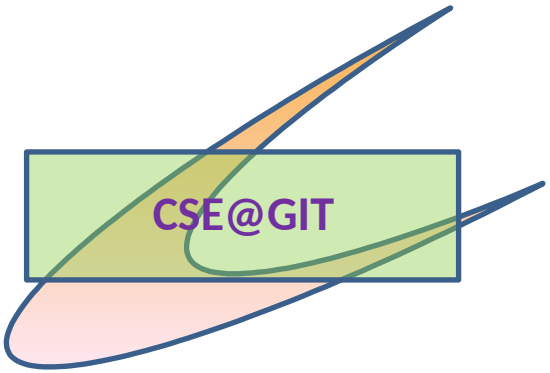
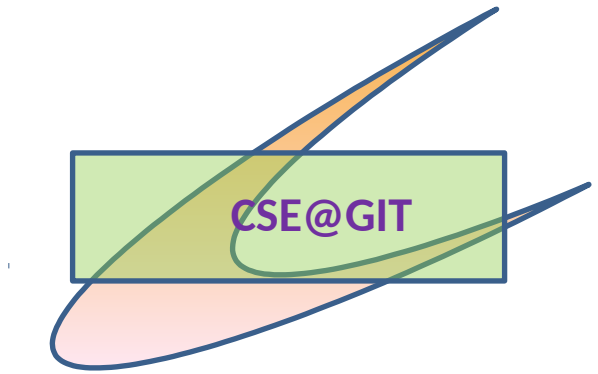
Experiment No. 10

Problem Definition: Consider system( ) as a higherlevel interface and duplicate its functionality using the mechanism of process creation.



Objectives of the Experiment:

1. To demonstrate the use of macros of POSIX.1 standard
2. To develop an understanding of working of system API
3. To differentiate between user and system functions.



Need of the Experiment

- To invoke a OS command from a C++ program.
- To understand the exit status of the executed command.
- To understand how system() handles all of the details of calling fork(2), execl(3), and waitpid(2).

Theoretical Background of the Experiment

- Process creation concept.(fork, exec, wait)
- Various exec functions.

Points to remember

There are six exec functions:

1. #include <unistd.h>
2. int **execl**(const char *pathname, const char *arg0,... /* (char *)0 */);
3. int **execv**(const char *pathname, char *const argv [ ]);
4. int **execle**(const char *pathname, const char *arg0,... /*(char *)0, char *const envp */);
5. int **execve**(const char *pathname, char *const argv[ ], char *const envp[]);
6. int **execlp**(const char *filename, const char *arg0, ... /* (char *)0 */); int **execvp**(const char *filename, char *const argv [ ]);

Concepts to remember

system(): executes a command specified in command by calling /bin/sh -c command, and returns after the command has been completed.

exec() family of functions replaces the current process image with a new process image.

execl() function is one among the exec() family of functions.

waitpid() system call suspends execution of the calling process until a child specified by pid argument has changed state

Syntax

int system(const char *cmdstring)

cmdstring: is the parameter to be executed by system command

Like: date, who, ls

Algorithm for the experiment

1. Declare the required header files limits.h, unistd.h, iostream.h
2. Call system function with parameter in main()
3. Create a process with fork() in the function
4. Call exec function to overwrite the existing process
5. If process busy call wait function for process to wait
6. Display the result.

Pseudo Code / Outline of the Algorithm

Working with main function:

```
int main()

  {
    int status;
    // Read the no of commands – n
    for i=1 to n
     {
       read cmd
       status = system(“cmd")
     }
  }
```

Pseudo Code / Outline of the Algorithm

Working with system function:

```
int system(const char *cmdstring)
  (
    if ((pid = fork()) < 0)
     {
       status = -1;
     }
   else if (pid == 0)
     {
       /* child */
        execl("/bin/sh", "sh", "-c", cmdstring, (char *)0); _exit(127);
     }
```

Pseudo Code / Outline of the Algorithm

Working with system function:

```
else
/* parent */
 while (waitpid(pid, &status, 0) < 0)
  {
     if (errno != EINTR)
       status = -1;
        /* error other than EINTR from waitpid() */ break;
  }
 return(status);
```

Sample Run

Run the Program:

```
[root@localhost /]# cc system.c
root@localhost /]# ./a.out
```

Sample Output:

```
Wed Jan 17 18:03:31 IST 2017
root   pts/1
2017-01-17 17:35 (:0.0)
```

Learning Outcomes of the Experiment

At the end of the session, students should be able to :

1) Identify the use of system function. [L1]

2) Make use of system function to execute commands.[L3]