Assignment

Spring 2022

CSE-204 Operating Systems

Submitted by:

Muhammad Arsalan Kamran

Registration number:

20PWCSE1957

Class Section: C

“On my honor, as a student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work”

Submitted to:

Dr. Tariq Kamal

Wednesday, June 15, 2022

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Assignment 2

Question 1:

Discuss the inheritance in terms of data and control information when a process creates a child process. Describe the implication of data inheritance.

Answer:

Inheritance of Data and Control Information:

In Linux/Unix processes after a ‘fork( )’, a child process inherits data and control information from its parent process, while in Windows OS, a new process created with ‘CreateProcess( )’ does not inherit any data and control information from a parent process.

While using fork() to create the child processes, they'll inherit everything that's global for the actual parent process' context:

- Environment variable settings
- Opened file descriptors and etc.

Implication of Data Inheritance:

Global scope variables will be copied to the child process context from the state they are. Changes to these variables will not be reflected in the parent process. A child process can inherit several properties and resources from its parent process. We can also prevent a child process from inheriting properties from its parent process. The inheritance includes:

- Environment variables.
- The current directory.
- The error mode, as set by the SetErrorMode function.
- The processor affinity mask.
- The association with a job.
- Parent's standard handles

Also, the child process cannot inherit the following things:

- Priority class
- GDI or USER handles
- Pseudo handles
- DLL module handles returned by the LoadLibrary function.

Question 2:

If you are given a task to add fields to PCB. What fields would you add and why?

Answer:

A process control block (PCB) contains information about the process i.e., registers etc. It is used to track the process's execution status.

The following fields would be added to the PCB (Process Control Block):

- **Process-Id/Process Number:** It represents the identification number of a certain process.
- **Pointer:** It is a stack pointer which is required to be saved when the process is switched from one state to another state to retain the current position of the process.
- **Process State**: It stores the respective state of the process i.e. ready, running, wait etc.
- **Program Counter**: It stores the counter which contains the address of the next instruction to be executed.
- **Memory limits:** This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.
- **Process Priority**: It gives the priority to the certain process to be executed depending upon the state of the process.
- **Accounting Information**: This field includes information about the amount of CPU used, time constraints, jobs, or process number, etc.
- **Open files list**: This information includes the list of files opened for a process.

Question 3:

Write a C program that performs addition of two NxN matrices;

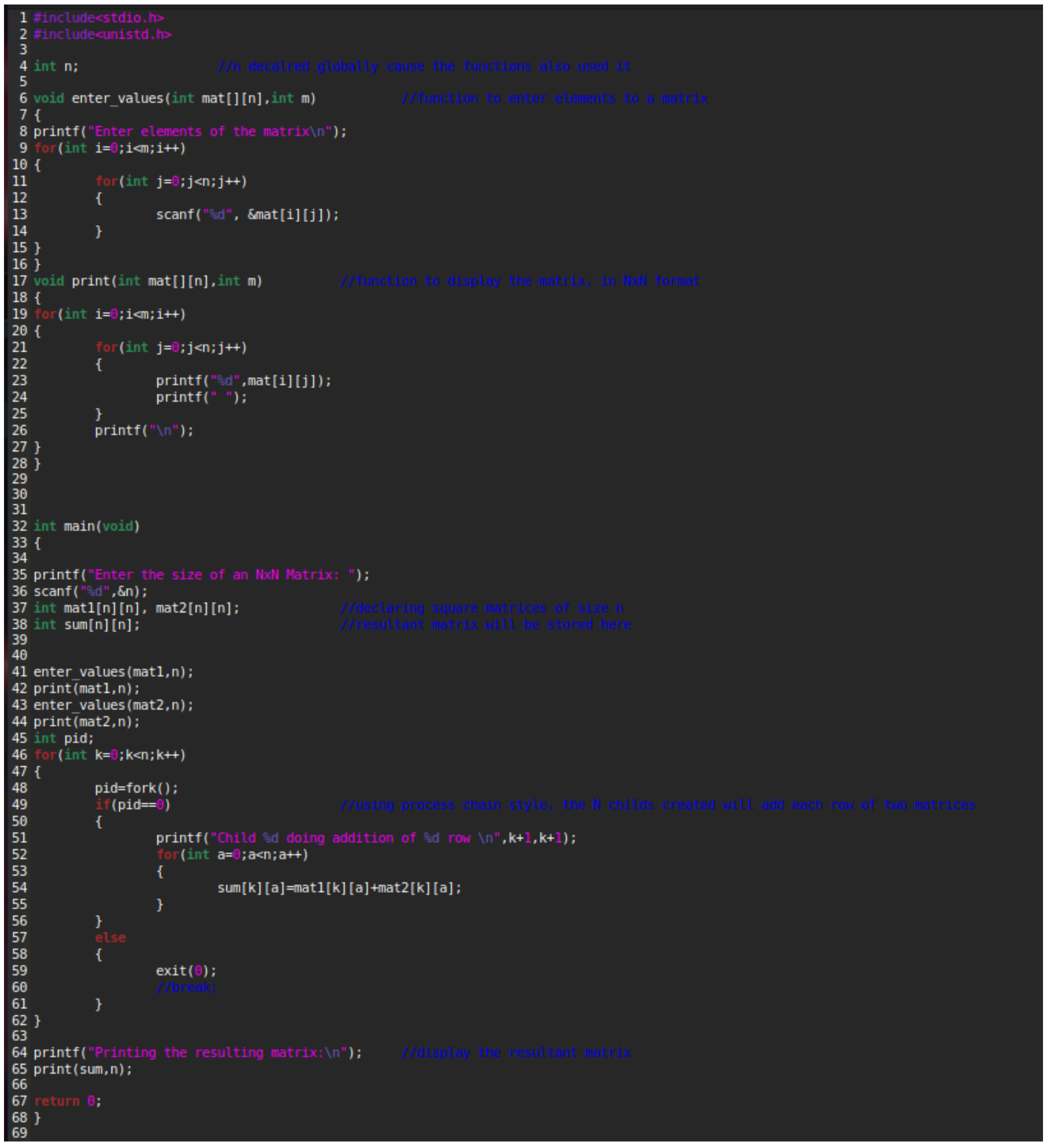
- Use multi-processing with N processes.
- Use multi-threading with N threads.

Answer:

By using multi-processing:

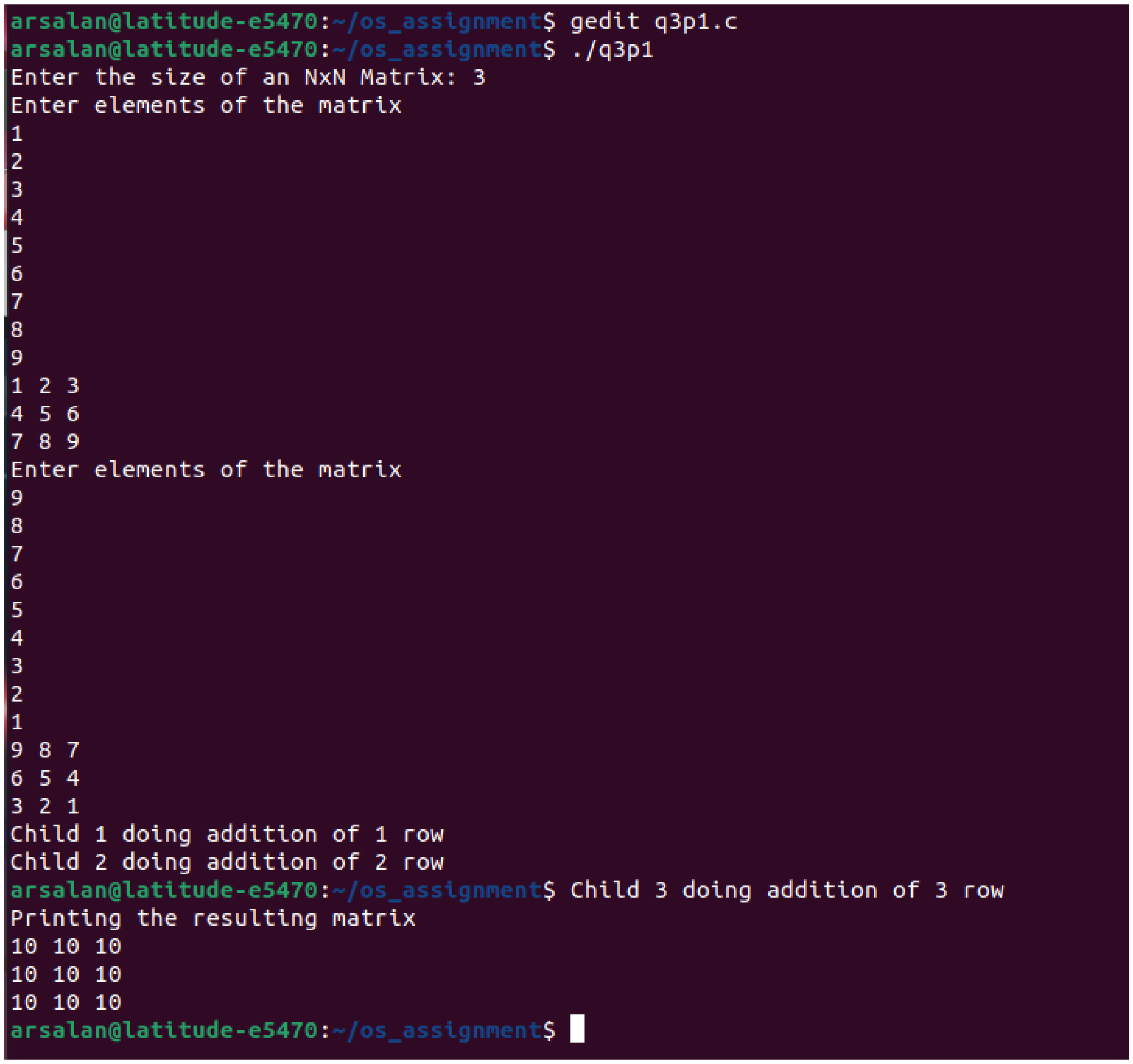
To add NxN matrix by multiprocessing, I created N processes each adding one row of two matrices. Example, process 1 adds first rows of first and second matrix. Similarly, process 2 adds second rows of first and second matrix and so on.

Code:



```c
#include<stdio.h>
#include<unistd.h>

int n;                  //n decalred globally cause the functions also used it

void enter_values(int mat[][n],int m)           //function to enter elements to a matrix
{
printf("Enter elements of the matrix\n");
for(int i=0;i<m;i++)
{
        for(int j=0;j<n;j++)
        {
                scanf("%d", &mat[i][j]);
        }
}
}
void print(int mat[][n],int m)          //function to display the matrix, in NxN format
{
for(int i=0;i<m;i++)
{
        for(int j=0;j<n;j++)
        {
                printf("%d",mat[i][j]);
                printf(" ");
        }
        printf("\n");
}
}



int main(void)
{

printf("Enter the size of an NxN Matrix: ");
scanf("%d",&n);
int mat1[n][n], mat2[n][n];             //declaring square matrices of size n
int sum[n][n];                          //resultant matrix will be stored here


enter_values(mat1,n);
print(mat1,n);
enter_values(mat2,n);
print(mat2,n);
int pid;
for(int k=0;k<n;k++)
{
        pid=fork();
        if(pid==0)                      //using process chain style, the N childs created will add each row of two matrices
        {
                printf("Child %d doing addition of %d row \n",k+1,k+1);
                for(int a=0;a<n;a++)
                {
                        sum[k][a]=mat1[k][a]+mat2[k][a];
                }
        }
        else
        {
                exit(0);
                //break;
        }
}

printf("Printing the resulting matrix:\n");     //display the resultant matrix
print(sum,n);

return 0;
}

```

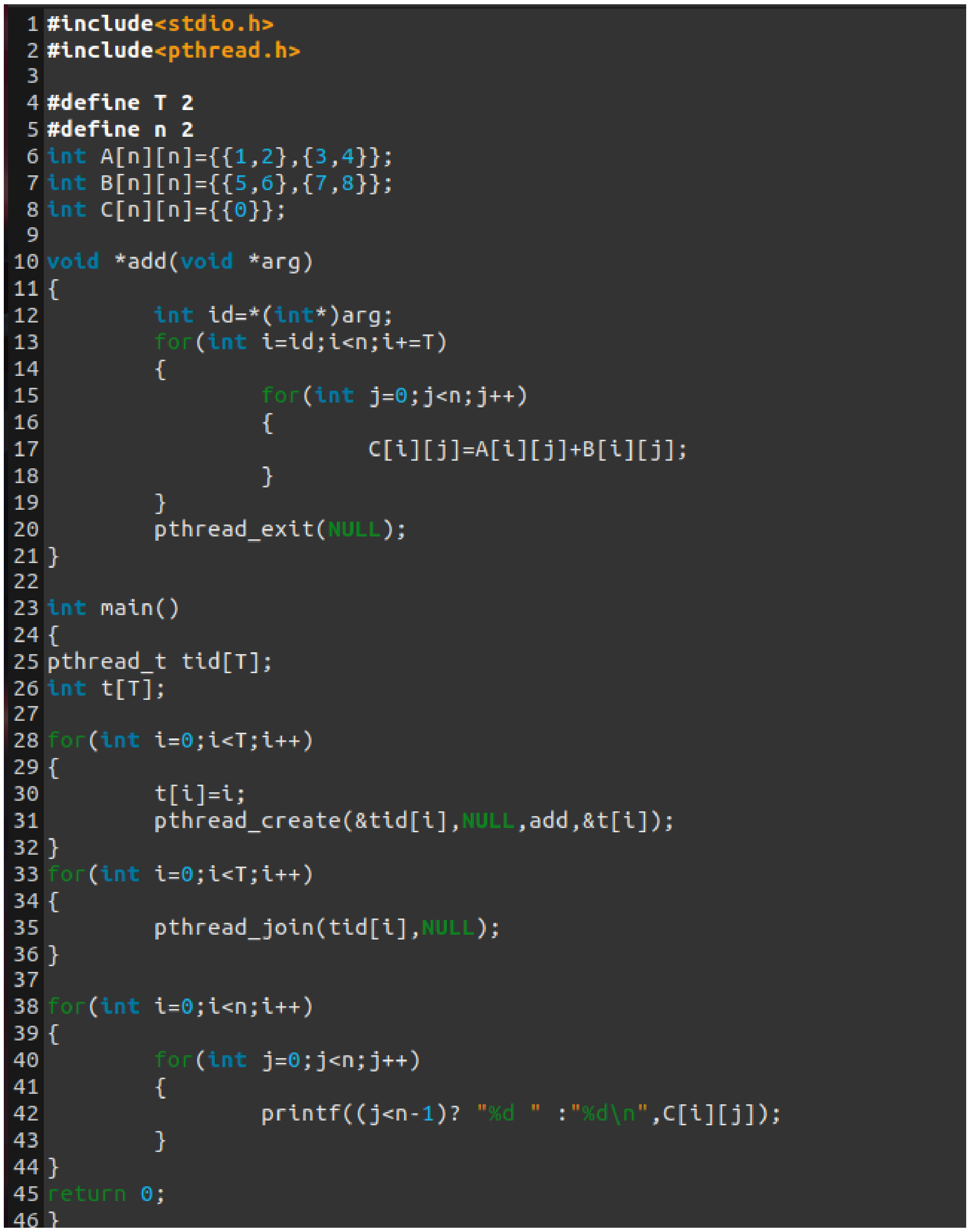
Output:



```
arsalan@latitude-e5470:~/os_assignment$ gedit q3p1.c
arsalan@latitude-e5470:~/os_assignment$ ./q3p1
Enter the size of an NxN Matrix: 3
Enter elements of the matrix
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
Enter elements of the matrix
9
8
7
6
5
4
3
2
1
9 8 7
6 5 4
3 2 1
Child 1 doing addition of 1 row
Child 2 doing addition of 2 row
arsalan@latitude-e5470:~/os_assignment$ Child 3 doing addition of 3 row
Printing the resulting matrix
10 10 10
10 10 10
10 10 10
arsalan@latitude-e5470:~/os_assignment$
```

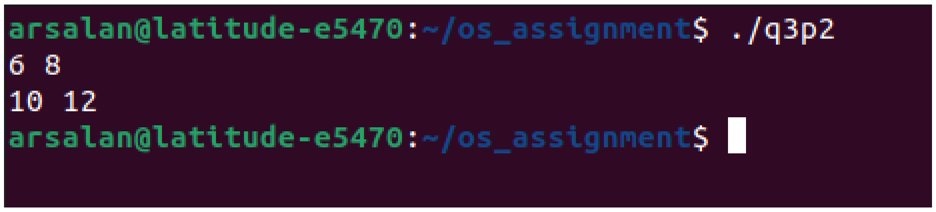
By using multi-threading:

Code:



```c
#include<stdio.h>
#include<pthread.h>

#define T 2
#define n 2
int A[n][n]={{1,2},{3,4}};
int B[n][n]={{5,6},{7,8}};
int C[n][n]={{0}};

void *add(void *arg)
{
        int id=*(int*)arg;
        for(int i=id;i<n;i+=T)
        {
                for(int j=0;j<n;j++)
                {
                        C[i][j]=A[i][j]+B[i][j];
                }
        }
        pthread_exit(NULL);
}

int main()
{
pthread_t tid[T];
int t[T];

for(int i=0;i<T;i++)
{
        t[i]=i;
        pthread_create(&tid[i],NULL,add,&t[i]);
}
for(int i=0;i<T;i++)
{
        pthread_join(tid[i],NULL);
}

for(int i=0;i<n;i++)
{
        for(int j=0;j<n;j++)
        {
                printf((j<n-1)? "%d " :"%d\n",C[i][j]);
        }
}
return 0;
}
```

Output:



```
arsalan@latitude-e5470:~/os_assignment$ ./q3p2
6 8
10 12
arsalan@latitude-e5470:~/os_assignment$
```