

Лабораторная 4.

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

Для разработки приложения, которое запускает несколько программ и определяет идентификаторы соответствующих процессов в Linux, необходимо использовать системные вызовы `fork()` и `exec()`. Эти функции позволяют создавать дочерние процессы и запускать в них новые программы.

1. Для создания нового процесса используется системный вызов `fork()`. Этот вызов создает копию текущего процесса (родителя), и оба процесса (родитель и потомок) продолжают выполнение с точки, где был вызван `fork()`.

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

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        perror("Ошибка fork");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        printf("Дочерний процесс с PID: %d\n", getpid());
        exit(EXIT_SUCCESS);
    } else {
        printf(
            "Родительский процесс с PID: %d, создал дочерний процесс с PID: %d\n",
            getpid(), pid);
    }

    return 0;
}
```

2. После создания дочернего процесса, можно использовать функции из семейства `exec` для запуска других программ, например, `exec()` или `execvp()`.

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

int main() {
    pid_t pid = fork();
```

```

if (pid < 0) {
    perror("Ошибка fork");
    exit(EXIT_FAILURE);
} else if (pid == 0) {
    execl("/bin/ls", "ls", "-l", NULL);
    perror("Ошибка exec");
    exit(EXIT_FAILURE);
} else {
    printf(
        "Родительский процесс с PID: %d, создал дочерний процесс с PID: %d\n",
        getpid(), pid);
    wait(NULL);
}

return 0;
}

```

3. Каждый созданный дочерний процесс имеет уникальный идентификатор - PID, который можно использовать для отслеживания их состояния. Родительский процесс может ожидать завершения своих дочерних процессов с помощью функции wait().

```

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

int main() {
    for (int i = 0; i < 3; i++) {
        pid_t pid = fork();

        if (pid < 0) {
            perror("Ошибка fork");
            exit(EXIT_FAILURE);
        } else if (pid == 0) {
            printf("Дочерний процесс #%d с PID: %d\n", i + 1, getpid());
            execl("/bin/sleep", "sleep", "2", NULL);
            perror("Ошибка exec");
            exit(EXIT_FAILURE);
        }
    }

    for (int i = 0; i < 3; i++) {
        wait(NULL);
    }
}

```

```

}
printf("Все дочерние процессы завершены.\n");
return 0;
}

```

Результат работы программ:

```

tania@TaniaLaptop:~/5sem/os/os_labs/lab4$ gcc -o 4 lab4.c
tania@TaniaLaptop:~/5sem/os/os_labs/lab4$ ./4
Родительский процесс с PID: 167290, создал дочерний процесс с PID: 167291
Дочерний процесс с PID: 167291
tania@TaniaLaptop:~/5sem/os/os_labs/lab4$ gcc -o 41 lab4_1.c
tania@TaniaLaptop:~/5sem/os/os_labs/lab4$ ./41
Родительский процесс с PID: 167393, создал дочерний процесс с PID: 167394
total 96
-rwxr-xr-x 1 tania tania 16128 Nov 10 22:10 4
-rwxr-xr-x 1 tania tania 16216 Nov 10 22:10 41
-rwxr-xr-x 1 tania tania 16256 Nov 10 17:58 42
-rw-r--r-- 1 tania tania 498 Nov 10 22:00 lab4.c
-rw-r--r-- 1 tania tania 539 Nov 10 22:07 lab4_1.c
-rw-r--r-- 1 tania tania 601 Nov 10 22:08 lab4_2.c
-rw-r--r-- 1 tania tania 36410 Nov 8 23:13 Ла64-2023.pdf
tania@TaniaLaptop:~/5sem/os/os_labs/lab4$ gcc -o 42 lab4_2.c
tania@TaniaLaptop:~/5sem/os/os_labs/lab4$ ./42
Дочерний процесс #1 с PID: 167485
Дочерний процесс #2 с PID: 167486
Дочерний процесс #3 с PID: 167487
Все дочерние процессы завершены.
tania@TaniaLaptop:~/5sem/os/os_labs/lab4$ |

```

Заключение:

Используя системные вызовы `fork()` и `exec()`, можно эффективно создавать и управлять несколькими процессами в Linux. Каждый созданный процесс имеет свой уникальный идентификатор, который позволяет отслеживать его состояние и устанавливать родственные связи между ними.

Используемые источники:

<https://habr.com/ru/articles/447182/>