Name: Sneha Tiwari

Roll: 11500119052 | Group: B1

Assignment 9

**1. Running multiple fork system call.**

**Solution:**

**C Program**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

fork();

fork();

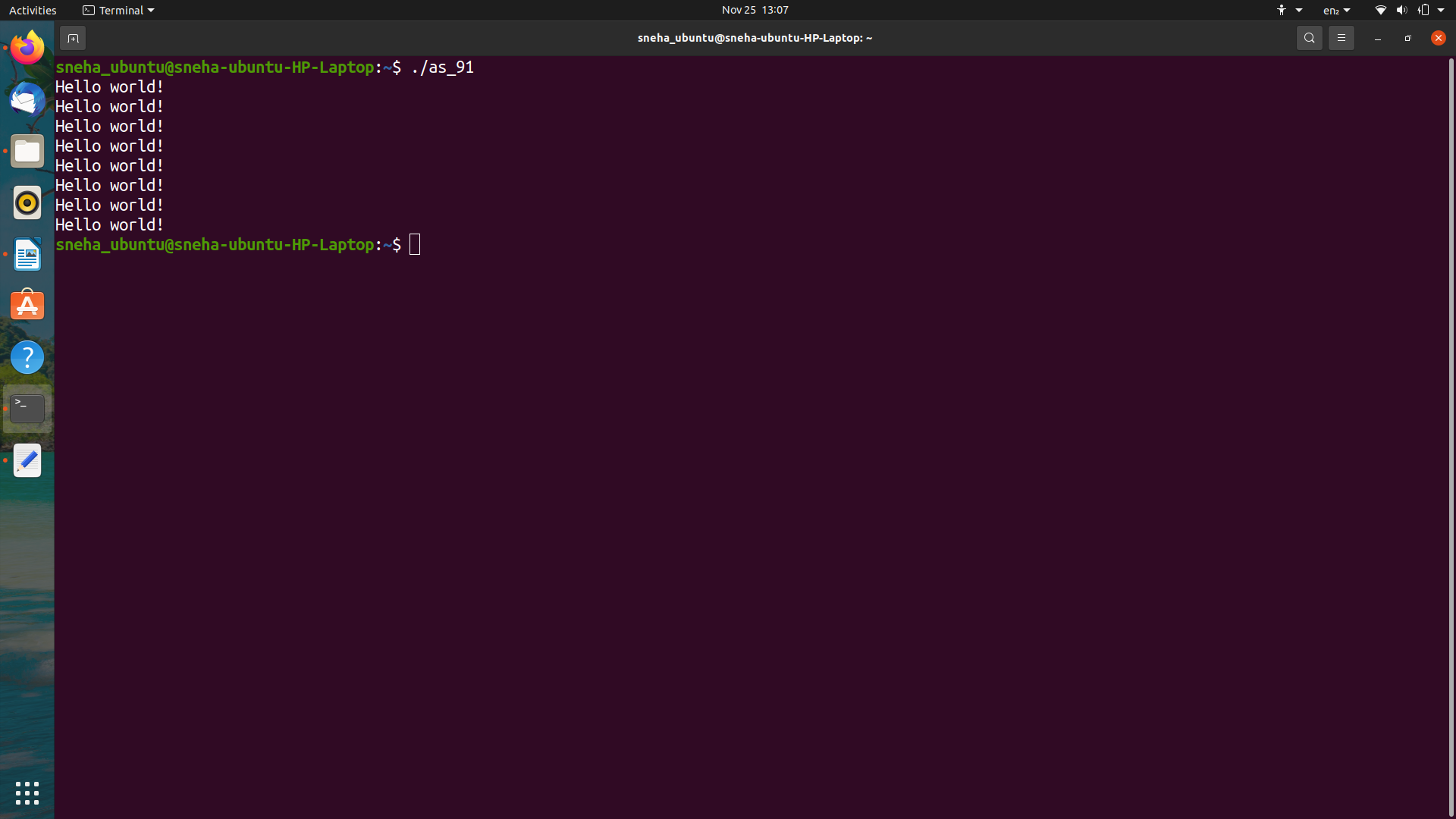
fork();

printf("Hello world!\n");

return 0;

}

**Output**



**2. Use fork () system call to execute child and parent process to check the pids.**

**Solution:**

**C Program**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main(void) {

pid\_t pid = fork();

if(pid == 0) {

printf("Child => PPID: %d \nChild => PID: %d\n", getppid(), getpid());

exit(EXIT\_SUCCESS);

}

else if(pid > 0) {

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

printf("Waiting for child process to finish.\n");

wait(NULL);

printf("Child process finished.\n");

}

else {

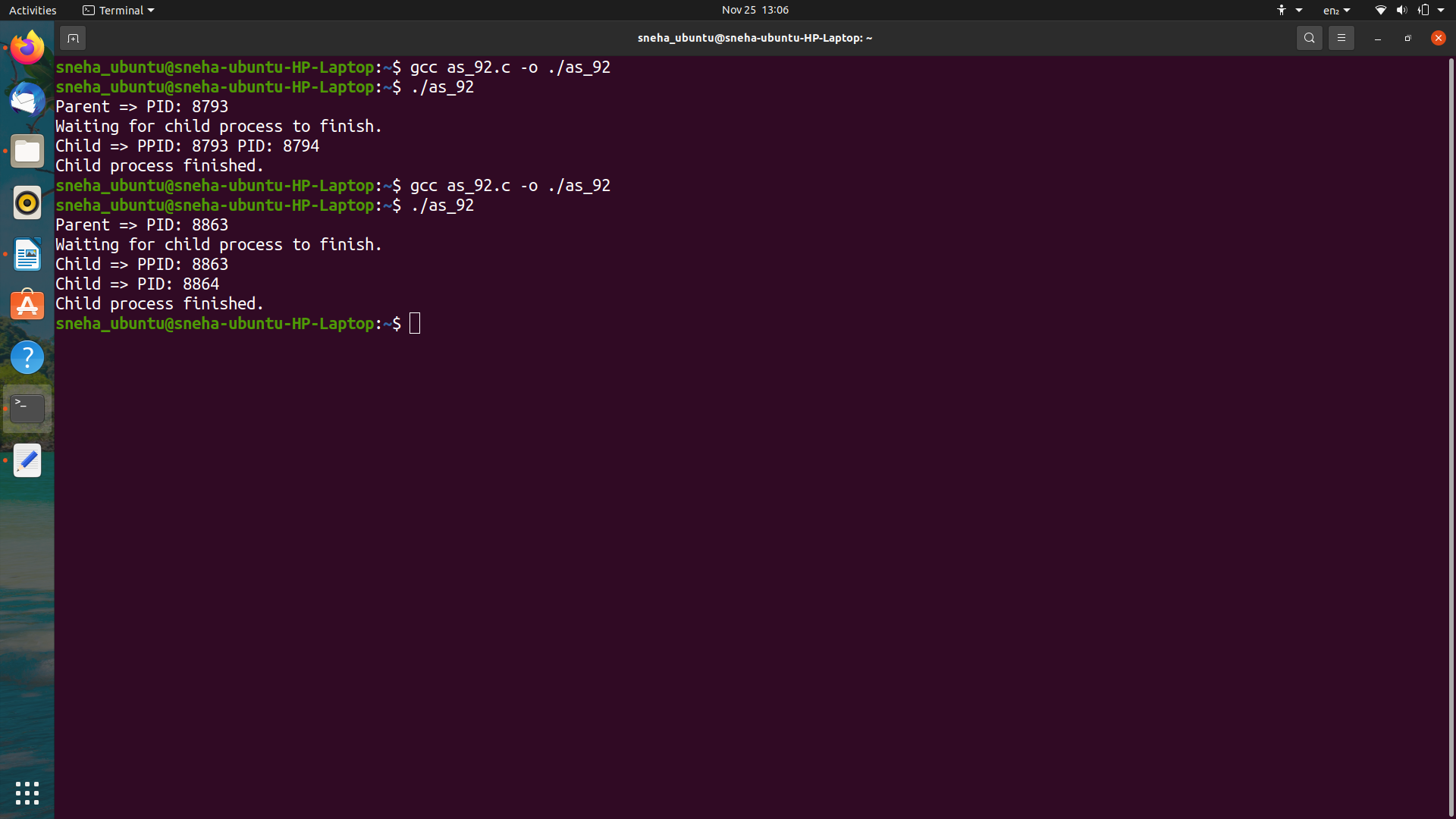
printf("Unable to create child process.\n");

}

return EXIT\_SUCCESS;

}

**Output**

****

**3. Running 3 fork() system calls to check the number of child processes.**

**Solution:**

**C Program**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int fd[2];

int count = 0;

int i;

printf("Running 3 fork() system call...\n");

pipe(fd);

for(i=0; i<3; i++) { // for 3 times fork() call

if(fork() == 0) {

write(fd[1], &i, 1);

count += 1;

}

}

close(fd[1]);

if(count == 0) { /// original process

i=0;

while(read(fd[0],&count, 1) != 0)

i += 1;

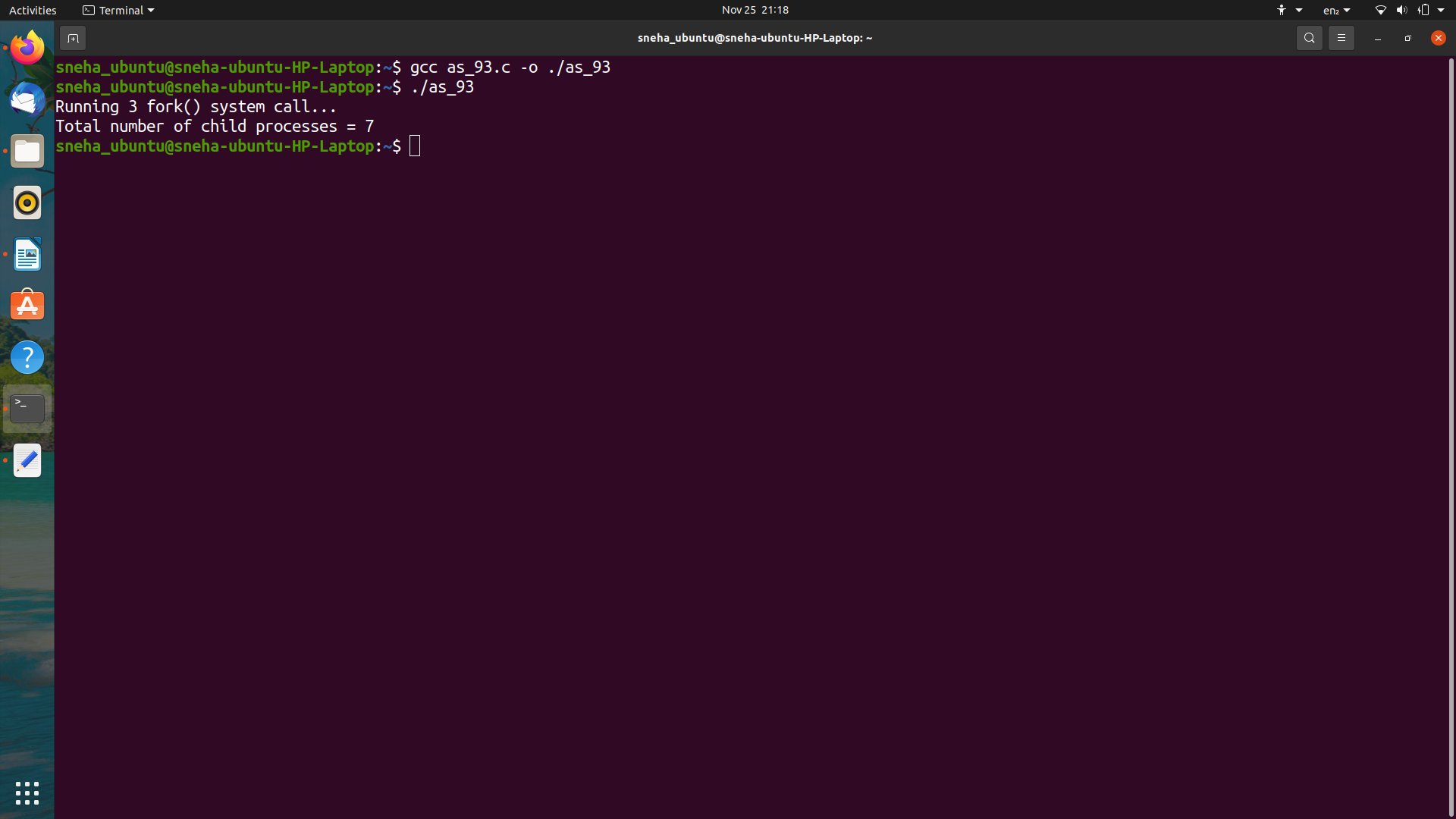
printf("Total number of child processes = %d\n", i);

}

return 0;

}

**Output**

****

**4. Check the output by using getpid to get the process ids of different process.**

**Solution:**

**C Program**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main(void) {

pid\_t pid = fork();

pid = fork();

pid = fork();

if(pid == 0) {

printf("Child => PID: %d \tChild => PPID: %d\n", getpid(), getppid());

exit(EXIT\_SUCCESS);

}

else if(pid > 0) {

printf("Parent => PID: %d \tParent => PPID: %d\n", getpid(), getppid());

wait(NULL);

}

else {

printf("Unable to create child process.\n");

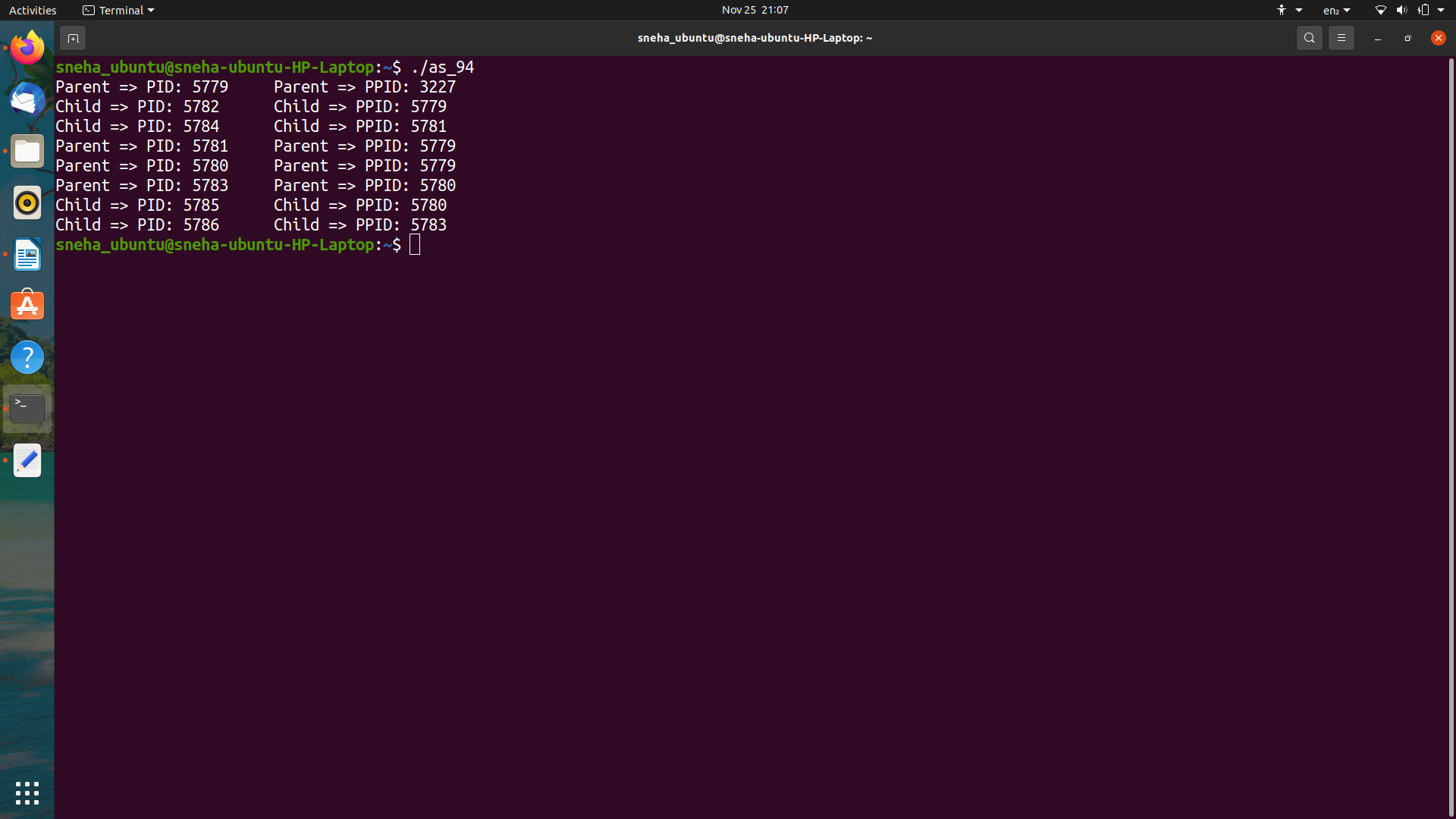
}

return EXIT\_SUCCESS;

}

**Output**

**# Here, first parent process which is the original process has separate PPID, while rest 7 duplicate processes or child processes have similar PPIDs.**

****

**5. Running multiple Child processes using fork().**

**Solution:**

**C Program**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

for(int i=0;i<5;i++)

{

if(fork() == 0)

{

printf("[child] pid %d from [parent] pid %d\n",getpid(),getppid());

exit(0);

}

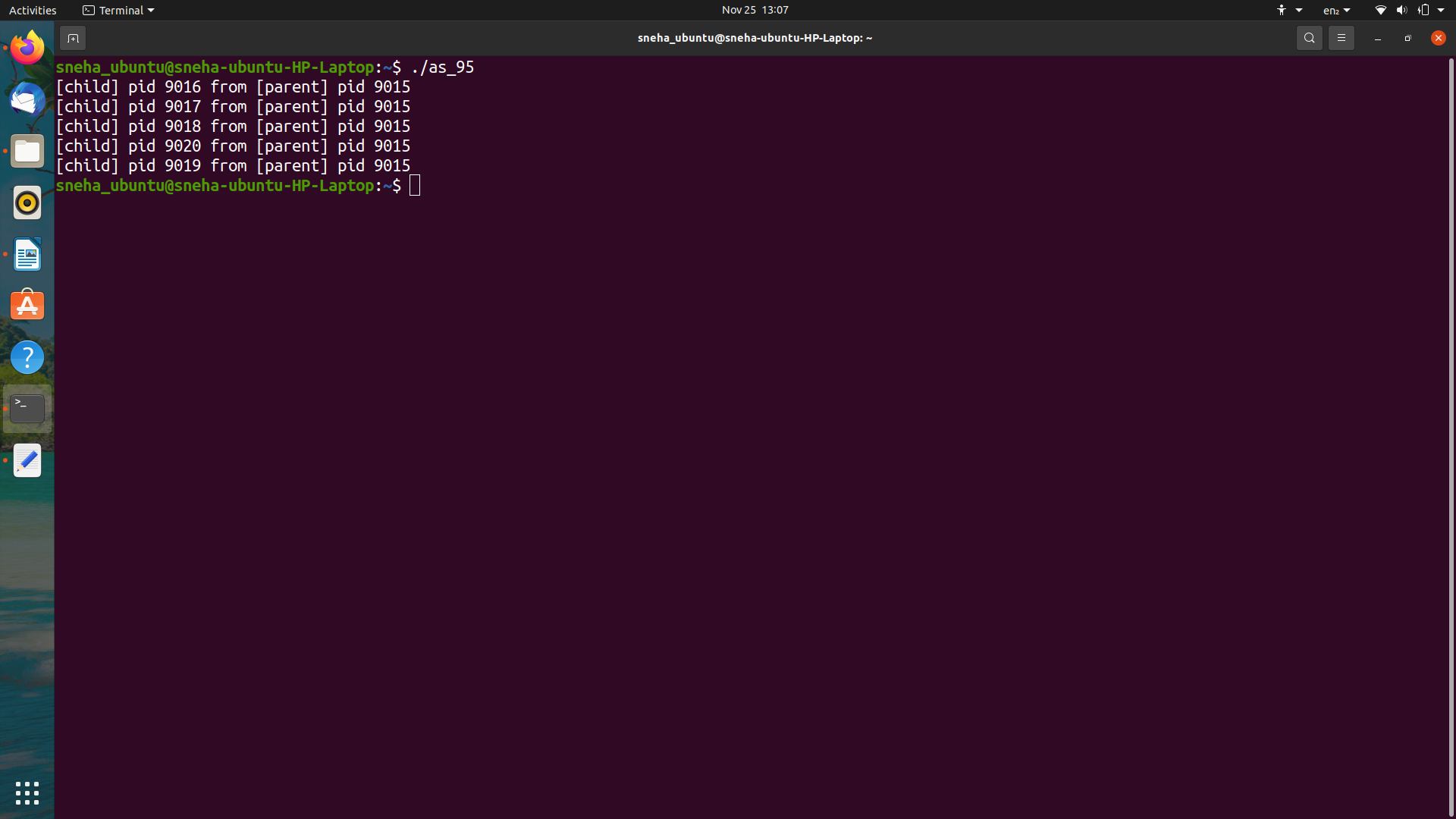
}

for(int i=0;i<5;i++)

wait(NULL);

}

**Output**

****

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