THE UNIVERSITY OF AUCKLAND

SECOND SEMESTER, 2015
Campus: City

COMPUTER SCIENCE / SOFTWARE ENGINEERING

Operating Systems

TEST

(Time Allowed: 45 minutes)

NOTE:

- Calculators are NOT permitted.
- Compare the test version number on the Teleform sheet supplied with the version number above. If they do not match, ask the test supervisor for a new sheet.
- Enter your name and student ID (in pencil) on the Teleform sheet. Your name and Student Id should both be entered left aligned. If your name is longer than the number of boxes provided, truncate it.
- Answer all questions on the Teleform answer sheet provided.
- Use a dark pencil to shade in your answers in the multiple choice answer boxes on the Teleform sheet. Check that the question number on the sheet corresponds to the question number in this question book. If you spoil your sheet, ask the supervisor for a replacement.
- Questions are worth marks as indicated. There are 26 questions worth 30 marks in total.

THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK.

MULTIPLE CHOICE QUESTIONS

For each question, choose the best answer according to the information presented in lectures and in the textbook. Select your preferred answer on the Teleform answer sheet by shading in the appropriate box in pencil. There are 26 questions worth 30 marks in total.

Question 1
[1 mark] Which of the following was NOT a theme/model for discussing operating systems in lectures?

(a) The bare machine model
(b) The resource allocator model
(c) The manager model
(d) The dustbin model
(e) The onion model

Question 2
[1 mark] Which of the following statements about environmental subsystems is FALSE?

(a) The MS-DOS API could be provided as an environmental subsystem.
(b) Windows NT and several of its descendants came with a number of environmental subsystems.
(c) MS-DOS was the first operating system with environmental subsystems.
(d) An environmental subsystem is conceptually very like application virtualisation.
(e) An environmental subsystem provides an API which enables applications written for one operating system to run on another one.

Question 3
[1 mark] Which of the following shows the correct chronological ordering (earliest to latest) of these operating system and hardware developments?

(a) batch systems, resident monitors, interrupt handling, SPOOLing
(b) interrupt handling, protected memory, resident monitors, time sharing systems
(c) SPOOLing, resident monitors, batch systems, time sharing systems
(d) resident monitors, disk drives, time sharing systems, batch systems
(e) resident monitors, protected memory, batch systems, time sharing systems

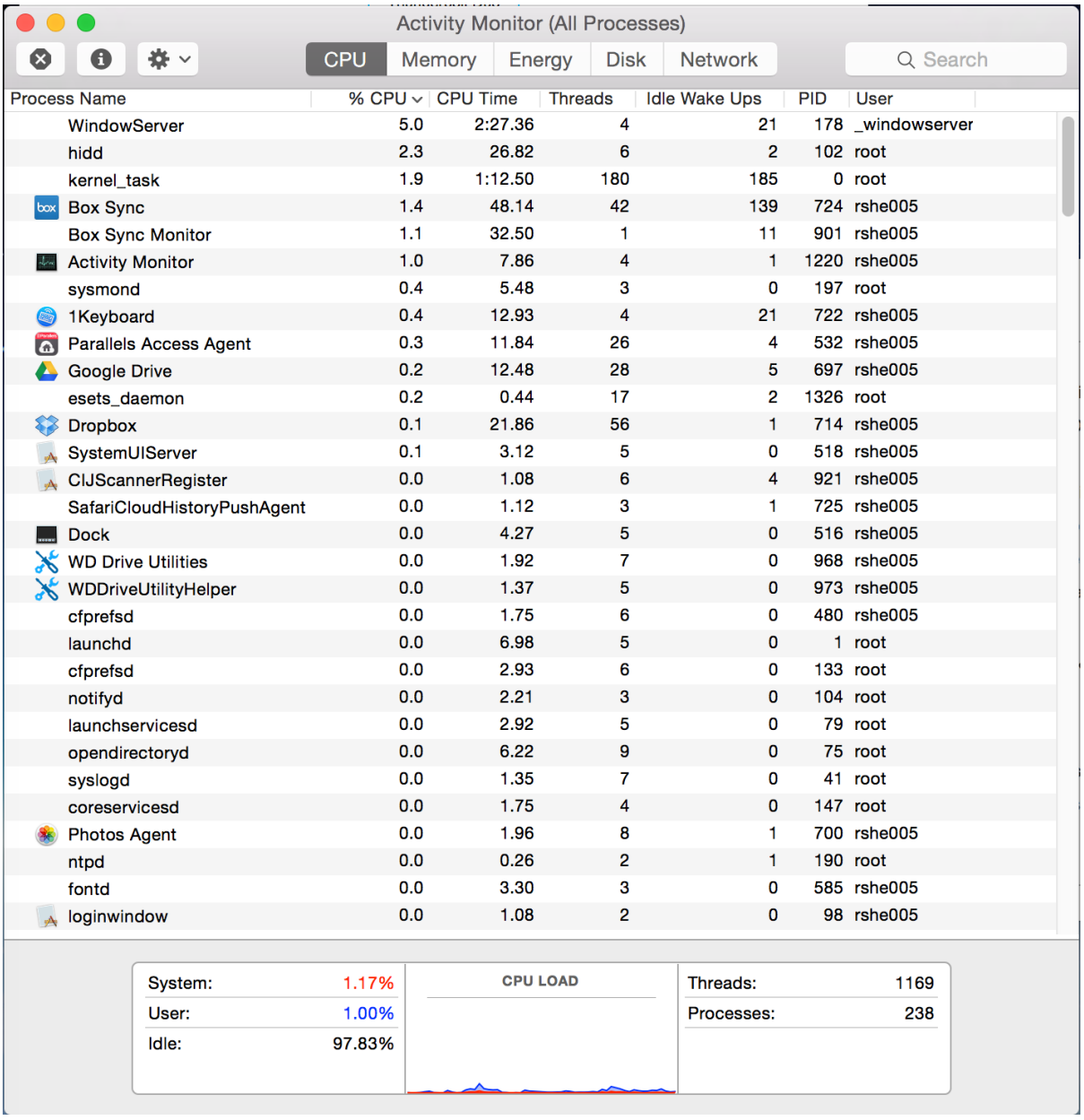
Question 4
[1 mark] Why were time sharing systems developed?

(a) To give multiple users direct access to the computer at the same time.
(b) Because the hardware was too expensive to allow individuals to use the computer.
(c) To keep the computer running close to its maximum capacity.
(d) To use terminal hardware from batch systems which was not being used at the time.
(e) To enable multiprogramming on single CPU machines.

Question 5

[1 mark] Which of these types of programs/operating systems first had all of the standard operating system requirements of managing processes, protecting memory and disk file systems?

(a) Time sharing systems
(b) Resident monitor systems
(c) Personal computer systems
(d) Distributed operating systems
(e) Batch systems



Question 6
[1 mark] What does the above image of processes in a modern operating system indicate?

(a) That modern machines have a large number of processes alive in the system at any one time.
(b) That many processes are owned by the super user (root).
(c) That very few processes in modern systems are single threaded.
(d) That most of the time, modern desktop/laptop computers do little processing.
(e) All of the above

Question 7
[1 mark] Which of the following statements is FALSE?

(a) Early phone operating systems had to be very efficient because of memory and battery requirements.
(b) Smart phone operating systems have always included virtual memory management.
(c) The Symbian phone operating system has true memory protection for processes and kernel.
(d) Android is based on a Linux kernel.
(e) iOS does not page data to backup storage.

Question 8
[1 mark] Which of the following is NOT an advantage of using virtual machines?

(a) The ability to simultaneously run multiple operating systems on the same machine.
(b) Ease of creation of new servers based on existing ones.
(c) Live migration of services.
(d) The development and testing of changes to operating systems.
(e) None of the above.

Here is part of the `man` page for `sched_scheduler` the system call to set the scheduling policy for a Linux process:

```
Currently, Linux supports the following "normal" (i.e., non-real-time)
scheduling policies:

       SCHED_OTHER   the standard round-robin time-sharing policy;

       SCHED_BATCH   for "batch" style execution of processes; and

       SCHED_IDLE    for running very low priority background jobs.

The following "real-time" policies are also supported, for special time-
critical applications that need  precise  control over the way in which
runnable processes are selected for execution:

       SCHED_FIFO    a first-in, first-out policy; and

       SCHED_RR      a round-robin policy.
```

Question 9
[1 mark] Which of the policies listed above would you choose for a shell login process?

(a) SCHED_FIFO
(b) SCHED_BATCH
(c) SCHED_RR
(d) SCHED_OTHER
(e) SCHED_IDLE

Question 10
[1 mark] Which of the policies would you choose for a process which you don't want interfering too much with the smooth running of other processes?

(a) SCHED_IDLE
(b) SCHED_FIFO
(c) SCHED_BATCH
(d) SCHED_RR
(e) SCHED_OTHER

Question 11
[1 mark] Which of the following is NOT a true difference between preemptive and non-preemptive (or cooperative) scheduling?

(a) Non-preemptive scheduling is more efficient than preemptive scheduling.
(b) Non-preemptive schedulers are less predictable then preemptive schedulers.
(c) Preemptive scheduling makes it easier to protect critical sections of code.
(d) Preemptive scheduling provides greater control over runaway processes.
(e) None of the above.

The following four questions concerning assignment 1 all assume there is only ONE processor or core in the system, i.e. only one process running at a time.

Question 12
[1 mark] In assignment 1 where does the dispatcher select the next process to run from?

(a) From the bottom of the stack of runnable processes.
(b) From the top of the stack of runnable processes.
(c) From the top of the set of waiting processes.
(d) From the bottom of the set of waiting processes.
(e) The process which currently has the keyboard focus.

Question 13
[1 mark] In assignment 1 which of the following steps must be taken by the dispatcher after data is entered into a waiting interactive process?

(a) If there is a currently running process it is paused. The interactive process which received the input is moved to the top of the runnable stack. The interactive process is restarted from its current position.
(b) The data buffer of the interactive process is cleared. The interactive process is moved to the top of the waiting list. The currently running process continues.
(c) If there is a currently running process it is paused. The next process on the stack is restarted. The interactive process is moved from its current position to the bottom of the stack.
(d) The process with the keyboard focus is paused. The process at the top of the runnable stack is moved to the first available position of the set of waiting processes. The interactive process continues running.
(e) The process with the keyboard focus is restarted. The original process at the top of the stack continues running.

Question 14
[1 mark] Given the following sequence of commands being redirected from a file into the a1.py program from assignment 1 which process would be running when the `p` command begins.

```
nbnif2
10
nbnbnit3
pq
```

(a) 1
(b) 4
(c) 5
(d) 3
(e) 2

Question 15
[1 mark] What could cause the `State.killed` check in the following code from assignment 1's Process class to be true?

```
    def main_process_body(self):
        self.block_event.wait()
        if self.state == State.killed:
            _thread.exit()
        self.iosys.write(self, "*")
        sleep(0.1)
```

(a) The process has set its state to killed which is not allowed and so it needs to exit.
(b) The dispatcher has indicated that the process has been killed and the process thread needs to check to see if it should exit.
(c) There has been an error in the curses code which means that this process thread is no longer connected to a panel and it should exit.
(d) A waiting process may have been woken up accidentally by the IO system and needs to be killed before it can run with incorrect data.
(e) None of the above.

Use the following process information in the next two questions.

Process	Arrival time	CPU burst time
A	0	3
B	1	4
C	2	1
D	6	1

Question 16

[2 marks] What is the average waiting time using first come first served (FCFS) without preemption (i.e. when a new process arrives, even if it has a shorter burst than the currently running process it does not preempt the running process)?

(a) 2.25
(b) 1.5
(c) 2
(d) 2.5
(e) None of the above

Question 17

[2 marks] What is the average waiting time using shortest job first (SJF) with preemption (i.e. if a process with a shorter remaining burst time arrives it can preempt the running process)? If two processes have the same shortest compute time and one is already running, leave it running.

(a) 1.5
(b) 2.25
(c) 2
(d) 2.5
(e) None of the above

Use this information for the next two questions. Here are some possible schedules for two real-time processes A (3, 5, 5) and B (3, 9, 9) - where the numbers of compute time, period and deadline are all in milliseconds. The tables show only the first 11 milliseconds of the calculated schedules. Each millisecond is used to recalculate the priorities.

W)

A	A	B	B	A	A	B	A	A	B	B

X)

A	A	A	B	B	B	A	A	A	B	B

Y)

A	A	A	B	B	A	A	B	A	B	A

Z)

A	A	A	B	B	A	A	A	B	B	B

Question 18
[2 marks] Which of the schedules is generated by Shortest Completion Time (SCT)? If the completion time is the same for both A and B give the priority to the currently running process, if there is no currently running process choose A.

(a) W
(b) Z
(c) X
(d) Y
(e) None of the above.

Question 19
[2 marks] Which of the schedules is generated by Least Slack Time (LST)? If the slack time is the same for both A and B give the priority to the currently running process, if there is no currently running process choose A.

(a) Y
(b) X
(c) W
(d) Z
(e) None of the above.

Here is a very simple attempt at implementing a lock:

```
lock:
      while locked
      end
      locked = true
unlock:
      locked = false
```

Question 20
[1 mark] Which of the following statements about the above lock code is FALSE?

(a) It doesn't work, multiple threads could gain the lock simultaneously.
(b) It is unfair, there is no guarantee a thread will progress through the lock.
(c) It works most of the time. But most of the time is not good enough.
(d) It wastes CPU cycles checking the value of the locked variable.
(e) It doesn't work on multicore or multiprocessors but it does work on single processors.

Question 21
[1 mark] Even with the Global Interpreter Lock (GIL) in standard Python which means that only one thread in a Python program can run Python byte code at a time there is still a need for locks. Which of the following reasons best explains why this is so?

(a) The GIL is associated with threads, the locks which are still required are associated with processes.
(b) Instruction reordering can still occur even with the GIL.
(c) If run on a multicore machine more than one thread can run Python byte code simultaneously in the Python program.
(d) A thread could still be paused (and another thread scheduled) while it is accessing a resource.
(e) The GIL sometimes doesn't work allowing multiple threads to run simultaneously.

Question 22
[1 mark] How does the Bakery algorithm guarantee a unique ordering of process requests to use a shared resource?

(a) The algorithm uses the scheduling priority of each process as its unique ordering identifier.
(b) The process id of the process is appended to the ticket number distributed by the algorithm and this is a unique ordering identifier.
(c) The ticket number distributed by the algorithm is appended to the process id of the process and this is a unique ordering identifier.
(d) A hash value is generated from the process id of the process and the ticket number distributed by the algorithm.
(e) The ticket number distributed by the algorithm is kept unique by the use of a spin lock.

Question 23
[1 mark] Which of the following statements about semaphores is TRUE?

(a) When returning a resource with no process waiting the semaphore value stays the same.
(b) Semaphores are more powerful than monitors and locks.
(c) No semaphore operations have to be atomic.
(d) Semaphore waits and condition variable waits are the same.
(e) None of the above.

Question 24
[1 mark] Which of the following statements about the producer/consumer problem is FALSE?

(a) Each item of data produced must be consumed.
(b) The producer/consumer problem can be generalized to multiple producers and consumers.
(c) Each item of data produced must be consumed only once.
(d) The order of data being consumed must match the order it was produced.
(e) None of the above.

Question 25

[1 mark] How many times does the ps command get called from the following program? Remember that in Python 0 is equivalent to False.

```
import os
count = 0
while count < 2:
    os.system('ps')
    if os.fork():
        os.system('ps')
    count += 1
```

(a) 2
(b) 6
(c) 8
(d) 4
(e) None of the above

Question 26

[1 mark] In the following pseudocode solution to the Dining Philosophers' problem, what could go wrong? The `simultaneous_wait` only returns when both parameters are available.

```
do forever:
  status = "waiting"
  simultaneous_wait(left, right)
  status = "eating"
  simultaneous_signal(left, right)
  status = "thinking"
```

(a) An unlucky process might never be able to get both left and right forks simultaneously.
(b) Some processes will get extra turns to eat on a regular basis, violating the principle of fair treatment.
(c) One process can monopolise the forks causing those processes on its left and right to be unable to eat.
(d) All processes might pick up one fork causing deadlock.
(e) Nothing is wrong; this is a good solution to the problem.

Rough Working – This page will not be marked