**2022-CS-36**

**Task1:**#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

    pid\_t pid1, pid2, pid3;

    // Create first child

    pid1 = fork();

    // Check for error

    if (pid1 < 0) {

        fprintf(stderr, "Fork failed\n");

        return 1;

    } else if (pid1 == 0) {

        // Child process 1

        printf("Child 1 - My PID is: %d\n", getpid());

        exit(0);

    } else {

        // Parent process

        // Create second child

        pid2 = fork();

        // Check for error

        if (pid2 < 0) {

            fprintf(stderr, "Fork failed\n");

            return 1;

        } else if (pid2 == 0) {

            // Child process 2

            printf("Child 2 - My PID is: %d\n", getpid());

            exit(0);

        } else {

            // Parent process

            // Create third child

            pid3 = fork();

            // Check for error

            if (pid3 < 0) {

                fprintf(stderr, "Fork failed\n");

                return 1;

            } else if (pid3 == 0) {

                // Child process 3

                printf("Child 3 - My PID is: %d\n", getpid());

                exit(0);

            } else {

                // Parent process

                printf("Parent - My PID is: %d\n", getpid());

                // Wait for all child processes to finish

                wait(NULL);

                wait(NULL);

                wait(NULL);

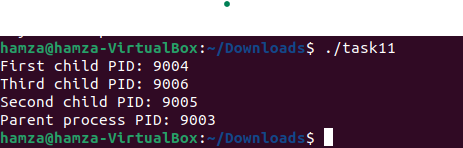
            }

        }

    }

    return 0;

}



**Task2:**

#include <stdio.h>

int main()

{

    printf("Hello from hello.c!\n");

    return 0;

}

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

    printf("Executing example.c\n");

    // Call exec to replace current process with hello

    if (execl("./hello", "hello", NULL) == -1)

    {

        perror("exec");

        exit(EXIT\_FAILURE);

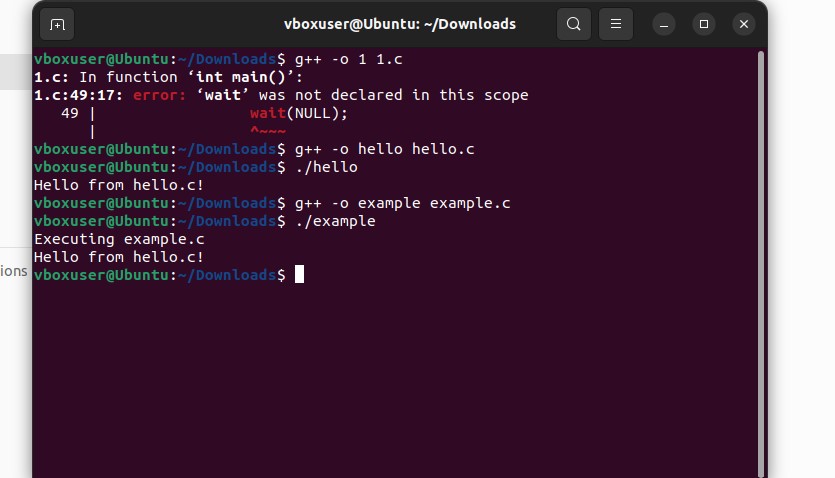
    }

    // This part will not be executed if exec succeeds

    printf("This line will not be executed.\n");

    return 0;

}

****

**Task3:**

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int a = 10, b = 20;

    printf("Addition: %d + %d = %d\n", a, b, a + b);

    return 0;

}

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

    printf("Executing multiplication.c\n");

    // Call exec to replace current process with addition

    if (execl("./addition", "addition", NULL) == -1)

    {

        perror("exec");

        exit(EXIT\_FAILURE);

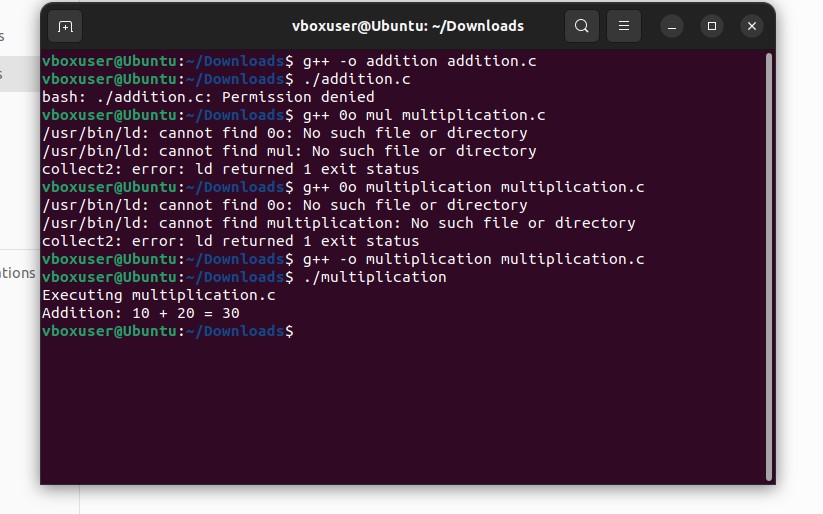
    }

    // This part will not be executed if exec succeeds

    printf("This line will not be executed.\n");

    return 0;

}

****