OS assignment

Shaik. Abdul Rahim
AP21110011308

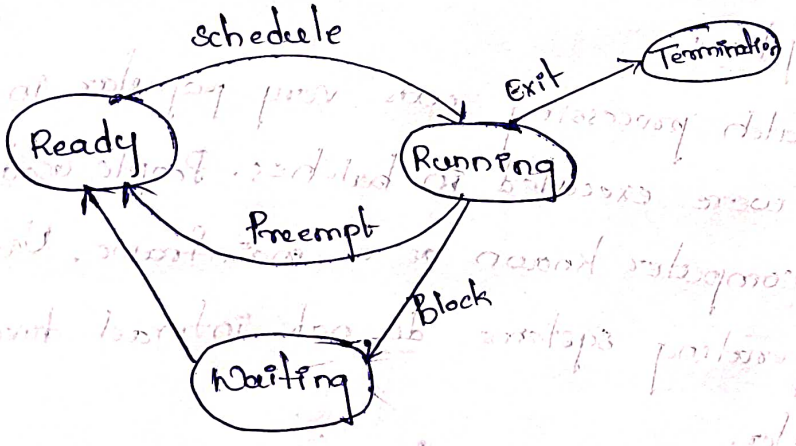
1) What is process ? What do you mean by operations on process?

A) Process:-

A process is a running program that serves as the foundation for all computation. The procedure is not the same as computer code, although it is very similar.

Operations on a process:-

The execution of a process is a complex activity. It involves various operations. Following are the operations that are performed while execution of a process:



1) Creation:- This is the initial step of process execution activity. process creation means the construction of a new process for the execution.

2) Scheduling / Dispatching: The event or activity in which the state of the process is changed from ready to running. Dispatching is done by OS when the resources are free or the the process has higher priority than the ongoing process.

3) Blocking: When a process invokes an input-output system

call that blocks the process and operating system put in block mode.

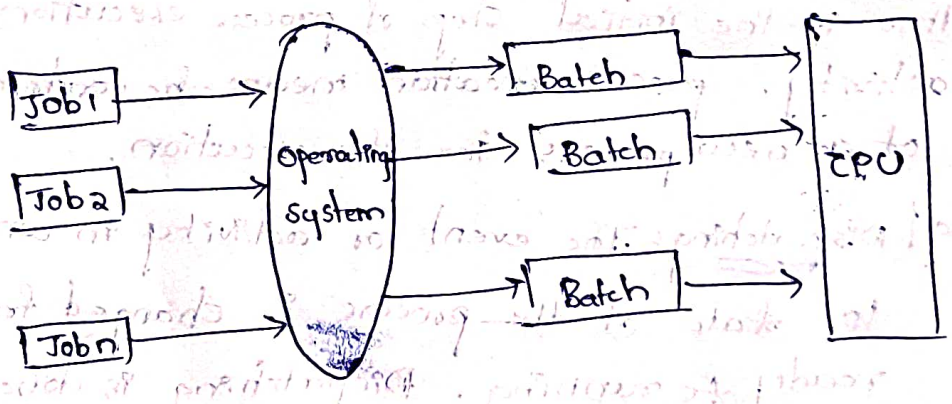
4) Preemption:- When a timeout occurs that means the process hadn't been terminated in the allotted time interval and next process is ready to execute, then the operating system preempts the process.

5) Termination:- Process termination is the activity of ending the process.

2) What is the behaviour of process in two different operating system 1. batch system, 2. time sharing system

A) 1) Batch System:-

Batch processing was very popular in the 1970s. The jobs were executed in batches. People used to have a single computer known as a mainframe. User using batch operating systems do not interact directly with the computer.



The batch operating system grouped jobs that perform similar functions. These job groups are treated as a batch and executed simultaneously.

2) Time-sharing Systems:-

Time-sharing operating systems is one of the important type of operating system.

Time-sharing enables many people, located at various terminals, to use a particular computer system at the same time. Multitasking or Time-sharing systems is a logical extension of multiprogramming.

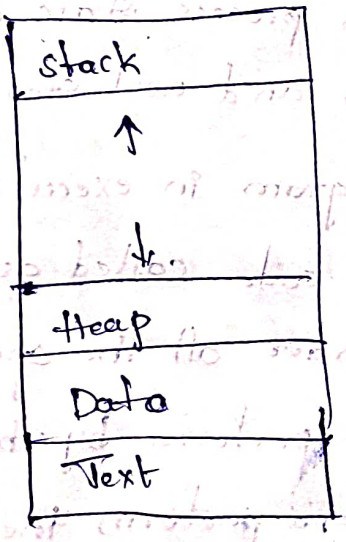
The main difference between Time-sharing systems and multiprogrammed Batch systems is that in case of multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-sharing Systems, the objective is to minimize response time.

3) What are the multiple parts of the processes?

A) A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections.

* They are:- stack, heap, text and data.



a) Stack:- The process stack contains the temporary data such as method/function parameters, return address and local variables.

b) Heap:- This is dynamically allocated memory to a process during its run time.

c) Text:- This includes the current activity represented by the value of program counter and the contents of the processor's registers.

d) Data:- This section contains the global & static variables

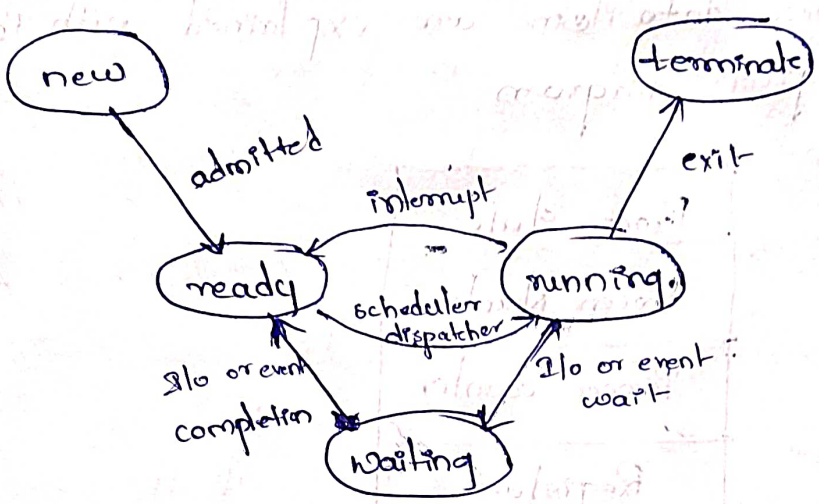
4) Program is an passive entity and process is an active entity. Justify with proper example?

A) A program is a passive entity that has a set of codes and instructions required to accomplish a task. A process, on the other hand, is an active entity of a program that is started with the execution of the program. Thus, the process is started by the program, once it is executed.

5) What do you mean process state diagram explain each of the states in process state diagram with previous state relation and next state relation?

A) A process is a program in execution and it is more than a program code called as text section and this concept works under all the operating system because all the task perform by the operating system needs a process to perform the task.

The process executes when it changes the state. The state of a process is defined by the current activity of the process.



Each process may be in any one of the following states-

⇒ New - The process is being created.

⇒ Running - In this state the instructions are being executed.

⇒ Waiting - The process is in waiting state until an event occurs like I/o operation completion or receiving a signal.

⇒ Ready - The process is waiting to be assigned to a processor.

⇒ Terminated - The process has finished execution.

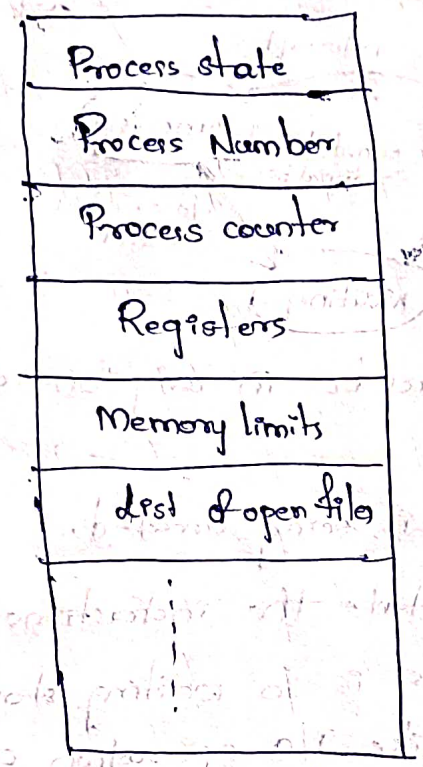
6) What do you mean process control block? explain the term program counter, CPU registers in PCB and its working.

A) Process control block:

process control block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

Structure of the process control Blocks:

The process control stores many data items that are needed for efficient process management some of these data items are explained with the help of the given diagram



Process control block (PCB)

Program counter:- This contains the address of the next instruction that needs to be executed in the process.

CPU Registers:- This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

5) What do you mean context switching? Apart from round robin algorithm is there any algorithm?

A) Context Switching:-

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

Apart from round robin algorithm the context switching is used in preemptive scheduling algorithms like Preemptive SJF.

6) What do you mean by process scheduling? how many types of scheduling queeues? explain...

A) Process scheduling:-

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a multiprogramming operating systems. There are two types of scheduling. They are-

1) Non-preemptive
2) Preemptive.

Process scheduling Queues:-

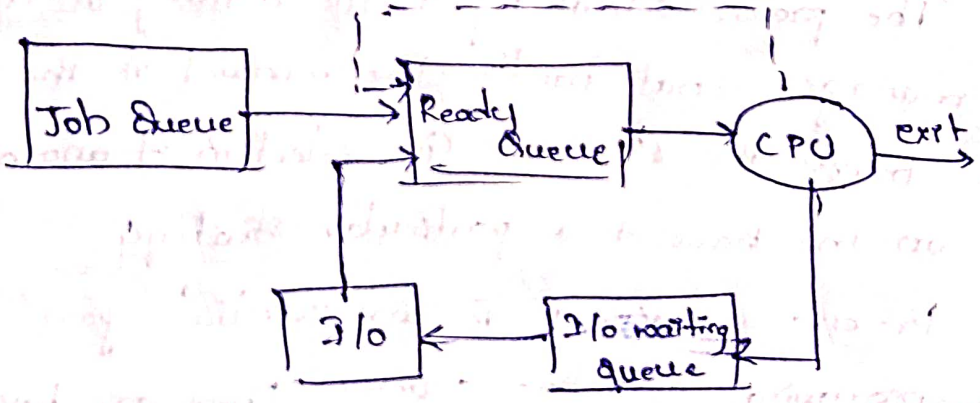
The OS maintains all process control Blocks (PCBs) in process scheduling Queues. The OS maintains a seperate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue.

There different scheduling Queues. They are.

1) Job Queue - This queue keeps all the processes in the system.

2) Ready Queue - This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.

3) Device Queue - The processes which are blocked due to unavailability of an I/O device.



Q) What do you mean schedulers? explain mid-term scheduler?

A) Schedulers-

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types -

1) Long - term scheduler
2) Short - term scheduler
3) Medium - term scheduler

Medium - term scheduler-

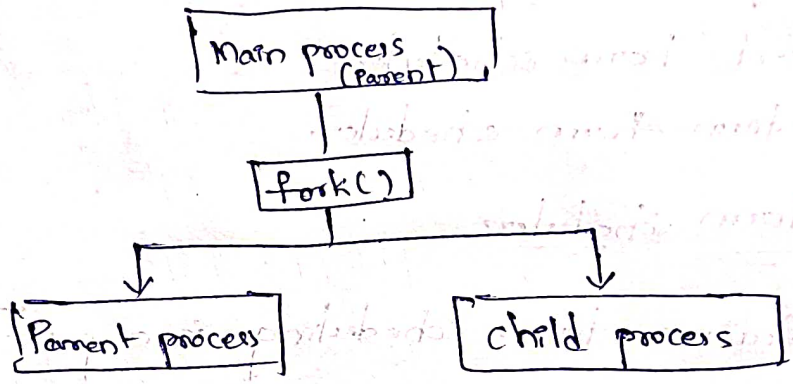
Medium - term scheduling is a part of swapping. It removes the process from the memory. It reduces the degree of multiprogramming. The medium - term scheduler is in-charge of handling the swapped-out processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping.

16) Process creation and termination with example?

A) Process creation:-

Process creation is achieved through the fork() system call. The newly created process is called the child process and the process that initiated it is called the parent process. After the fork() system call, now we have two processes - parent and child processes.




```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

Process termination:-

A process can terminate in either of the two ways -

a) Abnormally, occurs on delivery of certain signals, say terminate signal.

b) Normally, using _exit() system call (or _Exit() system call) or exit() library function.

The difference between _exit() and exit() is mainly the cleanup activity.

11) How many interprocess communication will happen explain ways of each type of communication.

A) Inter process communication (IPC):

A process can be of two types:

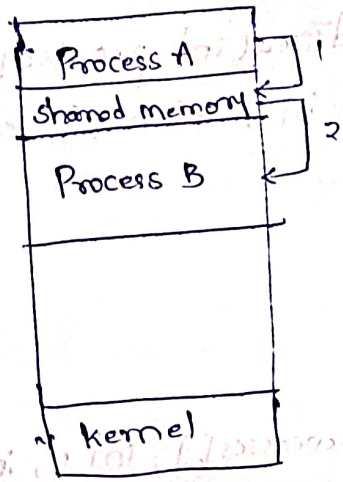
1) Independent process

2) Co-operating process.

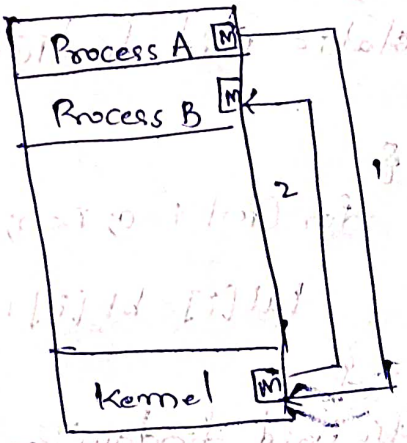
An independent process is not affected by the execution of other processes while a co-operating process can be affected by other executing processes. IPC is a mechanism that allows process to communicate with each other and synchronize their actions. processes can communicate with each other through both:

1) shared Memory

2) Message passing



Shared Memory



Message passing

1) Shared Memory:- Shared memory is a faster IPC system. It allows cooperating processes to access the same pieces of data concurrently.

2) Message passing:- Message passing model allows multiple processes to read and write data to the message queue without being connected to each other

12) Write down the pseudo codes of following algorithms.

1. FCFS, 2. SJF (Non-preemptive), 3. SRJF (Preemptive), 4. RR, 5. longest job first, 6. Producer-consumer & 7. bounded and buffer.

A)

1) FCFS:-

```
import java.text.ParseException;
class FCFS {
    static void findWaitingTime(int processes[], int n,
                                int bt[], int wt[])
    {
        wt[0] = 0;
        for (int i=1; i<n; i++) {
            wt[i] = bt[i-1] + wt[i-1];
        }
    }
    static void findTurnAroundTime(int processes[], int n,
                                   int b[], int wt[], int tat[])
    {
        for (int i=0; i<n; i++) {
            tat[i] = bt[i] + wt[i];
        }
    }
    static void findAvgTime(int processes[], int n, int bt[])
    {
        int wt[] = new int[n], tat = new int[n];
        int total_wt = 0, total_tat = 0;
        findWaitingTime(processes, n, bt, wt);
        findTurnAroundTime(processes, n, bt, wt, tat);
        System.out.printf("Processes Burst time waiting" + "turn around time\n");
```

```
for (int i=0; i<n; i++){
    total_wt = total_wt + wt[i];
    total_tat = total_tat + tat[i];
    System.out.printf("%d", (i+1));
    System.out.printf("%d", bt[i]);
    System.out.printf("%d", wt[i]);
    System.out.printf("%d\n", tat[i]);
}
public static void main(String[] args) throws ParseException
{
    int processes[] = {1, 2, 3};
    int n = processes.length;
    int burst_time[] = {10, 5, 8};
    findavgTime(processes, n, burst_time);
}
}
```

2) SJF (Non-preemptive):-

```
import java.io.*;
import java.util.*;
public class main{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        int n;
        int [][]A = new int[100][4];
        int total=0;
        float avg_wt, avg_tat;
        System.out.println("Enter no. of process:");
        n=input.nextInt();
        System.out.println("Enter Burst Time:");
        for(int i=0; i<n; i++) {
            System.out.print("p"+(i+1)+":");
```

```
A[i][i] = input.nextInt();
A[i][0] = i+1;
}
for (i=0; i<n; i++) {
    int index = i;
    for (int j = i+1; j<n; j++) {
        if (A[j][i] < A[index][i]) {
            index = j;
        }
    }
    int temp = A[i][i];
    A[i][i] = A[index][i];
    A[index][i] = temp;
    temp = A[i][0];
    A[i][0] = A[index][0];
    A[index][0] = temp;
}
A[0][2] = 0;
for (int i=1; i<n; i++) {
    A[i][2] = 0;
    for (int j=0; j<i; j++)
        A[i][2] += A[j][i];
    total += A[i][2];
}
avg_wt = (float)total/n;
total = 0;
avg_tat = (float)total/n;
System.out.println("Average Waiting Time = " + avg_wt);
System.out.println("Average Turnaround Time = "
                   + avg_tat);
}
}
```

3) SRJF :-

```
class process
{
    int pid;
    int bt;
    int art;
    public process (int pid, int bt, int art)
    {
      this.pid = pid;
      this.bt = bt;
      this.art = art;
    }
}
public class GFC
{
    public void findwaitingTime (process proc[], int n, int wt[])
    {
      int rt[] = new int[n];
      for (int i=0; i<n; i++)
          rt[i] = proc[i].bt;
          int complete = 0, t=0, minm = Integer.MAX_VALUE;
          int shortest = 0; finish_time;
            boolean check = false;
          while (complete != n){
            for (int j=0; j<n; j++)
            { if ((proc[j].art<=t) && (rt[j]<minm) && rt[j]>0)
              { minm = rt[j];
                shortest = j;
                check = true;
              }}
            if (check == false) {
                t++;
                continue;
            }
            rt[shortest]--;
```

```
min m = rt[shortest];
if (rem m == 0)
    min m = Integer.MAX_VALUE;
if (rt[shortest] == 0) {
    complete ++;
    check = false;
    finish_time = t+1;
    wt[shortest] = finish_time - proc[shortest].bt
                   - proc[shortest].art;
    if (wt[shortest] < 0)
        wt[shortest] = 0;
}
t++;
}
}
public static void main (String[] args)
{
    process proc[] = { new process(1,6,1),
                       new process(2,8,1),
                       new process(3,7,2),
                       new process(4,3,3) };
    findavgTime(proc, proc.length);
}
}
```

4) RR:

```
public class RR
{
    static void findWaitingTime(int processes[], int n,
                                int bt[], int wt[], int quantum
    int rem_bt[] = new int[n];
    for(int i=0; i<n; i++)
        rem_bt[i] = bt[i];
    int t=0;
```

```
while (true)
{
  boolean done = true;
  for (int i=0; i<n; i++)
    if (rem_bt[i] >0)
    {
      done = false;
      if (rem_bt[i] >quantum)
      {
        t= quantum;
        rem_bt[i] -= quantum;
      }
      else
      {
        t = t + rem_bt[i];
        wt[i] = t - bt[i];
        rem_bt[i] = 0;
      }
    }
  public static void main (String[] args)
  {
    int processes [] = {1, 2, 3};
      int n = processes.length;
      int burst_time[] = {10, 5, 8};
      int quantum=2;
    findavgTime (processes, n, burst_time, quantum);
  }
}
```

5) Longest Job First:

```
import java.util.*;
class GFG
{
  static class process{
    int processno;
    int AT, BT;
```

```
    int BTbackup;
    int WT;
    int TAT;
    int CT;
}

static void findCT()
{
    int max = 0, i;
    for (i=0; i<4; i++) {
        if (p[i].AT <= at) {
            if (p[i].BT > p[max].BT)
                max = i;
        }
    }
    return max;
}
Public static void main(String[] args)
{
    int i;
    for (i=0; i<4; i++) {
        p[i] = new process();
        p[i].processno = i+1;
    }
    for (i=0; i<4; i++)
        p[i].AT = i+1;
    for (i=0; i<4; i++) {
        p[i].BT = 2*(i+1);
        p[i].BTbackup = p[i].BT;
        prefinaltotal += p[i].BT;
    }
    for (i=0; i<4; i++) {
        System.out.print(p[i].processno + "\t");
        System.out.print(p[i].AT + "\t");
        System.out.print(p[i].BTbackup + "\t");
```

```
        System.out.print(p[i].CT + "\t");
        System.out.print(p[i].TAT + "\t");
        System.out.println(p[i].WT + "\t");
    }
    System.out.println();
    System.out.println("Total WT = " + totalWT);
    System.out.println("Average WT = " + totalWT/4.0);
  }
}
```

6) ~~Producer~~ Producer - Consumer:

```
import java.util.LinkedList;
public class ThreadExample {
    public static void main(String[] args)
        throws InterruptedException
    {
        final PC pc = new PC();
        Thread t1 = new Thread(new Runnable()
        {
            public void run()
            {
                try {
                    pc.produce();
                }
                catch(InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        Thread t2 = new Thread(new Runnable() {
            public void run()
            {
                try {
                    pc.consume();
                }
```

```
        catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
});
t1.start();
t2.start();
t1.join();
t2.join();
}
public void consume() throws InterruptedException
{
    while (true) {
        synchronized (this)
        {
            while (list.size() == 0)
            wait();
            int val = list.removeFirst();
            System.out.println("Consumer consumed- " + val);
            notify();
            Thread.sleep(1000);
        }
    }
}
```

4) Bound and Buffer:

```
import java.util.LinkedList;
public class Threadex {
    public static void main(String[] args)
        throws InterruptedException
    {
        final PC pc = new PC();
        Thread t1 = new Thread(new Runnable(){
        public void run()
```

```
{
    try {
        pc.produce();
    }
    catch(InterruptedException e){
        e.printStackTrace();
    }
}
}
});
```

```
public void consume() throws InterruptedException
{
    while (true) {
        synchronized (this)
        {
            while (list.size()==0)
                wait();
            int val = list.removeFirst();
            System.out.println("consumer consumed:"
                                        +val);
            notify();
            Thread.sleep(1000);
        }
    }
}
```