# Aufgabe 6:mini_sh (14 Punkte, Abgabe bis Do., 03.07.08 16:00)

**Die Aufgabe kann in 2er-Gruppen bearbeitet werden. Die Abgabe erfolgt durch einen Bearbeiter.**

Entwerfen und programmieren Sie eine einfache Shell mini_sh (**mini**mal **sh**ell) in einer Datei **mini_sh.c**, die Programme (im Weiteren als Kommandos bezeichnet) ausführen kann.

a) Kommandoausführung und Statusausgabe

Ihr Programm soll als Promptsymbol den String *"mini_sh> "* ausgeben. Die eingelesene Zeile soll in Kommandoname und Argumente zerlegt werden, wobei Leerzeichen und Tabulatoren als Trennzeichen dienen (**strtok(3)**). Das Kommando soll dann in einem neu erzeugten Prozess (**fork(2)**) mit korrekt übergebenen Argumenten ausgeführt werden (**execvp(3)**). Die Shell soll auf das Terminieren der Kommandoausführung warten (**wait(2)**) und den Exitstatus ausgeben. Bei der Statusausgabe soll unterschieden werden, ob der Prozess sich selbst beendet hat, oder ob der Prozess durch ein Signal beendet wurde:

1. Fall: Prozess beendet sich selbst (in diesem Beispiel mit Exitstatus 0):

```
mini_sh> ls -l
...
Exitstatus [ ls -l ] = 0
```

2. Fall: Prozess wird durch ein Signal beendet (in diesem Beispiel ein INT-Signal (SIGINT=2)):

```
mini_sh> sleep 10
Signal [ sleep 10 ] = 2
```

Nach der Ausgabe des Exitstatus soll die Shell wieder eine neue Eingabe entgegennehmen. Das Shell-Programm soll terminieren, wenn es beim Lesen vom Standardeingabekanal ein End-of-File (CTRL-D) erhält.

b) Ignorieren des INT-Signals

Erweitern Sie die Shell nun so, dass das INT-Signal (erzeugt z.B. durch Drücken von CTRL-C) von Ihrer Shell (aber nicht von den erzeugten Kindprozessen!) ignoriert wird. Sie sollten nun laufende Kindprozesse durch Eingabe von CTRL-C abbrechen können, ohne jedoch dabei Ihre Shell zu beenden.

c) Laufzeitbegrenzung für Kindprozesse

Die Shell soll Kindprozessen nun ein KILL-Signal zustellen, wenn diese länger als 10 Sekunden gelaufen sind ohne sich zu beenden. Verwenden Sie hierzu die Funktion **alarm(2)**, mit der Sie sich nach Ablauf einer wählbaren Zeitspanne ein SIGALRM zustellen lassen können. Verwenden Sie dann **kill(2)**, um dem zu lange gelaufenen Prozess das KILL Signal zuzustellen.

**Hinweise:**

- Ihr Programm muss POSIX-konform sein und mit folgenden Flags warnungs- und fehlerfrei kompilieren:
  *gcc -ansi -pedantic -Wall -Werror -D_POSIX_SOURCE -o mini_sh mini_sh.c*
- Sie können vereinfachend davon ausgehen, dass die Länge einer Kommandozeile maximal 1023 Zeichen beträgt. In anderen Fällen muss Ihre Shell nicht mehr korrekt funktionieren, es dürfen jedoch keine Pufferüberläufe auftreten.
- Wenn Sie in einem Terminalfenster z.B. durch Drücken von CTRL-C ein Signal auslösen, so wird dieses Signal allen Prozessen in der Prozessgruppe des Terminalfensters zugestellt, also inbesondere sowohl der mini_sh als auch einem evtl. gerade laufenden Kindprozess.
- Das Kommando **sleep(1)** eignet sich zum Testen der Reaktion auf Signale. Sie können mit dem Kommando **ps -U $USER | grep sleep** die PID Ihres sleep-Prozesses herausfinden und diesem mit Kommando **kill(1)** ein beliebiges Signal zustellen.
- Sie können die Exitstatusausgabe testen, indem Sie das Kommando **/proj/i4gdi/pub/aufgabe6/gdi-exit** mit dem entsprechenden Status als Parameter aufrufen.