

EXPERIMENT: -5 (Processes Management using System Calls)

Objective: To introduce the concept of creating a new child process, performing operations on processes and working with orphan and zombie processes.

Lab Exercise Solution(s):

Q1. Write a program using system calls for operation on process to simulate that n fork calls create $(2^n - 1)$ child processes.

Ans1.

Solution: -

Step1: - nano 5_1.c

Step2: -

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

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

```
#include<sys/types.h>
```

```
int main(){
```

```
    int n;
```

```
    printf("# Enter the no. of times you want to run the fork system call: ");
```

```
    scanf("%d", &n);
```

```
    for(int i=0; i<n; i++){
```

```
        pid_t r;
```

```
        r = fork();
```

```
        if(r==0){
```

```
            printf("Current child process pid is %d \n", getpid());
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

Step3: - gcc 5_1.c

Step4: - ./a.out

nano 5_1.c

```
GNU nano 7.2 Exp5_1.c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>

int main(){
    int n;
    printf("# Enter the no. of times you want to run the fork system call: ");
    scanf("%d", &n);
    for(int i=0; i<n; i++){
        pid_t r;
        r = fork();
        if(r==0){
            printf("Current child process pid is %d \n", getpid());
        }
    }
    return 0;
}
```

Output of the program

```
(shreygrg@ Linux23) - [~/Shrey_Garg]
$ nano Exp5_1.c

(shreygrg@ Linux23) - [~/Shrey_Garg]
$ gcc Exp5_1.c

(shreygrg@ Linux23) - [~/Shrey_Garg]
$ ./a.out
# Enter the no. of times you want to run the fork system call: 3
Current child process pid is 23697
Current child process pid is 23701
Current child process pid is 23700
Current child process pid is 23702
Current child process pid is 23698

Current child process pid is 23703
Current child process pid is 23699
```

Q2. Write a program using systems for operations on processes to create a hierarchy of processes P1 → P2 → P3. Also print the id and parent id for each process.

Ans2.

Solution: -

Step1: - nano 5_2.c

Step2: -

```
#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include <stdlib.h>

int main()

{

    printf("Parent PID : %d \n", (int) getpid());

    pid_t pid = fork();

    if(pid == 0)

    {

        printf("Child 1 PID : %d Parent PID : %d\n", (int) getpid(), (int) getppid());

        pid_t pid_1 = fork();

        if(pid_1 == 0)

        {

            printf("Child 2 PID : %d Parent PID (Child 1) : %d \n", (int) getpid(), (int) getppid());

            exit(0);

        }

        else

        {

            exit(0);

        }

    }

    else

    {

        exit(0);

    }

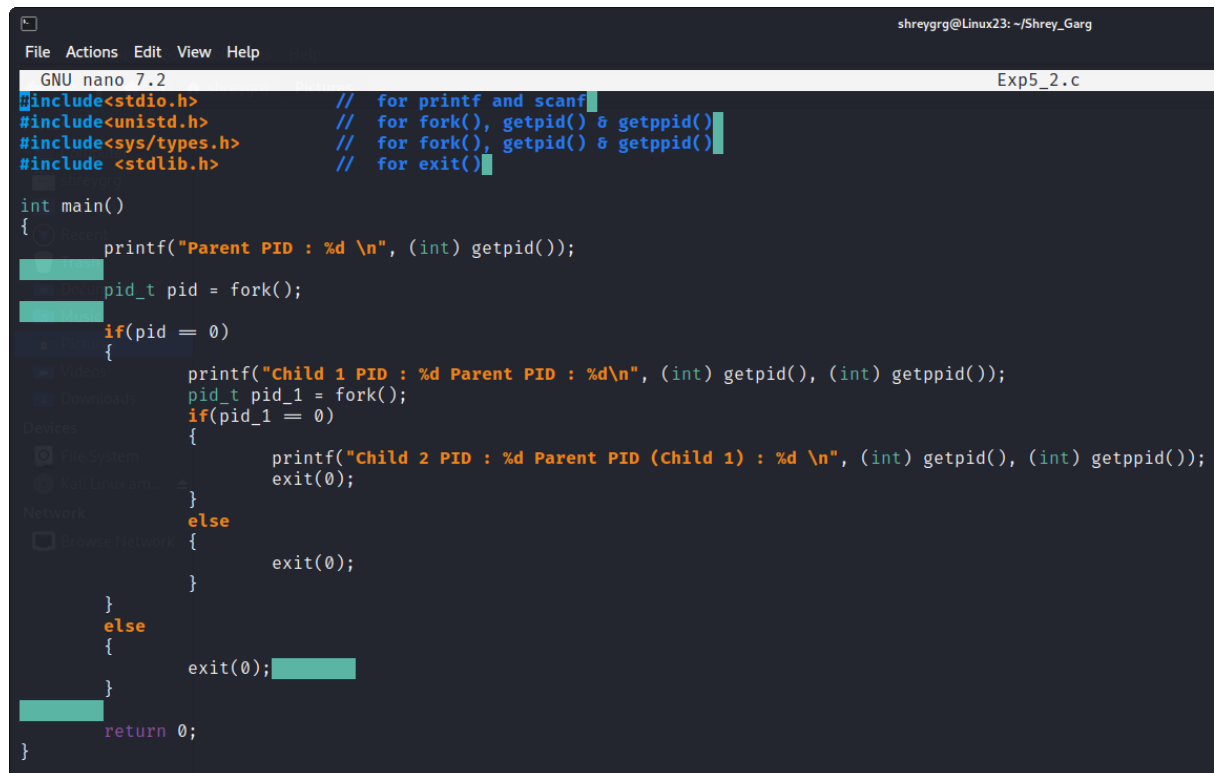
    return 0;

}
```

Step3: - gcc 5_2.c

Step4: - ./a.out

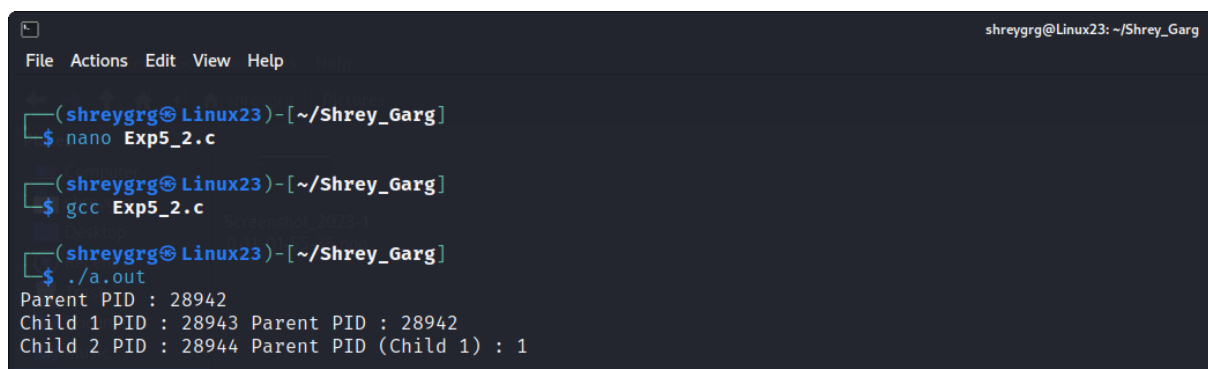
nano 5_2.c



```
GNU nano 7.2 Exp5_2.c
#include<stdio.h>           // for printf and scanf
#include<unistd.h>          // for fork(), getpid() & getppid()
#include<sys/types.h>       // for fork(), getpid() & getppid()
#include <stdlib.h>         // for exit()

int main()
{
    printf("Parent PID : %d \n", (int) getpid());
    pid_t pid = fork();
    if(pid == 0)
    {
        printf("Child 1 PID : %d Parent PID : %d\n", (int) getpid(), (int) getppid());
        pid_t pid_1 = fork();
        if(pid_1 == 0)
        {
            printf("Child 2 PID : %d Parent PID (Child 1) : %d \n", (int) getpid(), (int) getppid());
            exit(0);
        }
        else
        {
            exit(0);
        }
    }
    else
    {
        exit(0);
    }
    return 0;
}
```

Output of the program



```
(shreygrg@ Linux23) - [~/Shrey_Garg]
$ nano Exp5_2.c
(shreygrg@ Linux23) - [~/Shrey_Garg]
$ gcc Exp5_2.c
(shreygrg@ Linux23) - [~/Shrey_Garg]
$ ./a.out
Parent PID : 28942
Child 1 PID : 28943 Parent PID : 28942
Child 2 PID : 28944 Parent PID (Child 1) : 1
```

Q3. Write a program using system calls for operation on processes to create a hierarchy of processes as given in figure 1. Also, simulate process p4 as orphan and P5 as zombie.

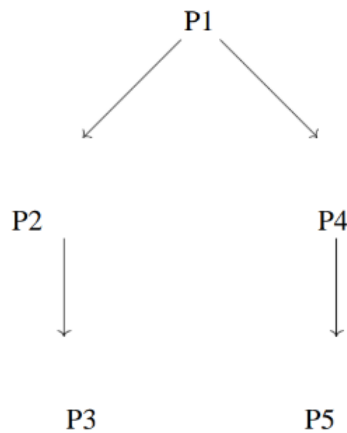


Figure 1: Hierarchy of Processes

Ans3.

Solution: -

Step1: - nano 5_3.c

Step2: -

```

#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>

int main()
{
    printf("P1 PID : %d \n", (int) getpid());
    pid_t pid = fork();

    if(pid == 0)
    {
        printf("P4 PID : %d P1 PID : %d\n", (int) getpid(), (int) getppid());
        printf("Child process P4 is sleeping \n");
        pid_t pid_1 = fork();
        sleep(5);
        if(pid_1 == 0)
        {

```

```

        printf("P5 PID : %d P4 PID : %d \n", (int) getpid(), (int) getppid());
        printf("Zombie process P5's PID : %d \n", (int) getpid());
    }
    else{
        printf("Orphan child process P4's PID : %d \n", (int) getpid());
        printf("P4's New Parent PID : %d \n", (int) getppid());
    }
}
else
{
    pid = fork();
    if(pid == 0)
    {
        printf("P2 PID : %d P1 PID : %d\n", (int) getpid(), (int) getppid());
        pid_t pid_1 = fork();
        if(pid_1 == 0)
        {
            printf("P3 PID : %d P2 PID : %d \n", (int) getpid(), (int) getppid());
            exit(0);
        }
        else
        {
            exit(0);
        }
    }
    else
    {
        exit(0);
    }
}

return 0;
}

```

Step3: - gcc 5_3.c

Step4: - ./a.out

nano 5_3.c

```
GNU nano 7.2 Exp5_3.c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>

int main()
{
    printf("P1 PID : %d \n", (int) getpid());
    pid_t pid = fork();
    if(pid == 0)
    {
        printf("P4 PID : %d P1 PID : %d\n", (int) getpid(), (int) getppid());
        printf("Child process P4 is sleeping \n");
        pid_t pid_1 = fork();
        sleep(5);
        if(pid_1 == 0)
        {
            printf("P5 PID : %d P4 PID : %d \n", (int) getpid(), (int) getppid());
            printf("Zombie process P5's PID : %d \n", (int) getpid());
        }
        else{
            printf("Orphan child process P4's PID : %d \n", (int) getpid());
            printf("P4's New Parent PID : %d \n", (int) getppid());
        }
    }
    else{
        pid = fork();
        if(pid == 0)
        {
            printf("P2 PID : %d P1 PID : %d\n", (int) getpid(), (int) getppid());
            pid_t pid_1 = fork();
            if(pid_1 == 0)
            {
                printf("P3 PID : %d P2 PID : %d \n", (int) getpid(), (int) getppid());
                exit(0);
            }
            else
            {
                exit(0);
            }
        }
        else
        {
            exit(0);
        }
    }
    return 0;
}
```

Output of the program

```
shreygrg@Linux23: ~/Shrey_Garg
$ nano Exp5_3.c

shreygrg@Linux23: ~/Shrey_Garg
$ gcc Exp5_3.c

shreygrg@Linux23: ~/Shrey_Garg
$ ./a.out
P1 PID : 31646
P2 PID : 31648 P1 PID : 31646
P4 PID : 31647 P1 PID : 1
Child process P4 is sleeping

P3 PID : 31650 P2 PID : 1
Orphan child process P4's PID : 31647
P4's New Parent PID : 1
P5 PID : 31649 P4 PID : 31647
Zombie process P5's PID : 31649
```

Q4. Write a program using system calls for operation on processes to create a hierarchy of processes 2. Also, simulate P4 as an orphan process and P7 as a zombie.

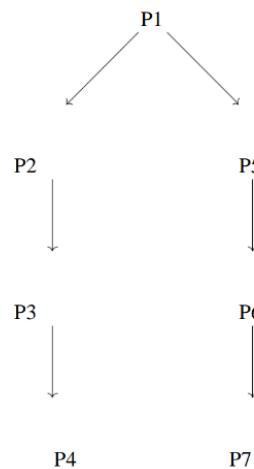


Figure 2: Hierarchy of Processes

Ans4.

Solution: -

Step1: - nano 5_4.c

Step2: -

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<stdlib.h>

int main()

{

printf("P1 PID : %d \n", (int) getpid());

pid_t pid = fork();

if(pid == 0)

{

printf("P5 PID : %d Parent P1 PID : %d\n", (int) getpid(), (int) getppid());

pid_t pid_1 = fork();


```

        if(pid_1 == 0)
        {
            printf("P6 PID : %d Parent P5 PID : %d \n", (int) getpid(), (int)
getppid());

            pid_t pid_2 = fork();
            sleep(5);

            if(pid_2 == 0)
            {
                printf("Zombie process P7's PID: %d \n", (int) getpid());
                printf("Parent P6 PID : %d \n", (int) getppid());
            }
            else
            {
                exit(0);
            }
        }
        else
        {
            exit(0);
        }
    }

else
{
    pid = fork();
    if(pid == 0)
    {

```

```

    printf("P2 PID : %d Parent P1 PID : %d\n", (int) getpid(), (int)
getppid());

    pid_t pid_1 = fork();
    if(pid_1 == 0)
    {
        printf("P3 PID : %d Parent P2 PID : %d \n", (int) getpid(), (int)
getppid());

        pid_t pid_2 = fork();
        if(pid_2 == 0)
        {
            sleep(3);
        }
        else
        {
            printf("Orphan child process P4's PID : %d \n", (int)
pid_2);

            printf("P4's New Parent PID : %d \n", (int) getppid());
        }
    }
    else
    {
        exit(0);
    }
}

return 0;
}

```

nano 5_4.c

```
File Actions Edit View Help
GNU nano 7.2 Exp5_4.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>

int main()
{
    printf("P1 PID : %d\n", (int) getpid());
    pid_t pid = fork();
    if(pid == 0)
    {
        printf("P5 PID : %d Parent P1 PID : %d\n", (int) getpid(), (int) getpid());
        pid_t pid_1 = fork();
        if(pid_1 == 0)
        {
            printf("P6 PID : %d Parent P5 PID : %d\n", (int) getpid(), (int) getpid());
            pid_t pid_2 = fork();
            sleep(5);
            if(pid_2 == 0)
            {
                printf("Zombie process P7's PID: %d\n", (int) getpid());
                printf("Parent P6 PID : %d\n", (int) getpid());
            }
            else
            {
                exit(0);
            }
        }
        else
        {
            exit(0);
        }
    }
    else
    {
        pid = fork();
        if(pid == 0)
        {
            printf("P2 PID : %d Parent P1 PID : %d\n", (int) getpid(), (int) getpid());
            pid_t pid_1 = fork();
            if(pid_1 == 0)
            {
                printf("P3 PID : %d Parent P2 PID : %d\n", (int) getpid(), (int) getpid());
                pid_t pid_2 = fork();
                if(pid_2 == 0)
                {
                    sleep(3);
                }
                else
                {
                    printf("Orphan child process P4's PID : %d\n", (int) pid_2);
                    printf("P4's New Parent PID : %d\n", (int) getpid());
                }
            }
            else
            {
                exit(0);
            }
        }
    }
    return 0;
}
```

Output of the program

```
shreygrg@Linux23: ~/Shrey_Garg
$ nano Exp5_4.c

shreygrg@Linux23: ~/Shrey_Garg
$ gcc Exp5_4.c

shreygrg@Linux23: ~/Shrey_Garg
$ ./a.out
P1 PID : 40102
P5 PID : 40103 Parent P1 PID : 40102
P2 PID : 40104 Parent P1 PID : 1
P3 PID : 40105 Parent P2 PID : 1
Orphan child process P4's PID : 40106
P4's New Parent PID : 1

P6 PID : 40107 Parent P5 PID : 1
shreygrg@Linux23: ~/Shrey_Garg
$ Zombie process P7's PID: 40109
Parent P6 PID : 40107
```

EXPERIMENT: -6 (Creation of Multithreaded Processes using Pthread Library)

Objective: Introduce the operations on threads, which include initialization, creation, join and exit functions of thread using pthread library.

Lab Exercise Solution(s):

Q1. Write a program using pthread to concatenate the strings, where multiple strings are passed to thread function.

Ans1.

Solution: -

Step1: - nano 6_1.c

Step2: -

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

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

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

```
#include<string.h>
```

```
char str1[100], str2[100];
```

```
char result[1000];
```

```
void *concatenatestrings(){
```

```
    strcat(result, str1);
```

```
    strcat(result, str2);
```

```
    pthread_exit(NULL);
```

```
}
```

```
int main(){
```

```
    pthread_t thread;
```

```
    printf("* Enter the first string: ");
```

```
    scanf("%s", str1);
```

```
    printf("* Enter the second string: ");
```

```

scanf("%s", str2);

pthread_create(&thread, NULL, concatenatestrings, NULL);

pthread_join(thread, NULL);

printf("@ Final result is: %s \n", result);

return 0;
}

```

Step3: - gcc 6_1.c

Step4: - ./a.out

nano 6_1.c

```

UW PICO 5.09 File: 6_1.c
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#include<string.h>

char str1[100], str2[100];
char result[1000];

void *concatenatestrings(){
    strcat(result, str1);
    strcat(result, str2);
    pthread_exit(NULL);
}

int main(){
    pthread_t thread;
    printf("* Enter the first string: ");
    scanf("%s", str1);
    printf("* Enter the second string: ");
    scanf("%s", str2);

    pthread_create(&thread, NULL, concatenatestrings, NULL);
    pthread_join(thread, NULL);

    printf("@ Final result is: %s \n", result);
    return 0;
}

```

Output of the program

```

srikarmerugu@Srikars-MacBook-Air ~ % nano 6_1.c
srikarmerugu@Srikars-MacBook-Air ~ % gcc 6_1.c
srikarmerugu@Srikars-MacBook-Air ~ % ./a.out
* Enter the first string: srikar
* Enter the second string: merugu
@ Final result is: srikarmerugu
srikarmerugu@Srikars-MacBook-Air ~ %

```

Q2. Write a program using pthread to find the length of string, where strings are passed to thread function.

Ans2.

Solution: -

Step1: - nano 6_2.c

Step2: -

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

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

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

```
#include<string.h>
```

```
char length1[100];
```

```
int length=0;
```

```
void *lengthstr(){
```

```
    length=strlen(length1);
```

```
    pthread_exit(NULL);
```

```
}
```

```
int main(){
```

```
    pthread_t thread;
```

```
    printf("* Enter the String: ");
```

```
    scanf("%[^\n]s", length1);
```

```
    pthread_create(&thread, NULL, lengthstr, NULL);
```

```
    pthread_join(thread, NULL);
```

```
    printf("* Total length of string is: %d \n", length);
```

```
    return 0;
```

```
}
```

Step3: - gcc 6_2.c

Step4: - ./a.out

nano 6_2.c

```
UW PICO 5.09 File: 6_2.c
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#include<string.h>

char length1[100];
int length=0;

void *lengthstr(){
    length=strlen(length1);
    pthread_exit(NULL);
}

int main(){
    pthread_t thread;
    printf("* Enter the String: ");
    scanf("%[^\n]s", length1);

    pthread_create(&thread, NULL, lengthstr, NULL);
    pthread_join(thread, NULL);
    printf("* Total length of string is: %d \n", length);
    return 0;
}
```

Output of the program

```
[srikarmerugu@Srikars-MacBook-Air ~ % nano 6_2.c
[srikarmerugu@Srikars-MacBook-Air ~ % gcc 6_2.c
[srikarmerugu@Srikars-MacBook-Air ~ % ./a.out
* Enter the String: srikar
* Total length of string is: 6
srikarmerugu@Srikars-MacBook-Air ~ %
```

Q3. Write a program that performs statistical operations of calculating the average, maximum and minimum for a set of numbers. Create three threads where each performs their respective operations.

Ans3.

Solution: -

Step1: - nano 6_3.c

Step2: -

#include<stdio.h>

#include<pthread.h>

int arr[10] = {99, 22, 00, 88, 11, 102, 33, 66, 44, 55};

```
void *sort(){  
    for(int i=0; i<10; i++){  
        for(int j=0; j<10; j++){  
            if(arr[i] < arr[j]){  
                int temp = arr[i];  
                arr[i] = arr[j];  
                arr[j] = temp;  
            }  
        }  
    }  
}
```

```
void *min(){  
    int min = arr[0];  
    printf("* Minimum element is = %d\n", min);  
    pthread_exit(NULL);
```



```
}
```

```
void *max(){  
    int max = arr[9];  
    printf("* Maximum element is = %d \n", max);  
    pthread_exit(NULL);  
}
```

```
void *avg(){  
    int sum=0;  
    for(int i=0;i<10;i++)  
    {  
        sum = sum + arr[i];  
    }  
    sum = sum/10;  
  
    printf("* The average of the elements = %d \n", sum);  
    printf("\n");  
    pthread_exit(NULL);  
}
```

```
int main(){  
  
    printf("\n");  
  
    /*  
    printf("Enter 10 elements in the array: ");
```

```
for(int i=0; i<10; i++)
{
    scanf("%d", &arr[i]);
}
printf("\n");
*/
```

```
printf("# Initial input array is: ");
for(int i=0; i<10; i++){
    printf("%d ", arr[i]);
}
printf("\n");
```

```
pthread_t sort_thread, max_thread, min_thread, avg_thread;
```

```
pthread_create(&sort_thread, NULL, sort, NULL);
pthread_join(sort_thread, NULL);
```

```
pthread_create(&max_thread, NULL, max, NULL);
pthread_join(max_thread, NULL);
```

```
pthread_create(&min_thread, NULL, min, NULL);
pthread_join(min_thread, NULL);
```

```
pthread_create(&avg_thread, NULL, avg, NULL);
pthread_join(avg_thread, NULL);
return 0;
```

```
}
```

Step3: - gcc 6_3.c

Step4: - ./a.out

nano 6_3.c



```
GNU nano 2.2
#include <stdio.h>
#include <pthread.h>

int arr[10] = {99, 22, 88, 88, 11, 102, 33, 66, 44, 55};

void *sort(){
    for(int i=0; i<10; i++){
        for(int j=0; j<10; j++){
            if(arr[i] < arr[j]){
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

void *min(){
    int min = arr[0];
    printf("Minimum element is = %d\n", min);
    pthread_exit(NULL);
}

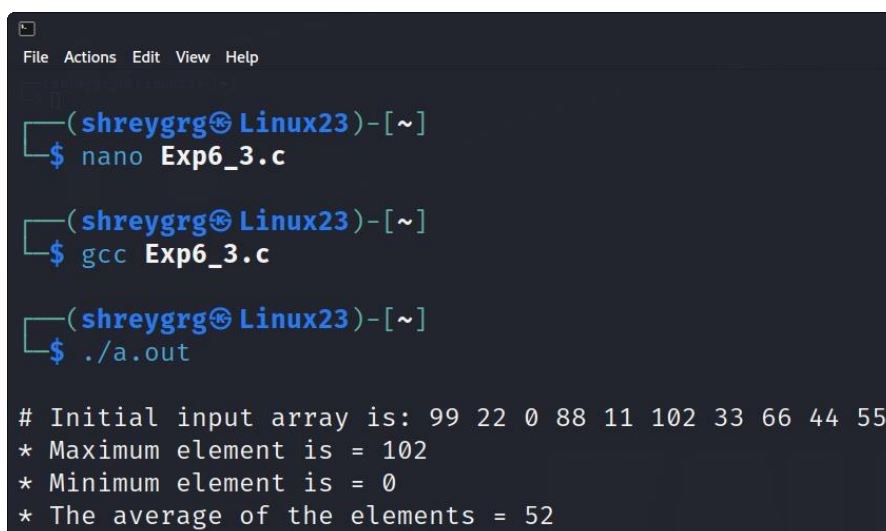
void *max(){
    int max = arr[0];
    printf("Maximum element is = %d\n", max);
    pthread_exit(NULL);
}

void *avg(){
    int sum=0;
    for(int i=0; i<10; i++){
        sum = sum + arr[i];
    }
    sum = sum/10;
    printf("The average of the elements = %d\n", sum);
    printf("\n");
    pthread_exit(NULL);
}

int main(){
    printf("\n");
    /*
    printf("Enter 10 elements in the array: ");
    for(int i=0; i<10; i++){
        scanf("%d", &arr[i]);
    }
    printf("\n");
    */
    printf("Initial input array is: ");
    for(int i=0; i<10; i++){
        printf("%d ", arr[i]);
    }
    printf("\n");

    pthread_t sort_thread, max_thread, min_thread, avg_thread;
    pthread_create(&sort_thread, NULL, sort, NULL);
    pthread_join(sort_thread, NULL);
    pthread_create(&max_thread, NULL, max, NULL);
    pthread_join(max_thread, NULL);
    pthread_create(&min_thread, NULL, min, NULL);
    pthread_join(min_thread, NULL);
    pthread_create(&avg_thread, NULL, avg, NULL);
    pthread_join(avg_thread, NULL);
    return 0;
}
```

Output of the program



```
(shreygrg@ Linux23)-[~]
$ nano Exp6_3.c

(shreygrg@ Linux23)-[~]
$ gcc Exp6_3.c

(shreygrg@ Linux23)-[~]
$ ./a.out

# Initial input array is: 99 22 0 88 11 102 33 66 44 55
* Maximum element is = 102
* Minimum element is = 0
* The average of the elements = 52
```

Q4. Write a multithreaded program where an array of integers is passed globally and is divided into two smaller lists and given as input to two threads. The thread will sort their half of the list and will pass the sorted list to a third thread which merges and sorts the list. The final sorted list is printed by the parent thread.

Ans4.

Solution: -

Step1: - nano 6_4.c

Step2: -

#include<stdio.h>

#include<pthread.h>

int arr[10] = {99, 22, 00, 88, 11, 100, 33, 66, 44, 55};

int arr_first_half[5], arr_second_half[5], final_arr[10];

void *final_merge_sort(){

for(int i=0; i<5; i++){

final_arr[i] = arr_first_half[i];

final_arr[i+5] = arr_second_half[i];

}

printf("# Merged array is: ");

for(int i=0; i<10; i++){

printf("%d ", final_arr[i]);

}

printf("\n");

```

for(int i=0; i<10; i++){
    for(int j=0; j<10; j++){
        if(final_arr[i] < final_arr[j]){
            int temp = final_arr[i];
            final_arr[i] = final_arr[j];
            final_arr[j] = temp;
        }
    }
}

printf("@ Final Merged & Sorted array is: ");
for(int i=0; i<10; i++){
    printf("%d ", final_arr[i]);
}
printf("\n");
printf("\n");

pthread_exit(NULL);
}

void *individual_sort(){
    for(int i=0; i<5; i++){
        for(int j=0; j<5; j++){
            if(arr_first_half[i] < arr_first_half[j]){
                int temp = arr_first_half[i];
                arr_first_half[i] = arr_first_half[j];
                arr_first_half[j] = temp;
            }
        }
    }
}

```

```

        }
        if(arr_second_half[i] < arr_second_half[j]){
            int temp = arr_second_half[i];
            arr_second_half[i] = arr_second_half[j];
            arr_second_half[j] = temp;
        }
    }
}

pthread_exit(NULL);
}

int main()
{
    printf("\n");

    /*
    printf("Enter 10 elements in the array: ");
    for(int i=0; i<10; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("\n");
    */

    printf("# Initial input array is: ");
    for(int i=0; i<10; i++){
        printf("%d ", arr[i]);
    }

```

```
printf("\n");
```

```
for(int i=0; i<5; i++){  
    arr_first_half[i] = arr[i];  
    arr_second_half[i] = arr[i+5];  
}
```

```
pthread_t parent_thread;  
pthread_create(&parent_thread, NULL, individual_sort, NULL);  
pthread_join(parent_thread, NULL);
```

```
printf("* First half sorted array is: ");  
for(int i=0; i<5; i++){  
    printf("%d ", arr_first_half[i]);  
}  
printf("\n");
```

```
printf("* Second half sorted array is: ");  
for(int i=0; i<5; i++){  
    printf("%d ", arr_second_half[i]);  
}  
printf("\n");
```

```
pthread_create(&parent_thread, NULL, final_merge_sort, NULL);  
pthread_join(parent_thread, NULL);
```

```
return 0;
```

```
}
```

Step4: - ./a.out

nano 6_4.c

Output of the program

```

(shreygrg@Linux23)-[~]
$ nano Exp6_4.c

(shreygrg@Linux23)-[~]
$ gcc Exp6_4.c

(shreygrg@Linux23)-[~]
$ ./a.out

# Initial input array is: 99 22 0 88 11 100 33 66 44 55
* First half sorted array is: 0 11 22 88 99
* Second half sorted array is: 33 44 55 66 100
# Merged array is: 0 11 22 88 99 33 44 55 66 100
@ Final Merged & Sorted array is: 0 11 22 33 44 55 66 88 99 100

```

[illegible]

Submitted to: -

Mr. Ashish Kumar Singh

Section: K22QY

Submitted By: -

**Shrey Garg
(12218692)**

**Merugu Srikar
(12220269)**

**Soumya Srivastava
(12203355)**