# Blatt 06

*Besprechung: 14. Juni 2012*

## 1. man.c (10P)

Ihr sollt eine einfache Version von `man` schreiben. Die Aufgabe von `man` ist es, verschiedene Verzeichnisse nach einer passenden Manpage zu durchsuchen und sie, falls gefunden, formatiert anzuzeigen.

```
man SECTION NAME
```

Die Sektion und genau ein Name muessen uebergeben werden.

Gesucht werden soll die Manpage in Standardverzeichnissen, wie `/usr/local/share/man`, `/usr/share/man`, `/usr/man`, oder, falls die Umgebungsvariable `MANPATH` gesetzt ist, in den Verzeichnissen die in ihr (durch Doppelpunkt getrennt) definiert sind. (Mit *getenv()* kann man eine Umgebungsvariable abfragen.)

Die Verzeichnisse enthalten jeweils Unterverzeichnisse fuer jede Sektion. In denen ist dann die Manpage enthalten. Auf GNU/Linux ist ein ueblicher Pfad: `/usr/share/man/man1/date.1.gz`. Die Namen komprimierter Manpages enden in '.gz'. Unkomprimierte Manpages haben diese Endung nicht. Beide Arten sollt ihr unterstuetzen. Andere Komprimierungsverfahren koennt ihr ignorieren.

Folgende Pipeline formatiert eine Manpage und stellt sie im Pager dar:

```
zcat /usr/man/man1/date.1.gz | nroff –Tlp –man | \
    col –x | less –is
```

(Im Falle einer unkomprimierten Manpage ist *cat* statt *zcat* zu verwenden.)

Euer Programm soll nach der ersten passenden Manpage suchen und sie formatiert ausgeben. Verwendet *pipe()* zum Aufbau der Pipeline. Achtet darauf keine Zombies zu produzieren.

## 2. Verbesserungen (5P)

Verbessert euer Programm um Aufrufe folgender Art zu unterstuetzen:

```
man [SECTION] NAME...
```

Wird die Sektion weggelassen, soll euer Programm die Manpage in der ersten passenden Sektion liefern. Eine Liste von Standardsektionen findet ihr auf GNU/Linux-Systemen in `/etc/man.conf`. Ueberlegt euch wie ihr unterscheidet ob eine Sektion angegeben wurde oder nicht.

Werden mehrere Namen angeben, dann sollen diese Manpages nacheinander angezeigt werden.

*Seid kreativ und orientiert euch an bestehenden Implementierungen!*

Zum Einreichen der Loesung:

```
submit ss2 6 [team] [notes] man.c
```

**Fragen zur Selbstkontrolle**

- Wie erzeugt man eine Pipe zwischen zwei bereits existierenden Prozessen?
- Kann man `dup2(newfd, oldfd)` nicht einfach immer durch `close(oldfd); dup(newfd)` ersetzten? Braucht man *dup2()* ueberhaupt?
- Auf was muss man achten wenn zwei Prozesse in beide Richtungen Daten austauschen.
- Warum sind Pipes nicht full-duplex?
- Was ist SIGPIPE, wann kommt es dazu und wiso ist das Signal notwenig?
- Warum gibt es fuer die andere Richtung kein Signal als Gegenstueck zu SIGPIPE?
- Wie unterscheidet sich der Zugriff ueber einen Dateideskriptor fuer eine normale Datei vom Zugriff ueber einen Dateideskriptor fuer eine Pipe aus Programmierersicht?
- Was ist der Unterschied aus Betriebssystemsicht?