Vorbereitung

- Booten Sie den Rechner unter Linux und melden Sie sich mit Ihrem Account an (Passwort ist evtl. die Matrikelnummer). Öffnen Sie dann ein Terminalfenster (Konsole, xterm etc.).
- Legen Sie (einmalig!) in Ihrem Home-Verzeichnis ein Unterverzeichnis `bspraktikum` an, das geht mit dem Befehl
  `mkdir bspraktikum`
- Wechseln Sie in das neue Verzeichnis:
  `cd bspraktikum`
- Laden Sie das Archiv mit den Aufgaben-Dateien herunter, das geht mit
  `wget http://hm.hgesser.de/bs-ss2009/prakt01.tgz`
- Entpacken Sie das Archiv mit `tar`:
  `tar xzf prakt01.tgz`
- Dadurch entsteht ein neues Unterverzeichnis `prakt01`, in das Sie hinein wechseln:
  `cd prakt01`

1. Prozesse und Signale

Öffnen Sie ein Kommandozeilenfenster (Konsole, `xterm` etc.) und rufen Sie darin einen Editor auf, z. B. gedit:

```
gedit &
```

(Mit & starten Sie den Editor im Hintergrund und können in der Shell weiterarbeiten.)
Suchen Sie nun mit

```
ps auxw | grep gedit
```

nach dem Prozess und finden Sie seine Prozess-ID heraus. Mit dem Befehl `kill` können Sie dem Prozess Signale senden, z. B.

- `SIGSTOP` zum Anhalten des Prozesses (`kill -STOP` *`ID`*)
- `SIGCONT` zum Fortsetzen des Prozesses (`kill -CONT` *`ID`*)
- `SIGTERM` zum Beenden des Prozesses (`kill -TERM` *`ID`* oder `kill` *`ID`*)

Probieren Sie zunächst die ersten beiden Signale aus: Schicken Sie dem Editor das `STOP`-Signal und versuchen Sie dann, im Editor-Fenster Text einzugeben. Wechseln Sie danach zurück in die Konsole und schicken Sie dem Editor das `CONT`-Signal. Sehen Sie nun den Text, den Sie im gestoppten Zustand eingegeben haben?

Beenden Sie schließlich den Prozess, indem Sie ihm das `TERM`-Signal schicken.

[☛ Lösungsblatt]

a) Suchen Sie nach einem Prozess, der Ihnen nicht gehört. Was passiert, wenn Sie diesem ein Signal (z. B. SIGSTOP) schicken?

b) Lesen Sie die Manpage zu `killall` durch. Was würde passieren (nicht ausprobieren...), wenn Sie den Befehl `killall ksh` bzw. `killall bash` (je nach Standardshell auf Ihrem Linux-System) eingeben?

2. Das Kommando ps

Rufen Sie mit `ps auxw` eine Liste aller Prozesse auf. Schlagen Sie in der Manpage zu `ps` (die Sie mit `man ps` aufrufen) die Bedeutung der verschiedenen Spalten nach. Besonders interessant sind hier: `USER`, `PID`, `%CPU`, `STAT`, `START` und `TIME` -- je nach `ps`-Version können die Spalten auch andere Namen (oder deren deutsche Übersetzungen) als Titel haben.

a) Finden Sie heraus, wie Sie die Ausgabe mit der Option `-o xxx,yyy,zzz...` so anpassen, dass Sie nur die folgenden Informationen erhalten:

- Prozess-ID
- Prozess-ID des Vaterprozesses (Parent Process ID)
- User-ID (nicht User-Name) des Prozessbesitzers
- Kommando, das ausgeführt wird

[☛ Lösungsblatt] Nennen Sie das vollständige Kommando.

b) Mit welcher Option für `ps` können Sie die Ausgabe auf Prozesse beschränken, die einen bestimmten Namen haben? Geben Sie probeweise eine Liste aller Shell-Prozesse (Name: `bash`) aus.

[☛ Lösungsblatt] Nennen Sie das vollständige Kommando.

3. jobs statt ps

Das Kommando `jobs` zeigt auch eine Prozessliste an. Es ist ein fest in die Shell eingebautes Kommando, dessen Beschreibung Sie mit `help jobs` aufrufen können. `jobs` zeigt nur Prozesse an, die Sie aus der laufenden Shell heraus gestartet haben. Probieren Sie das aus: Starten Sie einen grafischen Editor in der Shell und geben Sie dabei hinter dem Kommando `&` an, damit Sie direkt in der Shell weiterarbeiten können. Prüfen Sie dann, ob der Editor in der Job-Liste auftaucht.

[☛ Lösungsblatt] Wie ändert sich die Job-Liste, wenn Sie `disown -a` eingeben? (Hinweis: Die Hilfe zu `disown` erreichen Sie über `help disown`.)

4. fork() im C-Programm

Betrachten Sie das kleine C-Programm `forktest.c`, das zweimal `fork()` aufruft, um jeweils einen neuen Prozess zu erzeugen:

```
main() {
  printf ("vor dem Fork-Aufruf \n");
  int pid1=fork();
  int pid2=fork();
  printf ("nach dem Fork-Aufruf (pid1=%i, pid2=%i) \n", pid1, pid2);
}
```

Das Programm (das im Archiv `prakt01.tgz` enthalten ist) können Sie mit `gcc -o forktest forktest.c` kompilieren und dann mit `./forktest` (wichtig: mit führendem `./`) aufrufen.

[☛ Lösungsblatt]

a) Welche Zeilen gibt das Programm aus?

b) Erklären Sie die Ausgabe und nennen Sie die Anzahl der Prozesse, die liefen.

c) Wie viele Prozesse starten, wenn Sie einen dritten `fork()`-Aufruf einbauen?

Lösungsblatt – Übung 1

Punkte:

	Name	**Matrikelnummer**
Teilnehmer 1		
Teilnehmer 2		

Bitte schreiben Sie Ihre Antworten in die freien Bereiche.

1)

2)

3)

4)