Лабораторная работа № 2 по курсу : Операционные системы

Выполнил студент группы М8О-206Б-17 *Новиков Павел Сергеевич.*

Условие

- Постановка задачи:

  Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в операционной системе Linux. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

- Вариант задания

  Вычисление факториала

Метод решения

Каждый уровень рекурсии выполняется в отдельном процессе. Процесс создается с помощью системного вызова fork.

Описание программы

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int fd[2];

#define READ    0
#define WRITE   1

int factorial (int n) {
    if (n == 0) return 1;
    pid_t pid;
    int res;
    pid = fork();
```

```
    if (pid < 0) {
        perror("fork");
    } else if (pid > 0) {
        wait(0);
        read(fd[READ], &res, sizeof(res));
    } else if (pid == 0) {
        res = factorial(n - 1);
        close(fd[READ]);
        if (write(fd[WRITE], &res, sizeof(res)) < 0) {
            perror("write");
        }
        exit(0);
    }
    return n * res;
}

int main() {
    if (pipe(fd) < 0) {
        perror("pipe");
        return 0;
    }
    int n = 0;

    printf("Enter N > 0 to calculate Factorial(N)\n");
    scanf("%d", &n);
    if (n < 0) {
        printf("Bad input\n");
        return 0;
    }
    printf("Factorial(N) equals:\n%d\n", factorial(n));

    return 0;
}
```

Вывод strace

```
Enter N > 0 to calculate Factorial(N)
5
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f4f38f02810) = 16829
```

```
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED,
si_pid=16829, si_uid=1000, si_status=0, si_utime=0,
si_stime=0} ---
Factorial(N) equals:
120
+++ exited with 0 +++
```

Выводы

Вызов *fork* дублирует породивший его процесс со всеми его переменными, файловыми дескрипторами, приоритетами процесса, рабочий и корневой каталоги, и сегментами выделенной памяти.

Ребёнок **не** наследует:

- идентификатора процесса (PID, PPID);
- израсходованного времени ЦП (оно обнуляется);
- сигналов процесса-родителя, требующих ответа;
- блокированных файлов (record locking).

В процессе выполнения лабораторной работы были приобретены навыки практического применения создания, обработки и отслеживания их состояния. Для выполнения данного варианта задания создание потоков как таковых не требуется, так как всю работу выполняет системный вызов «exec».