Freie Universität Berlin

Übung 3

Betriebs- und Kommunikationssysteme
Barry Linnert, Sommersemester 2019

Ausgabe 03.05.2019
Abgabe 17.05.2019, 10:15

Bitte beachten Sie die allgemeinen Hinweise auf Übungszettel 1

Aufgabe 1: Prozesse

Begriffe

1. Beschreiben Sie kurz, was einen Prozess von einem Programm unterscheidet.
2. Was ist der *PCB*? Welche Einträge hat er und wozu dienen diese?
3. Grenzen Sie *Prozess* und *Thread* voneinander ab. Was ist ein *Task*?

Prozesszustände

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

a) 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.

b) 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?

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

Aufgabe 2: stdio in C

Lösen Sie die folgende Aufgabe mit den Mitteln der `stdio.h` Bibliothek. Führen Sie Fehlerbehandlung durch. Die Ausgaben sollen auf `stdout` ausgegeben werden, die Fehler auf `stderr`.

a)

Implementieren Sie zwei Programme, welche jeweils einen Dateinamen als Parameter erhalten:

1. `my_cat` gibt den Inhalt dieser Datei aus.
2. `my_wc` gibt die Anzahl der Zeilen, Wörter und Bytes dieser Datei aus.

b)

Passen Sie nun Ihre Implementierungen derart an, dass für den Fall, dass kein Dateiname angegeben wurde, die Eingabe von stdin eingelesen wird. Die Eingabe

```
$ ./my_cat file_name | ./my_wc
```

in einer UNIX-Shell soll die Anzahl von Zeilen, Wörter und Bytes ausgeben, die in der Datei `file_name` enthalten sind.

c)

Erweitern Sie nun Ihre Programme `my_cat` und `my_wc`. Falls mehrere Dateinamen als Parameter angegeben werden, sollen diese jeweils durch einen eigenen (Kind-)Prozess abgearbeitet werden (siehe *fork()* in `unistd.h`). Die Reihenfolge der Ausgabe soll dabei der Reihenfolge der angegebenen Dateien entsprechen (hier hilft Ihnen die `sys/wait.h` Bibliothek).

Zusatzfrage: Ist das Erzeugen von Kindprozessen hier wirklich sinnvoll? Lösen Sie die folgende Aufgabe mit den Mitteln der `stdio.h` Bibliothek. Führen Sie Fehlerbehandlung durch. Die Ausgaben sollen auf `stdout` ausgegeben werden, die Fehler auf `stderr`.

a)

Implementieren Sie zwei Programme, welche jeweils einen Dateinamen als Parameter erhalten:

1. `my_cat` gibt den Inhalt dieser Datei aus.
2. `my_wc` gibt die Anzahl der Zeilen, Wörter und Bytes dieser Datei aus.

b)

Passen Sie nun Ihre Implementierungen derart an, dass für den Fall, dass kein Dateiname angegeben wurde, die Eingabe von stdin eingelesen wird. Die Eingabe

```
$ ./my_cat file_name | ./my_wc
```

in einer UNIX-Shell soll die Anzahl von Zeilen, Wörter und Bytes ausgeben, die in der Datei `file_name` enthalten sind.

c)

Erweitern Sie nun Ihre Programme `my_cat` und `my_wc`. Falls mehrere Dateinamen als Parameter angegeben werden, sollen diese jeweils durch einen eigenen (Kind-)Prozess abgearbeitet werden (siehe *fork()* in `unistd.h`). Die Reihenfolge der Ausgabe soll dabei der Reihenfolge der angegebenen Dateien entsprechen (hier hilft Ihnen die `sys/wait.h` Bibliothek).

Zusatzfrage: Ist das Erzeugen von Kindprozessen hier wirklich sinnvoll?