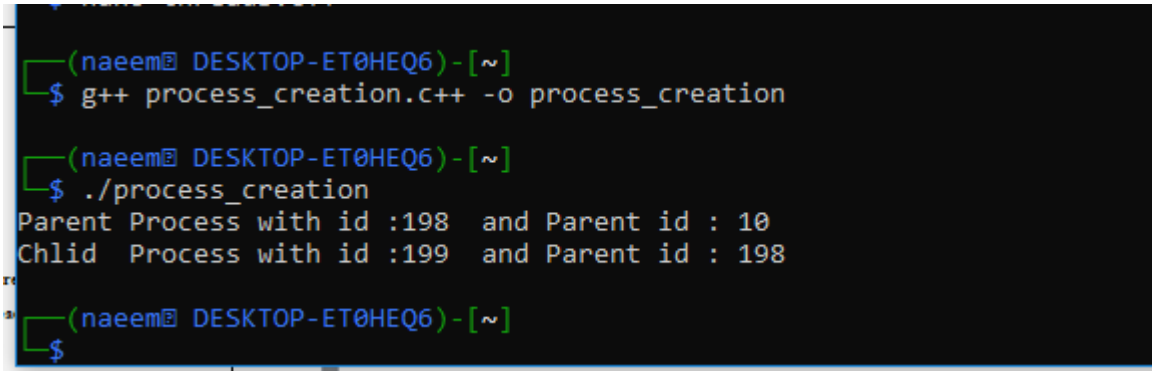
Creating the process in C++:

Code:

```
#include<iostream>

#include<unistd.h>

#include<sys/wait.h>

using namespace std;

int main(){

	int id;

	id= fork (); //create a process

	if(id<0){   // Error creating process

	cout<<"Can't create a file (Error)"<<endl;

	exit(-1);

	}

	if (id!=0){

		cout<<"Parent Process with id :"<<getpid()<<"  and Parent id :
"<<getppid()<<endl;

	}else{

		cout<<"Chlid  Process with id :"<<getpid()<<"  and Parent id :
"<<getppid()<<endl;

	}


return  0;

}
```

Output:



Creating the n child process in C++:

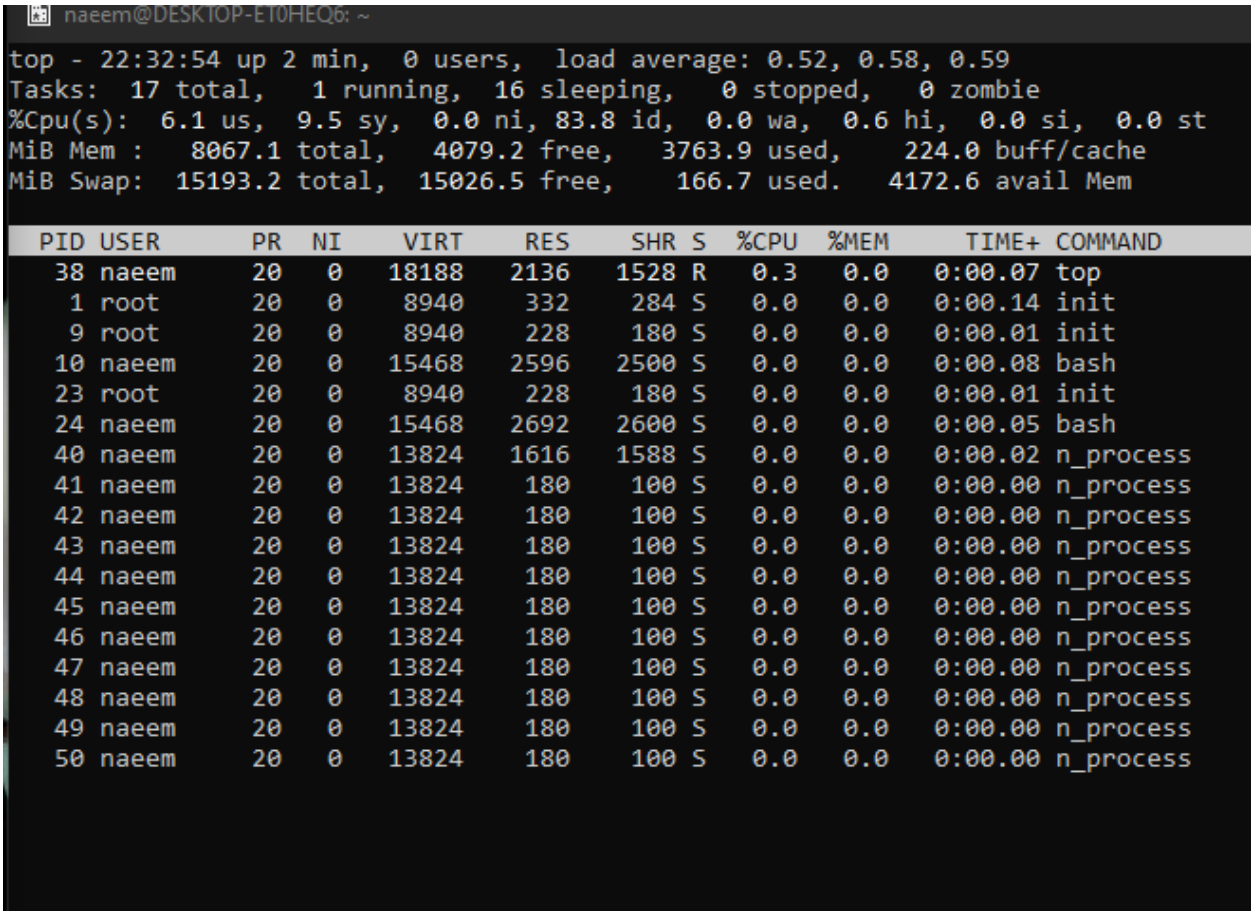
Code:

```
#include <iostream>
#include <unistd.h>
#include <sys/wait.h>
using namespace std;
int main(){
int n;
cout <<"Enter the no Child process : ";cin>>n;
int id;
for(int j =1;j<=n;j++){
      if(fork()==0){
      cout<<"Child process with id :"<<getpid()<<" with parent id : "<<getppid()<<endl;
      exit(0);} }
wait(NULL);
return 0;
}
```

Output:

```
┌──(naeem㉿ DESKTOP-ET0HEQ6)-[~]
└─$ nano n_process.c++

┌──(naeem㉿ DESKTOP-ET0HEQ6)-[~]
└─$ g++ n_process.c++ -o n_process

┌──(naeem㉿ DESKTOP-ET0HEQ6)-[~]
└─$ ./n_process
Enter the no Child process : 4
Child process with id :208 with parent id : 207
Child process with id :209 with parent id : 207
Child process with id :210 with parent id : 207
Child process with id :211 with parent id : 207

┌──(naeem㉿ DESKTOP-ET0HEQ6)-[~]
└─$
```

Show in Linux (Top):



Creating the threads in C++:

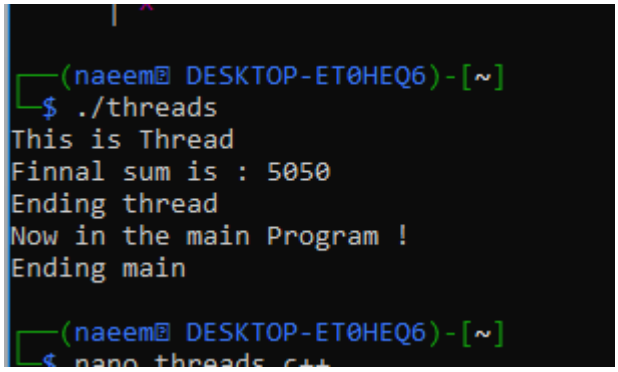
In this program thread is created and sum of first 100 integers are calculated in its function.

Code:

```
#include<iostream>
#include<unistd.h>
#include<pthread.h>
using namespace std;
void *thread_fun(void *arg);
int main(){
pthread_t mythread; //declare thread
pthread_create(&mythread,NULL,thread_fun,NULL); //create thread
pthread_join(mythread,NULL); //wait for completing thread
cout<<"Now in the main Program !"<<endl;
sleep(1);
cout<<"Ending main"<<endl;
sleep(1);
return 0;
}
void *thread_fun(void *arg){


cout<<"This is Thread"<<endl;
sleep(1);
 int sum=0;
      for (int i=1;i<=100;i++){
```

```
    sum+=i;

}

cout<<"Finnal sum is : "<<sum<<endl;

cout<<"Ending thread"<<endl;

}
```

Output:



To Create a Threads in LINUX and Assign the cores to threads:

In this program 2 threads are created and write the CPU intensive Function that is performed by the threads. Then we assign the different cores to the threads.

Thread 1 to logical processor 1

Thread 2 to logical Processor 2

Script:

```
// thread_core_assign.c++ *(Program file name)

#define _GNU_SOURCE

#include<iostream>

#include<pthread.h>

#include<math.h>

#include<stdlib.h>
```

```
#include<unistd.h>
#include<errno.h>
using namespace std;


#define handle_error_en(en,msg)\
        do {errno = en; perror(msg);exit(EXIT_FAILURE);} while (0);


void *threadFunction(void *args){    // Declare the Time taking function  for the Thread.
        float x = 1.5f;
        while(1){
                x *= sin(x)/atan(x)*tanh(x)*sqrt(x);
        }
}
int main(){
        cpu_set_t cpuset1;
        CPU_ZERO(&cpuset1);
        CPU_SET(1,&cpuset1);
        cpu_set_t cpuset2;
        CPU_ZERO(&cpuset2);
        CPU_SET(2,&cpuset2);
        pthread_t t1;
        pthread_t t2;
        int t1_op = pthread_create(&t1,NULL,threadFunction,NULL);
        int t2_op = pthread_create(&t2,NULL,threadFunction,NULL);
        int s1 = pthread_setaffinity_np(t1,sizeof(cpu_set_t),&cpuset1);
```

```
    if(s1 != 0){
    handle_error_en (s1,"pthread_setaffinity_np,s1");
    }
    int s2 = pthread_setaffinity_np(t1,sizeof(cpu_set_t),&cpuset2);
    if(s2 != 0){
    handle_error_en (s2,"pthread_setaffinity_np,s2");
  }
    pthread_join(t1,NULL);
    pthread_join(t2,NULL);
return 0;
}
```

Assign the Core to the Threads (Htop_preview):