Aufgabe 6: josh (14 Punkte) Bearbeitung in Zweier-Gruppen

Programmieren Sie **basierend auf der vorgegebenen** clash eine um Signalbehandlung und Pipes erweiterte Shell: **josh** (**jo**b **sh**ell). Die unten gestellten Fragen sind in **josh.txt** zu erläutern.

a) Sofortiges Aufsammeln von Zombieprozessen

clash sammelt angefallene Zombieprozesse jeweils vor der Ausgabe eines Promptsymbols auf. Ändern Sie dieses Verhalten so, dass Zombieprozesse direkt nach deren Entstehen aufgesammelt werden (**sigaction(2)**). Die Ausgabe des Exit-Status und der Kommandozeile kann unmittelbar nach Aufsammeln des Zombieprozesses auf dem Standardfehlerkanal erfolgen.

b) Signalhandler

josh soll nun das INT-Signal (**CTRL-C**, geht an **alle** Prozesse des Terminals) behandeln. In der Signalbehandlungsfunktion soll nur die Meldung "Interrupt!\n" auf den Standardfehlerkanal ausgegeben werden. Was passiert mit dem laufenden Vordergrundprozess und evtl. laufenden Hintergrundprozesse, wenn Sie CTRL-C tippen (☞ Doku)? Ändern Sie josh nun so, dass die Hintergrundprozesse das INT-Signal **ignorieren**. Was ändert sich dadurch am Verhalten bei einem **CTRL-C** (☞ Doku)?

c) Nebenläufigkeitsprobleme

Durch die nebenläufige Arbeit auf der Jobliste durch Signalbehandlungen und den eigentlichen Programmablauf kann es zu sog. *Race Conditions* kommen. Identifizieren Sie die möglichen Probleme und Lösungen und beschreiben Sie diese in der Dokumentation (**sigprocmask(2)**) und setzen Sie die Lösungen in Ihrer Implementierung um.

d) Durch Pipes verknüpfte Kommandos

Zuletzt soll die Shell noch mehrere durch Pipes verknüpfte Kommandos erlauben (**pipe(2), dup2(2)**). Hierbei können mehrere Kommandos durch ein |-Symbol getrennt in einer Kommandozeile auftauchen. Die Standardausgabe des Kommandos vor dem |-Symbol wird dann mit der Standardeingabe des Kommandos nach dem |-Symbol durch eine Pipe verknüpft. Es können mehrere Pipes in einer Kommandozeile verwendet werden. Beispiel:

```
$ cat wlist | sort -u | wc &
```

Im Beispiel wird die Standardausgabe von **cat** mit der Standardeingabe von **sort** verknüpft, dessen Standardausgabe wiederrum mit der Standardeingabe von **wc** verknüpft wird. Ist das letzte Token des letzten Teilkommandos wie im obigen Beispiel das Zeichen &, so werden alle Teilkommandos im Hintergrund ausgeführt. Die Statusausgabe soll für jedes Teilkommando einzeln erfolgen, wobei hierbei immer nur dessen eigene Kommandozeile ausgegeben wird. Bei der Ausführung im Vordergrund wird nur auf das Terminieren des hintersten Teilkommandos gewartet, die anderen Teilkommandos werden wie Hintergrundprozesse behandelt. Verwenden Sie die vorgegebene Funktion **parseCommandLine**, welche Ihnen die Kommandozeile nach Pipes getrennt in Teilkommandos zerlegt und für jedes Teilkommando das argv-Array erstellt. Die vorgegebene clash verwendet diese Funktion bereits, beachtet jedoch nur das erste Teilkommando.

Hinweise

- Ihre Shell soll mit Optimierungsstufe **-O3** des gcc-Compilers funktionieren.
- Verändern Sie keine Funktionen in den vorgegebenen **plist**- und **shellutils**-Modulen. Wenn Sie in einem der Module einen Fehler finden, teilen Sie uns dies bitte per E-Mail mit. Wir werden den Fehler dann beheben und die Vorgabe aktualisieren. Aus diesem Grund sollten Sie die Vorgaben auch nicht kopieren, sondern direkt aus dem **pub**-Verzeichnis verwenden (das vorgegebene Makefile tut dies bereits).

Abgabe: bis spätestens Montag, 28.06.2010, 17:30 Uhr