Prof. Dr. M. Jäger
FB MNI

19. Januar 2016

Übungsblatt 6 - Betriebssysteme I
Gemischte Aufgaben/ehemalige Klausur

Aufgabe 1 (2+2+2 Punkte)

Punkte ☐ *von* **6**

a) Geben Sie an, aus welchen Einzelschritten die Ausführung eines Maschinenbefehls besteht

b) Was versteht man im Hinblick auf die Befehlsausführung innerhalb des Prozessors unter „Pipelining“?

c) Was heißt RISC und welche Vorteile hat RISC?

Aufgabe 2 (2+1+2+2+1 Punkte)

Punkte ☐ *von* **8**

Ein Prozessor benutzt einen Daten-Cache um Zugriffe auf den Hauptspeicher zu beschleunigen. Ein „write through“-Cache synchronisiert bei schreibendem Zugriff den Cache sofort mit dem Hauptspeicher. Ein „copy back“-Cache markiert den geänderten Eintrag lediglich als „modifiziert“ und synchronisiert erst bei Bedarf.

a) Welche Informationen könnten in einem Cache-Eintrag stehen?

b) Welchen Vorteil hat die „copy back“-Strategie?

c) Die Programmausführung kann durch den CPU-Cache bei lesenden Speicherzugriffen erheblich beschleunigt werden. Gilt dies auch für schreibende Zugriffe in einer CPU mit Pipelining?

d) Unter welchen Umständen könnte ein lesender Hauptspeicherzugriff bei einem „copy back“-Cache besonders lange dauern?

e) Warum ist ein CPU-Cache bei Bus-basierten Multiprozessorsystemen besonders wichtig?

Aufgabe 3 (2+2+3 Punkte)

Punkte ____ *von* **7**

a) Ein C-Programm ermittelt mit dem Systemaufruf *getpid()* seine Prozessnummer. Was ist und wie funktioniert ein „Systemaufruf“? Beschreiben Sie in Stichworten die technische Realisierung.

b) In welche logischen Bereiche ist der Programmspeicher eines Prozesse aufgeteilt?

c) Geben Sie für jeden der nachfolgenden Zustandsübergänge ein Ereignis an, das diesen Zustandsübergang auslösen könnte:

Zustandsübergang	**auslösendes Ereignis**
bereit $\rightarrow$ *blockiert*	
blockiert $\rightarrow$ *beendet*	
bereit $\rightarrow$ *ausführend*	

Aufgabe 4 (6 Punkte)

Punkte ____ *von* **6**

Welche Ausgaben könnte das folgende Programm erzeugen, wenn der Elternprozess die Nummer 4 und der Subprozess die Nummer 5 hat?

```
#include <stdio.h>
#include <unistd.h>
int main(){
  int a=fork();
  printf("%d\n",a);
  if (a) { a++; printf("%d\n",a); }
  else     a--;
}
```

Aufgabe 5 (8 Punkte)

Punkte ____ *von* **8**

Ein Prozess P soll zwei nebenläufige Subprozesse erzeugen. Beide Subprozesse verhalten sich gleich: Zuerst wird die Funktion *f1*, danach die Funktion *f2* aufgerufen. Keiner darf aber *f2* aufrufen, bevor der andere den Aufruf von *f1* beendet hat. Beide Subprozesse von P müssen also ggf. auf ihre „Bruder"-Prozesse warten.

Geben Sie eine C-Implementierung an, die Mutexe zur Synchronisation verwendet. Diese sind als Typ `mutex_t` vorgegeben und können gemäß folgender Schnittstelle verwendet werden:

```
void mutex_init(mutex_t m);
void mutex_down(mutex_t m);
void mutex_up(mutex_t m);
```

Vervollständigen Sie dazu das Programm:

```
#include <stdio.h>
#include <wait.h>
#include <unistd.h>
#include "mutex.h"


void f1 () { printf("hier f1, aufgerufen von %s\n", getpid()); }
void f2 () { printf("hier f2, aufgerufen von %s\n", getpid()); }

int main(){
  pid_t pid1, pid2;
```

Aufgabe 6 (10 Punkte)

Punkte ____ *von* **10**

Ein Programm soll die Anzahl der Zeichen einer Textdatei ermitteln und ausgeben. Die Datei „meintext.txt.gz“ ist allerdings komprimiert. Das Programm soll zwei Subprozesse erzeugen. Der erste dekomprimiert die Datei (`zcat meintext.txt.gz`), der zweite zählt die Zeichen (`wc -c`).

Vervollständigen Sie dazu den nachfolgenden Quelltext. Benutzen Sie nur die folgenden Systemaufrufe: *fork, execlp, open, read, write, pipe, close, dup2*. Verzichten Sie auf Fehlerbehandlung. Achten Sie auf Verklemmungsfreiheit.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <fcntl.h>

int main() {
```

Aufgabe 7 (2+4 Punkte)

Punkte ____ *von* **6**

Ein System verwendet virtuellen Speicher mit Paging. Für die Hauptspeicherverwaltung wird eine dreistufige Seitentabelle benutzt. Physikalische Adressen sind 64-Bit groß. Eine virtuelle Adresse ist 32 Bit groß und von der Form

p_1	p_2	p_3	D

mit folgender Aufteilung:

- p_1: 4 Bit für die Seitentabelle der 1. Stufe
- p_2: 8 Bit für die Seitentabelle der 2. Stufe
- p_2: 8 Bit für die Seitentabelle der 3. Stufe
- D: 12 Bit für die Distanz

Annahme: Der virtuellen Adresse v=0xa311f203 entspricht die reale Adresse r=0x2000340a0e000203.

a) Geben Sie an, wie der TLB-Eintrag zu (v,r) aussieht

b) Geben Sie an, wie die Rahmennummer im Beispiel bestimmt wird. Zeichnen Sie dazu eine Skizze mit den benötigten Seitentabelleneinträgen.

Aufgabe 8 (1+3 Punkte)

Punkte ____ *von* **4**

In einem Multiprozessorsystem führen zwei Threads gleichzeitig **asynchrone** Plattenzugriffe aus. Der Systemkern blockiert dabei die Threads während des Plattenzugriffs nicht, sondern erzeugt nur je einen neuen Eintrag in der Auftragsliste für den Plattencontroller. Beim Zugriff auf die Liste ist gegenseitiger Ausschluss nötig. Das System verwendet ein „Smart Lock“, das sich je nach Situation wie ein Spinlock oder ein Mutex verhält.

a) Was ist der wesentliche Unterschied zwischen Spinlock und Mutex?

b) Wann verhält sich ein „Smart Lock“ wie ein Spinlock und warum?

Aufgabe 9 (4 Punkte)

Punkte ____ *von* **4**

In einem System mit Paging steht ein nur 3 Rahmen großer Hauptpeicher zur Verfügung. Geben Sie für die Auslagerungsstrategien OPT, FIFO und LRU die Speicherbelegung und die Anzahl der Seitenfehler nach folgenden Seitenzugriffen an:

8 5 8 2 3 8 2 5 3 8

OPT

Zugriffe	**8**	**5**	**8**	**2**	**3**	**8**	**2**	**5**	**3**	**8**
Rahmen 1	8	8	8							
Rahmen 2		5	5							
Rahmen 3										

Seitenfehler:

FIFO

Zugriffe	**8**	**5**	**8**	**2**	**3**	**8**	**2**	**5**	**3**	**8**
Rahmen 1	8	8	8							
Rahmen 2		5	5							
Rahmen 3										

Seitenfehler:

LRU

Zugriffe	**8**	**5**	**8**	**2**	**3**	**8**	**2**	**5**	**3**	**8**
Rahmen 1	8	8	8							
Rahmen 2		5	5							
Rahmen 3										

Seitenfehler:

Aufgabe 10 (3+3 Punkte)

Punkte ____ *von* **6**

a) Welche Möglichkeiten hat ein Datenwiederhesrtellungsprogramm, eine auf einem UNIX-Dateisystem versehentlich gelöschte Datei wieder zu restaurieren?

b) Das Tools *fsck* („file system check“) findet bei einer Prüfung eines korrupten UNIX-Dateisystems zwei „verloren gegangene“ Dateien und stellt sie wieder her. Wie kann es dazu kommen, dass eine Datei in einer Weise verloren geht, die eine vollständige Wiederherstellung erlaubt? (Tipp: Caching)