Федеральное агентство связи
Сибирский государственный университет телекоммуникаций и информатики

Лабораторная работа №4

Выполнил: студент группы ИП-211
Оганесян Альберт
Лацук Андрей
Проверил:
Профессор кафедры ПМиК
Малков Е. А.

Новосибирск 2024

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

Задание: разработать приложение, запускающее несколько программ. Определить идентификаторы соответствующих процессов. Установить родственные связи между ними.

Инструментарий:

Редактор кода VSCode, компилятор GCC.

Ход работы:

1. **Создаем процессы:** В цикле for мы вызываем fork(), создавая дочерний процесс. Возвращаемое значение fork() позволяет нам различать родительский и дочерний процессы. Если fork() возвращает 0, это дочерний процесс, в противном случае — родительский.

```
for (int i = 0; i < NUM_PROC; i++)
{
    pids[i] = fork(); // Создаем новый процесс

    if (pids[i] < 0)
    {
        perror("fork failed"); // Проверяем на ошибки
        exit(EXIT_FAILURE);
    }

    if (pids[i] == 0)
    { // В дочернем процессе
        // Выполняем программу
        execlp(programs[i], programs[i], NULL);
        perror("execlp failed"); // Проверяем на ошибки execlp
        exit(EXIT_FAILURE);      // Завершаем в случае ошибки
    }
}

// В родительском процессе
for (int i = 0; i < NUM_PROC; i++)
{
    int status;
    waitpid(pids[i], &status, 0); // Ждем завершения дочернего процесса
    if (WIFEXITED(status))
    {
        printf("\nChild process %d: PID = %d, PPID = %d\n", i, getpid(), getppid());
        printf("finished with exit status %d\n", WEXITSTATUS(status));
    }
}
```

Рис. 1 Фрагмент кода с созданием процессов

2. **Выполнение программ:** В дочернем процессе вызывается execlp(), который заменяет его образ на программу, указанную в массиве programs. Мы указываем полный путь программы и не передаем дополнительных аргументов, следовательно, передаем NULL.

```
if (pids[i] == 0)
{ // В дочернем процессе
    // Выполняем программу
    execlp(programs[i], programs[i], NULL);
    perror("execlp failed"); // Проверяем на ошибки execlp
    exit(EXIT_FAILURE);      // Завершаем в случае ошибки
}
```

Рис. 2 Фрагмент кода с выполнением программ

3. **Ожидание завершения:** После создания всех дочерних процессов, родительский процесс переходит в режим ожидания завершения каждого из них с помощью waitpid(). Этот вызов также позволяет обработать статус завершения каждого дочернего процесса.

```
for (int i = 0; i < NUM_PROC; i++)
{
    int status;
    waitpid(pids[i], &status, 0); // Ждем завершения дочернего процесса
    if (WIFEXITED(status))
    {
        printf("\nChild process %d: PID = %d, PPID = %d\n", i, getpid(), getppid());
        printf("finished with exit status %d\n", WEXITSTATUS(status));
    }
}
```

Рис. 3 Фрагмент кода с ожиданием завершения дочерних процессов

4. Структуру процессов можно изобразить следующим образом:

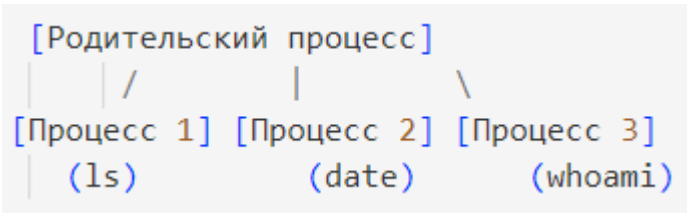

Рис. 4 Структура процессов

5. **Вывод:** При выполнении приложение выводит информацию о процессах (их PID и PPID). При выполнении программ (например, ls, date, whoami) мы получаем вывод, соответствующий их выполнению. В конце для каждого дочернего процесса выводится статус завершения.

```
danyowoj@danyowojlaptop:/mnt/c/Users/latsu/GitHub_projects/OS/4$ ./main
Parent process: PID = 526
Executing programs...

Tue Oct 15 11:41:51 +07 2024
danyowoj
main  main.c

Child process 0: PID = 14189, PPID = 526
finished with exit status 0

Child process 1: PID = 14189, PPID = 526
finished with exit status 0

Child process 2: PID = 14189, PPID = 526
finished with exit status 0
```

Рис. 5 Вывод программы

Вывод: Мы получили навыки использования функций API и научились созданию процессов на платформе Linux.