13.01.20

09.30 - 11.30am

CMPU 2017 Operating Systems 2

Nat Stad, Irish Athl Boxing Cent

Programme Code: DT228, DT282
Module Code: CMPU 2017
CRN: 24240, 26463

TECHNOLOGICAL UNIVERSITY DUBLIN

KEVIN STREET CAMPUS

BSc. (Honours) Degree in Computer Science

BSc. (Honours) Degree in Computer Science (International)

Year 2

SEMESTER 1 EXAMINATIONS 2019/20

Operating Systems 2

Mr. Denis Manley
Dr. Deirdre Lillis
Mr. Patrick Clarke

2 hrs

Answer Questions 1 and any two others

Question 1 is worth 40 marks, all the rest are worth 30.

1

a) Given the following arrival times and CPU time for 4 processes determine the average turnaround time for:

a round robin schedule algorithm with a time slice of 6 *ms*
the shortest remaining time.

(8 marks)

Arrival Time	0	1	2	3
Job	A	B	C	D
CUP cycle time	8	4	9	5

b) Describe the 4 requirement for deadlock to occur **(4 marks)**

c) Explain, using a simple example, the purpose of the linux fork() and how it achieves its purpose. **(8 marks)**

d) Explain, using suitable examples, the steps required to add a node to an ordered linked list.

To an empty list or at the start of the list
In the middle or end of the list

(10 marks)

e) Clearly explain if the following sample code will delete a node from the head of a queue. **(10 marks)**

```
char function 1(QueueNode** headPtr, QueueNode** tailPtr)
{
  if (*headPtr != NULL){
  char value = (*headPtr)->data;
  QueueNode* tempPtr = headPtr;
  headPtr = (*headPtr)->nextPtr;
  if (*headPtr == NULL) {
    tailPtr = NULL;
  }
  free(tempPtr);
  return value;
}
Else
    – printf("the list is empty....");
}
```

2

a) Distinguish between single and multi-threading processes. **(4 marks)**

b) In C a thread is created using the following code:

int pthread_create(*pthread_t *tidp, pthread_attr_t *attr, *start_rtn, void * arg*)

Clearly explain what each of the arguments in the thread create function mean.
(8 marks)

c) Explain, in your own words, the following code: **(10 marks)**

```
#include<pthread.h>
#include <stdio.h>
int value;
void *my_thread(void *param);

int main (int argc, char *argv[])
{
pthread_t tid; int retcode;

//what does this do?
if (argc != 4) {
   fprintf (stderr, "usage: a.out <integer value>\n");
   exit(0);
}

//what does this do?
retcode = pthread_create(&tid,NULL,my_thread,argv[2]);

//what does this do
if (retcode != 0) {
            fprintf (stderr, "Unable to create thread\n");
            exit (1);
    }

// What does this do?
    pthread_join(tid,NULL);
    printf ("I am the parent: thread: the value returned by my child thread = %d\n",
value);
```

```
} //main

// What does this do?
void *my_thread(void *param)
{
    int i = atoi (param);
    printf ("I am the child, I am passed value %d\n", i);
    value =  i*i*i*i;
    pthread_exit(0);
}
```

d) What would be the output of the executable code of the above program, explaining your answer, where the following is input at the command prompt: **(4 marks)**

../a.out 3, 2
./a.out 5, 4, 7

e) What would be the two outcomes in the above program if the *pthread_join* command was removed and the command line input was *./a.out 6, 3, 2,* ? Explain the reason for your answer. **(4 marks)**

3

a) Explain, using an example, why it is critical to ensure that concurrency is carefully controlled for processes accessing the same data item; in other words the *race* problem. **(6marks)**

b) Two ways to prevent the race problem are Test and Set and Wait and Signal. Distinguish between each approach. **(4 marks)**

c) Explain, in detail, what the following two threads are doing. **(12 Marks)**

```
// what does this thread do
void * signal(void *t)
{
for (i=0; i < 3; i++) {
                                        /* what is happening here */
   pthread_mutex_lock(&count_mutex);
   count++;
if (count == 4) {      // count is a global variable
                                              /* what is happening here */
      pthread_cond_signal(&Condition_Value);

   }

printf("inc_count(): thread %ld, count = %d, unlocking mutex\n",
          my_id, count);
                                          /* what is happening here */
pthread_mutex_unlock(&count_mutex);
sleep(1);
pthread_exit();
}
```

```
// what does this thread do

void *wait(void *t){

                                        /*  what is happening here */
   pthread_mutex_lock(&count_mutex);
    while (count < COUNT_LIMIT) {
                                          /* what is happening here */
      pthread_cond_wait(&Condition_Value, &count_mutex);
       count += 90;
      printf("watch_count(): thread %ld count now = %d.\n", my_id, count);
```

```
    • }
                                        /* what is happening here */
      pthread_mutex_unlock(&count_mutex);
       pthread_exit();
}
```

d) If the *main() function of this c program* has the following statements:

```
long t1=2, t2=5, t3=7;

pthread_create(&threads[0], &attr, wait, (void *)t2);
 pthread_create(&threads[1], &attr, signal, (void *)t1);
 pthread_create(&threads[2], &attr, signal, (void *)t3);

/
 for (i = 0; i < 3; i++) {
  pthread_join(threads[i], NULL);
 }
```

give a sample output of this multithread program and explain your reasoning.

(8 marks)

4:

a) Explain, using an example, how to convert the logical address of a process to its physical address. **(6 marks)**

b) Describe the purpose of each of the following fields in the page map table of a virtual; memory management system:

status field,
the reference field. **(4 marks)**

c) Page swapping is an essential element of virtual memory: The *First In First Out (FIFO)* is one commonly used swapping algorithm. Using a demand page memory management system how many page faults will be generate by the following sequence and page frame numbers? Clearly show how you arrived at the answer

Reference Sequence = [A,B, C, B, D, A, E, F, B, E, D, F**].**

Using *3 page frames*
Using *4 page frames*.

(12 marks)

d) The Least Recently Used (LRU) is another page swapping algorithm: Explain how this could be implemented. **(8 Marks)**