**21BCE7371**
**RADHA KRISHNA GARG**

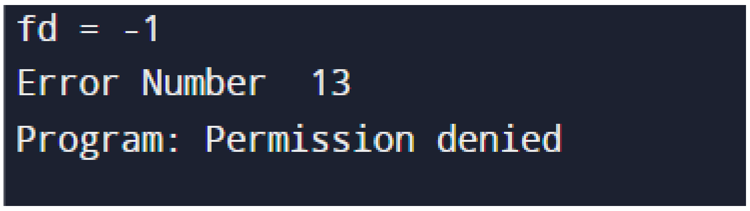
# SYSTEM CALLS

1. **OPEN()**

CODE:

```
// C program to illustrate
// open system call
#include<stdio.h>
#include<fcntl.h>
#include<errno.h>
extern int errno;
int main()
{
        // if file does not have in directory
        // then file foo.txt is created.
        int fd = open("foo.txt", O_RDONLY | O_CREAT);

        printf("fd = %d\n", fd);

        if (fd ==-1)
        {
                // print which type of error have in a code
                printf("Error Number % d\n", errno);

                // print program detail "Success or failure"
                perror("Program");
        }
        return 0;
}
```

**OUTPUT**



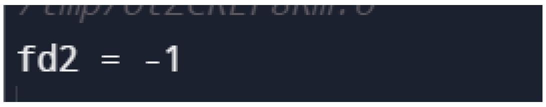
```
fd = -1
Error Number  13
Program: Permission denied
```

## 2. CLOSE()

CODE:

```
// C program to illustrate close system Call
#include<stdio.h>
#include<fcntl.h>
int main()
{
        // assume that foo.txt is already created
        int fd1 = open("foo.txt", O_RDONLY, 0);
        close(fd1);

        // assume that baz.tzt is already created
        int fd2 = open("baz.txt", O_RDONLY, 0);

        printf("fd2 = % d\n", fd2);
        exit(0);
}
```

### OUTPUT



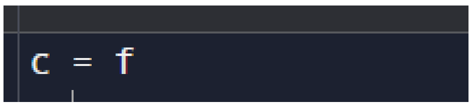
```
fd2 = -1
```

## 3. READ()

CODE:

```
// C program to illustrate
// read system Call
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<stdlib.h>

int main()
{
        char c;
        int fd1 = open("sample.txt", O_RDONLY, 0);
        int fd2 = open("sample.txt", O_RDONLY, 0);
        read(fd1, &c, 1);
        read(fd2, &c, 1);
        printf("c = %c\n", c);
        exit(0);
}
```

**OUTPUT**



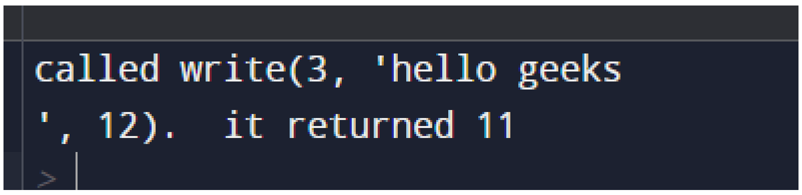
```
c = f
```

## 4. write()

CODE:

```
// C program to illustrate
// write system Call
#include<stdio.h>
#include <fcntl.h>
main()
{
int sz;

int fd = open("foo.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
if (fd < 0)
{
        perror("r1");
        exit(1);
}

sz = write(fd, "hello geeks\n", strlen("hello geeks\n"));

printf("called write(% d, \"hello geeks\\n\", %d)."
        " It returned %d\n", fd, strlen("hello geeks\n"), sz);

close(fd);
}
```

**OUTPUT**



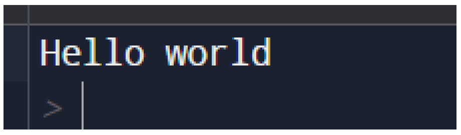
```
called write(3, 'hello geeks
', 12).  it returned 11
```

## 5. I/O SYSTEM CALL

CODE:

```
// C program to illustrate
// I/O system Calls
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>

int main (void)
{
        int fd[2];
        char buf1[12] = "hello world";
        char buf2[12];

        // assume foobar.txt is already created
        fd[0] = open("foobar.txt", O_RDWR);
        fd[1] = open("foobar.txt", O_RDWR);

        write(fd[0], buf1, strlen(buf1));
        write(1, buf2, read(fd[1], buf2, 12));

        close(fd[0]);
        close(fd[1]);

        return 0;
}
```

## OUTPUT

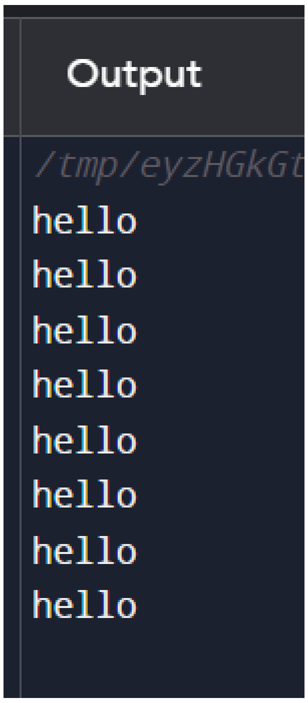


```
Hello world
>
```

## 6. fork()

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

## OUTPUT



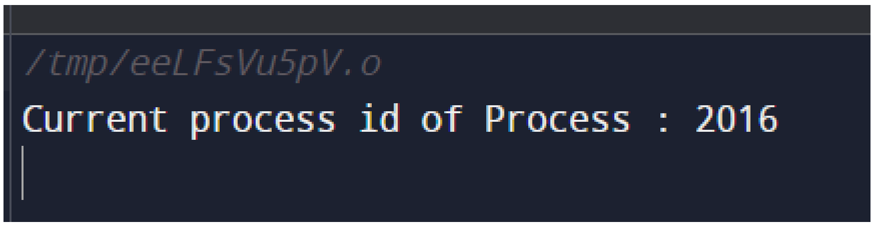
Output

```
/tmp/eyzHGkGt
hello
hello
hello
hello
hello
hello
hello
hello
```

## 7.getpid()

```
// C++ Code to demonstrate getpid()
#include <iostream>
#include <unistd.h>
using namespace std;

// Driver Code
int main()
{
        int pid = fork();
        if (pid == 0)
                cout << "\nCurrent process id of Process : "
                        << getpid() << endl;
        return 0;
}
```

## OUTPUT

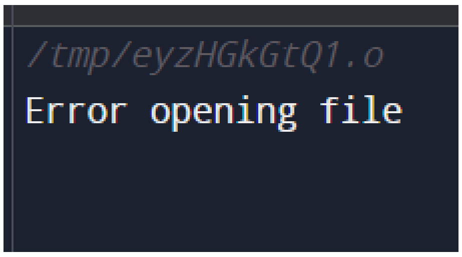


```
/tmp/eeLFsVu5pV.o
Current process id of Process : 2016
```

## 8. exit()

```
/* exit example */
#include <stdio.h>
#include <stdlib.h>

int main ()
{
FILE * pFile;
pFile = fopen ("myfile.txt", "r");
if (pFile == NULL)
{
        printf ("Error opening file");
        exit (1);
}
else
{
        /* file operations here */
}
return 0;
}
```

## OUTPUT



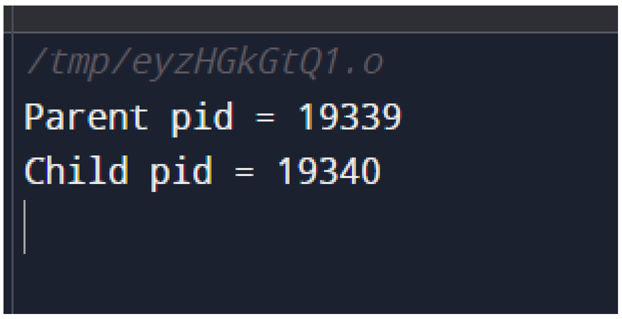
```
/tmp/eyzHGkGtQ1.o
Error opening file
```

## 9. Wait()

```
// C program to demonstrate working of wait()
#include<stdio.h>
#include<stdlib.h>
#include<sys/wait.h>
```

```
#include<unistd.h>

int main()
{
        pid_t cpid;
        if (fork()== 0)
                exit(0);            /* terminate child */
        else
                cpid = wait(NULL); /* reaping parent */
        printf("Parent pid = %d\n", getpid());
        printf("Child pid = %d\n", cpid);

        return 0;
}
```

## OUTPUT



```
/tmp/eyzHGkGtQ1.o
Parent pid = 19339
Child pid = 19340
```