Lösung zur Klausur Systeme I

vom 14.03.2011

Hinweise

Diese Lösungen können Fehler enthalten, daher immer kontrollieren, ob die Lösung wirklich richtig ist.

Aufgabe 1

a

Das erste Zeichen muss eine Ziffer zwischen 1 und 9 sein, es können weitere Ziffern zwischen 0 und 9 folgen - der reguläre Ausdruck passt also auf alle ganzen Dezimalzahlen größer Null.

b

Ohne führende Nullen: `([1-9][0-9]*[02468][2468])`, mit führenden Nullen: `[0-9]*[02468]` (auch möglich, weil in Aufgabenstellung nicht gefordert).

c

`grep`, `sed`

Aufgabe 2

a

`eintrag1` ist ein Verzeichnis, `eintrag2` ist eine Datei und `eintrag3` ist ein symbolischer Link.

b

Auf Einträge des Verzeichnisses `eintrag1` kann lesend und schreibend zugegriffen werden. Zudem können neue Einträge neu angelegt werden. Eine Lise existierender Einträge im Verzeichnis kann ebenfalls ausgelesen werden. Als Nutzer `meier` mit Gruppenzugehörigkeit `students` darf man auf die Datei `eintrag2` weder lesend, noch schreibend zugreifen.

c

Der Eintrag würde sich von `drwxrwx-` auf `drwxrws-` ändern. Die Auswirkung wäre, dass neu angelegte Ordner oder Dateien unterhalb des Verzeichnisses `eintrag1` automatisch der Gruppe `staff` gehören.

Aufgabe 3

a

- Create (`mkdir`)
- Delete (`rmdir`)
- Opendir (`ls`)
- Closedir (`ls`)
- Readdir (`ls`)
- Get Attributes (`ls`)
- Set Attributes (`chmod`)
- Rename (`mv`)

b

Verzeichnisse werden beim Anlegen oder Löschen von Dateien automatisch geändert. Ein explizites Schreiben in das Verzeichnis ist deshalb nicht unnötig - zudem ist es auch unerwünscht, weil dann Datenstrukturen, die betriebssystemintern genutzt werden, durch Nutzerprozesse verändert werden können.

Aufgabe 4

a

Kapitel 2.5 Folie 16. Hinzu kommen die Zustände `bereit/ausgelagert` und `blockiert/ausgelagert`. Neue Zustandsübergänge sind:

- `neu -> Zulassung -> bereit/ausgelagert`
- `bereit/ausgelagert -> aktivieren -> bereit`
- `bereit -> auslagern -> bereit/ausgelagert`
- `blockiert/ausgelagert -> Ereignis tritt ein -> bereit/ausgelagert`
- `blockiert/ausgelagert -> aktivieren -> blockiert`
- `blockiert -> auslagern -> blockiert/ausgelagert`

b

`Prozesse` getrennte Adressräume, geschützt gegen Zugriff anderer Prozesse
`Threads` gemeinsamer Adressraum, kein gegenseitiger Schutz (weniger Overhead)

Aufgabe 5

a

Der wechselseitige Ausschluss ist nicht gewährleistet (dazu müsste `turn[1]=0` in Prozess 2 sein). (Mit `turn[1]=0` in Prozess 2 würde gelten: Es wird davon ausgegangen, dass der Algoritmus von Peterson den wechselseitigen Ausschluss für zwei Prozesse garantiert. Es kann also immer nur ein Prozess (hier Prozess 0 oder Prozess 1) in den kritischen Abschnitt. Dies kann man dann als Pseudoprozess 0 für eine weitere Petersoninstanz betrachten. Durch diese neue Petersoninstanz kann ein weiterer Prozess (hier: Prozess 2) hinzugenommen werden. Der Algorithmus von Peterson wird also verschachtelt. Man kann sich das so vorstellen: `Peterson(Peterson(P0,P1),P2)`. Funktioniert Peterson für zwei Prozesse, so funktioniert auch diese Erweiterung auf drei Prozesse).

b

Semaphoren haben im Allgemeinen nicht dieses Problem. Bei der Implementierung im Betriebssystem werden Prozesse blockiert. Somit gibt es für den Prozess kein aktives Warten.

c

- `atomar`: ein einziger Befehl, dieser kann nicht von einem Prozesswechsel unterbrochen werden, wird entweder ganz ausgeführt oder gar nicht.
- `TSL`: prüft ob ein Lock gesetzt ist (**T**est) und setzt es (**S**et **L**ock).
- `SWP reg,mem`: Inhalt des Registers `reg` wird mit Speicherstelle `mem` vertauscht.

Aufgabe 6

a

Es gilt $\sum_{t=1}^{n} E_i = 8 \Rightarrow V \geq S$. Auch $V \geq 9$ gilt, vorausgesetzt es wird erklärt, dass noch mindestens eine freie Ressource nötig ist, damit kein Deadlock entsteht.

b

Offensichtlich; da $F = V - \sum_{t=1}^{n} E_i = 0$ und $A = (5, 3, 1, 3)$, kann im Worst Case kein Prozess weiter ausgeführt werden.

c

$V = 10$, dann gilt anfangs $F = 2$, womit p_3 und später p_2, p_1 und p_4 komplett ausgeführt werden können $\Rightarrow$ sicher.
Für $V = 9$ gilt anfangs $F = 1$, damit kann zwar p_3 auf jeden Fall fertig ausgeführt werden, danach ist aber $F = 2$, was im Worst Case nicht reicht, um p_1, p_2 oder p_3 weiter auszuführen $\Rightarrow$ unsicher.

Aufgabe 7

a

präemptiv: aktueller Prozess kann vom Btriebssystem unterbrochen und CPU kann ihm entzogen werden
nicht-prämptiv: Betriebssystem ist auf Kooperation des Prozesses angewiesen, Prozess muss CPU explizit abgeben, tut er dies nicht, kann er das ganze System blockieren.

b

Der Prozess käme doppelt so oft an die Reihe. Auf diese Art könnte man verschiedene Prioritäten implementieren.

c

- FCFS [3]: p_3, p_1, p_2, p_5, p_4

Zeitpunkt	0	4	9	13	16	18
Prozess	p_3	p_1	p_2	p_5	p_4	Ende

- SJF: [3]: p_3, p_2, p_5, p_4, p_1

Zeitpunkt	0	4	8	11	13	18
Prozess	p_3	p_2	p_5	p_4	p_1	Ende

Aufgabe 8

a

Dies vereinfacht die Umrechnung von der logischen in die physikalische Adresse. Man kommt dabei mit einfachen Operationen aus (Tablelookup und Addition bzw. logisches OR).

b

`Best Fit`: Es werden nacheinander die folgenden Blöcke belegt: 15 KB (B_8), 7 KB (B_3), 11 KB (B_6), 18 KB (B_5) `First Fit`: Es werden nacheinander die folgenden Blöcke belegt: 19 KB (B_4), 9 KB (B_1), 18 KB (B_5), 15 KB (B_8), `Next Fit`: Es werden nacheinander die folgenden Blöcke belegt: 19 KB (B_4), 18 KB (B_5), 11 KB (B_6), 15 KB (B_8).
Ebenfalls mögliche Lösung: 19 KB (B_4), 18 KB (B_5) (B_5 noch 12 KB frei!), 18 KB (B_5 noch 2 KB frei!), 15 KB (B_8)