Basic Linux für C

Konsolenbefehle

Mit diesen paar Konsolenbefehlen gelingt schon so manches.

pwd In welchem Ordner bin ich?

ls Was befindet sich in diesem Ordner?

cd Ordner wechseln cp Datei kopieren

cp -r Ordner (oder Datei) kopieren

rm Datei löschen rm -r Ordner löschen

wichtige Pfade

~ Heimverzeichnis

.. Ein Ordner über diesem

. Dieser Ordner

Beispiele

Wo sind wir denn?

[lucg@remus spick]\$ pwd
/home/lucg/teaching/spick

Aha, das ist also der Ordner in dem ich diesen Spickzettel schreibe. Was ist alles in dem Ordner?

[lucg@remus spick] \$ ls

spick.log

spick.pdf

spick.tex

Makefile

Löschen wir doch die nutzlose log Datei:

[lucg@remus spick] rm spick.log

und machen schnell ein Backup

[lucg@remus spick]\$ cp spick.tex spick.bk

Oops, vielleicht wäre ein anderer Name besser. Also

[lucg@remus spick] \$ mv spick.bk spick.tex.bk

Noch mal ein Blick in den Ordner zeigt

[lucg@remus spick] \$ ls

 ${\tt Makefile}$

 ${\tt spick.pdf}$

 ${\tt spick.tex}$

spick.tex.bk

GCC, der Compiler

C Quellcode (*source code*) muss kompiliert werden. Dafür nimmt man zum Beispiel gcc. In einem Schritt schreibt man

```
gcc foo.c -o foo
```

wobei foo.c der Name der Quelldatei ist und foo der gewünschte Name des Programms. Ausgeführt wird das Programm mit

```
./foo
```

Man kann auch getrennt kompilieren und linken:

```
gcc -c foo.c
gcc foo.o -o foo
```

dabei erstellt die erste Zeile eine Objektdatei foo.o. Welche in der zweiten Zeile (mit nichts sichtbarem) verlinkt wird.

Oft ist es sehr nützlich möglichst viele Warnungen zu bekommen (denn es sind fast immer Denkfehler). Das geschieht mittels

```
gcc -Wall foo.c -o foo
```

Anders als der Name vermuten lässt, aktiviert dies immer noch nicht alle Warnungen. Mehr findet man mittels

```
\mathtt{man}\ \mathtt{gcc}
```

oder Google.

Immer wieder das Gleiche...

...aber schnell und angenehm solls gehen. Mit den hoch ↑ und runter ↓ Tasten bekommt man mal ausgeführte Befehle wieder auf die Zeile. Falls es schon länger her war kann man mit C-r (das ist Kurznotation für "Control + r") nach dem Befehl suchen, wieder holt C-r drücken sucht nach immer weiter zurück liegenden Befehlen. Mit C-c kommt man da wieder raus. C-c unterbricht auch euer Programm falls es in einer endlos Scheife steckt. Oft muss man ein Programm kompilieren und dann sofort ausführen und nach einer Weile sieht die Konsole so aus:

```
[lucg@remus foo]$ gcc foo.c -o foo
[lucg@remus foo]$ ./foo
[lucg@remus foo]$ gcc foo.c -o foo
[lucg@remus foo]$ ./foo
[lucg@remus foo]$ ./foo
```

Eigentlich geht das auch in einem:

```
[lucg@remus foo] $ gcc foo.c -o foo && ./foo
und funktioniert ganz algemein
```

```
command1 && command2
```

führt command2 aus, falls command1 erfolgreich war.

Tab-completion

Tab-completion ist wirklich super! Ihr habt all eure Dateien ordentlich abgelegt und der Pfad für die aktuelle Übung sieht so aus

```
~/classes/c_programming/ex_1/hello_world.c
```

Uff, das ist aber viel zu tippen... Falsch,

```
cd cl<Tab>c_<Tab>ex<Tab>1/
```

könnte reichen. Immer wenn ihr in der Konsole Tab drückt versucht die Konsole zu vervollständigen.

Wo ist mein USB-Stick?

```
[lucg@remus ~]$ lsblk
NAMF.
                   SIZE MOUNTPOINT
                232.9G
sda
                   243M /boot
  sda1
  sda5
                174.1G
                   60G /
    ubuntu-root
                   45G /usr/local/home
   ubuntu-home
                 58.6G /usr/local/share/virtualbox
  sda6
                 57.8G
sdb
  sdb1
                 57.8G /media/grosheintz/LUC
```