**2022-CS-36**

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

#include <stdlib.h>

#include <unistd.h>

int main()

{

    int n = 5;

    int i;

    printf("Parent process id: %d\n", getpid());

    for (i = 0; i < n; i++)

    {

        pid\_t pid = fork();

        if (pid < 0)

        {

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

            return 1;

        }

        else if (pid == 0)

        {

            // Child process

            printf("Child process id: %d, parent id: %d\n", getpid(), getppid());

            exit(0);

        }

    }

    // Parent process waits for all child processes to complete

    for (i = 0; i < n; i++)

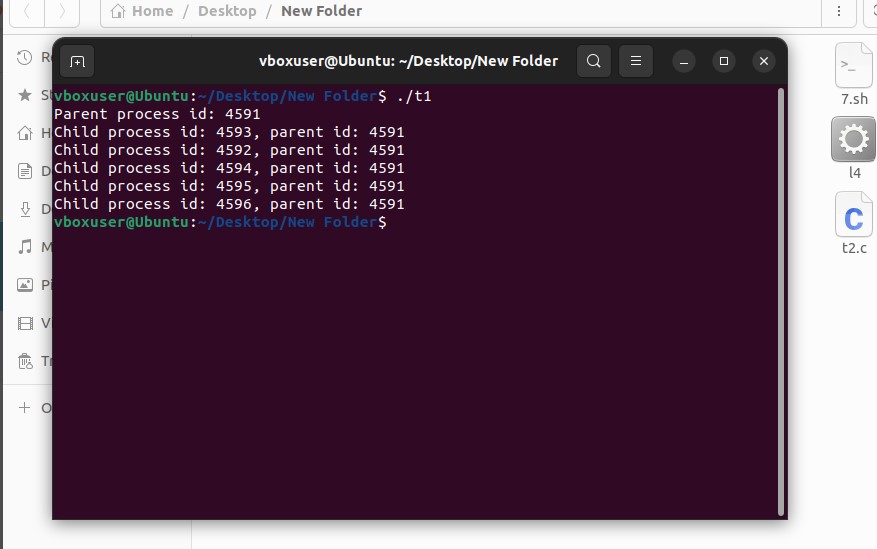
    {

        wait(NULL);

    }

    return 0;

}

****

**Task2:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int calculateSum(int n)

{

    int sum = 0;

    for (int i = 1; i <= n; i++)

    {

        sum += i;

    }

    return sum;

}

long long calculateProduct(int n)

{

    long long product = 1;

    for (int i = 1; i <= n; i++)

    {

        product \*= i;

    }

    return product;

}

int main()

{

    int n;

    printf("Enter a number: ");

    scanf("%d", &n);

    pid\_t pid = fork();

    if (pid < 0)

    {

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

        return 1;

    }

    else if (pid == 0)

    {

        long long product = calculateProduct(n);

        printf("Child process calculated product: %lld\n", product);

    }

    else

    {

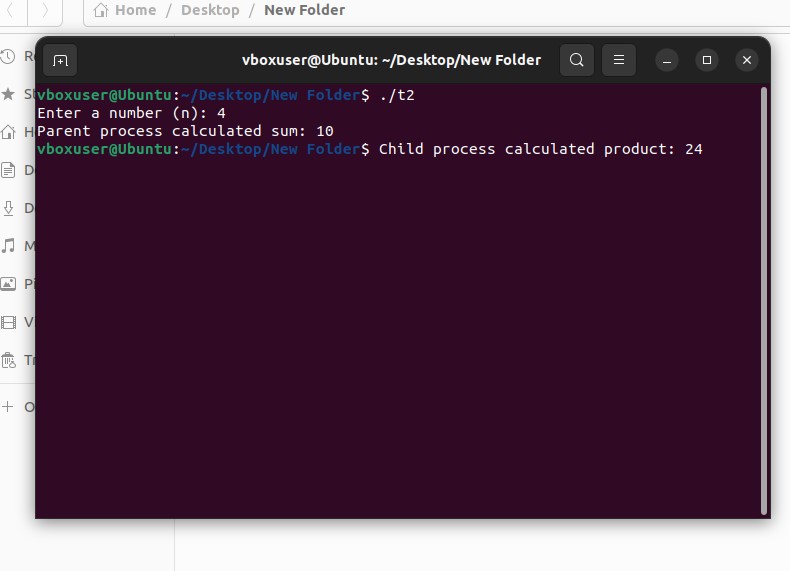
        int sum = calculateSum(n);

        printf("Parent process calculated sum: %d\n", sum);

    }

    return 0;

}

****

**Task3:**#include <stdio.h>

#include <unistd.h>

int main()

{

    int counter = 0;

    // Fork into two processes

    int pid = fork();

    if (pid == 0)

    {

        // Child process

        printf("Child process. PID: %d\n", getpid());

        for (int i = 0; i < 1000000; ++i)

        {

            counter++;

        }

        printf("Child counter value: %d\n", counter);

    }

    else if (pid > 0)

    {

        // Parent process

        printf("Parent process. PID: %d, Child PID: %d\n", getpid(), pid);

        for (int i = 0; i < 1000000; ++i)

        {

            counter++;

        }

        printf("Parent counter value: %d\n", counter);

    }

    else

    {

        // Error occurred

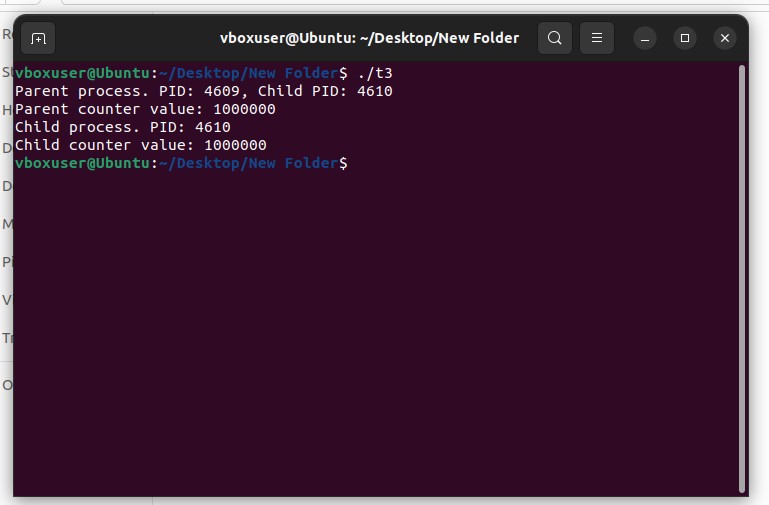
        perror("fork");

        return 1;

    }

    return 0;

}

****