## Ex. No: 6 Date: 06-10-2023

# PROCESS MANAGEMENT

## Aim:

To implement the process management by using fork(), exec() and wait() system calls.

## Procedure:

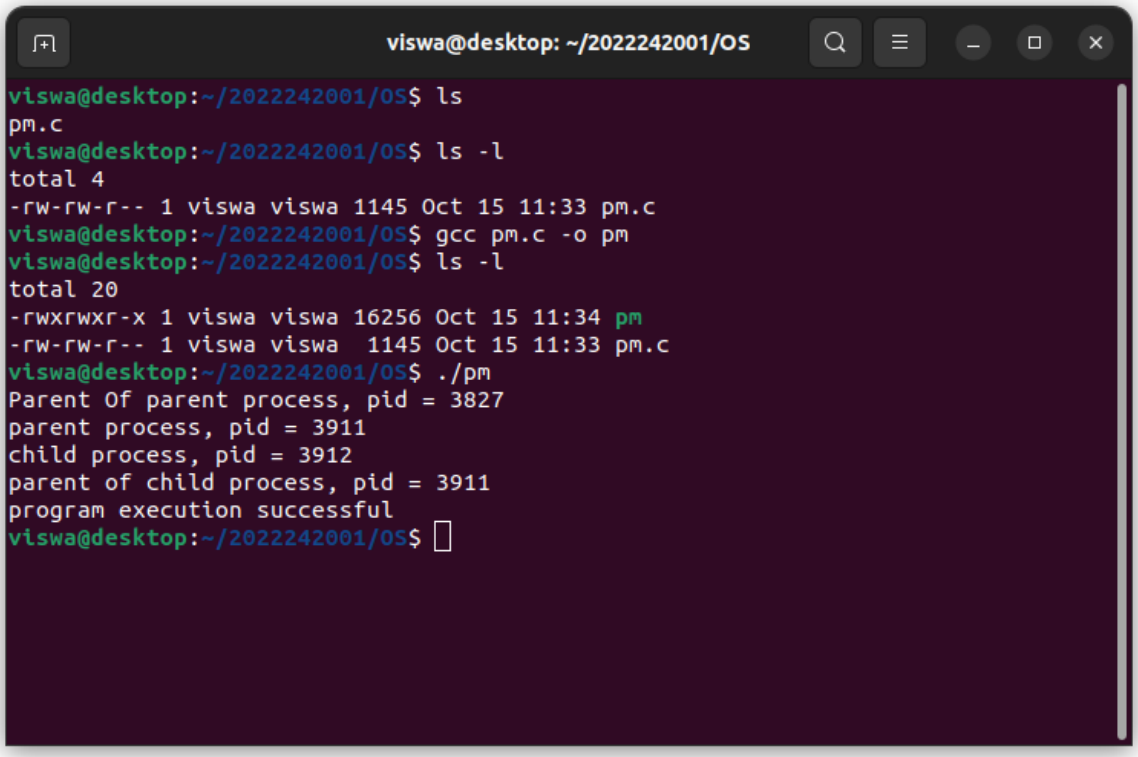
**Step 1** - Create a new process from a parent process using the fork() function.
**Step 2** - If the process is created successfully, start executing it, otherwise display an error message.
**Step 3** - Start executing the process using the execv() function.
**Step 4** - The parent process waits till the child process is executed.
**Step 5** - If the child process doesn't execute successfully, display an error message.
**Step 6** - If the process is executed successfully end the program.

## Program :

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/wait.h>
int main(){
    pid_t pid;
    int ret = 1;
    int status;
    pid = fork();
    if (pid == -1){
        printf("Cannot fork, Error occured\n");
        exit(EXIT_FAILURE);
    }
    else if (pid == 0){
        printf("child process, pid = %u\n", getpid());
        printf("parent of child process, pid = %u\n", getppid());
        char *argv_list[] = {"ls", "-lart", "/home", NULL};
        execv("ls", argv_list);
        exit(0);
    }
    else{
        printf("Parent Of parent process, pid = %u\n",
getppid());
        printf("parent process, pid = %u\n", getpid());
        if (waitpid(pid, &status, 0) > 0){
            if (WIFEXITED(status) && !WEXITSTATUS(status))
                printf("program execution successful\n");
            else if (WIFEXITED(status) && WEXITSTATUS(status)){
                if (WEXITSTATUS(status) == 127){
                    printf("Execution failed\n");
                }
                else
```

```
                    printf("Program terminated normally, but
    returned a non-zero status\n");
                }
                else
                    printf("Program didn't terminate normally\n");
            }
            else{
                printf("waitpid() failed\n");
            }
            exit(0);
        }
        return 0;
    }
```

**Output:**




**RESULT :**

Thus, the process management were implemented by using fork(), exec() and wait() system calls.