islington college
(इस्लिङ्टन कलेज)

LONDON METROPOLITAN UNIVERSITY

Module Code & Module Title

CS5001NA Networks and Operating System

Assessment Weightage & Type

20% Individual Coursework

Year and Semester

2020-21 Autumn

Student Name: Niwahang Angbuhang

London Met ID: 20048942

College ID: np01cp4s210237

Assignment Due Date: 25th April 2022

Assignment Submission Date: 24th April 2022

Word Count (TASK B):

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

I. Task A

1. Introduction

The program developed shows the information on some music bands and members. A password is used to enter the program. The user will have to guess the best band and can select a list of band members to view their information. Bash script is used to develop the program therefore it is denoted by #! /bin/bash.

2. Script

```
1. #!/bin/bash
2. if ! [[ $1 =~ [Aa-Zz]+$ && $2 =~ ^[0-9]+$ ]]
3. then
4.         echo -e "Please enter your name.\nPlease enter your ID.\nNote:
   Please input alphabets and numbers in name and ID respectively."
5.         exit
6. fi
7. password=7777
8. n=3
9. while [ $n -ge 1 ]
10. do
11.         echo -e "Please enter the password: \c"
12.         read -s key
13.         if [ $key == $password ]
14.         then
15.                 break
16.         else
17.                 echo -e "The key you've entered is incorrect.\nPlease input
   again."
18.         fi
19.         (( n-- ))
20.         if [ $n -lt 1 ]
21.         then
22.                 echo -e "You have entered the password incorrectly 3
   times.\nPlease open the program again."
23.                 exit
24.         fi
25. done
26.
27. echo -e "\n-----------------------------------"
28. echo -e "|       Welcome To The System     |"
29. echo -e "-----------------------------------"
30. echo -e "|         ID : $2           |"
31. echo -e "|        Name: $1           |"
32. echo -e "-----------------------------------"
33. echo -e "|        Date:\c";date +"%m-%d-%Y    |"
34. echo -e "-----------------------------------"
35. echo -e "|       Time: \c";date +"%r          |"
36. echo -e "-----------------------------------"
37.
38. function repeat() {
39.         echo -e "Do you wish to repeat the program? (Y/N) \c"
40.         read ask
41.         case $ask in
42.         y|Y) Band ;;
43. n|N) echo -e "\nThank you for using the program."
44. exit ;;
45.         *) echo -e "Choose either (Y/y) or (N/n): \c"
46.                 read ask
47.         esac
48. }
49. function Select() {
50.         PS3="Please select an option from the below list: "
51.         select member in ${arr[@]}
52.         do
```

```
                case $member in
                        JL) cat JL ;;
                        FM) cat FM ;;
                        KC) cat KC ;;
                        AY|DH) echo The file is not given.
                                Select ;;
                        *) echo Please choose from the options only.
                                Select ;;
                esac
                break
        done
        repeat
}
function Members() {
        cat Members
        echo -e "The given names are the star band members from the above mentioned bands. Choose any three star band members from the given table using the member codes."
        while true
        do
            echo -e "Member codes: \c"
                read mem1 mem2 mem3
                declare -a arr=($mem1 $mem2 $mem3)
                if [ -z arr ]
                then
                        echo "Please enter the member codes."
                        Members
                elif [ ${#arr[@]} -ne 3 ]
                then
                        echo "Please enter the three codes from the above member codes."
                        Members
                elif [ $mem1 == $mem2 ] || [ $mem1 == $mem3 ] || [ $mem2 == $mem3 ]
                then
                        echo "Please do not repeat the same members."
                        Members
                else
                        for n in ${arr[@]}
                        do
                                if [ $n == "JL" ] || [ $n == "AY" ] || [ $n == "FM" ] || [ $n == "DH" ] || [ $n == "KC" ]
                                then
                                        continue
                                elif ! [[ $n =~ [A-Z] ]]
                                then
                                        echo "Please use captital letters for the member codes."
                                        Members
                                else
                                        echo "Please select the member codes from the given options."
                                        Members
                                fi
                        done
                fi
                 break
          done
          Select
```

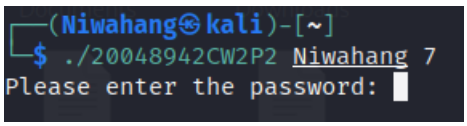
```
105. }
106. function Band() {
107.         cat Bands
108.         correct=n
109.         echo -e "Can you guess which is the best music band from the
   given options?\nUse the band codes to choose one out of the options."
110.         echo -e "Your guess: \c"
111.         read code
112.         until [ $correct == "y" ]
113.         do
114.                 if [ $code == "NIR" ]
115.                 then
116.                         echo -e "You have guessed it right.\n"
117.                         cat Nirvana
118.                         correct=y
119.                 elif ! [[ $code =~ [A-Z] ]]
120.                 then
121.                         echo "Please use capital letters for the codes."
122.                         echo -e "Guess again: \c"
123.                         read code
124.                 else
125.                         echo "You have guessed it wrong."
126.                         echo -e "Guess again: \c"
127.                         read code
128.                 fi
129.         done
130.         Members
131. }
132. Band
133.
```

3. Testing

Test 1

Test No.	1
Objective	To run the program
Action	The program is initiated with a name and ID as parameters.
Expected Result	The program should ask for a password.
Actual Result	The program asked for a password.
Conclusion	The test is successful.

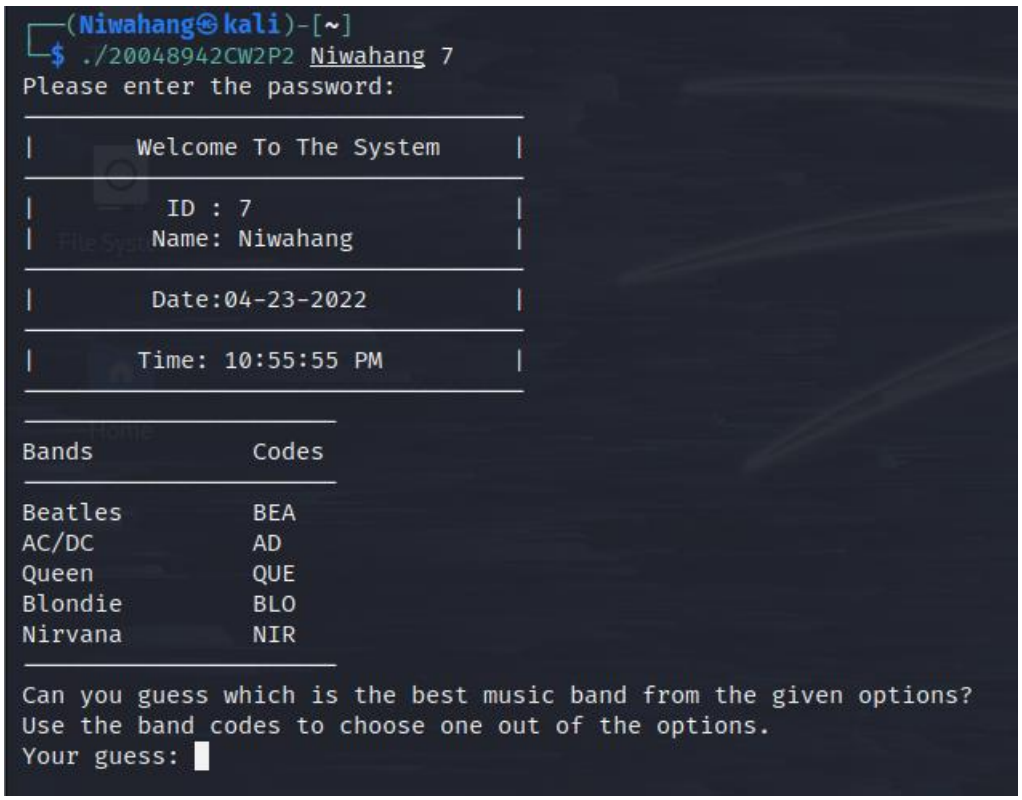
Screenshots:



Test 2

Test No.	2
Objective	To run the program
Action	A correct password is entered.
Expected Result	A welcome screen and lists of bands should be displayed.
Actual Result	A welcome screen and lists of bands were displayed.
Conclusion	The test is successful.

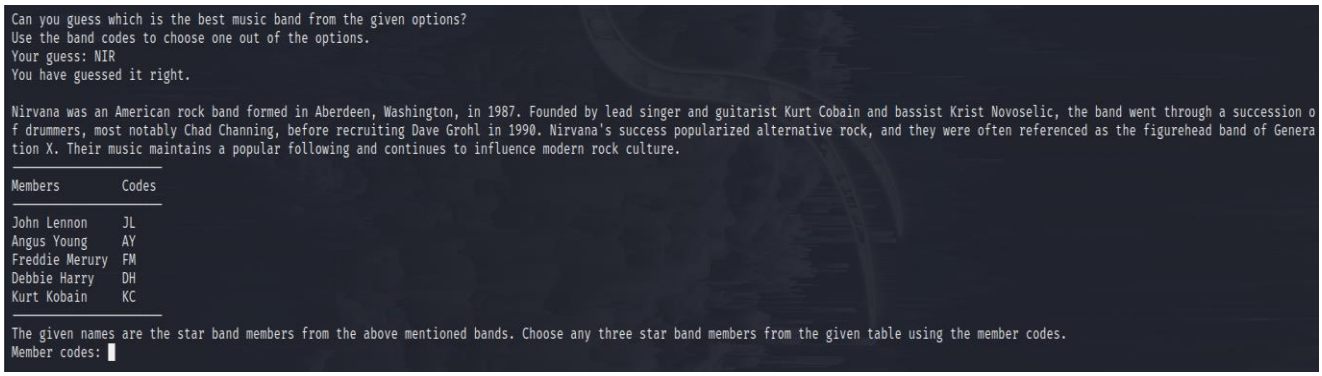
Screenshots:



Test 3

Test No.	3
Objective	To run the program
Action	The correct choice of band is entered in the guessing part.
Expected Result	Description of the band and a list of members should be displayed.
Actual Result	Description of the band and a list of members were displayed.
Conclusion	The test is successful.

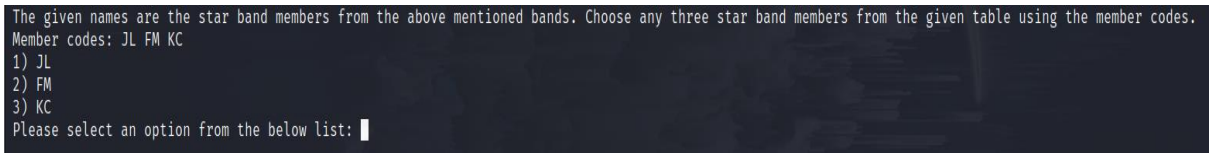
Screenshots:



Test 4

Test No.	4
Objective	To run the program
Action	Codes of three members are inputted.
Expected Result	An option to choose anyone from the selected members should be shown.
Actual Result	An option to choose anyone from the selected members was shown.
Conclusion	The test is successful.

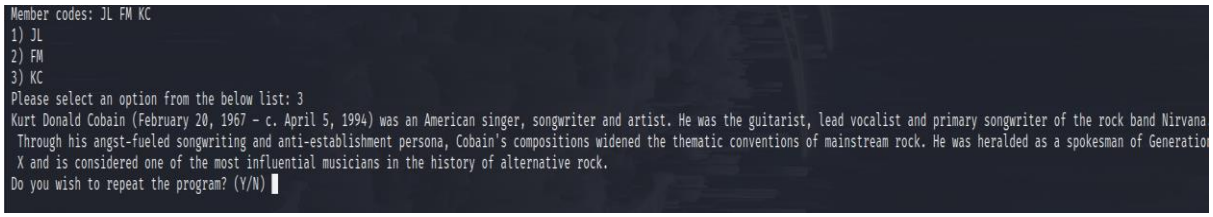
Screenshots:



Test 5

Test No.	5
Objective	To run the program
Action	One member is chosen from the options.
Expected Result	Details of the selected member and an option to repeat the program should be shown.
Actual Result	Details of the selected member and an option to repeat the program were shown.
Conclusion	The test is successful.

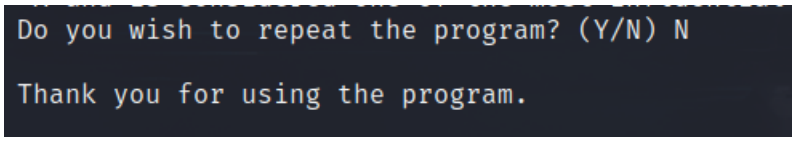
Screenshots:



Test 6

Test No.	6
Objective	To run the program
Action	'N' is inputted for the repeat option.
Expected Result	A thank you message should be shown, and the program should end.
Actual Result	A thank you message was shown and the program ended.
Conclusion	The test is successful.

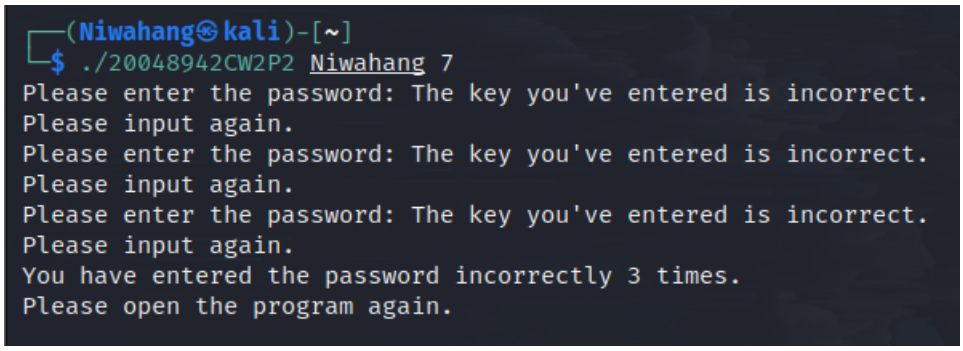
Screenshots:



Test 7

Test No.	7
Objective	To run the program
Action	The wrong password is entered three times.
Expected Result	The program should exit after a third failed login attempt.
Actual Result	The program exited after a third failed login attempt.
Conclusion	The test is successful.

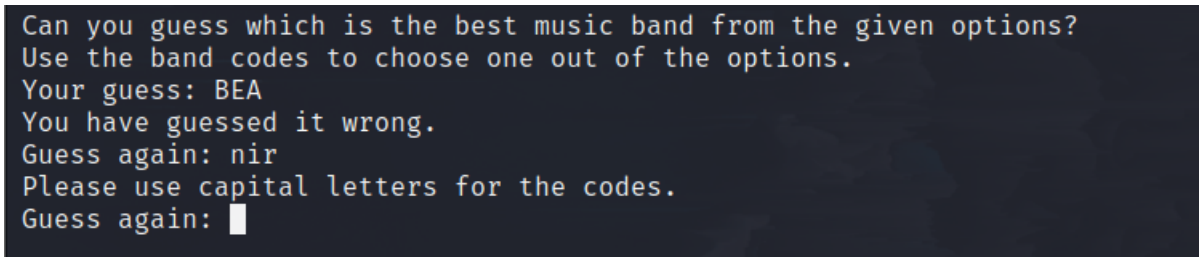
Screenshots:



Test 8

Test No.	8
Objective	To run the program
Action	The wrong band code is entered.
Expected Result	The program should show a message accordingly after the wrong band is entered.
Actual Result	The program showed a message accordingly after the wrong band was entered.
Conclusion	The test is successful.

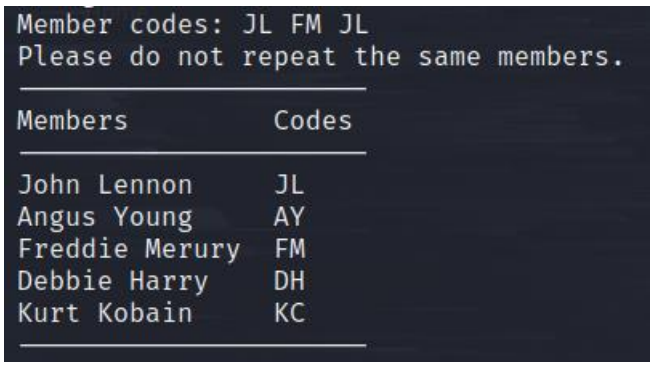
Screenshots:

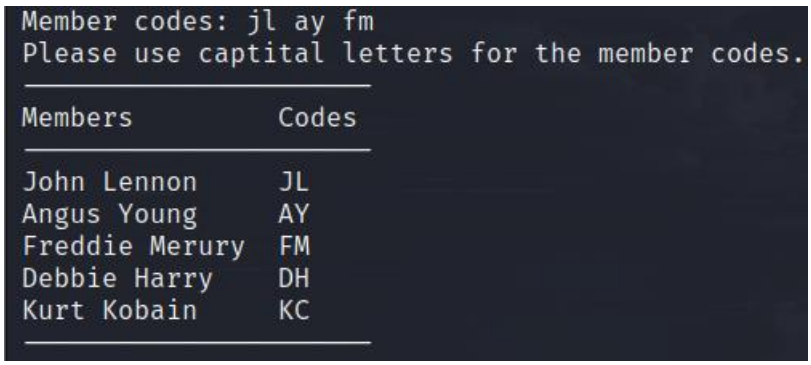


Test 9

Test No.	9
Objective	To run the program
Action	Same members are inputted, and characters are inputted in lower case.
Expected Result	A message should be shown accordingly.
Actual Result	A message was shown accordingly.
Conclusion	The test is successful.

Screenshots:

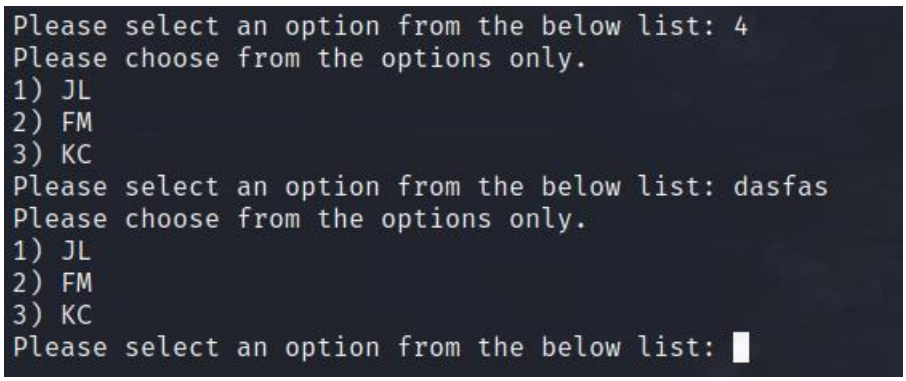




Test 10

Test No.	10
Objective	To run the program
Action	Out of index number is inputted when selecting one from the three members.
Expected Result	A message should be shown accordingly.
Actual Result	A message was shown accordingly.
Conclusion	The test is successful.

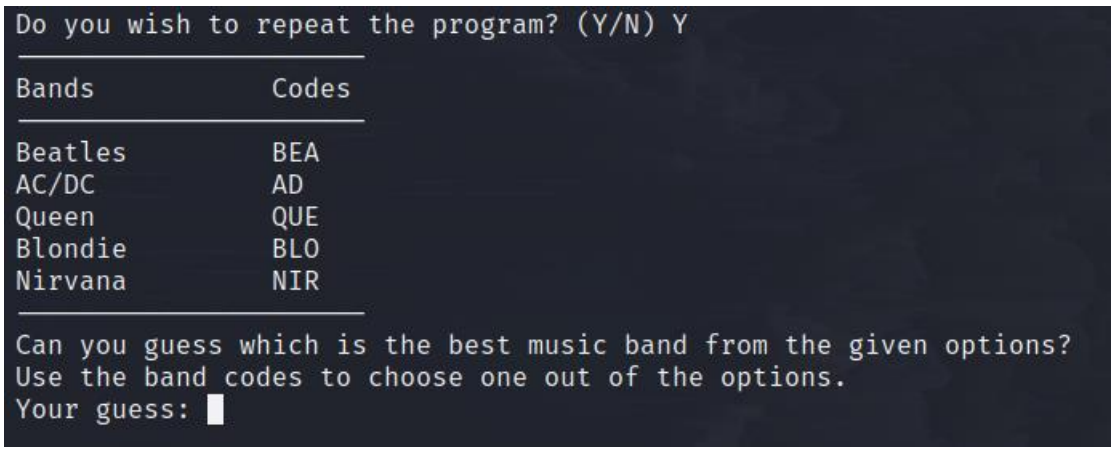
Screenshots:



Test 11

Test No.	11
Objective	To run the program
Action	'Y' is inputted to repeat the program.
Expected Result	The program should continue again from the selection of bands.
Actual Result	The program continued again with the selection of bands.
Conclusion	The test is successful.

Screenshots:



4. Contents of three files: (TEXTS)

Nirvana:

Nirvana was an American rock band formed in Aberdeen, Washington, in 1987. Founded by lead singer and guitarist Kurt Cobain and bassist Krist Novoselic, the band went through a succession of drummers, most notably Chad Channing, before recruiting Dave Grohl in 1990. Nirvana's success popularized alternative rock, and they were often referenced as the figurehead band of Generation X. Their music maintains a popular following and continues to influence modern rock culture.

JL:

John Winston Ono Lennon (born John Winston Lennon; 9 October 1940 – 8 December 1980) was an English singer, songwriter, musician, and peace activist who achieved worldwide fame as the founder, co-songwriter, co-lead vocalist and rhythm guitarist of the Beatles. Lennon was characterised by the rebellious nature and acerbic wit in his music, writing and drawings, on film, and in interviews. His song writing partnership with Paul McCartney remains the most successful in history.

FM:

Freddie Mercury (born Farrokh Bulsara; 5 September 1946 – 24 November 1991) was a British singer-songwriter, who was best known as the lead vocalist of the rock band Queen. Regarded as one of the greatest singers in the history of rock music, he was known for his flamboyant stage persona and four-octave vocal range. Mercury defied the conventions of a rock frontman with his theatrical style, influencing the artistic direction of Queen.

KC:

Kurt Donald Cobain (February 20, 1967 – c. April 5, 1994) was an American singer, songwriter, and artist. He was the guitarist, lead vocalist and primary songwriter of the rock band Nirvana. Through his angst-fuelled song writing and anti-establishment persona, Cobain's compositions widened the thematic conventions of mainstream rock. He was heralded as a spokesman of Generation X and is considered one of the most influential musicians in the history of alternative rock.

5. Conclusion

The program required by coursework 2 was developed and tested successfully. For developing the program, bash shell (bash) was used.

II. Task B

1. Introduction

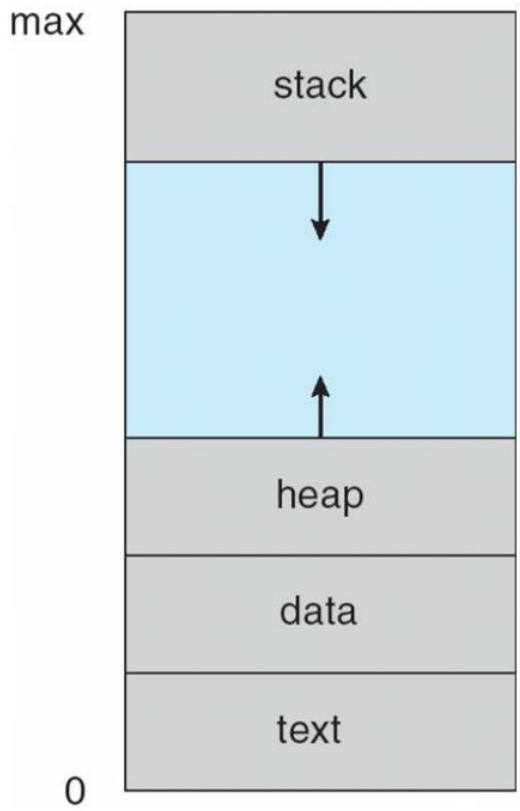
A process is a program that is being run. A computer runs many processes within milliseconds and the operating system handles all the processes and the resources that are being used. The handling of the different tasks like creation, scheduling, termination of processes, and a deadlock can be said as process management. Process management includes the structure of the process, hierarchies involved in the process, deadlock, scheduling of process, process states, process state models, process control block, policies, and algorithms of process, inter-process communication, and implementation of the process.

2. Background:

In early computers, only one program could be executed at a time. That one program used to take up all the resources of the system. There was no implementation of process management before. Slowly after new research and studying, the idea was developed. There was no idea for multitasking, the processes were still run one after another in the early phases. After some time, the idea of multitasking was made possible. Only two state model was used back then and after finding many drawbacks, newer state models were designed. Nowadays, computers can run many programs at the same time and the time to execute the process takes very less.

3. Process Structure

The structure diagram of the process is given below.



(Silberschatz, et al., 2012)

A process is divided into four different sections. They are text, data, heap, and stack.

Text:

The text section includes the current activity which is represented by the value of the Program Counter.

Data:

The data section contains the static and global variables.

Heap:

The heap section dynamically allocates memory to a process for its run time.

Stack:

The stack section stores temporary data like function parameters, return parameters, return addresses, and local variables.

4. Process Hierarchies

A computer is required to run many processes at a time according to the user's needs. Some processes create new processes after it is executed. When a process creates another process, then the parent and the child's processes seem to associate with one another in certain ways and further on. The child process may require creating other child processes. This parent-child structure of processes forms a hierarchy that is called process hierarchy.

There are different ways for the creation of a new child process. Some of them are:

Execution:

The child process is executed by the parent process at the same time, or it waits till all the children processes get terminated.

Sharing:

The parent or child process shares all the resources like memory or files, or the children process shares a segment of the parent’s resources, or the parent and children process share no resource in common.

5. Deadlock

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause. (Tanenbaum & Woodhull, 2006)

In a local area network, since most of the resources are shared, it becomes congested when more than two processes are in a deadlock state. For example, getting two processes to write different names on a printer may lead to both names being written on top of one or another. Since the operating system manages the resources, it is one of the important roles of the OS to manage deadlocks.

Deadlock can be caused if four of the following conditions become true.

Mutual Exclusion:

The mutual exclusion condition says that a resource can be used by only one process at a time.

Hold and Wait:

Hold and wait condition says a process is holding at least one resource and is waiting for other resources.

No Preemption:

No preemption condition says that if a process is holding a resource, it cannot be taken back from it by force. The process can only be released voluntarily.

Circular Wait:

The circular wait condition says a set of processes are waiting for each other for the resources in a circular form.

A resource scheduler can be used to detect deadlock as it keeps track of all the resources that are being used by different processes. After the deadlock is detected, the following methods can be used to recover from it.

Process Termination:

Deadlock can be eliminated by killing the processes involved in a deadlock.

Resource Preemption:

Deadlock can be eliminated by taking resources from a process and giving it to others.

6. Process Scheduling:

The activity of the process manager which determines if the process is in the ready state and should be moved to the running state is called process scheduling. The main aim of process scheduling is to maximize CPU use and have minimum response time for all programs.

There are three types of process schedulers. They are:

Long term or job scheduler:

A long-term scheduler select which processes should be brought to the ready queue. It is used when a process changes its state from new to ready.

Short term or CPU scheduler:

A short-term scheduler selects which process should be executed next and allocates the CPU. It increases system performance per the chosen set of criteria.

Medium-term scheduler:

A medium-term scheduler removes the processes from the memory and reduces the degree of multiprogramming. It is a process swapping scheduler.

6.1 Process States

A process goes through different phases as it is executed. The different states of process are described below:

New:

This is the first step of process state. The process is being created.

Ready:

After the process is created, the process is loaded into the main memory. The process is waiting to be assigned to a processor.

Running:

The CPU chooses the process for execution and instructions are executed.

Blocked or Wait:

In this state, the process is waiting for an event to occur. The process continues to wait in the main memory until the user or I/O operation is completed.

Terminated:

The process has finished execution. The process is killed, and PCB is deleted.

Ready Suspended:

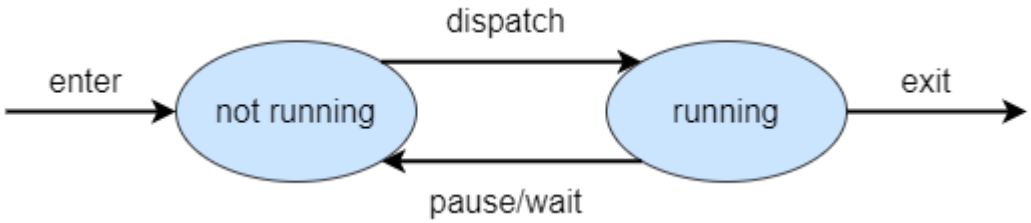
A process that is in ready state is moved to secondary memory from main memory due to lack of resources. The process will go back to ready state again when the resources are vacant.

Blocked Suspended:

This state is similar to ready suspended state. A process that was performing I/O operation is moved to secondary memory due to lack of resources. When the work is completed it may go to ready suspended.

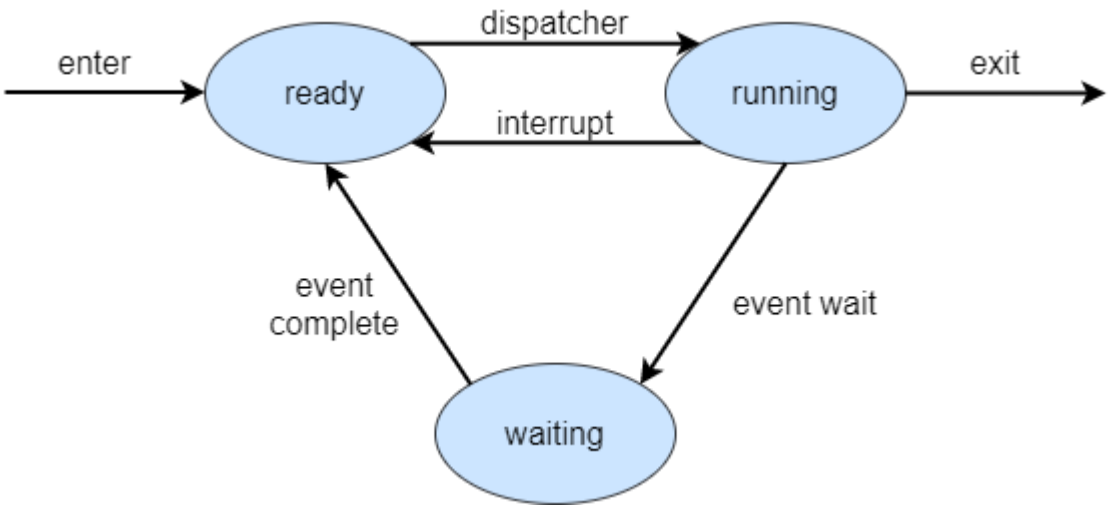
There are different process state models in the operating system. They are:

Two State Process Model:



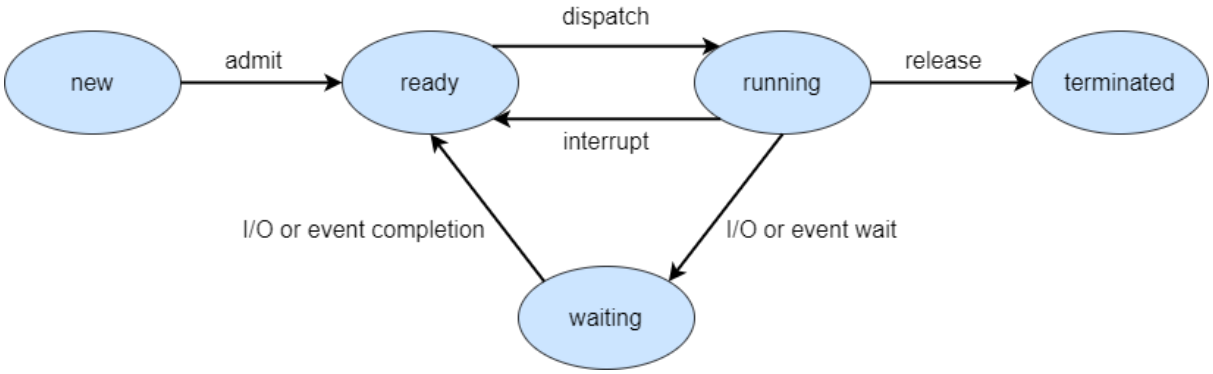
The two-state process block contains two states i.e., not running state and running state. A two-state can be created whether a process is being executed or not. When a process is first created then the process enters the not running state. When the CPU executes the process, it goes to the running state. If a process of higher priority is found the currently running process enters the not running state and higher priority process enters the running state. Every process in not running state is maintained in a queue. The use of two-state model can degrade performance as the process brought by the dispatcher might still be waiting for some event to occur or I/O request. Three state process model is designed to overcome this problem.

Three State Process Model:



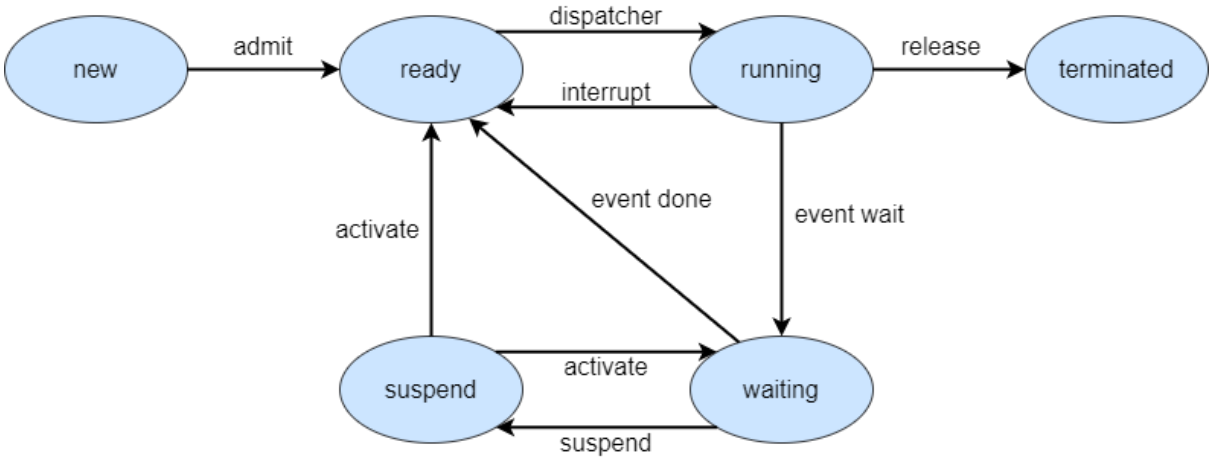
In three state process model, the not running state is divided into two parts: ready state and waiting or blocked state. The process in ready state is ready for execution whereas the process in waiting or blocked state is waiting for an event to occur. The operating system preserves a separate queue for both states. But in modern days, programs are very large, and loading all processes in the main memory is impossible. To overcome this problem, five state model is designed.

Five State Process Model:



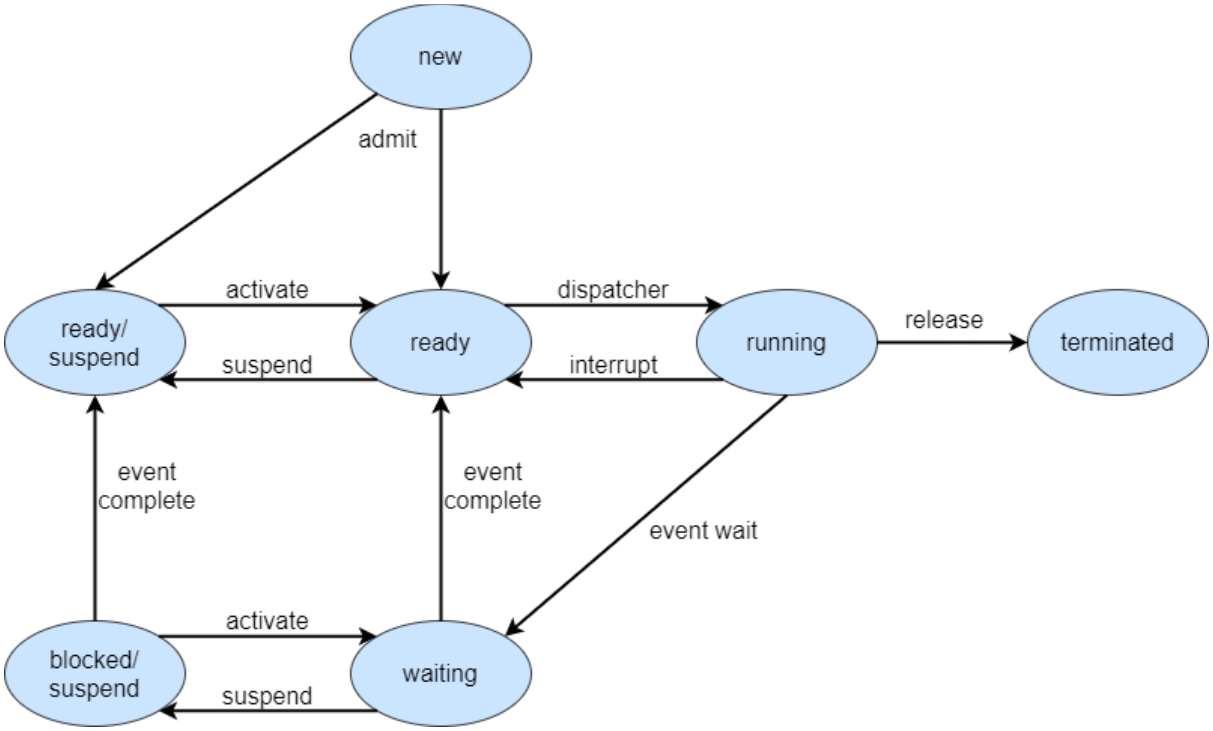
In five state process model, two new states are created: new state and terminated state. The process is just being created in the new state and not yet loaded in the main memory. In the terminated state, the process that is aborted due to some circumstances or that is finished running. One major drawback of this model is a situation may arise where the processor executes so fast that every process may move to waiting state and no process will be in ready state. It may lead to low CPU utilization. To overcome this problem, six-state model is designed.

Six State Process Model:



In six state process model, a new state is created: suspend state. In the suspend state, the process in waiting or block state is suspended and moved to secondary memory. Every process in suspend state is kept in a queue. With the help of this state, the CPU can take another process to the main memory. One major drawback of this model is the CPU might not know which process in the suspend queue is ready for execution. To overcome this problem, seven state model is designed.

Seven State Process Model:



In seven state process model, the suspend state is divided into two states: Blocked Suspend and Ready Suspend. In blocked suspend, the process in secondary memory is waiting and not ready for execution and in ready suspend, the process in secondary memory is ready for execution. It is also known as five state process model with two suspended states.

6.2 Process Control Blocks

A Process Control Block (PCB) is a data structure preserved by the operating system for every process. The PCB is used to track the process's execution status and store all the information about the process. Each block of memory contains information that is given below.

Process Control Block
Process State
Program Counter
CPU Registers
CPU Scheduling Information
Accounting Information
Memory Management Information
I/O Status Information

Process State:

A process can be new, ready, running, waiting, etc.

Program Counter:

The program counter stores the address of the next instruction to be executed for that process.

CPU Registers:

CPU registers include accumulators, index and general-purpose registers, and information on condition code.

CPU Scheduling Information:

CPU scheduling information includes process priority information, pointers to scheduling queues, and different scheduling parameters.

Accounting Information:

Accounting information includes the amount of user and kernel CPU time consumed, and time utilities like real-time used, job or process numbers, etc.

Memory Management Information:

Memory management information includes the page or segment tables and value of the base and limit registers. This depends on the memory system, which is used by the operating system.

I/O Status Information:

I/O status information includes the list of I/O devices and list of open file tables that are allocated to the process, etc.

6.3 Policies and Algorithm

There are two important process scheduling policies. They are:

Preemptive Scheduling:

In preemptive scheduling, task of higher priority is run even when tasks of lower priority are running. The lower priority task waits for some time until the higher task is completed.

Non-Preemptive Scheduling:

In non-preemptive scheduling, the process that occupies the CPU resources will release it either by terminating or switching context. This is the only method used for many hardware platforms.

There are six types of process scheduling algorithms. They are:

First Come First Serve (FCFS):

In FCFS, the process that requests the CPU early gets the CPU allocation first. It follows the First In, First Out (FIFO) queue. FCFS is easy to use.

Shortest-Job-First (SJF) Scheduling:

In FJF scheduling, the process that has the shortest exaction period is selected first for execution. It helps to decrease the average waiting time for processes waiting to be executed.

Shortest Remaining Time (SRT) :

In SRT, the process is allocated to the task that is closest to completion. It helps to prevent a new ready state process from taking the completion of an older process.

Round Robin Scheduling:

In round-robin scheduling, each process gets an equal share of time and when the process is not finished on time then another process will be executed, and the current process will remain in queue. It helps in starvation-free execution of processes.

Priority Based Scheduling:

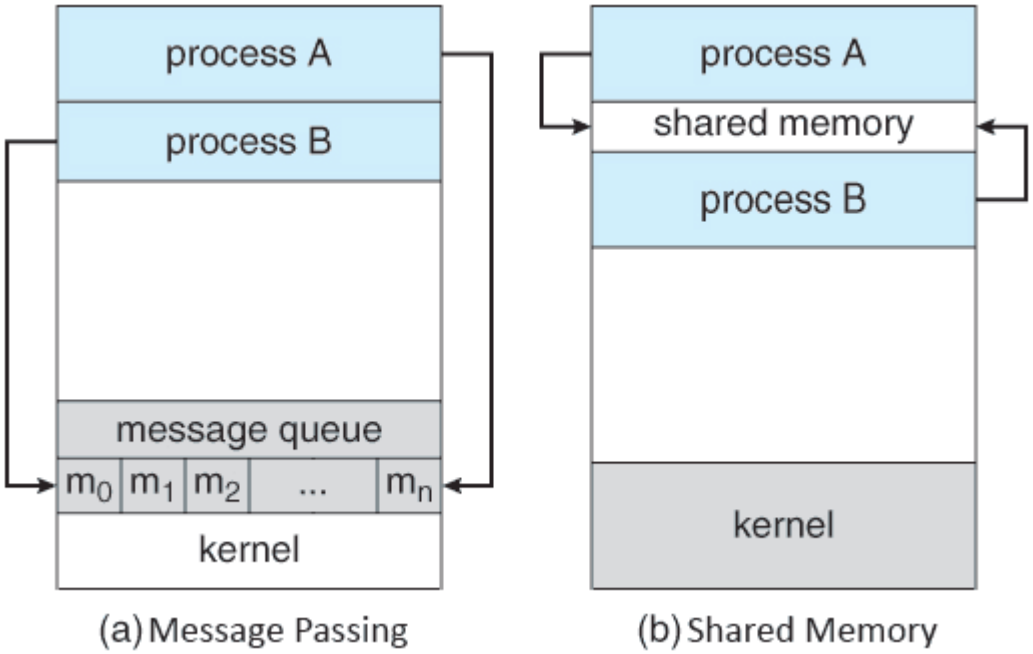
In priority-based scheduling, as the name suggests processes are executed based on their priority. The process with higher priority is executed first then processes with lower or equal priorities are executed on FCFS or round-robin method.

Multiple Level Queue Scheduling:

In multiple-level queue scheduling, the ready queue is separated into multiple queues, and processes are assigned to queues based on their specific property like size of memory, process priority, etc. Each queue can have different scheduling algorithms.

7. Inter-process Communication (IPC)

Inter-process communication (IPC) is a way to exchange data between various processes. IPC helps to speed up modularity. It also helps the OS to communicate with one another and synchronize their actions. There are some approaches for inter-process communication. Some of them are given below:



Message Passing:

Message passing is a way for a process to synchronize and communicate. The process communicates without using shared variables in message passing.

Shared Memory:

Shared memory is a memory that is used by multiple processes simultaneously. It is used for communication between processes. The memory is protected from each other through synchronization across all the processes.

8. Implementation of Process

To implement the process model, the operating system maintains a table (an array of structures), called the process table, with one entry per process. This entry contains important information about the process' state, including its program counter, stack pointer, memory allocation, the status of its open files, its accounting and scheduling information, and everything else about the process that must be saved when the process is switched from running to ready or blocked state so that it can be restarted later as if it had never been stopped. (Tanenbaum & Bos, 2015)

9. Conclusion

To sum up everything, a process is a program being executed. It plays an important role in the operating system. Due to the help of process management, the computer can perform tasks efficiently at a very high speed. The process control block helps to keep track of the process execution status. The process scheduling helps to handle remove a running process from the CPU and select a new or ready process to execute according to the plan.

References

Silberschatz, A., Galvin, P. B. & Gagne, G., 2012. *Operating System Concepts.* 9 ed. s.l.:John Wiley & Sons, Inc..

Tanenbaum, A. S. & Bos, H., 2015. *Modern Operating System.* 4th ed. s.l.:Pearson Education Inc.

Tanenbaum, A. S. & Woodhull, A. S., 2006. *Operating Systems Design and Implementation.* Third ed. s.l.:Pearson Education, Onc.