1. ls

mkdir threads

cd threads

nano thread.c

**Write following text in nano window.**

//basic creation os thread

#include<stdio.h>

#include<pthread.h>

#include<time.h>

void \* show(void \* u){

printf("New thread\n");

}

int main(){

pthread\_t tid;

time\_t t;

time(&t);

printf("\nThis program has been writeen at (date and time): %s", ctime(&t));

pthread\_create(&tid,NULL,&show,NULL);

printf("Main thread\n");

pthread\_join(tid,NULL);

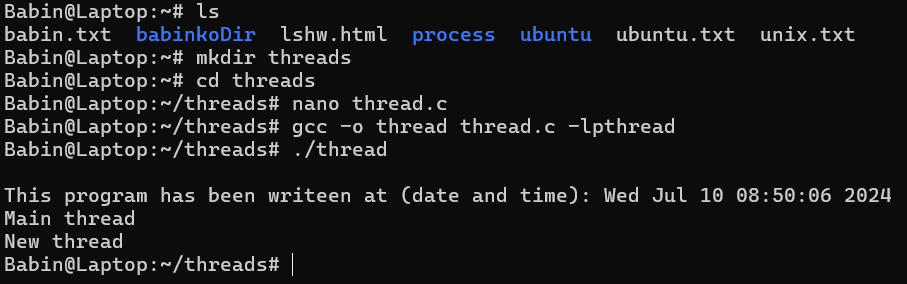
return 0;

}

**Interpretation:**

Creating new thread using pthread\_create() function and executing a user-defined

function show().



**2. nano thread\_two.c**

Write following text in nano

//contains two threads

//both calling different function but modifying same global variable

#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#include <unistd.h>

#include<time.h>

int value=1;

void \* sleep\_a(void \* u){

printf("New boy\n");

value=value+5;

printf("\nI am going to sleep for %d seconds: ",value);

sleep(value);

printf("\nI slept for %d seconds: \n",value);

}

void \* sleep\_b(void \*u){

printf("old boy\n");

value=value+2;

printf("\nI am going to sleep for %d seconds: ",value);

sleep(value);

printf("\nI slept for %d seconds: \n",value);

}

int main(){

pthread\_t tid,tid2;

time\_t t;

time(&t);

printf("\nThis program has been writeen at (date and time): %s", ctime(&t));

pthread\_create(&tid,NULL,&sleep\_a,NULL);

pthread\_create(&tid2,NULL,&sleep\_b,NULL);

printf("Main thread\n");

pthread\_join(tid,NULL);

pthread\_join(tid2,NULL);

return 0;

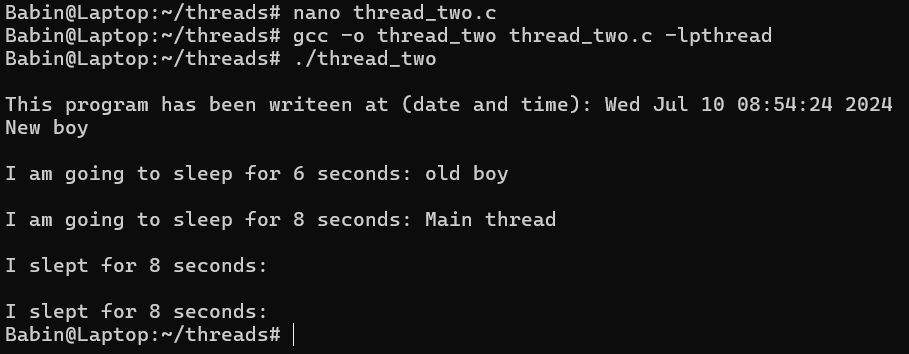
}

**Interpretation:**

The pthread\_join() function waits for the thread specified by thread to terminate.

If that thread has already terminated, then pthread\_join() returns immediately.

**Output:**

****

1. **Nano thread\_process.c**

#include<stdio.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>

#include<time.h>

void \* show(void \* u){

int pid;

printf("OLD BOY\n");

printf("\nThis is thread its pid is no. %d\n",getpid());

}

int main(){

int pid;

pthread\_t tid;

pthread\_t tid\_child;

time\_t t;

time(&t);

printf("\nThis program has been writeen at (date and time): %s", ctime(&t));

pthread\_create(&tid,NULL,&show,NULL);

printf("Main thread\n");

printf("\nThe pid of main thread is %d: \n",getpid());

printf("\nThe ppid of main thread is %d: \n",getppid());

pid=fork();

if(pid==0){

printf("\nThis is child\n");

printf("\n The id of child is %d\n",getpid());

printf("\n My parent is %d: ",getppid());

exit(0);

}

pthread\_join(tid,NULL);

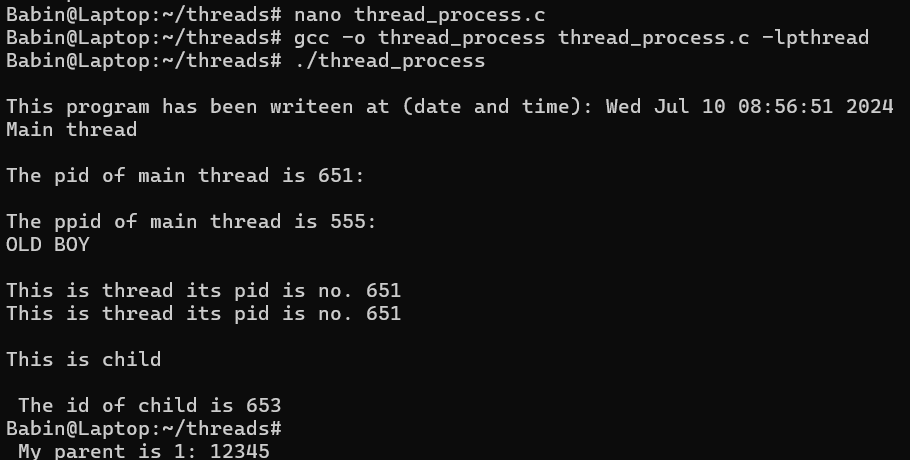
return 0;

}

**Interpretation:**

Creating thread and creating its child process

**Output:**

****

1. **Nano thread\_argument.c**

#include <pthread.h>

#include <stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<time.h>

#include<stdlib.h>

int global;

void square(){

global = global\*global;

}

void cube(){

global = global\*global\*global;

}

void \*PrintHello(void \*choice) {

int \*id\_ptr, taskid;

sleep(1);

id\_ptr = (int \*) choice;

taskid = \*id\_ptr;

printf("\nEnter a number: ");

scanf("%d",&global);

if(taskid==0){

square();

printf("\nthe square is %d: \n",global);

}

else if(taskid==1){

cube();

printf("\nthe cube is %d: \n",global);

}

pthread\_exit(NULL);

}

int main( ) {

pthread\_t threads;

int \*choice=malloc(sizeof(int\*));

int rc;

time\_t t;

time(&t);

printf("\nThis program has been writeen at (date and time): %s", ctime(&t));

printf("\n Enter a choice: 0 for square and 1 for cube number: ");

scanf("%d",choice);

printf("Creating thread \n");

rc = pthread\_create(&threads, NULL, PrintHello, (void \*)choice);

if (rc) {

printf("ERROR; return code from pthread\_create() is %d\n", rc);

exit(-1);

}

pthread\_exit(NULL);

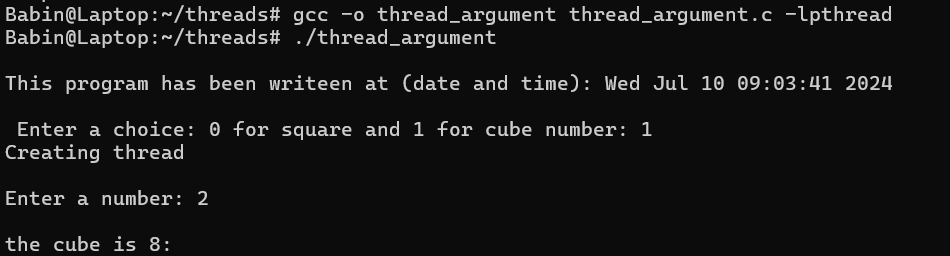
}

**Interpretation:**

Demonstration of creating thread by passing function and its argument as parameter

to the pthread\_create() function.

**Output**

****