INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE
End Term Examination - Term II (2023-24)

EG301 Operating Systems

Time: 2 Hours
Max. Marks: 60

Note:
1. There are 4 questions and they carry equal marks. Answer all questions.
2. Let your answers be to the point. If you make any assumptions, make it explicit and make sure it is consistent with what is taught/discussed in the class.

1. [15 marks=2+2+3+2+2+2+2]
   a. Assume an operating system supports no address translation at all. What would be the challenges in writing/compiling and running multiple programs at the same time on such a system?
   b. What is an MMU(Memory management unit), where is it located on a modern computer system and what role does it play in memory management?
   c. Consider a virtual memory system with pure segmented memory management with the 4 segments – Stack Segment, Data Segment, Code Segment and Heap (or Extra) Segment (SS,DS,CS,HS).
      i. What information does the MMU maintain to enable address translation and support protection?
      ii. Describe with illustration how a virtual address to physical address translation takes place and when the protection (against any kind of address related violation) is enforced.
      iii. How much of this translation happens in hardware and how much in software?
   d. Illustrate and show how external fragmentation happens in pure segmented memory management.
   e. Does Paging solve the external fragmentation problem? If so how?
   f. A process P has a virtual address space with some M pages. What is demand paging and how does it impact the number of physical memory frames the process occupies?
   g. What is copy on write? How is it related to the fork() system call? What is the advantage of copy on write feature?

2. Regarding paging: [15 marks=3+2+3+2+2+3]
   a. Assume a physical and logical address size of 48 bits and a page and frame size of 4KB(2^{12}bytes).
      i. In paging how is the virtual address 0x7f00 split up?
      ii. Which value is used to access the PTE?
      iii. Illustrate how this virtual address (va) converted to a physical address(pa)?
   b. Assuming the above, how many entries will a linear page table have (as a power of two)? What is a challenge with this linear page table scheme?
   c. Draw how a hierarchical Page Table is organized in x86-64. In the picture hint at how the indexing is done. Explain why this saves space compared to a linear page table.
   d. How is the address 0x7f00 looked up in this hierarchical page table structure?
   e. What is a TLB? What is its purpose and where is it located in the processor? Why do we need a TLB?
   f. What is a page fault? Can a valid page have a page fault? Describe how a page fault is handled including when a swap in or swap out is needed. You may simply use a flowchart for your description if you prefer.

3. This is a question about threads, mutual exclusion and synchronization. [15 marks=2+2+3+(2+2+4)]
   a. Mention two ways in which threads (created using pthread_create()) are different from the usual notion of processes created using fork().
   b. In class we saw that there can be a race if two threads execute a statement such as balance=balance+100; Explain in detail how this is possible assuming there is only one CPU.

c. Explain the operation of the two standard semaphore operations: sem_wait() and sem_post() by sketching pseudo code to show what they do.

d. There is a box with 20 slots in it. Each slot can hold an item. There are a large number of processes (called putters) which put an item into a free slot in the box(using put_item() function) and some process (called getters) which remove an item from a filled slot in the box(using get_item() function). The simplistic putter and getter code are as below:

```
void putter() {                          void  getter() {
       put_item();                              get_item():
}                                        }
```

We have two additional conditions:
COND1: All invocations to these *_item() functions must be mutually exclusive.
COND2: We don't want any call to put_item() when there are no free slots. Similarly, we don't want any call to get_item() when there are no filled slots.
Think of a solution using **three sempahores** and no other mutexes, counters or any other data structures. The three semaphores are called **M, Free** and **Filled**. One is a mutual exclusion semaphore and the other two are counting semaphores.

   i. What are these semaphores as related to COND1 and COND2 used for and what are their initial values?
   For the next two parts assume these semaphores are declare globally and initialized.
   ii. What semaphore operations are done to ensure COND1? Show by modifying the given code using the semaphores for this. (exact syntax of semaphore calls not important)
   iii. What semaphore operations are done to additionally ensure COND2? Show the code using your semaphore(s) for this by modifying the code you wrote in the previous part.

4. This is regarding IPC with sockets. [15 marks= 2+2+2+2+2+2+3]

a. What is the difference between a sequential server and a concurrent server in the context of TCP servers – both written to handle multiple connections?

b. Sketch the loop (no earlier initialization code needed) of a sequential TCP server. Assume a function server_func() implements the server functionality with regards to its communication with the client. (Assume socket is ready to accept() when loop is entered.)

c. Sketch the code of the loop of a concurrent server. Assume a function server_func() implements the server functionality with regards to its communication with the client. (Assume socket is ready to accept() when loop is entered.)

d. We know that a socket completely defines a connection. However, we understood that in the context of concurrent servers, each server side socket connection uses the same port and possibly the same IP address on the server. Then how are the sockets different? Explain this by also saying what composes a fully connected socket. You may refer to your code in the above parts if you wish.

e. The call socket(AddressFamily, Protocol, type) is used to create a new socket. In this the last parameter is usually 0. Show how the first two are specified for
   i. a TCP socket
   ii. a UDP socket
   iii. a UNIX Domain connection oriented socket

f. For a UDP socket we specify the recipient address each time we send a message, whereas we do not specify that in the case of a TCP send. Why ?

g.
(i) Mention two ways in which communication paradigm with signals (using the kill() and signal() system calls) differs from communication using sockets.
(ii) Also mention one way the Linux OS itself uses signals giving examples of at least two signals.