National University of Computer and Emerging Sciences

Islamabad Campus

0118

Operating Systems (CS2006)

Course Instructor(s):
Ms.Sidra Khalid, Ms.Maryam Shahbaz
Section(s): (CS-A,B,C,D)

Sessional-II Exam

Total Time (Hrs): 1
Total Marks: 50
Total Questions: 4

Date: Nov 5, 2024

[redacted] C [redacted]

Roll No **Course Section** **Student Signature**

Do not write below this line.

Attempt all the questions.

Complete all the questions on the provided answer sheet. Extra bonus mark(s) will be awarded if all the questions are attempted on the answer sheet and in order.

[CLO 1: Understand, design and implement solutions employing concepts of Processes and Threads.]
[CLO 2: Compare and contrast the commonly used for scheduling of tasks in operating systems and analyze different synchronization issues and implement synchronization mechanisms like Semaphores, Monitors, etc.]

Question 1: **[8 Marks]**

Write the key Term'(Name) related to each concept given below from the glossary.

i. No. of processes that complete their execution per time unit.
ii. Possible to get the CPU back from a process, even if the process has not completed its execution.
iii. Type of scheduler that determines which programs are admitted to the system.
iv. A thread I/O Operation can block the process.
v. Threads of a process can be scheduled on multiple processors.
vi. Several processes access and manipulate the same data concurrently and Outcome of the execution depends on the particular order in which the access takes place.
vii. Each process has a code segment in which the shared data is accessed.
viii. If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the process that will enter the critical section next cannot be postponed indefinitely.

Glossary:
Kernel Level Thread, ~~User Level Thread~~, Bounded Wait, Mutual Exclusion, ~~Progress~~, ~~Critical Section~~, ~~Race condition~~, Medium term schedular, Short term schedular, ~~Long term schedular~~, Response time, ~~Throughput~~, Non-Preemptive scheduling, and *~~Preemptive Scheduling~~.*

[CLO 1: Understand, design and implement solutions employing concepts of Processes and Threads.]

Question 2: **[12 Marks]**

a. Given the following C program, what is the exact output? If multiple outputs are possible, describe any one possibilities. Assume that all system calls and pthread library calls complete successfully. Further, assume that the main thread ID is 1984, with any child threads numbered 100, 101, 102, etc. [3 Marks]

Q1

i) Throughput

ii) Preemtive Scheduling

iii) Long Term Scheduler

iv) User Level Thread

v) Kernel Level Thread

vi) Race Condition

vii) Critical Section

viii) Progress

Q2

a)

```
MAIN: Creating thread for task 1
MAIN: Creating thread for task 4
THREAD 100: I'm alive 1
MAIN: Creating thread for task 7
THREAD 101: I'm alive 4
MAIN: Creating thread for task 10
THREAD 102: I'm alive 7
THREAD 103: I'm alive 10
MAIN: Joined child thread
MAIN: Joined child thread
MAIN: Joined child thread
MAIN: Joined child thread
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void * action( void * arg )
{
   int t = *(int *)arg;
   printf( "THREAD %u: I'm alive %d\n",(unsigned int)pthread_self(), t
);
   sleep( t );
   return NULL;
}

int main()
{
  pthread_t tid[4];
  int i, t;
  for ( i = 0 ; i < 4 ; i++ )
  {
      t = 1 + i * 3;
      printf( "MAIN: Creating thread for task %d\n", t );
      pthread_create( &tid[i], NULL, action, &t );
   }
   for ( i = 0 ; i < 4 ; i++ )
   {
       pthread_join( tid[i], NULL );
       printf( "MAIN: Joined child thread %u\n", (unsigned int)tid[i]
);
   }
return 0;
}
```

b. What is the behavior of the given code if the second for loop (i.e., the loop that includes thepthread_join() call) is entirely removed? Write anyone expected output. [3 Marks]

c. What is the range of x at the end of this code. [2 Marks]

```
int x = 0;
int i, j;
void AddToX()
{
     for (i = 0; i < 100; i++)
           x++;
}
/*Thread B*/
void SubFromX()
{
   for (j = 0; j < 100; j++)
          x--;
}
```

d. Consider this code and answer below questions(i-iv) [4 Marks]

```
int balance = 0;
void *mythread(void *arg)
{
      int result = 0;
      result = result + 200;
      balance = balance + 200;
      printf("Result is %d\n", result);
      printf("Balance is %d\n", balance);
      return NULL;
}
int main(int argc, char *argv[])
{
      pthread_t p1, p2;
      pthread_create(&p1, NULL, mythread, "A");
      pthread_create(&p2, NULL, mythread, "B");
      pthread_join(p1, NULL);
      pthread_join(p2, NULL);
      printf("Final Balance is %d\n", balance);
}
```

i. How many total threads are part of this process? [1 Mark]

ii. When thread p1 prints "Result is %d\n", what value of result will be printed? [1 Mark]

iii. When thread p1 prints "Balance is %d\n", what value of balance will be printed? [1 Mark]

iv. When "Final Balance is %d\n" is printed, what value of balance will be printed? [1 Mark]

b) Threads will not be able to join

MAIN: Creating thread for task 1
MAIN: Creating thread for task 4
~~MAIN~~ THREAD 100: I'm alive 1~~00~~
MAIN: Creating thread for task 7
THREAD: 101 : ~~I'm alive 4 Creating thread for~~
MAIN: Creating thread for task 10
THREAD: 102: I'm alive 7
THREAD: 103 : I'm alive 10

c) 0 to 100 (1)

d)
i) 2 ✗
ii) 200 ~
iii) 200 ✓ (2)
iv) 400 ✗

[CLO 2: Compare and contrast the commonly used for scheduling of tasks in operating systems and analyze different synchronization issues and implement synchronization mechanisms like Semaphores, Monitors, etc.]

Q3: Solve the below parts a,b and c related to synchronization problems. [10 Marks]

a. Consider the following solution to the critical section problem involving only two processes. [5 Marks]

```
//shared variable
   1. boolean flag[2] // Shared. Initially false
   1.1 _____________
   2. flag[i] = true;
   2.1_____________________

  //Declare intent to access *
   3. while (flag[1-i] && _____________);
   5. Critical Section
   6. flag[i] = false;
```

flag[2]
flag[0] = true
while(flag[1]);
flag[1] = true
(flag[0])
CS()

i- Which requirement for shared access to a critical section is being violated by the above algorithm. [2 marks]

ii- Write the missing code for line 1.1,2.1 and line 3 to solve this problem. [3 marks]

b: Consider this variant of bakery Algorithm that does not allow a process to enter into critical section if it has same token already taken by another process. **[3 Marks]**

Algorithm

```
1  var choosing: shared array [0..n-1] of boolean;
2      number: shared array [0..n-1] of integer;
       ...
3  repeat
4       choosing[i] := true;
5       number[i] := max(number[0],number[1],...,number[n-1]) + 1;
6       choosing[i] := false;
7       for j := 0 to n-1 do begin
8          while choosing[j]  do nothing;
9          while number[j] <> 0 and
10                                  (number[j]) < (number[i])
11              do nothing;
12      end;
13      (* critical section *)
14      number[i] := 0;
15      (* remainder section *)
16      until false;
```

i- Which line number is causing this problem? [1 Mark]

ii- Fix the code to solve this problem. [2 Marks]

C- Add the missing code in below parts **[2 Marks]**

i) Add the missing line in TSL so that it can work as expected. [1 Mark]

```
boolean TestAndSet (boolean *target){
            _____________________
          *target = TRUE;
           return rv:
}
```

ii) Use the swap/exchange instruction to avoid critical section. [1 mark]

```
// Lock is a shared variable and set to False
// key is local variable for each pi
  key:=1
  while(key)
      _____________________________
   Critical_section();
   Lock =0;
   Remainder_section();
```

[CLO 2: Compare and contrast the commonly used for scheduling of tasks in operating systems and analyze different synchronization issues and implement synchronization mechanisms like Semaphores, Monitors, etc.]

Q4: Consider the following set of processes, with associated processing times and priorities mentioned in the table below. **[20 Marks]**

Q3

a)

i) Progress is being violated. Deadlock can occur if 2 threads run side by side, both will busy wait causing deadlock. (2)

ii)

1.1 : int turn; (3)

2.1 : turn = 1 - i ;

3 : (flag[1-i] && turn == 1-i);

b)

i) Line 10

ii) (number[j], j) < (number[i], i)

(5)

c) i) boolean *rv = target;

ii) Swap (key, lock);

Process Name	Processing Time	Priority
A	4	3
B	1	1
C	2	3
D	1	4
E	2	2

For each scheduling algorithm, fill in the table with the process that is running on the CPU (for time slice-based algorithms, assume a 2 unit time slice).

- A smaller priority number implies a higher priority.
- For RR and Priority, assume that an arriving process is running at the beginning of its arrival time, if the scheduling policy allows it.
- All of the processes arrive at time 0 in the order Process A, B, C, D, E.
- Assume the currently running process is not in the ready queue while it is running.

Note- Attempt this question on answer sheet by making same table on answer sheet and add values in it accordingly.

Time Slice	FIFO ***[5 Marks]***	RR ***[5 Marks]***	SRTF ***[5 Marks]***	Priority***[5 Marks]***
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
Average Turnaround Time				

-Good Luck-

Time Slice	FIFO	RR	SRTF	Priority
0	A	A	B	B
1	A	A	D	E
2	A	B	C	E
3	A	C	C	A
4	B	C	E	A
5	C	D	E	A
6	C	E	A	A
7	D	E	A	C
8	E	A	A	C
9	E	A	A	D
10				
11				
Avg. TAT (s)	6.8	6.4	4.6	6

Rough : ↘

Fifo : 4-0 + 5-0 + 7 + 8 + 10

$\frac{34}{5} = 6.8$

6.8
5 | 34
-30
40
-

4.6
5 | 23

10 + 3 + 5 + 6 + 8 = 42

10 + 6 + 4 + 2 + 1

7 + 9 + 10 + 3 + 1 → 30