МГТУ им. Баумана

Лабораторная работа №4

По курсу: "Операционные системы"

Процессы. Системные вызовы fork() и exec()

Работу выполнил: студент группы ИУ7-53Б
Наместник Анастасия

Преподаватель: Рязанова Н. Ю.

Москва, 2020

На листинге 1 представлена первая программ, демонстрирующая появление процессов-сирот.

Листинг 1: Задание 1

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

int main()
{
    int process1;
    if ((process1 = fork()) == -1)
    {
        perror("Can\'t fork.\n");
        exit(1);
    }

    else if (process1 == 0)
    {
        sleep(3);
        printf("Child1: PID = %d, PGID = %d, PPID = %d\n",
        getpid(), getpgrp(), getppid());

        exit(0);
    }

    int process2;
    if ((process2 = fork()) == -1)
    {
        perror("Can\'t fork.\n");
        exit(1);
    }
    else if (process2 == 0)
    {
        sleep(3);
        printf("Child2: PID = %d, PGID = %d, PPID = %d\n",
        getpid(), getpgrp(), getppid());

        exit(0);
    }
```

```
37      else
38      {
39      printf("Parent: PID = %d, PGID = %d, CHILD1 = %d,
            CHILD2 = %d\n",
40          getpid(), getpgrp(), process1, process2);
41      }
42      return 0;
43  }
```

На рисунке 1 приведен результат работы программы. Процессы-потомки, которые продолжают выполняться после завершения процесса-предка, получают идентификатор предка, равный 1.

```
MBP-Anastasia:lab4 anastasia$ ./f1
Parent: PID = 84192, PGID = 84192, CHILD1 = 84193, CHILD2 = 84194
MBP-Anastasia:lab4 anastasia$ Child1: PID = 84193, PGID = 84192, PPID = 1
Child2: PID = 84194, PGID = 84192, PPID = 1

MBP-Anastasia:lab4 anastasia$
```

Рис 1.: Результат работы программы

На листинге 2 представлена вторая программа, демонстрирующая работу системного вызова wait().

Листинг 2: Задание 2

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <sys/wait.h>
5
6  int main()
7  {
8      int process1, process2;
9      int stat_val;
10     pid_t child_pid;
11
12     if ((process1 = fork()) == -1)
13     {
14         perror("Can\'t fork.\n");
15         exit(1);
16     }
```

```
17
18      else if (process1 == 0)
19      {
20          printf( "Child1: PID = %d, PGID = %d, PPID = %d\n",
21          getpid(), getpgrp(), getppid());
22
23          exit(0);
24      }
25
26      if ((process2 = fork()) == -1)
27      {
28          perror("Can\'t fork.\n");
29          exit(1);
30      }
31      else if (process2 == 0)
32      {
33          printf( "Child2: PID = %d, PGID = %d, PPID = %d\n",
34          getpid(), getpgrp(), getppid());
35
36          exit(0);
37      }
38      else
39      {
40          child_pid = wait(&stat_val);
41
42          if (WIFEXITED(stat_val))
43              printf("Child1 (PID = %d) has terminated
                    normally with code %d\n", child_pid,
                    WEXITSTATUS(stat_val));
44
45          if (WEXITSTATUS(stat_val))
46              printf("Child1 (PID = %d) has terminated due to
                     the receipt of a signal %d\n that was not
                    caught", child_pid, WTERMSIG(stat_val));
47
48          if (WIFSTOPPED(stat_val))
49              printf("Child1 (PID = %d) is currently stopped
                    due to the receipt of a signal %d\n",
                    child_pid, WSTOPSIG(stat_val));
50
```

```
51          child_pid = wait(&stat_val);
52
53          if (WIFEXITED(stat_val))
54              printf("Child2 (PID = %d) has terminated
                    normally with code %d\n", child_pid,
                    WEXITSTATUS(stat_val));
55
56          if (WEXITSTATUS(stat_val))
57              printf("Child2 (PID = %d) has terminated due to
                     the receipt of a signal %d\n that was not
                    caught", child_pid, WTERMSIG(stat_val));
58
59          if (WIFSTOPPED(stat_val))
60          printf("Child2 (PID = %d) is currently stopped due
                to the receipt of a signal %d\n", child_pid,
                WSTOPSIG(stat_val));
61
62          printf("Parent: PID = %d, PGID = %d, CHILD1 = %d,
                CHILD2 = %d\n",
63          getpid(), getpgrp(), process1, process2);
64      }
65      return 0;
66  }
```

На рисунке 2 приведен результат работы программы. Идентификаторы процессов-потомков больше идентификатора процесса-предка на 1 и 2, соответственно.

```
MBP-Anastasia:lab4 anastasia$ ./f2
Child1: PID = 84363, PGID = 84362, PPID = 84362
Child2: PID = 84364, PGID = 84362, PPID = 84362
Child1 (PID = 84363) has terminated normally with code 0
Child2 (PID = 84364) has terminated normally with code 0
Parent: PID = 84362, PGID = 84362, CHILD1 = 84363, CHILD2 = 84364
MBP-Anastasia:lab4 anastasia$
```

Рис 2.: Результат работы программы

На листинге 3 представлена третья программа, демонстрирующая работу системного вызова exec(). Был использован суффикс l, так как аргументы командной строки передаются в форме списка и их количество известно. Системному вызову execl() в качестве параметров переданы

имя исполняемого файла (print) и параметры, которые будут переданы этой программе. Исполняемый файл print находится в рабочем каталоге.

Листинг 3: Задание 3

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

int main()
{
    int process1, process2;
    int stat_val;
    pid_t child_pid;

    if ((process1 = fork()) == -1)
    {
        perror("Can\'t fork.\n");
        exit(1);
    }

    else if (process1 == 0)
    {
        printf( "Child1: PID = %d, PGID = %d, PPID = %d\n",
        getpid(), getpgrp(), getppid());

        if (execl("print", "Hello", NULL) == -1)
        {
            perror("Can\'t exec.\n");
            exit(1);
        }

        exit(0);
    }

    if ((process2 = fork()) == -1)
    {
        perror("Can\'t fork.\n");
        exit(1);
    }
```

```
37      else if (process2 == 0)
38      {
39          printf("Child2: PID = %d, PGID = %d, PPID = %d\n",
40          getpid(), getpgrp(), getppid());
41
42          if (execl("print", "BMSTU", "IU7!", NULL) == -1)
43          {
44              perror("Can\'t exec.\n");
45              exit(1);
46          }
47
48          exit(0);
49      }
50      else
51      {
52          child_pid = wait(&stat_val);
53
54          if (WIFEXITED(stat_val))
55              printf("\nChild1 (PID = %d) has terminated
                    normally with code %d\n", child_pid,
                    WEXITSTATUS(stat_val));
56
57          if (WEXITSTATUS(stat_val))
58              printf("\nChild1 (PID = %d) has terminated due
                    to the receipt of a signal %d\n that was not
                     caught", child_pid, WTERMSIG(stat_val));
59
60          if (WIFSTOPPED(stat_val))
61              printf("\nChild1 (PID = %d) is currently
                    stopped due to the receipt of a signal %d\n"
                    , child_pid, WSTOPSIG(stat_val));
62
63          child_pid = wait(&stat_val);
64
65          if (WIFEXITED(stat_val))
66              printf("\nChild2 (PID = %d) has terminated
                    normally with code %d\n", child_pid,
                    WEXITSTATUS(stat_val));
67
68          if (WEXITSTATUS(stat_val))
```

```
69              printf("\nChild2 (PID = %d) has terminated due
                   to the receipt of a signal %d\n that was not
                    caught", child_pid, WTERMSIG(stat_val));
70
71          if (WIFSTOPPED(stat_val))
72          printf("\nChild2 (PID = %d) is currently stopped
               due to the receipt of a signal %d\n", child_pid,
                WSTOPSIG(stat_val));
73
74          printf("Parent: PID = %d, PGID = %d, CHILD1 = %d,
               CHILD2 = %d\n",
75          getpid(), getpgrp(), process1, process2);
76      }
77      return 0;
78  }
```

На листинге 4 представлена программа, которая выводит на экран строку, переданную ей в качестве аргумента.

Листинг 4: Программа print.c

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[])
4  {
5      int i = 0;
6
7      while (argv[i] != NULL)
8      {
9          printf("%s ", argv[i]);
10         i++;
11     }
12     return 0;
13 }
```

На рисунке 3 приведен результат работы программы.

```
MBP-Anastasia:lab4 anastasia$ ./f3
Child1: PID = 84539, PGID = 84538, PPID = 84538
Child2: PID = 84540, PGID = 84538, PPID = 84538
Hello
Child1 (PID = 84539) has terminated normally with code 0
BMSTU IU7!
Child2 (PID = 84540) has terminated normally with code 0
Parent: PID = 84538, PGID = 84538, CHILD1 = 84539, CHILD2 = 84540
MBP-Anastasia:lab4 anastasia$
```

Рис 3.: Результат работы программы

На листинге 5 представлена программа, демонстрирующая обмен сообщением между предком и потомком через программный канал.

Листинг 5: Задание 4

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>
#define MAXLEN 256

int main()
{
    int process1, process2;
    int stat_val;
    pid_t child_pid;

    char str1[] = "Here should be the first text";
    char str2[] = "Here should be the second text";

    int fd[2];
    if (pipe(fd) == -1)
    {
        perror("Can\'t pipe.\n");
        exit(1);
    }

    if ((process1 = fork()) == -1)
    {
        perror("Can\'t fork.\n");
        exit(1);
```

```
28      }
29
30      else if (process1 == 0)
31      {
32           /* Child process closes up input side of pipe */
33          close(fd[0]);
34          write(fd[1], str1, (strlen(str1) + 1));
35
36          exit(0);
37      }
38
39      if ((process2 = fork()) == -1)
40      {
41          perror("Can\'t fork.\n");
42          exit(1);
43      }
44      else if (process2 == 0)
45      {
46
47           /* Child process closes up input side of pipe */
48          close(fd[0]);
49          write(fd[1], str2, (strlen(str2) + 1));
50
51          exit(0);
52      }
53      else
54      {
55          /* Parent process closes up output side of pipe */
56          close(fd[1]);
57
58          char str1[MAXLEN];
59          /* Read in a string from the pipe */
60          read(fd[0], str1, MAXLEN);
61
62          char str2[MAXLEN];
63          /* Read in a string from the pipe */
64          read(fd[0], str2, MAXLEN);
65
66          printf("String1 = %s, String2 = %s", str1, str2);
67
```

```
68          child_pid = wait(&stat_val);
69
70          if (WIFEXITED(stat_val))
71              printf("\n\nChild1 (PID = %d) has terminated
                    normally with code %d", child_pid,
                    WEXITSTATUS(stat_val));
72
73          if (WEXITSTATUS(stat_val))
74              printf("\nChild1 (PID = %d) has terminated due
                    to the receipt of a signal %d that was not
                    caught", child_pid, WTERMSIG(stat_val));
75
76          if (WIFSTOPPED(stat_val))
77              printf("\nChild1 (PID = %d) is currently
                    stopped due to the receipt of a signal %d",
                    child_pid, WSTOPSIG(stat_val));
78
79          child_pid = wait(&stat_val);
80
81          if (WIFEXITED(stat_val))
82              printf("\nChild2 (PID = %d) has terminated
                    normally with code %d", child_pid,
                    WEXITSTATUS(stat_val));
83
84          if (WEXITSTATUS(stat_val))
85              printf("\nChild2 (PID = %d) has terminated due
                    to the receipt of a signal %d that was not
                    caught", child_pid, WTERMSIG(stat_val));
86
87          if (WIFSTOPPED(stat_val))
88          printf("\nChild2 (PID = %d) is currently stopped
                due to the receipt of a signal %d", child_pid,
                WSTOPSIG(stat_val));
89
90          printf("\nParent: PID = %d, PGID = %d, CHILD1 = %d,
                CHILD2 = %d\n",
91          getpid(), getpgrp(), process1, process2);
92      }
93      return 0;
94  }
```

На рисунке 4 приведен результат работы программы.

```
MBP-Anastasia:lab4 anastasia$ ./f4
String1 = Here should be the first text, String2 = Here should be the second text

Child1 (PID = 84755) has terminated normally with code 0
Child2 (PID = 84756) has terminated normally with code 0
Parent: PID = 84754, PGID = 84754, CHILD1 = 84755, CHILD2 = 84756
MBP-Anastasia:lab4 anastasia$
```

Рис 4.: Результат работы программы

На листинге 6 представлена программа, демонстрирующая работу системного вызова signal(). Установлена реакция на поступление сигнала SIGINT. При этом ход выполнения программы меняется: при нажатии CTRL-C процессы-потомки записывают сообщение в программный канал, иначе - завершение программы.

Листинг 6: Задание 5

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>
#include <signal.h>
#define MAXLEN 256

int sig_receipt = 0;

void catch_sig(int sig_numb)
{
    signal(sig_numb, catch_sig);
    printf("\nCTRL-C pressed, caught signal = %d\n",
       sig_numb);
    sig_receipt = 1;
}

int main()
{
    int process1, process2;
    int stat_val;
```

```
22      pid_t child_pid;
23
24      char str1[] = "Here should be the first text";
25      char str2[] = "Here should be the second text";
26
27      signal(SIGINT, catch_sig);
28
29      printf("Press CTRL-C if you want to write messages\n");
30      sleep(5);
31
32      if (!sig_receipt)
33      {
34          printf("Exiting...\n");
35          sleep(2);
36          return 0;
37      }
38
39      int fd[2];
40      if (pipe(fd) == -1)
41      {
42          perror("Can\'t pipe.\n");
43          exit(1);
44      }
45
46      if ((process1 = fork()) == -1)
47      {
48          perror("Can\'t fork.\n");
49          exit(1);
50      }
51
52      else if (process1 == 0)
53      {
54           /* Child process closes up input side of pipe */
55          close(fd[0]);
56          write(fd[1], str1, (strlen(str1) + 1));
57
58          exit(0);
59      }
60
61      if ((process2 = fork()) == -1)
```

```
62      {
63          perror("Can\'t fork.\n");
64          exit(1);
65      }
66      else if (process2 == 0)
67      {
68
69           /* Child process closes up input side of pipe */
70          close(fd[0]);
71          write(fd[1], str2, (strlen(str2) + 1));
72
73          exit(0);
74      }
75      else
76      {
77          /* Parent process closes up output side of pipe */
78          close(fd[1]);
79
80          char str1[MAXLEN];
81          /* Read in a string from the pipe */
82          read(fd[0], str1, MAXLEN);
83
84          char str2[MAXLEN];
85          /* Read in a string from the pipe */
86          read(fd[0], str2, MAXLEN);
87
88          printf("String1 = %s, String2 = %s", str1, str2);
89
90          child_pid = wait(&stat_val);
91
92          if (WIFEXITED(stat_val))
93              printf("\n\nChild1 (PID = %d) has terminated
                    normally with code %d", child_pid,
                    WEXITSTATUS(stat_val));
94
95          if (WEXITSTATUS(stat_val))
96              printf("\nChild1 (PID = %d) has terminated due
                    to the receipt of a signal %d that was not
                    caught", child_pid, WTERMSIG(stat_val));
97
```

```
98          if (WIFSTOPPED(stat_val))
99              printf("\nChild1 (PID = %d) is currently
                    stopped due to the receipt of a signal %d",
                    child_pid, WSTOPSIG(stat_val));
100
101         child_pid = wait(&stat_val);
102
103         if (WIFEXITED(stat_val))
104             printf("\nChild2 (PID = %d) has terminated
                    normally with code %d", child_pid,
                    WEXITSTATUS(stat_val));
105
106         if (WEXITSTATUS(stat_val))
107             printf("\nChild2 (PID = %d) has terminated due
                    to the receipt of a signal %d that was not
                    caught", child_pid, WTERMSIG(stat_val));
108
109         if (WIFSTOPPED(stat_val))
110         printf("\nChild2 (PID = %d) is currently stopped
                due to the receipt of a signal %d", child_pid,
                WSTOPSIG(stat_val));
111
112         printf("\nParent: PID = %d, PGID = %d, CHILD1 = %d,
                CHILD2 = %d\n",
113         getpid(), getpgrp(), process1, process2);
114     }
115     return 0;
116 }
```

На рисунке 5 приведен результат работы программы, в случае если пользователь не нажал CTRL-C.

```
MacBook-Pro-Anastasia:lab4 anastasia$ ./f5
Press CTRL-C if you want to write messages
Exiting...
MacBook-Pro-Anastasia:lab4 anastasia$
```

Рис 5.: Результат работы программы

На рисунке 6 приведен результат работы программы, в случае если пользователь нажал CTRL-C.

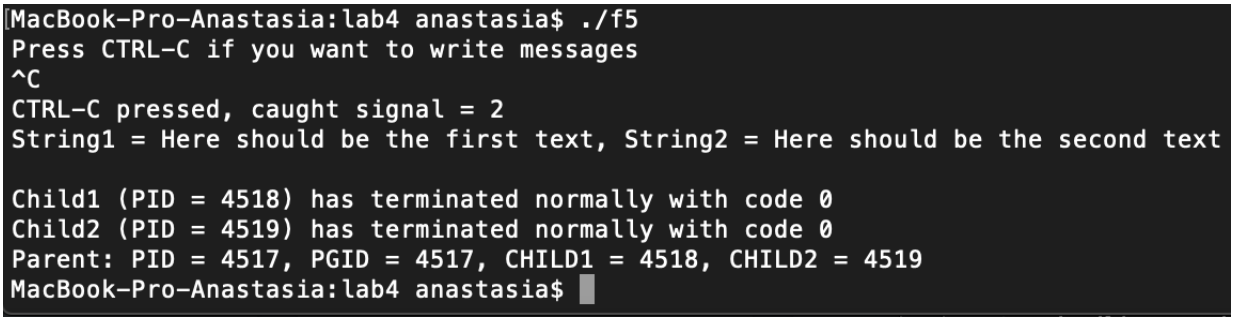

Рис 6.: Результат работы программы