

Department of CSE

LINUX PROGRAMMING PROBLEM STATEMENTS

| S. NO | Problem Statements |
| --- | --- |
| 1 | Write a shell script that receives any number of file names as arguments, checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.  *for x in $\**  *do*  *if [ -f $x ]*  *then*  *echo " $x is a file "*  *echo " no of lines in the file are "*  *wc -l $x*  *elif [ -d $x ]*  *then*  *echo " $x is a directory "*  *else*  *echo " enter valid filename or directory name "*  *fi*  *done*  Output:  guest-glcbIs@ubuntu:~$sh lprg4.sh dir1 d1  dir1 is a directory  d1 is a file  no of lines in the file are 2 |
| 2 | Write a shell script to display the sum of digits (123=1+2+3=6)  *echo "Enter a number"*  *read num*  *sum=0*  *while [ $num -gt 0 ]*  *do*  *mod=$((num % 10)) #It will split each digits*  *sum=$((sum + mod)) #Add each digit to sum*  *num=$((num / 10)) #divide the number by 10.*  *done*  *echo $sum* |
| 3 | Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it. |
| 4 | Write a shell script to check if the given number is a palindrome or not?  *read num*  *s=0*  *rev=""*  *temp=$num*    *while [ $num -gt 0 ]*  *do*  *# Get Remainder*  *s=$(( $num % 10 ))*  *# Get next digit*  *num=$(( $num / 10 ))*  *# Store previous number and*  *# current digit in reverse*  *rev=$( echo ${rev}${s} )*  *done*    *if [ $temp -eq $rev ];*  *then*  *echo "Number is palindrome"*  *else*  *echo "Number is NOT palindrome"*  *fi* |
| 5 | Write a shell script that displays a list of all the files in the current directory to which the user has read, write and execute permissions.  *for file in \**  *do*  *# check if it is a file*  *if [ -f $file ]*  *then*  *# check if it has all permissions*  *if [ -r $file -a -w $file -a -x $file ]*  *then*  *# print the complete file name with -l option*  *ls -l $file*  *# closing second if statement*  *fi*  *# closing first if statement*  *fi*  *done*  *Execute Code –*  *First Change Permissions*  *chmod 777 f1.sh* |
| 6 | Write a shell program to check whether the given argument is a file or directory  *for x in $\**  *do*  *echo "--------------------------------"*  *if [ -f $x ]*  *then*  *echo "$x is a file"*  *echo "File contents are "*  *cat $x*  *elif [ -d $x ]*  *then*  *echo "Directory contents are "*  *echo "$x is a directory"*  *ls $x*  *fi*  *done* |
| 7 | Write a shell program whether the given number is single or two or three or more than 3 print it as a more than 3 digit number. |
| 8 | Write a C program that makes a copy of a file using standard I/O and system calls.  *#include<stdio.h>*  *#include<fcntl.h>*  *#include<sys/types.h>*  *#include<unistd.h>*  *#define MAX\_SIZE 1000*  *int main()*  *{*  *int fd1,fd2,r1,w1;*  *char buffer[MAX\_SIZE];*  *char sourceName[100],destName[100];*  *printf("enter the source file\n");*  *scanf("%s",sourceName);*  *printf("enter a new file name");*  *scanf("%s",destName);*  *fd1=open(sourceName,O\_RDONLY);*  *r1=read(fd1,buffer,MAX\_SIZE);*  *fd2=open(destName,O\_CREAT|O\_RDWR,0600);*  *w1=write(fd2,buffer,r1);*  *return 0;*  *}* |
| 9 | Implement in C the following Unix commands using system calls.  a. Cat b.mv  *#include<fcntl.h>*  *#include<stdio.h>*  *#include<sys/types.h>*  *#include<unistd.h>*  *#define MAX\_SIZE 500*  *int main()*  *{*  *int fd1,n;*  *char buf[MAX\_SIZE],fname[20];*  *printf("enter a filename\n");*  *scanf("%s",fname);*  *fd1=open(fname,O\_RDONLY);*  *if(fd1==-1)*  *printf("the file does not exist");*  *else*  *{*  *printf("The contents of file %s are:\n",fname);*  *n=read(fd1,buf,MAX\_SIZE);*  *write(1,buf,n);*  *}*  *return 0;*  *}*  ***b.MV***  *#include<fcntl.h>*  *#include<sys/stat.h>*  *#include<stdio.h>*  *#include<unistd.h>*  *int main(int argc,char \*argv[])*  *{*  *open(argv[1],O\_RDONLY);*  *creat(argv[2],S\_IWUSR);*  *rename(argv[1],argv[2]);*  *unlink(argv[1]);*  *}* |
| 10 | Write an AWK script to count the number of lines in a file that do not contain vowels.  *echo 'enter a filename'*  *read fn*  *awk '$0 !~/[aeiou]/ { c=c+1 }*  *END { print("The no.of lines that do not contain vowels:",c) }' $fn* |
| 11 | Write an AWK script to find the number of characters, words and lines in a file.  *echo "enter a filename"*  *read fn*  *awk '{ w=w+NF*  *c=c+length($0)*  *}*  *END { print("No.of lines:",NR)*  *print("No.of words:",w)*  *print("No.of characters:",c)*  *}' $fn* |
| 12 | Write an AWK program to find the sum of individual digits of a number.  *BEGIN { printf "Enter the number: " getline n < "-" printf("\n") sum = 0; while ( n != 0) {     r = n % 10;     sum = sum + r;     n = int(n / 10);     } print "SUM=",sum }* |
| 13 | Write an AWK program to find the reverse of a number?  *BEGIN { printf "Enter the number: " getline n < "-" printf("\n") rev = 0; while ( n != 0) {     r = n % 10;     rev = (rev \* 10) + r;     n = int(n / 10);     } print "rev=",rev }* |
| 14 | Write an AWK script to find the average marks of a student |
| 15 | Write a C Program to list files in a directory.  *#include <stdio.h>*  *#include <dirent.h>*  *int main(void)*  *{*  *struct dirent \*de;*  *DIR \*dr = opendir(".");*  *if (dr == NULL)*  *{*  *printf("Could not open current directory" );*  *return 0;*  *}*  *while ((de = readdir(dr)) != NULL)*  *printf("%s\n", de->d\_name);*  *closedir(dr);*  *return 0;*  *}* |
| 16 | Write a C program to list for every file in a directory, its inode number and file name.  *#include<stdlib.h>*  *#include<stdio.h>*  *#include<string.h>*  *main(int argc, char \*argv[])*  *{*  *char d[50];*  *if(argc==2)*  *{*  *bzero(d,sizeof(d));*  *strcat(d,"ls ");*  *strcat(d,"-i ");*  *strcat(d,argv[1]);*  *system(d);*  *}*  *else*  *printf("\nInvalid No. of inputs");*  *}*  Execution Process-  *—>mkdir dd*  *—>cd dd*  *—>cat >f1*  *—>hello*  *—>^z*  *—>cd*  *—>gcc –o flist.out flist.c*  *—>./flist.out dd*  *—>hello*  *—>46490 f1* |
| 17 | Write a C program to create a child process and allow the parent to display “parent” and the child to display “child” on the screen.  *#include <stdio.h>*  *#include<fcntl.h>*  *#include <sys/wait.h>*  *int main(void)*  *{*  *int pid;*  *pid=fork( );*  *if(pid == -1)*  *{*  *perror("bad fork");*  *exit(1);*  *}*  *if (pid == 0)*  *printf("I am the child process.\n");*  *else {*  *printf("I am the parent process.\n");*  *}*  *}* |
| 18 | Write a C program to create a Zombie process.  *include <stdlib.h>*  *#include <sys/types.h>*  *#include <unistd.h>*  *int main()*  *{*  *pid\_t child\_pid = fork();*  *if (child\_pid > 0){*  *wait(0);*  *sleep(10);*  *printf("Parent process\n");*  *}*  *else{*  *printf("Child process\n");*  *exit(0);*  *}*  *return 0;*  *}* |
| 19 | Write a C program that illustrates how an orphan is created.  *#include <stdlib.h>*  *#include <sys/types.h>*  *#include <unistd.h>*  *int main()*  *{*  *pid\_t child\_pid = fork();*  *if (child\_pid > 0){*  *printf("Parent process\n");*  *}*  *else{*  *sleep(0.1);*  *printf("Child process\n");*  *exit(0);*  *}*  *return 0;*  *}* |
| 20 | Write C programs that illustrate communication between two unrelated processes using a named pipe(FIFO).  *#include<stdio.h>*  *#include<stdlib.h>*  *#include<errno.h>*  *#include<unistd.h>*  *int main()*  *{*  *int pfds[2];*  *char buf[30];*  *if(pipe(pfds)==-1)*  *{*  *perror("pipe");*  *exit(1);*  *}*  *printf("writing to file descriptor #%d\n", pfds[1]);*  *write(pfds[1],"test",5);*  *printf("reading from file descriptor #%d\n ", pfds[0]);*  *read(pfds[0],buf,5);*  *printf("read\"%s\"\n" ,buf);*  *}* |
| 21 | Write a C program in which a parent writes a message to a pipe and the child reads the message.  *#include<stdio.h>*  *#include<fcntl.h>*  *#include<stdlib.h>*  *#include<string.h>*  *#include<unistd.h>*  *#include<ctype.h>*  *int main()*  *{*  *int fd[2],n,pid,count=0;*  *char str[]="welcome to lp lab";*  *char buff[84];*  *pipe(fd);*  *pid=fork();*  *if(pid==-1)*  *{*  *printf("CHILD PROCESS CREATION ERROR \n");*  *exit(0);*  *}*  *else if(pid>0)*  *{*  *close(fd[0]);*  *write(fd[1],str,strlen(str));*  *}*  *else if(pid==0)*  *{*  *close(fd[1]);*  *n=read(fd[0],buff,sizeof(buff));*  *while(count<=n)*  *{*  *buff[count]=toupper(buff[count]);*  *count++;*  *}*  *printf("THE UPPER CASE DATA IS %s\n",buff);*  *}*  *return 0;*  *}* |