INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
End-Spring Semester Examination 2022-23

Subject No.: CS30002 / CS31202 **Subject: OPERATING SYSTEMS**

Duration: 3 hours **Full Marks: 100**

Department/Center/School: Computer Science and Engineering

Specific charts, graph paper, log book etc., required: NA

Special Instructions (if any):

- **Attempt all questions. All parts of the same question should be answered together.**
- **Keep your answers brief and to-the-point.**
- **No clarifications can be provided during the exam. If necessary, make reasonable assumptions, and state any assumption made.**

Question 1 **[3 + 3 = 6]**

(a) How many times will "N" be printed by the following code segment?

```
if ( fork( ) && fork( ) || fork( ) )
    fork( );
printf ( "N" );
```

(b) Are the following statements True or False? Correct the statement if False (No marks will be awarded if the correction is incorrect)

(i) Threads of the same process share both global variables and heap.

(ii) In the following code snippet, the two printed values might be same or different.

```
int a;  if (fork( ) == 0) printf("%d", &a);  else printf("%d", &a);
```

(iii) The standard C functions malloc( ), exit( ) and sleep( ) always invoke system calls

Question 2 **[(3 + 2 + 2) x 2 = 14]**

A database program reads data from a file (r), performs some complex calculations on the data (c) and then writes the outputs and transaction logs back to the same file (w). It is estimated that 30% of the time is taken for reading data, 40% for the calculations and 30% for writing back.

In a **uniprocessor** system, 4 instances of the program (P1, P2, P3, P4) are run concurrently on different data sections of the same file. It is estimated that *in isolation,* P1, P2, P3 and P4 takes 10, 10, 20 and 30 *msecs* respectively to run. During the read and write phases of any process, the *CPU is freed and can be allocated another process. Also assume that the memory controller takes charge of the r/w operations*, which allows multiple simultaneous data transfer operations between the memory and the secondary storage. In other words, the system hardware allows multiple processes to read/write data from a file simultaneously.

To prevent data corruption, the database program uses appropriate semaphores and mutex locks to satisfy the ***readers-writers*** problem – multiple read operations may occur simultaneously from the file, but the write operations are mutually exclusive and are always given higher priority than read operations. Also assume that context switching between ready processes is much faster than switching from I/O to ready phase.
Draw the (i) **Gantt Chart,** and calculate the (ii) **turnaround time** and the (iii) **waiting time** for ***each of the 4 instances*** of the program, assuming the scheduling algorithm used is (a) *Shortest Remaining Time First (preemptive)*, and (b) *Round Robin scheduling with time quantum of 3 msec.*

Question 3 **[1 + 2 + 2 = 5]**

You need to implement mutex functionality using a *pipe* through the functions shown below. Fill in the definitions of pipe_mutex_init, pipe_mutex_lock and pipe_mutex_unlock. Each blank below should consist of *only one* C statement. You should be able to achieve the same functionality as the pthread versions of lock and unlock. The mutex should be unlocked by default, i.e., when initialized. You may assume that this pipe_mutex will be used only in a multi-processing environment, not a multi-threading one. You may further assume that the processes will not be interrupted by kernel signals arising from unexpected errors.
Hint: For this to be a true mutex, you need to ensure that statements (ii) and (iii) are atomic. Also you need to ensure that the process waiting to acquire the mutex should be blocked, not busy waiting.

```
typedef struct pipe_mutex { int fd[2]; char buf;        } pipe_mutex;

void pipe_mutex_init(pipe_mutex* m) {
        pipe(m->fd);
        _____(i)_____
        return 0;
}
void pipe_mutex_lock(pipe_mutex* m) {_____(ii)_____}
void pipe_mutex_unlock(pipe_mutex* m) {_____(iii)_____}
```

Question 4 **[14 x 0.5 = 7]**

In a system, three *worker processes* are responsible for carrying out three tasks A, B and C. Apart from this, a fourth *book-keeping process* is used to perform a book-keeping task D. **The processes must be synchronized to execute the tasks in the following manner: A → D → B → D → C → D → A, to be repeated cyclically.** Three *binary* semaphores X, Y and Z are used to synchronize between the worker processes. Additionally, two more binary semaphores M and N are used to synchronize between the worker processes and the book-keeping process. Achieve the desired synchronization by filling in the blanks given below with either **P(S)** or **V(S)** operations, where S = {X, Y, Z, M, N}. Assume that the system starts with the following default values for every semaphore: X = 1, Y = 0, Z = 0, M = 1, N = 0. Each blank must have only one P( ) or V( ) operation.

while (1) {
_____
_____
// Task A
_____
_____
}
	while (1) {
	_____
	_____
	// Task B
	_____
	_____
	}
		while (1) {
		_____
		_____
		// Task C
		_____
		_____
		}
			while (1) {
			_____
			// Task D
			_____
			}

Question 5 **[6 + 2 = 8]**

Assume that in a system, there are 5 processes P0, P1, P2, P3, P4, and 4 types of resources A (10 instances), B (13 instances), C (9 instances), D (12 instances).

(a) Given the allocation and max requirement matrices shown below, is the system in a safe state? Assume that at every point, the processes are considered in the cyclic order P0 → P1 → … → P4 → P0 → … Show the working at every step based on the *Banker's Algorithm*, especially the initial **need matrix** and the changing content of the **work vector**. The working should follow the order of execution of the algorithm.

(b) If a request from P1 now arrives for (1, 1, 0, 1) [vector represents the requirement for each of the resources A, B, C and D respectively], can it be granted immediately (in other words, will the resultant state of the system after granting this request be safe)?

	Allocation Matrix				Maximum Requirement Matrix			
	A	B	C	D	A	B	C	D
P0	0	0	1	2	3	3	1	2
P1	3	1	2	1	4	2	5	2
P2	2	1	0	3	3	4	1	6
P3	1	3	1	2	2	3	2	4
P4	1	4	3	2	3	5	6	5

Question 6 **[12 x 2 = 24]**

Answer the following questions:

(a) A linker specifies an entry point address during loading an executable---the executable data is to be loaded in that address. This process is called relocation. However, we learned in class that in modern OS with virtual memory every executable will have addresses starting from zero and at run-time the addresses will be translated into physical addresses using page tables. Justify if relocation is needed in modern OS or is it redundant.

(b) Design a workload (a reference string for a memory of 5 physical frames) where LRU algorithm will give you the worst-case performance. In other words, all accesses will result in page faults.

(c) In memory we learned about base and limit registers. Would there be separate base, limit registers allocated to every process? Why or why not?

(d) State True or False with justification: *page-based memory management systems do not suffer from external fragmentation.*

(e) Can there be a case where compared to a single-level page table a multi-level page table will use less memory? If yes, state such a case. If no, explain why not.

(f) A parent process creates a shared memory segment (it is never detached and deleted in parent or children) and then forks two children. Can there be a case where the child processes will not be able to access the shared memory at some point during their execution? If yes, state such a case. If no, explain why not.

(g) Recall that in a process stack and heap share the same address space. Can you support this mechanism using only base and limit registers? Why or why not?

(h) State True or False with justification: *In a Unix-like File system it's better to store file names into directories instead of inodes.*

(i) State True or False with justification: *It is easier to delete a large file compared to a set of smaller files which has the same size in total.*

(j) State True or False with justification: *In linux, directories also have inodes.*

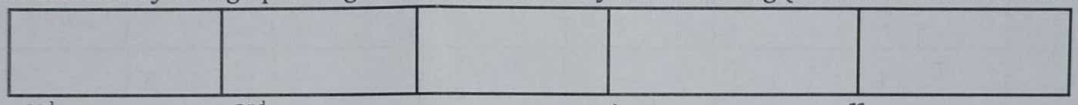
(k) A memory mapped file is defined as follows (from wikipedia): "A memory-mapped file is a segment of virtual memory that has been assigned a direct byte-for-byte correlation with some portion of a file or file-like resource." State a possible use case of memory mapped file during process creation.

(l) In XFS (an advanced file system for linux) the extents allocated to a file are stored in a B+ tree structure. Justify this design decision.

Question 7 **[(2 + 2 + 2) + (3 + 2) = 11]**

7.1. Your company is tasked to write a memory management system for a x86-64 computer which has a 64-bit processor. You decided on a design which will support 48-bit virtual address space and 32-bit physical address space. The 48-bit virtual address occupies the lower bits of the 64-bit address---so the first bits are zero. Each page in your system is 4KB and you will be using a 4-level page table with each page table entry (PTE) size is 4 bytes.

(a) Describe which bits of the 48-bit virtual address are used for each part of the virtual address translation by filling up the figure below and state your reasoning (no marks without reasoning):



(b) What is the number of physical pages in the system?

(c) How many total entries will there be in the page table?

7.2. Now you are experimenting with a new type of TLB for the system (mentioned in 7.1). Instead of mapping virtual page numbers to physical page numbers your new TLB will map the 3rd outer page index to the physical address of the 2nd level outer page table. So, your TLB only caches the first level of address translation in your multi-level paging scheme.

(a) If your TLB is fully associative and contains 64 entries, what will be the minimum size of TLB in bits? Assume each TLB entry contains the translation and a valid bit.

(b) Identify a case where your TLB will perform better than a traditional TLB

Question 8 **[(5 + 4) + 3 = 12]**

8.1. The hardware of a memory management system is equipped with an L1-cache (average miss-rate: 0.3, access-time: 1 clock) and L2-cache (average miss-rate: 0.15, access-time: 5 clocks) and a main-memory (access-time: 150 clocks), a TLB (average miss-rate: 0.5, access-time: 1 clock). Consider a virtual memory for this system using a two-level hierarchical page table system and a 50-bit virtual address space. Any page table in this system has the same size (irrespective of the level), with each PTE of size 4 bytes and are stored in memory. The page size in this system is the same as that of the size of the page table. Page faults (average rate: 0.001, service time till process can use page: 50000 clocks) are serviced in the usual interrupt-based manner.

(a) Calculate the average time for retrieving data when the CPU has only the virtual address at hand? Take the complete flow into account. Note that pages of page tables can also page fault.

(b) What is the page size used in this system?

8.2. We studied the least recently used (LRU) algorithm in class. However, although LRU is useful, it's practically impossible to keep track of arbitrary numbers of references in the past. So, LRU-k algorithm is developed, which only keeps track of last k memory references at any given point of time. Prove or disprove that LRU-k is a stack algorithm for k > number of frames.

Question 9 **[4 + 3 = 7]**

Imagine that you want to design a file system over a SSD for a smart washing machine your client is selling. The SSD has a block size of 4 KB, and the latencies to read and write a single block are 150 micro sec, and 200 micro sec, respectively (1 micro sec = 10-6 sec). However, there is one optimization--latencies to read and write contiguous 4 blocks are 200 micro sec and 500 microsecond respectively due to hardware support.
In your file system built over this SSD, you decided to use a hybrid file system comprising both index and extents. In your file system, each file has an inode with an inode structure: 10 direct pointers, a single indirect pointer and a doubly indirect pointer. Each direct pointer points to an extent of 1 block and the indirect pointers finally point to extents of size 4 blocks. Each pointer is 4 bytes. Now Answer the following:

(a) There is an application readdata.exe which requests your filesystem to read random data chunks of size 256 bytes (e.g., for benchmarking) from the storage. What is the rate of reading data for readdata.exe (Ignore block transfer time)
(b) Modify your file system (e.g., inode structure, access algorithm) to improve this rate of randomly reading data.

Question 10 **[2 + 2 + 2 = 6]**

You are working on a legacy OS with a legacy extent-based file system which supports variable extent size 1, 2 and 4 blocks. The disk has 512-byte block size and 4 byte block pointers. First 512 MB in the disk stores the inodes (in an inode table) and non-extendable. Each file has an inode and its file number is the index in the inode table. Each inode contains 16 bits of additional information other than the extent address. Directories are also considered files and the root directory is hardcoded into the first inode (in the 0th block of the disk). Aside from the inode table the disk also contains a free bitmap for finding free extents. Answer the following:

(a) What is the maximum size of disk supported by this file system?
(b) What is the maximum number of files possible in this file system?
(c) At a maximum how many blocks are required to store the free bitmap