T.H. Rijswijk

KLAS(SEN)	:	EH48	BLAD	:	1 van 4 BLADEN
OND. DEEL	:	Real-time Embedded Programming	DOCENT	:	Harry Broeders
AFK. O.E.	:	**ESWE1C1**	DATUM	:	9 april 2003
KWART.	:	1 TYPE : toets	TIJD	:	3 + 4

Tijdens dit tentamen mogen **alle** boeken, dictaten, aantekeningen enz. worden gebruikt.

Bij elke opgave staat tussen haakjes het maximale aantal te behalen punten vermeld.
Eindcijfer = (aantal behaalde punten + 10) / 10.

1. Op pagina 3 en 4 van dit tentamen vind je een vereenvoudigde versie van het programma dat je bij practicumopgave 2 hebt geanalyseerd.

A. (10) De semafoor `semMutualExclusive` die, bij het op het practicum geanalyseerde programma, zorgde voor wederzijdse uitsluiting ontbreekt hier. Toch ontstaan er geen synchronisatie problemen. Verklaar dit!

B. (20) Wat is de uitvoer van dit programma? Verklaar je antwoord door duidelijk stap voor stap aan te geven wat er gebeurt.

2. Gegeven zijn de volgende Java classes:

```
public class Calculate implements Runnable {
    public void run() {
        /* long calculation */
    }
}

public class TestCalculate {
    public static void main(String args[]) {
        Calculate myCalculation = new Calculate();
        myCalculation.run();                /*1*/
        new Thread(myCalculation).start();  /*2*/
}
```

(10) Leg duidelijk uit wat het **verschil** is tussen de regels gemarkeerd met `/*1*/` en `/*2*/`.

3. Gegeven is het volgende C programma:

```
#include <pthread.h>

// ...

void* t1(void* p) {
    a();
    b();
    return NULL;
}

void* t2(void* p) {
    c();
```

Zie volgende blad ⇨

```
    d();
    return NULL;
}

int main(void) {
    pthread_t p1, p2;

    check( pthread_create(&p1, NULL, t1, NULL) );
    check( pthread_create(&p2, NULL, t2, NULL) );

    check( pthread_join(p1, NULL) );
    check( pthread_join(p2, NULL) );
    return EXIT_SUCCESS;
}
```

(25) Thread `t1` roept de (niet gegeven) functies `a` en `b` aan. Thread `t2` roept de (ook niet gegeven) functies `c` en `d` aan. Er moet nu voor gezorgd worden dat `d` pas aangeroepen wordt als `a` klaar is. Geef aan hoe dit met behulp van een POSIX message queue geimplementeerd kan worden. Geef de complete code.

4.

In een multitasking systeem zijn de volgende taken gedefinieerd:

taak	T	D	C
A	8	5	4
B	20	10	4
C	20	12	4

T = periode tijd. D = deadline. C = calculation time.

A. (10) Welke prioriteit moet aan deze taken worden toegekend om een optimale scheduling te garanderen?

B. (15) Bereken de response tijden van de 3 taken bij de in antwoord op opgave A gegeven prioriteiten.

Succes!

T.H. Rijswijk

KLAS(SEN) : EH48		BLAD	: 3 van 4 BLADEN
OND. DEEL : Real-time Embedded Programming		DOCENT	: Harry Broeders
AFK. O.E. : **ESWE1C1**		DATUM	: 9 april 2003
KWART. : 1	TYPE : toets	TIJD	: 3 + 4

Tijdens dit tentamen mogen **alle** boeken, dictaten, aantekeningen enz. worden gebruikt.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>

#define SIZE 4

void check_errno(int error) {
     if (error<0) {
          perror("Error");
          exit(EXIT_FAILURE);
     }
}

void check(int error) {
     if (error!=0) {
          fprintf(stderr, "Error: %s\n", strerror(error));
          exit(EXIT_FAILURE);
     }
}

char buffer[SIZE];
int indexGet;
int indexPut;
sem_t semEmpty;
sem_t semFilled;

void put(const char c) {
     check_errno( sem_wait(&semEmpty) );
     buffer[indexPut++]=c;
     indexPut&=SIZE-1;
     check_errno( sem_post(&semFilled) );
}

char get(void) {
     char c;
     check_errno( sem_wait(&semFilled) );
     c=buffer[indexGet++];
     indexGet&=SIZE-1;
     check_errno( sem_post(&semEmpty) );
     return c;
}

void* producer(void* arg) {
     char c=*(char*)arg;
     int i;
     check_errno( putchar('1') );
     for (i=0; i<10; ++i) {
          put(c);
     }
     check_errno( putchar('2') );
}

void* consumer(void* arg) {
     int i;
     check_errno( putchar('3') );
     for (i=0; i<20; ++i) {
          check_errno( putchar(get()) );
     }
     check_errno( putchar('4') );
}
```

T.H. Rijswijk

KLAS(SEN)	:	EH48		BLAD	: 4 van 4 BLADEN
OND. DEEL	:	Real-time Embedded Programming		DOCENT	: Harry Broeders
AFK. O.E.	:	**ESWE1C1**		DATUM	: 9 april 2003
KWART.	:	1	TYPE : toets	TIJD	: 3 + 4

Tijdens dit tentamen mogen **alle** boeken, dictaten, aantekeningen enz. worden gebruikt.

```
int main(int argc, char *argv[]) {
     int p;
     struct sched_param sp, spc, spp1, spp2;
     pthread_attr_t ptac, ptap1, ptap2;
     pthread_t ptp1, ptp2, ptc;
     char bamihap='B', nasischijf='N';

     check( pthread_getschedparam(pthread_self(), &p, &sp) );
     sp.sched_priority=60;
     check( pthread_setschedparam(pthread_self(), SCHED_FIFO, &sp) );

     check_errno( sem_init(&semEmpty, 0, SIZE) );
     check_errno( sem_init(&semFilled, 0, 0) );

     check( pthread_attr_init(&ptac) );
     check( pthread_attr_init(&ptap1) );
     check( pthread_attr_init(&ptap2) );

     check( pthread_attr_setinheritsched(&ptac, PTHREAD_EXPLICIT_SCHED) );
     check( pthread_attr_setinheritsched(&ptap1, PTHREAD_EXPLICIT_SCHED) );
     check( pthread_attr_setinheritsched(&ptap2, PTHREAD_EXPLICIT_SCHED) );

     check( pthread_attr_setschedpolicy(&ptac, SCHED_RR) );
     check( pthread_attr_setschedpolicy(&ptap1, SCHED_RR) );
     check( pthread_attr_setschedpolicy(&ptap2, SCHED_RR) );

     check( pthread_attr_getschedparam(&ptac, &spc) );
     check( pthread_attr_getschedparam(&ptap1, &spp1) );
     check( pthread_attr_getschedparam(&ptap2, &spp2) );

     spc.sched_priority=21;
     spp1.sched_priority=20;
     spp2.sched_priority=22;

     check( pthread_attr_setschedparam(&ptac, &spc) );
     check( pthread_attr_setschedparam(&ptap1, &spp1) );
     check( pthread_attr_setschedparam(&ptap2, &spp2) );

     check( pthread_create(&ptc, &ptac, consumer, 0) );
     check( pthread_create(&ptp1, &ptap1, producer, &bamihap) );
     check( pthread_create(&ptp2, &ptap2, producer, &nasischijf) );

     check( pthread_join(ptc, 0) );
     check( pthread_join(ptp1, 0) );
     check( pthread_join(ptp2, 0) );

     check_errno( sem_destroy(&semEmpty) );
     check_errno( sem_destroy(&semFilled) );

     check( pthread_attr_destroy(&ptac) );
     check( pthread_attr_destroy(&ptap1) );
     check( pthread_attr_destroy(&ptap2) );

     return EXIT_SUCCESS;
}
```