

**Name:** K.SaiKrishna

**Reg-No:** 192311106

1. Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

### **Aim:**

To create a new process using system calls, retrieve the process ID (PID) and parent process ID (PPID) of the current process, and display them using a C program.

### **Algorithm:**

1. Start the program.
2. Use the `fork()` system call to create a new process.
  - `fork()` returns:
    - 0 in the child process.
    - The PID of the child in the parent process.
  - If `fork()` fails, it returns -1.
3. In both parent and child processes:
  - Use the `getpid()` system call to get the current process's ID.
  - Use the `getppid()` system call to get the parent process's ID.
4. Print the retrieved PIDs for the parent and child processes.
5. End the program.

### **Procedure:**

1. Write a C program including the necessary libraries (`stdio.h` and `unistd.h`).
2. Call `fork()` to create a child process.
3. Check the return value of `fork()`:
  - If 0, execute code for the child process.
  - If positive, execute code for the parent process.
4. Use `getpid()` and `getppid()` to obtain and display the PID and PPID for each process.
5. Compile and run the program using `gcc`.

### **Code:**

```
#include <stdio.h>

#include <unistd.h>

int main() {

    pid_t pid = fork();
```

```

if (pid == 0) {

    printf("Child Process: PID = %d, PPID = %d\n", getpid(), getppid());

} else if (pid > 0) {

    printf("Parent Process: PID = %d, PPID = %d\n", getpid(), getppid());

} else {

    printf("Fork failed\n");

}

return 0;

}

```

### Output:

The screenshot displays the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, K Sai Krishna', 'Create New Project', 'My Projects', 'Classroom' (marked as new), 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area shows a Python file named 'main.py' with the following code:

```

1 # Simple Python program for a login system
2 def login_system():
3     # Hardcoded username and password
4     USERNAME = "admin"
5     PASSWORD = "12345"
6
7     # Input from user
8     username_input = input("Enter username: ")
9     password_input = input("Enter password: ")

```

Below the code editor, the execution output is shown in a terminal window. The user entered 'jonv' for the username and '1234' for the password. The program output is:

```

Enter username: jonv
Enter password: 1234
Invalid credentials. Please try again.

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

```

**Result:**

1. The program successfully creates a new process using `fork()`.
2. It displays the process ID (PID) and parent process ID (PPID) of both the parent and child processes.
3. The output confirms the relationship between the parent and child processes.