Fakultät für Informatik und Mathematik
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Betriebssysteme, SoSe 2020

Praktische Übung 3: Prozesse

Ziele des Labortermins

- Mit Prozessen in der Shell umgehen
- Prozesse in C, im speziellen fork verstehen

Aufgabe 3.1: Verständnisfragen

a) Was ist der Effekt der Latenzreduzierung durch Pseudo-Parallelität?

b) Benennen Sie zwei Element der eigenen Maschinensicht eines Prozesses.

c) Was ist der Unterschied zwischen `SIGTERM` und `SIGKILL`?

d) Warum gibt es den Zustandsübergang `waiting` zu `running` nicht?

e) Warum gibt es den Zustandsübergang `ready` zu `waiting` nicht?

f) Es ist kein Prozess aktuell lauffähig, was macht das Betriebssystem?

g) Wovon hängt die Implementierung eines Kontextwechsels primär ab?

Aufgabe 3.2: Prozesse in der Shell (praktisch)

a) Starten Sie einen länger laufenden Prozess, zum Beispiel `sleep` (entweder in einem zweiten Terminal, oder im Hintergrund). Senden Sie dem Prozess ein Signal um es zu stoppen und danach ein Signal um es zu weiterlaufen zu lassen. Schauen Sie sich zwischendurch die Ausgabe von `ps` an. Welchen Zustand haben die Prozesse.

Hinweis: Schauen Sie sich in den man-pages der Befehle um, insbesondere die Sie in der Vorlesung kennengelernt haben um Signale zu senden.

b) Vollziehen Sie die "Entstehungsgeschichte" des Terminals in dem Sie sich gerade befinden, bis hin zum vom init-Prozess (PID 1).

c) Identifizieren Sie die PID eines Prozess Ihres Users und schauen Sie sich in `/proc/<PID>/` um. Im virtuellen Dateisystem `/proc` finden sich unter anderem Informationen zu den Prozessen. Finden Sie heraus wie oft der Prozess freiwillig und unfreiwillig einen Kontextwechsel durchgeführt hat. Was sind freiwillige und welche sind unfreiwillige Kontextwechsel?

Aufgabe 3.3: Prozesse in C (praktisch)

a) Vervollständigen Sie das Programm (`fork_print.c` im Ordner `fork` dahingehend, dass Sie ein `fork()` ausführen und in Eltern- und Kind-Prozess die Rückgabe des Aufrufs anzeigen. Führen Sie das Programm aus.

b) Starten Sie das Programm jetzt so, dass Sie einen Kernel Trace erhalten. Dabei hilft Ihnen das Skript `lttng.sh <programm>`. Schauen Sie sich mit `lttng-scope` den erstellten Trace in `trace/kernel` an.

c) Legen Sie ein weiteres Programm `fork_sleep.c` an. Führen Sie wieder einen Fork durch und starten Sie diesmal im Kindprozess das Programm `sleep` mit dem Argument `30`. Warten Sie im Eltern-Prozess darauf, dass der Kindprozess sich beendet. Erstellen Sie wieder einen Trace und analysieren ihn.

d) Legen Sie ein letztes Programm `fork_and_kill.c` an. Forken Sie wieder wie zuvor. Der Kind-Prozess soll in einer Endlosschleife landen. Der Eltern-Prozess soll 20 Sekunden warten und dann den Kind-Prozess killen. Schauen Sie sich auch wieder den Trace an.