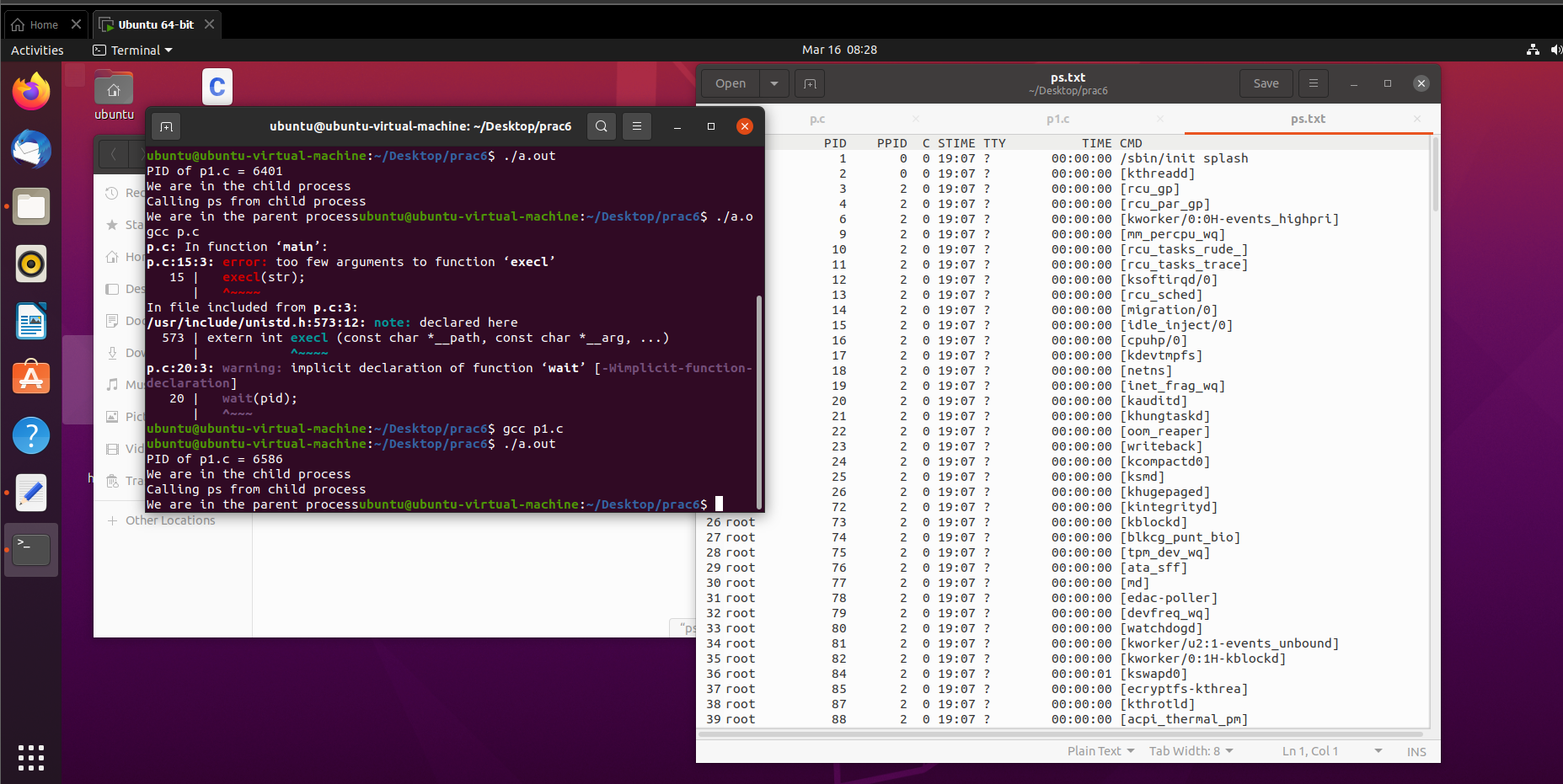
**Программа 1:**

**Результат:**

****

**Описание:**

Программа определяет в каком процессе находится выполнение программы: родительском или дочернем. Для этого описаны условия. Вызов функции fork() сопровождается возвращением статуса завершения. Где -1 равно ошибке. 0 есть дочерний процесс, а 1 родительский соответственно. Если же выполнение программы находится в дочернем процессе вызывается команда Общего количества запущенных процессов - ps, результат которой записывается в файл ps.txt

**Листинг:**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

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

{

char \* str="ps > ps.txt";

printf("PID of p1.c = %d\n", getpid());

pid\_t p;

p = fork();

if(p==-1)

{

printf("There is an error while calling fork()");

}

if(p==0)

{

printf("We are in the child process\n");

printf("Calling ps from child process\n");

execv("ps > ps.txt", argv);

}

else

{

printf("We are in the parent process");

sleep(1);

}

return 0;

}

**Программа 2:**

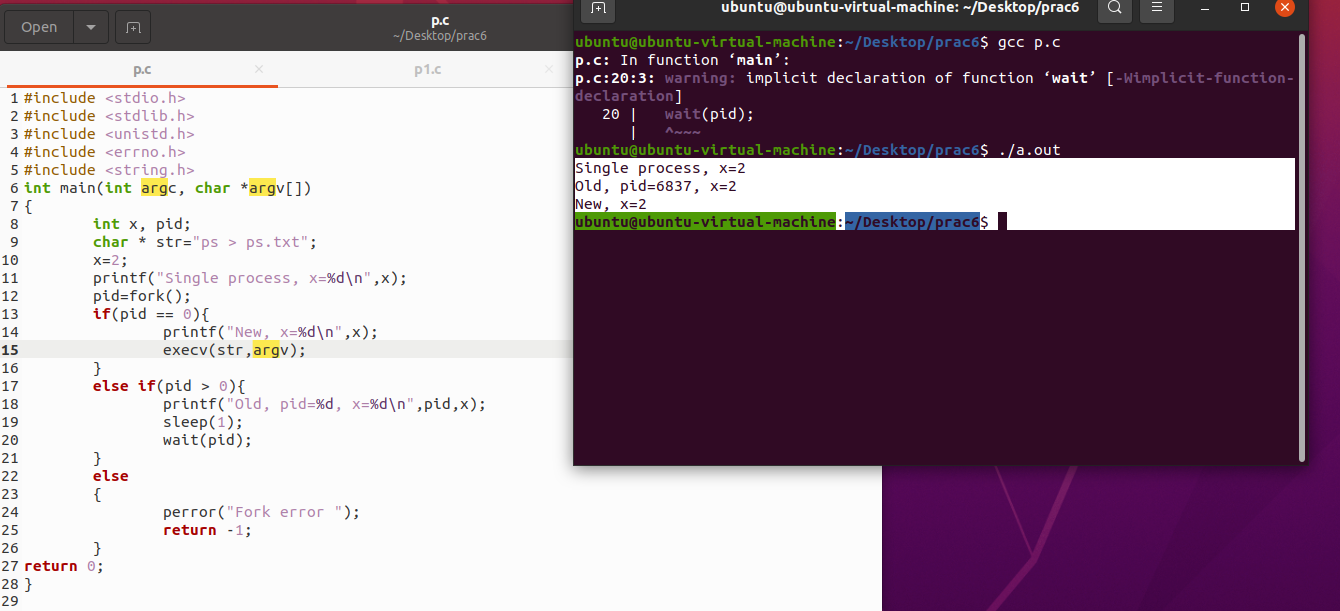
**Описание:**

Программа создает процесс если процесс дочерний то выводится код 2. Далее запускается команду str противном случае если процесс родительский т.е. pid>0. Выводится информация о том что процесс старый, его pid и код равный 2. Последнее условие отводится под ошибку запуска процесса.

Возвращается значение -1.

Продолжение ниже

**Результат:**



**Листинг:**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

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

{

char \* str="ps > ps.txt";

printf("PID of p1.c = %d\n", getpid());

pid\_t p;

p = fork();

if(p==-1)

{

printf("There is an error while calling fork()");

}

if(p==0)

{

printf("We are in the child process\n");

printf("Calling ps from child process\n");

execv("ps > ps.txt", argv);

}

else

{

printf("We are in the parent process");

sleep(1);

}

return 0;

}