## МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

## «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра Вычислительной техники**

## ОТЧЕТ

**по лабораторной работе №3**

## по дисциплине «Организация процессов и программирование в среде Linux»

**Тема: Создание и идентификация процессов**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Яловега Н.В. |
| Преподаватель |  | Разумовский Г.В. |

Санкт-Петербург

2022

# Цель работы.

Целью лабораторной работы является изучение и использование системных функций, обеспечивающих порождение и идентификацию процессов.

# Задание.

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. Откомпилировать программу и запустить ее 3 раза с различными сочетаниями задержек.

# Выполнение работы

Компиляция программ:

Рисунок 1. Компиляция программ

Запуск программы с тремя разными значениями задержек:

Рисунок 2. Запуск программ

Результат второго запуска представлен на рисунке 3. В программе-родителе сначала создается процесс с помощью fork, этот процесс ждет 2 секунды и первым пишет в файл. Затем родитель создает процесс через vfork, который vfork ждет 3 секунды, блокируя родителя. По окончании этого времени родитель разблокируется и переходит к своему ожиданию (1 секунда). За это время vfork пишет свои атрибуты. Последним в файл в пишет процесс родитель, так как он был заблокирован на время ожидания vfork.

*Рисунок 3. Результат выполнения*

Результат второго запуска представлен на рисунке 4. В программе-родителе сначала создается процесс с помощью fork, этот процесс ждет 3 секунды. Затем родитель создает процесс через vfork, который ждет 2 секунды, блокируя родителя. По окончании этого времени родитель разблокируется и переходит к своему ожиданию (1 секунда). За это время vfork пишет свои атрибуты. После этого в файл в пишет процесс родитель, так как он был заблокирован на время ожидания vfork. Последним в файл пишет процесс fork, так как у него наибольшее ожидание.

Рисунок 4. Результат выполнения

Результат третьего запуска представлен на рисунке 4. В программе-родителе сначала создается процесс с помощью fork, этот процесс ждет 1 секунду и первым пишет в файл. Затем родитель создает процесс через vfork, который vfork ждет 3 секунды, блокируя родителя. По окончании этого времени родитель разблокируется и сразу пишет в файл. В это время vfork передает управление другой программе, которая пишет в файл последней.

Рисунок 5. Результат выполнения

# Также хочется обратить внимание на то, что если процесс-родитель завершит своё выполнение до vfork-потомка, то у vfork-потомка ppid будет равен ppid systemd (init). Это связано с тем, что родитель ожидает завершения fork-потомка, но не ждёт, пока завершится vfork-потомок.

# Вывод

При выполнении лабораторной работы мы ознакомились с функциями создания процессов в системе Linux, а также с особенностями их работы и взаимоотношений предков с потомками. Кроме того мы смогли произвести запуск кода сторонней программы из основной, используя при этом дочерний процесс.

# Приложение

# main.cpp

# vfork.cpp