

Monash University

EXAMINATIONS 1999
FACULTY OF INFORMATION TECHNOLOGY

TITLE OF PAPER CSE2302 Operating Systems

INSTRUCTIONS TO CANDIDATES

1) CANDIDATES ARE REMINDED THAT THEY SHOULD HAVE NO BOOKS, NOTES PAPER OR OTHER MATERIAL IN THEIR POSSESSION UNLESS THEIR USE IS SPECIFICALLY PERMITTED BY "INSTRUCTIONS TO CANDIDATES" SET OUT BELOW

- 2) READING TIME IS OF 10 MINUTES DURATION
- 3) EXAMINATION TIME IS OF 3 HOURS DURATION
- 4) THERE ARE THREE SECTIONS: A, B, C; EACH SECTION IS WORTH 40 MARKS TOTAL
- 5) ANSWER ALL QUESTIONS IN EACH SECTION
- 6) WRITE ALL ANSWERS ON THE EXAMINATION PAPER
- 7) TOTAL MARKS 120
- 8) CALCULATORS ARE NOT ALLOWED

SECTION A: Multiple Choice (40 marks total)

Each question in this section is worth *2 marks*. Circle the one alternative (a,b,c,d,e) that you think is the **best** answer to the question. If you make a mistake, draw a single stroke “/” through your incorrect answer, then circle your revised answer.

- 1) A kernel is:
 - a) that part of an operating system that is responsible for resource allocation
 - b) that part of an operating system that is always running
 - 2** c) that part of an operating system that is always resident in memory
 - d) all of the above
 - 1** e) some (but not all) of the above

- 2) Early operating systems used punched cards
 - a) for program entry
 - b) for data entry
 - c) for job control
 - 2** d) all of the above
 - 1** e) some (but not all) of the above

- 3) Spooling in an operating system allows
 - 2** a) the overlap of input and output with job execution
 - b) the overlap of multiple processes or jobs
 - c) email to be read offline
 - d) all of the above
 - e) none of the above

- 4) Multiprogramming in an operating system allows
 - a) more than one programming language to be used
 - 2** b) more than one user on the system
 - c) more than one CPU in the system
 - d) all of the above
 - e) none of the above

- 5) Symmetric multiprocessing
 - 1** a) requires multiple processors, all the same
 - 1** b) requires each processor to run identical copies of the operating system
 - 1** c) allows many processes to run at once without performance degradation
 - 2** d) all of the above
 - e) none of the above

- 6) An interrupt
 - 1** a) is essential to the design of operating systems
 - 1** b) improves the scheduling of waiting processes
 - 1** c) is always generated by a software trap
 - 2** d) all of the above
 - e) none of the above

- 7) A direct memory access (DMA) is
- a) a fast kind of interrupt
 - 2** b) a fast kind of I/O transfer
 - c) a fast translation look-aside buffer
 - d) all of the above
 - e) none of the above
- 8) Supervisor mode
- 1 a) protects the operating system against damage
 - 1** b) prevents the user from accessing I/O devices
 - 1** c) allows the execution of privileged instructions
 - 2** d) all of the above
 - e) some (but not all) of the above
- 9) The base register in memory protection systems
- a) allows access to any location in memory
 - 2** b) defines the starting address of a process image
 - c) contains the ORG address from the assembler code
 - d) all of the above
 - e) none of the above
- 10) Privileged instructions are required to
- 1 a) change the limit register of a memory protection system
 - 1** b) handle a page fault
 - 1** c) load the timer
 - 2** d) all of the above
 - e) none of the above
- 11) A system trap occurs when
- 2** a) a system call is issued
 - b) a process is waiting
 - c) the interrupt vector overflows
 - d) all of the above
 - e) some (but not all) of the above
- 12) Dispatch latency is
- a) the time a process waits before being executed by the dispatcher
 - b) the time a process waits before being terminated by the dispatcher
 - 1** c) the time it takes for the dispatcher to start a waiting process
 - 2** d) the time it takes for the dispatcher to switch between processes
 - e) the time a process executes before being pre-empted by the dispatcher
- 13) Memory-mapped I/O is used because it
- 2** a) reduces the need for interrupts
 - 1** b) improves the speed of the I/O bus
 - c) removes the need for separate I/O instructions
 - 1** d) improves the programming of some I/O transfers
 - e) all of the above
- 14) A demand segmented memory management scheme

- a) has homogeneous units of memory allocation
 - b) suffers from internal fragmentation
 - 2** c) suffers from external fragmentation
 - d) two or more of the above
 - e) none of the above
- 15) Given a request for an area of size n bytes, First-Fit allocation for storage space (memory or disk) searches the list of free areas for
- a) the area that is the smallest possible over all areas on the free list
 - b) the first area that is $< n$
 - 2** c) the first area that is $\geq n$
 - d) the best area that minimizes the wasted space over all areas
 - e) none of the above
- 16) Indexed allocation in file system organization is where
- a) each file is stored in contiguous sectors
 - b) the file sectors are linked in a list
 - c) pointers to the file sectors are stored in an indexed hash table
 - 2** d) each file has its own block of pointers to the sectors of the file
 - e) none of the above
- 17) Starvation in disk scheduling may occur in the algorithms:
- a) FCFS (First Come First Served) and SCAN
 - b) FCFS and LOOK
 - 1** c) SSTF (Shortest Seek Time First) and C-SCAN
 - 1** d) SSTF and FCFS
 - 2** e) none of the above
- 18) Suppose $D_1 = \{ \langle O_1, rw \rangle, \langle O_2, rx \rangle \}$, $D_2 = \{ \langle O_1, rp \rangle, \langle O_2, w \rangle \}$ are two protection domains, where r=read right, w=write right, x=execute right and p=print right. Then
- a) In D_1 , O_1 can be executed
 - b) In D_1 , O_1 can be printed
 - 2** c) In D_2 , O_1 can be printed
 - d) In D_2 , O_2 can be read
 - e) In D_2 , O_2 can be executed
- 19) In question 18 above, suppose object O_1 is executing in domain D_1 and calls O_2 which then starts executing in domain D_2 . Then
- a) O_2 can print itself
 - b) O_2 can write to O_1
 - c) O_2 can copy itself
 - d) O_1 can write to O_2
 - 2** e) none of the above
- 20) Login emulation programs that record passwords are a form of the security threat known as
- 2** a) Trojan Horse
 - 1** b) Trap Door
 - c) Worm
 - d) Virus
 - e) XINU

SECTION B: Answer all questions in this section (40 marks total)

1)

- a) Briefly explain what an operating system is, and what its main purpose from a user perspective should be.

Answer: *An operating system exists to*

i) manage resources [2 marks]

ii) provide transparent access for the user to those resources [2 marks]

[4 marks]

- b) Name and define the five states of a process as managed by an operating system. Draw a state diagram showing the transitions between these states, and label the edges with the events that cause those transitions.

Answer:

i) created

ii) ready

iii) running

iv) waiting

v) terminated

[5 marks]

Transitions are

i) created to ready

ii) ready to running

iii) running to ready

iv) running to waiting

v) running to terminated

vi) waiting to ready

1 Mark for diagram, lose $\frac{1}{2}$ mark for any missing transitions

[3 marks]

[8 marks]

2) *The Cigarette Smokers Problem.* Consider a system with three *smoker* processes and one *agent* process. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper, and matches. Each smoker has a supply of exactly one of these ingredients: one has tobacco, the next paper, and the third matches. The agent has a supply of all three ingredients, and on each cycle places two arbitrarily chosen ingredients on the table. The smoker who has the third ingredient then makes and smokes a cigarette, signalling the agent on completion, who then puts out two more arbitrarily chosen ingredients, and the cycle repeats. Write a program (pseudo code will suffice) to synchronize the agent and the smokers. You may assume the following shared data structures:

```
var a : array [0..2] of semaphore /* initialized to 0 */;
    agent : semaphore /* initialized to 1 */;
```

Answer: **Agent Process**

```
while (1) {
    wait(agent);
    i = choose_ingredient();
    signal(a[i]);
}
```

[6 marks]

Smoker Process (*parameterized by missing ingredient i*)

```
while (1) {
    wait(a[i]);
    smoke_cigarette();
    signal(agent);
}
```

[6 marks]

[12 marks]

3)

a) An operating systems supports a paged virtual memory, using a processor with a cycle time of 1 microsecond. Pages have 1024 words, and the paging device is a drum that rotates at 3000 revolutions per minute and transfers 1 million ($\approx 2^{20}$) words per second. The following statistical measurements were obtained from the system.

- 0.1 percent of all instructions executed access a page other than the current page.
- Of the instructions that accessed another page, 80 percent accessed a page already in memory.
- When a new page was required, the replaced page was modified 50 percent of the time.

Calculate the effective access time on this system, assuming that there is only one process running, and that the processor is idle during drum transfers.

Answer: *The effective access time (EAT) is defined as*

$$EAT = \alpha \times MAT + (1 - \alpha) \times PRT$$

where α is the hit rate, 1 if all to memory, 0 if all references page fault; MAT is the Memory Access Time; PRT is the Page Replacement Time.

Since the Page Replacement Time itself depends on how often we have to write out the old page, we define that further:

$$\begin{aligned} PRT &= \beta \times PTT + 2(1 - \beta) \times PTT \\ &= (2 - \beta)PTT \end{aligned}$$

where PTT is the Page Transfer Time, β is the fraction of pages that don't have to be written out, and the factor 2 is because when the pages do have to be written out, 2 page transfers are required.

The Page Transfer Time has two parts: the actual transfer time, which is $1024/2^{20} = 2^{-10}$ seconds or 1ms; and the disk latency, which is half the time for the disk to rotate once $\frac{1}{2} \times \frac{60}{3000} = \frac{1}{100}$ seconds or 10ms; total 11ms.

$$PRT = 1.5 \times 11 = 16.5ms$$

The hit rate is 99.9%, plus 80% of the 0.1% remaining, i.e., 99.98%

Hence

$$\begin{aligned} EAT &= 0.9998 \times 1 + 0.0002 \times 16500 \\ &= 0.9998 + 3.3 \\ &= 4.2998\mu s \end{aligned}$$

[4 marks]

b) Consider the following page reference string:
 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

How many page faults would occur for the following replacement algorithms and physical memory sizes? Assume the memory is initially empty, and include the initial frame filling page faults.

- i) LRU with 3 frames
- ii) FIFO with 3 frames
- iii) Optimal with 3 frames
- iv) LRU with 6 frames
- v) FIFO with 6 frames
- vi) Optimal with 6 frames

Answer:

f f f f f f f f f f f f f f f 15 faults
 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3
 1 2 3 4 2 1 5 6 6 1 2 3 7 6 3 3 1 2

f f f f f f f f f f f f f f f 16 faults
 1 2 3 4 4 1 5 6 2 1 1 3 7 6 6 2 1 1 3 6
 1 2 3 3 4 1 5 6 2 2 1 3 7 7 6 2 2 1 3
 1 2 2 3 4 1 5 6 6 2 1 3 3 7 6 6 2 1

f f f f f f f f f f f f f f f 11 faults
 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
 1 2 2 4 2 1 1 6 2 1 2 3 7 6 3 2 1 2 3
 1 1 1 4 2 2 1 6 6 6 6 3 7 6 3 3 1 2

f f f f f f f f f 7 faults
 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3
 1 2 3 4 2 1 5 6 6 1 2 3 7 6 3 3 1 2
 1 1 3 4 2 1 5 5 6 1 2 2 7 6 6 6 1
 3 4 4 4 4 5 6 1 1 1 7 7 7 7
 3 3 3 3 4 5 5 5 5 5 5 5 5

f f f f f f f f f f f f f f f 10 faults
 1 2 3 4 4 4 5 6 6 6 6 6 7 7 7 7 1 2 3 3
 1 2 3 3 3 4 5 5 5 5 5 6 6 6 6 7 1 2 2
 1 2 2 2 3 4 4 4 4 4 5 5 5 5 6 7 1 1
 1 1 1 2 3 3 3 3 3 4 4 4 4 5 6 7 7
 1 2 2 2 2 2 3 3 3 3 4 5 6 6
 1 1 1 1 1 2 2 2 2 3 4 5 5

f f f f f f f f f 7 faults
 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3
 1 2 3 4 2 1 5 6 6 1 2 3 7 6 3 3 1 2
 1 1 3 4 2 1 5 5 6 1 2 2 7 6 6 6 1
 3 4 4 4 4 5 6 1 1 1 7 7 7 7
 3 3 3 3 4 5 5 5 5 5 5 5 5

[6 marks]

4)

- a) Give three data structures that might be used to represent directory information in a file system, and state the advantages offered by that data structure.

Answer: *directory information includes stuff such as file address, file size, file name, file type (text/binary/executable/subdirectory)*

- i) **fixed size area for each entry/array:** Easy to index and access subfields*
- ii) **variable sized area:** copes with arbitrary sizes better, such as very large files, or very long file names*
- iii) **pointers to other disk blocks:** Copes with variable length data, easy to index within directory*

[3 marks]

- b) For each of your three data structures, identify any problems that might arise in implementing directories with that data structure.

Answer:

- i) **fixed size area for each entry/array:** Limits the size of key fields, such as maximum length of a file, or file name*
- ii) **variable sized area:** more difficult to manage and index - what happens it becomes so large that the disk block is overflowed, for example?*
- iii) **pointers to other disk blocks:** Requires an extra disk access to retrieve the information*

[3 marks]

SECTION C: Answer all questions in this section (40 marks total)

1)

a) Name the three queues involved in process scheduling.

Answer:

- i) job queue*
- ii) ready queue*
- iii) wait queue (or I/O queue)*

[3 marks]

b) Draw a diagram showing the movement of processes between those queues.

Answer:

- i) job to ready*
- ii) ready to (running)*
- iii) (running) to wait*
- iv) (running) to ready*
- v) wait to ready*

1 for diagram, lose $\frac{1}{2}$ for each wrong or missing arc

[3 marks]

c) What is the difference between short-term and long-term scheduling?

Answer:

- *long-term scheduling moves jobs into the job queue*
- *short-term scheduling moves jobs in and out of the ready queue*

[2 marks]

2) Consider the following solution to the bounded buffer problem:

```
Data:
    int n,in,out;
    ItemType buffer[n],nextp,nextc;

Producer:
    do {
        ...
        produce(nextp);
        ...
        while ((in+1) % n == out) {noop;}
        buffer[in] = nextp;
        in = (in+1) % n;
    } until 0;

Consumer:
    do {
        while (in==out) {noop;}
        nextc = buffer[out];
        out = (out+1) % n;
        ...
        consume(nextc);
        ...
    } until 0;
```

a) This is a valid solution to the problem, but is inefficient in that it only uses $n-1$ slots of the buffer. Explain why.

Answer: Assume that the buffer is initialized with $in=out=0$. Next assume that no items are removed from the buffer, so that $out=0$, while items are added by the producer. The number of items added will thus be equal to in . The condition for the buffer being full is $(in+1)\%n == out$, which, since $out=0$, reduces to $(in+1)\%n=0$. But if in hasn't wrapped around, this is $in+1=n$, or $in=n-1$. Since nothing has been removed, this means that the number of items added, in or $n-1$, is the maximum capacity of the buffer. QED.

[2 marks]

b) It also uses busy waiting. Explain what this is, and why it is inefficient.

Answer: The busy wait occurs in the `while () {noop;}` code fragments. It involves repeated testing of a condition, waiting for it to become false. The repeated execution consumes CPU cycles to no real purpose and is wasteful.

[4 marks]

- c) Rewrite the bounded-buffer solution, so that it uses all n slots of the buffer, and uses semaphores to avoid the busy waiting.

Answer:

Data:

```
int n,in,out,count=0;
ItemType buffer[n],nextp,nextc;
semaphore notempty,notfull; /*initialized to 0,1*/
```

Producer:

```
do {
    ...
    produce(nextp);
    ...
    if (count==n) {wait(notfull);}
    buffer[in] = nextp;
    in = (in+1) % n; count++;
    signal(notempty);
} until 0;
```

Consumer:

```
do {
    if (count==0) {wait(notempty);}
    nextc = buffer[out];
    out = (out+1) % n; count--;
    signal(notfull);
    ...
    consume(nextc);
    ...
} until 0;
```

1 mark for declaring appropriate semaphores

1 mark for count variable

2 marks for correct increment/decrement of count

2 marks for correct wait operation and placement on notfull,notempty

2 marks for correct signal operation and placement on notfull,notempty

[8 marks]

3) Consider the following information about an operating system and the allocation of resources within it.

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				

a) Explain the meaning of the terms *Allocation*, *Max*, and *Available*, as they are used in the Banker's Algorithm.

Answer:

Allocation	<i>The number of resources currently allocated to a process</i>
Max	<i>The maximum number of resources that this process will ever request</i>
Available	<i>The number of resources currently available, i.e., not allocated to any process</i>

2 marks each

[6 marks]

b) Show that the system is in an unsafe state.

Answer: *This question was wrong: it should have said "show that the system is in an unsafe state if P_1 is granted a request for 0 6 2 0". Since it didn't, students were marked leniently, and 2 marks were given to anyone who attempted to use the Banker's algorithm to show unsafeness. Statements then that the system was safe were awarded full marks, as were hypotheses that granting requests in non-safe order would lead to unsafeness.*

[4 marks]

c) Show that it is nevertheless possible for the processes to complete execution without deadlock occurring.

Answer: *Since part b was wrong, leniency was given here also. Full marks to those who pointed out that allowing processes to terminate and return their resources would always lead to non-deadlock, since all requests can be granted if each process executes in isolation, or if P_2 to P_4 execute to completion first.*

[4 marks]

- 4) Suppose a disk drive has 5000 cylinders, numbered 0 to 4999. The disk is currently serving a request at cylinder number 143. The queue of pending requests, in FIFO order, is
86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance in cylinders that the disk arm moves in order to satisfy all pending requests, assuming the

- a) FCFS
- b) SSTF

disk-scheduling algorithms?

Answer: FCFS

<i>Current</i>	<i>New</i>	<i>Distance</i>
143	86	57
86	1470	1384
1470	913	557
913	1774	861
1774	948	826
948	1509	561
1509	1022	487
1022	1750	728
1750	130	1620
total		7081

Check that "current" is in FCFS order

[2 marks]

SSTF

<i>Current</i>	<i>New</i>	<i>Distance</i>
143	130	13
130	86	44
86	913	827
913	948	35
948	1022	74
1022	1470	448
1470	1509	39
1509	1750	241
1750	1774	24
total		1745

Check that "New" is monotonic after first

[2 marks]

[4 marks]