Министерство цифрового развития, связи
и массовых коммуникаций Российской Федерации

Сибирский государственный университет
телекоммуникаций и информатики

Кафедра прикладной математики и кибернетики

ЛАБОРАТОРНАЯ РАБОТА №4

По дисциплине: «Операционные системы»

Выполнили:

Студенты 3 курса группы ИП-111
Корнилов А.А.,
Попов М.И.,
Толкач А.А.

Проверил:

Профессор кафедры ПМиК
Малков Е.А.

Новосибирск, 2023

Задание: разработайте приложение, запускающее несколько программ. Определите идентификаторы соответствующих процессов. Установите родственные связи между ними.

Цель: получение навыков использования функций API создания процессов на платформе Linux.

Выполнение работы:

Для выполнения работы была написана программа, создающая два дочерних процесса и запускающая из них программы ps и ls.

```
int main() {
    pid_t child1, child2;
    child1 = fork();

    if (child1 < 0){
        perror("Ошибка fork()");
        exit(1);
    }
    if (child1 == 0){
        printf("Дочерний процесс 1 (PID: %d)\n", getpid());
        execlp("/bin/ls", "ls", NULL);
        perror("Ошибка execlp()");
        exit(1);
    }
…
```

В данном фрагменте программы мы создаем новый дочерний процесс «`child1 = fork();`» если он успешно создан то мы получаем его PID «`printf("Дочерний процесс 1 (PID: %d)\n", getpid());`» дочерний процесс заменяем его при помощи «`execlp("/bin/ls", "ls", NULL);`» на программу ps.

```
    else{
        child2 = fork();// Создаем второй дочерний процесс
        if (child2 < 0){
            perror("Ошибка fork()");
            exit(1);
        }

        if (child2 == 0) {//запуск ps в втором дочернем процессе
            printf("Дочерний процесс 2 (PID: %d)\n", getpid());
            execlp("/bin/ps", "ps", NULL);
            perror("Ошибка execlp()");
            exit(1);

…
```

Точно также поступаем с вызовом ls, мы создаем второй дочерний процесс и если он успешно запустился то получаем его PID и запускаем программу ls.

```
        } else {
            printf("Родительский процесс (PID: %d)\n", getpid());
            getchar();
            wait(NULL);
            wait(NULL);
            printf("Оба дочерних процесса завершились.\n");
        }
    }

    return 0;
}
```

В конце программы получаем родительский PID или PPID «`printf("Родительский процесс (PID: %d)\n", getpid());`» и ждем завершения дочерних процессов

```
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/Sibsutis/3 Year/OS/4$ ps -e -o pid,ppid,pgid,sid,state,command | grep 1081
  1081     458    1081    458 S ./prog1
  1082    1081    1081    458 Z [ls] <defunct>
  1083    1081    1081    458 Z [ps] <defunct>
  1085     894    1084    894 S grep --color=auto 1081
```

Во время остановки программы вызовом «`getchar();`» вызываем таблицу ps и отбираем процессы с указанием нашего родительского PID. В результате мы видим что дочерние процесс 1082 и 1083 уже завершились и находиться в состоянии Z и ждут выполнения родительского процесса.