|  |
| --- |
| «Санкт-Петербургский государственный электротехнический университет |
| «ЛЭТИ» им. В.И.Ульянова(Ленина)» |
| (СПбГЭТУ «ЛЭТИ») Кафедра вычислительной техники |
|  |
| **Отчет**  **по лабораторным работам № 3**  **по дисциплине** **«Организация процессов и программирование в среде Linux»** |
| Тема: «Создание и идентификация процессов» |
|  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Студент гр. 9307 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_Брызгалова Е. А. |
|  |  |
|  |  |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Разумовский Г. В. |

Оглавление

[Цель работы: 4](#_Toc115037057)

[Задание 4](#_Toc115037058)

[Распечатки файлов, содержащих параметры вызова программы и атрибуты процессов 5](#_Toc115037059)

[Тексты программ 7](#_Toc115037062)

[Текст программы main.cpp 7](#_Toc115037063)

[Текст программы 1 9](#_Toc115037064)

[Вывод 10](#_Toc115037099)

Цель работы:

Изучение и использование системных функций, обеспечивающих порождение и идентификацию процессов.

Задание

1. Разработать программу, которая порождает 2 потомка. Первый потомок порождается с помощью fork, второй – с помощью vfork с последующей заменой на другую программу. Все 3 процесса должны вывести в один файл свои атрибуты с предварительным указанием имени процесса (например: Предок, Потомок1, Потомок2). Имя выходного файла задается при запуске программы. Порядок вывода атрибутов в файл должен определяться задержками процессов, которые задаются в качестве параметров программы и выводятся в начало файла.
2. Откомпилировать программу и запустить ее 3 раза с различными сочитаниями задержек.

## **Распечатки файлов, содержащих параметры вызова программы и атрибуты процессов**

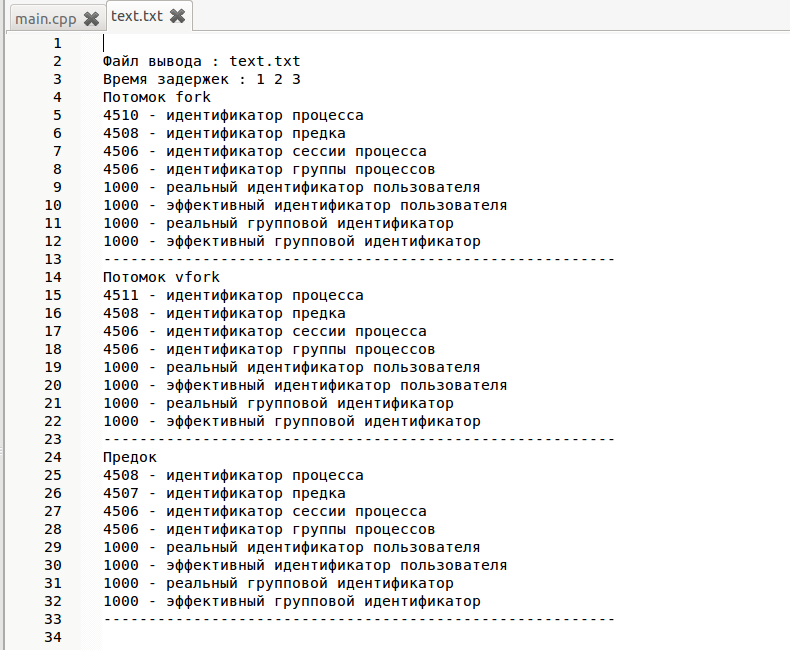


Рисунок 1

Процессы имеют одинаковые идентификаторы сессии процесса, идентификаторы группы процессов, реальные идентификаторы пользователя, эффективные идентификаторы пользователя, реальные групповые идентификаторы и эффективные групповые идентификаторы. У порожденных процессов одинаковые идентификаторы предка, совпадающие с идентификатором родительского процесса

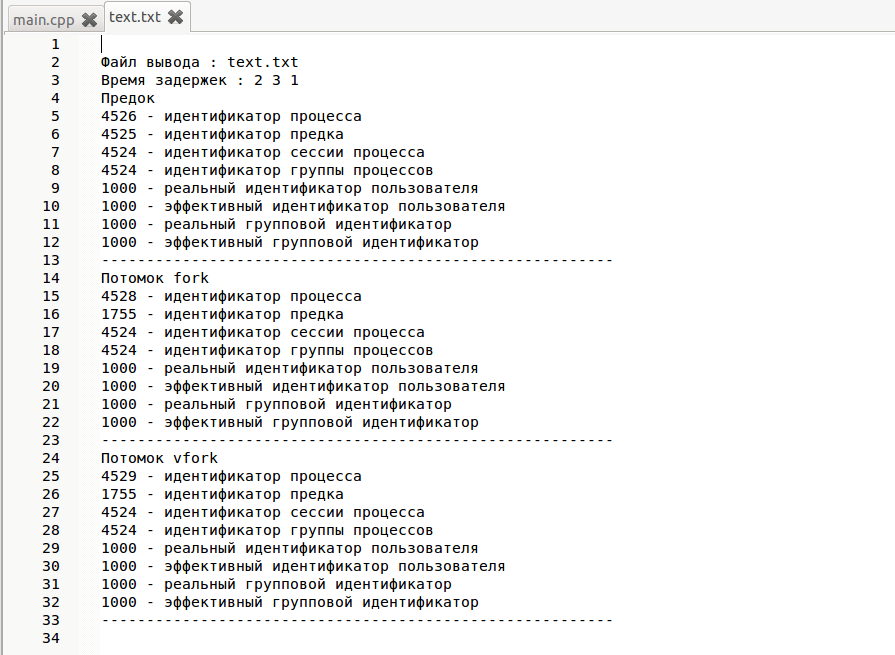


Рисунок 2

Идентификаторы предка порожденных процессов совпадают между собой, но отличаются от родительского так как он завершился раньше дочерних.

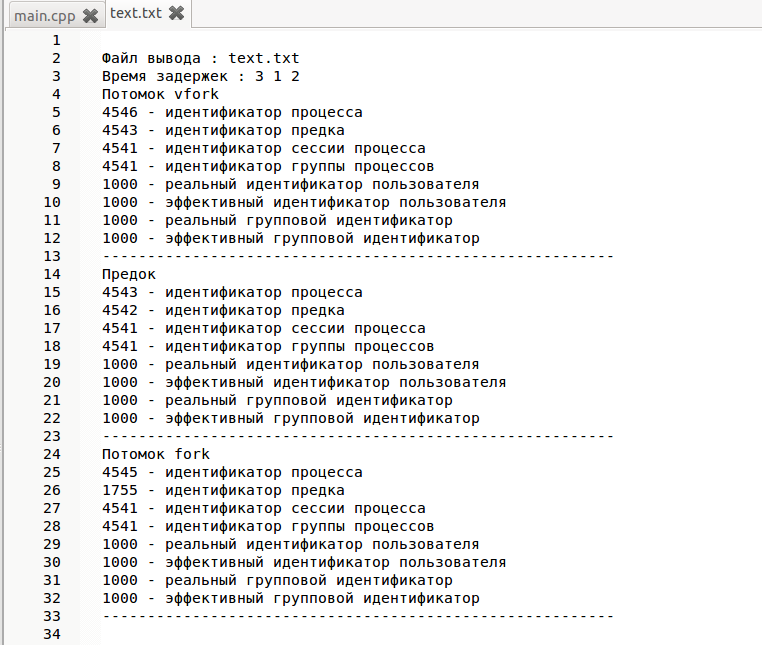


Рисунок 3

Идентификатор предка порожденного процесса vfork совпадает с родительским и отличается от идентификатора предка процесса fork. Как и в прошлом примере различия наблюдаются из-за очередности завершения процессов.

Тексты программ

### Текст программы main.cpp

#include <iostream>

#include <sys/types.h>

#include <unistd.h>

#include <fstream>

#include <signal.h>

#include <stdio.h>

#include <sys/wait.h>

#include <stdlib.h>

using namespace std;

void write\_atributs(string processName,string filename){//функция для вывода атрибутов

ofstream out;

filename = "./"+filename;

out.open(filename.c\_str(), ios::app);

if (out.is\_open()) {

pid\_t pid = getpid();

out<< processName << endl;

out << getpid()<< " - идентификатор процесса"<< endl;

out << getppid()<< " - идентификатор предка"<< endl;

out << getsid(pid)<< " - идентификатор сессии процесса"<< endl;

out << getpgid(pid)<< " - идентификатор группы процессов"<< endl;

out << getuid()<< " - реальный идентификатор пользователя"<< endl;

out << geteuid()<< " - эффективный идентификатор пользователя"<< endl;

out << getgid()<< " - реальный групповой идентификатор"<< endl;

out << getegid()<< " - эффективный групповой идентификатор"<< endl;

out<< "---------------------------------------------------------" << endl;

}

out.close();

}

int main()

{

string filename;

unsigned del1,del2,del3;

cout << "Entre delay :" << endl;

cin >>del1>>del2>>del3;

cout << "Entre file name :" << endl;

cin >>filename;

// преобразование filename

char\* c;

c = (char \*)malloc(10 \* sizeof(char));

int v = 0; //количество цифр в числе n

//разбиваем на отдельные символы число n

while (del2 > 9)

{

c[v++] = (del2 % 10) + '0';

del2 = del2 / 10;

}

c[v++] = del2 + '0';

c[v] = '\0';

char t;

//инвертируем массив символов

for (int i = 0; i < v / 2; i++)

{

t = c[i];

c[i] = c[v - 1 - i];

c[v - 1 - i] = t;

}

ofstream out;

string Filename = "./"+filename;

out.open(Filename.c\_str(), ios::app);

if (out.is\_open()) {

out << "Файл вывода :"<< " "<<filename<<endl;

out << "Время задержек :"<< " "<<del1<< " "<<del2<< " "<<del3<<endl;

}

if (!fork()) {//создание процесса с помощью fork

sleep(del1);

write\_atributs("Потомок fork",filename);

}

else{

if (!vfork()) {//создание процесса с помощью vfork

execl("./1",c,filename.c\_str(),NULL);// файлы без разрешения воспринимаются как exe

}

else{//печать атрибутов родительского процесса

sleep(del3);

write\_atributs("Предок",filename);

}

}

return 0;}

### Текст программы 1

#include <iostream>

#include <sys/types.h>

#include <unistd.h>

#include <fstream>

#include <signal.h>

#include <stdio.h>

#include <sys/wait.h>

#include <iostream>

#include <stdlib.h>

using namespace std;

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

{

string s = string(argv[0]);

int i = atoi(s.c\_str());

sleep(i);

ofstream out;

string s1 = string(argv[1]);

s1 ="./"+s1;

out.open(s1.c\_str(), ios::app);

if (out.is\_open()) {

pid\_t pid = getpid();

out<< "Потомок vfork" << endl;

out << getpid()<< " - идентификатор процесса"<< endl;

out << getppid()<< " - идентификатор предка"<< endl;

out << getsid(pid)<< " - идентификатор сессии процесса"<< endl;

out << getpgid(pid)<< " - идентификатор группы процессов"<< endl;

out << getuid()<< " - реальный идентификатор пользователя"<< endl;

out << geteuid()<< " - эффективный идентификатор пользователя"<< endl;

out << getgid()<< " - реальный групповой идентификатор"<< endl;

out << getegid()<< " - эффективный групповой идентификатор"<< endl;

out<< "---------------------------------------------------------" << endl;

}

out.close();

}

Вывод

В процессе выполнения лабораторной работы изучили системные функций, обеспечивающие порождение и идентификацию процессов. Узнали, что при завершении родительского процесса раньше дочернего, дочерним назначается новый родительский процесс.

.