

WALCHAND COLLEGE OF ENGINEERING SANGLI

(A Government Aided Autonomous College)



Department of Information Technology

UNIX OPERATING SYSTEM LAB (5IT)

Year of Studentship: 2021-22

Term: Semester-II

Class: T.Y. IT	
Name	PRN. No
Rushikesh Randive	2019BTEIT00008

Submission Date: 07/04/2022

Guided By: Mr. A.J.Umbarkar sir

UNIX Operating System Assignment.

Fork System call():

What is a Fork()?

In the computing field, `fork()` is the primary method of process creation on Unix-like operating systems. This function creates a new copy called the *child* out of the original process, that is called the *parent*. When the parent process closes or crashes for some reason, it also kills the child process.

System call `fork()` is used to create processes. It takes no arguments and returns a process ID. The purpose of `fork()` is to create a new process, which becomes the child process of the caller. After a new child process is created, both processes will execute the next instruction following the `fork()` system call. Therefore, we have to distinguish the parent from the child.

This can be done by testing the returned value of `fork()`:

1. If `fork()` returns a negative value, the creation of a child process was unsuccessful.
2. `fork()` returns a zero to the newly created child process.
3. `fork()` returns a positive value, the process ID of the child process, to the parent. The returned process ID is of type `pid_t` defined in `sys/types.h`. Normally, the process ID is an integer. Moreover, a process can use function `getpid()` to retrieve the process ID assigned to this process.
4. Therefore, after the system call to `fork()`, a simple test can tell which process is the child.

Codes:

Program 1

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    /* fork a process */
    fork();
    /* the child and parent will execute every line of code after the fork (each separately)*/
    printf("Hello world!\n");
    return 0;
}
```

OUTPUT

```
Hello world!  
Hello world!
```

Program 2

```
#include <stdio.h>  
#include <sys/types.h>  
#include <unistd.h>  
#include <stdlib.h>  
int main()  
{  
    fork();  
    fork();  
    fork();  
    printf("Hello world!\n");  
    return 0;  
}
```

OUTPUT

```
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!
```

Program 3

```
#include <stdio.h>  
#include <sys/types.h>  
#include <unistd.h>  
#include <stdlib.h>  
  
int main() {  
    if(fork() == 0)  
        if(fork())
```

```
    printf("Hello world!!\n");
    exit(0);
}
```

OUTPUT

```
gcc fork1.c
./a.out
Hello world!!
```

Program 4

```
/* Note that write() is used instead of printf() since the */
/* latter is buffered while the former is not.           */
/*                                                         */

#include <stdio.h>
#include <string.h>
#include <sys/types.h>

#define MAX_COUNT 200
#define BUF_SIZE 100

void main(void)
{
    pid_t pid;
    int i;
    char buf[BUF_SIZE];

    fork();
    pid = getpid();
    for (i = 1; i <= MAX_COUNT; i++) {
        sprintf(buf, "This line is from pid %d, value = %d\n", pid, i);
        write(1, buf, strlen(buf));
    }
}

// make two identical copies of address spaces, one for the parent and the other for the child.
//Both processes will start their execution at the next statement following the fork() call.
//In this case, both processes will start their execution at the assignment statement
// Both processes start their execution right after the system call fork().
//both processes have identical but separate address spaces, those variables initialized before the
//fork() call have the same values in both address spaces.
//if the parent changes the value of its variable, the modification will only affect the variable in the
parent process's address space.
//Other address spaces created by fork() calls will not be affected even though they have identical
variable names.
```

OUTPUT

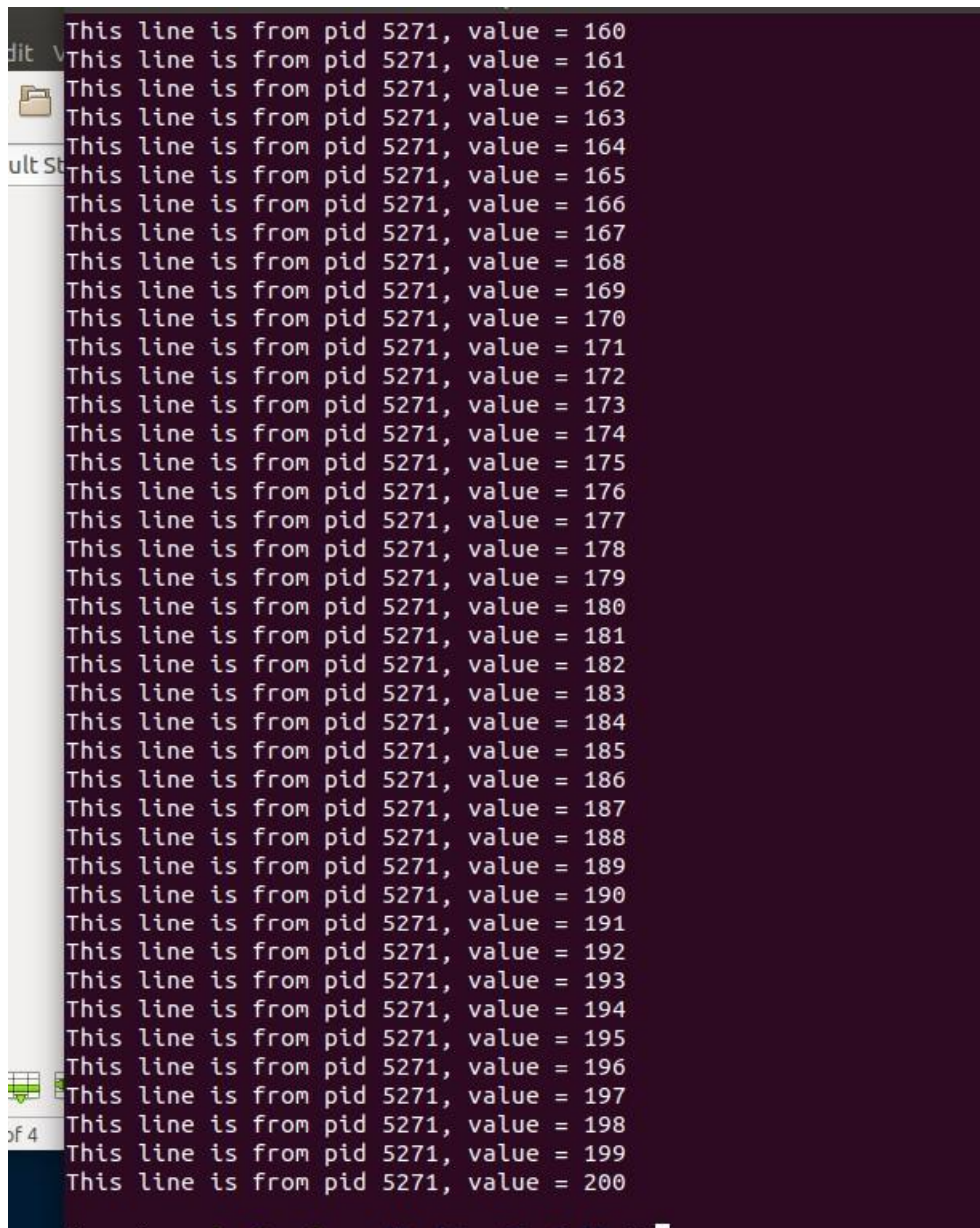
```
This line is from pid 5270, value = 1
This line is from pid 5270, value = 2
This line is from pid 5270, value = 3
This line is from pid 5270, value = 4
This line is from pid 5270, value = 5
This line is from pid 5270, value = 6
This line is from pid 5270, value = 7
This line is from pid 5270, value = 8
This line is from pid 5270, value = 9
This line is from pid 5270, value = 10
This line is from pid 5270, value = 11
This line is from pid 5270, value = 12
This line is from pid 5270, value = 13
This line is from pid 5270, value = 14
This line is from pid 5270, value = 15
This line is from pid 5270, value = 16
This line is from pid 5270, value = 17
This line is from pid 5270, value = 18
This line is from pid 5270, value = 19
This line is from pid 5270, value = 20
This line is from pid 5270, value = 21
This line is from pid 5270, value = 22
This line is from pid 5270, value = 23
This line is from pid 5270, value = 24
This line is from pid 5270, value = 25
This line is from pid 5270, value = 26
This line is from pid 5270, value = 27
```

```
This line is from pid 5270, value = 25
This line is from pid 5270, value = 26
This line is from pid 5270, value = 27
This line is from pid 5270, value = 28
This line is from pid 5270, value = 29
This line is from pid 5270, value = 30
This line is from pid 5270, value = 31
This line is from pid 5270, value = 32
This line is from pid 5270, value = 33
This line is from pid 5270, value = 34
This line is from pid 5270, value = 35
This line is from pid 5270, value = 36
This line is from pid 5270, value = 37
This line is from pid 5270, value = 38
This line is from pid 5270, value = 39
This line is from pid 5270, value = 40
This line is from pid 5270, value = 41
This line is from pid 5270, value = 42
This line is from pid 5270, value = 43
This line is from pid 5270, value = 44
This line is from pid 5270, value = 45
This line is from pid 5270, value = 46
This line is from pid 5270, value = 47
This line is from pid 5270, value = 48
This line is from pid 5270, value = 49
This line is from pid 5270, value = 50
This line is from pid 5270, value = 51
This line is from pid 5270, value = 52
This line is from pid 5270, value = 53
This line is from pid 5270, value = 54
This line is from pid 5270, value = 55
This line is from pid 5270, value = 56
This line is from pid 5270, value = 57
This line is from pid 5270, value = 58
This line is from pid 5270, value = 59
This line is from pid 5270, value = 60
This line is from pid 5270, value = 61
This line is from pid 5270, value = 62
This line is from pid 5270, value = 63
This line is from pid 5270, value = 64
This line is from pid 5270, value = 65
This line is from pid 5270, value = 66
This line is from pid 5270, value = 67
```



```
This line is from pid 5270, value = 65
This line is from pid 5270, value = 66
This line is from pid 5270, value = 67
This line is from pid 5270, value = 68
This line is from pid 5270, value = 69
This line is from pid 5270, value = 70
This line is from pid 5270, value = 71
This line is from pid 5270, value = 72
This line is from pid 5270, value = 73
This line is from pid 5270, value = 74
This line is from pid 5270, value = 75
This line is from pid 5270, value = 76
This line is from pid 5270, value = 77
This line is from pid 5270, value = 78
This line is from pid 5270, value = 79
This line is from pid 5270, value = 80
This line is from pid 5270, value = 81
This line is from pid 5270, value = 82
This line is from pid 5270, value = 83
This line is from pid 5270, value = 84
This line is from pid 5270, value = 85
This line is from pid 5270, value = 86
This line is from pid 5270, value = 87
This line is from pid 5270, value = 88
This line is from pid 5270, value = 89
This line is from pid 5270, value = 90
This line is from pid 5270, value = 91
This line is from pid 5270, value = 92
This line is from pid 5270, value = 93
This line is from pid 5270, value = 94
This line is from pid 5270, value = 95
This line is from pid 5270, value = 96
This line is from pid 5270, value = 97
This line is from pid 5270, value = 98
This line is from pid 5270, value = 99
This line is from pid 5270, value = 100
This line is from pid 5270, value = 101
This line is from pid 5270, value = 102
This line is from pid 5270, value = 103
This line is from pid 5270, value = 104
This line is from pid 5270, value = 105
This line is from pid 5270, value = 106
This line is from pid 5270, value = 107
```

```
This line is from pid 5270, value = 190
This line is from pid 5270, value = 191
This line is from pid 5270, value = 192
This line is from pid 5270, value = 193
This line is from pid 5270, value = 194
This line is from pid 5270, value = 195
This line is from pid 5270, value = 196
This line is from pid 5270, value = 197
This line is from pid 5270, value = 198
This line is from pid 5270, value = 199
This line is from pid 5270, value = 200
This line is from pid 5271, value = 1
This line is from pid 5271, value = 3
This line is from pid 5271, value = 4
This line is from pid 5271, value = 5
This line is from pid 5271, value = 6
This line is from pid 5271, value = 7
This line is from pid 5271, value = 8
This line is from pid 5271, value = 9
This line is from pid 5271, value = 10
This line is from pid 5271, value = 11
This line is from pid 5271, value = 12
This line is from pid 5271, value = 13
This line is from pid 5271, value = 14
This line is from pid 5271, value = 15
This line is from pid 5271, value = 16
This line is from pid 5271, value = 17
This line is from pid 5271, value = 18
This line is from pid 5271, value = 19
This line is from pid 5271, value = 20
This line is from pid 5271, value = 21
This line is from pid 5271, value = 22
This line is from pid 5271, value = 23
This line is from pid 5271, value = 24
This line is from pid 5271, value = 25
This line is from pid 5271, value = 26
```

```
This line is from pid 5271, value = 160
This line is from pid 5271, value = 161
This line is from pid 5271, value = 162
This line is from pid 5271, value = 163
This line is from pid 5271, value = 164
This line is from pid 5271, value = 165
This line is from pid 5271, value = 166
This line is from pid 5271, value = 167
This line is from pid 5271, value = 168
This line is from pid 5271, value = 169
This line is from pid 5271, value = 170
This line is from pid 5271, value = 171
This line is from pid 5271, value = 172
This line is from pid 5271, value = 173
This line is from pid 5271, value = 174
This line is from pid 5271, value = 175
This line is from pid 5271, value = 176
This line is from pid 5271, value = 177
This line is from pid 5271, value = 178
This line is from pid 5271, value = 179
This line is from pid 5271, value = 180
This line is from pid 5271, value = 181
This line is from pid 5271, value = 182
This line is from pid 5271, value = 183
This line is from pid 5271, value = 184
This line is from pid 5271, value = 185
This line is from pid 5271, value = 186
This line is from pid 5271, value = 187
This line is from pid 5271, value = 188
This line is from pid 5271, value = 189
This line is from pid 5271, value = 190
This line is from pid 5271, value = 191
This line is from pid 5271, value = 192
This line is from pid 5271, value = 193
This line is from pid 5271, value = 194
This line is from pid 5271, value = 195
This line is from pid 5271, value = 196
This line is from pid 5271, value = 197
This line is from pid 5271, value = 198
This line is from pid 5271, value = 199
This line is from pid 5271, value = 200
```

Advanced example

When a process creates a new process, then there are two possibilities for the execution exit:

- The parent continues to execute concurrently with its child.
- The parent waits until some or all of its children have terminated.

Program 5

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

```

int main(int argc, char *argv[]) {

    /* fork a child process */
    pid_t pid = fork();

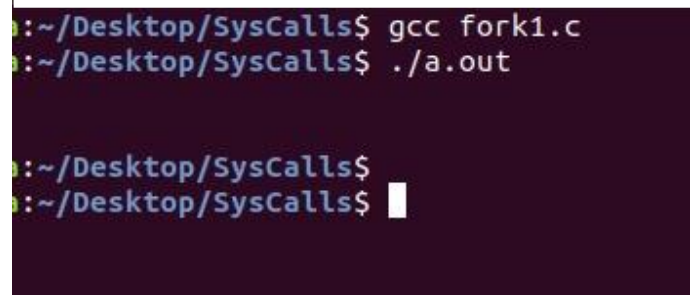
    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        printf("I'm the child \n"); /* you can execute some commands here */
    }

    else { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        /* When the child is ended, then the parent will continue to execute its code */
        printf("Child Complete \n");
    }
}

// The wait call system wait(NULL) will make the parent process wait until the child process has
executed all of its commands.

```

OUTPUT



```

~/Desktop/SysCalls$ gcc fork1.c
~/Desktop/SysCalls$ ./a.out

~/Desktop/SysCalls$
~/Desktop/SysCalls$

```

Program 6

```

#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    printf("I am: %d\n", (int) getpid());

    pid_t pid = fork();

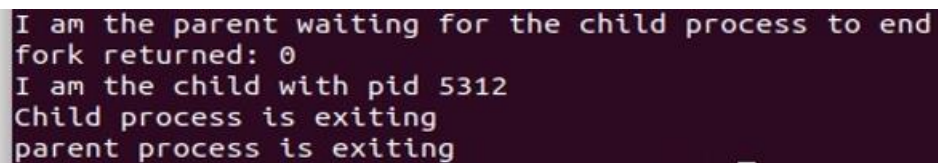
```

```
printf("fork returned: %d\n", (int) pid);

if (pid < 0) { /* error occurred */
    perror("Fork failed");
}
if (pid == 0) { /* child process */
    printf("I am the child with pid %d\n", (int) getpid());
    printf("Child process is exiting\n");
    exit(0);
}
/* parent process */
printf("I am the parent waiting for the child process to end\n");
wait(NULL);
printf("parent process is exiting\n");
return(0);
}

// When the main program executes fork(), an identical copy of its address space, including the
// program and all data, is created.
//System call fork() returns the child process ID to the parent and returns 0 to the child process.
```

OUTPUT



```
I am the parent waiting for the child process to end
fork returned: 0
I am the child with pid 5312
Child process is exiting
parent process is exiting
```