TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Wiskunde en Informatica

Examination Operating Systems (2IN05)
on January 9, 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 exam (because they already completed that) can skip the part pertaining to block A. 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 the block B+C 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. What is meant with the term ‘race condition’? Give an example (simple, please).

4. What is meant with the concept of ‘fairness’? Give a simple example of fair and non fair behavior.

5. In a program with condition variables we use critical sections of the form
   P(m); **while not** *COND* **do** *Wait (m, cv)* **od;** *critical section body*; *Signalling; V(m)*
   with *COND* a boolean expression and *cv* a condition variable. Explain why a repetition is in general better at this place than a selection (i.e., an **if**-statement).

6. What is the difference between the *signal-and-continue* and the *signal-and-wait* signalling policies of a monitor?

7. Explain the difference between a process and a thread.

8. Explain the difference between a mode switch and a context switch.

PART I
block B+C (4 points)

1. Mention three reasons to use buffering inside the kernel.

2. Loading a new page implies overwriting (replacing) an old one.
   a. explain why it is generally better to replace an unmodified page;
   b. give circumstances when replacing a modified page is advisable.

3. Given are the following two file organizations:
   a. a linked list of blocks
   b. a series of adjacent blocks

   Explain which of the following four concerns are better addressed better by a. or b. : efficient file access, file extension, file seeking, maintaining free disk space.

4. Give the four criteria that determine deadlock occurrence.

5. Explain the two places in the i/o system where standardization occurs.

6. What is *system reliability?*

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)

1. There are two groups of threads, corresponding to the following two procedures.

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

   a) (1 pt) Synchronize this system using action synchronization such that *X* and *Y* actions alternate, beginning with *X*. Explain and motivate your reasoning.
   b) (3 pt) As a new exercise, introduce variables and conditional critical regions to synchronize according to the following:
      - After three *X* actions, two *Y* actions occur; after two *Y* actions, three *X* actions occur, and so on.
      - Concurrent execution of actions of the same type is allowed and should not be forbidden by your solution.
   c) (1 pt) Discuss the fairness of your solution in b. and propose a fair alternative if your solution is not fair.

2. (1 pt) Consider the following procedure.

   var *x: int;* { *x* is initialized to 0 }
   proc *A* = |[**var** *r: int;* **while** *x* <> *10* **do** *r:=x; x := r+1* **od**]|

   Assume that two threads both execute *A,* hence, two instances of *A* run concurrently. Is it possible for variable *x* to obtain a value larger than 10? If yes, give a trace; if no, give an argument.

PART II
Block B+C (6.5 points)

1. Consider mapping a virtual memory of 1GB onto a physical memory organized into 256 page frames of 4KB each. Moreover, assume that the smallest addressable unit is 1 byte.

   a) (0.5 pt) Does the page table fit in main memory? Motivate your answer with a calculation.
   b) (0.25 pt) Does the frame table fit in a single page? Motivate your answer with a calculation.

   Given the same physical memory organization as above and a memory management strategy that always keeps at least 1 page frame per active process resident in main memory for paging purposes.

   c) (0.75 pt) What is the maximum size of the virtual address space given that any memory access may yield at most two page faults?
   d) (0.25 pt) Give the corresponding address map (algorithm).

   Furthermore, assume that there is hardware support in the form of a Translation Look-aside Buffer. Let p0, p1, and p2 be the probabilities that a memory access generates 0,1, or 2 page faults, respectively. Let the time for a memory access without a page fault be t, the time for a memory access with a single page fault 8t, and the time for a memory access with two page faults 12t. Consider the following cases:

   Case 1: p0 = 3/4, p1= 3/16 and p2 = 1/16 with usage of the TLB.
   Case 2. p0 = 1, p1 = p2 = 0 without usage of the TLB

   e) (0.75 pt) Determine the hit-ratio of the TLB at which the two cases have the same performance. You may assume that there is no time penalty for accessing the TLB.

2. (1 pt) Show how to realize a binary semaphore using Fetch & Add. The Fetch & Add primitive atomically adds a value to a variable, returning the old value.
   - *F&A (s,x): <rtn := s; s := s+x; return (rtn)>*

3. Given is the following taskset.

Task	Arrival	Computation	R1	R2
A	0	12	2	3
B	2	4	1	2
C	3	6	3	2
D	1	8	1	3

Columns *R1* and *R2* represent maximum amounts of resources of type *R1* and *R2* needed by the tasks.

a. (1 pt) Determine turnaround times for all tasks under the scheduling policies Shortest Remaining Time and Round Robin with a quantum of 1.
b. (0.5 pt) For these two cases, determine the amount of resources *R1* and *R2* that are required by the system.
c. (1.5) We limit resource availability to 3 for type *R1* and 4 for type *R2*. Assume that 1 resource of both types is requested upon task start and the remainder after 50% of the computation time has been spent.
    - Adjust the given round robin schedule to take resource limitations into account. Indicate where a deadlock occurs.
    - Adjust the round robin schedule to include decisions of the bankers algorithm to avoid deadlock. Use diagrams to explain your findings.