TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Wiskunde en Informatica

Examination Operating Systems (2IN05)
on June 15, 2009, 14.00h-17.00h.

The exam consists of two parts that are handed in separately. Part I consists of knowledge-questions that must be answered *on a separate sheet.* Extensive explanations are not required in part I, compact answers will be appreciated, answers will be judged correct/wrong only. After you have handed in part I you may use the course book and the slides from the lectures for part II. (**Note**: just the lecture slides, no slides concerning exercises or previous exams or any other notes.)

The exam covers the entire course and is split in a block A and a block B+C part, graded separately. Students that do not want to take the block A or the block B+C exam (because they already completed that) can skip the exam they don't want to take. Students that do take both exams are kindly requested to give the answers on separate sheets.

Both exams are estimated at 1.5h. 20 minutes for part I and 70 minutes for part II. We do not enforce this, hence, students that just take one partial exam are lucky to have more time.

Work clearly. Read the entire parts first before you start. Scores for exercises are indicated between parentheses. There are 5 pages in total.

PART I
block A (4 points)

---

1. Give at least three motivations for the existence of operating systems.

2. What are the steps in processing a system call?

3. Under what circumstances is busy waiting acceptable? And where is it applied?

4. Mention the four correctness concerns in concurrent programs or systems.

5. Wat are the two principles of condition synchronization?

6. Give three ways to pass control to the kernel.

7. What is the difference between 'streaming' and 'message-based' communication? Give an example of both.

8. What are advantages (at least two) of the use of multiple threads above the use of multiple processes?

PART I
block B+C (4 points)

1. Give two machine instructions that can be used for implementing binary semaphores.

2. What is the working set and what determines the lower and the upper limit to its size?

3. Which (possibly conflicting) requirements determine the choice of the page-size in a virtual memory system?

4. In which two ways are devices mapped to the instruction set of a computer, and what are advantages and disadvantages?

5. What are advantages (at least two) of dynamic memory partitions over fixed partitions?

6. What is the return path from the kernel and where is it used?

7. Explain what is meant by spatial locality and by temporal locality.

8. What is the asynchronous part of a device driver?

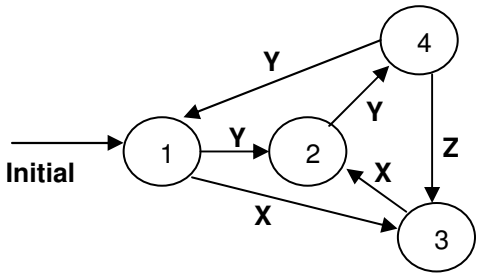
PART II
Block A (6 points)

2. There are three groups of threads, corresponding to the following three procedures.

var *st: int { initially 1 }*

proc *A* (*i*: *int*) = **|[while** *true* **do** *X* **od]|;**
proc *B* (*i*: *int*) = **|[while** *true* **do** *Y* **od]|;**
proc *C* (*i:int)* = **|[while** *true* **do** *Z* **od]|;**

Execution of these threads must be constrained according to the following state transition diagram. Variable *st* records the state and must be modified together with the execution of the actions *X, Y* and *Z.*



a) (2.5 pt) Synchronize this system using condition variables according to this diagram. Add the modifications of variable *st* and make sure that proper exclusion is guaranteed.
b) (1.5) Discuss fairness of your solution; if it is not fair modify it such that it becomes fair.
c) (1.0) Use action synchronization with just semaphores (no condition variables) to limit execution to
$$I: 0 \leq \mathbf{c}Y\text{-}\mathbf{c}Z \leq 1$$
Actions must still exclude each other. What is the resulting behavior in terms of actions sequences?
d) (1.0) Is it required that accesses to *st* are performed inside a critical section only? Is it required that the actions *X, Y* and *Z* occur only within critical sections? Explain your answer.

PART II
Block B+C (6.5 points)

1. A given file system has the file descriptors contained in directory nodes. The file system is mapped onto a disk of 100000 blocks. A four-byte integer is needed to address each block. A block is 1024 bytes.
   a) (1.5 pt) Assume a linked-list allocation strategy, where the linked list is stored in the first blocks on the disk (Fig.10-12c in the book). How many blocks are needed for this list? What is the largest possible file that can be represented in this way?
   b) (2.0 pt) Design your own solution (not necessarily using a linked list) with the following restrictions. 1) Files shorter than 128 bytes require no additional disk access (besides reading the directory); 2) Blocks of files with a length until 1K can be accessed directly (with one access besides the access of the descriptor); 3) No restrictions for larger files. Explain what you need in the descriptor and also explain in words how a file is accessed.

2. (1 pt) Show how to realize a binary semaphore using *SWAP*. The *SWAP* primitive atomically swaps values of two variables.
   - *SWAP (s,t):* < |[**var** *tmp: int; tmp := s; s := t; t := tmp*]| >

3. Given is the following taskset.

Task	R1	R2
A	2	3
B	1	2
C	3	3
D	2	2

   Columns *R1* and *R2* represent maximum numbers of resources of type *R1* and *R2* needed by the tasks.
   a) (1.0) Is a deadlock possible when there are 6 type *R1* resources and 8 type *R2* resources? Motivate your answer.
   b) (1.0) Now there are 4 resources available of both types. Consider the situation after *A: req(R1, 2); A: acq(R1,2); B: req(R2, 2): B: acq(R2,2).* Will the bankers algorithm grant the following requests? Motivate your answer with a diagram.
      i. C: req(R1, 2)
      ii. D: req(R1, 2)