ชื่อ..... รหัสนักศึกษา.....

PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Midterm Examination: 1st Semester | Academic Year: 2008

Date: 27 July 2008 | Time: 13:30-16:30

Subject: 240-306/241-304 Computer Operating Systems | Room: A401, หัวหุ่นยนต์

---

ทุจริตในการสอบ โทษขั้นต่ำคือ ปรับตกในรายวิชาที่ทุจริต และพักการเรียน 1 ภาคการศึกษา

Instruction:

- Please write your name and student id on all pages. There are 12 pages.
- This examination has 11 questions. Please answer all questions. Your answer can be in Thai.
- There is one extra credit question on the last page which is an optional question.

NOTE that I can only grade what I can read.

คำสั่ง

- กรุณาเขียนชื่อและรหัสนักศึกษาบนข้อสอบทุกหน้า ข้อสอบมีทั้งหมด 12 หน้า
- ข้อสอบมี 11 ข้อ กรุณาตอบทุกข้อ คุณสามารถตอบเป็นภาษาไทยได้
- ข้อสอบพิเศษในหน้าสุดท้ายนั้นคุณจะตอบหรือไม่ก็ได้

อาจารย์จะสามารถตรวจได้เฉพาะคำตอบที่อาจารย์อ่านออกเท่านั้น หากอาจารย์อ่านคำตอบคุณไม่ออก คุณจะไม่ได้คะแนน

ชื่อ..... รหัสนักศึกษา.....

ข้อมูลเพื่อใช้ในการตอบคำถาม

- Interrupt เมื่อตัวควบคุมอุปกรณ์ไอโอทำงานของมันเสร็จสิ้นลง ตัวควบคุมอุปกรณ์ก็จะบอกซีพียูโดยใช้ อินเตอร์รัพท์
- Exception หรือ trap เกิดขึ้นจากซอฟแวร์เนื่องจากมีข้อผิดพลาดเกิดขึ้นในโปรแกรม หรือ การที่โปรแกรมร้องขอการบริการจากระบบปฏิบัติการ
- Privileged instructions คือคำสั่งที่จะต้องรันในโหมด privilege หรือ โหมดเคอเนล เท่านั้น
- Starvation ปัญหาการอดอยาก ซึ่งคือการที่โพรเซสหรืองานใดๆ ต้องรอเป็นระยะเวลาอันยาวนานหรือในบางกรณีเป็นการรอแบบไม่มีที่สิ้นสุด
- Preemptive เมื่อโพรเซสได้เริ่มรันแล้ว ระบบจะสามารถหยุดการทำงานของโพรเซสและเรียกทรัพยากรคืนเพื่อไปแจกจ่ายให้กับโพรเซสอื่นได้ ซึ่งในระหว่างนั้น ตัวโพรเซสที่โดนพรีเอ็มก็จะไปรออยู่ใน ready queue
- Non-preemptive เมื่อโพรเซสได้เริ่มรันแล้ว โพรเซสจะใช้งานซีพียูไปจนกระทั้งเสร็จสิ้นการทำงาน
- Waiting time เวลาที่โพรเซสต้องรออยู่ใน ready queue
- Mutual Exclusion: หมายถึงในเวลาขณะใดขณะหนึ่ง หากมีโพรเซสหนึ่งกำลังรันอยู่ใน critical section โพรเซสอื่นๆ จะไม่สามารถรันใน critical section
- Progress: หมายถึง หากไม่มีโพรเซสใดกำลังรันอยู่ใน critical section และมีโพรเซสอื่นๆ ที่กำลังรอจะเข้าไปรันใน critical section แล้วนั้น การเลือกโพรเซสเพื่อเข้าไปรันใน critical section จะไม่สามารถเลื่อนออกไปได้
- Bounded waiting: หมายถึง มีการกำหนดจำนวนครั้งหรือจำนวนโพรเซสอื่นซึ่งเข้าไปรันใน critical section ในระหว่างที่โพรเซสหนึ่งรอขอเข้าไปรันใน critical section จนถึงเวลาที่โพรเซสนั้นได้รับอนุญาตให้รันได้
- two-phase locking protocol จะกำหนดให้การล็อคและปลดล็อคเป็นไปแบบเฟส (phase) ซึ่งการร้องขอเพื่อถือครองล็อคจะทำในเฟสโกรวิง (growing phase) ทรานเซกชันจะไม่มีการปลดล็อคใดๆในเฟสนี้จะร้องขอล็อคได้เพียงอย่างเดียว ส่วนการปลดล็อคจะทำในเฟสชริงกิง (shrinking phase) เช่นเดียวกันในเฟสนี้จะเป็นการปลดล็อคเพียงอย่างเดียว จะไม่มีการร้องขอล็อคใดๆ
- timestamp protocol ใช้ timestamp เป็นตัวกำหนดลำดับการทำงาน

ชื่อ..... รหัสนักศึกษา.....

**Question 1**: (10 points) Categorize the following as one of the following: Interrupt, Exception, or Neither.
โปรดจัดกลุ่มของหัวข้อต่อไปนี้ โดยเลือกจากสามกลุ่มต่อไปนี้ Interrupt, Exception, หรือ Neither (ไม่ใช่ทั้ง interrupt และ Exception)

a. Timer
b. Keyboard input
c. Divide by zero
d. Procedure call or Function call
e. System Call

**Question 2**: (15 points) Which of the following instructions should be privileged?
คำสั่งต่อไปนี้ คำสั่งข้อใดควรจะเป็นแบบ privilege (กรุณาเลือกให้ครบทุกตัว)

a. Set value of timer
b. Read the cock
c. Clear memory
d. Issue a trap instruction
e. Turn off interrupts
f. Modify entries in device-status table
g. Switch from user to kernel mode
h. Access I/O device

**Question 3**: (15 points) Fill in the blank
โปรดเติมคำในช่องว่าง

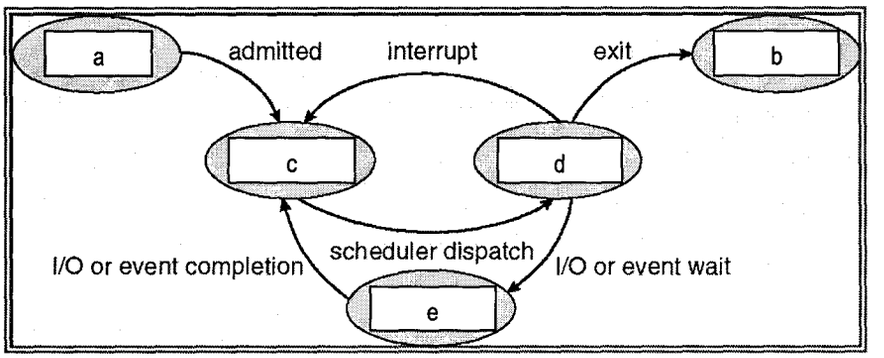


**Diagram of process state**

a. _____ b. _____

c. _____ d. _____

e. _____

ชื่อ..... รหัสนักศึกษา.....

**Question 4**: (20 points) What is wrong with the following pseudo code? How would you fix it?

จงอธิบายว่าโครงร่างของโปรแกรมต่อไปนี้มีปัญหาอะไร และคุณจะแก้ไขอย่างไร

```
pid = fork()
create pipe
if pid == 0
    read from the pipe
else
   write to the pipe
```

**Question 5**: (10 points) What will be the output at LINE A? Explain your answer

ผลลัพธ์ที่ LINE A คืออะไร โปรดอธิบายคำตอบของท่าน

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 10;

int main(void) {
    pid_t pid;

     if(pid == 0) { /* child process */
          value += 25;
    }
     else if(pid > 0) {/* parent process */
          wait(NULL);
          value += 5;
          printf("PARENT: value = %d \n",value); /* LINE A */
          exit(0);
     }
}
```

ชื่อ..... รหัสนักศึกษา.....

**Question 6**: (20 points) What would be the output from the following problem at LINE C and LINE P? Explain your answer
ผลลัพธ์ที่ LINE C และ LINE P คืออะไร โปรดอธิบายคำตอบของท่าน

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

int value = 20;
void *runner(void *param);

int main(int argc, char *argv[]) {
  pthread_t tid;
  pthread_attr_t attr;
  int pid;

 pid = fork();

 if (pid == 0) {
    pthread_attr_init(&attr);
    pthread_create(&tid,&attr,runner,NULL);
    pthread_join(tid,NULL);
    printf("CHILD: value = %d\n",value);  /* LINE C */
  }
  else if(pid > 0) {
    wait(NULL);
    value = 5;
    printf("PARENT: value = %d\n",value);  /* LINE P */
  }
}

void *runner(void *param) {
  value = 15;
  pthread_exit(0);
}
```

ชื่อ..... รหัสนักศึกษา.....

**Question 7**: (20 points) Please select TRUE or FALSE (โปรดเลือกว่าข้อความต่อไปนี้ ถูก หรือ ผิด)

TRUE FALSE System calls allow user-level processes to request services of the operating system
System calls อนุญาตให้โพรเซสผู้ใช้ร้องขอการบริการจากระบบปฏิบัติการ

TRUE FALSE The main purposes of an operating system are (1) to allocate the resources and (2) to control the system
หน้าที่หลักของระบบปฏิบัติการคือการแจกจ่ายทรัพยากรของระบบ และการควบคุมระบบ

TRUE FALSE A CPU scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, there are n possible different schedules
CPU scheduling algorithm กำหนดลำดับการทำงานของโพรเซส หากเรามีโพรเซส n ตัวเพื่อที่จะสเก็ตดูลบนหน่วยประมวลผลเดียว เราจะมีได้ถึง n สเก็ตดูลที่แตกต่างกัน

TRUE FALSE Whether a communication is direct or indirect, messages exchanged by communicating processes reside in a temporary queue. Basically, such queues can be implemented in three ways: zero capacity buffering, bounded capacity buffering, and unbounded capacity buffering.
ไม่ว่าการสื่อสารจะเป็นแบบโดยตรงหรือโดยอ้อม ข้อความที่แลกเปลี่ยนกันจะต้องไปอยู่ในคิวชั่วคราว ซึ่งคิวอันนี้สามารถเป็นได้สามแบบคือ แบบ zero capacity buffering, bounded capacity buffering, และ unbounded capacity buffering

TRUE FALSE Each process is represented in the operating system by a process control block (PCB) which is also called a task control block.
ระบบปฏิบัติการใช้ PCB (Process control block) เป็นตัวแทนของแต่ละโพรเซสในระบบ ซึ่ง PCB นั้นเรียกอีกอย่างหนึ่งว่า TCB (Task control block)

ชื่อ..... รหัสนักศึกษา.....

**Question 8**: Consider the following set of processes, with the length of the CPU burst given in milliseconds:

| Process | Burst Time | Priority |
|---|---|---|
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

The processes are assumed to have arrived in the order P1, P2, P3, P4, and P5. All processes arrived at time 0.
จากข้อมูลของโพรเซสทั้งห้าตัวที่กำหนดให้ข้างต้น สมมุติให้โพรเซสทั้งหมดเข้ามาในระบบตามลำดับ P1, P2, P3, P4, และ P5 ณ เวลา time = 0

a) (40 points) draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: First-Come-First-Served, Shortest-Job-First, Non-preemptive priority (a smaller priority number implies a higher priority), and Round-Robin (time quantum = 1). Use FCFS to break tie.
จงวาด Gantt charts เพื่อแสดงการทำงานของโพรเซสแต่ละตัว เมื่อเราใช้ First-Come-First-Served, Shortest-Job-First, Non-preemptive priority (ตัวเลขน้อยหมายถึงค่าไพออริทีมาก) และ Round-Robin (time quantum = 1) หากมีตัวเลือกสองตัวให้เลือกทำงานโพรเซสที่เข้ามาก่อนเสมอ

First-Come-First-Served

Shortest-Job-First (preemptive)

Non-preemptive priority (a smaller priority number implies a higher priority)

Round-Robin (time quantum = 1)

b) (20 points) What is the turnaround time and waiting time of each process for each of these scheduling algorithms? (จงกรอกข้อมูลค่า waiting time ของโพรเซสแต่ละตัว เมื่อเราใช้สเก็ตดูลเลอร์แต่ละวิธี)

| algorithm | Waiting time (milliseconds) | | | | |
|---|---|---|---|---|---|
| | P1 | P2 | P3 | P4 | P5 |
| First-come-First-Served | | | | | |
| Shortest-Job-First | | | | | |
| Non-preemptive priority | | | | | |
| Round-Robin | | | | | |

c) (10 points) Which of the algorithms results in the minimum average waiting time (over all processes)? อัลกอรีทึมใดได้ค่าเฉลี่ยการรอคอย (average waiting time) ที่ต่ำที่สุด

**Question 9**: (30 points) Following is the structure of process Pi in Peterson's solution, please explain how the Peterson's solution preserves the mutual exclusion and satisfies the progress and the bounded waiting requirements. Note that Peterson's solution focuses on two processes and the two processes are shared the following variables:
ข้างล่างคือโครงร่างโปรแกรมของโพรเซส Pi ในวิธีการแบบ Peterson โปรดอธิบายว่าทำไมวิธีการดังกล่าวจึงสามารถทำให้ mutual execution คงอยู่และสามารถรองรับ progress และ bounded waiting ด้วย ทั้งนี้วิธีการแบบ Peterson จะเน้นไปที่สองโพรเซสและทั้งสองโพรเซสจะมีการใช้ตัวแปรต่อไปนี้ร่วมกัน

*int turn;*
*boolean flag[2];*

```
while (true) {
    flag[i] = TRUE;
    turn = j;
    while (flag[j] && turn == j);

        critical section

    flag[i] = FALSE;

        remainder section

}
```

ชื่อ..... รหัสนักศึกษา.....

**Question 10**: (20 points) Explain how the following solution to the Reader-Writer problem can cause a starvation problem? Who suffer?

จงอธิบายว่าทำไมโปรแกรมที่ให้ข้างล่างนี้จึงทำให้เกิดปัญหาความอดอยาก (starvation) และใครเป็นคนอดอยาก

```
int readerCount = 0;
Semaphore mutex = new Semaphore(1);
Semaphore db = new Semaphore(1);
```

```
/* Reader */
while(true) {
  mutex.acquire();
  ++readerCount;

  if(readerCount == 0) { // first reader
    db.acquire();
  }
  mutex.release();

  // read from the database

  mutex.acquire();
  --readerCount;

  if(readerCount == 0) { // last reader
    db.release();
  }
  mutex.release();
}
```

```
/* Writer */
while(true) {
  db.acquire();

  // update the database

  db.release();
}
```

```
/* semaphore acquire method */
acquire() {
  while value <= 0
    ; // no operation
  value--;
}

/* semaphore release method */
release(){
  value++;
}
```

ชื่อ..... รหัสนักศึกษา.....

**Question 11:** (20 points) List questions that you would like to see in this exam and give the answers to your questions. If you have more than one question, please assign the score to each of your questions. However, the total point can not exceed 20 points.

เขียนโจทย์ที่ท่านคาดหวังว่าจะเจอในข้อสอบชุดนี้แต่ไม่เจอ กรุณาเขียนโจทย์และตอบคำถามของท่านเองด้วย หากท่านมีโจทย์มากกว่าหนึ่งข้อ กรุณากำหนดคะแนนให้โจทย์แต่ละข้อด้วย (แต่ต้องรวมกันแล้วไม่เกิน 20 คะแนน)

ชื่อ..... รหัสนักศึกษา.....

**Extra**: (20 points) Please explain why the following schedule is possible under the two-phase locking protocol but not possible under the timestamp protocol. Assuming that transaction $T_0$ arrives before transaction $T_1$.

กรุณาอธิบายว่าทำไมสเก็ตดูลข้างล่างนี้จึงเกิดขึ้นได้ในระบบ two-phase locking protocol แต่จะไม่เกิดขึ้นในระบบ timestamp protocol สมมุติให้ทรานแซกชัน $T_0$ มาก่อนให้ทรานแซกชัน $T_1$

| Step | Transaction $T_0$ | Transaction $T_1$ |
|---|---|---|
| 1 | Lock-Share(A) | |
| 2 | Read(A) | |
| 3 | | Lock-Exclusive(B) |
| 4 | | Write(B) |
| 5 | | Unlock(B) |
| 6 | Lock-Share(B) | |
| 7 | Read(B) | |
| 8 | Unlock(A) | |
| 9 | Unlock(B) | |