Name: Ayush. O. Chougule.
Rollno: 06 TY-CSE 01
PRN: 1032231720

OS

Real 12/8/25

Assingment - 02

* Problem statement:
Write a program using a fork to create a child process.

a). The parent process should sort number in ascending order and child process should sort numbers in descending order.
b) Orphan process
c). Zombie process

* Objective:
i) To understand the creation of process using the fork() system call.
ii) To observe and differentiate between zombie and orphan procen through coding examples.

* Theory

i) Fork() system call, Orphan procen, Zombie procen.

→ i) Fork(), system call: It is used in linux/Unix to create a new process called Child procen.
It duplicates parent procen copying it's memory, enviornment and open file descriptions. After a successful "fork()" 2 procen run independently. The parent procen recieves "PID" of child, while child gets 0 as the return Value.

ii) Orpan procen: Is a Child procen whose parent procen has terminated before child procen finishes execution
The system's init process (PID 1) adops the orphan process. This ensures Child process can complete execution properly and its resources can be cleaned up afterward.

iii) Zombie process: Is the one that has completed execution but still
in the process table because it's parent has'nt ye
read it's exit status using wait(). It doesn't consume CPU but do
occupy system resources unnecessarily.

* Input: 1 9 10 -2 3 5 4 -6 7 8

Output: Ascending order: -6 -2 1 3 4 5 7 8 9 10
Descending order: 10 9 8 7 5 4 3 1 -2 -6

* Conclusion: Thus, we have studied the concept of process control, orphan and zombie process and different system calls

* FAQ's:

1]. Explain the 5 state process model with neat diagram!

→ New : Process is being Created
Ready : Process is waiting to be assinged to CPU.
Running : Process is being executed.
Waiting : Process is waiting for an event (like I/O)
Terminated : Process has finished execution.

2] Explain mode switching and Context switching.

→ - mode switching: Refers to transition between user mode and kernel mode in CPU. It happens when a user process requests system-level operations (eg = file access) or when interrupts occurs. In kernel mode, the CPU can access protected system resources.

- Context switching: Is the process of saving current state (context) of a running process and loading state of another process. It enables multitasking by allowing CPU to switch between multiple process. Context switching involves saving registers, program counters and memory info. Though necessary it adds overhead and can affect performance frequently.

3] How can you eliminate zombie process?

→ zombie process can be eliminated by ensuring the parent process calls the wait() or waitpid() system call to read the child's exit status. Alternatively, the parent can be terminated so that the 'init' process adopts and cleans up the zombie.

4] Why do orphan process get adopted by 'init' process?

→ orphan process are adopted by the 'init' process (PID 1) to ensure proper resource cleanup. 'init' monitors and calls wait() on adopted process, preventing them from becoming zombies and keeping the system's process table clean and efficient.

Code:

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

int main()
{
   int n;
   printf("Enter number of elements: ");
   scanf("%d", &n);

   int arr[n];
   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++)
      scanf("%d", &arr[i]);

   pid_t num_pid = fork();

   if(num_pid < 0) {
      printf("error in fork execution");
   }

   else if(num_pid == 0) {
      for (int i = 0; i < n-1; i++)
         for (int j = 0; j < n-i-1; j++)
            if (arr[j] < arr[j+1]) {
               int temp = arr[j];
               arr[j] = arr[j+1];
               arr[j+1] = temp;
            }

      printf("\nChild Sorted in Descending Order:\n");
      for (int i = 0; i < n; i++)
         printf("%d ", arr[i]);
      printf("\n");

      printf("this is the child process id %d\n", getpid());
      printf("this is the parent of child process id %d\n", getppid());
   }

   else {
      for (int i = 0; i < n-1; i++)
         for (int j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1]) {
```

```
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }

    printf("\nParent Sorted in Ascending Order:\n");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    printf("this is the parent id %d\n", getpid());
    printf("this is the parent of parent id %d\n", getppid());
    wait(NULL);
  }

  exit(0);
}
```

Output: