University of Sargodha

BS 5th Term Exam 2018

Subject: Computer Science **Course: Operating System (CS:3621)**

Time Allowed: 2:30 Hours **Maximum Marks: 60**

Objective Part Compulsory

Q.No.1: Attempt all parts and each require answer 2 – 3 line (12*2=24)

i. Differentiate between I/O bound processes and CPU bound processes.

CPU Bound means the rate at which process progresses is limited by the speed of the CPU. A task that performs calculations on a small set of numbers, for example multiplying small matrices, is likely to be CPU bound. A program is CPU bound if it would go faster if the CPU were faster.

I/O Bound means the rate at which a process progresses is limited by the speed of the I/O subsystem. A task that processes data from disk, for example, counting the number of lines in a file is likely to be I/O bound. A program is I/O bound if it would go faster if the I/O subsystem was faster.

ii. What is meant by context switch?

A context switch is a procedure that a computer's CPU (central processing unit) follows to change from one task (or process) to another while ensuring that the tasks do not conflict. Effective context switching is critical if a computer is to provide user-friendly multitasking.

iii. Discuss the benefits and drawbacks of short time quantum.

Advantages:

- ✓ Every Thread / Process gets a chance to run.
- ✓ CPU is shared between all processes.
- ✓ Threads with the same priority are handled perfectly with Round Robin.

Disadvantages:

- × Low Priority tasks may wait for more time if the many tasks are given high priority.
- × High Priority tasks may not execute the full instruction given stipulated amount of time.

iv. State any three advantages of multiprocessor system.

The advantages of the multiprocessing system are:

- ✓ **Increased Throughput** – By increasing the number of processors, more work can be completed in a unit time.
- ✓ **Cost Saving** – Parallel system shares the memory, buses, peripherals etc. Multiprocessor system thus saves money as compared to multiple single systems. Also, if a number of programs are to operate on the same data, it is cheaper to store that data on one single disk and shared by all processors instead of using many copies of the same data.
- ✓ **Increased Reliability** – In this system, as the workload is distributed among several processors which results in increased reliability. If one processor fails then its failure may slightly slow down the speed of the system but system will work smoothly.

v. What is real time Systems?

A real-time operating system (RTOS) is any operating system (OS) intended to serve real-time applications that process data as it comes in, typically without buffer delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter increments of time. A real time system is a time bound system which has well defined fixed time constraints. Processing must be done within the defined constraints or the system will fail.

vi. Define seek time and latency time?

Seek time is the time taken for a hard disk controller to locate a specific piece of stored data. The seek time of a hard disk measures the amount of time required for the read/write heads to move between tracks over the surfaces of the platters.

Rotational latency or latency time is the delay waiting for the rotation of the disk to bring the required disk sector under the read-write head. It depends on the rotational speed of a disk (or spindle motor), measured in revolutions per minute (RPM).

vii. What are various layers of a file system?

Layers of a file system are as follows:

- ✓ Applications Programs
- ✓ Logical File System (Directory Level)
- ✓ File Organization Module
- ✓ Basic or Flat File System
- ✓ I/O Control
- ✓ Devices

viii. What are the advantages of Contiguous allocation?

In contiguous memory allocation, all the available memory space remains together in one place. It means freely available memory partitions are not scattered here and there across the whole memory space.

In the contiguous memory allocation, both the operating system and the user must reside in the main memory. The main memory is divided into two portions one portion is for the operating and other is for the user program.

In the contiguous memory allocation when any user process request for the memory a single section of the contiguous memory block is given to that process according to its need

ix. What is meant by pure demand paging?

There are cases when no pages are loaded into the memory initially, pages are only loaded when demanded by the process by generating page faults. This is called Pure Demand Paging.

In pure demand paging, even a single page is not loaded into memory initially. Hence pure demand paging causes a page fault

When starting execution of a process with no pages in memory, the operating system sets the instruction pointer to the first instruction of the process, which is on a non-memory resident page, the process immediately faults for the page. After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory. At that point, it can execute with no more faults. This schema is pure demand paging.

x. Define access time.

Access time is total time it takes a computer to request data, and then that request to be met. Access time is the time from the start of one storage device access to the time when the next access can be started. Access time consists of latency and transfer time.

xi. Explain belady's anomaly.

In computer storage, Bélády's anomaly is the phenomenon in which increasing the number of page frames results in an increase in the number of page faults for certain memory access patterns. This phenomenon is commonly experienced when using the first-in first-out (FIFO) page replacement algorithm. In FIFO, the page fault may or may not increase as the page frames increase, but in Optimal and stack-based algorithms like LRU, as the page frames increase the page fault decreases. László Bélády demonstrated this in 1969.

xii. What is the responsibility of dispatcher?

A dispatcher is a special program which comes into play after the scheduler. When the scheduler completes its job of selecting a process, it is the dispatcher which takes that process to the desired state/queue. The dispatcher is the module that gives a process control over the CPU after it has been selected by the short-term scheduler. This function involves the following:

- ✓ Switching context
- ✓ Switching to user mode
- ✓ Jumping to the proper location in the user program to restart that program

Subjective Part (3*12)

Q2. **[3+3+3+3]**

a) If the hit ratio to a TLB is 80% and it takes 15 nanoseconds to search the TLB and 15 nanoseconds to access the main memory, then what must be the effective memory access time in nanoseconds?

Ans. 195 nanoseconds is the effective memory access time.

b) If the no of pages in a 32-bit machine is 8kB then what is the size of the page table?

Ans. 8KB is the size to page table.

c) In a 64-bit machine, with 256 MB RAM, and a 4KB page size, how many entries will there be in the page table if its inverted?

Ans. 2^{16} entries will be in page table if its inverted.

d) If the total number of available frames is 50, and there are 2 processes one of 10 pages and the other of 5 pages then how much of memory would be proportionally allocated to each of these processes?

Ans. 33 and 16 frames respectively would be allocated to each of these processes.

Q3. **[6+6]**

a) A barbershop consists of a waiting room with n chairs and the barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

We use 3 semaphores. Semaphore customers counts waiting customers; semaphore barbers is the number of idle barbers (0 or 1); and mutex is used for mutual exclusion. A shared data variable customers1 also counts waiting customers. It is a copy of customers. But we need it here because we can't access the value of semaphores directly.

We also need a semaphore cutting which ensures that the barber won't cut another customer's hair before the previous customer leaves.

```
// shared data
semaphore customers = 0;
semaphore barbers = 0;
semaphore cutting = 0;
semaphore mutex = 1;
int customer1 = 0;

void barber() {
        while(true) {
                wait(customers);        //sleep when there are no waiting customers
                wait(mutex);            //mutex for accessing customers1
                customers1 = customers1 - 1;
                signal(barbers);
                signal(mutex);
                cut_hair();
        }
}

void customer() {
        wait(mutex);            //mutex for accessing customers1
        if (customers1 < n) {
                customers1 = customers1 + 1;
                signal(customers);
                signal(mutex);
                wait(barbers); //wait for available barbers
                get_haircut();
        }


                else {  //do nothing (leave) when all chairs are used.
                        signal(mutex);
                }
        }
        cut_hair(){
                waiting(cutting);
        }
        get_haircut(){
                get hair cut for some time;
                signal(cutting);
        }
```

(b) Consider the segment table:

What is the physical address for the following logical addresses?

a. 0,430
b. 1,10
c. 1,11
d. 2,500

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Answer:

a) 219+430 = 649.
b) 2300+10=2310.
c) 2300+11=2311
d) Illegal address since size of segment 2 is 100 and the offset in logical address is 500.

Q4. (a) Discuss different advantages of multi-threading. [6]

Multithreading allows the execution of multiple parts of a program at the same time. These parts are known as threads and are lightweight processes available within the process. So multithreading leads to maximum utilization of the CPU by multitasking.

The benefits of multi-threaded programming can be broken down into four major categories:

1. **Responsiveness**

Multithreading in an interactive application may allow a program to continue running even if a part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

In a non multi-threaded environment, a server listens to the port for some request and when the request comes, it processes the request and then resume listening to another request. The time taken while processing of request makes other users wait unnecessarily. Instead a better approach would be to pass the request to a worker thread and continue listening to port.

For example, a multi-threaded web browser allow user interaction in one thread while an video is being loaded in another thread. So instead of waiting for the whole web-page to load the user can continue viewing some portion of the web-page.

2. **Resource Sharing**

Processes may share resources only through techniques such as:

- ✓ Message Passing
- ✓ Shared Memory

Such techniques must be explicitly organized by programmer. However, threads share the memory and the resources of the process to which they belong by default.

The benefit of sharing code and data is that it allows an application to have several threads of activity within same address space.

3. **Economy**

Allocating memory and resources for process creation is a costly job in terms of time and space. Since, threads share memory with the process it belongs, it is more economical to create and context switch threads. Generally, much more time is consumed in creating and managing processes than in threads. In Solaris, for example, creating process is 30 times slower than creating threads and context switching is 5 times slower.

4. **Scalability**

The benefits of multi-programming greatly increase in case of multiprocessor architecture, where threads may be running parallel on multiple processors. If there is only one thread then it is not possible to divide the processes into smaller tasks that different processors can perform. Single threaded process can run only on one processor regardless of how many processors are available. Multi-threading on a multiple CPU machine increases parallelism.

(b) Let's define a new page replacement algorithm in which we replace the page that is least frequently used (LFU). Once a page is replaced and brought back again its counter will start again. Calculate the page faults for the following reference string when the total number of frames is three. In the case of tie FIFO will be used.

1,2,1,3,2,4,1,5,6,3,1,2,1,2,4,3,5,6

Q5. A) Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a) **What is the content of the matrix Need?**
b) **Is the system in a safe state?**
c) **If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately?**

Solution:

a) The values of Need for processes P0 through P4 respectively are (0, 0, 0, 0), (0, 7, 5, 0), (1,0, 0, 2), (0, 0, 2, 0), and (0, 6, 4, 2).
b) Yes. With Available being equal to (1,5, 2, 0), either process P0 or P3 could run. Once process P3 runs, it releases its resources, which allow all other existing processes to run.
c) Yes, it can. This results in the value of Available being (1, 1, 0, 0). One ordering of processes that can finish is P0, P2, P3, P1, and P4.

B) Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)?Which algorithm makes the most efficient use of memory?

a. First-fit:

1. 212K is put in 500K partition
2. 417K is put in 600K partition
3. 112K is put in 288K partition (new partition 288K = 500K - 212K)
4. 426K must wait

b. Best-fit:

1. 212K is put in 300K partition
2. 417K is put in 500K partition
3. 112K is put in 200K partition
4. 426K is put in 600K partition

c. Worst-fit:

1. 212K is put in 600K partition
2. 417K is put in 500K partition
3. 112K is put in 388K partition
4. 426K must wait

In this example, best-fit turns out to be the best.

Q6. A) What are different criteria's according to which we select a victim while aborting one by one in deadlock recovery process?

To eliminate deadlocks by aborting a process, we use one of two methods. In both methods, the system reclaims all resources allocated to the terminated processes.

- Abort all deadlocked processes. This method clearly will break the deadlock cycle, but at great expense; the deadlocked processes may have computed for a long time, and the results of these partial computations must be discarded and probably will have to be recomputed later.
- **Abort one process at a time until the deadlock cycle is eliminated**. This method incurs considerable overhead, since after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.

Aborting a process may not be easy. If the process was in the midst of updating a file, terminating it will leave that file in an incorrect state. Similarly, if the process was in the midst of printing data on a printer, the system must reset the printer to a correct state before printing the next job.

If the partial termination method is used, then we must determine which deadlocked process (or processes) should be terminated. This determination is a policy decision, similar to CPU-scheduling decisions. The question is basically an economic one; we should abort those processes whose termination will incur the minimum cost. Unfortunately, the term minimum cost is not a precise one. Many factors may affect which process is chosen, including:

- ✓ What the priority of the process is
- ✓ How long the process has computed and how much longer the process will compute before completing its designated task
- ✓ How many and what types of resources the process has used (for example, whether the resources are simple to preempt)
- ✓ How many more resources the process needs in order to complete
- ✓ How many processes will need to be terminated?
- ✓ Whether the process is interactive or batch

---

b) Consider the following set of processes, with the length of the CPU burst given in milliseconds. Draw Gantt and calculate average waiting times and average turnaround times according to SJF preemptive.

Process	Burst Time	Arrival Time
P0	10	5
P1	12	3
P2	5	2
P3	8	1
P4	6	10
P5	10	45