Name: Sneha Tiwari

Roll: 11500119052 | Group: B1

Assignment 10

**1. Write a program to show the use of exec() system call.**

**Solution:**

**C Program**

**// exec.c**

#include<stdio.h>

#include<unistd.h>

int main()

{

int i;

printf("exec.c has been called by execvp() command.");

printf("\n");

return 0;

}

**//exec\_demo.c**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

        char \*args[]={"./exec",NULL};

        execvp(args[0],args);

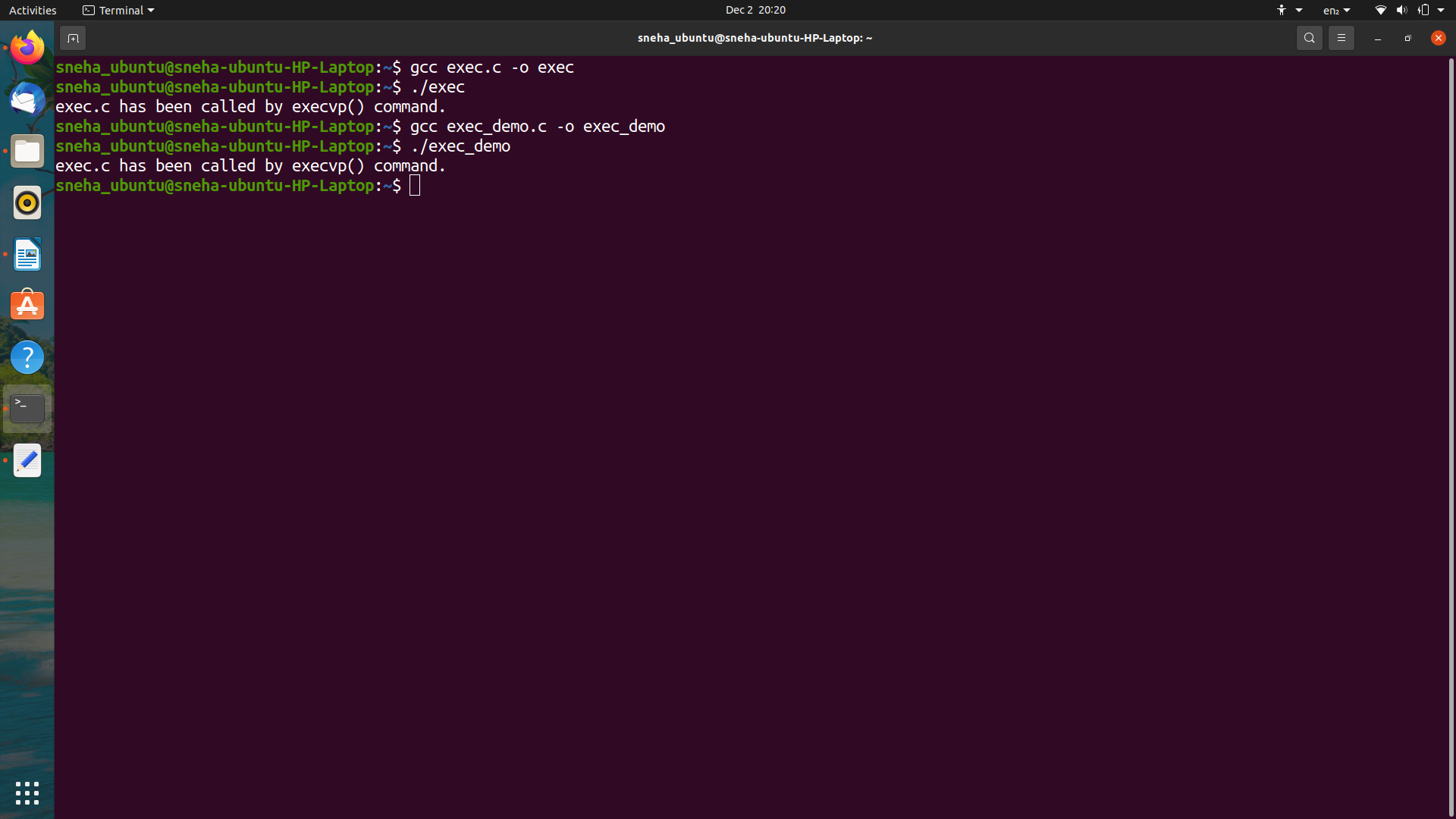
        // All statements are ignored after execvp() call

        printf("This line will not be printed.");

    return 0;

}

**Output**



**2. Write a program to show how Zombie process can cause problem.**

**Solution:**

**C Program**

#include<stdio.h>

#include<unistd.h>

#include<sys/wait.h>

#include<sys/types.h>

int main()

{

int i;

int pid = fork();

if (pid==0)

{

for (i=0; i<10; i++)

printf("I am Child\n");

}

else

{

//wait(NULL);

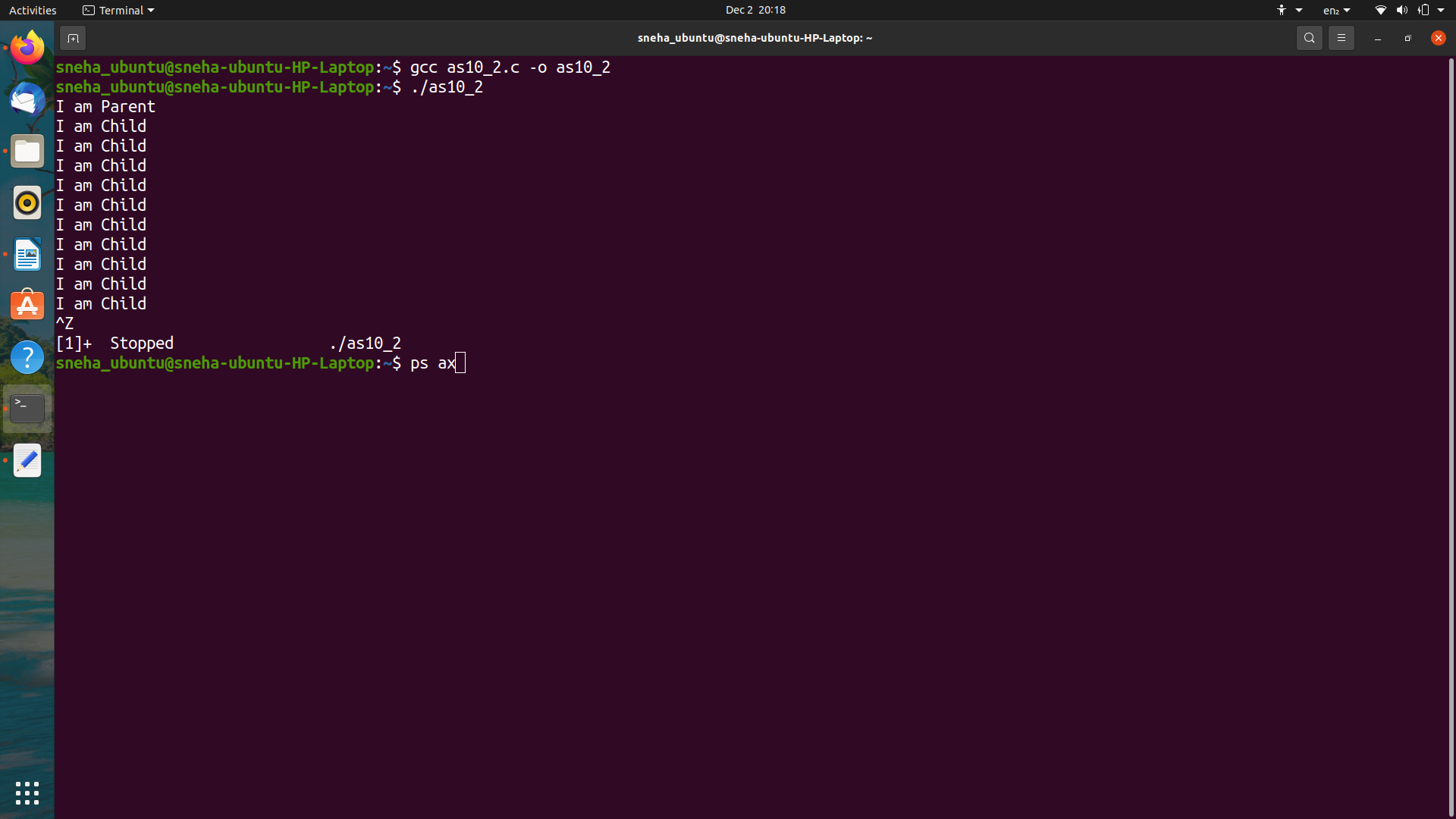
printf("I am Parent\n");

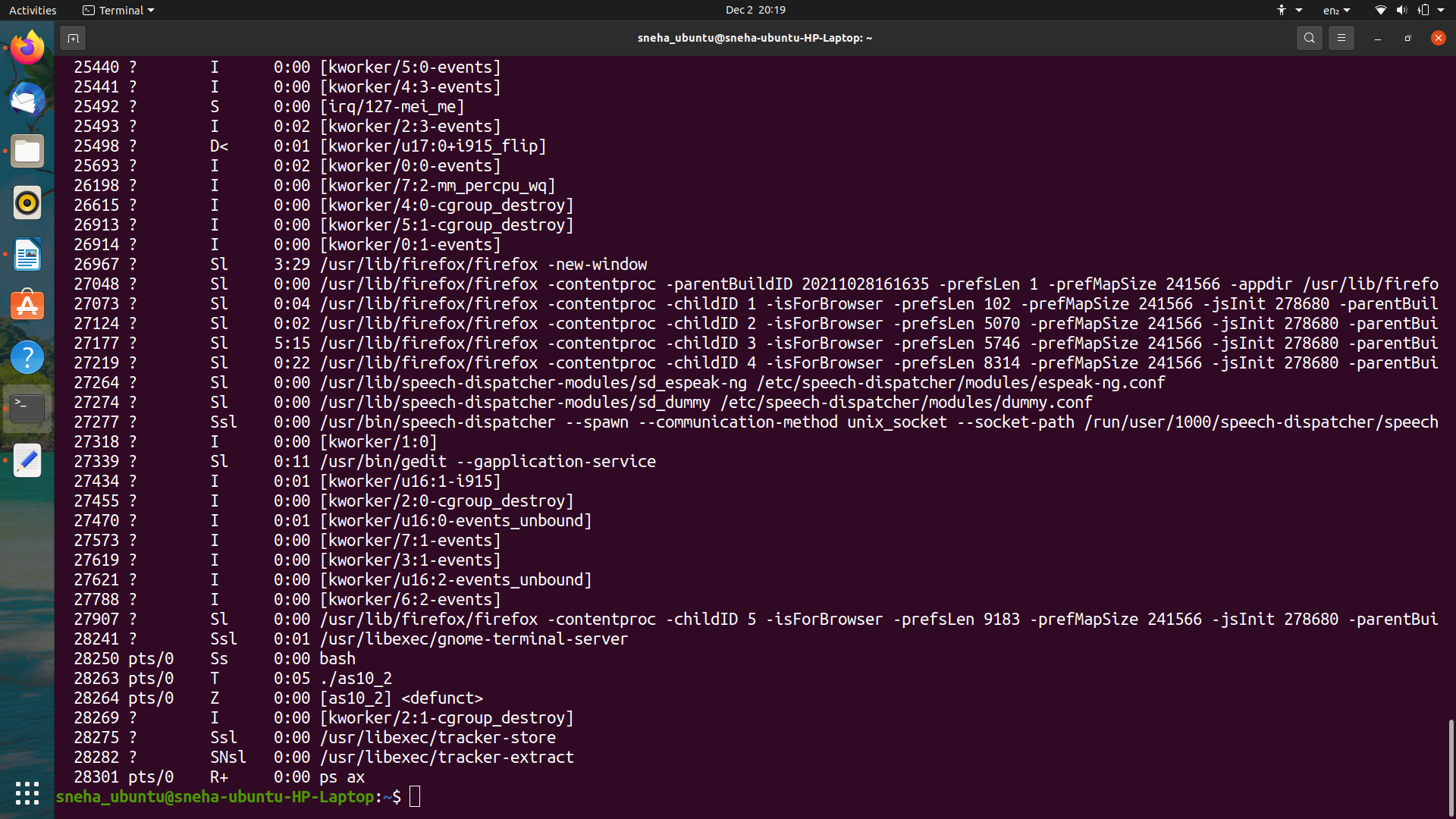
while(1);

}

}

**Output**

****

****

**3. Write a program to show the situation of a process to become an orphan.**

**Solution:**

**C Program**

#include<stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int pid = fork();

if (pid > 0)

printf("Parent process is printed.\n");

else if (pid == 0)

{

sleep(300);

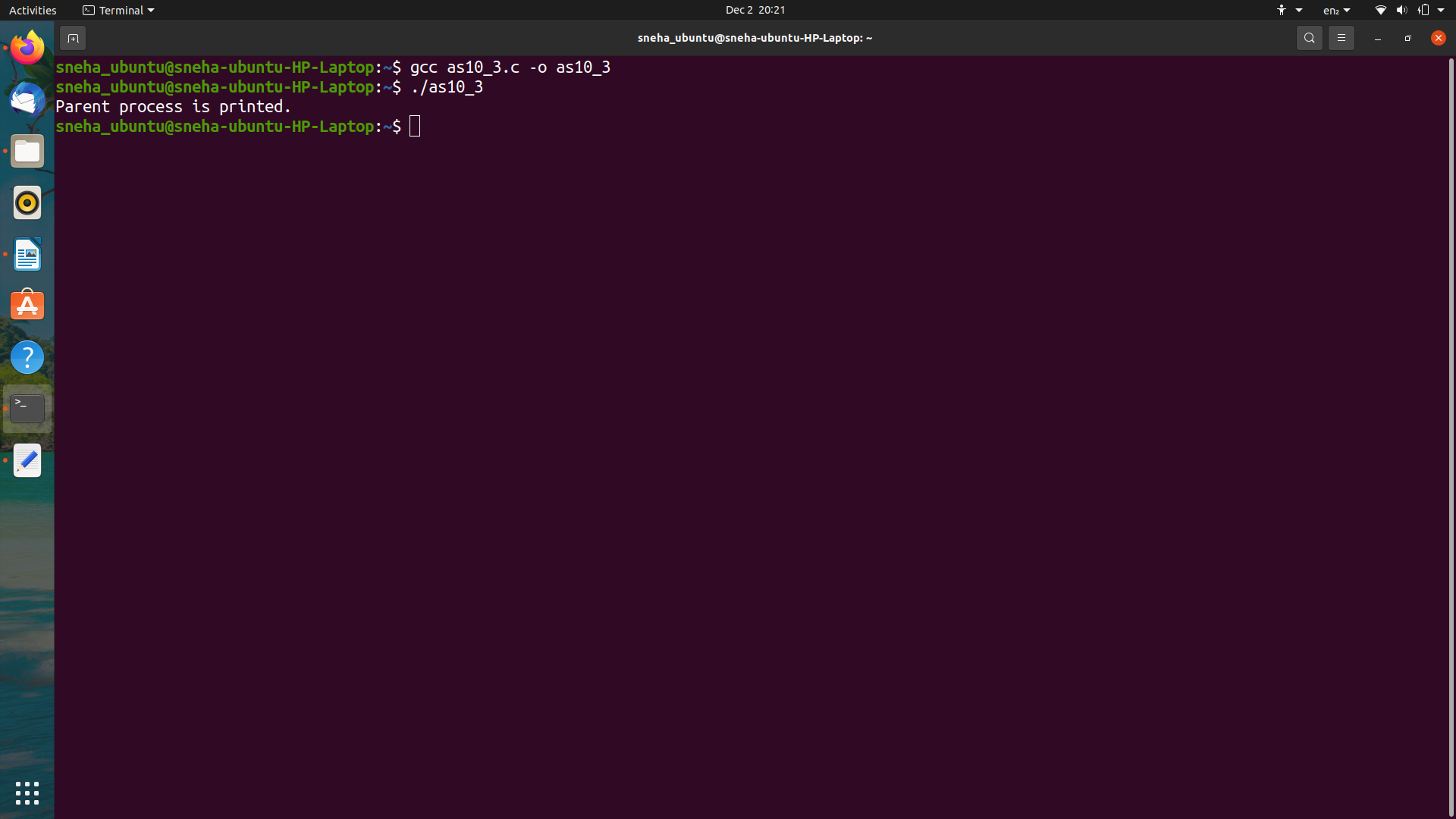
printf("Orphan process is not printed.\n");

}

return 0;

}

**Output**

****

**4. Write a program to create a child process, make the parent process intelligent enough to re spawn a new child if somehow the existing child being terminated/killed.**

**Solution:**

**C Program**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

    int pid = fork();

    if (pid > 0) {

        printf("Parent process created.\n");

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

    }

    else if (pid == 0) {

        printf("Respawning new child...\n");

        printf("New Child process created.\n");

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

        printf("Parent PID: %d\n\n", getppid());

        sleep(5);

        wait(NULL);

        printf("\nExisting Child process (which might/might not be executed)\n");

        printf("NOTE: Parent process has finished, so PPID of existing child has changed.\n");

        printf("\nPID: %d\n", getpid());

        printf("Parent PID: %d\n", getppid());

    }

    else {

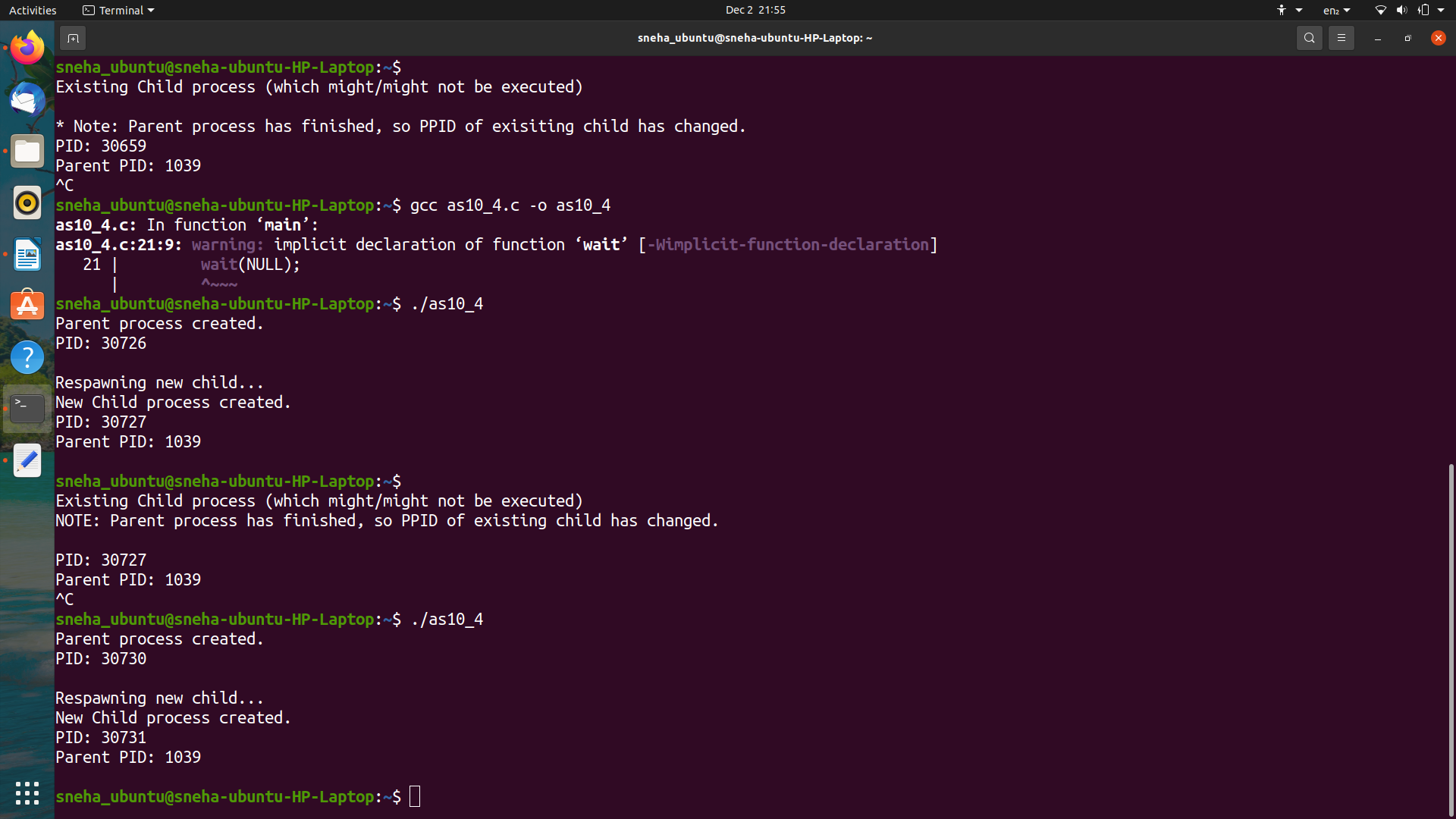
        printf("Failed to create child process");

    }

    return 0;

}

**Output**

****

-----------------------------------------------