


National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Operating System	Course Code:	CS-220
	Program:	BS(Computer Science)	Semester:	Fall 2020
	Duration:	1.5 hour	Total Marks:	45
	Paper Date:	25 th November, 2020	Weight:	15%
	Section:	ALL	Page(s):	4
	Exam:	Mid-2		

Instructions/Notes: Answer questions on the question paper. Attempt all questions. Programmable calculators are not allowed. Write answers clearly and precisely, if the answers are not easily readable then it will result in deduction of marks. Use extra sheet for rough work, **cutting and blotting on this sheet will result in deduction of marks.**

Name: _____ **Roll No.:** _____ **Section:** _____

Question 1 (2 points): The major concern with race conditions is (tick the correct option)

- | | |
|----------------------------------|-------------------------------------|
| (1) Inefficient use of resources | (3) Loss of data |
| (2) Integrity of data | (4) No optimal solutions available. |

- | | |
|----------------------------------|-------------------------------------|
| (1) Inefficient use of resources | (3) Loss of data |
| (2) Integrity of data | (4) No optimal solutions available. |

Question 2 (2 points): Which of the following is NOT the name of a Linux file system structure (tick the correct option)

- | | |
|-----------------------|----------------|
| (1) Dentry | (3) Inode |
| (2) Master File table | (4) Superblock |

- | | |
|------------------------------|----------------|
| (1) Dentry | (3) Inode |
| (2) Master File table | (4) Superblock |

Question 3 (2 points): The following is NOT an advantage of Semaphore (tick the correct option)

- | | |
|--------------------------|-------------------------------|
| (1) Resource management | (3) Ensuring mutual exclusion |
| (2) Sequencing execution | (4) Avoids deadlocks |

- | | |
|--------------------------|-------------------------------|
| (1) Resource management | (3) Ensuring mutual exclusion |
| (2) Sequencing execution | (4) Avoids deadlocks |

Question 4 (2 points): Linked allocation method in file system suffers from (tick the correct option)

- | | |
|--|---|
| (1) Performance and reliability issue | (3) Internal fragmentation and external fragmentation |
| (2) External fragmentation and performance issue | (4) External fragmentation and reliability issue. |

- | | |
|--|---|
| (1) Performance and reliability issue | (3) Internal fragmentation and external fragmentation |
| (2) External fragmentation and performance issue | (4) External fragmentation and reliability issue. |

Question 5 (2 points): Contiguous allocation method in file system suffers from (tick the correct option)

- | | |
|-------------------------------------|---------------------------------------|
| (1) Performance of reading the file | (3) Performance of expanding the file |
| (2) Performance of saving the file | (4) Data integrity |

- | | |
|-----------------------------------|--|
| (1) Performance of reading a file | (3) Performance of expanding a file |
| (2) Performance of saving a file | (4) Data integrity |

Question 6 (10 points): Assume that you are in a google meet question-answer session with your teacher. During this session, only one person can speak at a time, either the teacher or a student. Since there are many students, any one of the students can ask a question. The teacher accepts only one question at a time and speaks only when a question is asked. However, as soon as the question is completed, the teacher has to answer it. No other student can ask a question until the first question is answered. Remember, it is the students who will start the session by asking a question from the teacher. You are required to force this sequence of question-answer session using semaphore(s). In order to write your solution, you may use the following semaphores:

1. Semaphore *mutex_speak*: both the teacher and students will use this semaphore to speak; either to answer or ask the question respectively
2. Semaphore *question*: One of the students will use this semaphore to ask a question.
3. Semaphore *answer*: A teacher will enable himself to answer a question using this semaphore.

Your answer should contain the prototype of the code using the following syntax: “wait(speak); signal (avail_question)” etc. Furthermore, you are also required to initialize the semaphores.

Question 7 (5 points): The following numbered statements show the sequence of opening and reading a newly created file on a file system. Your task is to order the statements in the correct sequence.

1. The open() system calls searches the system-wide open-file table for the file if it is already in use.
2. The application program calls the logical file system to open a file.
3. The physical block numbers for the requested logical block numbers are to calculated.
4. The directory structure provides the pointer to the inode
5. The required physical blocks are loaded into memory.

Solution: 2, 1, 4, 3, 5.

Question 8 (10 points): Considering a file having a size of 20 KB. Using index allocation method, a block size of 256 bytes, and a pointer of 4 bytes, answer the following questions

1. How many data blocks will be required for storing this file in the secondary storage?
2. How many second level index blocks will be required to complete the mapping of this file?
3. If you have to read the 8,220th byte, which second level index block will you use?
4. If you have to read the 8,220th byte, which index of second level index block will you use?
5. What offset in the data block will be used to read the 8,220th byte?

Solution:

1. $20480/256 = 80$ data blocks
2. $80/64 = 1.25$, meaning 2 index block.

3. $8220/(256 * 64) = 0.5$, meaning 0^{th} index block
4. First $8220 \bmod (256 * 64) = 8220$. Then, $8220/256 = 32.1$, meaning 32nd index.
5. $8220 \bmod 256 = 28$.

Question 9 (10 points): You are given two vectors a and b of equal length. You need to sum the two vectors element wise and save the result into another vector c . Note that at the end of the operation the length of a , b and c will be same. Besides the main thread, you can create only two additional threads. Write the code to do the task in a most efficient way. You do not need to initialize vectors a , b and c . Assume that their values are already populated, as shown in the code below.

```

// Better solutions are possible, but following would suffice
#define LENGTH xxxx

struct arr_limits{
    int start;
    int end;
};

int* a= NULL;
int* b= NULL;
int* c= NULL;

#define handle_error(msg) do { perror(msg); exit(EXIT_FAILURE); } while (0)

void* sum_array(*void params)
{
    arr_limits* limits = (arr_limits*) params;
    for (int i = arr_limits->start; i < arr_limits->end; ++i)
        c[i] = a[i] + b[i]
    pthread_exit(NULL);
}

int main()
{
    a = populate_a();
    b = populate_b();
    c = populate_c();

    arr_limits limit_1 ;
    limits_1.start = 0;
    limits_1.end = LENGTH/2;

    arr_limits limit_2 ;
    limits_2.start = LENGTH/2 + 1;
    limits_2.end = LENGTH;

    int s;
    pthread_attr_t attr;
    s = pthread_attr_init(&attr);
    if (s != 0)
        handle_error_en(s, "pthread_attr_init");
    pthread_t tid1 = 0;
    pthread_t tid2 = 0;

    s = pthread_create(&tid1, &attr, sum_array, &limits_1);
    if (s != 0)
        handle_error_en(s, "pthread_create");

    s = pthread_create(&tid2, &attr, sum_array, &limits_2);
    if (s != 0)
        handle_error_en(s, "pthread_create");

    s = pthread_join(tid1, NULL);
    if (s != 0)
        handle_error_en(s, "pthread_join");

    s = pthread_join(tid2, NULL);
    if (s != 0)
        handle_error_en(s, "pthread_join");
}

```