Begonnen am	Dienstag, 18. Juli 2023, 16:00
Status	Beendet
Beendet am	Dienstag, 18. Juli 2023, 17:40
Verbrauchte Zeit	1 Stunde 40 Minuten
Punkte	52,33/100,00
Bewertung	**57,57** von 110,00 (**52**%)

Frage **1**

Teilweise richtig

Erreichte Punkte 1,33 von 2,00

Eine Memory-Management Unit

- [ ] a. erhöht die Kosten des Systems.
- [x] b. erhöht die Zeit um eine Operation zu beenden. ✔
- [x] c. erhöht die Leistungsaufnahme des Systems. ✔

Die Antwort ist teilweise richtig.

Sie haben 2 richtig ausgewählt.
Die richtigen Antworten sind:
erhöht die Zeit um eine Operation zu beenden.,

erhöht die Leistungsaufnahme des Systems.,

erhöht die Kosten des Systems.

Frage **2**

Richtig

Erreichte Punkte 2,00 von 2,00

Beim Design eines Echtzeitsystems muss man berücksichtigen

- [x] a. das Kommunikationssystem. ✔
- [x] b. das Betriebssystem. ✔
- [x] c. die Hardware. ✔

Die Antwort ist richtig.

Die richtigen Antworten sind:
die Hardware.,

das Betriebssystem.,

das Kommunikationssystem.

Frage **3**

Richtig

Erreichte Punkte 2,00 von 2,00

Die Interrupt Latenzzeit (Latency) bei einem Realtime-System

- [x] a. muss minimal sein. ✔
- [ ] b. muss maximal sein.
- [ ] c. muss 0 sein.

Die Antwort ist richtig.

Die richtige Antwort ist:
muss minimal sein.

Frage **4**

Richtig

Erreichte Punkte 2,00 von 2,00

Welchen Compiler benötigt man wenn Host- und Target- System eine andere Architektur haben?

- [ ] a. Visual-Studio IDE
- [x] b. Einen Cross- Compiler ✔
- [ ] c. Einen C- Compiler

Die Antwort ist richtig.

Die richtige Antwort ist:
Einen Cross- Compiler

Frage **5**

Teilweise richtig

Erreichte Punkte 1,00 von 2,00

Was versteht man unter Synchroner Programmierung (Ablaufsteuerung)?

- [ ] a. Arbeitet nach dem EVA - Prinzip.
- [x] b. Die einzelnen Aufgaben oder Teilaufgaben werden zu festen Zeiten eingeplant. ✔
- [ ] c. Die Aufgaben werden dynamisch, während der Programmausführung, eingeplant.

Die Antwort ist teilweise richtig.

Sie haben 1 richtig ausgewählt.
Die richtigen Antworten sind:
Die einzelnen Aufgaben oder Teilaufgaben werden zu festen Zeiten eingeplant.,

Arbeitet nach dem EVA - Prinzip.

Frage **6**

Vollständig

Erreichte Punkte
10,00 von 15,00

Ist das folgende Taskset (Periodische Tasks, Ti (C, D, P)) ausführbar mit RMS?

T1 (1, 5, 5)
T2 (4, 8, 4)
T3(5, 9, 6)

Die Antwort ist zu begründen. Rechenweg muss nachvollziehbar sein.

T1 c=1 d=5 p=5
T2 c=4 d=8 p=9
T3 c=2 d=4 p=6

U = (1/5) + (4/9) + (2/6) = 0,9777

B(3) = 3 * (2^(1/2)-1) = 0,779

U = (1/5) + (4/9) = 0,644

B(2) = 2 * (2^(1/2)-1) = 0,828

precise test für T3

R3^0 = C1+C2+C3 = 2+4+6 = 12

R3^1 = C3 + (R3^0/T1) * C2 + (R3^0/T2) * C1 + (R3^0/T3)

R3^1 = 10

Kommentar:

UB-Test ist hier nicht anwendbar (Folgefehler zu DMS Frage).

Der Precise-Test muss für T3 und T2 angewendet werden.

Prio - Reihenfolge der Tasks stimm nicht, Precise Test unter der falschen Annahme ok

Frage **7**

Richtig

Erreichte Punkte
2,00 von 2,00

Beim Offline Scheduling

- [x] a. erlaubt eine maximale Prozessorauslastung. ✔
- [ ] b. ist flexibel, ggf. kein optimaler Plan möglich.
- [x] c. wird Plan wird vor Laufzeit des Systems entwickelt. ✔
- [ ] d. wird der Plan zur Laufzeit entwickelt.

Die Antwort ist richtig.

Die richtigen Antworten sind:
wird Plan wird vor Laufzeit des Systems entwickelt.,

erlaubt eine maximale Prozessorauslastung.

Frage **8**

Falsch

Erreichte Punkte
0,00 von 10,00

In welcher Reihenfolge werden beim folgenden Beispiel die Ausgaben erzeugt?

```
#include <sched.h>
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>

pthread_mutex_t mutex;
pthread_cond_t cond;

void * print_hello(void * arg)
{
  struct timespec now;
  struct timespec timeout;

  printf("<child>: Hello World! task with max priority \n");
  clock_gettime( CLOCK_REALTIME, &now );

  printf("\nnow tv_sec = %d, tv_nsec = %d\n", now.tv_sec, now.tv_nsec);

  timeout.tv_sec  = now.tv_sec + 3;
  timeout.tv_nsec = now.tv_nsec;

  printf("timeout tv_sec = %d, tv_nsec = %d\n", timeout.tv_sec, timeout.tv_nsec);
  printf("The task is coming to enter in a timed wait\n");
  pthread_cond_timedwait(&cond, &mutex, &timeout);
  printf("The task is coming out from the timed wait \n");
  return NULL;
}

void * print_hello_a(void * arg)
{
  printf(" <child>: Hello World! Task with lowest priority ");
  return NULL;
}


int main(int argc, char **argv)
```

```
{
  pthread_attr_t      attr;
  pthread_t           child1;
  pthread_t           child2;
  struct sched_param  param;

  (void) argc;
  (void) argv;

  pthread_attr_init(&attr);
  pthread_attr_setschedpolicy(&attr, SCHED_FIFO);
  pthread_mutex_init( &mutex, NULL );
  pthread_cond_init( &cond, NULL );

  printf("<main> Enter in the main \n");

  printf("Creating first task \n");
  param.sched_priority = sched_get_priority_max(SCHED_FIFO);
  pthread_attr_setschedparam(&attr, &param);
  if ( pthread_create( &child1, &attr, print_hello, NULL) ||
       pthread_setschedparam(child1, SCHED_FIFO, &param) ) {
    printf(
      "Thread cannot be created or you have not enough privileges \n"
      "    to set priority!!!!\n");
    exit(1);
  }

  printf("First Task created \n");
  printf("Creating second task \n");
  param.sched_priority = sched_get_priority_max(SCHED_FIFO) - 1;
  pthread_attr_setschedparam(&attr, &param);
  if ( pthread_create( &child2, &attr, print_hello_a, NULL) ||
       pthread_setschedparam(child2, SCHED_FIFO, &param) ) {
    printf(
      "Thread cannot be created or you have not enough privileges \n"
      "    to set priority!!!!\n");
```

```
    exit(1);
  }
  printf("Second task created \n");

  printf("<main> Out of the main\n");
  pthread_join( child1, NULL );
  pthread_join( child2, NULL );

  exit(0);
}

#if defined(__rtems__)
#include <bsp.h>

static void *POSIX_Init()
{
  return (void *)main(0, NULL);
}

#define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
#define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER

#define CONFIGURE_MAXIMUM_POSIX_THREADS              10
#define CONFIGURE_POSIX_INIT_THREAD_TABLE

#define CONFIGURE_INIT
#include <rtems/confdefs.h>
#endif
```

a. ...

<main> Out of the main

The task is coming out from the timed wait

<child>: Hello World! Task with lowest priority

b.

...

<child>: Hello World! Task with lowest priority

<main> Out of the main

The task is coming out from the timed wait

c.

...

<main> Out of the main

<child>: Hello World! Task with lowest priority

The task is coming out from the timed wait

Die Antwort ist falsch.

Die richtige Antwort ist:

...
<child>: Hello World! Task with lowest priority
<main> Out of the main
The task is coming out from the timed wait

Kommentar:

Frage **9**

Richtig

Erreichte Punkte
2,00 von 2,00

Der Einplantest nach Liu/Layland

- [x] a. hat einen konstanten Aufwand. ✔
- [x] b. ist pessimistisch. ✔
- [x] c. hat einen niedrigen Aufwand. ✔

Die Antwort ist richtig.

Die richtigen Antworten sind:
ist pessimistisch.,

hat einen niedrigen Aufwand.,

hat einen konstanten Aufwand.

Frage **10**

Richtig

Erreichte Punkte 4,00 von 4,00

Damit ein Deadlock entstehen kann, müssen folgende Bedingungen gleichzeitig erfüllt sein

- [ ] a. Rekursive Funktionsaufrufe (recursive calls)
- [x] b. Anforderung weiterer Betriebsmittel (hold and wait) ✔
- [ ] c. Häufige Funktionsaufrufe (massive calls)
- [x] d. Wechselseitiger Ausschluss (mutual exclusion) ✔
- [ ] e. Geschachtelte *for*-Schleifen (recursive for loops)
- [x] f. Zyklische Wartebedingung (circular wait) ✔
- [x] g. Ununterbrechbarkeit (no preemption) ✔
- [ ] h. Warteschlangen (waiting queues)

Die Antwort ist richtig.

Die richtigen Antworten sind:
Wechselseitiger Ausschluss (mutual exclusion),

Anforderung weiterer Betriebsmittel (hold and wait),

Ununterbrechbarkeit (no preemption),

Zyklische Wartebedingung (circular wait)

Frage **11**

Falsch

Erreichte Punkte 0,00 von 2,00

Was versteht man unter Asynchroner Programmierung (Ablaufsteuerung)?

- a. Die einzelnen Aufgaben oder Teilaufgaben werden zu festen Zeiten eingeplant.
- b. Arbeitet nach dem EVA - Prinzip. ✖
- c. Die Aufgaben werden dynamisch, während der Programmausführung, eingeplant. ✔

Die Antwort ist falsch.

Die richtige Antwort ist:
Die Aufgaben werden dynamisch, während der Programmausführung, eingeplant.

Frage **12**

Richtig

Erreichte Punkte 2,00 von 2,00

Prioritäts Umkehr kann gelöst werden

- a. durch das Zwei-phasen-Protokoll.
- b. durch ein Zeit-Protokoll.
- c. durch das Priority Inheritance Protocol. ✔

Die Antwort ist richtig.

Die richtige Antwort ist:
durch das Priority Inheritance Protocol.

Frage **13**

Falsch

Erreichte Punkte
0,00 von 2,00

Beim RMS | Jobs mit kurzer Laufzeit haben eine höhere Priorität ✖

Die Antwort ist falsch.

Die richtige Antwort lautet:
Beim RMS [hängt die Piorität nicht von der Laufzeit eines Jobs ab]

Frage **14**

Teilweise richtig

Erreichte Punkte
1,33 von 2,00

Welchen Vorteil haben Mutex im Vergleich zu binären Semaphoren?

- [x] a. Mutexe sind einfach zu implementieren. ✔
- [x] b. Mutexe realisieren einen wechselseitigen Ausschluss/Schutz einer *Critical Section*. ✔
- [ ] c. Mutexe realisieren die Ownership eines Synchronisationsobjekts.

Die Antwort ist teilweise richtig.

Sie haben 2 richtig ausgewählt.
Die richtigen Antworten sind:
Mutexe sind einfach zu implementieren.,

Mutexe realisieren einen wechselseitigen Ausschluss/Schutz einer *Critical Section*.,

Mutexe realisieren die Ownership eines Synchronisationsobjekts.

Frage **15**

Teilweise richtig

Erreichte Punkte 1,00 von 2,00

Was bedeutet der Begriff *Idle-Task*?

- [ ] a. der Task mit der geringsten Priorität.
- [x] b. stellt sicher, dass das System noch reagiert ('lebt'). ✔
- [ ] c. sind alle tasks die nicht im *ready* Zustand sind.

Die Antwort ist teilweise richtig.

Sie haben 1 richtig ausgewählt.
Die richtigen Antworten sind:
der Task mit der geringsten Priorität.,

stellt sicher, dass das System noch reagiert ('lebt').

Frage **16**

Richtig

Erreichte Punkte 2,00 von 2,00

Harte Realtime - Systeme haben [weniger] ✔ Jitter als Soft-Realtime - Systeme.

Die Antwort ist richtig.

Die richtige Antwort lautet:
Harte Realtime - Systeme haben [weniger] Jitter als Soft-Realtime - Systeme.

Frage **17**

Falsch

Erreichte Punkte
0,00 von 2,00

Wozu wird ein Einplanungstest genutzt?

- [ ] a. ob eine gegebene Taskmenge unter einem gegebenen Schedulingverfahren planbar ist.
- [ ] b. ob ein einzelner Task unter einem gegebenen Schedulingverfahren planbar ist.
- [x] c. ob eine gegebene Taskmenge unter einem beliebigen Schedulingverfahren planbar ist. ✖

Die Antwort ist falsch.

Die richtige Antwort ist:
ob eine gegebene Taskmenge unter einem gegebenen Schedulingverfahren planbar ist.

Frage **18**

Falsch

Erreichte Punkte 0,00 von 2,00

Welche der folgenden Aussagen bzgl. der Interruptsteuerung trifft zu?

- [ ] a. Wurde gerade ein Flanken-gesteuerter Interrupt ausgelöst, so muss erst ein Pegelwechsel der Interruptleitung stattfinden, damit erneut ein Interrupt ausgelöst werden kann.
- [x] b. Interrupts sind eine Besonderheit von Intel-Microprozessoren. Auf anderen Architekturen kommen POSIX-Signale zum Einsatz. ✖
- [x] c. Pegel-gesteuerte Interrupts werden beim Wechsel des Pegels ausgelöst, daher der Name. ✔
- [ ] d. Pegelgesteuerte Interrupts müssen durch Polling des Pegels abgefragt werden.

Die Antwort ist falsch.

Die richtigen Antworten sind:
Pegel-gesteuerte Interrupts werden beim Wechsel des Pegels ausgelöst, daher der Name.,

Pegelgesteuerte Interrupts müssen durch Polling des Pegels abgefragt werden.,

Wurde gerade ein Flanken-gesteuerter Interrupt ausgelöst, so muss erst ein Pegelwechsel der Interruptleitung stattfinden, damit erneut ein Interrupt ausgelöst werden kann.

Frage **19**

Richtig

Erreichte Punkte 2,00 von 2,00

Beim Online Scheduling

- [ ] a. ermöglicht eine maximale Prozessorauslastung.
- [ ] b. wird der Plan wird vor Laufzeit des Systems entwickelt.
- [x] c. ist flexibel, ggf. kein optimaler Plan möglich. ✔
- [x] d. wir der Ablauf zur Laufzeit entwickelt. ✔

Die Antwort ist richtig.

Die richtigen Antworten sind:
wir der Ablauf zur Laufzeit entwickelt.,

ist flexibel, ggf. kein optimaler Plan möglich.

Frage **20**

Richtig

Erreichte Punkte 2,00 von 2,00

Welche Kernel werden bei Realtime - Systemen eingesetzt?

- [ ] a. Preemtive oder non-preemtive kernel.
- [x] b. Preemtive kernel. ✔
- [ ] c. Weder preemtive noch non-preemtive kernel.
- [ ] d. Non-preemtive kernel.

Die Antwort ist richtig.

Die richtige Antwort ist:
Preemtive kernel.

Frage **21**

Teilweise richtig

Erreichte Punkte 1,33 von 2,00

Warum werden Mutexe verwendet?

- [ ] a. um einen Besitzaspekt zu realisieren.
- [x] b. um Race Conditions zu vermeiden/Ressourcen zu schützen ✔
- [x] c. um wechselseitigen Ausschluss zu erwirken. ✔

Die Antwort ist teilweise richtig.

Sie haben 2 richtig ausgewählt.
Die richtigen Antworten sind:
um Race Conditions zu vermeiden/Ressourcen zu schützen,

um wechselseitigen Ausschluss zu erwirken.,

um einen Besitzaspekt zu realisieren.

Frage **22**

Vollständig

Erreichte Punkte
0,00 von 10,00

Schreiben Sie ein C-Programm (threads-example-sync.c) mit dem sie 10 Threads erzeugen die folgende Funktion aufrufen:

```
// ethread: example thread
void *ethread (void *arg)
{
    int my_id = *((int *) arg);
    // Take a nap
    sleep (1);
    // say hello and terminate
    printf ("Thread %d: Hello World!\n", my_id);
    sprintf (ret_status [my_id], "Thread %d: %d", my_id, my_id + 10);
    // pass your id to the thread waiting for you to terminate
    // using pthread_join.
    pthread_exit (ret_status [my_id]);
}
```

Es muss sichergestellt werden, dass die Ausgaben in der Reihenfolge von my_id ausgegeben werden. Es muss im main-task auf die Beendigung der Threads gewartet werden und der von jedem Thread zurückgegebene Status ausgegeben werden.

```
#include <xyz.h>
```

```
#include <xyz.h>
```

```
void *ethread (void *threadid)
{
```

```
int my_id = *((int *) threadid);
```

```
sleep(1)
```

```
sprintf (ret_status [my_id], "Thread %d: %d", my_id, my_id + 10);
```

pthread_exit(NULL);

```
}
int main()
{
```

```
pthread_t threads[Num_Threads
int thread_id[Num_Theads]
```

```
int rc
```

```
int i
```

```
for (i=0 ; i < Num_Threads; i++)
{
```

```
thread_id[i| = i;
```

```
rc = pthread_create(Thread[i], NULL, void *threadid[i]);
```

```
return 0;
```

```
}
```

Kommentar:

Dieses 'Programm' folgt nicht der Anforderung.

Frage **23**

Richtig

Erreichte Punkte 2,00 von 2,00

Wozu werden Realtime Systeme eingesetzt?

- [ ] a. Für interaktive realtime- Benutzer.
- [ ] b. Für die Programm Entwicklung.
- [ ] c. Vorrangig für Mainframe-Computer genutzt.
- [x] d. Zur Überwachung von Ereignissen. ✔

Die Antwort ist richtig.

Die richtige Antwort ist:
Zur Überwachung von Ereignissen.

Frage **24**

Teilweise richtig

Erreichte Punkte 1,33 von 2,00

Der Einplantest mit der Ratenmonotonen Analyse

- [x] a. hat einen nicht determinstischen Aufwand. ✔
- [x] b. ist exakt. ✔
- [ ] c. hat einen hohen Aufwand.

Die Antwort ist teilweise richtig.

Sie haben 2 richtig ausgewählt.
Die richtigen Antworten sind:
ist exakt.,

hat einen hohen Aufwand.,

hat einen nicht determinstischen Aufwand.

Frage **25**

Richtig

Erreichte Punkte 2,00 von 2,00

In [Realtime] ✔ - Betriebssystemen ist die Antwortzeit besonders kritisch.

Die Antwort ist richtig.

Die richtige Antwort lautet:
In [Realtime] - Betriebssystemen ist die Antwortzeit besonders kritisch.

Frage **26**

Richtig

Erreichte Punkte 2,00 von 2,00

Wie kann man einen nichtunterbrechbaren Kritischen Abschnitt (Non-preemptive critical section) effizient implementieren?

- [ ] a. man legt den Bereich in den Cache.
- [ ] b. man speichert den Bereich im NVRAM.
- [x] c. Zu Beginn des Ressourcenzugriffs werden die Interrupts ausgeschaltet und nach Komplettierung desselben wieder erlaubt. ✔

Die Antwort ist richtig.

Die richtige Antwort ist:
Zu Beginn des Ressourcenzugriffs werden die Interrupts ausgeschaltet und nach Komplettierung desselben wieder erlaubt.

Frage **27**

Vollständig

Erreichte Punkte
5,00 von 15,00

Ist das folgende Taskset (Periodische Tasks, Ti (C, D, P)) ausführbar mit DMS?

T1 (1, 5, 5)
T2 (4, 8, 4)
T3(5, 9, 6)

Die Antwort ist zu begründen. Rechenweg muss nachvollziehbar sein.

T1 c=1 d=5 p=5
T2 c=4 d=8 p=9
T3 c=2 d=4 p=6

U = (1/5) + (4/9) + (2/6) = 0,977 = 97,7 %

B(3) = 3 * (2^(1/2)-1) = 0,779

U Deadline = (1/5) + (4/8) + (2/4) = 1,2

UB Test nicht erfolgreich

UB Test für T1 und T2
U = (1/3) + (4/8) = 0,833

B(2) = 2 * (2^(1/2)-1) = 0,828

UB Test nicht erfolgreich

UB Test für T1
U = (1/3) = 0,3333

B(1) = 1 * (2^(1/2)-1) = 1

UB Test erfolgreich

--> Daher Precise Test

precise test für T3

R3^0 = C1+C2+C3 = 2+4+6 = 12

R3^1 = C3 + (R3^0/T1) * C2 + (R3^0/T2) * C1 + (R3^0/T3)

R3^1 = 10

Kommentar:

Bei DMS ist die Prio-Reihenfolge T3, T1, T2 (-5 Punkte)

Der UB-Tests ist nur anwendbar wenn D = P. (-5 Punkte)

Frage **28**

Richtig

Erreichte Punkte 2,00 von 2,00

Race conditions

- [x] a. treten auf, wenn zwei tasks/threads den Wert der gleichen Speicherstelle ändern können. ✔
- [ ] b. können auf einem Single-Prozessor-System nicht vermieden werden.
- [x] c. sind schwer auffindbare Programmfehler. ✔
- [x] d. können durch den konsequenten Einsatz von Sempahoren/Mutexes vermieden werden. ✔
- [x] e. hängen von dem zeitlichen Verlauf des gesamten Systems ab. ✔

Die Antwort ist richtig.

Die richtigen Antworten sind:
treten auf, wenn zwei tasks/threads den Wert der gleichen Speicherstelle ändern können.,

hängen von dem zeitlichen Verlauf des gesamten Systems ab.,

sind schwer auffindbare Programmfehler.,

können durch den konsequenten Einsatz von Sempahoren/Mutexes vermieden werden.

◄ Kursraum online VE

Direkt zu:

BeispielKlausur(nicht Moodle) aus SoSe 22 ►