– Praktikumsaufgabe 8 –

Thema: *Prozesserzeugung, -überlagerung und -beendigung mittels C*

Zielstellung: Erlernen der Nutzung der grundlegenden Unix-Systemrufe zur Prozessverwaltung: `fork()`, `execl()`, `wait()`, `exit()`.

1. Schreiben Sie ein C-Programm, das eine Zufallszahl als Rückgabewert übergibt. Testen Sie dann mit dem Skript aus Aufgabe 1.a) des letzten Praktikums, ob die Beschränkung auf den Wertebereich [0; 255] auch in diesem Falle gilt.

   Hinweise:

   - Die Programmbeendigung erfolgt mittels `exit()`.
   - Eine Zufallszahl im Intervall [0; RAND_MAX] erhalten Sie mit `random()`, das genauso wie `exit()` in `stdlib.h` definiert ist

2. Schreiben Sie ein C-Programm, das eine variable Anzahl von Prozessen, die jeweils eine Zeit von 10 Sekunden schlafen und sich dann beenden, startet. Die Anzahl zu erzeugender Prozesse legen Sie entweder mittels `#define` im Voraus fest oder Sie übergeben diese als Kommandozeilenparameter. Im letzteren Fall müssen Sie den Parameter von Zeichenkette in Integer wandeln, was z. B. mittels `atoi()` erfolgen kann.

   Hinweis: Testen Sie vorsichtig! Können Sie Auswirkungen auf das Systemverhalten bemerken? Ab welcher Prozesszahl?

3.* Überlegen Sie sich ein Verfahren, einen oder mehrere Prozesse für einen beobachtbaren Zeitraum in den Zombiezustand zu bringen.

4. Machen Sie sich mit `execl()` vertraut. Schreiben Sie ein C-Programm, das ein als Kommandozeilenparameter übergebenes Kommando ausführt, *ohne* einen neuen Prozess zu starten.

   Hinweis: `execl()` durchsucht nicht, wie aus der Shell gewohnt, die Pfadvariable nach einem ausführbaren Programm. Sie müssen daher dem Namen des aufzurufenden Programmes den entsprechenden Pfad, den Sie beispielsweise mit dem Kommando `which` ermitteln können, voranstellen.

5. Implementieren Sie ein Programm, das einen Sohn erzeugt, der wiederum einen Sohn erzeugt usw. Die Generationstiefe soll als Kommandozeilenparameter eingelesen werden.

   Beispiel: `prg 3` soll einen Sohn, dieser wieder einen Sohn und dieser nochmals einen Sohn (wir kommen also bis zum Urenkel) erzeugen. Beginnend mit dem Urenkel soll jeder Prozess ausgeben, in welcher Generation er sich befindet. Im Beispiel würde also

die Zahlen `3,2,1,0` in dieser Reihenfolge ausgegeben.

Hinweis:

- Nutzen Sie `wait()` zur Synchronisation.