Übungen
zur Vorlesung
Betriebssysteme
Prof. Dr. Kornmayer

Übungen 2 - Prozesse und Threads

Aufgabe 2.1: Beschreiben sie die wesentlichen Unterschiede zwischen Programm und Prozess!

Aufgabe 2.2: Systemaufrufe unter Linux sind als Wrapperfunktionen in der statischen Bibliothek libc.a zusammengefasst. Diese Datei wird durch die Installation des C-Compiler gcc im System verfügbar.
Nach der Installation suchen Sie diese Datei im System und lösen Sie die folgenden Aufgaben unter Verwendung der Befehle (find, ar, grep, wc).

- Wie viele Wrapperfunktionen sind in der Bibliothek enthalten?
- Sind die Funktionen fork.o und creat.o auch enthalten?
  Wenn ja, an welcher Position kommen die Funktionen vor?

Aufgabe 2.3: Beschreiben Sie die Hauptbestandteile des Datenmodells eines Prozesses!

Aufgabe 2.4: UNIX Prozessverwaltung
Erzeugen Sie zwei Hintergrundprozesse. Jeder dieser Prozesse soll eine bestimmte Anzahl von Sekunden lang nichts tun (siehe sleep), dann eine Meldung ausgeben (“Hier Prozess 1“ bzw. “Hier Prozess 2“), dann nochmals dieselbe Zeit schlafen und zum Abschluss nochmals die Meldung ausgeben.

- Schreiben Sie dazu ein Shellscript, das mit der Kennziffer und den Sekundenzahlen parametrisiert ist und das Sie zweimal aufrufen.
- Geben Sie den Prozessstatus des Systems aus, während die Prozesse schlafen (siehe pr)
- Beenden Sie einen Prozess vorzeitig (siehe kill)
- Starten Sie einen Hintergrundprozess, der seinerseits drei neue Hintergrundprozesse startet. Jeder dieser drei Prozesse soll unmittelbar hintereinander 200-mal die Meldung „Hallo, hier ist Prozess x“ ausgeben, wobei x je nach Prozess „A“ „B“ oder „C“ ist. Auch hier lassen sich gut Shellscripts einsetzen.
  Was beobachten Sie bezüglich der Reihenfolge der Prozessausführung?

Dokumentieren Sie die UNIX-Kommandos, mit denen Sie die Aufgaben gelöst haben.

Aufgabe 2.5: Verwendung von fork()
Schreiben Sie unter C ein Programm, das die Funktionen fork verwendet, um ein Kindprozess zu erzeugen. Vater und Kind sollen jeweils in einer Schleife mit 500 Durchläufen jeweils den Schleifenzähler ausgeben. Lassen Sie das Programm mehrmals laufen und vergleichen Sie die Ausgaben! Erklären Sie das Verhalten des Programms.
(Es kann notwendig sein, dass Programm deutlich mehr als zweimal laufen zu lassen!)

Aufgabe 2.6: POSIX-Threads
Erstellen sie mit der POSIX Thread Library ein Programm, das die Aufgabe 2.5 mit Threads anstatt mit dem Systemaufruf fork löst.
(Unter Umständen muss in der Ausgabe-Schleife noch eine kleine Last eingebaut werden. (while-loop))

Aufgabe 2.7: Java Threads / Race Conditions

- Erstellen Sie eine Klasse, die als Container für zwei globale Variablen j und k dient (static). Initialisieren Sie die Werte j=0, k=5.
- Erstellen Sie eine Thread-Klasse Race, die in der run()-Methode 10-mal eine Schleife durchläuft, in der zuerst den Wert j gesetzt wird und anschließend die Operation k=k+j durchführt wird.
- Erzeugen Sie in der Main-Methode zwei Threads, die den Wert j auf 1 bzw. auf -1 setzen.

- Lassen Sie das Programm mehrmals laufen. Beschreiben Sie das Ergebnis!
- Erleichtern Sie nach jeder Operation innerhalb der Schleife einen Thread-Wechsel durch Verwendung von Thread.yield(). Wie ändert sich das Ergebnis?