

Midterm

donn, ZHAW Dauer: 45 Minuten

Name:

Klasse:

Aufgabe	Max. Punkte	Punkte
1	1.0	
2	1.0	
3	2.0	
4	1.5	
5	2.0	
6	2.0	
7	3.0	
Total	12.5	
	Note	

Hinweise

- Korrektur: Resultate werden nur als richtig oder falsch bewertet - bei Multiple Choice Aufgaben kann pro Teilaufgabe mehr als eine Antwort zutreffen, eine Teilaufgabe gilt als richtig, wenn sämtliche Antworten der Teilaufgabe korrekt sind.

Hilfsmittel

- erlaubt sind Taschenrechner, 2 Blätter selbstverfasste Notizen - nicht erlaubt sind PCs (Laptops), Handhelds/xPads und Handys.

Bedingungen

- kein Bleistift, keine roten Stifte und kein TipEx
- nur die Resultate auf den Aufgabenblättern eintragen
- Zusatzblätter werden nicht eingesammelt oder korrigiert
- Resultate als ganze Zahlen resp. mit 2 Stellen nach dem Komma

Aufgabe 1 Cache Zugriffe

1.0P

Gegeben ist folgender Programmausschnitt:

```
...  
#define N (10*1000*1000)  
...  
int arVL[N];  
...  
for (int i = 0; i < N; i++) {  
    sum += arVL[i];  
}
```

Wie gross ist auf einem 32-Bit Prozessor die Hitrate h_c während dem Zugriff auf $arV[i]$, wenn das Data Cache eine Blockgrösse (Cache Line Size) von 64 Bytes hat? Hinweis: es genügt die Überlegung für ein Rechnersystem mit einem Cache Level zu machen.

$h_c =$

Aufgabe 2 Speicherzugriffszeit

1.0P

Ein 2 GHz Prozessor (2 GHz entspricht einem Clockzyklus von 0.5 ns) hat eine dreistufige Cache Hierarchie, wobei folgende Daten gegeben sind:

Cache Level 1 Zugriffszeit 4 Clockzyklen

Cache Level 2 Zugriffszeit 10 Clockzyklen

Cache Level 3 Zugriffszeit 40 Clockzyklen

Hauptspeicher Zugriffszeit 60 ns

Wie gross ist die mittlere Speicherzugriffszeit T_a , wenn die Hitrate auf jedem Level 90% beträgt?

$T_a =$

Aufgabe 3 Make

a) 1.0P, b) 1.0P

Gegeben ist folgendes Makfile:

```
CFL = -g
CMP = gcc $(CFL)
app:    main1.o mythread.o scheduler.o queues.o mylist.o
        $(CMP) main1.o mythread.o scheduler.o queues.o mylist.o -o $$@.e
.C.O:
        $(CMP) -c $<
```

und ein Listing (ls -al) des Verzeichnisses in dem make ausgeführt wird:

```
total 144
drwxr-xr-x 2 usr usr 4096 Feb 24 09:21 ./
drwxr-xr-x 3 usr usr 4096 Feb 24 09:32 ../
-rw-r--r-- 1 usr usr 1894 Feb 24 09:16 main1.c
-rw-r--r-- 1 usr usr 4220 Feb 24 09:21 main1.o
-rw-r--r-- 1 usr usr 642 Feb 24 09:19 makefile
-rw-r--r-- 1 usr usr 4152 Feb 24 09:31 mylist.c
-rw-r--r-- 1 usr usr 2317 Feb 24 09:16 mylist.h
-rw-r--r-- 1 usr usr 6428 Feb 24 09:21 mylist.o
-rw-r--r-- 1 usr usr 3975 Feb 24 09:16 mythread.c
-rw-r--r-- 1 usr usr 5224 Feb 24 09:16 mythread.h
-rw-r--r-- 1 usr usr 7448 Feb 24 09:21 mythread.o
-rw-r--r-- 1 usr usr 5264 Feb 24 09:25 queues.c
-rw-r--r-- 1 usr usr 891 Feb 24 09:16 queues.h
-rw-r--r-- 1 usr usr 7648 Feb 24 09:21 queues.o
-rw-r--r-- 1 usr usr 11135 Feb 24 09:16 scheduler.c
-rw-r--r-- 1 usr usr 2434 Feb 24 09:16 scheduler.h
-rw-r--r-- 1 usr usr 10432 Feb 24 09:21 scheduler.o
```

In diesem Verzeichnis wird nun der Befehl make ausgeführt:

a) Geben Sie die Namen derjenigen C-Files an, die durch den Aufruf von make übersetzt werden:

b) Wie heisst das ausführbare File, das durch den Aufruf von make erzeugt wird:

Aufgabe 4 Prozesserzeugung

a) 1.0P, b) 0.5P

Gegeben ist folgender Codeausschnitt:

```
if (fork() > 0)
    fork();
else {
    fork();
    if (fork() > 0)
        printf("Hello World\n");
}
```

Annahme: alle fork()'s werden erfolgreich ausgeführt

a) Wie viele Prozesse werden mit diesem Codesausschnitt **zusätzlich** erzeugt?

Anzahl

b) Wie oft wird Hello World ausgegeben?

Anzahl:

Aufgabe 5 Real Time Scheduling

2.0P

Gegeben sind 3 periodische Echtzeit-Tasks mit folgenden Daten:

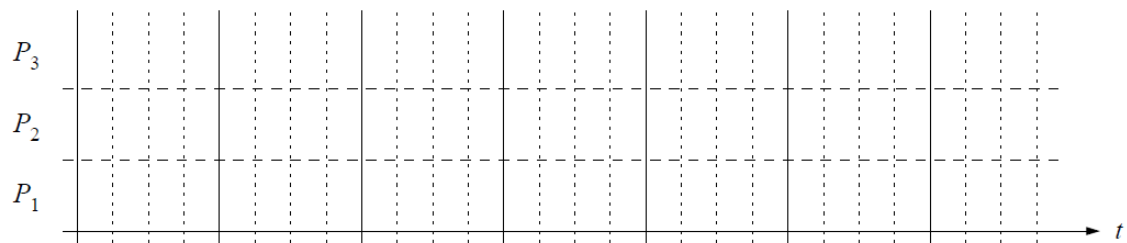
Task	Periode	Rechenzeit
1	4ms	1ms
2	8ms	4ms
3	12ms	3ms

Zeichnen Sie das Balkendiagramm für Deadline-Scheduling mit folgenden Bedingungen:

- Re-Scheduling: alle 4 ms oder wenn sich ein Task suspendiert
- bei gleichen Deadlines hat der Task mit der kürzeren Periode höhere Priorität



Sollte beim ersten mal Zeichnen etwas schief gelaufen sein:



Aufgabe 6 Scheduling

2.0P

Gegeben sind 5 Prozesse mit folgenden Prioritäten (hohe Zahl \rightarrow hohe Priorität), Ankunftszeiten T_A und Ausführungszeiten T_B :

Prozess	Priorität	Ankunftszeit T_A	Ausführungszeit T_B
P1	0	0	4
P2	1	1	3
P3	1	2	2
P4	0	5	3
P5	0	7	2

Annahmen: Multilevel Scheduling (nicht Multilevel Feedback), kein Prozess blockiert, RR mit Timeslot $q = 1$

Skizzieren Sie den Schedule als Balkendiagramm in folgendem Zeitraster:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P_1															
P_2															
P_3															
P_4															
P_5															

Sollte beim ersten mal Zeichnen etwas schief gelaufen sein:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P_1															
P_2															
P_3															
P_4															
P_5															

Aufgabe 7 Kurzfragen

0.5P pro Antwort, total 3P

Pro Teilfrage können eine, mehrere oder keine Antworten zutreffen, kreuzen Sie die richtige(n) Antwort(en) an.

Bewertung: wenn Sie eine Teilfrage richtig beantwortet haben (sämtlichen Kästchen korrekt), erhalten Sie 0.5 Punkte sonst 0 Punkte.

a) Ein Programm führt unten stehende Aktivitäten aus. Für welche dieser Aktivitäten muss vom Betriebssystem ein System-Call zur Verfügung gestellt werden?

- den Wert einer Variablen auf dem Bildschirm ausgeben
- den Wert von zwei Variablen vergleichen
- 1 Sekunde schlafen
- den Wert einer Integer Variablen inkrementieren
- den Wert einer Fließkomma Variablen um eins erhöhen

b) Die Systemfunktionen pthread join()

- terminiert alle User-Level Threads
- kann jederzeit durch sleep(0) ersetzt werden
- verhindert, dass der Prozess vor Beendigung der Threads terminiert
- blockiert die CPU, bis alle Threads terminiert haben

c) Wieso werden System-Calls werden über Softwareinterrupts aufgerufen?

- weil sie schneller sind als Prozeduraufrufe
- weil Prozeduraufrufe (CALL) nicht in den System Mode umschalten können
- weil Anwenderprogramme die Adressen der System-Call Routinen nicht kennen

d) Der PCB ist ein Datenbereich

- in dem ein Prozesse seine globalen Daten ablegt
- vom Betriebssystem Statusinformation zum Prozesse abgelegt wird
- ein Prozesse kleine Datenmengen effizient zwischenspeichern kann
- die Prozessidentifikationsnummer abgespeichert ist

e) Ein Zombie ist ein

- Thread, der zu keinem Prozess mehr gehört
- ein Prozess, der keinen Elternprozess hat
- ein Prozess, der als Eltern den init-Prozess hat
- auf den der Elternprozess nicht gewartet hat (wait(), waitpid())
- ein Prozess, beim dem alle gestarteten Threads terminiert haben

f) Ein Daemon ist

- ein Virus auf Linuxsystemen
- ein Prozess ohne Kontrollterminal
- ein Prozess, der eine Shell als Elternprozess hat
