

Hinweis: Verwenden Sie die im Pool installierte Version der Virtual Box oder ein auf Ihrem Rechner installiertes LINUX.

Zu der Virtual Box gibt es in Moodle einen Kursraum bei Prof. Regensburger, in dem alle benötigten Informationen enthalten sind. Als Benutzer wählen Sie Lars Tragl.

Als Entwicklungsumgebung können Sie eclipse, kdevelop, oder vim (oder nano), cc, make verwenden.

Bitte frischen Sie, falls notwendig, Ihre C-Kenntnisse auf, so dass Sie sicher in C programmieren können.

1) Aufgabe (Duplizieren von UNIX-Prozessen: fork()):

Hinweis: Informationen zu den genannten Funktionen erhalten Sie in den man-pages mit dem Kommando: `man <section> <function>`. Beachten Sie hierbei, dass die C-Systemfunktionen in den Sektionen 2 oder 3 beschrieben werden.

Sollten Sie mit der Beschreibung nicht weiterkommen, wenden Sie sich an den Dozenten. Falls Sie noch keine Erfahrung mit SW-Entwicklung unter UNIX haben, verwenden Sie als IDE kdevelop.

- a) Erzeugen Sie 3 Sohnprozesse mit der Funktion `fork()`. Beachten Sie die unterschiedlichen Rückgabewerte der Funktion:
 - 0: Kennzeichnet den Sohnprozess. Geben Sie hierbei eine von Ihnen vergebene Prozessnummer {1, 2, 3}, sowie die UNIX Prozessnummer, die Sie mit `getpid()` erhalten, aus.
 - -1: Kennzeichnet den Fehlerfall. Geben Sie hierbei eine Fehlermeldung mit `perror()` aus. Und verlassen Sie das Programm mit `exit(1)`.
 - >0: Kennzeichnet den Vaterprozess. Keine spezielle Aktion erforderlich.
- b) Erweitern Sie den Sohnprozess dahingehend, dass er jeweils dreimal einen kritischen und im Anschluss einen unkritischen Bereich unsynchronisiert als Simulation durchläuft. Die Bereiche sollen jeweils 1 Sekunde betreten werden. Verwenden Sie hierzu die Funktion `sleep()`.
Damit Sie das Verhalten der Prozesse untersuchen können, verwenden Sie Ausgaben auf `stdout`. Achten Sie hierbei darauf, dass jede Ausgabe mit einem `\n` terminiert wird, mit dem die Ausgabepuffer geleert werden. Andernfalls würden Ihre Beobachtungen nicht mit der Realität übereinstimmen.
- c) Interpretieren Sie die Ausgaben.

2) Aufgabe (Synchronisation von Prozessen mit Semaphoren (IPC)):

- a) Erweitern Sie nun die Lösung aus Aufgabe 1 um Synchronisationsmechanismen. Verwenden Sie hierzu den Semaphore-Mechanismus aus dem IPC-Paket. (`man 5 ipc`). Gehen Sie hierzu wie im folgenden Aufgabentext vorgeschlagen schrittweise vor. Beachten Sie hierbei unbedingt, dass Sie bei jedem Systemaufruf mögliche Fehlermeldungen des

Betriebssystem abfangen und in diesem Fall das Programm mit `perror` und `exit(1)` beenden.

- b) Erzeugen Sie sich mit der Funktion `ftok(<datei>, <int>)` einen eindeutigen Schlüssel für die zu erzeugenden Semaphore.
- c) Verschaffen Sie sich Zugriff auf einen Satz von einem Semaphor mit der Funktion `semget()`. Verwenden Sie als Zugriffsrechte: `IPC_CREAT | 0666`, damit der Semaphorsatz mit geeigneten Rechten erzeugt wird, falls er noch nicht existiert.
- d) Implementieren Sie die folgenden Funktionen:
 - `init_sem()`: Hierbei wird unter Verwendung der Funktion `semctl()` mit der Option `SETVAL` das Semaphor mit 1 („Betriebsmittel frei“) initialisiert.
 - `P(int sem_num)`: Hierbei wird mit der Funktion `semop()` versucht das Semaphor zu dekrementieren. Beachten Sie hierbei, dass als Operand ein array von Semaphoren übergeben werden muss.
 - `V(int sem_num)`: Hierbei wird mit der Funktion `semop()` versucht das Semaphor zu inkrementieren. Beachten Sie hierbei, dass als Operand ein array von Semaphoren übergeben werden muss.
 - Schützen Sie nun die kritischen Bereiche des Prozesses mit P- und V-Operationen.
- e) Interpretieren Sie nun die Ausgaben.