System Software (CS306)

Assignment - 1

U19CS012

Aim: To study the **Basics of System Call** and **System Library**.

System Calls:

A.) **fork()** [Process Control System Call]

- ✓ **Purpose**: Used to **create a new separate & duplicate process** [Child Process] which runs concurrently with the parent process.
- ✓ On success, the **PID of the child process** is returned in the parent, and **0** is returned in the child.
- ✓ On failure, **-1** is returned to the parent, no child process is created, and err_no is set appropriately.

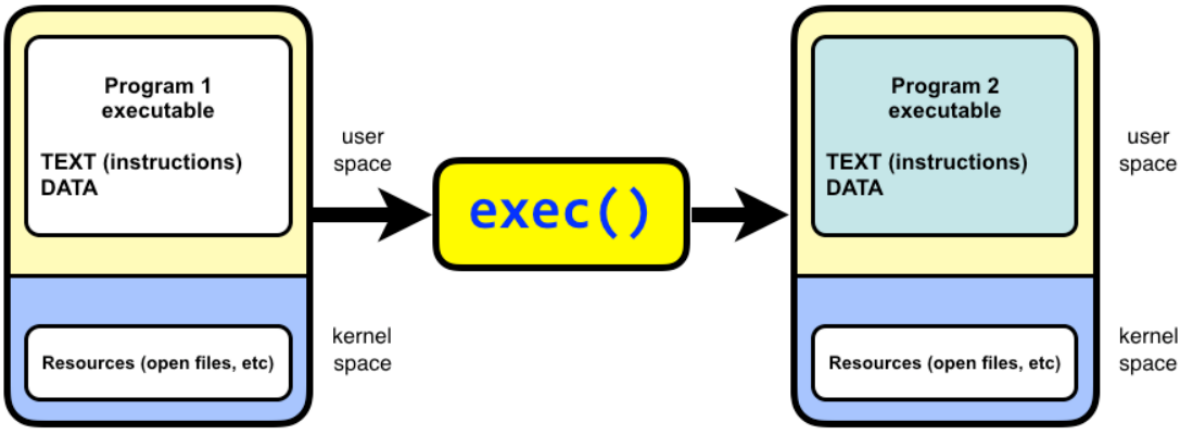
Example:

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

int main()
{
        fork();

        printf("U19CS012 in S.S. Lab!\n");
        printf("PID of Current Process = %d\n", getpid());
        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc ./a.c && ./a.out
U19CS012 in S.S. Lab!
PID of Current Process = 7729
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ U19CS012 in S.S. Lab!
PID of Current Process = 7730
```

B.) **exec()** [Process Control System Call]

Purpose: To execute a file which is residing in an active process (user wants to launch a new file or program in the same process.)



Example:

ex1.c

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

int main(int argc, char*argv[])
{
        printf("PID of ex1.c = %d\n", getpid());
        char *args[] = {"U19CS012","Bhagya","Rana",NULL};
        execv("./ex2", args);
        printf("Back to ex1.c\n");
        return 0;
}
```

ex1.c | ex2.c

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

int main(int argc, char*argv[])
{
        printf("We are in ex2.c\n");
        printf("PID of ex2.c = %d\n", getpid());
        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc ex1.c -o ex1
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc ex2.c -o ex2
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./ex1
PID of ex1.c = 7914
We are in ex2.c
PID of ex2.c = 7914
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$
```

exec() system call

C.) **getpid()** [Process Control System Call]

Purpose: to get Process ID of Current Process

Example:

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

int main()
{
        printf("U19CS012 in S.S. Lab!\n");
        printf("PID of Current Process = %d\n", getpid());
        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc ./a.c && ./a.out
U19CS012 in S.S. Lab!
PID of Current Process = 7431
```

D.) **exit()** [Process Control System Call]

Purpose:

- ✓ A process terminates when it **finishes executing its final statement** and asks the operating system to **delete it** using the exit() system call.
- ✓ At that point, the process may return a **status value** (typically an integer) to its parent process (via the **wait() system call**).
- ✓ Only **Parent process** can kill its Child Process.

E.) wait()

Purpose: Helps the Parent Process to wait, until all its child processes are complete.

Example:

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

int main()
{
        // printf("U19CS012 in S.S. Lab!\n");
        if(fork()==0)
        {
                // Inside Child Process
                printf("Hi from Child Process!\n");
                exit(0);
        }else if(fork()>0){
                // Inside Parent Process
                printf("Hi from Parent Process!\n");
                // This will allow all child process to execute first
                wait(NULL);
                // Control Reaches here, when all Child Process are Completed
                printf("Child Terminated!\n");
        }

        printf("Parent Terminated\n");

        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc ./a.c -o a.out
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./a.out
Hi from Parent Process!
Hi from Child Process!
Child Terminated!
Parent Terminated
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ Parent Terminated
```

F.) stat()

Purpose: To check the **status of a file** such as to check when the file was accessed

Example:

```c
#include<stdio.h>
#include<sys/stat.h>

int main()
{
        // printf("U19CS012 in S.S. Lab!\n");

        // Pointer to stat struct
        struct stat sfile;

        // stat() system call
        stat("stat.c", &sfile);

        // accessing data member of stat structure
        printf("st_mode = %o \n", sfile.st_mode);

        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc stat.c -o stat
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./stat
st_mode = 100664
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$
```

G.) opendir()

Purpose:

- ✓ The opendir() function shall open a directory stream corresponding to the directory named by the dirname argument.
- ✓ The opendir() function **opens a directory** and returns a pointer to the directory stream
- ✓ The stream is positioned at the **first entry in the directory**.

H. readdir()

Purpose:

- ✓ readdir - read a directory
- ✓ The readdir() function gives **next directory entry** in the directory stream
- ✓ It returns NULL on reaching the end of the directory stream

Example:

```c
#include<stdio.h>
#include<unistd.h>
#include<dirent.h>

int main()
{
        // Pointer to Directory
        DIR *d;

        // For Reading the Contents
        struct dirent *de;

        // "." -> Current Directory
        d = opendir(".");

        while(de=readdir(d))
        {
                printf("%s\n", de->d_name);
        }

        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ touch dir.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gedit dir.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc dir.c -o dir
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./dir
stat
dir
ex2.c
stat.c
dir.c
ex2
ex1.c
..
.
p1.out
a.c
ex1
a.out
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$
```

opens the current directory using opendir() & prints its content using readdir()

I.) chdir()

Purpose: changes the current working directory to that specified in path

Example:

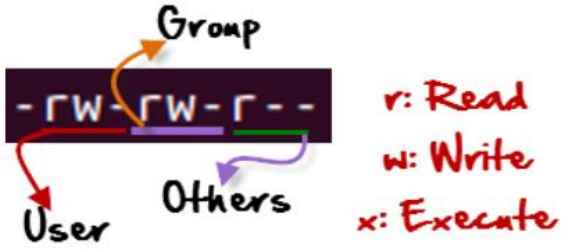
```c
#include<stdio.h>
#include<unistd.h> // Contains chdir()
int main()
{
        char s[100];

        // Print Current Working Directory
        printf("%s\n", getcwd(s,100));

        chdir("..");

        // Print Current Working Directory {After Changing it}
        printf("%s\n", getcwd(s,100));

        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ touch chdir.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gedit chdir.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc chdir.c -o chdir
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./chdir
/home/bhagya/Desktop/ss_lab1
/home/bhagya/Desktop
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$
```

Directory Changed using chdir()

J.) chmod()

Purpose: change permissions of a file



Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

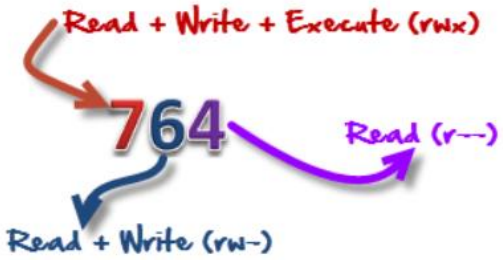
Example:

A.) Checking Current File Permissions

```
bhagya@bhagya-VirtualBox:~/Desktop/Departments/MECHANICAL/HMT$ ls -l p2.c
-rw-rw-r-- 1 bhagya bhagya 0 Jul 27 16:02 p2.c
```

B.) chmod 764 and Checking Permissions Again

```
bhagya@bhagya-VirtualBox:~/Desktop/Departments/MECHANICAL/HMT$ ls -l p2.c
-rw-rw-r-- 1 bhagya bhagya 0 Jul 27 16:02 p2.c
bhagya@bhagya-VirtualBox:~/Desktop/Departments/MECHANICAL/HMT$ chmod 764 p2.c
bhagya@bhagya-VirtualBox:~/Desktop/Departments/MECHANICAL/HMT$ ls -l p2.c
-rwxrw-r-- 1 bhagya bhagya 0 Jul 27 16:02 p2.c
```



K.) kill()

Purpose:

The kill() system call can be used to send any signal to any process group or process.

Example:

```c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<signal.h>
// U19CS012
int main()
{
        printf("Process PID : %d\n",getpid());
        sleep(5);
        kill(getpid(),SIGSEGV);
        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ touch kill.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gedit kill.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc kill.c -o kill
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./kill
Process PID : 8816
Segmentation fault (core dumped)
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$
```

#include <fcntl.h> for read(), write(), open() & close().

L.) read()

Purpose:
int read(int handle, void *buffer, int nbyte);

- ✓ The read() function attempts to read nbytes from the file associated with handle, and places the characters read into buffer.
- ✓ The function returns the number of bytes read.
- ✓ On end-of-file, 0 is returned, on error it returns -1, setting errno to indicate the type of error that occurred.

M.) write()

Purpose:

int write(int handle, void *buffer, int nbyte);

- ✓ The write() function attempts to **write nbytes from buffer to the file** associated with handle.
- ✓ The function returns the **number of bytes written to the file**.
- ✓ A return value of -1 indicates an error, with errno set appropriately.

N.) open()

Purpose:
open(char *filename, int access, int permission);

- ✓ The open() function returns an integer value, which is used to refer to the file.
- ✓ If unsuccessful, it returns -1, and sets the global variable errno to indicate the error type.

O.) close()

Purpose:
int close(int handle);

- ✓ The close() function closes the file associated with handle.
- ✓ The function returns 0 if successful, -1 to indicate an error, with errno set appropriately.

P.) lseek()

Purpose: used to change the location of the read/write pointer of a file descriptor.

Example for open(), read(), write(), lseek(), close() :

```c
#include<stdio.h>
#include<fcntl.h>
int main()
{
        int fd;
        char buffer[80];
        static char mess[] = "HI, U19CS012 in SS Lab!";

        fd = open("lorem.txt", O_RDWR);

        if(fd!=-1)
        {
                printf("lorem.txt Opened with Read & Write Access");
                write(fd, mess, sizeof(mess));

                lseek(fd,0,0);

                read(fd, buffer, sizeof(mess));

                printf("%s - written to lorem.txt\n", buffer);

                close(fd);
        }

        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gedit filefx.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc filefx.c -o filefx
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./filefx
lorem.txt Opened with Read & Write Access
HI, U19CS012 in SS Lab! - written to lorem.txt
```

Q.) time()

Purpose: the time as the number of seconds since the Epoch

- ✓ The time() function is defined in time.h (ctime in C++) header file.
- ✓ This function returns the time since 00:00:00 UTC, January 1, 1970 (Unix timestamp) in seconds.
- ✓ If second is not a null pointer, the returned value is also stored in the object pointed to by second.

Example:

```c
#include<stdio.h>
#include<time.h>
// U19C012
int main()
{
        time_t seconds;
        time(&seconds);
        printf("Seconds since January 1, 1970 = %ld\n", seconds);
        return 0;
}
```

```
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ touch time.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gedit time.c
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ gcc time.c -o time
bhagya@bhagya-VirtualBox:~/Desktop/ss_lab1$ ./time
Seconds since January 1, 1970 = 1642648278
```

R.) mount()

Purpose: mount a filesystem

Example:

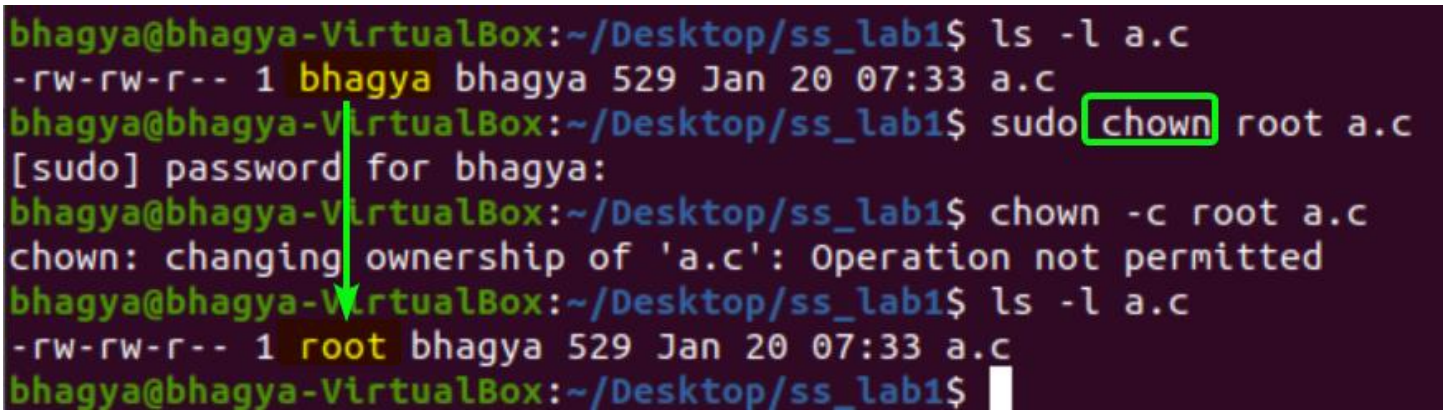
```
cindy@cindy-nyc:~$ sudo parted -l
Model: ATA SAMSUNG MZNLN128 (scsi)
Disk /dev/sda: 128GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system     Name                  Flags
 1      1049kB  538MB   537MB   fat32           EFI System Partition  boot, esp
 2      538MB   120GB   119GB   ext4
 3      120GB   128GB   8463MB  linux-swap(v1)


Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 7803MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  5369MB  5368MB  ext4         primary

cindy@cindy-nyc:~$ sudo mount /dev/sdb1 /my_usb/
cindy@cindy-nyc:~$
```

S.) chown()

Purpose:

chown command is used to change the file Owner or group. Whenever you want to change ownership you can use chown command.

Example:



SUBMITTED BY: U19CS012

BHAGYA VINOD RANA