UC
UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

University of Canterbury

End-of-year Examinations 2016

Prescription Number: ENCE 360

Paper Title: Operating Systems

Time Allowed: 2 hours

Number of Pages: 7

- This exam is worth a total of 100 marks.
- Contribution to final grade: 50%.
- Length: 12 questions.
- Calculators are not allowed.
- Use the separate *Answer Booklet* for answering *all* questions.
- *This is a closed book test.*
  But you can have 2 sides of an A4 page of *handwritten* notes.
- Please answer *all* questions carefully and to the point. Check carefully the number of marks allocated to each question. The number of marks suggests the degree of detail required in each answer and the amount of time you should spend on the question.

Question 1 [6 marks total]

If any of the following is not an operating system, then explain why it is not an operating system.

- Linux
- Windows
- Android
- ROS
- macOS
- DOS
- iOS
- Arduino

Question 2 [6 marks]

Using the code below in your discussion, describe the differences between creating a thread and creating a process and differences between running a thread and running a process.

```
void main (void)
{
         pid_t childId = fork();
        if (childId == 0)
                printf("I am Tweedledee\n");
        else    printf("I am Tweedledum\n");
}
```

```
void* myFunction (void* arg)  {  printf("I am Tweedledum\n");   }

void main (void)
{
        pthread_t  childId;
        pthread_create (&childId, NULL, myFunction, NULL);
        printf("I am Tweedledee\n");
}
```

Question 3 [6 marks total]

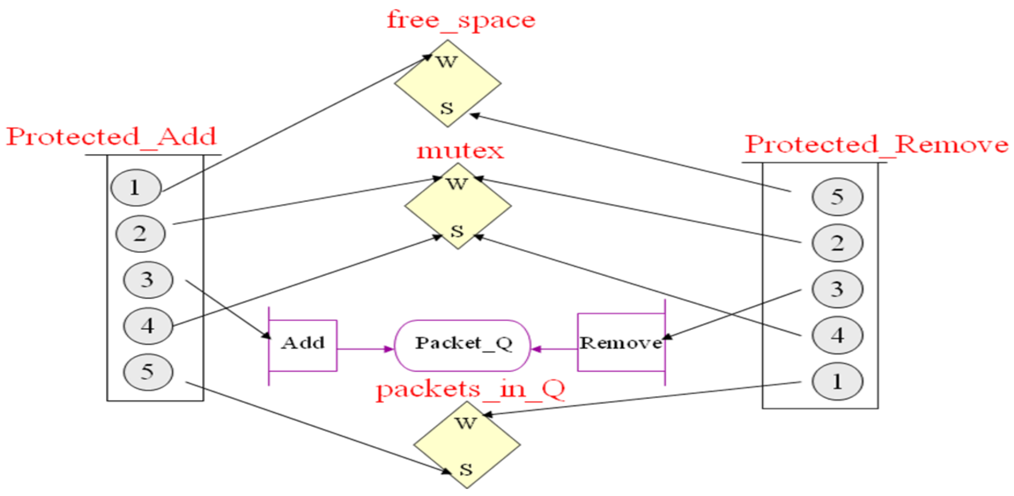
In 1971, Edsger Dijkstra set an examination question on a synchronization problem where five computers competed for access to five shared tape drive peripherals. Soon afterwards the problem was retold by Tony Hoare as the “dining philosophers problem”.

(a) Name and briefly describe the two common multi-thread synchronization computing problems in concurrency that the dining philosophers problem is illustrating. [2 marks]

(b) Describe how these two problems are solved in terms of semaphores and mutexes. [4 marks]

Question 4 [8 marks]

In terms of <u>*semaphores*</u>, <u>*mutexes*</u>, <u>*threads*</u> and <u>*processes*</u>, **explain the operation of a pipe using the following diagram** (which represents a pipe). (Ensure your explanation encompasses *semaphores*, *mutexes*, *threads* and *processes*.)



Question 5 [8 marks]

Describe differences (API and internal) between unnamed pipes and named pipes using the code below.

unnamed pipe:

```
// create and open an unnamed pipe:
int pid[2];
pipe(pid);

write(pid[1], buffer, strlen(buffer));
read(pid[0], buffer, BUFSIZE);
```

named pipe:

```
// create and open a named pipe:
int pid0, pid1;
mknod("./named_pipe_filename", S_IFIFO | 0666, 0);
pid1 = open("./named_pipe_filename", O_WRONLY);
pid0 = open("./named_pipe_filename", O_RDONLY);

write(pid1, buffer, strlen(buffer));
read(pid0, buffer, BUFSIZE);
```

Question 6 [6 marks total]

For the following code fragment:

(a) describe the purpose of the two *dup2()* functions [4 marks]

(b) describe the purpose of missing *close()* function(s) [2 marks]

```
        pipe(fd);
        if (( childpid=fork())==0) { // in child:
                dup2(fd[1],STDOUT_FILENO);
                execl("/bin/ls","ls","-l",NULL);
        } else { // in parent:
                dup2(fd[0], STDIN_FILENO);
                execl("/bin/sort", "sort","-n","+4",NULL);
        }
```

Question 7 [8 marks total]

The Banker's algorithm is a resource allocation & deadlock avoidance algorithm developed by Edsger Dijkstra.

(a) [2 marks] What are some practical problems with this algorithm?

(b) [6 marks] Given the following processes (P1,P2,P3) and resources (A,B,C,D), use the banker's algorithm to enable all processes to run to completion without deadlock. Show your working.

```
    Currently available resources:
     A B C D
Free 0 1 3 2

    Currently allocated resources:
     A B C D
P1   1 2 2 1
P2   1 0 3 3
P3   1 1 2 0

    Resources needed by each process to complete:
     A B C D
P1   3 3 2 2
P2   1 2 3 4
P3   1 1 5 0
```

Question 8 [6 marks]

Explain why desktop computers and notebooks now typically use serial interfaces for external devices, rather than parallel interfaces with multiple data lines (multiple wires).

Question 9 [14 marks total]

(a) In terms of processes and pipes and execl, describe how you would implement the following shell command? Draw a diagram. [2 Marks]

ls | tail -n 3 || awk '{print tolower($0)}')

(b) Why is perror called without checking if there is an error in the following code? [2 Marks]

```
execl("/bin/ls", "ls", "-l", NULL);
perror("The exec of ls failed");
```

(c) Explain what the point of the process 'rank' in an MPI program is? [2 Marks]

(d) What is the point in transposing one matrix first, when implementing matrix multiplication? [2 Marks]

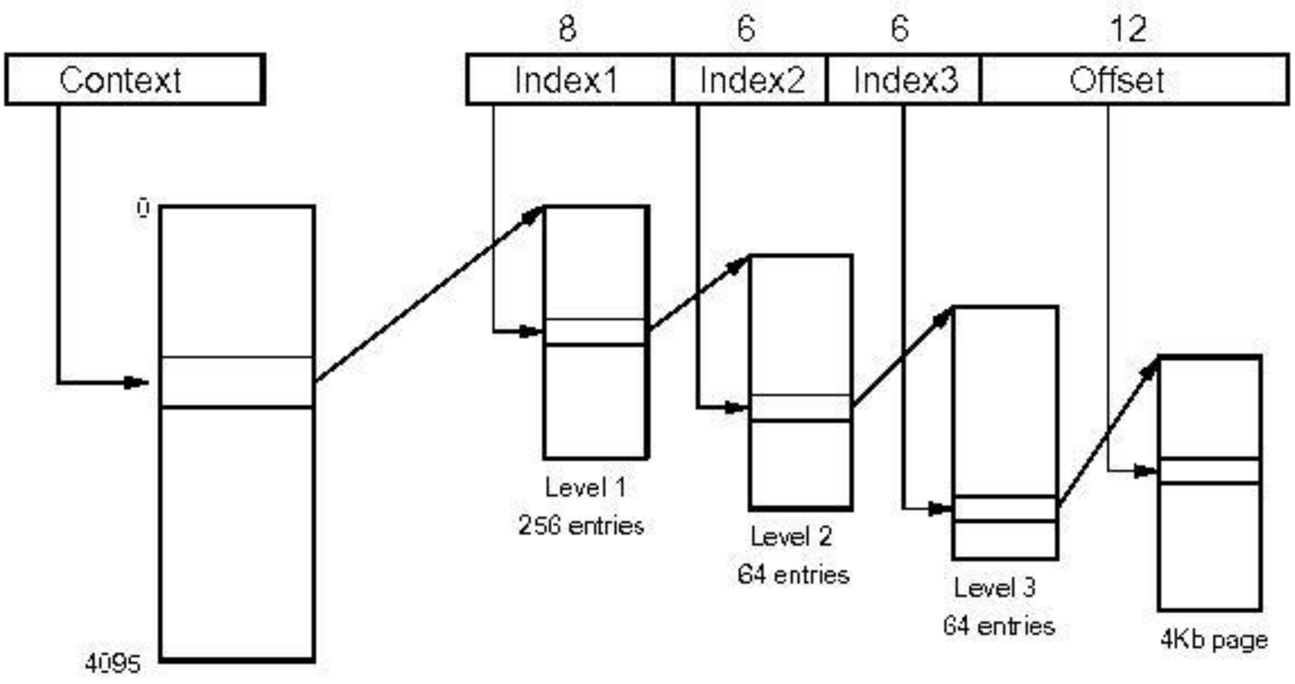
(e) When reading a file one character at a time, which would you use? fread or read, and why? [2 Marks]

(f) in getaddrinfo, the parameter 'service' is a string - why would you chose to provide "http" instead of "80" (the port number for http)? [2 Marks]

(g) Why would you include MAP_SHARED in the flags passed to the mmap function? [2 Marks]

Question 10 [12 marks]

The 32-bit SPARC 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 SPARC system with a 5MByte text segment, a 20MByte data segment, and a 20MByte stack. Assume that the text segment starts at 0x0, that the data segment follows the text segment, and that the stack grows down from 0XFFFFFFFF. Level 1 nodes are 1024 bytes each, level 2 and 3 nodes are 256 bytes each. Show your working.

Question 11 [8 marks total]

Describe the four levels of cache (and approximate sizes) associated with a multi-core processor and used by a modern operating system.

Question 12 [12 marks total]

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-back, write-allocate, and a replacement policy of least-recently-used (LRU).

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

Consider the following five memory operations (in binary) for a 64KByte virtual memory space.

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

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

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

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

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

For each:

- indicate the number of memory transfers assuming that the bus width is 4 bytes,
- show the state of the “valid” and “dirty” bits of the corresponding cache entry after the operation.

An additional two marks will be given for correctly calculating the number of bits in the tag, set number and offset. **[2 marks]**

Show all your working.

END OF PAPER