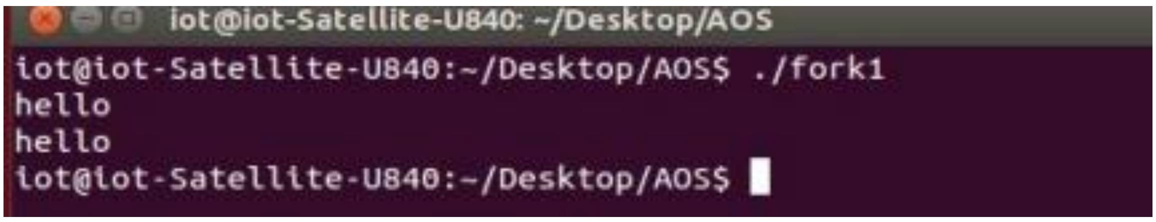
BITS Pilani - Hyderabad Campus
Advanced Operating Systems
Lab Assignment-1
1st Semester 2025-26

1. Fork System Call

The fork() call returns different values in parent(pid of child) and child(0). This is used to differentiate a child from a parent process. Normally, a different program is run in the child process (using exec() family).

```
#include<stdio.h>
#include<unistd.h>
int main()
{
        fork();
        printf("hello \n");
        return 0;
}
```

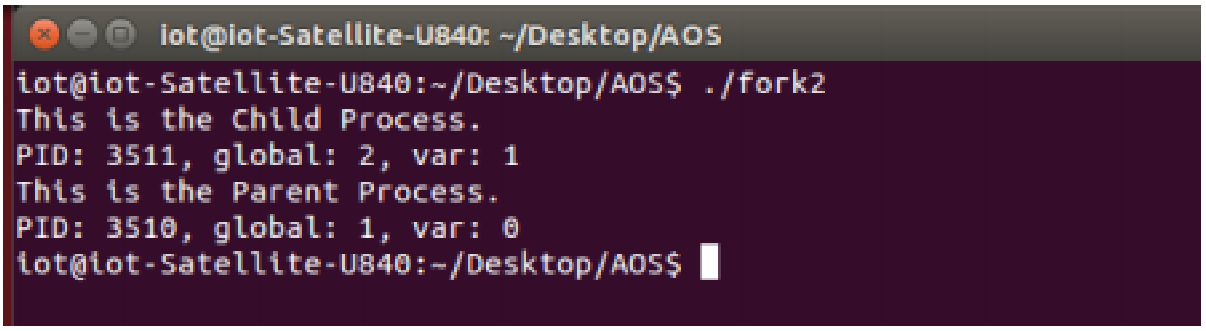


```
iot@iot-Satellite-U840: ~/Desktop/AOS
iot@iot-Satellite-U840:~/Desktop/AOS$ ./fork1
hello
hello
iot@iot-Satellite-U840:~/Desktop/AOS$
```

2. Creating a Child process using fork()

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h
int global=1;
int main()
{
        int pid,var = 1;
        if((pid=fork())==0) // Child process
        {
                global++;var++;
                printf("This is the Child Process.\n");
                printf("PID: %d, global: %d, var: %d\n",getpid(),global,var);
        }
        else // Parent process
        {
                sleep(2);
                printf("This is the Parent Process.\n");
                printf("PID: %d, global: %d, var: %d\n",getpid(),global,var);
        }
return 0;
```

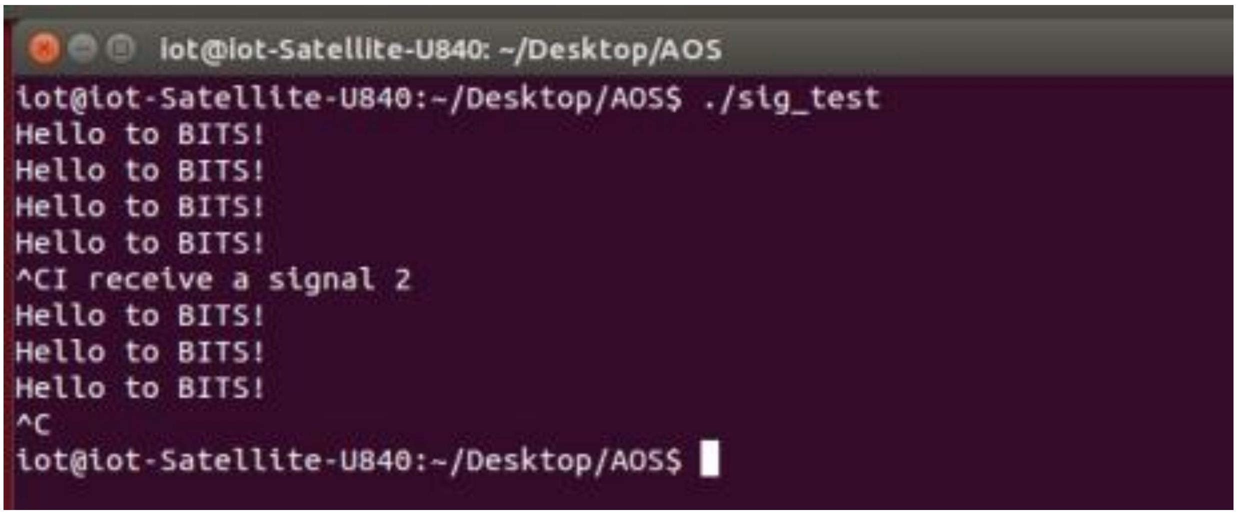
}



```
iot@iot-Satellite-U840: ~/Desktop/AOS
iot@iot-Satellite-U840:~/Desktop/AOS$ ./fork2
This is the Child Process.
PID: 3511, global: 2, var: 1
This is the Parent Process.
PID: 3510, global: 1, var: 0
iot@iot-Satellite-U840:~/Desktop/AOS$
```

3. Signals are defined in <signal.h>
   - man 7 signal for complete list of signals and their numeric values.
   - kill –l for full list of signals on a system
   - Programs can handle signals using the signal library function.
   - **void (*signal(int signo, void (*func)(int)))(int);**

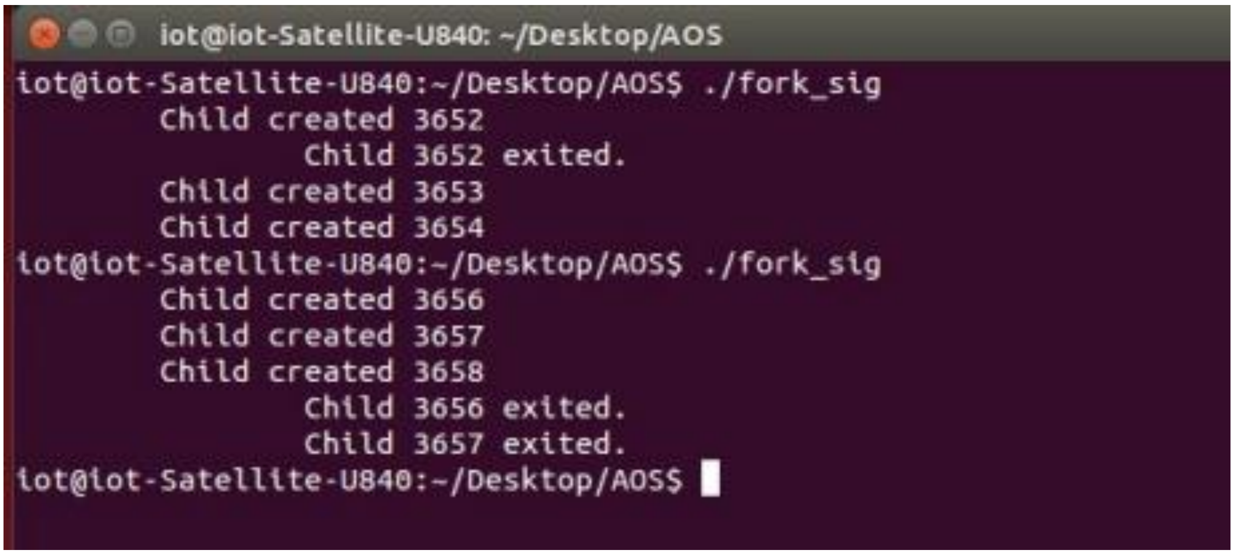
```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
void sig(int s)
{
        printf("I receive a signal %d\n", s);
        (void) signal(SIGINT, SIG_DFL);
}
int main()
{
        (void) signal(SIGINT, sig);
        while(1)
        {
                printf("Hello to BITS!\n");
                sleep(1);
        }
return 0;
}
```



```
iot@iot-Satellite-U840: ~/Desktop/AOS
iot@iot-Satellite-U840:~/Desktop/AOS$ ./sig_test
Hello to BITS!
Hello to BITS!
Hello to BITS!
Hello to BITS!
^CI receive a signal 2
Hello to BITS!
Hello to BITS!
Hello to BITS!
^C
iot@iot-Satellite-U840:~/Desktop/AOS$
```

4. Implement a C program to demonstrate the use of SIGCHLD signal. When a child process stops or terminates, SIGCHLD is sent to the parent process. The default response to the signal is to ignore it. The signal can be caught and the exit status from the child process can be obtained by immediately calling wait() . This allows zombie process entries to be removed as quickly as possible.

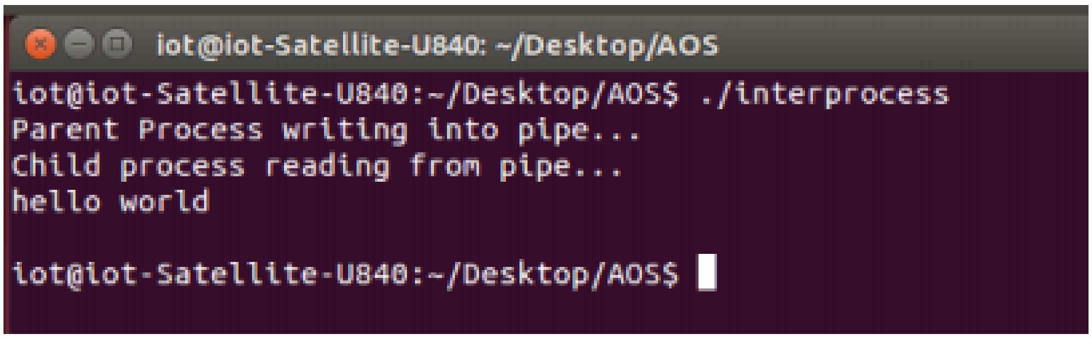
```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include<stdlib.h>

void handler(int sig)
{
 pid_t pid;
 pid = wait(NULL);
 printf("\t\tChild %d exited.\n", pid);
 signal(SIGCHLD, handler);
}
int main()
{
 int i;
 signal(SIGCHLD, handler);
 for(i=0;i<3;i++)
 switch(fork())
 {
          case 0:
 printf("\tChild created %d\n", getpid());
 exit(0);
 }
 sleep(2);
 return 0;
}
```



```
iot@iot-Satellite-U840: ~/Desktop/AOS
iot@iot-Satellite-U840:~/Desktop/AOS$ ./fork_sig
        Child created 3652
                Child 3652 exited.
        Child created 3653
        Child created 3654
iot@iot-Satellite-U840:~/Desktop/AOS$ ./fork_sig
        Child created 3656
        Child created 3657
        Child created 3658
                Child 3656 exited.
                Child 3657 exited.
iot@iot-Satellite-U840:~/Desktop/AOS$ 
```

5. Inter Process Communication using Pipes, between parent and child process

```
#include<stdio.h>
#include<stdlib.h>
#define MAXLINE 200
int main()
{
	int n,fd[2];
	pid_t pid;
	char line[MAXLINE];
	if(pipe(fd)<0)
		printf("PIPE ERROR!\n");
	if((pid=fork())<0)
		printf("FORK ERROR!\n");
	if(pid==0)
	{
		close(fd[1]);
		printf("Child process reading from pipe...\n");
		n=read(fd[0],line,MAXLINE);
		puts(line);
	}
	else
	{
		close(fd[0]);
		printf("Parent Process writing into pipe...\n");
		write(fd[1],"hello world\n",12);
	}
return 0;
}
```



```
iot@iot-Satellite-U840:~/Desktop/AOS$ ./interprocess
Parent Process writing into pipe...
Child process reading from pipe...
hello world

iot@iot-Satellite-U840:~/Desktop/AOS$ 
```