



Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего
образования
«Московский государственный технический
университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4

По дисциплине: **Операционные Системы**

Тема: Процессы. Системные вызовы fork() и exec().

Студент Чаушев А.К..

Группа ИУ7-56Б

Оценка (баллы) _____

Преподаватели Рязанова Н. Ю.

Москва — 2020 г.

Задание 1

Написать программу, запускающую новый процесс системным вызовом `fork()`. В предке вывести собственный идентификатор (функция `getpid()`), идентификатор группы (функция `getpgrp()`) и идентификатор потомка. В процессе-потомке вывести собственный идентификатор, идентификатор предка (функция `getppid()`) и идентификатор группы. Убедиться, что при завершении процесса-предка потомок, который продолжает выполняться, получает идентификатор предка (PPID), равный 1 или идентификатор процесса-посредника.

```
1  int main() {
2      pid_t child1_pid, child2_pid;
3
4      if ((child1_pid = fork()) == -1) {
5          puts("Can't create new proc");
6          exit(1);
7      } else if (child1_pid == 0) {
8          sleep(1);
9          printf("CHILD1");
10         printf("pid%i_group_id%i_parent_id%i\n",
11             getpid(), getpgrp(), getppid());
12         exit(0);
13     }
14
15     if ((child2_pid = fork()) == -1) {
16         puts("Can't create new proc");
17         exit(1);
18     } else if (child2_pid == 0) {
19         sleep(1);
20         printf("CHILD2");
21         printf("pid%i_group_id%i_parent_id%i\n",
22             getpid(), getpgrp(), getppid());
23         exit(0);
24     }
25
26     //sleep(1);
27     printf("PARENT");
28     printf("pid%i_group_id%i\n", \
29         getpid(), getpgrp());
30     printf("child1_id%i_child2_id%i\n",
31         child1_pid, child2_pid);
32
33     return 0;
34 }
```

```
sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04> ./ex01
:::PARENT:: pid 21996 group id 21996 child 1 id 21997 child 2 id 21998
sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04>
:::CHILD1::: pid 21997 group id 21996 parent id 2589
:::CHILD2::: pid 21998 group id 21996 parent id 2589
```

Рис. 1: Результат работы

Задание 2

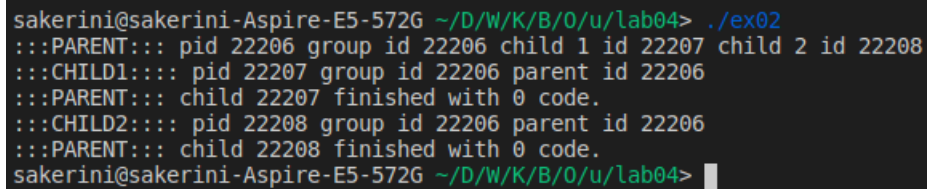
Написать программу по схеме первого задания, но в процессе-предке выполнить системный вызов `wait()`. Убедиться, что в этом случае идентификатор процесса потомка на 1 больше идентификатора процесса-предка.

```
1 void report_proc();
2
3 int main() {
4     pid_t child_pid1, child_pid2;
5
6     if ((child_pid1 = fork()) == -1) {
7         puts("Can't create new proc");
8         exit(1);
9     } else if (child_pid1 == 0) {
10        //sleep(1);
11        printf(":::CHILD1:::");
12        printf("pid%i_groupid%i_parentid%i\n",
13            getpid(), getpgrp(), getppid());
14        // exit(12);
15    } else {
16        if ((child_pid2 = fork()) == -1) {
17            puts("Can't create new proc");
18            exit(1);
19        } else if (child_pid2 == 0) {
20            //sleep(1);
21            printf(":::CHILD2:::pid%i_groupid%i_parentid%i\n", getpid(),
22                // exit(12);
23            } else {
24                printf("PARENT");
25                printf("pid%i_groupid%i\n", \
26                    getpid(), getpgrp());
27                printf("child1id%i_child2id%i\n",
28                    child_pid1, child_pid2);
29
30                report_proc();
31                report_proc();
32            }
33        }
34
35        return 0;
36    }
```

```

1
2 void report_proc() {
3     int status = 0;
4     pid_t wait_rc = wait(&status);
5
6     if (WIFEXITED(status)) {
7         printf(":::PARENT:::");
8         printf("child_%i_finished_with_%i_code.\n",
9             wait_rc, WEXITSTATUS(status));
10    } else if (WIFSIGNALED(status)) {
11        printf(":::PARENT:::");
12        printf("child_%i_finished_from_signal_with_%i_code.\n",
13            wait_rc, WTERMSIG(status));
14    } else if (WIFSTOPPED(status)) {
15        printf(":::PARENT:::");
16        printf("child_%i_finished_from_signal_with_%i_code.\n",
17            wait_rc, WSTOPSIG(status));
18    }
19 }

```



```

sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04> ./ex02
:::PARENT::: pid 22206 group id 22206 child 1 id 22207 child 2 id 22208
:::CHILD1::: pid 22207 group id 22206 parent id 22206
:::PARENT::: child 22207 finished with 0 code.
:::CHILD2::: pid 22208 group id 22206 parent id 22206
:::PARENT::: child 22208 finished with 0 code.
sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04> 

```

Рис. 2: Результат работы

Задание 3

Написать программу, в которой процесс-потомок вызывает системный вызов `exec()`, а процесс-предок ждет завершения процесса-потомка.

```
1 void report_proc();
2 int main() {
3     pid_t child1_pid;
4     if ((child1_pid = fork()) == -1) {
5         puts("Can't create child 1");
6         exit(1);
7     } else if (child1_pid == 0) {
8         // child 1
9         sleep(1);
10        printf(":::CHILD1:::");
11        printf("pid%i group id%i parent id%i\n",
12              getpid(), getpgrp(), getppid());
13        if (execlp("/bin/ls", "ls", "-l", NULL) == -1) {
14            puts(":::CHILD1::: Can not exec");
15        }
16    } else {
17        int child2_pid = fork();
18        if (child2_pid == -1) {
19            puts("Can't create child 2");
20            exit(1);
21        } else if (child2_pid == 0) {
22            // child 2
23            printf(":::CHILD2:::");
24            printf("pid%i group id%i parent id%i\n",
25                  getpid(), getpgrp(), getppid());
26            if (execlp("ps", "ps", "al", NULL) == -1) {
27                puts(":::CHILD2::: Can not exec");
28            }
29        } else {
30            // parent
31            printf(":::PARENT:::");
32            printf("pid%i group id%i parent id%i\n", \
33                  getpid(), getpgrp(), getppid());
34            printf("child1 id%i child2 id%i\n",
35                  child1_pid, child2_pid);
36            report_proc();
37            report_proc();
38        }
39    }
40    return 0;
41 }
```

```

1 void report_proc() {
2     int status = 0;
3     pid_t wait_rc = wait(&status);
4
5     if (WIFEXITED(status)) {
6         printf(":::PARENT:::");
7         printf("child_%i_finished_with_%i_code.\n",
8             wait_rc, WEXITSTATUS(status));
9     } else if (WIFSIGNALED(status)) {
10        printf(":::PARENT:::");
11        printf("child_%i_finished_from_signal_with_%i_code.\n",
12            wait_rc, WTERMSIG(status));
13    } else if (WIFSTOPPED(status)) {
14        printf(":::PARENT:::");
15        printf("child_%i_finished_from_signal_with_%i_code.\n",
16            wait_rc, WSTOPSIG(status));
17    }
18 }

```

```

sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04> ./ex03
:::PARENT::: pid 22960 group id 22960 parent id 21849 child 1 id 22961 child 2 id 22962
:::CHILD2::: pid 22962 group id 22960 parent id 22960
F  UID  PID  PPID PRI  NI   VSZ  RSS  WCHAN  STAT TTY      TIME COMMAND
4  124  1504  1238  20   0 163876 5376 -      Ssl+ tty1      0:00 /usr/lib/gdm3/gdm-x-session gnome-session --autostar
4   0  1508  1504  20   0 245416 31464 -      Sl+  tty1      0:02 /usr/lib/xorg/Xorg vt1 -displayfd 3 -auth /run/user/
0  124  2234  1504  20   0 263540 10940 -      Sl+  tty1      0:00 /usr/lib/gnome-session/gnome-session-binary --system
4 1000  2617  2567  20   0 163876 5440 poll s Ssl+ tty2      0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_S
4   0  2619  2617  20   0 279600 75976 -      Sl+  tty2      9:24 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/
0 1000  2672  2617  20   0 189516 10652 poll s Sl+  tty2      0:00 /usr/lib/gnome-session/gnome-session-binary --system
0 1000 21131 16443 20   0 23520 7344 poll s Ss+ pts/1     0:00 /usr/bin/fish -i
0 1000 21849 21731 20   0 171632 8644 do wai Ss pts/0     0:00 /usr/bin/fish
0 1000 22386 22378 20   0 163356 8020 poll s Ss+ pts/2     0:00 fish
0 1000 22960 21849 20   0 2480 776 do wai S+ pts/0     0:00 ./ex03
1 1000 22961 22960 20   0 2348 80 hrTime S+ pts/0     0:00 ./ex03
4 1000 22962 22960 20   0 11372 3136 -      R+  pts/0     0:00 ps al
:::PARENT::: child 22962 finished with 0 code.
:::CHILD1::: pid 22961 group id 22960 parent id 22960
total 88
drwxr-xr-x 2 sakerini sakerini 4096 сен 8 15:35 docs
-rwxr-xr-x 1 sakerini sakerini 17000 дек 1 20:12 ex01
-rw-r--r-- 1 sakerini sakerini 1750 дек 1 20:12 ex01.c
-rwxr-xr-x 1 sakerini sakerini 17088 дек 1 20:11 ex02
-rw-r--r-- 1 sakerini sakerini 1899 сен 8 15:35 ex02.c
-rwxr-xr-x 1 sakerini sakerini 17176 дек 1 20:22 ex03
-rw-r--r-- 1 sakerini sakerini 2010 сен 8 15:35 ex03.c
-rw-r--r-- 1 sakerini sakerini 2329 сен 8 15:35 ex04.c
-rw-r--r-- 1 sakerini sakerini 2785 сен 8 15:35 ex05.c
-rw-r--r-- 1 sakerini sakerini 238 сен 8 15:35 Makefile
:::PARENT::: child 22961 finished with 0 code.
sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04>

```

Рис. 3: Результат работы

Задание 4

Написать программу, в которой предок и потомок обмениваются сообщением через программный канал.

```
1 #define MSG_SIZE 19
2 void report_proc();
3
4 int main() {
5     pid_t child1_pid = 0;
6     int pipe_descr[2];
7     if (pipe(pipe_descr) == -1) {
8         puts("Can't create pipe");
9         exit(1);
10    }
11
12    if ((child1_pid = fork()) == -1) {
13        puts("Can't create child 1");
14        exit(1);
15    } else if (child1_pid == 0) {
16        // child 1
17        printf(":::CHILD1:::pid%i group%i parent%i\n",
18            getpid(), getpgrp(), getppid());
19
20        close(pipe_descr[0]);
21        char msg[] = "Hello from child 1";
22        write(pipe_descr[1], msg, MSG_SIZE);
23        exit(0);
24    } else {
25        int child2_pid = fork();
26        if (child2_pid == -1) {
27            puts("Can't create child 2");
28            exit(1);
29        } else if (child2_pid == 0) {
30            // child 2
31            printf(":::CHILD2:::pid%i group%i parent%i\n",
32                getpid(), getpgrp(), getppid());
33
34            close(pipe_descr[0]);
35            char msg[] = "Hello from child 2";
36            write(pipe_descr[1], msg, MSG_SIZE);
37            exit(0);
38        } else {
39            // parent
40            close(pipe_descr[1]);
41            printf(":::PARENT:::");
```



```

42         printf("pid_%i_group_%i_parent_%i\n", \
43                getpid(), getpgrp(), getppid());
44         printf("child_1_%i_child_2_%i\n",
45                child1_pid, child2_pid);
46
47         report_proc();
48         report_proc();
49
50         char buff[MSG_SIZE];
51
52         while (read(pipe_descr[0], buff, MSG_SIZE) > 0) {
53             printf("%s\n", buff);
54         };
55     }
56 }
57
58     return 0;
59 }

1 void report_proc() {
2     int status = 0;
3     pid_t wait_rc = wait(&status);
4
5     if (WIFEXITED(status)) {
6         printf(":::PARENT:::");
7         printf("child_%i_finished_with_%i_code.\n",
8                wait_rc, WEXITSTATUS(status));
9     } else if (WIFSIGNALED(status)) {
10        printf(":::PARENT:::");
11        printf("child_%i_finished_from_signal_with_%i_code.\n",
12               wait_rc, WTERMSIG(status));
13    } else if (WIFSTOPPED(status)) {
14        printf(":::PARENT:::");
15        printf("child_%i_finished_from_signal_with_%i_code.\n",
16               wait_rc, WSTOPSIG(status));
17    }
18 }

```

```
sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04> ./ex04
:::PARENT:: pid 23300 group id 23300 parent id 21849 child 1 id 23301 child 2 id 23302
:::CHILD1:: pid 23301 group id 23300 parent id 23300
:::PARENT:: child 23301 finished with 0 code.
:::CHILD2:: pid 23302 group id 23300 parent id 23300
:::PARENT:: child 23302 finished with 0 code.
Hello from child 1
Hello from child 2
sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04> █
```

Рис. 4: Результат работы

Задание 5

В программу с программным каналом включить собственный обработчик сигнала. Использовать сигнал для изменения хода выполнения программы.

```
1 #define MSG_SIZE 19
2 void report_proc();
3 void sigint_handler(int signum);
4
5 int signal_catched = 0;
6
7 int main() {
8     signal(SIGINT, sigint_handler);
9     signal(SIGSTOP, SIG_DFL);
10
11
12     pid_t child1_pid = 0;
13     int pipe_descr[2];
14     if (pipe(pipe_descr) == -1) {
15         puts("Can't pipe");
16         exit(1);
17     }
18
19     if ((child1_pid = fork()) == -1) {
20         puts("Can't create child 1");
21         exit(1);
22     } else if (child1_pid == 0) {
23         // child 1
24         printf(":::CHILD1:::pid%i_group%i_parent%i\n",
25             getpid(), getpgrp(), getppid());
26
27         close(pipe_descr[0]);
28         char msg[] = "Hello from child 1";
29         write(pipe_descr[1], msg, MSG_SIZE);
30         exit(0);
31     } else {
32         int child2_pid = fork();
33         if (child2_pid == -1) {
34             puts("Can't create child 2");
35             exit(1);
36         } else if (child2_pid == 0) {
37             // child 2
38             printf(":::CHILD2:::pid%i_group%i_parent%i\n",
39                 getpid(), getpgrp(), getppid());
40
41             close(pipe_descr[0]);
```

```

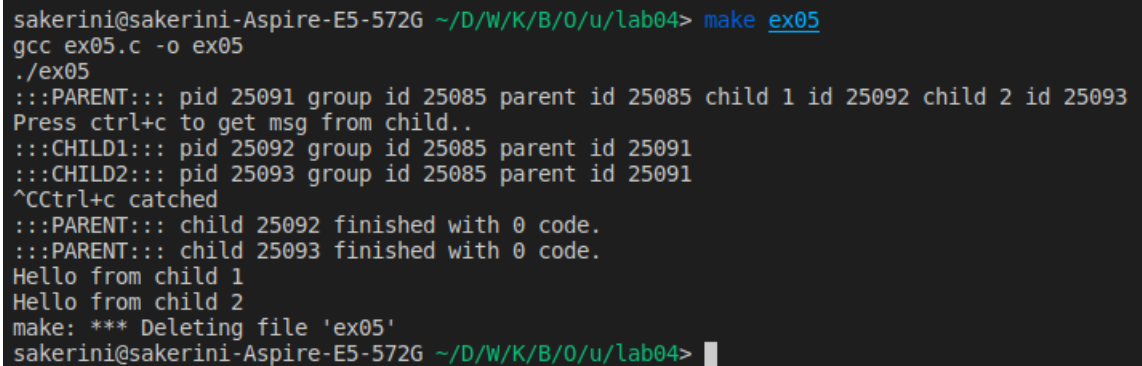
42         char msg[] = "Hello_from_child_2";
43         write(pipe_descr[1], msg, MSG_SIZE);
44         exit(0);
45     } else {
46         // parent
47         close(pipe_descr[1]);
48         printf(":::PARENT:::");
49         printf("pid%i_group_id%i_parent_id%i\n", \
50             getpid(), getpgrp(), getppid());
51         printf("child1_id%i_child2_id%i\n",
52             child1_pid, child2_pid);
53
54         puts("Press_ctrl+c_to_get_msg_from_child..");
55         sleep(5);
56
57         report_proc();
58         report_proc();
59
60         if (signal_catched) {
61             char buff[MSG_SIZE];
62
63             while (read(pipe_descr[0], buff, MSG_SIZE) > 0) {
64                 printf("%s\n", buff);
65             };
66         }
67     }
68 }
69
70 return 0;
71 }

```

```

1 void report_proc() {
2     int status = 0;
3     pid_t wait_rc = wait(&status);
4
5     if (WIFEXITED(status)) {
6         printf(":::PARENT:::");
7         printf("child_%i_finished_with_%i_code.\n",
8             wait_rc, WEXITSTATUS(status));
9     } else if (WIFSIGNALED(status)) {
10        printf(":::PARENT:::");
11        printf("child_%i_finished_from_signal_with_%i_code.\n",
12            wait_rc, WTERMSIG(status));
13    } else if (WIFSTOPPED(status)) {
14        printf(":::PARENT:::");
15        printf("child_%i_finished_from_signal_with_%i_code.\n",
16            wait_rc, WSTOPSIG(status));
17    }
18 }
19
20 void sigint_handler(int signum) {
21     puts("Ctrl+c caught");
22     signal_caught = 1;
23 }

```



```

sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04> make ex05
gcc ex05.c -o ex05
./ex05
:::PARENT::: pid 25091 group id 25085 parent id 25085 child 1 id 25092 child 2 id 25093
Press ctrl+c to get msg from child..
:::CHILD1::: pid 25092 group id 25085 parent id 25091
:::CHILD2::: pid 25093 group id 25085 parent id 25091
^C Ctrl+c caught
:::PARENT::: child 25092 finished with 0 code.
:::PARENT::: child 25093 finished with 0 code.
Hello from child 1
Hello from child 2
make: *** Deleting file 'ex05'
sakerini@sakerini-Aspire-E5-572G ~/D/W/K/B/O/u/lab04>

```

Рис. 5: Результат работы