**CPP Lab**

**Assignment-3**

**Name-Ashish Goyal**

**Id-2016ucp1100**

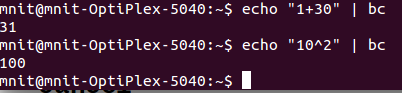
**Batch-A (1, 2)**

1. **Perform these linux commands:**

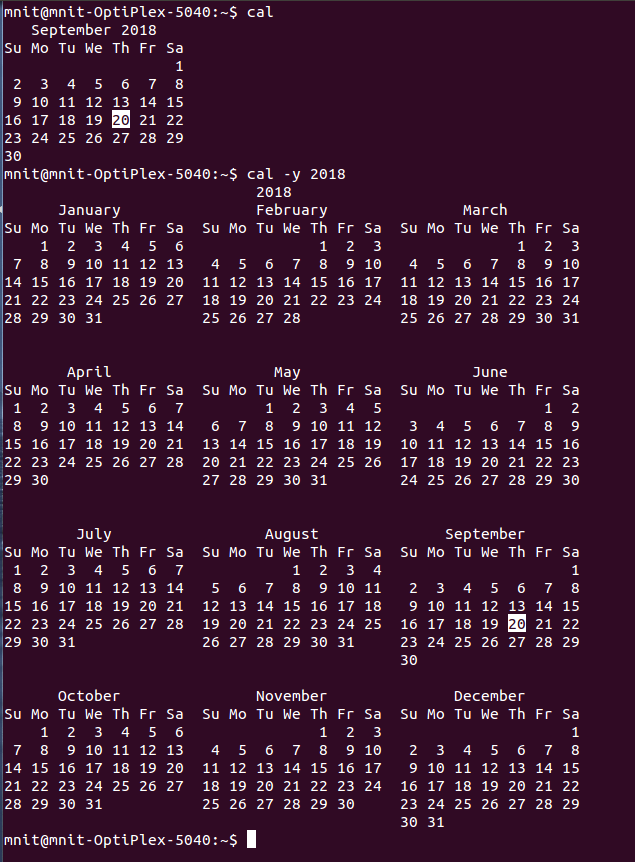
* **Date**: - **date command** is used to display the system **date** and time. **date command** is also used to set **date** and time of the system.

2date.png

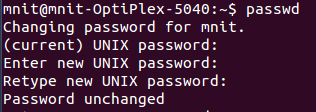
* **Echo**: - **echo** is widely used built-in command for Linux bash that typically used in scripting language and batch files to display a line of text/string on standard output or a file.3echo.png
* **Bc**: - **bc** command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.



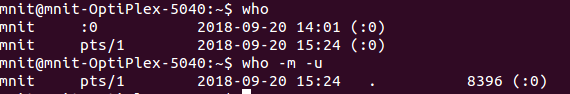
* **Cal**: - The **cal** command displays a simple, formatted calendar in the terminal.



* **Passwd**: - The **passwd command** is used to change the**password** of a user account. A normal user can run**passwd** to change their own **password**, and a system administrator (the superuser) can use **passwd**to change another user's **password**, or define how that account's **password** can be used or changed.

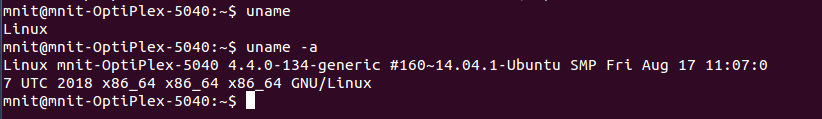


* **Who**: - **who** command is used to find out the following information :  
  1. Time of last system boot  
  2. Current run level of the system  
  3. List of logged in users and more.



* **Uname**: - Print information about the current system.

-a, --all Prints all information, omitting –p and –I if the information unknown



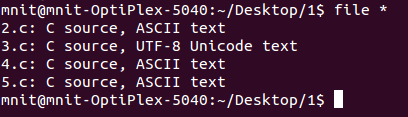
* **Tty**: - The tty command basically prints the file name of the terminal connected to standard input.

8tty.png

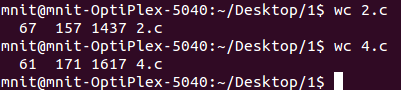
* **Cancel**: - *cancel* cancels existing print jobs. The *-a* option will remove all jobs from the specified destination.

9cancel.png

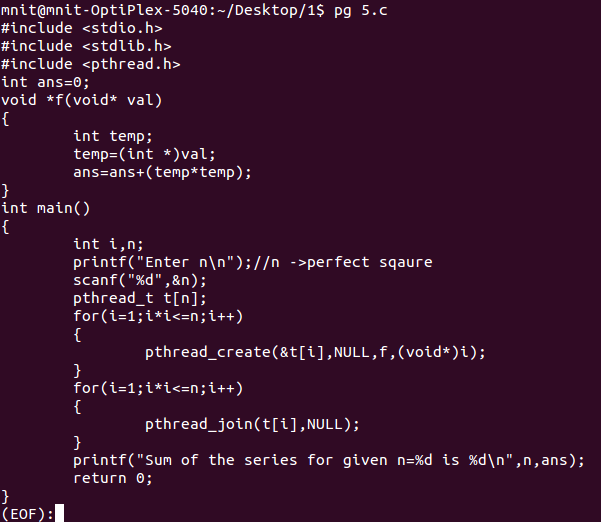
* **file**: - The **file** command is used to determine a file's type.



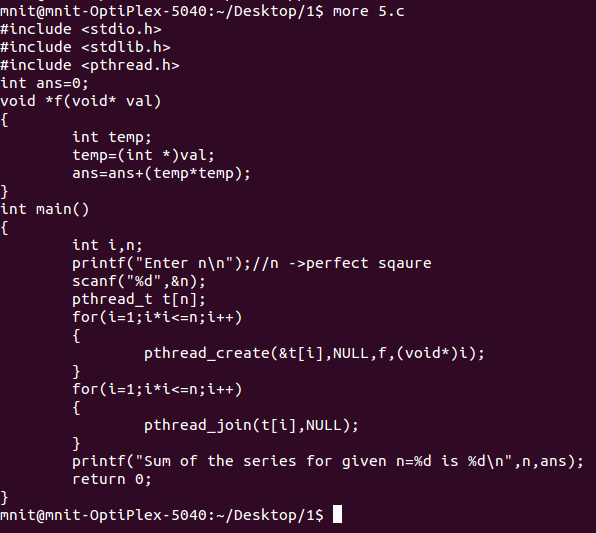
* **Wc**: - wc stands for **word count**. As the name implies, it is mainly used for counting purpose.It is used to find out **number of lines**, **word count**, **byte and characters count** in the files specified in the file arguments.By default it displays **four-columnar output.**First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.



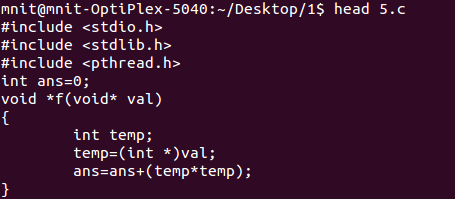
* **Pg**: - **pg** is a terminal pager program on Unix and Unix-like systems for viewing text files.



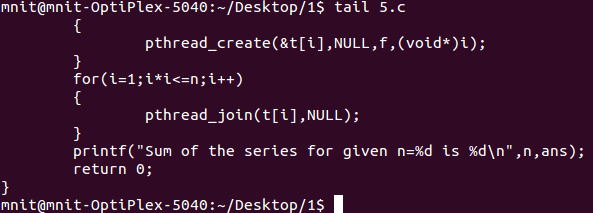
* **More**: - The **Linux more command** lets you view text files or other output in a scrollable manner. It displays the text one screenful at a time, and lets you scroll backwards and forwards through the text, and even lets you search the text.



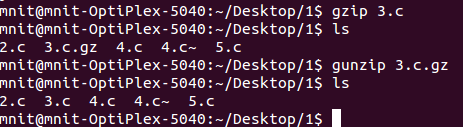
* **Head**: - The head command, as the name implies, print the top N number of data of the given input. By default it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is precedes by its file name.



* **Tail**: - The **tail command** is a **command**-line utility for outputting the last part of files given to it via standard input. It writes results to standard output. By default**tail** returns the last ten lines of each file that it is given.



* **Gzip**: - **Gzip** (GNU zip) is a compressing tool, which is used to truncate the file size. By default original file will be replaced by the compressed file ending with extension (.gz).
* **Gunzip**: - gunzip can currently decompress files created by gzip, zip, compress, compress -H or pack.



* **Chmod**: - **chmod** is used to change the [permissions](https://www.computerhope.com/jargon/p/permissi.htm) of [files](https://www.computerhope.com/jargon/f/file.htm) or [directories](https://www.computerhope.com/jargon/d/director.htm).

Here the digits **7**, **5**, and **4** each individually represent the permissions for the user, group, and others, in that order. Each digit is a combination of the numbers **4**, **2**, **1**, and **0**:

**0** stands for "no permission."

**1** stands for "execute"

**2** stands for "write",

**4** stands for "read",

**17chmod.png**

1. **Write a program for matrix multiplication using pthread\_join(). No mutex should be used.**

**CODE-**

#include<stdio.h>

#include<stdlib.h>//for rand();

#include<pthread.h>

#define size 4

#define size\_THREAD 4

int arrA[size][size];

int arrB[size][size];

int arrC[size][size];

int count = 0;

void\* matrix(void\* arg)

{

int concurrent,i,j,k;

concurrent= count++;

for(i=concurrent\*size/4;i<(concurrent+1)\*size/4;i++)

{

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

{

for(k=0;k<size;k++)

{

arrC[i][j]+=arrA[i][k]\*arrB[k][j];

}

}

}

}

int main()

{

int i,j,k;

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

{

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

{

arrA[i][j]=rand()%2;

arrB[i][j]=rand()%2;

}

}

printf("Matrix A\n");

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

{

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

printf("%d ",arrA[i][j]);

printf("\n");

}

printf("\nMatrix B\n");

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

{

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

printf("%d ",arrB[i][j]);

printf("\n");

}

pthread\_t threads[size\_THREAD];

for (i = 0; i < size\_THREAD; i++)

{

int\* p;

pthread\_create(&threads[i], NULL, matrix, (void\*)(p));

}

for (i = 0; i < size\_THREAD; i++)

pthread\_join(threads[i], NULL);

printf("\nMultiplication of A and B\n");

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

{

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

printf("%d ",arrC[i][j]);

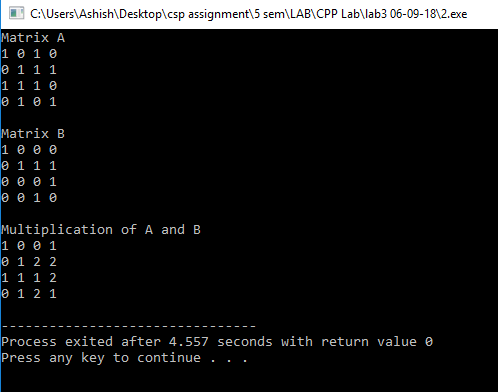
printf("\n");

}

return 0;

}

**OUTPUT-**



1. **Write a C program in which a filename is passed as a command line argument. In case a wrong name or no file name is passed is passed using CLA it should print an error message using perror().**

**CODE-**

#include <stdio.h>

#include <error.h>

#include <stdlib.h>

int main(int argc, char \*argv[])

{

FILE \*fp;

if (argc == 1)

{

puts("File Argument Missing");

exit(EXIT\_FAILURE);

}

fp = fopen(argv[1],"r");

// perror() simplifies reporting of such specific errors to user. Itâ€™s prototype is as follows

// prototype:: void perror(char const \*message);

if (fp == NULL)

{

perror("File Not Found");

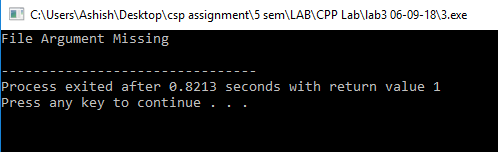
exit(EXIT\_FAILURE);

}

return 0;

}

**OUTPUT-**



1. **Write a C program that continuously print(with a sleep time of 1 second) the process id and the total sleep time. This program should send a SIGINT signal using signal() call, when a particular key is pressed.**

**CODE-**

#include<signal.h>

#include<stdio.h>

#include<unistd.h>

void qwerty(int key)

{

printf("\nkey pressed: %d\n",key);

(void)signal(SIGINT,SIG\_DFL);

}

int main()

{

(void)signal(SIGINT,qwerty);

pid\_t p;

int c=0;

while(1)

{

printf("process id is %d\n",p);

printf("sleep time: %d\n",c);

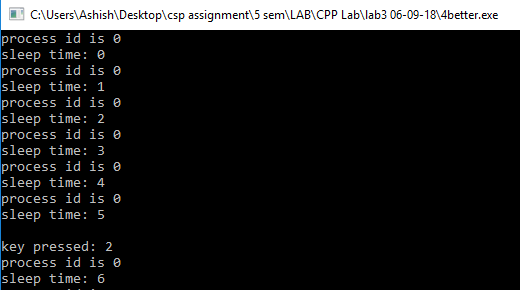
c++;

sleep(1);

}

}

**OUTPUT-**

****

1. **Write a program using pthread to find out the sum of following series:1+4+9+16+.......+ n. Here main function should write the final output on screen. Main thread will create the child threads and child threads will find out the sum of series.**

**CODE-**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int ans=0;

void \*f(void\* val)

{

int temp;

temp=(int \*)val;

ans=ans+(temp\*temp);

}

int main()

{

int i,n;

printf("Enter n\n");//n ->perfect sqaure

scanf("%d",&n);

pthread\_t t[n];

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

{

pthread\_create(&t[i],NULL,f,(void\*)i);

}

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

{

pthread\_join(t[i],NULL);

}

printf("Sum of the series for given n=%d is %d\n",n,ans);

return 0;

}

**OUTPUT-**

