Die Dateien zur heutigen Übung finden Sie auf der Kurswebseite oder direkt unter http://ohm.hgesser.de/sp-ss2014/prakt/sp-ss2014-ue06.tgz. Das Archiv enthält die Beispielprogramme aus dem heutigen Foliensatz.

10. Pipes mit drei Prozessen

Lesen Sie den Code in der Datei `pipes3.c`, übersetzen Sie das Programm und führen Sie es aus.

11. Pipes und Umleitungen in der Mini-Shell

Erweitern Sie Ihre Mini-Shell (eine beliebige Version davon) um die Verarbeitung von Pipes, also Kommandos der Form `prog1 | prog2 | ...`. Sie können dabei davon ausgehen, dass nicht mehr als zehn Kommandos über eine Pipeline verbunden werden. Speichern Sie trotzdem alle nötigen Datenstrukturen in Arrays, die sich prinzipiell auch auf längere Pipelines erweitern lassen.

Eine Pipeline soll dabei immer als Vordergrundprozess arbeiten, Sie müssen also keine Konstruktionen wie `prog1 | prog2 | ... | progn &` zulassen.

12. Memory Mapping mit mmap()

Modifizieren Sie die Adressdatenbank (aus Übung 5, **Aufgabe 7**) so, dass sie die Datenbankdatei als Memory Mapped File mit `mmap()` verwendet. Das veränderte Programm soll nach dem Öffnen der Datei keine weiteren Zugriffe (mit `read()` oder `write()`) ausführen.

Sie beenden das Memory Mapping mit `munmap()` (siehe `man munmap`).

Lösungshinweise (Spoiler)

Lesen Sie diese Hinweise nur, wenn Sie Probleme mit der Bearbeitung der Aufgaben haben.

Zu 11)

In der Datei `pipes3.c` finden Sie ja bereits ein funktionierendes Programm, das drei Prozesse startet und über zwei Pipelines miteinander verknüpft. Diesen Ansatz können Sie verallgemeinern.

Zerlegen Sie den Gesamtbefehl durch Erkennen des Separators `|` in mehrere Einzelbefehle (die Sie separat speichern). Für jeden der Befehle müssen Sie einen eigenen Kindprozess erzeugen.

Um die Pipelines zu verwalten, können Sie ein doppeltes Array

```
int pipe_fds[10][2];  // Pipe-Deskriptor-Paare
```

deklarieren: Sie können dann die `i`-te Pipe erzeugen, indem Sie

```
pipe (pipe_fds[i]);
```

aufrufen. In `pipe_fds[i][0]` den lesenden Teil und `pipe_fds[i][0]` den schreibenden Teil der `i`-ten Pipe.

Beim Erzeugen der Pipeline können Sie drei Fälle unterscheiden:

- erster Prozess in der Pipeline (hat nur eine Pipeline-Verbindung „nach hinten“)
- mittlerer Prozess in der Pipeline (hat Pipeline-Verbindungen „nach vorn“ und „nach hinten“)
- letzter Prozess in der Pipeline (hat nur eine Pipeline-Verbindung „nach vorn“)

Zu 12)

Ersetzen Sie Kombinationen aus `lseek()` und `read()` bzw. `write()` durch Aufrufe von `memcpy()`.