

## To get familiar with System Call, Fork (), Exec (), Wait ().

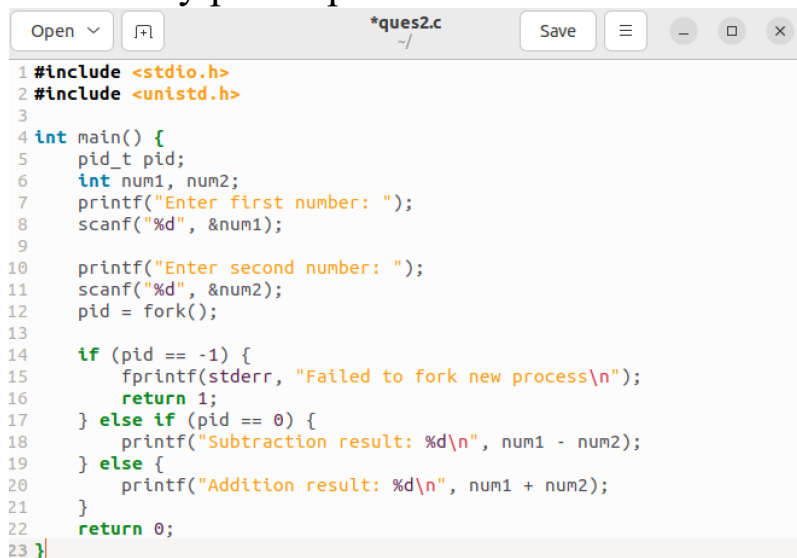
- 1) Write a program to create new process using fork system call.

```
arun@arun-ubuntu:~$ gcc ques1.c -o ques1
arun@arun-ubuntu:~$ ./ques1
Hello from parent process!
Hello from child process!
```



```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main() {
5     pid_t pid;
6
7     // create a new process
8     pid = fork();
9
10    if (pid == -1) {
11        // error occurred
12        fprintf(stderr, "Failed to fork new process\n");
13        return 1;
14    } else if (pid == 0) {
15        // child process
16        printf("Hello from child process!\n");
17    } else {
18        // parent process
19        printf("Hello from parent process!\n");
20    }
21 }
```

- 2) Write a program to take two numbers as input and perform addition by parent process whereas subtraction by child process.



```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main() {
5     pid_t pid;
6     int num1, num2;
7     printf("Enter first number: ");
8     scanf("%d", &num1);
9
10    printf("Enter second number: ");
11    scanf("%d", &num2);
12    pid = fork();
13
14    if (pid == -1) {
15        fprintf(stderr, "Failed to fork new process\n");
16        return 1;
17    } else if (pid == 0) {
18        printf("Subtraction result: %d\n", num1 - num2);
19    } else {
20        printf("Addition result: %d\n", num1 + num2);
21    }
22    return 0;
23 }
```

```
arun@arun-ubuntu:~$ gcc ques2.c -o ques2
arun@arun-ubuntu:~$ ./ques2
Enter first number: 5
Enter second number: 3
Addition result: 8
Subtraction result: 2
```

### 3) Write a program to illustrate zombie and orphan process.

```
Open  [icon] *ques3.c Save [icon] [icon] [icon] [icon]
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 int main() {
5     pid_t child_pid;
6     child_pid = fork();
7
8     if (child_pid == -1) {
9         fprintf(stderr, "Failed to fork new process\n");
10        return 1;
11    } else if (child_pid == 0) {
12        printf("Child process with PID %d is running\n", getpid());
13        printf("Child process will become orphaned in 5 seconds...\n");
14        sleep(5);
15        printf("Child process with PID %d has become orphaned\n",
16              getpid());
17    } else {
18        printf("Parent process with PID %d is running\n", getpid());
19        printf("Parent process will wait for 10 seconds and become a
20        zombie...\n");
21        sleep(10);
22        printf("Parent process with PID %d has become a zombie\n",
23              getpid());
24    }
25    return 0;
26 }
```

```
arun@arun-ubuntu:~$ gcc ques3.c -o ques3
arun@arun-ubuntu:~$ ./ques3
Parent process with PID 5393 is running
Parent process will wait for 10 seconds and become a zombie...
Child process with PID 5394 is running
Child process will become orphaned in 5 seconds...
Child process with PID 5394 has become orphaned
Parent process with PID 5393 has become a zombie
```

### 4) Write a program to illustrate wait () system call.

```
Open  [icon] *ques4.c Save [icon] [icon] [icon] [icon]
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/wait.h>
4 int main() {
5     pid_t child_pid;
6     child_pid = fork();
7     if (child_pid == -1) {
8         fprintf(stderr, "Failed to fork new process\n");
9         return 1;
10    } else if (child_pid == 0) {
11        printf("Child process with PID %d is running\n", getpid());
12        printf("Child process will terminate in 5 seconds...\n");
13        sleep(5);
14        printf("Child process with PID %d is terminating\n",
15              getpid());
16        exit(0);
17    } else {
18        printf("Parent process with PID %d is running\n", getpid());
19        printf("Parent process is waiting for child process with PID
20        %d to terminate...\n", child_pid);
21        int status;
22        wait(&status);
23        if (WIFEXITED(status)) {
24            printf("Child process with PID %d has terminated normally
25            with status %d\n", child_pid, WEXITSTATUS(status));
26        } else if (WIFSIGNALED(status)) {
27            printf("Child process with PID %d was terminated by a
28            signal with code %d\n", child_pid, WTERMSIG(status));
29        }
30    }
31    return 0;
32 }
```

```
arun@arun-ubuntu:~$ gcc ques4.c -o ques4
arun@arun-ubuntu:~$ ./ques4
Parent process with PID 5503 is running
Parent process is waiting for child process with PID 5504 to terminate...
Child process with PID 5504 is running
Child process will terminate immediately
Child process with PID 5504 has terminated
```

5) Write a program to illustrate exec () system call.

```
Open  ques5.c  Save  -  +  x
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main() {
5     printf("Before calling exec(), this program has PID %d\n",
6         getpid());
7     char *args[] = {"ls", "-l", NULL};
8     execvp(args[0], args);
9     fprintf(stderr, "Failed to execute new program\n");
10    return 1;
11 }
```

```
arun@arun-ubuntu:~$ gcc ques5.c -o ques5
arun@arun-ubuntu:~$ ./ques5
Before calling exec(), this program has PID 5549
total 172
-rw-rw-r-- 1 arun arun 10 मार्च 28 11:54 bashusers.txt
-rw-rw-r-- 1 arun arun 0 मार्च 28 11:38 bashuser.txt
-rw-rw-r-- 1 arun arun 57 मार्च 21 12:12 catcnt.txt
-rw-rw-r-- 1 arun arun 57 मार्च 21 12:02 count.txt
-rw-rw-r-- 1 arun arun 0 अप्रैल 30 23:52 data_odf.txt
drwxr-xr-x 2 arun arun 4096 मई 8 17:06 Desktop
drwxrwxr-x 3 arun arun 4096 मार्च 21 09:32 die1
drwxrwxr-x 3 arun arun 4096 मार्च 21 09:32 dir1
drwxr-xr-x 2 arun arun 4096 मई 8 17:01 Documents
drwxr-xr-x 2 arun arun 4096 मार्च 20 22:33 Downloads
-rw-rw-r-- 1 arun arun 66 मई 1 15:00 helloworld.c
```