300698-sample-answers

Operating Systems Programming (Western Sydney University)

WESTERN SYDNEY
UNIVERSITY

SAMPLE EXAM – AUTUMN/1H SESSION 2016

School of Computing, Engineering and Mathematics

Complete your details in this section when instructed by the Exam Supervisor at the start of the exam. *You should also complete your details on any answer booklets provided.*	
STUDENT SURNAME:	
STUDENT FIRST NAME:	
STUDENT ID:	

EXAM INSTRUCTIONS

Read all the information below and follow any instructions carefully before proceeding.
This exam is printed on both sides of the paper – ensure you answer all the questions.
You may begin writing when instructed by the Exam Supervisor at the start of the exam.
Clearly indicate which question you are answering on any Examination Answer Booklets used.

UNIT NAME:	Operating Systems Programming/Operating Systems Programming (Advanced)		
UNIT NUMBER:	300698/300943		
NUMBER OF QUESTIONS:	Five (5)		
VALUE OF QUESTIONS:	The values of the questions are as follows: Question 1: 10 marks. Question 2: 10 marks. Question 3: 10 marks. Question 4: 10 marks. Question 5: 10 marks. The total for all questions in the exam paper is 50 marks.		
ANSWERING QUESTIONS:	Answer all questions in the space provided. If more space is needed please use the back of the exam pages, and label your answer: *Continued on page X*.		
UNIT COORDINATOR:	Dr Evan Crawford		
TIME ALLOWED:	2 hours	TOTAL NUMBER OF PAGES:	Twelve (12)

RESOURCES ALLOWED

Only the resources listed below are allowed in this exam.

All printed materials, books, handwritten or printed notes are allowed.

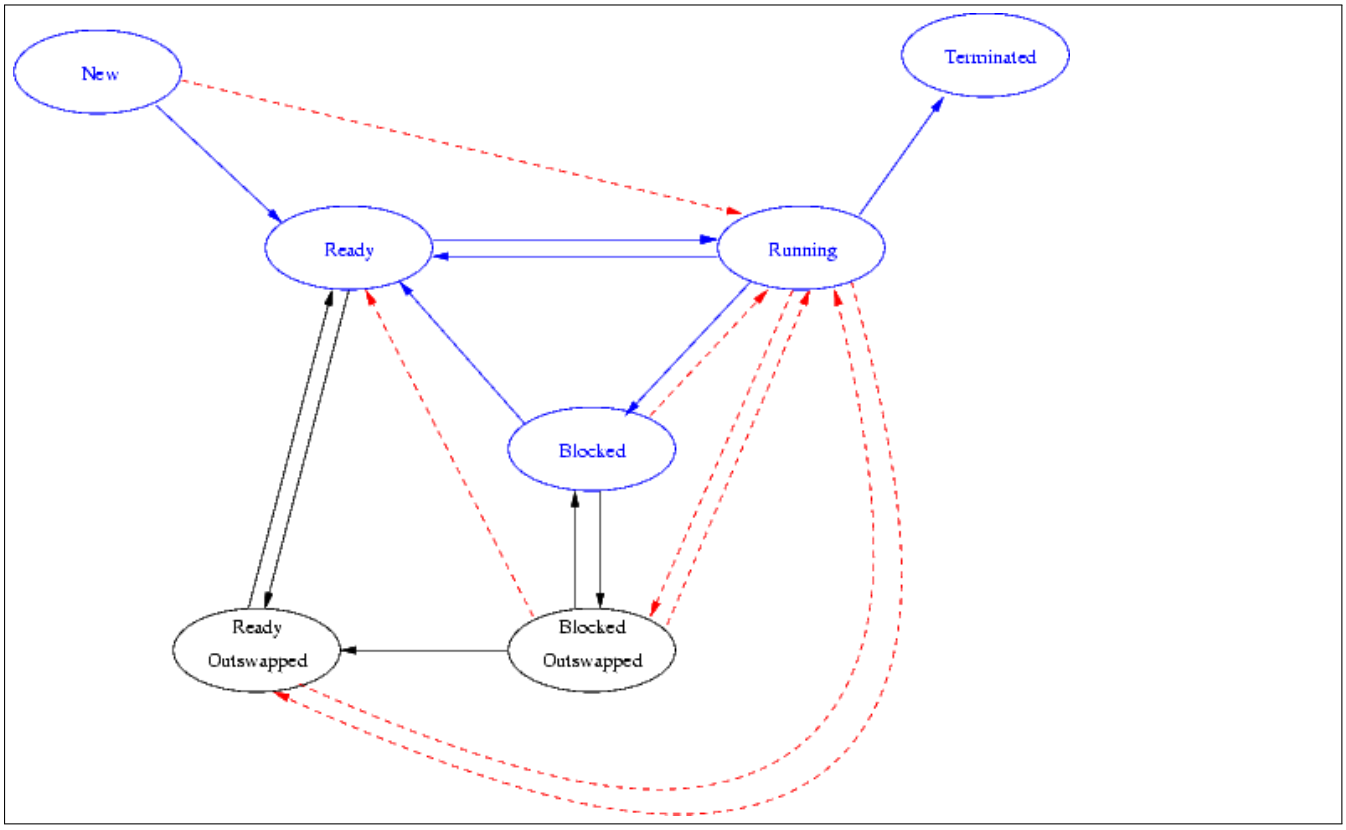
DO NOT TAKE THIS PAPER FROM THE EXAM ROOM

This page is intentionally left blank.

Question 1 **(5+(1+1+1+1+1) = 10 marks)**

Consider a 7-state process state transition diagram, which includes 2 new states: *ready outswapped* and *blocked outswapped* in addition to 5 basic states: *new*, *ready*, *blocked*, *running* and *terminated*.The new states represent processes which were swapped out of memory to secondary storage.

a) Draw a diagram showing all seven states, and all state transitions a process may go through in its lifetime.



b) For the following transitions, state if the transition is possible, and if it is then describe a sample event that can trigger it.

(i) New → Ready
Yes, new process

(ii) Blocked → Running
Yesy, timeout

(iii) Running → Ready
Yes, Dispatch

(iv) New → Blocked Outswapped
No

(v) Ready → Ready Outswapped
Yes, Memory pressure

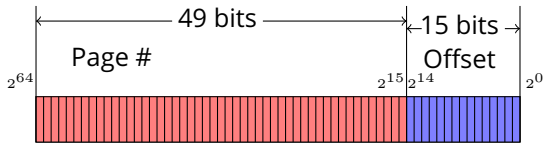
This page is intentionally left blank.

Question 2 **(2+2+2+2+2 = 10 marks)**

Consider a virtual memory system with the following parameters:

- 64-bit virtual address
- 32-kbyte virtual page size
- 64-bit Page Table Entry (PTE)
- 2-Gbyte physical memory

(a) Draw a diagram of the 64-bit address showing the length and placement of all bit fields i.e. the offset in the page(O), and the page number (P).



Virtual Address

(b) How many virtual pages are available in this system?

There are $2^{49} = 562949953421312$ virtual pages.
Note a power two is a sufficient answer to all parts of this question.

(c) How many physical frames are available in this system?

2Gb $= 2^{31}$ bytes. Number of frames is memory size divided by page size $= \frac{2^{31}}{2^{15}} = 2^{31-15} = 2^{16} = 65536$

(d) What is the size, in bytes, of the page table?

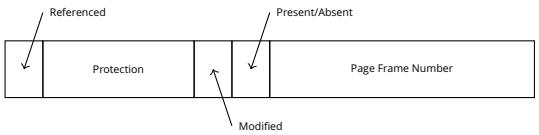
Size is the number of entries times the size of the entry
$= 2^{49} \times 64\text{bits} = 2^{49} \times 8\text{bytes} = 2^{49} \times 2^{3}\text{bytes} = 2^{49+3}\text{bytes} = 2^{52}\text{bytes}$

(e) Name TWO fields of the Page Table Entry.

From Lecture 8, Slide 22:
Each page table entry contains:

- page frame number
- valid bit, set if page is in memory
- other flags: modified (dirty), referenced, protection bits (read, write, execute, for user or kernel mode)

An example page table entry:



This page is intentionally left blank.

Question 3 **(1+1+8 = 10 marks)**

The model of protection in a computer system can be viewed as an access control matrix, with rows representing objects (for example files), and columns representing domains (for example users). Each entry in the matrix represents the access rights of the domain to the object. Consider the access control matrix below:

	Anna	Bill	Charles	Damian
Afile	rx	r	r	
Bfile	x	rx	x	
Cfile		rx	rw	rx
Dfile	r	r	r	rw

a) If the system implements access control lists, what access rights are associated with file Bfile in the table above?

Bfile: Anna (X), Bill (RX), Charles (X)

b) If the system implements capabilities, what capabilities are granted to user Anna?

Anna; Afile (RX), Bfile (X), Dfile (R)

c) Assume that each user (domain) is the owner of the file (object) with the name starting with their initial, i.e. Anna owns Afile, Bill owns Bfile etc. Represent the access control matrix in the table above using protection scheme available in UNIX. You will need to define groups of users.

File	Owner	Group	Permissions
Afile	**Anna**	**Bill, Charles**	**540 (r-xr—–)**
Bfile	**Bill**	**Anna, Charles**	**510 (r-x–x—)**
Cfile	**Charles**	**Bill, Damian**	**650 (rw-r-x—)**
Dfile	**Damian**	**Anna, Bill, Charles**	**640 (rw-r—–)**

This page is intentionally left blank.

Question 4 **(5+5 = 10 marks)**

a) Name FIVE UNIX I/O system calls.

```
open()
close()
read()
write()
lseek()
creat()
stat()
unlink()
fsync();
```

Any five of these.

b) Describe, including a prototype, each of the I/O system calls from part a).

- `int open(const char *path, int flags [, mode_t mode]);`
- `int close(int fd);`
- `ssize_t read(int fd, void* buf, size_t count);`
- `ssize_t write(int fd, const void* buf, size_t count);`
- `off_t lseek(int fd, off_t offset, int where);`
- `int creat(const char *path, mode_t mode);`
- `int stat(const char *restrict path, struct stat *restrict buf);`
- `int unlink(const char *path);`
- `int fsync(int fildes);`

And a short description, including parameter meanings e.g.:
`int fsync(int fildes);`
`fsync()` causes all modified data and attributes of `fildes` to be moved to a permanent storage device.
Returns 0 if successful -1 otherwise.

This page is intentionally left blank.

Question 5 **(2+2+2+4 = 10 marks)**

Briefly answer the following questions in the space provided.

a) What does the `fork()` system call do?
The `fork()` system call creates an almost exact duplicate of the calling process. It appears to return twice (if successful, if not it returns -1), returning 0 to the new process, and the PID (a number greater than 0) to the original process.

b) What does the `exec()` family of system calls do?
The `exec()` family of system calls load a new executable into the current process. This is a destructive operation, and if `exec()` succeeds then the calling process is unaware of it as the new program starts, otherwise an error has occurred and control returns to the original process.

c) What is a limitation of a `pipe`?
A `pipe` is only useful for communication between two closely related processes, parent/child or siblings due to the way they are created.

OR

A pipe is a one-way communication channel, two are needed for full-duplex communication.

d) Explain the difference between *buffered* and *unbuffered* message passing. What can *unbuffered* message passing achieve?
With buffered message passing, messages are placed into a mailbox for later retrieval.

With unbuffered message passing, messages are copied straight from the sender to the receiver.

As unbuffered message passing requires the sender or receiver to wait until the message is ready to be copied, it is able to achieve a rendezvous.

END OF EXAM PAPER

This page is intentionally left blank.