Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчет

Лабораторная работа №2

“Создание и управление процессов в UNIX-подобных ОС.”

Выполнили

Костюков В. С.

Пилат М. Д.

Студенты группы 121702

Проверил Цирук В. А.

Минск 2022

Цель: получить знания о процессах в операционной системе Linux.

Задание:
Написать программу, которая будет реализовывать следующие функции:

- сразу после запуска получает и сообщает свой ID и ID родительского процесса;
- перед каждым выводом сообщения об ID процесса и родительского процесса эта информация получается заново;
- порождает процессы, формируя генеалогическое дерево согласно варианту, сообщая, что "процесс с ID таким-то породил процесс с таким-то ID";
- перед завершением процесса сообщить, что "процесс с таким-то ID и таким- то ID родителя завершает работу";
- один из процессов должен вместо себя запустить программу, указанную в варианте задания.

На основании выходной информации программы предыдущего пункта изобразить генеалогическое дерево процессов (с указанием идентификаторов процессов). Объяснить каждое выведенное сообщение и их порядок в предыдущем пункте.

Индивидуальное задание:
В столбце **fork** описано генеалогическое древо процессов: каждая цифра указывает на относительный номер (не путать с pid) процесса, являющегося родителем для данного процесса. Например, строка 0 1 1 1 3 означает, что первый процесс не имеет родителя среди ваших процессов (порождается и запускается извне), второй, третий и четвертый - порождены первым, пятый - третьим.

В столбце exec указан номер процесса, выполняющего вызов **exec**, команды для которого указаны в последнем столбце. Запускайте команду обязательно с какими-либо параметрами.

5	0 1 1 2 2 3 3	5	df

Индивидуальное задание 5:

Исходный код:

```c
#include <iostream>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


int main() {

    printf("STR (%d -> %d)\n", getppid(), getpid());                    // 1st

    if (fork() == 0)
    {
        printf("2. (%d -> %d)\n", getppid(), getpid());                 // 2nd

        if (fork() == 0)
        {
            printf("4. (%d -> %d)\n", getppid(), getpid());             // 4th
        }
        else
        {
            if (fork() == 0)
            {
                printf("5. (%d -> %d)\n", getppid(), getpid());         //5th
                execl("/bin/df", "/bin/df", NULL);
            }
        }
    }
    else
    {
        if (fork() == 0)
        {
            printf("3. (%d -> %d)\n", getppid(), getpid());             // 3rd

            if(fork() == 0)
            {
                printf("6. (%d -> %d)\n", getppid(), getpid());         //6th
            }
            else
            {
                if(fork() == 0)
                {
                    printf("7. (%d -> %d)\n", getppid(), getpid());     //7th
                }
            }
        }
    }

    while(wait(NULL) > 0);
    printf("STP (%d -> %d)\n", getppid(), getpid());
    return 0;
}

```

Вывод:

```
STR (15529 -> 15588)                    //запуск родительского процесса
3. (15588 -> 15590)                     //запуск 3-его процесса
2. (15588 -> 15589)                     //запуск 2-ого процесса
4. (15589 -> 15591)                     //запуск 4-ого процесса
6. (15590 -> 15592)                     //запуск 6-ого процесса
STP (15589 -> 15591)                    //остановка 4-ого процесса, так как у него
                                        нет задач
STP (15590 -> 15592)                    //остановка 6-ого процесса, так как у него
                                        нет задач
7. (15590 -> 15594)                     //запуск 7-ого процесса
5. (15589 -> 15593)                     //запуск 5-ого процесса
STP (15590 -> 15594)                    //остановка 7-ого процесса, так как у него
                                        нет задач
STP (15588 -> 15590)                    //остановка 3-его процесса, так как его
                                         дочерние процессы завершили работу
Filesystem     1K-blocks     Used Available Use% Mounted on
tmpfs          748028        2208   745820   1% /run
/dev/nvme0n1p6  30061432 13182612  15326436  47% /
tmpfs          3740136       0   3740136   0% /dev/shm
tmpfs          5120          4       5116   1% /run/lock
/dev/nvme0n1p1      98304 53431 44873  55% /boot/efi
tmpfs          748024        4736   743288   1% /run/user/1000
STP (15588 -> 15589)                    //остановка 2-его процесса, так как его
                                         дочерние процессы завершили работу
STP (15529 -> 15588)                    //родительского процесса
//сообщение об остановке 5-ого процесса не было, так как он был заменен
командой
```

Генеалогическое древо процессов: