

Title: Process control system calls: The demonstration of FORK and WAIT system calls along with zombie and orphan states.

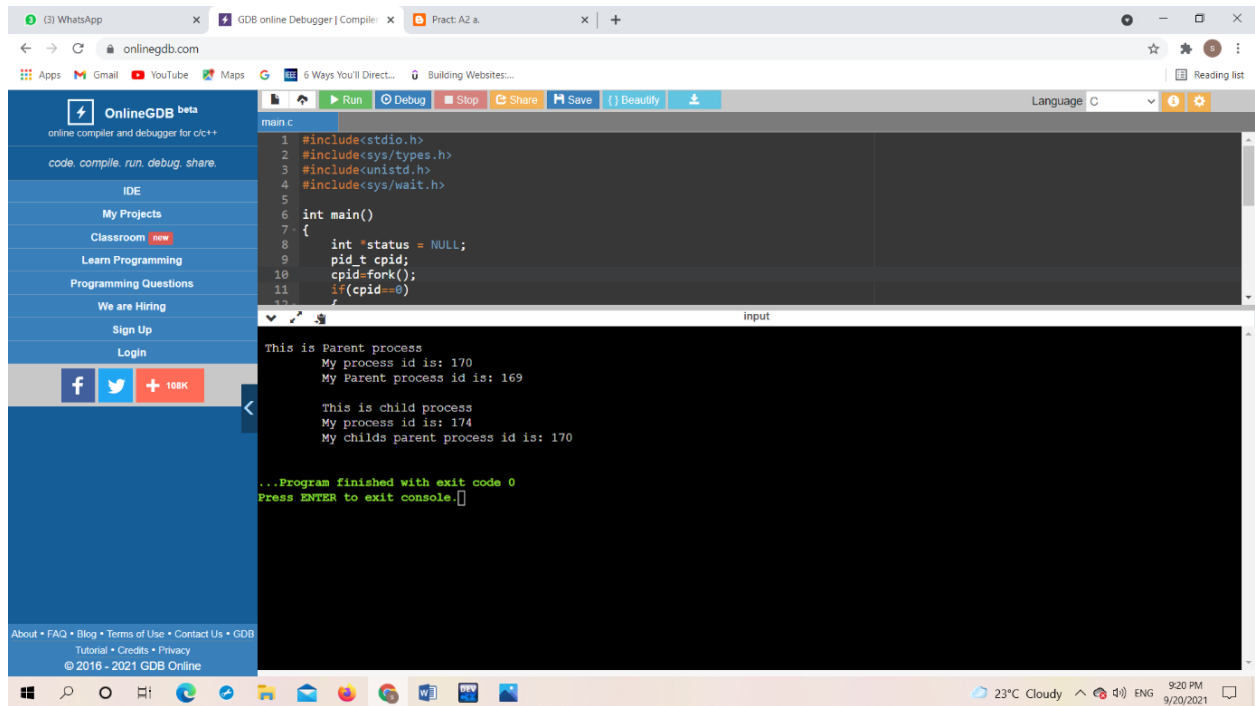
Problem Statement: a. Implement the C program in which main program accepts the integers to be sorted. Main program uses the FORK system call to create a new process called a child process. Parent process sorts the integers using a sorting algorithm and waits for the child process using the WAIT system call to sort the integers using any sorting algorithm. Also demonstrate zombie and orphan states.

//For Zombie State:

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

int main()
{
    int *status = NULL;
    pid_t cpid;
    cpid = fork();
    if (cpid == 0)
    {
        sleep(5);
        printf("\n\t This is child process");
        printf("\n\t My process id is: %d", getpid());
        printf("\n\t My childs parent process id is: %d\n", getppid());
    }
    else
    {
        sleep(5);
        printf("\n This is Parent process");
        printf("\n\t My process id is: %d", getpid());
        printf("\n\t My Parent process id is: %d\n", getppid());
    }
    return 0;
}
```

OUTPUT:-



The screenshot shows the OnlineGDB beta IDE interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, We are Hiring, Sign Up, and Login. The main editor displays a C program that uses `fork()` to create a child process. The output window shows the execution results, including process IDs and a confirmation that the program finished with exit code 0.

```
main.c
1 #include<stdio.h>
2 #include<sys/types.h>
3 #include<unistd.h>
4 #include<sys/wait.h>
5
6 int main()
7 {
8     int *status = NULL;
9     pid_t cpid;
10    cpid=fork();
11    if(cpid==0)
12    {
13
14    }
15    else
16    {
17
18    }
19    wait(status);
20    printf("Parent process id is: %d\n", getpid());
21    printf("Child process id is: %d\n", cpid);
22    return 0;
23 }
```

Input

```
This is Parent process
My process id is: 170
My Parent process id is: 169

This is child process
My process id is: 174
My childs parent process id is: 170

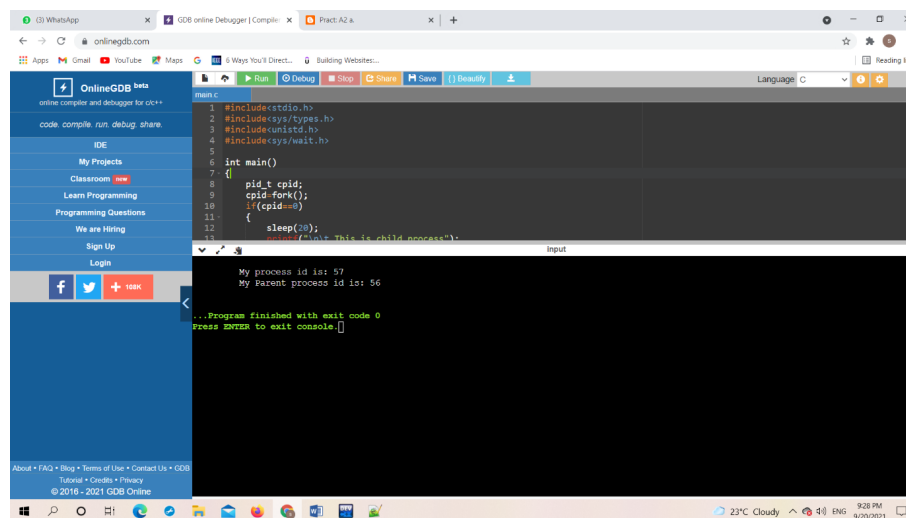
...Program finished with exit code 0
Press ENTER to exit console.
```

23°C Cloudy 9:20 PM 9/20/2021

//For Orphan state

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

int main()
{
    pid_t cpid;
    cpid = fork();
    if (cpid == 0)
    {
        sleep(20);
        printf("\n\t This is child process");
        printf("\n\t My process id is: %d", getpid());
        printf("\n\t My Parent process id is: %d\n", getppid());
    }
    else
    {
        sleep(2);
        printf("\n\t My process id is: %d", getpid());
        printf("\n\t My Parent process id is: %d\n", getppid());
    }
    return 0;
}
```



The screenshot shows the OnlineGDB web interface. The code from the previous block is pasted into the editor. The output window shows the following text:

```
My process id is: 57
My Parent process id is: 56
...Program finished with exit code 0
Press ENTER to exit console.
```

The status bar at the bottom indicates the program finished with exit code 0.

//Bubble sort

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

void bubble(int arr[], int n)
{
    int i, j, k;
    for (i = 0; i < (n - 1); i++)
    {
        for (j = 0; j < (n - i - 1); j++)
        {
            if (arr[j] > arr[j + 1])
            {
                k = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = k;
            }
        }
    }
}

void display(int arr[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
}

int main()
{
    pid_t id;
    int arr[20], n, i;
    printf("\nHow many elements do you want to sort?:");
    scanf("%d", &n);
    printf("\nEnter the elements:\n");

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

```

{
    scanf("%d", &arr[i]);
}

id = fork();

if (id == 0)
{
    printf("\nSorting in child process:");
    bubble(arr, n);
    display(arr, n);
}
else
{
    printf("\nSorting in parent process:");
    bubble(arr, n);
    display(arr, n);
}
}

```

The screenshot shows the OnlineGDB web interface. The code editor contains the same C program. The console output is as follows:

```

How many elements do you want to sort?:5
Enter the elements:
87
34
75
43
98

Sorting in parent process:34 43 75 87 98
Sorting in child process:34 43 75 87 98

...Program finished with exit code 0
Press ENTER to exit console.

```

The interface also shows a sidebar with navigation links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'We are Hiring', 'Sign Up', and 'Login'. The bottom status bar indicates the system temperature as 24°C and the date as 9/20/2021.