



JK Lakshmipat University Jaipur

BTech - IV Semester Computer Science and Engineering Operating Systems CS1108 Final Exam

Time: 2 hours

23 Apr 2024

Max Marks: 35

Instructions

- There are 8 questions, each with multiple parts, printed on both sides of the question paper.
- In case of any doubt, make your assumption, write it clearly and continue.
- For some of the fill in the blank questions options are given after the blank in brackets. You have to choose one of those given options.
- Write to the point responses to what is asked in the question. Irrelevant answers may lead to loss of marks.

1)

- a) Write three main functions of an operating system in a computer system.
- b) What are system calls and why are they needed?

3 + 2 = 5 marks [CO1,2]

2)

- a) What is the primary disadvantage in storing the data of a file as a linked list of blocks on a disk?
- b) Explain how this disadvantage is mitigated by the FAT system.
- c) Suppose a disk has total 15 data blocks numbered 0 to 14 (this is very unrealistic but we are just considering a toy example). Suppose the disk has only two files file1 and file2: file1 occupies the list of blocks 3 → 10 → 4 → 7, and file2 occupies the list of blocks 2 → 12 → 8. Draw the FAT table for this case.

1 + 1 + 2 = 4 marks [CO5]

3)

- a) Explain the difference between input/output using the programmed I/O and the DMA systems.
- b) Mention a situation where polled I/O might be better than interrupt based I/O, and explain why.
- c) A keyboard is considered a ———(block/character) device in Linux systems.
- d) Give one example of a block device on a Linux system.

$1 + 1 + 0.5 + 0.5 = 3$ marks [CO1,5]

4)

- a) In connection with semaphores, what do the `sem_wait` and `sem_post` function do? Clearly and precisely tell what happens when a thread calls `sem_wait(&m)` and `sem_post(&m)` for a semaphore `m`?
- b) How can we implement a lock using a semaphore. Show code that properly initialises a semaphore and locks the critical section using this semaphore.

$2 + 2 = 4$ marks [CO3]

5) In this question you can leave your answers as powers of 2 (e.g. whether you write 2^3 or 8, both will be accepted)

- a) Consider a virtual memory system with 16-bit addresses. What is the total size of this address space?
- b) Suppose we use paging on this address space with a page size of 64 bytes. How many virtual pages are there in this system?
- c) How many bits in a virtual address correspond to the page number and how many bits correspond to the offset?
- d) Suppose the physical memory is 512kb. How many physical frames are in the system? How many bits are needed to specify a physical frame number?
- e) Suppose that we save 3 bits of extra information along with the frame number in each page table entry (PTE). What is the size of each PTE?
- f) What is the total size of the page table?
- g) How many pages does one page table needs?
- h) If we use two level paging in this then show the three components, with their sizes, in which the 16 bit address splits up.

$0.5 \times 8 = 4$ marks [CO4]

6)

- a) What is TLB in a paged virtual memory system? Explain why it is needed and what are its relevant properties that enable it to achieve its purpose.
- b) The ordinary base-bound scheme for Virtual Memory suffers from ———(external/internal) fragmentation.
- c) In a caching system, if you know the full sequence of items that will be accessed, what cache replacement policy theoretically gives the lowest number of cache misses?

$2 + 0.5 + 0.5 = 3$ marks [CO4]

7) In the Completely Fair Scheduler (CFS) of Linux:

- a) the virtual runtime evolves slower for ——— (higher/lower) priority process,
- b) at any scheduling decision the process with the ——— (lowest/highest) runtime is scheduled
- c) A process with niceness value of 10 has ——— (higher/lower) priority than a process with niceness value of 8.
- d) What data structure does the CFS use to store the processes that it has to schedule?

0.5 × 4 = 2 marks [CO1,3]

8)

a) Explain usage of (i) fork, (ii) wait and (iii) exec. Write clearly what these are used for and what happens when a process calls them.

b) Consider the code below:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main(){
    int a=10;
    int pid=getpid();
    printf("Hello from PID %d\n", pid);
    int r=fork();
    if(r==0){
        pid=getpid();
        a=20;
        printf("In PID:%d Address of a is %p, and value is %d\n",pid,&a,a);
    }
    else{
        wait(NULL);
        printf("In PID:%d Address of a is %p, and value is %d\n",pid,&a,a);
    }
    printf("Bye from PID %d\n",pid);
}
```

The PID of parent is 125 and of child is 126.

The output of this program is shown below with a few portions blanked out. Fill in the blanked portions.

```
Hello from PID ----
In PID:---- Address of a is 0x7ffddf822e4c, and value is --
Bye from PID ----
In PID:---- Address of a is ----, and value is --
Bye from PID ----
```

2 + 1 + 2 + 5 = 10 marks [CO1,2]

