**Experiment No: 1**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar 19CSU185 |
| **Semester /Section:** V/FS-A-1 |
| **Link to Code:** **https://github.com/NamitKumar16/OS** |
| **Date:** 6th August 2021 |
| **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?** 2. How will you List files from a directory? 3. How files in a directory can be removed? 4. How to find out a word in a file? 5. What are wildcards? |

**Student Work Area**

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

**Q1.** Linux is a Unix-like, open source and community-developed operating system for computers, servers, mainframes, mobile devices and embedded devices. It is supported on almost every major computer platform including x86, ARM and SPARC, making it one of the most widely supported operating systems.

**Q2.** The ls command is used to list files or directories in Linux and other Unix-based operating systems.

**Q3.** Files can be removed using the rm command.

**Q4.** Grep is a Linux / Unix command-line tool used to search for a string of characters in a specified file. The text search pattern is called a regular expression. When it finds a match, it prints the line with the result.

**Q5.** A wildcard in Linux is a symbol or a set of symbols that stands in for other characters. It can be used to substitute for any other character or characters in a string. For example, you can use a wildcard to get a list of all files in a directory that begin with the letter O

**Screenshots**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A picture containing text

Description automatically generated

A picture containing text, screenshot

Description automatically generated

History Command to show all the commands used :-

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**EXPERIMENT NO. 2**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar |
| **Semester /Section:** V/FS-A-1 |
| **Link to Code:** **https://github.com/NamitKumar16/OS** |
| **Date:** 10 August, 21 |
| **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 are performed by shell script. So, we will use vi editor to edit our files. |
| **Question Bank:**   1. What is a shell? 2. What is the significance of $#? 3. What are the different types of commonly used shells on a typical Linux system? 4. How will you pass and access arguments to a script in Linux? 5. Use sed command to replace the content of the file (emulate tac command) |

**Student Work Area**

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

**Q1** Shell is a program that takes commands from the keyboard and gives them to the operating system to perform.

**Q2** $# shows the count of the arguments passed to the script.

**Q3** Bash, Zsh, Korn, Tcsh, Fish are commonly used shells

**Q4** 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.

**Q5** sed ‘1! G; h;$!d’ file1

Here G command appends to the pattern space,

h command copies pattern buffer to hold buffer

and d command deletes the current pattern space.

**Screenshots**

1. Write A Shell Program of Hello World

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Write a shell program to find factorial of a number.

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Write a shell program to find gross salary of an employee.

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

1. 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

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**EXPERIMENT NO. 3**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar |
| **Semester /Section:** V/FS-A-1 |
| **Link to Code:** **https://github.com/NamitKumar16/OS** |
| **Date:** 17 August,2021 |
| **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?  2. What is the difference between soft and hard links?  3. Explain about loops and what are the loops available in LINUX?  4. What are absolute and relative paths.  5. How to debug a shell script. |

**Student Work Area**

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

**Q1** Ans - In shell or bash shell, we can comment on multiple lines using << and name of comment.

**Q2** Ans - Hard Links cannot be used across file systems whereas soft links can be used across file systems.

**Q3** Ans - The while loop

The for loop

The until loop

The select loop

**Q4** Ans - An absolute path always contains the root element and the complete directory list required to

locate the file.

A relative path needs to be combined with another path in order to access a file.

1. Write a shell program to find largest of three numbers

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Write a shell program to find average of N numbers

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Find average of 3 numbers

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

**EXPERIMENT NO. 4**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar |
| **Semester /Section:** V/FS-A-1 |
| **Link to Code:** **https://github.com/NamitKumar16/OS** |
| **Date:** 24 August,2021 |
| **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 Zombie Process? 2. What are different types of variables used in shell script? 3. What are the different types of modes available in Vi editor? 4. What are the different types of permission at file level in shell? 5. How to use comments in shell script. |

**Student Work Area**

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

**Q1** Ans - A zombie process is a process whose execution is completed but it still has an entry in the

process table.

**Q2** Ans - System-Defined Variables.

User-Defined Variables.

**Q3** Ans - Command Mode , Entry Mode And Last Line Mode

**Q4** Ans - Execute Permission

Write Permission

Write and execute permission.

Read Permission

**Q5** Ans - By Using #

**Screenshots**

1. Write a shell program to check whether a number is even or odd

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Write a shell program to find whether a number is prime or not.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Write a shell program to find whether a number is palindrome or not.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

**Experiment No: 5**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar |
| **Semester /Section:** V/FS-A1 |
| **Link to Code:** **https://github.com/NamitKumar16/OS** |
| **Date:** 31st August 2021 |
| **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**

#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;

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%d\t\t%d\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;

}

A screenshot of a computer

Description automatically generated with medium confidence

**Experiment No: 6**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar |
| **Semester /Section:** V/FS -A1 |
| **Link to Code: https://github.com/NamitKumar16/OS** |
| **Date: 8**th September 2021 |
| **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**

**SJF Non - Preemptive**

#include<stdio.h>

int main()

{

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

float avg\_wt,avg\_tat;

printf("Enter number of process:");

scanf("%d",&n);

printf("\nEnter Burst Time:\n");

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

{

printf("p%d:",i+1);

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

p[i]=i+1;

}

//sorting of burst times

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

{

pos=i;

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

{

if(bt[j]<bt[pos])

pos=j;

}

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0;

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

{

wt[i]=0;

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

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=(float)total/n;

total=0;

printf("\nProcesst Burst Time \tWaiting Time\tTurnaround Time");

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

{

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

total+=tat[i];

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

}

avg\_tat=(float)total/n;

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

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

}

A screenshot of a computer

Description automatically generated with medium confidence

**SJF Preemptive**

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

scanf("%d", &limit);

printf("\nEnter Details of %d Processesn", 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];

}

}

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

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

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

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

return 0;

}

Text

Description automatically generated