Family Name _____

First Name _____

Student Number |__|__|__|__|__|__|__|__|

Venue _____

Seat Number _____

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

No electronic/communication devices are permitted.

No exam materials may be removed from the exam room.

Computer Science and Software Engineering

EXAMINATION

End-of-year Examinations, 2017

ENCE360-17S2 (C) Operating Systems

Examination Duration: 120 minutes

Exam Conditions:

- Restricted Book exam: Approved materials only.
- No calculators are permitted

Materials Permitted in the Exam Venue:

- Restricted Book exam materials.
- Students may bring in one A4 sheet of paper. Both sides may be used but it must be HAND WRITTEN. Not typed or photocopied notes.

Materials to be Supplied to Students (if needed)**:**

- Extra sheets of write-on question paper (or answer book)

Instructions to Students:

- **Write your name and student ID above**
- This exam is worth a total of 100 marks
- Contribution to final grade: 50%
- Length: 13 questions
- Answer all questions.
- Check carefully the number of marks allocated to each question. This suggests the degree of detail required in each answer and therefore amount of time to spend on it.
- The amount of space provided also indicates the amount of detail expected.
- **Write strictly in the spaces allocated to each answer**. Do not write close to the margins, as the answer books will be scanned, and writing very close to the margin may not be picked up. If you require extra room, there is a blank page at the end of this booklet. You may also use additional sheets of paper; these must be fastened securely to your answer booklet. You should clearly indicate in the appropriate space that the answer is continued/provided elsewhere.

For Examiner Use Only

Question	Mark

Total _____

Questions Start on Page 3

Question 1 [6 marks]

Arrange the below layers in order, top (user) to bottom (hardware).

A. Device driver (e.g., mouse)
B. Computer game (e.g., FIFA 2017)
C. Shell (e.g., Bash)
D. Physical devices (e.g., Hard disk)
E. Operating System (e.g., Linux)
F. Program control (e.g., Task Manager)

Question 2 [6 marks]

Given an operating system with process states of READY, RUNNING, and BLOCKED.
Consider a new state called ON-DECK which has only one process at a time, the process that will be scheduled next.
Assuming the OS uses a Round-Robin scheduling algorithm, draw a state diagram showing all transitions between states. Be sure to clearly label all transitions and states.

Question 3 [8 marks total]

Consider the following processes with their CPU burst times:

Process	Burst
A	5
B	1
C	3

a) Assume Shortest Job First (SJF) scheduling. **[2 marks]**
Draw a chart illustrating the scheduling order and compute the average waiting time.

b) Assume Round Robin scheduling with a time quantum of 1. **[2 marks]**
Draw a chart illustrating the scheduling order and compute the average turn-around (completion) time.

c) Assume each context switch takes 0.1 units. What is the throughput in part (a)? In part (b)? **[2 marks]**
(Do not count the context switch times when the first process starts and the last process ends.)

d) Briefly describe the effect on throughput from reducing the quantum size in part (b). **[2 marks]**
Briefly describe the effect on response time from reducing the quantum size in part (b).

Question 4 [6 marks total]

```
#include <stdio.h>

int main() {
       fork();
       puts("hello\n");
       puts("goodbye\n");
     }
```

Assume all system calls succeed.
Also assume that puts() is atomic (cannot be interrupted)

(a) If the above code is compiled and run, what are all possibilities for output? **[2 marks]**

(b) Still using fork(), rewrite the code to have only one "hello" printed followed by one "goodbye" (or vice-versa). **[4 marks]**

Question 5 [6 marks total]

Consider a process `a.out` compiled from the approximate Unix code:

```
#include <stdio.h>

int main(int argc, char *argv[]) {

     int *p_num;
     int incr;
     int label;

     incr = atoi(argv[1]); /*convert 1st arg string to number*/

     label = 360;
     p_num = (int *)shm_create(label,sizeof(int));/*shared memory*/

     *p_num = *p_num + incr;

     printf("%d\n", *p_num);
}
```

Assume that `shm_create()` creates or attaches to a shared memory segment using the indicated label. Assume that when the shared memory segment is created, it is initialized to "`0`". Assume system calls never fail. Also assume that `sem_create()`, `shm_create()` and `printf()` are atomic (they cannot be interrupted).

(a) If two process run the above code simultaneously, one with the command "`a.out 1`" and the other with the command "`a.out -1`", what are all possible outputs? **[3 marks]**

(b) To guarantee that only two "`0`"s are printed, add system calls to the above code using only the below semaphore system calls. You may declare as many semaphores as needed (changing the variable name from sem as appropriate). Note: Do not write out all the code above. **[3 marks]**

```
sem = sem_create(KEY, i); /* create semaphore, initialized to i */
sem_wait(sem);
sem_signal(sem);
```

Question 6 [6 marks]

Given the following code fragments:

```
int process_id      /* process id */
int thread_id       /* thread id */
int registers[4] /* to hold the general purpose registers */
char *program_counter /* instruction currently executing */
char *pStack         /* pointer to the stack */
char *pHeap          /* pointer to the heap */
char *pCode          /* pointer to the code segment */
int state            /* one of READY,RUNNING,WAITING*/
```

Place the fragments inside the below structures such that the `ProcessControlBlock` contains *exactly* enough information to hold the context of a process and the `ThreadControlBlock` contains *exactly* enough information to hold the context of a thread.
From the thread you should be able to access the process.
(You may use the fragments more than once, if required.)

```
struct ProcessControlBlock {
```

```
}
```

```
struct ThreadControlBlock {
```

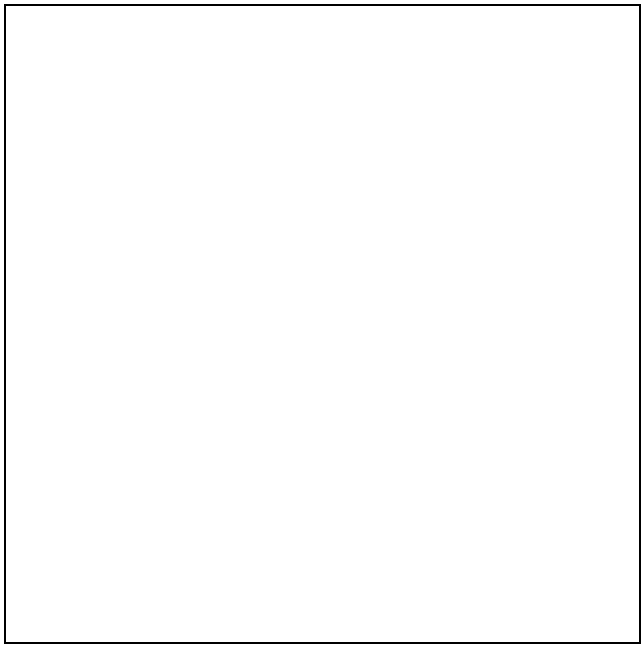
```
}
```

Question 7 [6 marks]

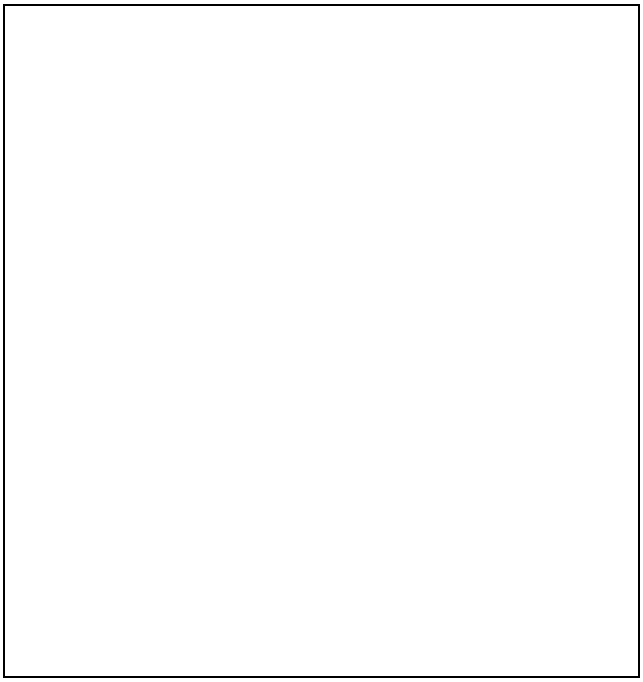
Assuming you need a TCP client-server system that can handle more than one client simultaneously. Arrange the below system calls in order of use under client/server. You do not need to add any other code. Note - system calls may be used more than once.

a `send()`
b `connect()`
c `bind()`
d `fork()`
e `close()`
f `recv()`
g `accept()`
h `listen()`
i `socket()`

CLIENT



SERVER



Question 8 [6 marks total]

Consider disk scheduling algorithms for sending requests to a disk for blocks (cylinders) for a disk that is busy (i.e., there are many requests).

(a) Why is a Scan (elevator) algorithm better than a Shortest Seek Time First algorithm? **[3 marks]**

(b) Now, assume the disk is not busy. Why is a Shortest Seek Time First algorithm better than the Scan algorithm? **[3 marks]**

Question 9 [14 marks total] *Caches*

The following is a fragment of a 16KByte, 2-way associative L1 cache with a line size of 16 bytes, an update policy of write- deferred, write-allocate, and a replacement policy of least-recently-used (LRU).

Line	Tag	Valid?	Dirty?	Tag	Valid?	Dirty?
47	101	Y	N	111	N	N
46	010	Y	N	110	Y	Y
45	011	Y	Y	000	N	N
44	111	Y	Y	101	Y	N
43	010	Y	N	111	Y	N

For each of the following memory operations (in binary) for a 64KByte virtual memory space, indicate the number of memory transfers assuming that the bus width is 4 bytes, and show the state of the "valid" and "dirty" bits of the corresponding cache entry after the operation.

Show your working. **[4 marks]**

(a) READ: 0100001011011010 **[2 marks]**

(b) READ: 1100001011100111 **[2 marks]**

(c) WRITE: 1010001011110100 **[2 marks]**

(d) READ: 1100001011110000 **[2 marks]**

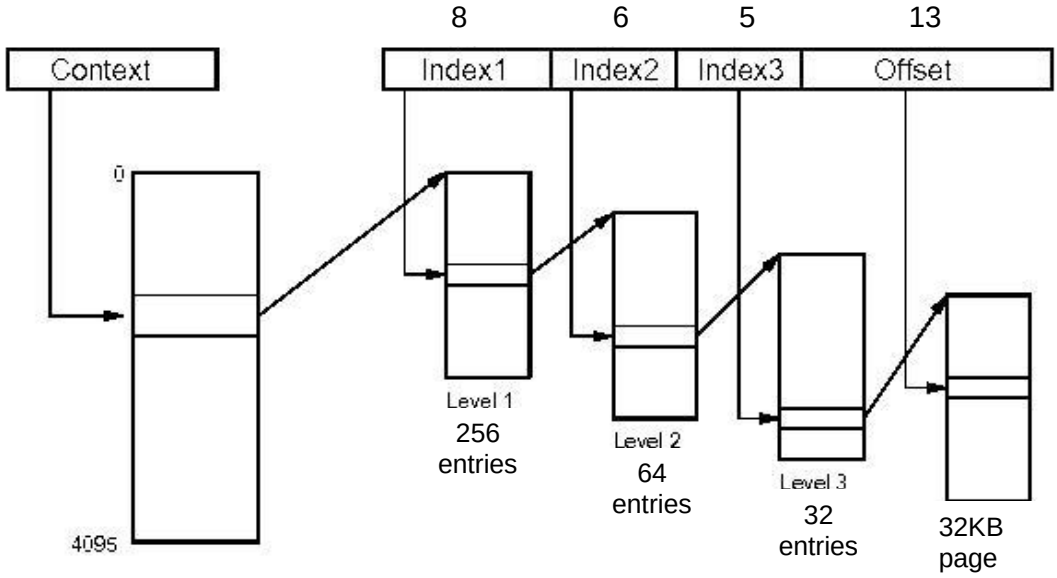
(e) WRITE: 0000001011111111 **[2 marks]**

Question 9 contd.

Question 9 contd.

Question 10 [12 marks] *Virtual memory*

A 32-bit architecture uses a 3-level page table as illustrated below.



Calculate how many bytes are required for the page table of a process on a system with a 4MB text segment, a 140MB data segment, and a 70MB stack.

Assume that:

- the text segment starts at 0x0,
- the data segment follows the text segment,
- the stack grows down from 0XFFFFFFFF.

Level 1 nodes are 8KB each, level 2 are 1KB each and level 3 nodes are 1KB each.

Hint 32 x 32KB = 1MB

Show your working.

Question 10 contd.

Question 11 [6 marks] *Virtualisation*

The first, second and third generations of virtualization were “full virtualisation”, “paravirtualization” and “hardware assisted virtualization”. Briefly describe one pro and one con for each of these three generations of virtualization.

Question 12 [12 marks total] *Caches lab*

You are implementing a very large simulation which can be easily broken down into many parallel tasks (e.g. an N-body physical gravity simulation).

(a) When implementing loop blocking for matrix-matrix multiplication, explain how you would you optimize memory access on an unknown CPU with unknown cache size(s). **[6 marks]**

(b) Explain in which priority order you would apply the following optimisations and why. **[6 marks]**

A. Use multi-threading or SIMD to compute many operations in parallel
B. Re-write the code in assembler to use the minimum number of instructions
C. Use loop blocking and optimise memory accesses

Question 13 [6 marks total] *Distributed Processing Lab*

In lab 6 you analysed the performance of a method to compute the length (l2 norm) of a vector using MPI_Scatter and MPI_Gather.

(a) How many local/remote processes would you use to run this method to get the best performance on two 4-core PCs? **[2 marks]**

(b) Name the two important factors that would change this and explain why. **[4 marks]**

. . . extra space . . .
If you use this page, please refer to it from the original question.

End of Examination