**Assignment No. 02**

**Problem Statement :** Write a program to simulate use of Linux commands like cp, grep with the usage of fork () and exec () system calls. Also show the usage of wait (), getpid () and exit () system calls.

**GREP Command Program**

**Code :**

#include<stdio.h>

#include<string.h>

int main(int argc, char\* argv[]) {

    if(argc != 3) {

        printf("You can not enter more or less than 3 arguments.");

        return 0;

        }

        char\* fn;

        char\* pat;

        char line[5000];

        FILE\* fp;

        char\* match;

        pat = argv[1];

        fn = argv[2];

        fp = open(fn, O\_RDONLY);

        while(!feof(fp)) {

            fgets(line, 5000, fp);

            match = strstr(line, pat);

            if(match) {

                \*match = '\0';

                printf("%s", line);

                printf("\033[31m%s\033[0m", pat);

                printf("%s", match + strlen(pat));

            }

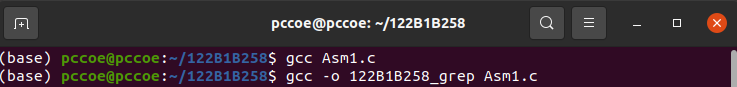
        }

    close(fp);

    return 0;

}

**Compilation :**

****

**CP Command Program**

**Code :**

#include<stdio.h>

#include<string.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<unistd.h>

int main(int argc, char\* argv[]) {

    if(argc != 3) {

        printf("You can not enter more or less than 3 arguments.");

        return 0;

    }

    char\* fn1;

    char\* fn2;

    char line[50];

    int fp1;

    int fp2;

    int n;

    fn1 = argv[1];

    fn2 = argv[2];

    fp1 = open(fn1, O\_RDONLY);

    fp2 = open(fn2, O\_WRONLY);

    while ((n = read(fp1, line, 50)) > 0) {

        write(fp2, line, n);

    }

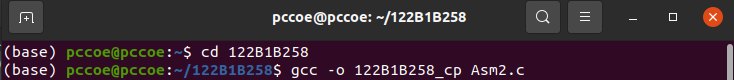
    close(fp2);

    close(fp1);

    return 0;

}

**Compilation :**

****

**Assignment Program**

**Code :**

#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

#include<stdlib.h>

int main() {

    int pid;

    int ppid;

    int fork\_var;

    int ch;

    pid = getpid();

    ppid = getppid();

    printf("You are in a Parent Process\n");

    printf("The PID of this process is %d\n", pid);

    printf("The PID of parent process is %d\n", ppid);

    do {

        printf("\nYou are welcome!!!\nChoose any one option:\n1. GREP Command\n2. CP Command\n3. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &ch);

        switch(ch) {

        case 1: {

            fork\_var = fork();

            if (fork\_var == 0) {

                printf("You are in a child process.\n");

                pid = getpid();

                ppid = getppid();

                printf("The PID of child process is %d\n", pid);

                printf("The PID of parent process is %d\n", ppid);

                char pat[10];

                char fl[10];

                printf("Enter a file name: ");

                scanf("%s", fl);

                printf("Enter a pattern: ");

                scanf("%s", pat);

                printf("Running GREP command:\n");

                execlp("./122B1B258\_grep", "122B1B258\_grep", pat, fl, NULL);

                exit(1);

            } else if (fork\_var > 0) {

                wait(NULL);

            } else {

                perror("Fork failed");

            }

            break;

        }

        case 2: {

            fork\_var = fork();

            if (fork\_var == 0) {

                printf("You are in a child process.\n");

                pid = getpid();

                ppid = getppid();

                printf("The PID of child process is %d\n", pid);

                printf("The PID of parent process is %d\n", ppid);

                char fl1[10];

                char fl2[10];

                printf("Enter a source file name: ");

                scanf("%s", fl1);

                printf("Enter a destination file name: ");

                scanf("%s", fl2);

                printf("Running CP command:\n");

                execlp("./122B1B258\_cp", "122B1B258\_cp", fl1, fl2, NULL);

                exit(1);

            } else if (fork\_var > 0) {

                wait(NULL);

            } else {

                perror("Fork failed");

            }

            break;

        }

        case 3: {

            printf("Thank you!!!\n");

            exit(1);

        }

        default: {

            printf("You entered wrong option!!! Please try again.\n");

            break;

        }

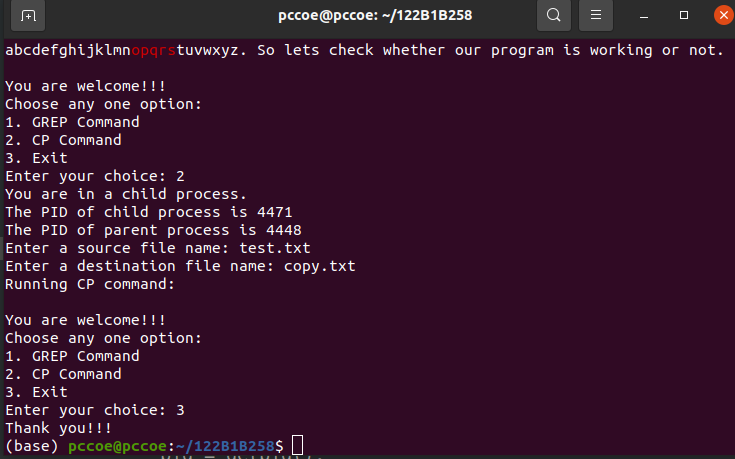
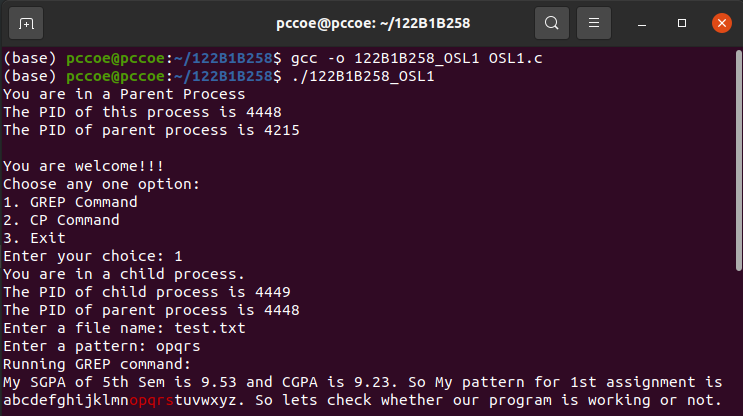
        }

    } while(ch != 3);

    return 0;

}

**Output :**

****