnis und in einer neuen Zeile das Dollarzeichen anzeigt. Ich nutze fast ausschließlich dieses komfortable Prompt. Wenn Ihr Terminal einen schwarzen Hintergrund hat, dann werden Sie die blaue Farbe schlecht erkennen können. Verwenden Sie in diesem Fall “\[\033[33m\]\n\w\n\[\033[0m\]\$“. Hier ist die 34m durch 33m ersetzt – schon ist die Farbe gelb. Gelb mögen Sie nicht? Verwenden Sie 32m, dann ist die Farbe grün.

Eigene Befehle: `alias`

Sie haben nun schon eine Reihe Befehle kennen gelernt. Manche sind einfacher, andere schwerer zu bedienen oder zu merken. Mittels des `alias` Befehls können Sie beliebig komplizierte Befehle mit einem einfachen Kürzel aufrufen. Sehen wir uns ein einfaches Beispiel an. Mit `ls -l` listen Sie den Inhalt des aktuellen Verzeichnisses mit Dateidetails (`-l`) auf (siehe Teil 2 in CLB 12/2003). Terminal 3 zeigt, wie Sie mit `alias` ein Kürzel zu diesem Befehl generieren.

Terminal 3

```
01    $ alias
02    $ alias l="ls -l"
03    $ alias
04    alias l='ls -l'
05    $ l
06    total 4
07    drwxrwxrwx  2 rw  staff
       1024 Sep  4  2003 Files
08    drwxr-xr-x  6 rw  staff
       1024 Sep  4  2003 Mail
09    $ unalias l
10    $ l
11    bash: l: command not found
12    $
```

In Zeile 1 von Terminal 3 testen wir mit dem Befehl `alias`, ob bereits Kürzel definiert sind. Auf meinem System ist das nicht der Fall, bei Ihnen kann das anders sein. Dann erzeugen wir mit `alias` ein Kürzel auf den Buchstaben `l`. Diese Zuordnung gleicht weitgehend dem Verfahren, wie wir in Terminal 2 der Variablen *PS1* einen Wert zugewiesen haben. In Zeile 3 in Terminal 3 testen wir ob der `alias` Befehl erfolgreich war. Jetzt reicht der Befehl `l` aus um `ls -l` auszuführen. Mit `unalias` heben Sie das Kürzel, wie in Zeile 9 gezeigt, wieder auf. Das Kürzel `l` existiert nicht mehr und erzeugt ab sofort eine Fehlermeldung. Insbesondere bei sehr langen Befehlen ist es sehr angenehm Kürzel zu haben.

Speichern der Konfiguration

Nun haben Sie sich ein schönes Shell-Prompt und vielleicht eine Reihe Kürzel erzeugt – wenn Sie sich das nächste Mal in Ihr System einloggen ist alles weg. Um solche Änderung auch für spätere Sitzungen zu speichern, müssen sie in eine Konfigurationsdatei namens *.bash_profile* eingetragen werden. Erinnern Sie sich? Der Punkt vor dem Dateinamen macht die Datei für den `ls` Befehl unsichtbar – sie lässt sich aber mit `ls -a` (*all*) anzeigen. Diese Datei befindet sich wahrscheinlich schon in Ihrem Home-Verzeichnis (mit `cd`, ohne Angabe eines Verzeichnisses, wechseln Sie in Ihr Home-Verzeichnis), andernfalls müssen Sie sie, z.B. mit `vi`, erstellen (siehe Teil 4, CLB 02/2004). In diese Datei fügen Sie einfach die Befehle (z.B. Terminal 2, Zeile 5 und Terminal 3, Zeile 4) zur Erzeugung des Shell-Prompt oder der Kürzel ein.

Neue Befehle in dieser Ausgabe

`cal`	ruft den Kalender auf
`alias`	erzeugt ein Kürzel zu einem Befehl