

## Lab 3: Inter-process Communication in C (5 Marks)

Student Name: *Muhammad 'Asif Danial Bin Mohd Shahir*

Github Repository for Lab 3: <https://github.com/asfdnl/ITT440-Lab3>

Matric No: 2021114813

Group: CS2554A

### 3.1 IPC using Signals

Signals, to be short, are various notifications sent to a process in order to notify it of various "important" events. Each signal has an integer number that represents it (1, 2 and so on), as well as a symbolic name that is usually defined in the file `/usr/include/signal.h` or one of the files included by it directly or indirectly such as:

- INT
- HUP
- TSTP
- ABRT

### Sending signal to process using Keyboard

- Ctrl + C  
Pressing this key causes the system to send an INT signal (SIGINT) to the running process. By default, this signal causes the process to immediately terminate.
- Ctrl + Z  
Pressing this key causes the system to send a TSTP signal (SIGTSTP) to the running process. By default, this signal causes the process to suspend execution
- Ctrl + \  
Pressing this key causes the system to send a SIGQUIT signal (SIGQUIT) to the running process.
- Ctrl + D  
Pressing this key causes the system to send a EOF signal (EOF) to the running process.

Please compile and run the following code on your Ubuntu/Linux. Press Ctrl + C on your keyboard while the program is running. Observe the output

```
#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <signal.h>

int main(void)
{

    void sigint_handler(int sig);

    char s[200];

    if (signal(SIGINT, sigint_handler) == SIG_ERR){

        perror("signal");

        exit(1);

    }

    printf("Enter a string:\n");

    if (fgets(s, 200, stdin) == NULL)

        perror("gets");

    else

        printf("you entered: %s\n", s);

    return 0;

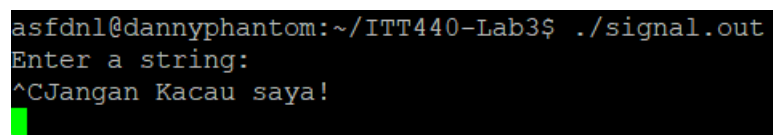
}

void sigint_handler(int sig)
{

    printf("Jangan Kacau saya! \n");

}
```

What is the terminal output when you press Ctrl + C on your keyboard? Please provide screenshot. Please briefly explain why **(0.5 Marks)**



```
asfdnl@dannyphantom:~/ITT440-Lab3$ ./signal.out
Enter a string:
^CJangan Kacau saya!
```

The program is coded to send the output when INT signal is received causes the process to immediately terminate.

Now based on above code, write a program that will capture all these Signals: SIGINT, SIGTSTP and SIGQUIT. When a signal is received, program will output: **“This is a special signal handler for <signal>”** substitute <signal> with the correct signal. Please provide screenshot of your program output **(0.5 Marks)**

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ./signalTSTP.out
Enter a string:
^ZCtrl+Z pressed!
```

### SIGTSTP

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ./signalQUIT.out
Enter a string:
^\\SIGQUIT pressed!
```

### SIGQUIT

## 3.2IPC using Pipe in Shell/Terminal

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing.

We can use pipe to redirect the output of one command to another command in our terminal/shell. The linkage between the two commands is facilitated by the GNU/Linux kernel, which takes care of connecting the two together.

Please run all of the following command on your linux terminal and observe the output.

```
$ ls -l | more
```

```
$ sort /etc/passwd | uniq
```

```
$ cat /etc/group | head -7 | tail -5
```

```
$ ls -l | find ./ -type f -name "*.txt" -exec grep "program" {} \;
```

```
$ cat /etc/passwd | grep "name" | tee file2.txt | wc -l
```

```
$ ifconfig | awk '{match($0,/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)/); ip = substr($0,RSTART,RLENGTH); print ip}'
```

Please provide screenshot for the output for all the command stated above **(0.5 Marks)**

\$ ls -l | more

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ls -l | more
total 72
-rw-rw-r-- 1 asfdnl asfdnl 456 Oct 27 01:29 signalINT.c
-rwxrwxr-x 1 asfdnl asfdnl 17048 Oct 27 01:30 signalINT.out
-rw-rw-r-- 1 asfdnl asfdnl 458 Oct 27 01:37 signalQUIT.c
-rwxrwxr-x 1 asfdnl asfdnl 17056 Oct 27 01:37 signalQUIT.out
-rw-rw-r-- 1 asfdnl asfdnl 456 Oct 27 01:33 signalTSTP.c
-rwxrwxr-x 1 asfdnl asfdnl 17056 Oct 27 01:33 signalTSTP.out
```

\$ sort /etc/passwd | uniq

```
asfdnl@dannyphantom:~/ITT440-Lab3$ sort /etc/passwd | uniq
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
asfdnl:x:1000:1000:DannyPhantom:/home/asfdnl:/bin/bash
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
games:x:5:60:games:/usr/games:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
root:x:0:0:root:/root:/bin/bash
sshd:x:112:65534::/run/sshd:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
uuid:x:107:112::/run/uuid:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

\$ cat /etc/group | head -7 | tail -5

```
asfdnl@dannyphantom:~/ITT440-Lab3$ cat /etc/group | head -7 | tail -5
bin:x:2:
sys:x:3:
adm:x:4:syslog,asfdnl
tty:x:5:syslog
disk:x:6:
```

\$ ls -l | find ./ -type f -name "\*.txt" -exec grep "program" {} \;

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ls -l | find ./ -type f -name "*.txt" -exec g
rep "program" {} \;
asfdnl@dannyphantom:~/ITT440-Lab3$
```

\$ cat /etc/passwd | grep "name" | tee file2.txt | wc -l

```
asfdnl@dannyphantom:~/ITT440-Lab3$ cat /etc/passwd | grep "name" | tee file2.txt | wc -l
>
```

\$ ifconfig | awk '{match(\$0,/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)/); ip = substr(\$0,RSTART,RLENGTH); print ip}'

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ifconfig | awk '{match($0,/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)/); ip = substr($0,RSTART,RLENGTH); print ip}'
10.0.2.15

192.168.56.103

127.0.0.1
```

Now use what you have learn about pipe in terminal, please provide the command to find a line in ifconfig command manual page which match a word “down” and find the word count from the line output **(0.5 Marks)**

---

---

### 3.3 IPC using Pipe in C

The pipe() system function is used to open file descriptors, which are used to communicate between different Linux processes. The syntax of the pipe() function is:

**int pipe(int pipefd[2]);**

The first element of the pipefd array, pipefd[0] is used for reading data from the pipe. The second element of the pipefd array, pipefd[1] is used for writing data to the pipe. On success, the pipe() function returns 0. If an error occurs during pipe initialization, then the pipe() function returns -1.

We can use pipe to send an integer value or even a float value. Please compile and run the following code on your Ubuntu/Linux. Observe the output

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(void) {
    int pipefds[2];
    int buffer;

    if(pipe(pipefds) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    int nombor = 22;

    printf("Writing a number to pipe...\n");
    write(pipefds[1], &nombor, sizeof(nombor));
    printf("Done.\n\n");

    printf("Reading a number from pipe...\n");
    read(pipefds[0], &buffer, sizeof(buffer));
    printf("Done.\n\n");

    printf("Number from pipe: %d\n", buffer);

    return EXIT_SUCCESS;
}
```

Observe how the number is saved and retrieve back from the pipe. Please provide screenshot for the output of the program above **(0.5 Marks)**

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ./pipe.out
Writing a number to pipe...
Done.

Reading a number from pipe...
Done.

Number from pipe: 22
```

We can also use pipe to send a character or a string value. Please compile and run the following code on your Ubuntu/Linux. Observe the output

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(void) {
    int pipefds[2];
    char buffer[5];

    if(pipe(pipefds) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    char *pin = "4128\0";

    printf("Writing PIN to pipe...\n");
    write(pipefds[1], pin, 5);
    printf("Done.\n\n");

    printf("Reading PIN from pipe...\n");
    read(pipefds[0], buffer, 5);
    printf("Done.\n\n");

    printf("PIN from pipe: %s\n", buffer);

    return EXIT_SUCCESS;
}
```

Observe and try to understand how the character is saved and retrieve back from the pipe. Please provide screenshot for the output of the program above **(0.5 Marks)**

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ./pipeString.out
Writing PIN to pipe...
Done.

Reading PIN from pipe...
Done.

PIN from pipe: 4128
```

### 3.4 Pipe() and Fork()

In code above, we used to write and read data from pipe in the same process. But usually pipe is used for inter-process communication.

Please compile and run the following code on your Ubuntu/Linux. Observe how the child process generate a value and send it to the parent process.

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

#define PIN_LENGTH 4
#define PIN_WAIT_INTERVAL 2

void getPIN(char pin[PIN_LENGTH + 1]) {
    srand(getpid() + getppid());

    pin[0] = 49 + rand() % 7;

    for(int i = 1; i < PIN_LENGTH; i++) {
        pin[i] = 48 + rand() % 7;
    }

    pin[PIN_LENGTH] = '\0';
}

int main(void) {
    while(1) {
        int pipefds[2];
        char pin[PIN_LENGTH + 1];
        char buffer[PIN_LENGTH + 1];

        pipe(pipefds);

        pid_t pid = fork();

        if(pid == 0) {
            getPIN(pin); // generate PIN
            close(pipefds[0]); // close read fd
            write(pipefds[1], pin, PIN_LENGTH + 1); // write PIN to pipe

            printf("Generating PIN in child and sending to parent...\n");

            sleep(PIN_WAIT_INTERVAL); // delaying PIN generation
            intentionally.

            exit(EXIT_SUCCESS);
        }

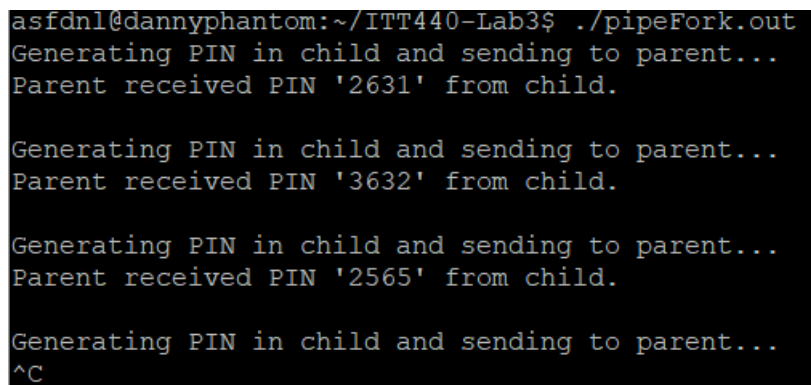
        if(pid > 0) {
            wait(NULL); // waiting for child to finish

            close(pipefds[1]); // close write fd
            read(pipefds[0], buffer, PIN_LENGTH + 1); // read PIN from pipe
            close(pipefds[0]); // close read fd
            printf("Parent received PIN '%s' from child.\n\n", buffer);
        }
    }

    return EXIT_SUCCESS;
}
```



You can stop the program from continuing creating a child and terminate the program by sending SIGINT signal by pressing Ctrl + C on your keyboard. Please provide screenshot for the output of the program above **(0.5 Marks)**



```
asfdnl@dannyphantom:~/ITT440-Lab3$ ./pipeFork.out
Generating PIN in child and sending to parent...
Parent received PIN '2631' from child.

Generating PIN in child and sending to parent...
Parent received PIN '3632' from child.

Generating PIN in child and sending to parent...
Parent received PIN '2565' from child.

Generating PIN in child and sending to parent...
^C
```

In real life scenario, we to use computer to calculate complex equation or create a server to handle multiple connection. Instead of computing things like that in the same process as the main program, you can just calculate the hash on a child process and return the hash to the main process. The child can return the value to the parent function by using pipes.

Please compile and run the following code on your Ubuntu/Linux. Observe how the child process generate a value and send it to the parent process.

```

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

int getPIN() {
    // use PPID and PID as the seed
    srand(getpid() + getppid());
    int secret = 1000 + rand() % 9000;
    return secret;
}

int main(void) {
    int fd[2];
    pipe(fd);
    pid_t pid = fork();

    if(pid > 0) {
        close(0);
        close(fd[1]);
        dup(fd[0]);

        int secretNumber;
        size_t readBytes = read(fd[0], &secretNumber, sizeof(secretNumber));
;
        printf("waiting for PIN...\n");
        wait(NULL);
        printf("Bytes read: %ld\n", readBytes);
        printf("PIN: %d\n", secretNumber);
    }
    else if(pid == 0) {
        close(1);
        close(fd[0]);
        dup(fd[1]);

        int secret = getPIN();
        write(fd[1], &secret, sizeof(secret));
        exit(EXIT_SUCCESS);
    }

    return EXIT_SUCCESS;
}

```

Observe how the child process do some calculation and return the value to parent process using a pipe. Please provide screenshot for the output of the program above **(0.5 Marks)**

```

asfdnl@dannyphantom:~/ITT440-Lab3$ ./pipeFork2.out
Waiting for PIN...
Bytes read: 4
PIN: 1881

```

### 3.5 Modify or create your own code

Now based on what you have learn in this lab, modify above code or create a new code to create a program which a child process will ask user to enter a number, then using a pipe the child pass the number to parent process to check whether it is prime number or not. After the parent process has printed out the checking result, it can end the process. Moreover, your program must have a signal handler for SIGINT signal. Please provide screenshot of your output. **(1 Marks)**

```
asfdnl@dannyphantom:~/ITT440-Lab3$ ./owncode.out
Please enter a number: 13

Input is a prime number
asfdnl@dannyphantom:~/ITT440-Lab3$ ./owncode.out
Please enter a number: 10

Input is not a prime number
asfdnl@dannyphantom:~/ITT440-Lab3$ ./owncode.out
Please enter a number: ^CSIGNAL INTERRUPTED
: SIGNAL INTERRUPTED
```

### References

[https://linuxhint.com/pipe\\_system\\_call\\_c/](https://linuxhint.com/pipe_system_call_c/)