**Operating Systems (CSL 303)**

**Lab Workbook**

Logo

Description automatically generated

**Faculty name: Dr. Monika Yadav**

**Student name: Avtar Singh**

**Roll No.: 20CSU241**

**Semester: 5th**

**Group: FSB**

Department of Computer Science and Engineering TheNorthcap University

Gurugram- 122001, India

Session 2022-23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INDEX** | | | | | |
| **S. No.** | **Experiment** | **Date of Experiment** | **Date of Submission** | **CO Covered** | **Signature** |
| 1 | To familiarize the students to Linux interface and install Linux. | **3/8/2022** | **4/8/2022** | **CO1** |  |
| 2 | To write the shell programming code for the following.  a) Write A Shell Program of Hello World  b) Write a shell program to find factorial of a number.  c) Write a shell program to find gross salary of an employee.  d) Write a shell program to display the menu and execute instructions accordingly  (i)List of files (ii)Process Status (iii) Date (iv) users in program (v) Quit | **10/8/2022** | **11/8/2022** | **CO1** |  |
| 3 | To write the shell programming code for the following.  a) Write a shell program to find Fibonacci series.  b) Write a shell program to find largest of three numbers.  c) Write a shell program to find average of N numbers | **17/8/2022** | **18/8/2022** | **CO1** |  |
| 4 | To write the shell programming code for the following.  a) Write a shell program to check whether a number is even or odd  b) Write a shell program to find whether a number is prime or not.  c) Write a shell program to find whether a number is palindrome or not.  d) Write a shell program to type number 1 to 7 and then print its corresponding day of week | **28/8/2022** | **29/8/2022** | **CO1** |  |
| 5 | Implement the following CPU scheduling Algorithms.   1. FCFS with Arrival time   ii)FCFS without Arrival time | **7/9/2022** | **8/9/2022** | **CO2** |  |
| 6 | Implement the following CPU scheduling Algorithms.   * SJF (Non-Preemptive) * SJTF (shortest remaining time first -Preemptive SJF) | **14/9/2022** | **15/9/2022** | **CO2** |  |
| 7 | Implement the priority scheduling. | **12/10/2022** | **13/10/2022** | **CO2** |  |
| 8 | Implement the Round Robin scheduling. | **19/10/2022** | **20/10/2022** | **CO2** |  |
| 9 | Write a program to implement reader/writer problem using semaphore | **19/10/2022** | **20/10/2022** | **CO3** |  |
| 10 | Write a program to implement Dining Philosopher’s problem using semaphore | **9/11/2022** | **9/11/2022** | **CO3** |  |
| 11 | Write a program to implement Banker’s algorithm for deadlock avoidance. | **9/11/2022** | **9/11/2022** | **CO3** |  |
| 12 | Write a program for page replacement policy using a) LRU b) FIFO c) Optimal. | **16/11/2022** | **16/11/2022** | CO4 |  |

**Experiment No: 1**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date: 3/8/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective(s):**  To familiarize the students to Linux interface. |
| **Outcome:**   * The students will understand commands used in Linux. |
| **Problem Statement:**  Implement the following things:   * Cygwin Installation * Basic Linux commands |
| **Background Study:**  Cygwin is a open source tool which provides that functionality of the Linux in windows Operating System. Cygwin is a large collection of GNU and Open Source tools which provide functionality similar to a [Linux distribution](https://en.wikipedia.org/wiki/Linux_distribution) on Windows. It is a DLL (cygwin1.dll) which provides substantial POSIX API functionality. |
| **Question Bank:**   1. **What is Linux?**   Ans : Linux® is an open source operating system (OS). An operating system is the software that directly manages a system's hardware and resources, like CPU, memory, and storage. The OS sits between applications and hardware and makes the connections between all of your software and the physical resources that do the work.   1. **How will you List files from a directory?**   Ans : The ls command is used to list files. "ls" on its own lists all files in the current directory except for hidden files.   1. **How files in a directory can be removed?**   Ans : 1. To remove an empty directory, use either rmdir or rm -d followed by the directory name: rm -d dirname rmdir dirname.  2. To remove non-empty directories and all the files within them, use the rm command with the -r (recursive) option: rm -r dirname.   1. **How to find out a word in a file?**   Ans : Grep is a Linux / Unix command-line tool used to search for a string of characters in a specified file.   1. **What are wildcards?**   Ans : A wildcard is a symbol that takes the place of an unknown character or set of characters. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Commands :**

***1. uname => Displays Linux system information***

***2. uname -r => Displays kernel release information***

***3. uptime => Displays how long the system has been running including load average***

***4. hostname => Shows the system hostname***

***5. hostname -i => Displays the IP address of the system***

***6. last reboot => Shows system reboot history***

***7. date => Displays current system date and time***

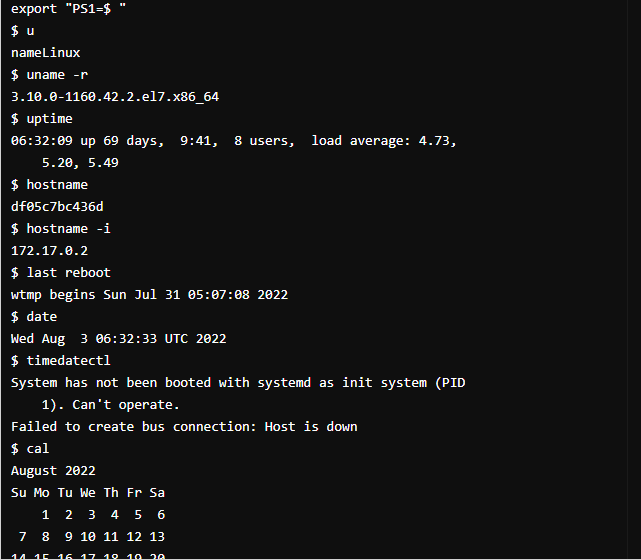
***8. timedatectl => Query and change the System clock***

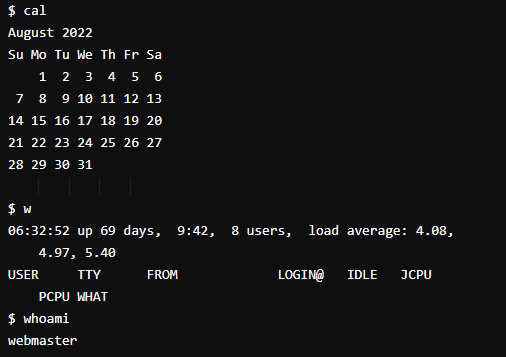
***9. cal => Displays the current calendar month and day***

***10. w => Displays currently logged in users in the system***

***11. whoami => Displays who you are logged in as***

***12. finger username => Displays information about the user***

****

****

**Hardware**

***13. dmesg => Displays bootup messages***

***14.cat /proc/cpuinfo => Displays more information about CPU e.g model, model***

***name, cores, vendor id***

***15 .cat /proc/meminfo => Displays more information about hardware memory e.g.***

***Total and Free memory***

***16. lshw => Displays information about system’s hardware configuration***

***17. lsblk => Displays block devices related information***

***18. free -m => Displays free and used memory in the system (-m flag***

***indicates memory in MB)***

***19. lspci -tv => Displays PCI devices in a tree-like diagram***

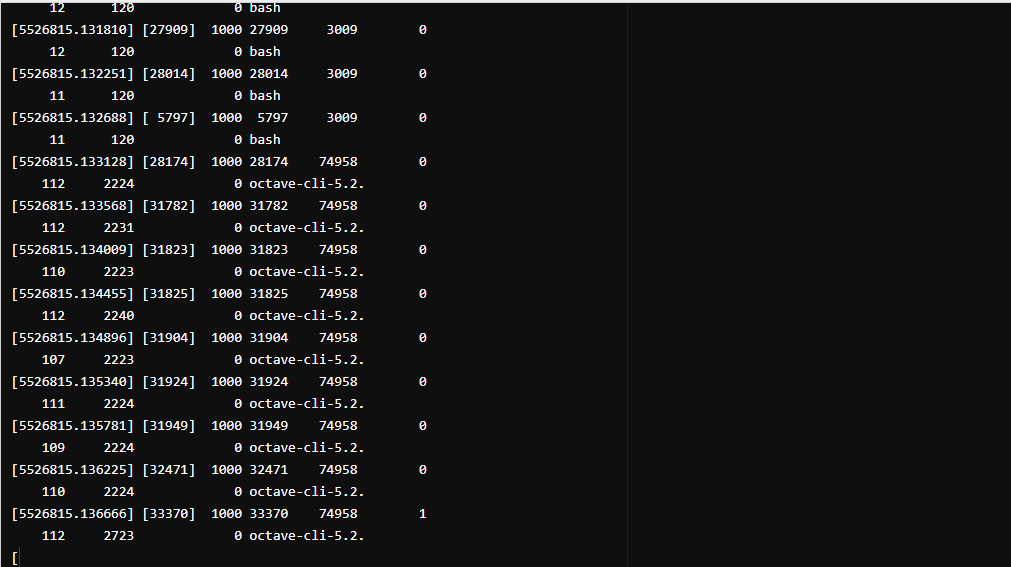
***20. lsusb -tv => Displays USB devices in a tree-like diagram***

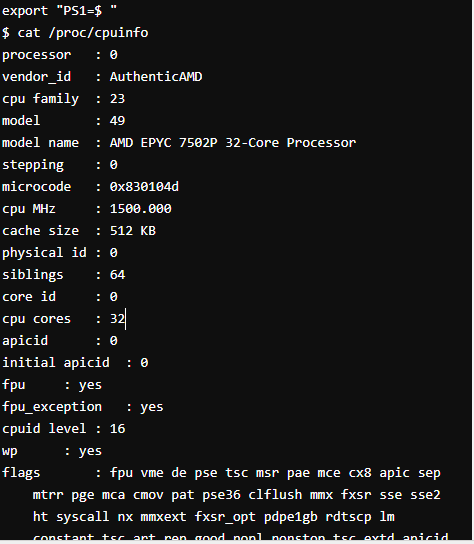
***21. dmidecode => Displays hardware information from the BIOS***

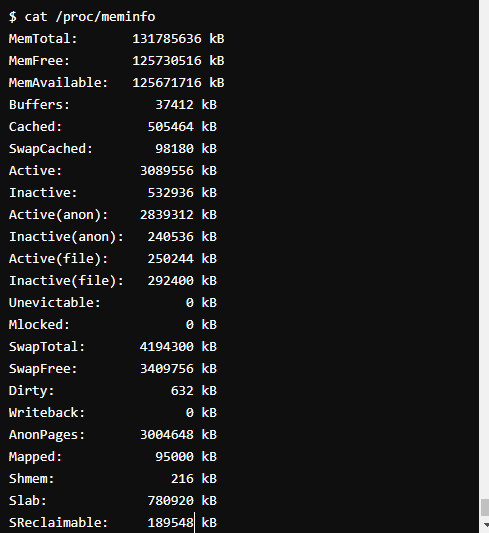
***22. hdparm -i /dev/xda => Displays information about disk data***

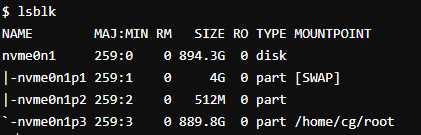
***23. hdparm -tT /dev/xda => Conducts a read speed test on device xda***

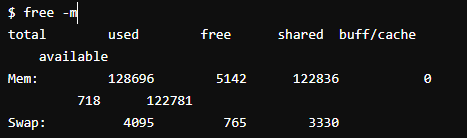
***24. badblocks -s /dev/xda => Tests for unreadable blocks on disk***

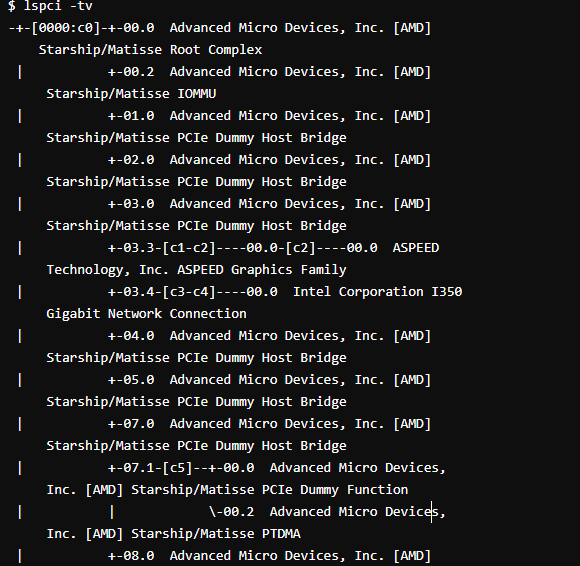
****

****

****

****

****

****

***User***

***1. id => Displays the details of the active user e.g. uid, gid, and***

***groups***

***2. last => Shows the last logins in the system***

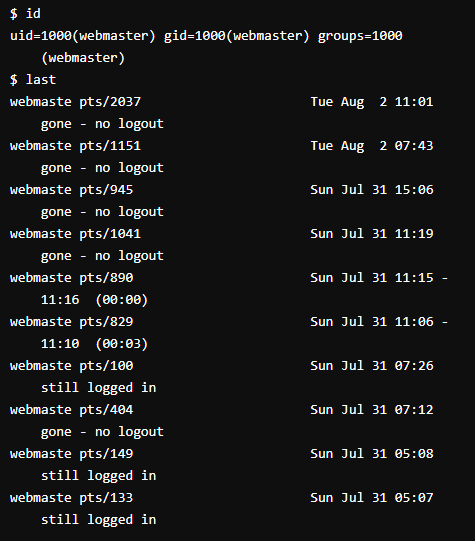
***3.who => Shows who is logged in to the system***

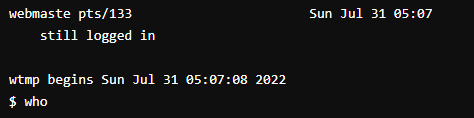
***4. groupadd “admin” => Adds the group ‘admin’***

***5. adduser “Sam” => Adds user Sam***

***6. userdel “Sam” => Deletes user Sam***

***7. usermod => Used for changing / modifying user information***

****

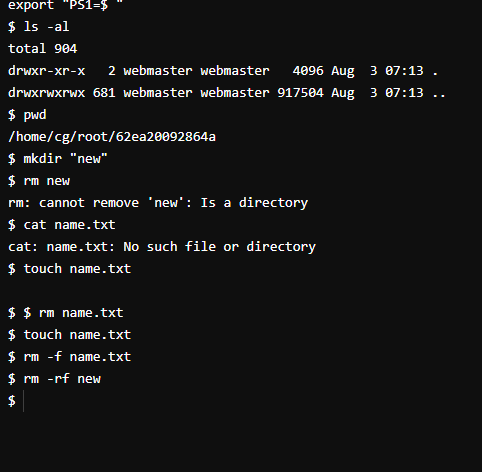
****

**File Commands**

1. ***ls -al => Lists files - both regular & hidden files and their permissions as well.***
2. ***pwd => Displays the current directory file path***
3. ***mkdir ‘directory\_name’ => Creates a new directory***
4. ***rm file\_name => Removes a file***
5. ***rm -f filename => Forcefully removes a file***
6. ***rm -r directory\_name => Removes a directory recursively***
7. ***rm -rf directory\_name => Removes a directory forcefully and recursively***
8. ***cp file1 file2 => Copies the contents of file1 to file2***
9. ***cp -r dir1 dir2 => Recursively Copies dir1 to dir2. dir2 is created if it does not***

***exist***

1. ***mv file1 file2 => Renames file1 to file2***
2. ***ln -s /path/to/file\_name***
3. ***link\_name***
   * ***Creates a symbolic link to file\_name***
4. ***touch file\_name => Creates a new file***
5. ***cat > file\_name => Places standard input into a file***
6. ***more file\_name => Outputs the contents of a file***
7. ***head file\_name => Displays the first 10 lines of a file***
8. ***tail file\_name => Displays the last 10 lines of a file***
9. ***gpg -c file\_name => Encrypts a file***
10. ***gpg file\_name.gpg => Decrypts a file***
11. ***wc => Prints the number of bytes, words and lines in a file***
12. ***xargs => Executes commands from standard input***

****

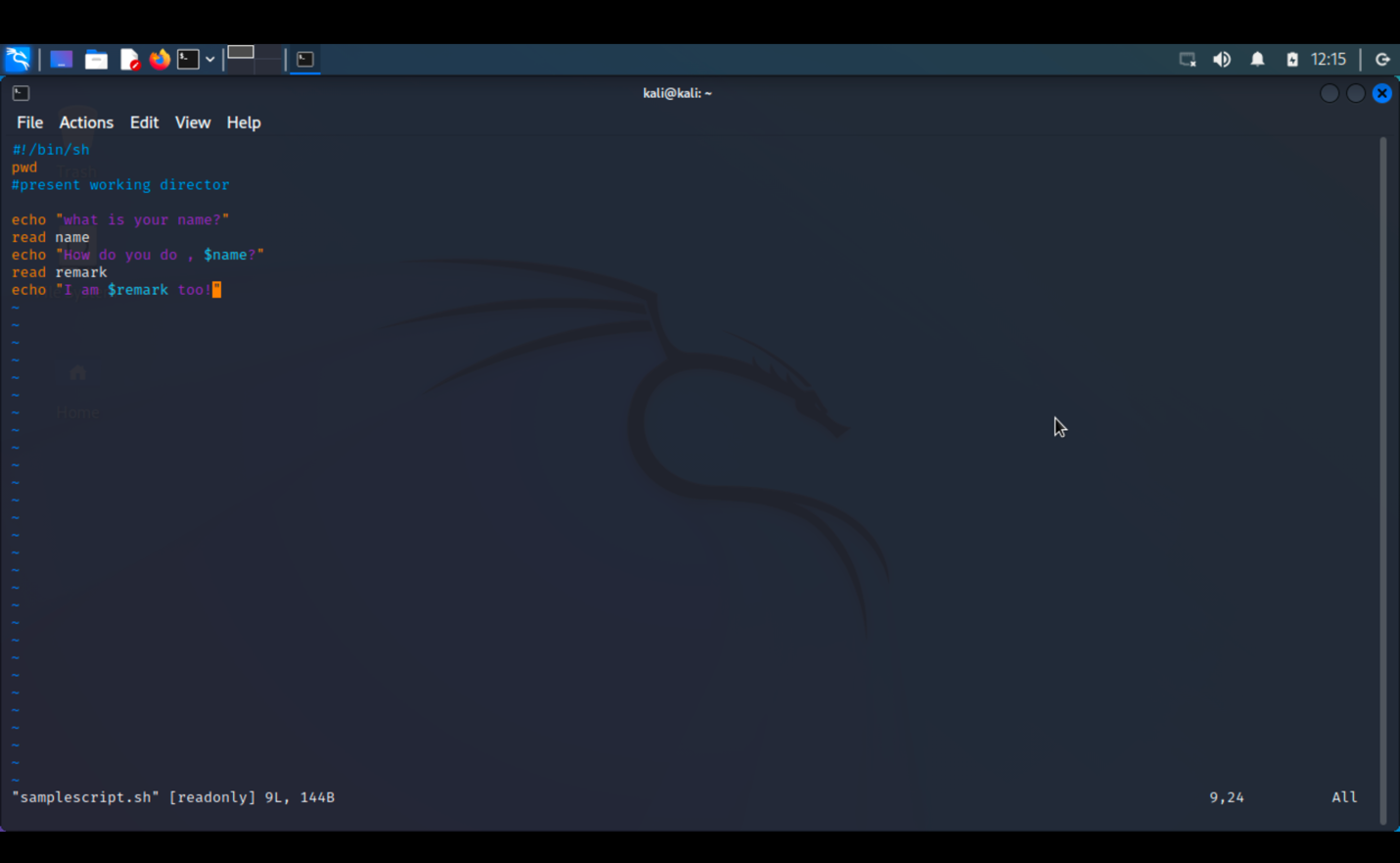
**EXPERIMENT NO. 2**

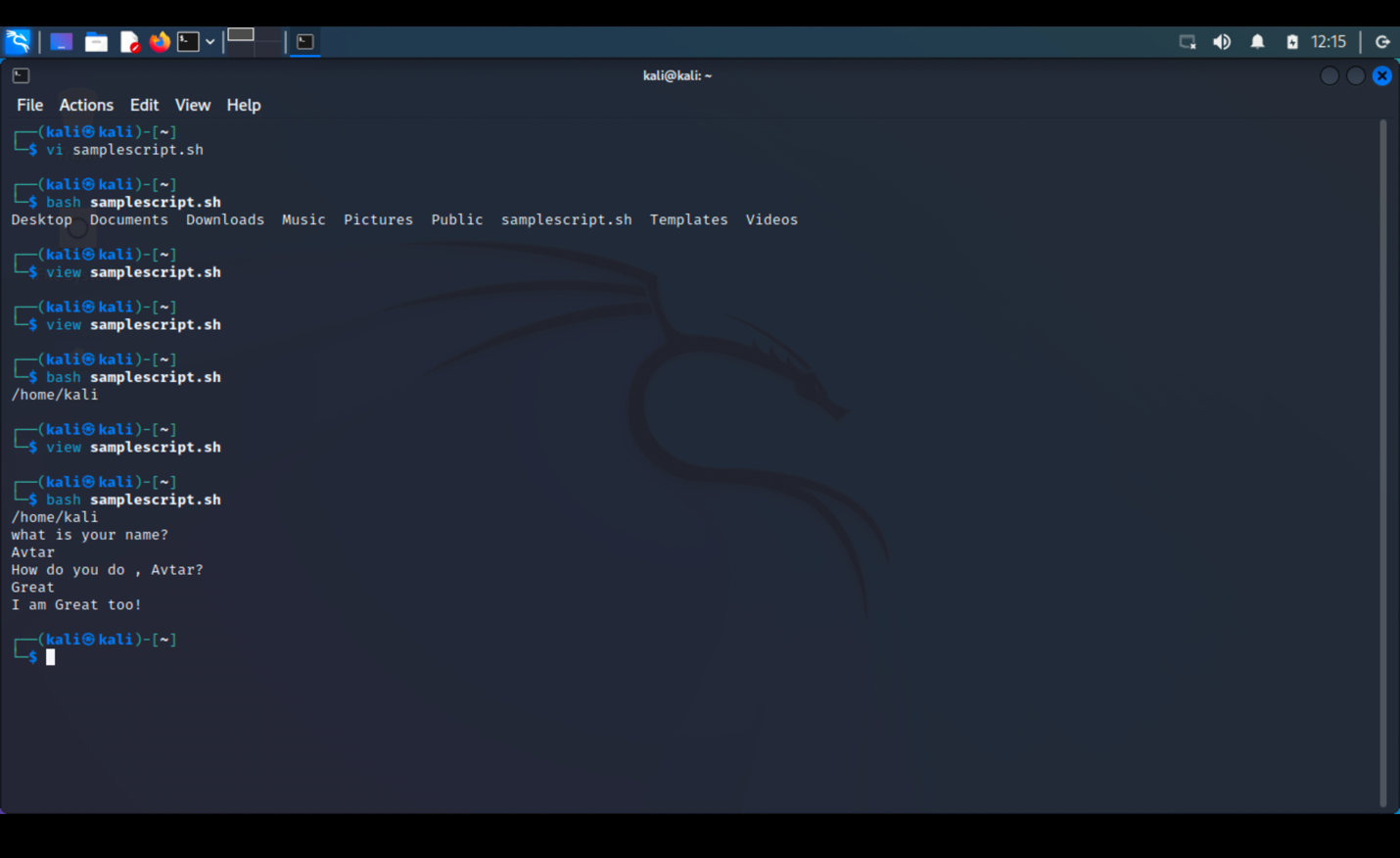
|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section:5th / FSB** |
| **Date:10/08/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  To write the shell programming code for the following. |
| **Outcome:**  Student is able to write code in shell programming |
| **Problem Statement:**  a) Write A Shell Program of Hello World  b) Write a shell program to find factorial of a number.  c) Write a shell program to find gross salary of an employee.  d) Write a shell program to display the menu and execute instructions accordingly  **(i)**List of files **(ii)**Process Status **(iii)** Date **(iv)** users in program **(v)** Quit |
| **Background Study:**  A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.  A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing is performed by shell script. So, we will use vi editor to edit our files. |
| **Question Bank:**   1. **What is a shell?**   Ans. The shell is **the layer of programming that understands and executes the commands a user enters**.   1. **What is the significance of $#?**   Ans.$# **shows the count of the arguments passed to the script**   1. **What are the different types of commonly used shells on a typical Linux system?**  * Ans.The Bourne Shell (sh) * The GNU Bourne-Again Shell (bash) * The C Shell (csh) * The Korn Shell (ksh) * The Z Shell (zsh)  1. **How will you pass and access arguments to a script in Linux?**   Ans.Arguments can be passed to the script when it is executed, **by writing them as a space-delimited list following the script file name**. Inside the script, the $1 variable references the first argument in the command line, $2 the second argument and so forth. The variable $0 references to the current script.   1. **Use sed command to replace the content of the file (emulate tac command) ?**   **sed -i 's/old-text/new-text/g' input.txt** |

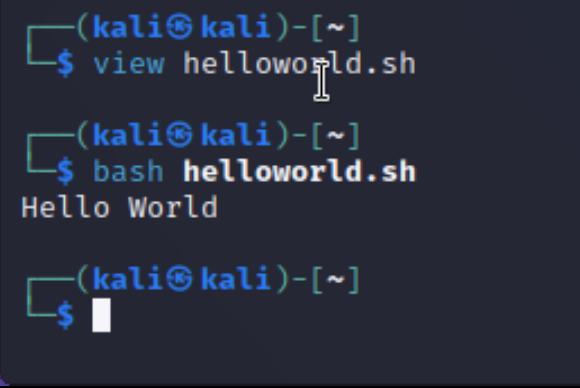
**Student Work Area**

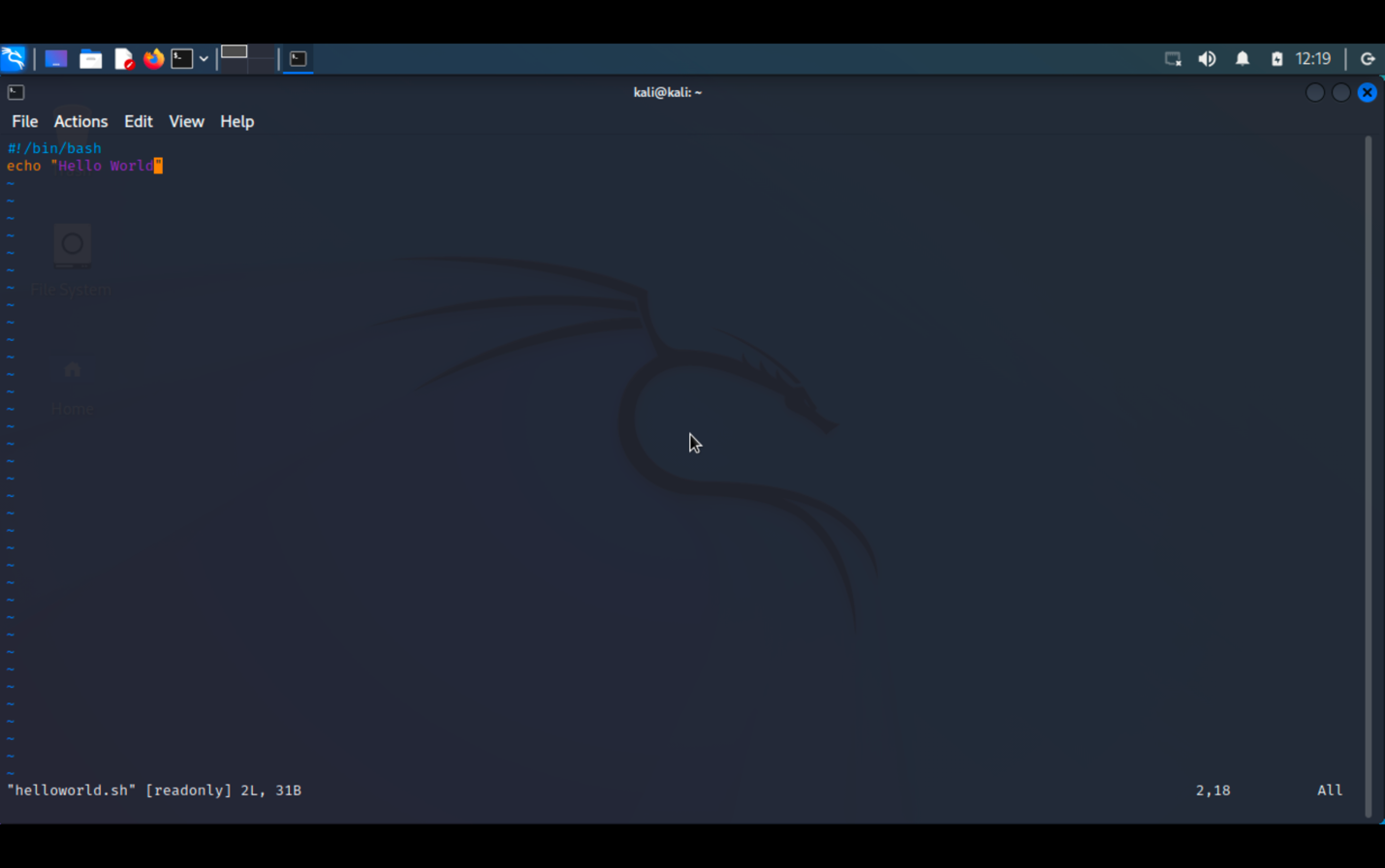
**Algorithm/Flowchart/Code/Sample Outputs**



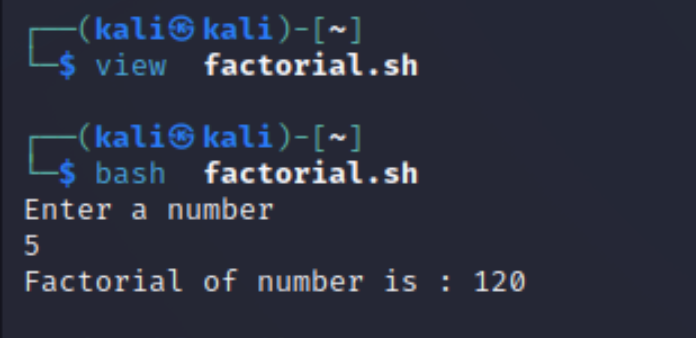


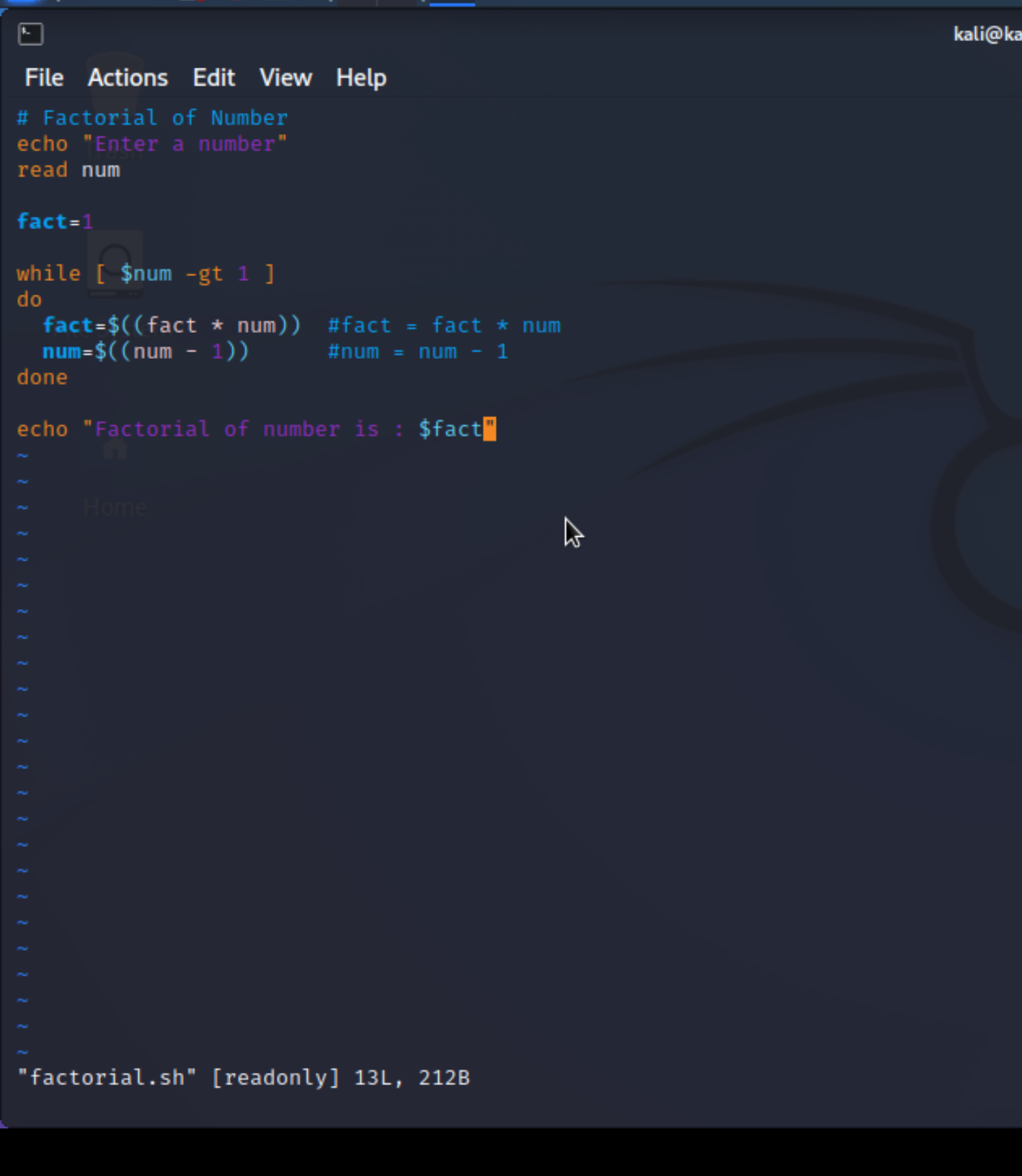
**a) Write A Shell Program of Hello World**

****

****

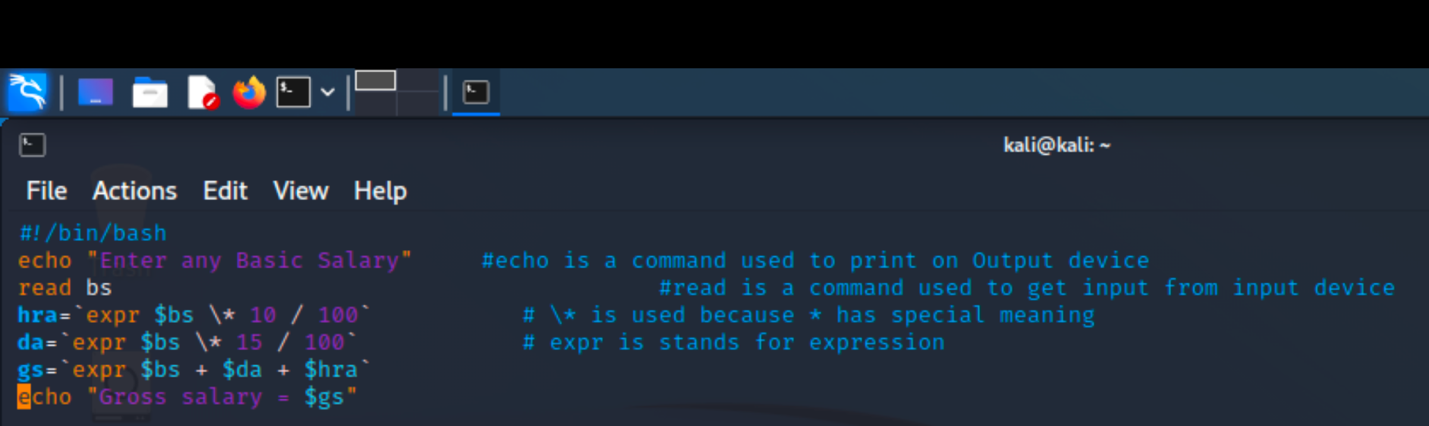
**b) Write a shell program to find factorial of a number.**





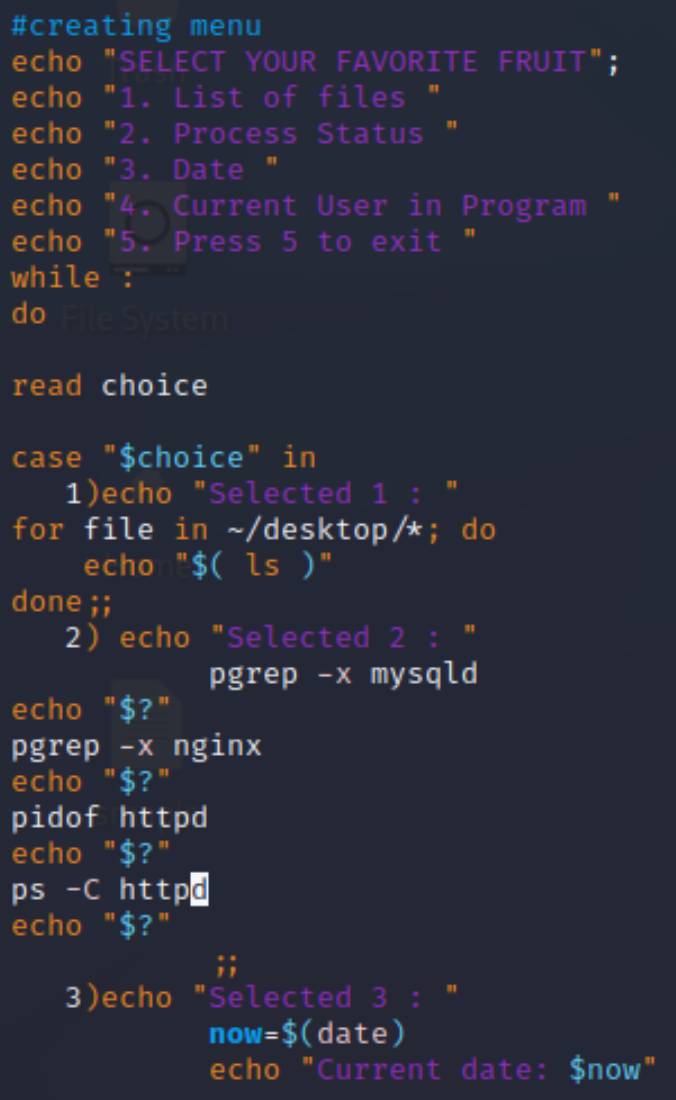
**c) Write a shell program to find gross salary of an employee.**

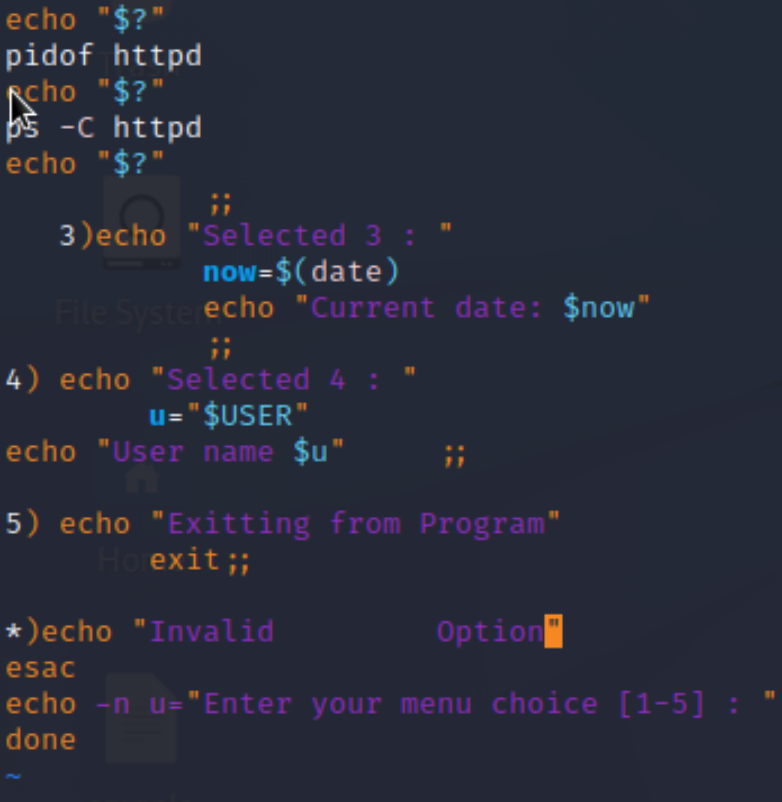


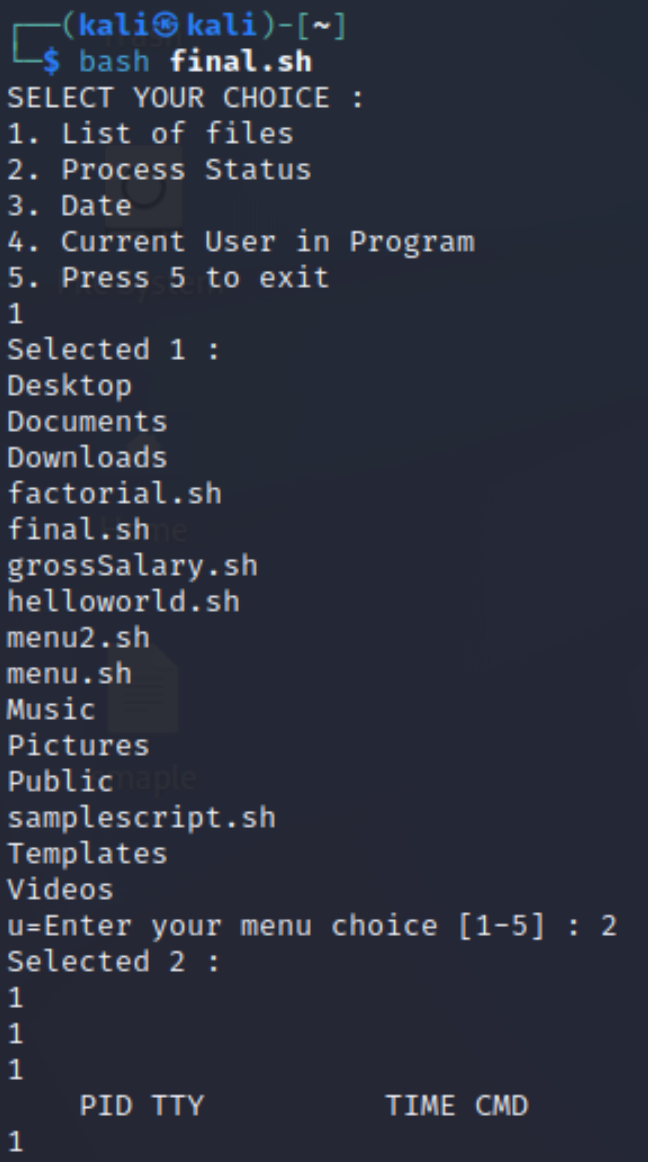


**d) Write a shell program to display the menu and execute instructions accordingly**

**(i)List of files (ii)Process Status (iii) Date (iv) users in program (v) Quit**







**Text

Description automatically generated**

**EXPERIMENT NO. 3**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20CSU241** |
| **Semester /Section: 5th / FSB** |
| **Date: 24/08/2022** |
| **Faculty Signature:** |
| **Marks:** |

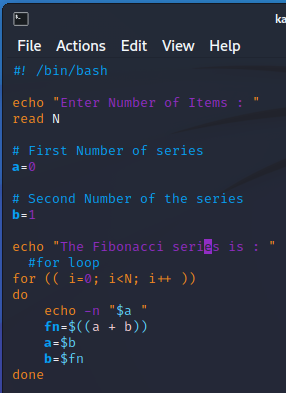
|  |
| --- |
| **Objective:**  To write the shell programming code for the following. |
| **Outcome:**  Student is able to write code in shell programming |
| **Problem Statement:**  a) Write a shell program to find Fibonacci series.  b) Write a shell program to find largest of three numbers.  c) Write a shell program to find average of N numbers |
| **Background Study:**  A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.  A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing is performed by shell script. So, we will use vi editor to edit our files. |
| **Question Bank:**   1. **How to use multi line comments in shell script?**   Ans : In Shell or Bash shell, we can comment on multiple lines **using << and name of comment**. we start a comment block with << and name anything to the block and wherever we want to stop the comment, we will simply type the name of the comment.   1. **What is the difference between soft and hard links?**   Ans : A hard link is a file all its own, and the file references or points to the exact spot on a hard drive where the Inode stores the data. A soft link isn't a separate file, it points to the name of the original file, rather than to a spot on the hard drive.   1. **Explain about loops and what are the loops available in LINUX?**   Ans : The for loop **operates on lists of items**. It repeats a set of commands for every item in a list.  There are **three types of shell loops** in UNIX/Linux: for loop. while loop. until loop.  You will use different loops based on the situation. For example, **the while loop executes the given commands until the given condition remains true; the until loop executes until a given condition becomes true**.   1. **What are absolute and relative paths.**   Ans : **An absolute path is defined as specifying the location of a file or directory from the root directory(/)**. In other words,we can say that an absolute path is a complete path from start of actual file system from / directory. Relative path is defined as the path related to the present working directly(pwd).   1. **How to debug a shell script.**   Ans : The debugging options available in the Bash shell can be switched on and off in multiple ways. Within scripts, we can either use the set command or add an option to the shebang line. However, another approach is to explicitly specify the debugging options in the command-line while executing the script. |

**Student Work Area**

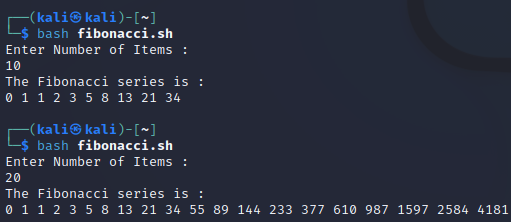
**Algorithm/Flowchart/Code/Sample Outputs**

**a) Write a shell program to find Fibonacci series.**

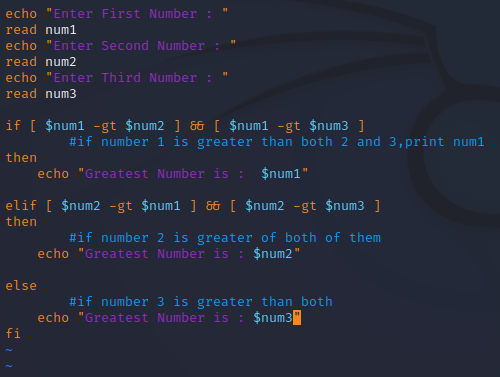
**Program :**



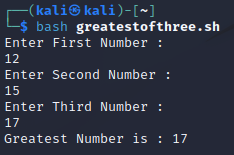
Output :



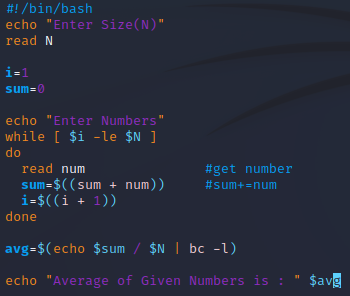
**b) Write a shell program to find largest of three numbers.**



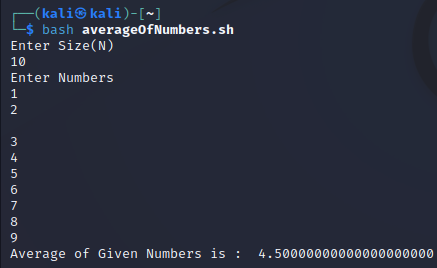
**OUTPUT :**



**c) Write a shell program to find average of N numbers**

****

**Output :**

****

**EXPERIMENT NO. 4**

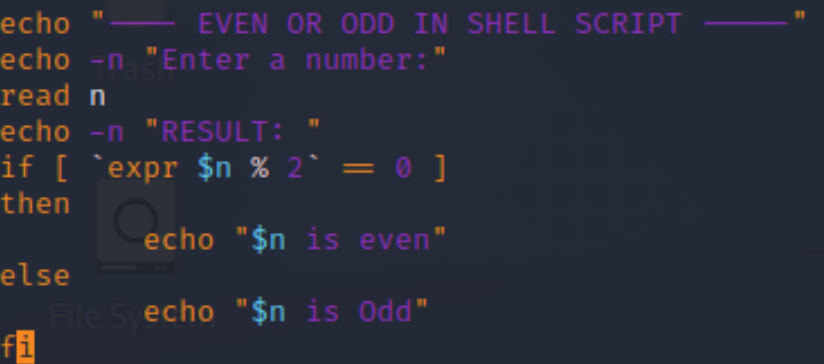
|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section: 5th /FSB** |
| **Date: 31/08/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  To write the shell programming code for the following. |
| **Outcome:**  Student is able to write code in shell programming |
| **Problem Statement:**  a) Write a shell program to check whether a number is even or odd  b) Write a shell program to find whether a number is prime or not.  c) Write a shell program to find whether a number is palindrome or not.  d) Write a shell program to type number 1 to 7 and then print its corresponding day of week |
| **Background Study:**  A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.  A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing is performed by shell script. So, we will use vi editor to edit our files. |
| **Question Bank:**   1. **What are Zoombie Process?**   On Unix and Unix-like computer operating systems, a zombie process or defunct process is a process that has completed execution (via the exit system call) but still has an entry in the process table: it is a process in the "Terminated state".   1. **What are different types of variables used in shell script?**   Two types of variables can be used in shell programming: Scalar variables. Array variables.   1. **What are the different types of modes available in Vi editor?** 2. Command Mode 3. Insert Mode 4. Escape Mode 5. **What are the different types of permission at file level in shell?**   1)Owner Permission  2)Group Permission  3)Other (World) Permission   1. **How to use comments in shell script.** #bash/shell |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample output**

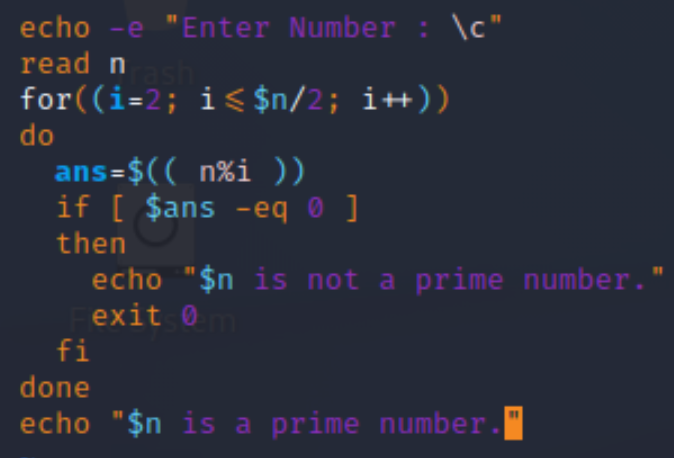
**a) Write a shell program to check whether a number is even or odd**

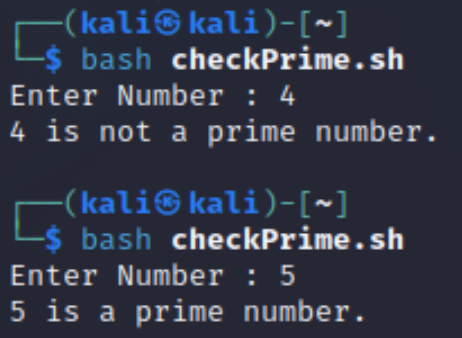
****

**Text

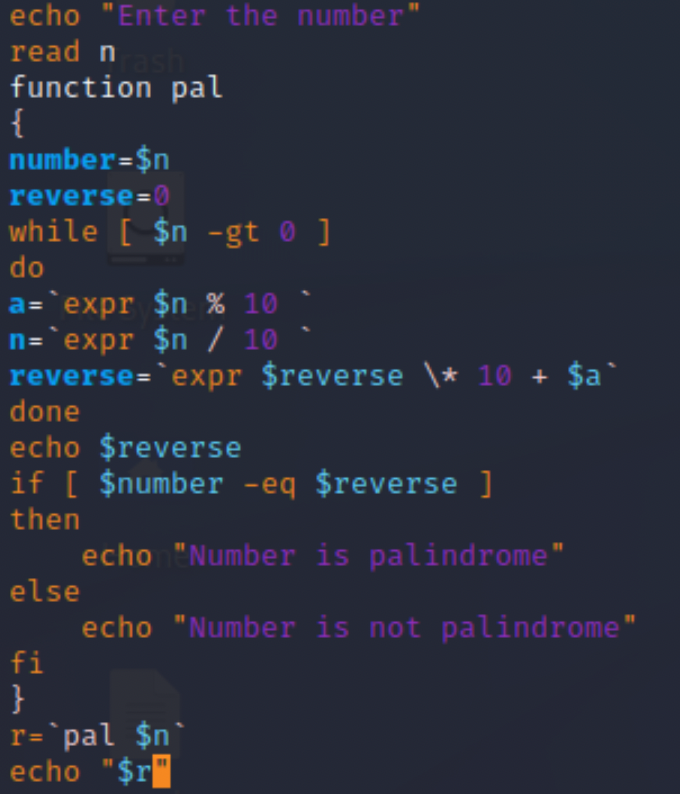
Description automatically generated**

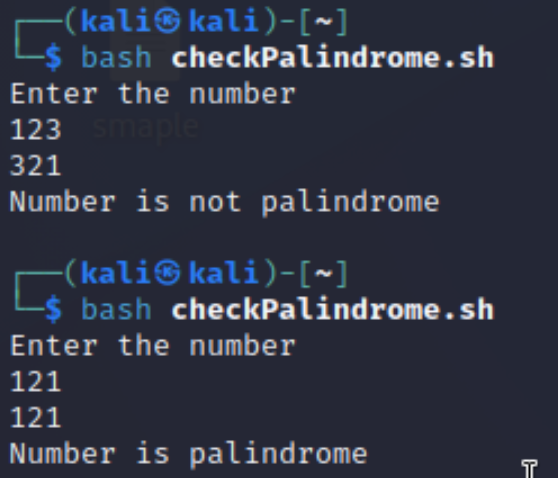
**b) Write a shell program to find whether a number is prime or not.**

****

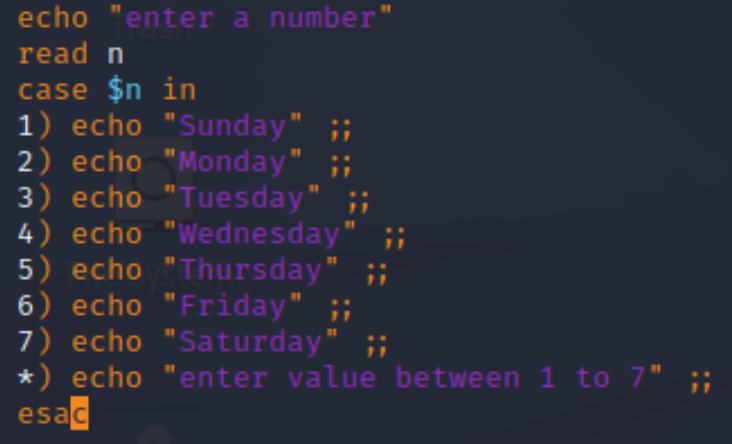
****

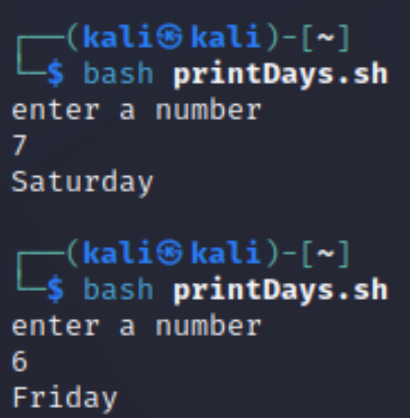
**c)Write a shell program to find whether a number is palindrome or not.**



****

**d) Write a shell program to type number 1 to 7 and then print its corresponding day of week**



****

**Experiment No: 5**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date: 7/9/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Write a program to implement CPU scheduling for first come first serve approach. |
| **Outcome:**  The students will understand the First-cum-first-serve algorithm |
| **Problem Statement:**  Implement the following CPU scheduling Algorithms.   1. FCFS with Arrival time 2. FCFS without Arrival time |
| **Background Study:**  **FCFS**   * The simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this algorithm, processes are assigned the CPU in the order they request it. * There is a single queue of ready processes. * The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. * The average waiting time under the FCFS policy, however, is often quite long. |
| **Question Bank:**   1. Which module gives control of the CPU to the process selected by the short-term scheduler?    1. **dispatche**r    2. interrupt    3. scheduler    4. none of the mentioned 2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called    1. job queue    2. **ready queue**    3. execution queue    4. process queue 3. The interval from the time of submission of a process to the time of completion is termed as    1. waiting time    2. **turnaround time**    3. response time    4. throughput 4. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?    1. **first-come, first-served scheduling**    2. shortest job scheduling    3. priority scheduling    4. none of the mentioned 5. In priority scheduling algorithm    1. **CPU is allocated to the process with highest priority**    2. CPU is allocated to the process with lowest priority    3. equal priority processes cannot be scheduled    4. none of the mentioned |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Q. Implement the following CPU scheduling Algorithms.**

1. **FCFS with Arrival time**

**Program :**

*#include <stdio.h>*

*int main() {*

*int bt[10] = {0}, at[10] = {0}, tat[10] = {0}, wt[10] = {0}, ct[10] = {0};*

*int n, sum = 0;*

*float totalTAT = 0, totalWT = 0;*

*printf("Enter number of processes ");*

*scanf("%d", &n);*

*printf("Enter arrival time and burst time for each process\n\n");*

*for (int i = 0; i < n; i++) {*

*printf("Arrival time of process[%d] ", i + 1);*

*scanf("%d", &at[i]);*

*printf("Burst time of process[%d] ", i + 1);*

*scanf("%d", &bt[i]);*

*printf("\n");*

*}*

*// calculate completion time of processes*

*for (int j = 0; j < n; j++) {*

*sum += bt[j];*

*ct[j] += sum;*

*}*

*// calculate turnaround time and waiting times*

*for (int k = 0; k < n; k++) {*

*tat[k] = ct[k] - at[k];*

*totalTAT += tat[k];*

*}*

*for (int k = 0; k < n; k++) {*

*wt[k] = tat[k] - bt[k];*

*totalWT += wt[k];*

*}*

*printf("Solution: \n\n");*

*printf("P#\t AT\t BT\t CT\t TAT\t WT\t\n\n");*

*for (int i = 0; i < n; i++) {*

*printf("P%d\t %d\t %d\t %d\t %d\t %d\n", i + 1, at[i], bt[i], ct[i], tat[i],*

*wt[i]);*

*}*

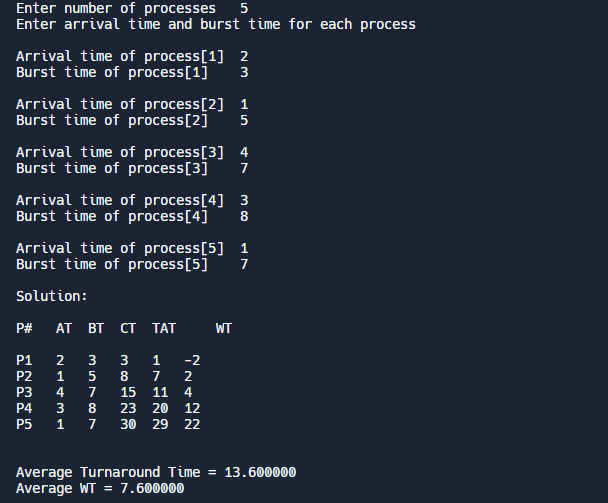
*printf("\n\nAverage Turnaround Time = %f\n", totalTAT / n);*

*printf("Average WT = %f\n\n", totalWT / n);*

*return 0;*

*}*

**OUTPUT :**



**b) FCFS without Arrival time**

**Program :**

*#include <stdio.h>*

*int main() {*

*int n, bt[20], wt[20], tat[20], avwt = 0, avtat = 0, i, j;*

*printf("Enter total number of processes(maximum 20):");*

*scanf("%d", &n);*

*printf("\nEnter Process Burst Time\n");*

*for (i = 0; i < n; i++) {*

*printf("P[%d]:", i + 1);*

*scanf("%d", &bt[i]);*

*}*

*wt[0] = 0; // waiting time for first process is 0*

*// calculating waiting time*

*for (i = 1; i < n; i++) {*

*wt[i] = 0;*

*for (j = 0; j < i; j++)*

*wt[i] += bt[j];*

*}*

*printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");*

*// calculating turnaround time*

*for (i = 0; i < n; i++) {*

*tat[i] = bt[i] + wt[i];*

*avwt += wt[i];*

*avtat += tat[i];*

*printf("\nP[%d]\t\t\t%d\t\t\t%d\t\t\t\t%d", i + 1, bt[i], wt[i], tat[i]);*

*}*

*avwt /= i;*

*avtat /= i;*

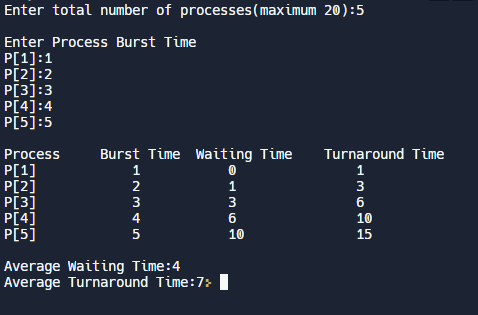
*printf("\n\nAverage Waiting Time:%d", avwt);*

*printf("\nAverage Turnaround Time:%d", avtat);*

*return 0;*

*}*

**OUTPUT :**

**

**Experiment No: 6**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date: 14/09/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Write a program to implement CPU scheduling for shortest job first (Preemptive and Non-Preemptive) |
| **Outcome:**  The students will understand the Shortest Job First scheduling mechanism |
| **Problem Statement:**  Implement the following CPU scheduling Algorithms.   * SJF (Non-Preemptive) * SJTF (shortest remaining time first -Preemptive SJF) |
| **Background Study:**   * Shortest Job first is having the advantage of a minimum average waiting time . * This algorithm associates with each process the length of the process next burst time.When CPU is available it assigned to the process that has the smallest next CPU burst time.if CPU burst time of two process is same then it follows FCFS. * It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing. * It is practically infeasible as Operating System may not know burst time and therefore may not sort them. |
| **Question Bank:**   1. [scheduling algorithm](https://t4tutorials.com/round-robin-process-scheduling-algorithm-in-operating-systems/) In multilevel feedback A. processes are not classified into groups **B. a process can move to a different classified ready queue…** C. classification of the ready queue is permanent D. none of the mentioned 2. Select one which algorithms tend to minimize the process flow time? A. First come First served B. Earliest Deadline First **C. Shortest Job First** D. Longest Job First 3. The process can be classified into many groups in A. [shortest job scheduling algorithm](https://t4tutorials.com/shortest-job-first-scheduling-sjf-process-scheduling-in-operating-systems/) **B. multilevel queue scheduling algorithm** C. round-robin scheduling algorithm D. priority scheduling algorithm 4. The turnaround time for short jobs during multiprogramming is usually Shortened and that for long jobs is slightly \_\_\_\_\_\_\_\_\_\_\_ A. Shortened B. Unchanged **C. Lengthened** D. Shortened 5. Time quantum can be said A. multilevel queue scheduling algorithm **B. round-robin scheduling algorithm** C. shortest job scheduling algorithm D. priority scheduling algorithm |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**1. Pre-emptive SJF Scheduling**

*#include <stdio.h>*

*int main() {*

*int arrival\_time[10], burst\_time[10], temp[10];*

*int i, smallest, count = 0, time, limit;*

*double wait\_time = 0, turnaround\_time = 0, end;*

*float average\_waiting\_time, average\_turnaround\_time;*

*printf("\n Pre-emptive(Can take Processor) SJF Scheduling : \t");*

*printf("\nCriteria : Burst Time (Execution Time )\t");*

*printf("\nEnter the Total Number of Processes:\t");*

*scanf("%d", &limit);*

*printf("\nEnter Details of %d Processes\n", limit);*

*for (i = 0; i < limit; i++) {*

*printf("\nEnter Arrival Time:\t");*

*scanf("%d", &arrival\_time[i]);*

*printf("Enter Burst Time:\t");*

*scanf("%d", &burst\_time[i]);*

*temp[i] = burst\_time[i];*

*}*

*burst\_time[9] = 9999;*

*for (time = 0; count != limit; time++) {*

*smallest = 9;*

*for (i = 0; i < limit; i++) {*

*if (arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] &&*

*burst\_time[i] > 0) {*

*smallest = i;*

*}*

*}*

*burst\_time[smallest]--;*

*if (burst\_time[smallest] == 0) {*

*count++;*

*end = time + 1;*

*wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest];*

*turnaround\_time = turnaround\_time + end - arrival\_time[smallest];*

*}*

*}*

*printf("\nCalculated Waiting Time is :\t%lf\n", average\_waiting\_time);*

*printf("\nCalculated Turnaround Time is:\t%lf\n", average\_turnaround\_time);*

*average\_waiting\_time = wait\_time / limit;*

*average\_turnaround\_time = turnaround\_time / limit;*

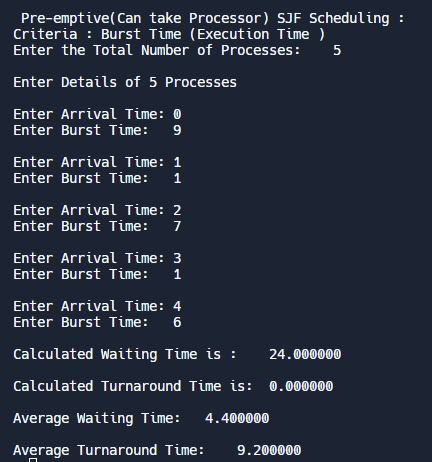
*printf("\nAverage Waiting Time:\t%lf\n", average\_waiting\_time);*

*printf("\nAverage Turnaround Time:\t%lf\n", average\_turnaround\_time);*

*return 0;*

*}*

**OUTPUT :**



**2. Non Pre-emptive SJF Scheduling**

*#include <stdio.h>*

*int main() {*

*int i, n, p[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, min, k = 1, btime = 0;*

*int bt[10], temp, j, at[10], wt[10], tt[10], ta = 0, sum = 0;*

*float wavg = 0, tavg = 0, tsum = 0, wsum = 0;*

*printf(" -------Shortest Job First Scheduling ( NP )-------\n");*

*printf("\nEnter the No. of processes :");*

*scanf("%d", &n);*

*for (i = 0; i < n; i++) {*

*printf("\tEnter the burst time of %d process :", i + 1);*

*scanf(" %d", &bt[i]);*

*printf("\tEnter the arrival time of %d process :", i + 1);*

*scanf(" %d", &at[i]);*

*}*

*/\*Sorting According to Arrival Time\*/*

*for (i = 0; i < n; i++) {*

*for (j = 0; j < n; j++) {*

*if (at[i] < at[j]) {*

*temp = p[j];*

*p[j] = p[i];*

*p[i] = temp;*

*temp = at[j];*

*at[j] = at[i];*

*at[i] = temp;*

*temp = bt[j];*

*bt[j] = bt[i];*

*bt[i] = temp;*

*}*

*}*

*}*

*/\*Arranging the table according to Burst time,*

*Execution time and Arrival Time*

*Arrival time <= Execution time*

*\*/*

*for (j = 0; j < n; j++) {*

*btime = btime + bt[j];*

*min = bt[k];*

*for (i = k; i < n; i++) {*

*if (btime >= at[i] && bt[i] < min) {*

*temp = p[k];*

*p[k] = p[i];*

*p[i] = temp;*

*temp = at[k];*

*at[k] = at[i];*

*at[i] = temp;*

*temp = bt[k];*

*bt[k] = bt[i];*

*bt[i] = temp;*

*}*

*}*

*k++;*

*}*

*wt[0] = 0;*

*for (i = 1; i < n; i++) {*

*sum = sum + bt[i - 1];*

*wt[i] = sum - at[i];*

*wsum = wsum + wt[i];*

*}*

*wavg = (wsum / n);*

*for (i = 0; i < n; i++) {*

*ta = ta + bt[i];*

*tt[i] = ta - at[i];*

*tsum = tsum + tt[i];*

*}*

*tavg = (tsum / n);*

*printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");*

*printf("\n RESULT:-");*

*printf("\nProcess\t Burst\t Arrival\t Waiting\t Turn-around");*

*for (i = 0; i < n; i++) {*

*printf("\n P%d\t\t \t %d\t\t %d\t\t\t %d\t\t\t%d", p[i], bt[i], at[i], wt[i],*

*tt[i]);*

*}*

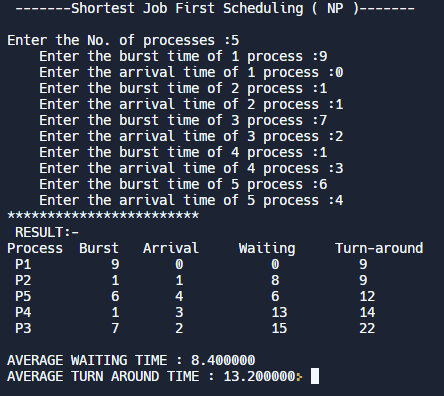
*printf("\n\nAVERAGE WAITING TIME : %f", wavg);*

*printf("\nAVERAGE TURN AROUND TIME : %f", tavg);*

*return 0;*

*}*

**OUTPUT :**



**Experiment No: 7**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date: 10/12/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Write a program to perform priority scheduling among a set of processes. |
| **Outcome:**  Student will understand the working of priority scheduling among a set of processes. |
| **Problem Statement:**  Implement the priority scheduling. |
| **Background Study:**  Priority scheduling is a non-preemptive algorithm and used in batch systems. Each process is assigned a priority. Process with highest priority is to be executed first and so on. |
| **Question Bank:**   1. What are advantages of Priority scheduling?   **Sol.Easy to use scheduling method. Processes are executed on the basis of priority so high priority does not need to wait for long which saves time.**   1. What are disadvantages of priority scheduling?   **Sol.The major disadvantage of priority scheduling is the process of indefinite blocking or starvation. This problem appears when a process is ready to be executed but it has to wait for the long time for execution by CPU because other high priority processes are executed by the CPU.**   1. At the ready queue when a process arrives In [priority scheduling](https://t4tutorials.com/priority-based-process-scheduling-in-operating-systems/) algorithm, the priority of this process is compared with the priority of? A. currently running process B. parent process C. all process **D. init process** 2. **Differentiate between pre-emptive and non pre-emptive scheduling**?   Sol.The basic difference between preemptive and non-preemptive scheduling is that in preemptive scheduling the CPU is allocated to the processes for the limited time. While in Non-preemptive scheduling, the CPU is allocated to the process till it terminates or switches to waiting state. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

*#include <stdio.h>*

*int main() {*

*int bt[20], p[20], wt[20], tat[20], pr[20], i, j, n, total = 0, pos, temp,*

*avg\_wt, avg\_tat;*

*printf("Enter Total Number of Process:");*

*scanf("%d", &n);*

*printf("\nEnter Burst Time and Priority\n");*

*for (i = 0; i < n; i++) {*

*printf("\nP[%d]\n", i + 1);*

*printf("Burst Time:");*

*scanf("%d", &bt[i]);*

*printf("Priority:");*

*scanf("%d", &pr[i]);*

*p[i] = i + 1; // contains process number*

*}*

*// sorting burst time, priority and process number in ascending order using*

*// selection sort*

*for (i = 0; i < n; i++) {*

*pos = i;*

*for (j = i + 1; j < n; j++) {*

*if (pr[j] < pr[pos])*

*pos = j;*

*}*

*temp = pr[i];*

*pr[i] = pr[pos];*

*pr[pos] = temp;*

*temp = bt[i];*

*bt[i] = bt[pos];*

*bt[pos] = temp;*

*temp = p[i];*

*p[i] = p[pos];*

*p[pos] = temp;*

*}*

*wt[0] = 0; // waiting time for first process is zero*

*// calculate waiting time*

*for (i = 1; i < n; i++) {*

*wt[i] = 0;*

*for (j = 0; j < i; j++)*

*wt[i] += bt[j];*

*total += wt[i];*

*}*

*avg\_wt = total / n; // average waiting time*

*total = 0;*

*printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");*

*for (i = 0; i < n; i++) {*

*tat[i] = bt[i] + wt[i]; // calculate turnaround time*

*total += tat[i];*

*printf("\nP[%d]\t\t\t %d\t\t %d\t\t\t%d", p[i], bt[i], wt[i], tat[i]);*

*}*

*avg\_tat = total / n; // average turnaround time*

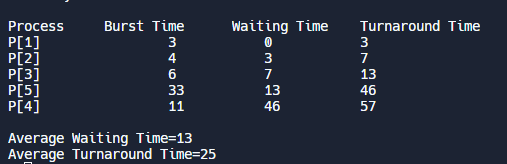
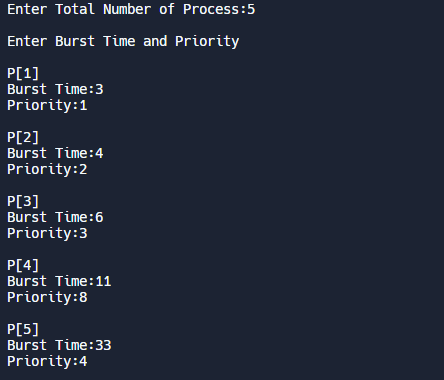
*printf("\n\nAverage Waiting Time=%d", avg\_wt);*

*printf("\nAverage Turnaround Time=%d\n", avg\_tat);*

*return 0;*

*}*

**OUTPUT :**



**Experiment No: 8**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date: 19/10/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective: Objective**  To familiarize the students about CPU scheduling Algorithms |
| **Program Outcome**  The students will understand the Round Robin Algorithm. |
| **Problem Statement:**  Implement the Round Robin Algorithm. |
| **Background Study:**   * In Round Robin each process is assigned a fixed time slot in a cyclic way and this is preemptive. It has a disadvantage of context switch and have quantum time |
| **Question Bank:**   1. What is Preemptive and Non- Preemptive CPU scheduling? Explain with examples.   Sol.Preemptive:  In this resources(CPU Cycle) are allocated to a process for a limited time.  Eg.Examples of preemptive scheduling are Round Robin and Shortest Remaining Time First.  Non-Preemptive  Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.  Eg.Examples of non-preemptive scheduling are First Come First Serve and Shortest Job First.   1. Explain the difference between short term, long term and medium term scheduling.   Sol.Long term:It selects processes from the pool and load them into memory for execution.  Medium term:Process can be reintroduced into the meat and its execution can be continued.  Short term:It selects from among the processes that are ready to execute.   1. Explain the function of Dispatcher and Context Switch mechanism.   Sol.Dispatcher is also responsible for:Context Switching, Switch to user mode, Jumping to proper location when process again restarted  The only work of scheduler is selection of processes.   1. What are the advantages and disadvantages of Round robin?   **Sol.Advantages –**  **Every process gets an equal share of the CPU.**  **RR is cyclic in nature, so there is no starvation.**  **Disadvantages –**  **Setting the quantum too short, increases the overhead and lowers the CPU efficiency, but setting it too long may cause poor response to short processes.**  **Average waiting time under the RR policy is often long.**   1. Give the application are of Robin Robin.   Sol.A popular use of the queue data structure is the scheduling problem in the operating system. Round-robin is one of the simplest scheduling algorithms for processes in an operating system, which assigns time slices to each process in equal portions and in order, handling all processes without priority. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

***// Use for loop to enter the details of the process like Arrival time and the***

***// Burst Time***

***for (i = 0; i < NOP; i++) {***

***printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i + 1);***

***printf(" Arrival time is: \t"); // Accept arrival time***

***scanf("%d", &at[i]);***

***printf(" \nBurst time is: \t"); // Accept the Burst time***

***scanf("%d", &bt[i]);***

***temp[i] = bt[i]; // store the burst time in temp array***

***}***

***// Accept the Time qunat***

***printf("Enter the Time Quantum for the process: \t");***

***scanf("%d", &quant);***

***// Display the process No, burst time, Turn Around Time and the waiting time***

***printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");***

***for (sum = 0, i = 0; y != 0;) {***

***if (temp[i] <= quant && temp[i] > 0) // define the conditions***

***{***

***sum = sum + temp[i];***

***temp[i] = 0;***

***count = 1;***

***} else if (temp[i] > 0) {***

***temp[i] = temp[i] - quant;***

***sum = sum + quant;***

***}***

***if (temp[i] == 0 && count == 1) {***

***y--; // decrement the process no.***

***printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i + 1, bt[i],***

***sum - at[i], sum - at[i] - bt[i]);***

***wt = wt + sum - at[i] - bt[i];***

***tat = tat + sum - at[i];***

***count = 0;***

***}***

***if (i == NOP - 1) {***

***i = 0;***

***} else if (at[i + 1] <= sum) {***

***i++;***

***} else {***

***i = 0;***

***}***

***}***

***// represents the average waiting time and Turn Around time***

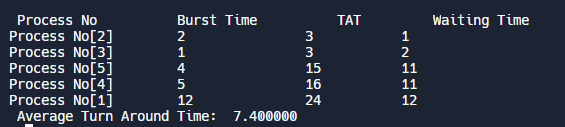
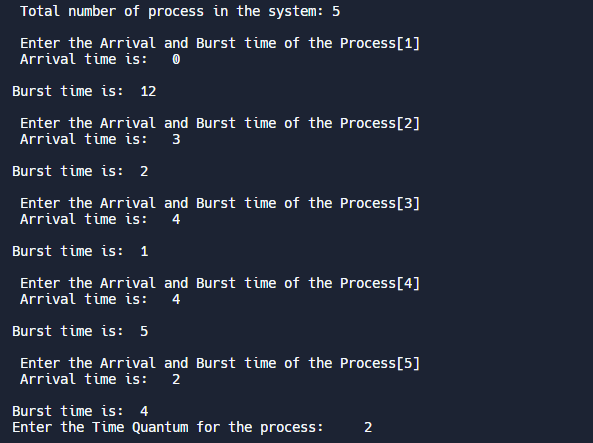
***avg\_wt = wt \* 1.0 / NOP;***

***avg\_tat = tat \* 1.0 / NOP;***

***printf("\n Average Turn Around Time: \t%f", avg\_wt);***

***printf("\n Average Waiting Time: \t%f", avg\_tat);***

***}***

****

**Experiment No: 9**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date:19/10/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:** Write a program to implement reader/writer problem using semaphore |
| **Program Outcome**  The students will understand the reader/writer problem using semaphore |
| **Problem Statement:**  Write a program to implement reader/writer problem using semaphore |
| **Background Study:** There is a shared resource which should be accessed by multiple processes. There are two types of processes in this context. They are reader and writer. Any number of readers can read from the shared resource simultaneously, but only one writer can write to the shared resource. When a writer is writing data to the resource, no other process can access the resource. A writer cannot write to the resource if there are non-zero number of readers accessing the resource at that time. |
| **Question Bank:**   1. An un-interruptible unit is known as \_\_\_\_\_\_\_\_\_\_\_\_ a) single **b) atomic** c) static d) none of the mentioned 2. TestAndSet instruction is executed \_\_\_\_\_\_\_\_\_\_\_\_ a) after a particular process b) periodically **c) atomically** d) none of the mentioned 3. Semaphore is a/an \_\_\_\_\_\_\_ to solve the critical section problem. a) hardware for a system b) special program for a system **c) integer variable** d) none of the mentioned 4. What are the two atomic operations permissible on semaphores? **a) wait** b) stop **c) hold** d) none of the mentioned 5. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place is called \_\_\_\_\_\_\_\_ a) dynamic condition **b) race condition** c) essential condition d) critical condition |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

***#include <pthread.h>***

***#include <semaphore.h>***

***#include <stdio.h>***

***sem\_t wrt;***

***pthread\_mutex\_t mutex;***

***int cnt = 1;***

***int numreader = 0;***

***void \*writer(void \*wno)***

***{***

***sem\_wait(&wrt);***

***cnt = cnt\*2;***

***printf("Writer %d modified count to %d\n",(\*((int \*)wno)),cnt);***

***sem\_post(&wrt);***

***}***

***void \*reader(void \*rno)***

***{***

***// Reader acquire the lock before modifying numreader***

***pthread\_mutex\_lock(&mutex);***

***numreader++;***

***if(numreader == 1) {***

***sem\_wait(&wrt); // If this id the first reader, then it will block the writer***

***}***

***pthread\_mutex\_unlock(&mutex);***

***// Reading Section***

***printf("Reader %d: Setting read count as %d\n",\*((int \*)rno),cnt);***

***// Reader acquire the lock before modifying numreader***

***pthread\_mutex\_lock(&mutex);***

***numreader--;***

***if(numreader == 0) {***

***sem\_post(&wrt); // If this is the last reader, it will wake up the writer.***

***}***

***pthread\_mutex\_unlock(&mutex);***

***}***

***int main()***

***{***

***int num\_of\_readers;***

***int num\_of\_writers;***

***printf("Enter Numbers of Readers : ");***

***scanf("%d",&num\_of\_readers);***

***printf("Enter Numbers of Writers : ");***

***scanf("%d",&num\_of\_writers);***

***pthread\_t read[num\_of\_readers],write[num\_of\_writers];***

***pthread\_mutex\_init(&mutex, NULL);***

***sem\_init(&wrt,0,1);***

***int a[10] = {1,2,3,4,5,6,7,8,9,10}; //Just used for numbering the producer and consumer***

***for(int i = 0; i < 10; i++) {***

***pthread\_create(&read[i], NULL, (void \*)reader, (void \*)&a[i]);***

***}***

***for(int i = 0; i < 5; i++) {***

***pthread\_create(&write[i], NULL, (void \*)writer, (void \*)&a[i]);***

***}***

***for(int i = 0; i < 10; i++) {***

***pthread\_join(read[i], NULL);***

***}***

***for(int i = 0; i < 5; i++) {***

***pthread\_join(write[i], NULL);***

***}***

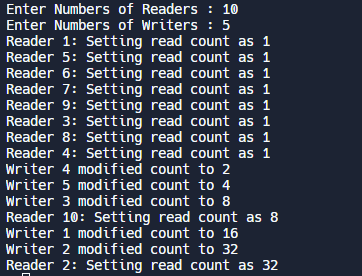
***pthread\_mutex\_destroy(&mutex);***

***sem\_destroy(&wrt);***

***return 0;***

***}***

**OUPUT :**

****

**Experiment No: 10**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date: 11/09/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Write a program to implement Dining Philosopher’s problem using semaphore |
| **Outcome:**  The students will understand the problem of synchronization among processes and its solution through Dining Philosopher’s problem using semaphore |
| **Problem Statement:**  Write a program to implement Dining Philosopher’s problem using semaphore |
| **Background Study:**  Five philosophers, spend their time thinking and eating spaghetti. They eat at a round table with five individual seats. For eating each philosopher needs two forks (the resources). There are five forks on the table, one left and one right of each seat. When a philosopher cannot grab both forks it sits and waits. Eating takes random time, then the philosopher puts the forks down and leaves the dining room. After spending some random time thinking he again becomes hungry, and the circle repeats itself. |
| **Question Bank:**   1. Which one of the following is a synchronization tool? a) thread b) pipe **c) semaphore** d) socket 2. A semaphore is a shared integer variable \_\_\_\_\_\_\_\_\_\_ **a) that can not drop below zero** b) that can not be more than zero c) that can not drop below one d) that can not be more than one 3. The bounded buffer problem is also known as \_\_\_\_\_\_\_\_\_\_\_\_ a) Readers – Writers problem b) Dining – Philosophers problem **c) Producer – Consumer problem** d) None of the mentioned 4. In the bounded buffer problem \_\_\_\_\_\_\_\_\_\_\_\_ a) there is only one buffer **b) there are n buffers ( n being greater than one but finite)** c) there are infinite buffers d) the buffer size is bounded 5. To ensure difficulties do not arise in the readers – writers problem \_\_\_\_\_\_\_ are given exclusive access to the shared object. a) readers **b) writers** c) readers and writers d) none of the mentioned |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

***#include<stdio.h>***

***#include<stdlib.h>***

***#include<pthread.h>***

***#include<semaphore.h>***

***#include<unistd.h>***

***sem\_t room;***

***sem\_t chopstick[5];***

***void \* philosopher(void \*);***

***void eat(int);***

***int main()***

***{***

***int i,a[5];***

***pthread\_t tid[5];***

***sem\_init(&room,0,4);***

***for(i=0;i<5;i++)***

***sem\_init(&chopstick[i],0,1);***

***for(i=0;i<5;i++){***

***a[i]=i;***

***pthread\_create(&tid[i],NULL,philosopher,(void \*)&a[i]);***

***}***

***for(i=0;i<5;i++)***

***pthread\_join(tid[i],NULL);***

***}***

***void \* philosopher(void \* num)***

***{***

***int phil=\*(int \*)num;***

***sem\_wait(&room);***

***printf("\nPhilosopher %d has entered room",phil);***

***sem\_wait(&chopstick[phil]);***

***sem\_wait(&chopstick[(phil+1)%5]);***

***eat(phil);***

***sleep(2);***

***printf("\nPhilosopher %d has finished eating",phil);***

***sem\_post(&chopstick[(phil+1)%5]);***

***sem\_post(&chopstick[phil]);***

***sem\_post(&room);***

***}***

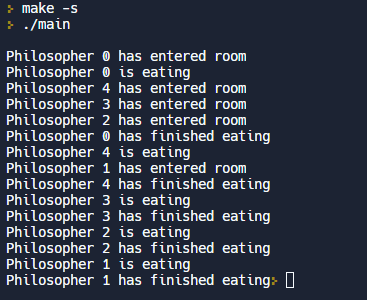
***void eat(int phil)***

***{***

***printf("\nPhilosopher %d is eating",phil);***

***}***

**OUTPUT :**

****

**Experiment No:11**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date: 11/09/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Write a program to implement Banker’s algorithm for deadlock avoidance. |
| **Outcome:**  The students will understand how system handles deadlock using Banker’s algorithm for deadlock avoidance |
| **Problem Statement:**  Write a program to implement Banker’s algorithm for deadlock avoidance. |
| **Background Study:**  The banker’s algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an “s-state” check to test for possible activities, before deciding whether allocation should be allowed to continue. Banker’s algorithm is named so because it is used in banking system to check whether loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders comes to withdraw their money then the bank can easily do it. |
| **Question Bank:**   1. Each request requires that the system consider the \_\_\_\_\_\_\_\_\_\_\_\_\_ to decide whether the current request can be satisfied or must wait to avoid a future possible deadlock. **a) resources currently available** b) processes that have previously been in the system c) resources currently allocated to each process d) future requests and releases of each process 2. Given a priori information about the \_\_\_\_\_\_\_\_ number of resources of each type that maybe requested for each process, it is possible to construct an algorithm that ensures that the system will never enter a deadlock state. a) minimum b) average **c) maximum** d) approximate 3. A deadlock avoidance algorithm dynamically examines the \_\_\_\_\_\_\_\_\_\_ to ensure that a circular wait condition can never exist. **a) resource allocation state** b) system storage state c) operating system d) resources 4. A state is safe, if \_\_\_\_\_\_\_\_\_\_\_\_ a) the system does not crash due to deadlock occurrence **b) the system can allocate resources to each process in some order and still avoid a deadlock** c) the state keeps the system protected and safe d) all of the mentioned 5. The two ways of aborting processes and eliminating deadlocks are \_\_\_\_\_\_\_\_\_\_\_\_ a) Abort all deadlocked processes b) Abort all processes **c) Abort one process at a time until the deadlock cycle is eliminated** d) All of the mentioned |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

***#include <stdio.h>***

***int current[5][5], maximum\_claim[5][5], available[5];***

***int allocation[5] = {0, 0, 0, 0, 0};***

***int maxres[5], running[5], safe = 0;***

***int counter = 0, i, j, exec, resources, processes, k = 1;***

***int main()***

***{***

***printf("\nEnter number of processes: ");***

***scanf("%d", &processes);***

***for (i = 0; i < processes; i++)***

***{***

***running[i] = 1;***

***counter++;***

***}***

***printf("\nEnter number of resources: ");***

***scanf("%d", &resources);***

***printf("\nEnter Claim Vector:");***

***for (i = 0; i < resources; i++)***

***{***

***scanf("%d", &maxres[i]);***

***}***

***printf("\nEnter Allocated Resource Table:\n");***

***for (i = 0; i < processes; i++)***

***{***

***for(j = 0; j < resources; j++)***

***{***

***scanf("%d", &current[i][j]);***

***}***

***}***

***printf("\nEnter Maximum Claim Table:\n");***

***for (i = 0; i < processes; i++)***

***{***

***for(j = 0; j < resources; j++)***

***{***

***scanf("%d", &maximum\_claim[i][j]);***

***}***

***}***

***printf("\nThe Claim Vector is: ");***

***for (i = 0; i < resources; i++)***

***{***

***printf("\t%d", maxres[i]);***

***}***

***printf("\nThe Allocated Resource Table:\n");***

***for (i = 0; i < processes; i++)***

***{***

***for (j = 0; j < resources; j++)***

***{***

***printf("\t%d", current[i][j]);***

***}***

***printf("\n");***

***}***

***printf("\nThe Maximum Claim Table:\n");***

***for (i = 0; i < processes; i++)***

***{***

***for (j = 0; j < resources; j++)***

***{***

***printf("\t%d", maximum\_claim[i][j]);***

***}***

***printf("\n");***

***}***

***for (i = 0; i < processes; i++)***

***{***

***for (j = 0; j < resources; j++)***

***{***

***allocation[j] += current[i][j];***

***}***

***}***

***printf("\nAllocated resources:");***

***for (i = 0; i < resources; i++)***

***{***

***printf("\t%d", allocation[i]);***

***}***

***for (i = 0; i < resources; i++)***

***{***

***available[i] = maxres[i] - allocation[i];***

***}***

***printf("\nAvailable resources:");***

***for (i = 0; i < resources; i++)***

***{***

***printf("\t%d", available[i]);***

***}***

***printf("\n");***

***while (counter != 0)***

***{***

***safe = 0;***

***for (i = 0; i < processes; i++)***

***{***

***if (running[i])***

***{***

***exec = 1;***

***for (j = 0; j < resources; j++)***

***{***

***if (maximum\_claim[i][j] - current[i][j] > available[j])***

***{***

***exec = 0;***

***break;***

***}***

***}***

***if (exec)***

***{***

***printf("\nProcess%d is executing\n", i + 1);***

***running[i] = 0;***

***counter--;***

***safe = 1;***

***for (j = 0; j < resources; j++)***

***{***

***available[j] += current[i][j];***

***}***

***break;***

***}***

***}***

***}***

***if (!safe)***

***{***

***printf("\nThe processes are in unsafe state.\n");***

***break;***

***}***

***else***

***{***

***printf("\nThe process is in safe state");***

***printf("\nAvailable vector:");***

***for (i = 0; i < resources; i++)***

***{***

***printf("\t%d", available[i]);***

***}***

***printf("\n");***

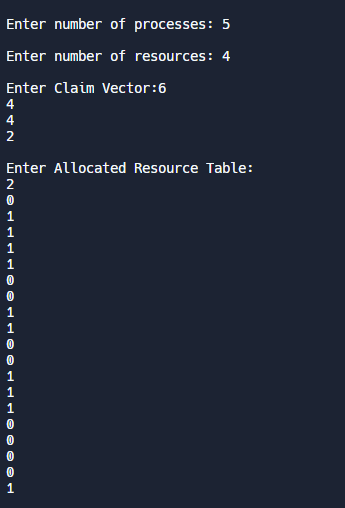
***}***

***}***

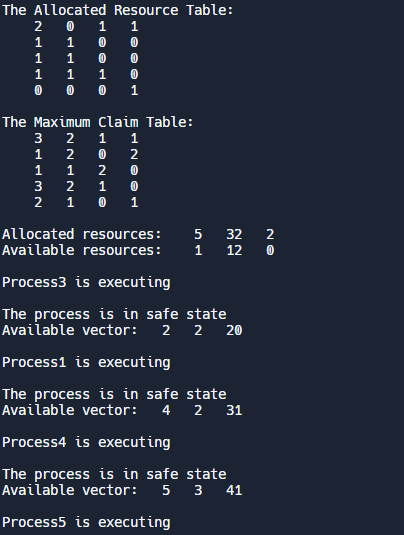
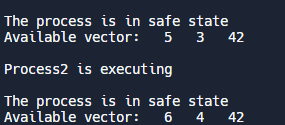
***return 0;***

***}***

**OUTPUT :**

****

****

** **

**Experiment No: 12**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh / 20csu241** |
| **Semester /Section: 5th / FSB** |
| **Date:16/09/2022** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Write a program for page replacement policy using a) LRU b) FIFO c) Optimal. |
| **Outcome:**  The students will understand concept of memory management through various page replacement policy using a) LRU b) FIFO |
| **Problem Statement:**  Write a program to implement page replacement policy using a) LRU b) FIFO c) Optimal. |
| **Background Study:**  In operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults. |
| **Question Bank:**  1. The main memory accommodates \_\_\_\_\_\_\_\_\_\_\_\_ **a) operating system** b) cpu c) user processes d) all of the mentioned  2. What is the operating system? a) in the low memory b) in the high memory **c) either low or high memory (depending on the location of interrupt vector)** d) none of the mentioned  3. In contiguous memory allocation \_\_\_\_\_\_\_\_\_\_\_\_ **a) each process is contained in a single contiguous section of memory** b) all processes are contained in a single contiguous section of memory c) the memory space is contiguous d) none of the mentioned  4. The relocation register helps in \_\_\_\_\_\_\_\_\_\_\_\_ a) providing more address space to processes b) a different address space to processes **c) to protect the address spaces of processes** d) none of the mentioned  5. With relocation and limit registers, each logical address must be \_\_\_\_\_\_\_ the limit register. **a) less than** b) equal to c) greater than d) none of the mentioned |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**LRU : Least Recently Used**

***#include <limits.h>***

***#include <stdio.h>***

***int checkHit(int incomingPage, int queue[], int occupied) {***

***for (int i = 0; i < occupied; i++) {***

***if (incomingPage == queue[i])***

***return 1;***

***}***

***return 0;***

***}***

***void printFrame(int queue[], int occupied) {***

***for (int i = 0; i < occupied; i++)***

***printf("%d\t\t\t", queue[i]);***

***}***

***int main() {***

***int temp;***

***printf("Enter Incoming String Size : ");***

***scanf("%d", &temp);***

***int incomingStream[temp];***

***printf("Enter Incoming String : \n");***

***for (int i = 0; i < temp; i++) {***

***printf("Enter string no. %d : ", i + 1);***

***scanf("%d", &incomingStream[i]);***

***}***

***int n = sizeof(incomingStream) / sizeof(incomingStream[0]);***

***int frames = 3;***

***int queue[n];***

***int distance[n];***

***int occupied = 0;***

***int pagefault = 0;***

***printf("Page\t Frame1 \t Frame2 \t Frame3\n");***

***for (int i = 0; i < n; i++) {***

***printf("%d: \t\t", incomingStream[i]);***

***// what if currently in frame 7***

***// next item that appears also 7***

***// didnt write condition for HIT***

***if (checkHit(incomingStream[i], queue, occupied)) {***

***printFrame(queue, occupied);***

***}***

***// filling when frame(s) is/are empty***

***else if (occupied < frames) {***

***queue[occupied] = incomingStream[i];***

***pagefault++;***

***occupied++;***

***printFrame(queue, occupied);***

***} else {***

***int max = INT\_MIN;***

***int index;***

***// get LRU distance for each item in frame***

***for (int j = 0; j < frames; j++) {***

***distance[j] = 0;***

***// traverse in reverse direction to find***

***// at what distance frame item occurred last***

***for (int k = i - 1; k >= 0; k--) {***

***++distance[j];***

***if (queue[j] == incomingStream[k])***

***break;***

***}***

***// find frame item with max distance for LRU***

***// also notes the index of frame item in queue***

***// which appears furthest(max distance)***

***if (distance[j] > max) {***

***max = distance[j];***

***index = j;***

***}***

***}***

***queue[index] = incomingStream[i];***

***printFrame(queue, occupied);***

***pagefault++;***

***}***

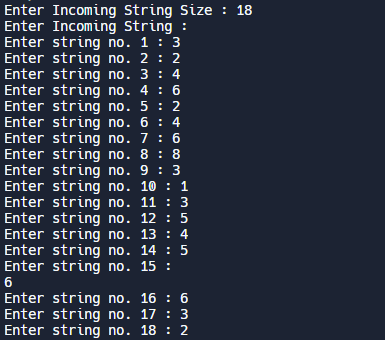
***printf("\n");***

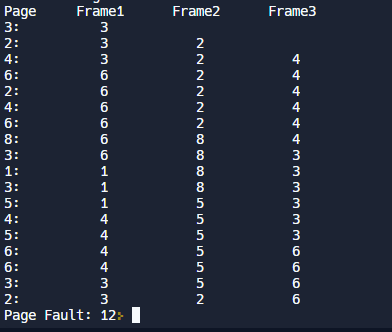
***}***

***printf("Page Fault: %d", pagefault);***

***return 0;***

***}***

**

**

***FIFO : First In First Out***

***#include <stdio.h>***

***int main() {***

***int temp;***

***printf("Enter Incoming String Size : ");***

***scanf("%d", &temp);***

***int incomingStream[temp];***

***printf("Enter Incoming String : \n");***

***for (int i = 0; i < temp; i++) {***

***printf("Enter string no. %d : ", i + 1);***

***scanf("%d", &incomingStream[i]);***

***}***

***int pageFaults = 0;***

***int frames = 3;***

***int m, n, s, pages;***

***pages = sizeof(incomingStream) / sizeof(incomingStream[0]);***

***printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");***

***int temp[frames];***

***for (m = 0; m < frames; m++) {***

***temp[m] = -1;***

***}***

***for (m = 0; m < pages; m++) {***

***s = 0;***

***for (n = 0; n < frames; n++) {***

***if (incomingStream[m] == temp[n]) {***

***s++;***

***pageFaults--;***

***}***

***}***

***pageFaults++;***

***if ((pageFaults <= frames) && (s == 0)) {***

***temp[m] = incomingStream[m];***

***} else if (s == 0) {***

***temp[(pageFaults - 1) % frames] = incomingStream[m];***

***}***

***printf("\n");***

***printf("%d\t\t\t", incomingStream[m]);***

***for (n = 0; n < frames; n++) {***

***if (temp[n] != -1)***

***printf(" %d\t\t\t", temp[n]);***

***else***

***printf(" - \t\t\t");***

***}***

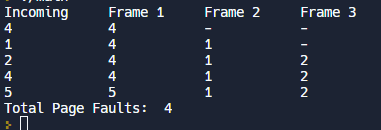
***}***

***printf("\nTotal Page Faults:\t%d\n", pageFaults);***

***return 0;***

***}***

***OUTPUT:***

******

**OPTIMAL PAGE REPLACEMENT :**

***#include <stdio.h>***

***int main() {***

***int no\_of\_frames, no\_of\_pages, frames[10], pages[30], temp[10], flag1, flag2,***

***flag3, i, j, k, pos, max, faults = 0;***

***printf("Enter number of frames: ");***

***scanf("%d", &no\_of\_frames);***

***printf("Enter number of pages: ");***

***scanf("%d", &no\_of\_pages);***

***printf("Enter page reference string: ");***

***for (i = 0; i < no\_of\_pages; ++i) {***

***scanf("%d", &pages[i]);***

***}***

***for (i = 0; i < no\_of\_frames; ++i) {***

***frames[i] = -1;***

***}***

***for (i = 0; i < no\_of\_pages; ++i) {***

***flag1 = flag2 = 0;***

***for (j = 0; j < no\_of\_frames; ++j) {***

***if (frames[j] == pages[i]) {***

***flag1 = flag2 = 1;***

***break;***

***}***

***}***

***if (flag1 == 0) {***

***for (j = 0; j < no\_of\_frames; ++j) {***

***if (frames[j] == -1) {***

***faults++;***

***frames[j] = pages[i];***

***flag2 = 1;***

***break;***

***}***

***}***

***}***

***if (flag2 == 0) {***

***flag3 = 0;***

***for (j = 0; j < no\_of\_frames; ++j) {***

***temp[j] = -1;***

***for (k = i + 1; k < no\_of\_pages; ++k) {***

***if (frames[j] == pages[k]) {***

***temp[j] = k;***

***break;***

***}***

***}***

***}***

***for (j = 0; j < no\_of\_frames; ++j) {***

***if (temp[j] == -1) {***

***pos = j;***

***flag3 = 1;***

***break;***

***}***

***}***

***if (flag3 == 0) {***

***max = temp[0];***

***pos = 0;***

***for (j = 1; j < no\_of\_frames; ++j) {***

***if (temp[j] > max) {***

***max = temp[j];***

***pos = j;***

***}***

***}***

***}***

***frames[pos] = pages[i];***

***faults++;***

***}***

***printf("\n");***

***for (j = 0; j < no\_of\_frames; ++j) {***

***if(frames[j]==-1){***

***printf(" ");***

***}else{***

***printf("%d\t", frames[j]);***

***}***

***}***

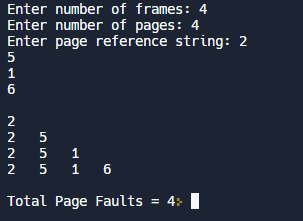
***}***

***printf("\n\nTotal Page Faults = %d", faults);***

***return 0;***

***}***

**OUTPUT :**

****