**NAME – SINDHU SINGH**

**ROLL NUM - 20CS8066**

**OS LAB ASSIGNMETN 4**

----------------------------------------------------------------\*\*\*\*\*\*\*\*---------------------------------------------------

1. Time taken for thread and Process creation

2.#include<bits/stdc++.h>   
3.#include<sys/types.h>   
4.#include<sys/wait.h>   
5.#include<unistd.h>   
6.#include<fstream>   
7.#include<chrono>   
8.#include<pthread.h>   
9.usingnamespacestd;   
10.

11.void\*f(void\*arg){pthread\_exit(NULL);} //////THREAD FUNCTION 12.

13.intmain(){   
14.intstat,rc;   
15. // thread creation   
16.

17.autostart = chrono::high\_resolution\_clock::now(); //Clock to measure time.

18.vector<pthread\_t> tid(20000); //vector of thread id.

19.for(inti=0;i< 20000;i++){   
20.rc = pthread\_create(&tid[i], NULL, f, NULL);   
21.if (rc) {   
22.cout<<"Error:unable to create thread,"<<rc<<endl; 23.exit(-1);   
24. }   
25. }   
26.for(inti = 0; i < 20000; i++){   
27.pthread\_join(tid[i], NULL);   
28. }   
29.autoend = chrono::high\_resolution\_clock::now();

30.doublet = chrono::duration\_cast<chrono::nanoseconds>(end- start).count();   
31.t \*= 1e-6 ; // time taken for thread creation   
32.cout<<"Time taken to create thread : "<<t<<"ms"<<endl; 33.

34.

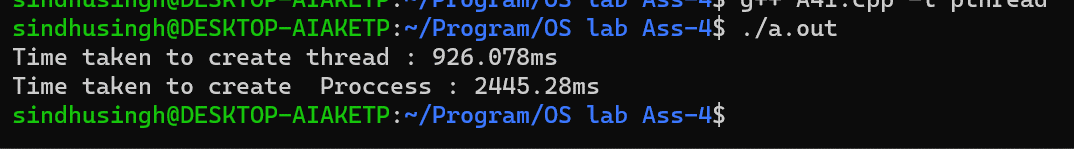
35. // Proccess creation ...

36.start=chrono::high\_resolution\_clock::now();   
37.for(inti=0; i<100000; i++){   
38.intid = fork();   
39.if(id == 0) exit(0);   
40. }   
41.while(wait(&stat)>0);   
42.end=chrono::high\_resolution\_clock::now();   
43.t = chrono::duration\_cast<chrono::nanoseconds>(end- start).count(); // time taken for procces creation.

44.t \*= 1e-6 ;   
45.cout<<"Time taken to create Proccess : "<<t<<"ms"<<endl; 46.

47.return0;   
48.}

Output -



2.

|  |
| --- |
| //#define \_GNU\_SOURCE |

|  |
| --- |
| #include<sched.h> |

|  |
| --- |
| #include<sys/types.h> |

|  |
| --- |
| #include<sys/wait.h> |

|  |
| --- |
| #include<unistd.h> |

|  |
| --- |
| #include<stdlib.h> |

|  |
| --- |
| #include<iostream> |

|  |
| --- |
| #include<math.h> |

|  |
| --- |
| #include<vector> |

|  |
| --- |
| usingnamespacestd; |

|  |
| --- |
| #define STACK 99999 |

|  |
| --- |
| intd, x, y; |

boolisPrime(intn){   
 if(n == 1) return0;   
 ints = sqrt(n) ;   
 for(inti = 2 ; i <= s ; i++){   
 if(n % i == 0) returnfalse;   
 }   
 returntrue;   
}

intfindPrimes(void\*arg){ //function for finding prime numbers.

inttemp = \*(int\*)arg;   
 intfirst = x + temp \* d;   
 intend = x + (temp + 1) \* d;   
 end = min(end,y);   
 //cout<<"("<<first<<" "<<end<<")"<<endl; for(inti = first ; i <= end ; i++){ if(isPrime(i)){   
 cout<<i<<" ";   
 }   
 }   
 return0;   
}

intmain(){   
intstat , rc, n;   
cout<<"Enter the range [x,y] = "; cin>>x>>y;   
cout<<"Enter the number of process = "; cin>>n;   
vector<int> tid(n,0);   
d = ceil((float)(y-x)/n);   
void \*\* stack = (void\*\*)malloc(sizeof(void\*)\*n); // stack for thread .

vector<int> arg(n,0);

if (stack == NULL ) {   
 printf("ERROR: Unable to allocate memory.\n"); exit(EXIT\_FAILURE);   
 }

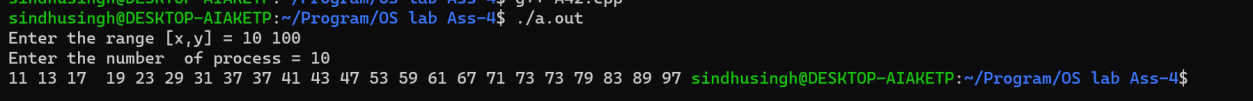
for(inti=0;i<n;i++){   
 arg[i] = i;   
 stack[i] = (char\*)malloc(STACK);   
 tid[i] = clone(findPrimes,(char\*)stack[i] +   
STACK,CLONE\_VM|CLONE\_THREAD|CLONE\_SIGHAND,&arg[i]);   
 if ( tid[i] < 0 ) {   
 printf("ERROR: Unable to create the child process.\n"); exit(EXIT\_FAILURE);   
 }   
 }

|  |
| --- |
| sleep(1); |

|  |
| --- |
| return0; |

|  |
| --- |
| } |

Output -



3.

#include<bits/stdc++.h>   
#include<sys/types.h>   
#include<sys/wait.h>   
#include<unistd.h>   
#include<fstream>   
#include<chrono>   
#include<pthread.h>   
usingnamespacestd;

fstreamfile ;   
inttemp = 0,d,x,y;

boolisPrime(intn){   
 if(n == 1) return0;   
 ints = sqrt(n) ;   
 for(inti = 2 ; i <= s ; i++){   
 if(n % i == 0) returnfalse;   
 }   
 returntrue;   
}

void\*findPrimes(void\*arg){   
 //file.open("prime.txt",ios::out | ios::app) ; intfirst = x + temp \* d;   
 intend = x + (++temp) \* d - 1;   
 end = min(end,y);   
 for(inti = first ; i <= end ; i++){   
 if(isPrime(i)){   
 // file<<i<<" ";   
 cout<<i<<" ";   
 }   
 }   
 //file.close();   
 returnarg;   
}

intmain(){   
file.open("prime.txt",ios::out);   
file.close() ;

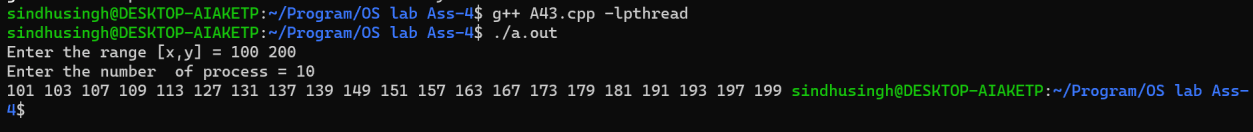
intstat, n, rc;   
cout<<"Enter the range [x,y] = "; cin>>x>>y; cout<<"Enter the number of process = "; cin>>n; vector<pthread\_t> tid(n);   
d = ceil((float)(y-x)/n);

for(inti=0;i<n;i++){   
 rc = pthread\_create(&tid[i], NULL, findPrimes, NULL); if (rc) {   
 cout<<"Error:unable to create thread,"<<rc<<endl; exit(-1);   
 }   
 }

for(inti = 0; i < n; i++){   
 pthread\_join(tid[i], NULL);   
 }

return0;   
}

Output



4.

|  |
| --- |
| #include<pthread.h> |

|  |
| --- |
| #include<iostream> |

|  |
| --- |
| #include<vector> |

|  |
| --- |
| usingnamespacestd; |

|  |
| --- |
| void\*attachable\_thread\_function(void\*arg){ |

|  |
| --- |
| int \* val = newint(\*(int\*)arg); |

|  |
| --- |
| cout<<"Pthread : "<<\*val<<" is attachable !"<<endl; |

|  |
| --- |
| pthread\_exit(val); |

}   
void\*detachable\_thread\_funtion(void\*arg){   
 int \* val = newint(\*(int\*)arg);   
 cout<<"Pthread : "<<\*val<<" is dettachable !"<<endl; pthread\_exit(NULL);   
}

intmain(){   
intn, res = 0, ret;   
cout<<"Enter number child = ";   
cin>>n;   
vector<pthread\_t> tid(n+1,0);   
vector<int> args(n+1,0);   
void \* val;

pthread\_attr\_tattr;   
ret = pthread\_attr\_init(&attr); // Initialized with default attributs ...

if(ret != 0){   
 cerr<<"Error to initialized with default attributes value !"<<endl; }   
 ret = pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_DETACHED); // Set the thread to detach state ...

if(ret != 0){   
 cerr<<"Error to set detachstate attribute !"<<endl; }

for(inti=1; i<=n; i++){   
 args[i] = i;   
 if(i%2) pthread\_create(&tid[i], &attr, detachable\_thread\_funtion, &args[i]); // Dettachable threads ...

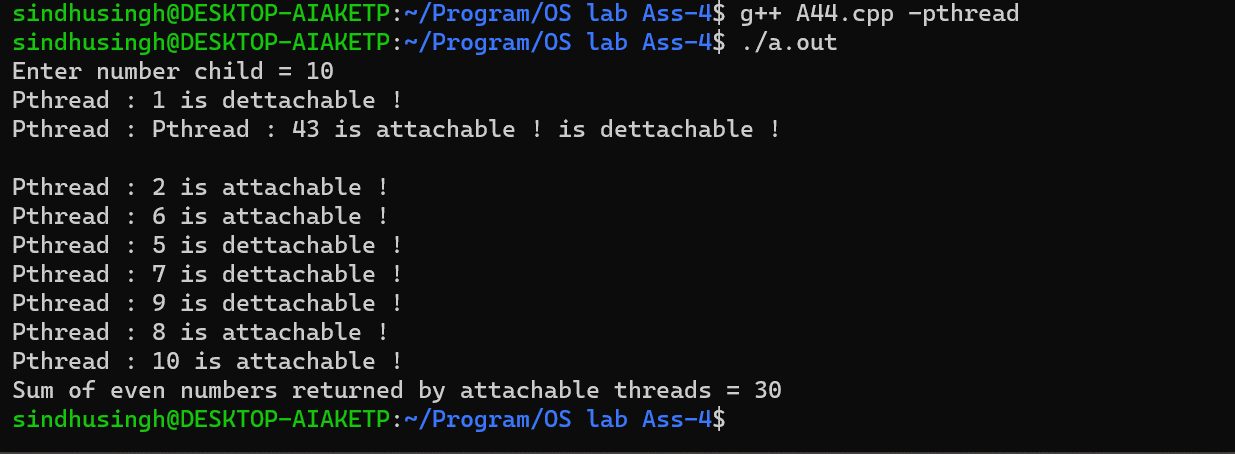
elsepthread\_create(&tid[i], NULL,   
attachable\_thread\_function,&args[i]); // Attachable threads ...

}

for(inti=2; i<=n; i+=2){   
 pthread\_join(tid[i], &val); // Waiting for attachable threads ...

res += \*(int\*)val;   
 }   
 cout<<"Sum of even numbers returned by attachable threads = "<<res<<endl; }

Output



5.

#include<iostream>   
#include<sys/ipc.h>   
#include<sys/shm.h>   
#include<sys/types.h>   
#include<unistd.h>   
#include<sys/wait.h>   
#include<errno.h>   
usingnamespacestd;

classstate{ //To reccord the state of shared memory.

public :   
 intshmid;   
 int \*shmaddr, \*s;   
 key\_tkey;   
 intshmflg;   
};

voidchildFunction(intm){   
 stateobj; /////// Initialization of state.

state \* p = &obj;   
p->key = 0x963;   
p->shmflg = 0666;

//////// Shared memory creation   
 p->shmid = shmget(p->key, sizeof(int), p->shmflg); if(p->shmid == -1){   
 perror("Shared memory !");   
 exit(0);   
 }

///////Attach to shared memory....

p->shmaddr = (int\*)shmat(p->shmid, NULL, 0); if(p->shmaddr == (int\*)-1){

perror("shared memory attach !");   
 exit(0);   
 }

p->s = p->shmaddr;   
 for(inti=0; i<m; i++){   
 (\*(p->s))++;   
 }

//////// Detaching to shread memory... if(shmdt(p->shmaddr)){   
 perror("Shmt !");   
 exit(0);   
 }   
 return;   
}

intmain(){   
intn, m;   
cout<<"Enter total numbers of child[n] = "; cin>>n;   
cout<<"How many times, you want to increase the value[m] = "; cin>>m;

/////// State object creation and initialization.

stateobj;   
state \* p = &obj;   
p->key = 0x963;   
p->shmflg = 0666|IPC\_CREAT;

///////// shared memry creation.

p->shmid = shmget(p->key, sizeof(int), p->shmflg); if(p->shmid == -1){   
 perror("Shared memory !");   
 exit(0);   
 }

//////////Attaching to shared memory.

p->shmaddr = (int\*)shmat(p->shmid, NULL, 0); if(p->shmaddr == (int\*)-1){   
 perror("shared memory attach !");   
 exit(0);   
 }

p->s = p->shmaddr;   
 (\*(p->s)) = 1;

////////// proccess creation ..   
 for(inti=0; i<n; i++){   
 if(!fork()){

childFunction(m);   
 exit(0);   
 }

|  |
| --- |
| } |

|  |
| --- |
| while(wait(0)>0); |

|  |
| --- |
| cout<<"value = "<<\*(p->s)<<endl; |

|  |
| --- |
| ///////// Dettaching shared memory. |

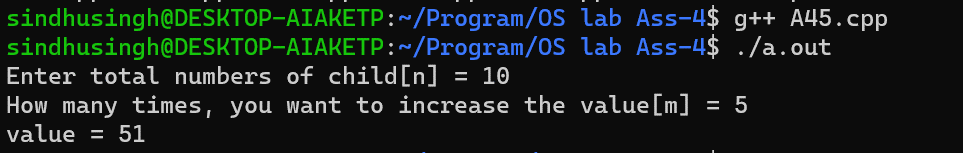
if (shmdt(p->shmaddr))   
 {   
 perror("Shmt !");   
 exit(0);

|  |
| --- |
| } |

|  |
| --- |
| ////////// Destroying shared memory. |

shmctl(p->shmid, IPC\_RMID, NULL);   
 return0;   
}

Output –



6.

#include <iostream>   
#include <sys/ipc.h>   
#include <sys/shm.h>   
#include <sys/types.h>   
#include <unistd.h>   
#include <sys/wait.h>   
#include <errno.h>   
#include<time.h>   
using namespace std;

class state   
{   
public:   
 int shmid;   
 int \*shmaddr, \*s;   
 key\_t key;   
 int shmflg;   
 size\_t size;   
};

void childFunction()   
{   
 state obj;   
 state \*p = &obj;   
 p->key = 0x963;   
 p->shmflg = 0666;   
 p->size = 11 \* sizeof(int);

p->shmid = shmget(p->key, sizeof(int) \* 10, p->shmflg); if (p->shmid == -1)   
 {   
 perror("Shared memory !");   
 exit(0);   
 }

p->shmaddr = (int \*)shmat(p->shmid, NULL, 0);   
 if (p->shmaddr == (int \*)-1)   
 {   
 perror("shared memory attach !");   
 exit(0);   
 }

p->s = p->shmaddr;

int sum = 0, i; float avg = 0;   
 for (int i = 1; i <= 10; i++)   
 {   
 if ((i % 2) && (\*(p->s + i) != 0))   
 {   
 \*((p->s) + (i+1)) = \*(p->s + i) + 2;   
 }   
 }

for (int i = 1; i <= 10; i += 2)   
 sum += \*((p->s) + i);   
 avg = (float)sum / 5;   
 cout << "Toatal Sum of odd index possition, calculated by child = " << sum << endl; cout << "Avgrage of even index possition, calculated by child = " << avg << endl;

if (shmdt(p->shmaddr))   
 {   
 perror("Shmt !");   
 exit(0);   
 }   
 return;   
}

int main()   
{   
 srand(time(0));   
 state obj;   
 state \*p = &obj;   
 p->key = 0x963;

p->shmflg = 0666 | IPC\_CREAT;   
 p->size = 11 \* sizeof(int);

p->shmid = shmget(p->key, p->size, p->shmflg);   
 if (p->shmid == -1)   
 {   
 perror("Shared memory !");   
 exit(0);   
 }

p->shmaddr = (int \*)shmat(p->shmid, NULL, 0);   
 if (p->shmaddr == (int \*)-1)   
 {   
 perror("shared memory attach !");   
 exit(0);   
 }

p->s = p->shmaddr;   
 for (int i = 1; i <= 10; i++)   
 {   
 \*((p->s) + i) = 0;   
 }

pid\_t pid;   
 if ((pid = fork()) == 0)   
 {   
 childFunction();   
 return 0;   
 }

for (int i = 1; i < 10; i += 2)

{   
 \*(p->s + i) = 1 + rand() % 100;   
 }   
 waitpid(pid, NULL, -1);   
 sleep(1);

int sum = 0;   
 float avg =0;   
 for (int i = 2; i <= 10; i += 2)   
 {   
 sum += \*((p->s) + i);   
 }   
 avg = (float)sum / 5;   
 cout << "Toatal Sum of even index possition, calculated by parent = " << sum << endl; cout << "Avgrage of even index possition, calculated by parent = " << avg << endl;

if (shmdt(p->shmaddr))   
 {   
 perror("Shmt !");   
 exit(0);   
 }   
 shmctl(p->shmid, IPC\_RMID, NULL);

return 0;   
}

Output –

