Федеральное агентство связи
Сибирский государственный университет телекоммуникаций и информатики

Лабораторная работа №3

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

Новосибирск 2024

Цель: знакомство с процессами Linux.

Упражнение 1. Протестируйте программы, рассмотренные на Лекции 3.

Упражнение 2. Создайте процесс с помощью вызова fork, с помощью команд ps и grep получите информацию о созданных вами родительском и дочернем процессах. Используя команду kill убейте родительский процесс, продолжил ли выполнялся дочерний процесс?

Упражнение 3. Создайте дерево процессов с помощью вызова fork. С помощью команды pstree найдите поддерево созданных процессов. В каталоге /proc виртуальной файловой системы найдите папки с именами, совпадающими с идентификаторами созданных процессов, и просмотрите содержимое папок

task/children.

Ход работы:

1. Воссоздадим программы, показанные на лекции и проверим их работоспособность (рис. 1.1) (рис. 1.2) (рис. 1.3)

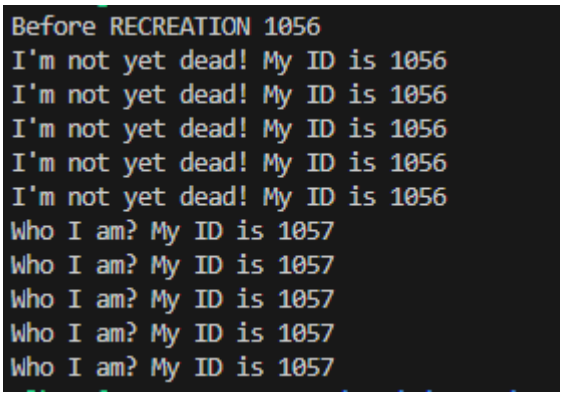

Рис. 1.1 результат работы первого примера

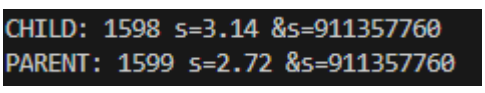

Рис. 1.2 результат работы второго примера

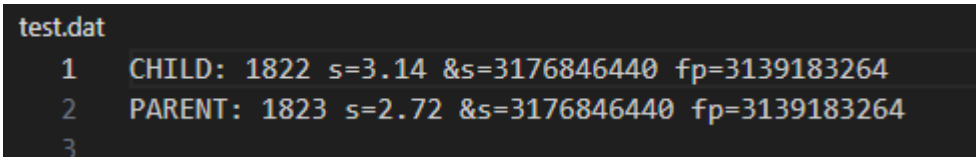

Рис. 1.3 результат работы третьего примера

2. Напишем программу для создания дерева процессов при помощи fork

Код программы:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

int main() {
    pid_t pid = fork();
    if (pid < 0) {
        perror("fork failed");
        exit(1);
    } else if (pid == 0) {
        while (1) {
            printf("Дочерний процесс (PID: %d) выполняется...\n", getpid());
            sleep(1);
        }
    } else {
        printf("Родительский процесс (PID: %d) создал дочерний процесс (PID: %d)\n", getpid(), pid);
        sleep(20);
        kill(pid, SIGTERM);
    }

    return 0;
}
```

Теперь скомпилируем и запустим программу (рис. 2.1) после чего откроем новый терминал и при помощи команды ps aux | grep unit2

посмотрим pid процессов (рис. 2.2) и убьем родительский процесс при помощи команды kill после этого заметим, что дочерний процесс продолжил работать (рис. 2.3)

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/3$ ./unit2
Родительский процесс (PID: 19076) создал дочерний процесс (PID: 19077)
Дочерний процесс (PID: 19077) выполняется...
Дочерний процесс (PID: 19077) выполняется...
Дочерний процесс (PID: 19077) выполняется...
Дочерний процесс (PID: 19077) выполняется...
Дочерний процесс (PID: 19077) выполняется...
Дочерний процесс (PID: 19077) выполняется...
```

Рис. 2.1 выполнение программы

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/3$ ps aux | grep unit2
albert     19451  0.0  0.0   2776   936 pts/0    S+   14:32   0:00 ./unit2
albert     19452  0.0  0.0   2776    88 pts/0    S+   14:32   0:00 ./unit2
albert     19506  0.0  0.0   4028  2072 pts/2    S+   14:32   0:00 grep --color=auto unit2
```

Рис. 2.2 PID дочерних процессов

```
Дочерний процесс (PID: 19452) выполняется...
Terminated
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/3$ Дочерний процесс (PID: 19452) выполняется...
Дочерний процесс (PID: 19452) выполняется...
Дочерний процесс (PID: 19452) выполняется...
Дочерний процесс (PID: 19452) выполняется...
Дочерний процесс (PID: 19452) выполняется...
Дочерний процесс (PID: 19452) выполняется...
Дочерний процесс (PID: 19452) выполняется...
Дочерний процесс (PID: 19452) выполняется...
```

Рис. 2.3 продолжение выполнения дочернего процесса

3. Теперь напишем программу для создания двух дочерних процессов и запустим её (рис. 3.1)

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/3$ ./unit3
Родительский процесс (PID: 24500)
Первый дочерний процесс (PID: 24501, родитель: 24500)
Второй дочерний процесс (PID: 24502, родитель: 24501)
```

Рис. 3.1 вывод программы

Теперь в отдельном терминале откроем поддерево созданных процессов (рис. 3.2) при помощи команды pstree -p | grep unit3

```
|-init-systemd(Ub(2)-+-SessionLeader(417)---Relay(420)(418)---bash(420)---unit3(29753)---unit3(29754)---unit3(29755)
```

Рис. 3.2 поддерево созданных процессов

И просмотрим файлы из директории /proc (рис. 3.3) (рис. 3.4) (рис. 3.5)

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/3$ cat /proc/29753/status
Name:   unit3
Umask:  0022
State:  S (sleeping)
Tgid:   29753
Ngid:   0
Pid:    29753
PPid:   420
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
FDSize: 256
Groups: 4 20 24 25 27 29 30 44 46 116 1000
NStgid: 29753
NSpid:  29753
NSpgid: 29753
NSsid:  420
VmPeak:     2776 kB
VmSize:     2776 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:       956 kB
VmRSS:       956 kB
RssAnon:             92 kB
RssFile:            864 kB
RssShmem:             0 kB
VmData:      224 kB
VmStk:       132 kB
VmExe:         4 kB
VmLib:      1796 kB
VmPTE:        40 kB
VmSwap:        0 kB
HugetlbPages:          0 kB
CoreDumping:    0
THP_enabled:    1
Threads:        1
SigQ:   1/31162
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 000001ffffffffff
CapAmb: 0000000000000000
NoNewPrivs:     0
Seccomp:        0
Seccomp_filters:        0
Speculation_Store_Bypass:       thread vulnerable
SpeculationIndirectBranch:      conditional enabled
Cpus_allowed:   fff
Cpus_allowed_list:      0-11
Mems_allowed:   1
Mems_allowed_list:      0
voluntary_ctxt_switches:        14
```

Рис. 3.3 информация о родительском процессе

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/3$ cat /proc/29754/status
Name:   unit3
Umask:  0022
State:  S (sleeping)
Tgid:   29754
Ngid:   0
Pid:    29754
PPid:   29753
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
FDSize: 64
Groups: 4 20 24 25 27 29 30 44 46 116 1000
NStgid: 29754
NSpid:  29754
NSpgid: 29753
NSsid:  420
VmPeak:     2776 kB
VmSize:     2776 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:        92 kB
VmRSS:        92 kB
RssAnon:              92 kB
RssFile:               0 kB
RssShmem:              0 kB
VmData:      224 kB
VmStk:       132 kB
VmExe:         4 kB
VmLib:      1796 kB
VmPTE:        40 kB
VmSwap:        0 kB
HugetlbPages:          0 kB
CoreDumping:    0
THP_enabled:    1
Threads:        1
SigQ:   1/31162
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 000001ffffffffff
CapAmb: 0000000000000000
NoNewPrivs:     0
Seccomp:        0
Seccomp_filters:        0
Speculation_Store_Bypass:       thread vulnerable
SpeculationIndirectBranch:      conditional enabled
Cpus_allowed:   fff
Cpus_allowed_list:      0-11
Mems_allowed:   1
Mems_allowed_list:      0
voluntary_ctxt_switches:        1
nonvoluntary_ctxt_switches:     0
```

Рис. 3.4 информация о первом дочернем процессе

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/3$ cat /proc/29755/status
Name:   unit3
Umask:  0022
State:  S (sleeping)
Tgid:   29755
Ngid:   0
Pid:    29755
PPid:   29754
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
FDSize: 64
Groups: 4 20 24 25 27 29 30 44 46 116 1000
NStgid: 29755
NSpid:  29755
NSpgid: 29753
NSsid:  420
VmPeak:     2776 kB
VmSize:     2776 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:        92 kB
VmRSS:        92 kB
RssAnon:              92 kB
RssFile:               0 kB
RssShmem:              0 kB
VmData:      224 kB
VmStk:       132 kB
VmExe:         4 kB
VmLib:      1796 kB
VmExe:         4 kB
VmLib:      1796 kB
VmPTE:        40 kB
VmSwap:        0 kB
HugetlbPages:          0 kB
CoreDumping:    0
THP_enabled:    1
Threads:        1
SigQ:   1/31162
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 000001ffffffffff
CapAmb: 0000000000000000
NoNewPrivs:     0
Seccomp:        0
Seccomp_filters:        0
Speculation_Store_Bypass:       thread vulnerable
SpeculationIndirectBranch:      conditional enabled
Cpus_allowed:   fff
Cpus_allowed_list:      0-11
Mems_allowed:   1
Mems_allowed_list:      0
```

Рис. 3.5 информация о втором дочернем процессе

Вывод: мы изучили познакомились с процессами Linux, научились создавать дочерние процессы при помощи fork(), узнали, что если удалить родительский процесс, то это не завершит дочерние и научились получать различную информацию о процессах