Betriebs- und Kommunikationssysteme

WS 14/15

Prof. Dr-Ing. Volker Roth

Freie Universität Berlin

4. Übung

Ausgabe	Abgabe
7.11.14	21.11.14

Bitte bei der Abgabe Name der Mitglieder einer Gruppe, Nummer der Übung/Teilaufgabe und Datum auf den Lösungsblättern nicht vergessen! Darauf achten, dass die Lösungen beim richtigen Tutor abgegeben werden. Achten Sie bei Programmieraufgaben außerdem darauf, dass diese im Linuxpool kompilierbar sind. Nutzen Sie dazu die Flags `-std=c99`, `-Wall` und `-pedantic`. Es sollten keine Warnungen auftauchen.
Zu spät abgegebene Lösungen werden nicht berücksichtigt!

Aufgabe 1: Prozesszustände und Speicherverwaltung (2 Punkte)

(a) Ein Programm, das direkt nach dem Starten die Adresse der Mainfunktion ausgibt und dann in eine Endlosschleife verfällt, wird dreimal parallel gestartet. Es wird dreimal dieselbe Adresse ausgegeben werden. Teilen sich diese drei Instanzen einen gemeinsamen Speicher oder nicht?

Begründen Sie ausführlich, was Betriebssystem und CPU bis zum physikalischen Speicherzugriff für Mechanismen anwenden.

Hinweis:

Probieren Sie es! Sie können die Speicheradresse mit:

```
1          //print main addr
2          printf("%p\n", main);
```

abfragen.

(b) In der Vorlesung haben Sie gelernt, dass sich Prozesse während ihrer Lebensdauer in verschiedenen Zuständen befinden können:

Zustand	Bedeutung
new	Prozess wird erzeugt (kreiert)
running	Anweisungen des Prozesses werden gerade ausgeführt
ready	Prozess ist bereit zur Ausführung, wartet aber noch auf freien Prozessor
waiting	Prozess wartet auf das Eintreten eines Ereignisses, z.B. darauf, dass ein von ihm belegtes Betriebsmittel fertig wird
blocked	Prozess wartet auf ein fremdbelegtes Betriebsmittel
killed	Prozess wird (vorzeitig) abgebrochen
terminated	Prozess ist beendet, d.h. alle Instruktionen sind abgearbeitet

Gegeben seien 4 Prozesse, die zu unterschiedlichen Zeitpunkten gestartet werden und sich in die ready-Warteschlange einreihen. Der Scheduler arbeitet nach dem FIFO-Verfahren, d.h. der Prozess, der zuerst auf **ready** steht, wird zuerst bedient.

Einige Prozesse benötigen Zugriff auf den Drucker. Die zweite Zeile der Tabelle gibt an, nach wie vielen Zeiteinheiten im Zustand **running** ein Prozess auf den Drucker zugreifen möchte. Während der 5 Zeiteinheiten dauernden Druckphase wird der Prozessor für andere

Prozesse freigegeben. Nach Beendigung des Druckvorgangs reiht sich der Prozess wieder in die ready-Warteschlange ein. Der Prozess P2 wird abgebrochen, nachdem er 2 Zeiteinheiten gerechnet hat.

Prozess	P1	P2	P3	P4
Startzeitpunkt	0	1	2	3
Drucken nach Zeiteinheit	4	6	1	-
Benötigte Zeiteinheiten im Zustand running	8	10	4	5

(1) Geben Sie zu den Zeitpunkten 0, 1, 2, ... den aktuellen Prozesszustand der Prozesse P1 bis P4 an. Beachten Sie, dass ein ankommender Prozess bei seiner Ankunft den Zustand **new** annimmt und erst nach einer Zeiteinheit in einen anderen Zustand wechselt, wobei keine CPU-Zeit verbraucht wird.

(2) Zu welchen Zeitpunkten befindet sich das System im Zustand des *busy waiting*, d.h. ein Prozess wartet auf ein Betriebsmittel und blockiert derweil die CPU?

(3) Nennen Sie ein Beispiel, bei dem der Abbruch eines Prozesses sinnvoll sein kann. Wie kann man unter dem Betriebssystem UNIX Prozesse abbrechen?

Aufgabe 2: Programmieraufgabe zur Prozessverwaltung (3 Punkte)

Implementieren Sie die durch das Framework vorgegebene Prozessverwaltung, die eine Menge von Prozessen aus der Datei `prcs.dat` auslesen, und diese in eine doppelt verkettete Liste einfügen soll.

Nun soll Prozess um Prozess der Reihenfolge ihrer IDs entsprechend aus der Liste entfernt werden, wobei die übrig gebliebenen Prozesse in jeder Iteration ausgegeben werden.

Die Prozesse sind in der Datei `prcs.dat` Zeile für Zeile in der Form:

```
ProzessId, Ankunftszeit, AusfÃ¼hrungszeit
```

gespeichert.