|  |  |
| --- | --- |
|  | **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

**Лабораторная работа № 4**

|  |  |
| --- | --- |
| Тема: Процессы. Системные вызовы fork() и exec(). **Студент** ВоякинА. Я.  **Группа ИУ7-54Б**  **Преподаватель** Рязанова Н. Ю. |  |

*Москва, 2020 г.*

**Задание 1:** Написать программу, запускающую не мене двух новых процессов системным вызовом fork(). В предке вывести собственный идентификатор (функция getpid()), идентификатор группы ( функция getpgrp()) и идентификаторы потомков. 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.

**Листинг программы:**

#**include** <stdio.h>  
#**include** <stdlib.h>  
#**include** <unistd.h>  
  
**int** **main**()  
{  
 **pid\_t** child\_1, child\_2;  
  
 **if** ((child\_1 = fork()) == -1)  
 {  
 perror("Fork failed ");  
 exit(1);  
 }  
 **else** **if** (child\_1 == 0)  
 {  
 printf("\nПотомок child\_1:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 sleep(2);  
 printf("\nПотомок child\_1:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 exit(0);  
 }  
  
 **if** ((child\_2 = fork()) == -1)  
 {  
 perror("Fork failed ");  
 exit(1);  
 }  
 **else** **if** (child\_2 == 0)  
 {  
 printf("\nПотомок child\_2:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 sleep(2);  
 printf("\nПотомок child\_2:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 exit(0);  
 }  
 printf("\nПредок:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор группы = %d\  
 \nИдентификатор потомка child\_1 (pid) = %d\  
 \nИдентификатор потомка child\_2 (pid) = %d\n",  
 getpid(), getpgrp(), child\_1, child\_2);  
 **return** 0;  
}

**Результат работы программы:**

Предок:   
Собственный идентификатор (pid) = 7193   
Идентификатор группы = 7193   
Идентификатор потомка child\_1 = 7194   
Идентификатор потомка child\_2 = 7195  
  
Потомок child\_1:   
Собственный идентификатор (pid) = 7194   
Идентификатор предка (ppid) = 7193   
Идентификатор группы = 7193  
  
Потомок child\_2:   
Собственный идентификатор (pid) = 7195   
Идентификатор предка (ppid) = 7193   
Идентификатор группы = 7193  
  
Потомок child\_1:   
Собственный идентификатор (pid) = 7194   
Идентификатор предка (ppid) = 1   
Идентификатор группы = 7193  
  
Потомок child\_2:   
Собственный идентификатор (pid) = 7195   
Идентификатор предка (ppid) = 1   
Идентификатор группы = 7193

**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.

**Листинг программы:**

#**include** <stdio.h>  
#**include** <stdlib.h>  
#**include** <unistd.h>  
#**include** <sys/wait.h>  
  
**int** **main**()  
{  
 **pid\_t** child\_1, child\_2;  
 **int** status, val;  
  
 **if** ((child\_1 = fork()) == -1)  
 {  
 perror("Fork failed ");  
 exit(1);  
 }  
 **else** **if** (child\_1 == 0)  
 {  
 printf("\nПотомок child\_1:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 exit(0);  
 }  
  
 **if** ((child\_2 = fork()) == -1)  
 {  
 perror("Fork failed");  
 exit(1);  
 }  
 **else** **if** (child\_2 == 0)  
 {  
 printf("\nПотомок child\_2:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 exit(0);  
 }  
  
 **while** ((val = wait(&status)) != -1) {  
 **if** (WIFEXITED(status))  
 printf("\nДочерний процесс (%d) завершён нормально c кодом (%d).\n", val, WEXITSTATUS(status));  
 **else** **if** (WIFSIGNALED(status))  
 printf("\nДочерний процесс (%d) завершён неперехватываемым сигналом №(%d)\n", val, WTERMSIG(status));  
 **else** **if** (WIFSTOPPED(status))  
 printf("\nДочерний процесс (%d) остановился, номер сигнала: (%d)\n", val, WSTOPSIG(status));  
 }  
   
 printf("\nПредок:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор группы = %d\  
 \nИдентификатор потомка child\_1 (pid) = %d\  
 \nИдентификатор потомка child\_2 (pid) = %d\n",  
 getpid(), getpgrp(), child\_1, child\_2);  
 **return** 0;  
}

**Результат работы программы:**

Потомок child\_1:   
Собственный идентификатор (pid) = 8200   
Идентификатор предка (ppid) = 8199   
Идентификатор группы = 8199  
  
Потомок child\_2:   
Собственный идентификатор (pid) = 8201   
Идентификатор предка (ppid) = 8199   
Идентификатор группы = 8199  
  
Дочерний процесс (8200) завершён нормально c кодом (0).  
  
Дочерний процесс (8201) завершён нормально c кодом (0).  
  
Предок:   
Собственный идентификатор (pid) = 8199   
Идентификатор группы = 8199   
Идентификатор потомка child\_1 = 8200   
Идентификатор потомка child\_2 = 8201

**Задание 3:** написать программу, в которой процесс-потомок вызывает системный вызов exec(), а процесс-предок ждет завершения процесса-потомка. Следует создать не менее двух потомков.

**Листинг программы:**

#**include** <stdio.h>  
#**include** <stdlib.h>  
#**include** <unistd.h>  
  
**int** **main**()  
{  
 **pid\_t** child\_1, child\_2;  
 **int** status, val;  
  
 **if** ((child\_1 = fork()) == -1)  
 {  
 perror("Can't fork");  
 exit(1);  
 }  
 **else** **if** (child\_1 == 0)  
 {  
 printf("\nПотомок child\_1:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
   
 **if** (execl("/bin/ls", "ls", 0) == -1)  
 {  
 perror("exec failed");  
 exit(2);  
 }  
   
 }  
  
 **if** ((child\_2 = fork()) == -1)  
 {  
 perror("Can't fork");  
 exit(1);  
 }  
 **else** **if** (child\_2 == 0)  
 {  
 printf("\nПотомок child\_2:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
  
 **if** (execl("/bin/ps", "ps", 0) == -1)  
 {  
 perror("exec failed");  
 exit(3);  
 }  
 }  
  
 **while** ((val = wait(&status)) != -1) {  
 **if** (WIFEXITED(status))  
 printf("\nДочерний процесс (%d) завершён нормально c кодом (%d).\n", val, WEXITSTATUS(status));  
 **else** **if** (WIFSIGNALED(status))  
 printf("\nДочерний процесс (%d) завершён неперехватываемым сигналом №(%d)\n", val, WTERMSIG(status));  
 **else** **if** (WIFSTOPPED(status))  
 printf("\nДочерний процесс (%d) остановился, номер сигнала: (%d)\n", val, WSTOPSIG(status));  
 }  
   
 printf("\nПредок:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор группы = %d\  
 \nИдентификатор потомка child\_1 (pid) = %d\  
 \nИдентификатор потомка child\_2 (pid) = %d\n",  
 getpid(), getpgrp(), child\_1, child\_2);  
  
 **return** 0;  
}

**Результат работы программы:**

Потомок child\_1:   
Собственный идентификатор (pid) = 8529   
Идентификатор предка (ppid) = 8528   
Идентификатор группы = 8528  
  
Потомок child\_2:   
Собственный идентификатор (pid) = 8530   
Идентификатор предка (ppid) = 8528   
Идентификатор группы = 8528  
  
Lab03\_Voyakin.docx task\_1.c task\_3.c  
task\_3.out task\_2.c ~$b03\_Voyakin.docx  
  
Дочерний процесс (8529) завершён нормально c кодом (0).  
  
 PID TTY TIME CMD  
 1530 ttys000 0:02.76 -zsh  
 8528 ttys000 0:00.00 ./a.out  
 8529 ttys000 0:00.00 (ls)  
  
Дочерний процесс (8530) завершён нормально c кодом (0).  
  
Предок:   
Собственный идентификатор (pid) = 8528   
Идентификатор группы = 8528   
Идентификатор потомка child\_1 = 8529   
Идентификатор потомка child\_2 = 8530

**Задание 4:** написать программу, в которой предок и потомок обмениваются сообщением через программный канал.

**Листинг программы:**

#**include** <stdio.h>  
#**include** <stdlib.h>  
#**include** <unistd.h>  
#**include** <signal.h>  
#**include** <sys/wait.h>  
  
**int** **main**()  
{  
 **pid\_t** child\_1, child\_2;  
 **int** status, val, fd[2];  
  
 **if** (pipe(fd) == -1)  
 {  
 perror("Pipe failed");  
 exit(1);  
 }  
  
 **if** ((child\_1 = fork()) == -1)  
 {  
 perror("Fork failed");  
 exit(2);  
 }  
 **else** **if** (child\_1 == 0)  
 {  
 printf("\nПотомок child\_1:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 **char** message[] = "(Сообщение от child\_1)";  
 close(fd[0]);  
 write(fd[1], message, **sizeof** message - 1);  
 printf("\nПотомок child\_1 написал: %s\n", message);  
 exit(0);  
 }  
  
 **if** ((child\_2 = fork()) == -1)  
 {  
 perror("Fork failed");  
 exit(3);  
 }  
 **else** **if** (child\_2 == 0)  
 {  
 printf("\nПотомок child\_2:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 **char** message[] = "(Сообщение от child\_2)";  
 close(fd[0]);  
 write(fd[1], message, **sizeof** message - 1);  
 printf("\nПотомок child\_2 написал: %s\n", message);  
 exit(0);  
 }  
  
 **while** ((val = wait(&status)) != -1) {  
 **if** (WIFEXITED(status))  
 printf("\nДочерний процесс (%d) завершён нормально c кодом (%d).\n", val, WEXITSTATUS(status));  
 **else** **if** (WIFSIGNALED(status))  
 printf("\nДочерний процесс (%d) завершён неперехватываемым сигналом №(%d)\n", val, WTERMSIG(status));  
 **else** **if** (WIFSTOPPED(status))  
 printf("\nДочерний процесс (%d) остановился, номер сигнала: (%d)\n", val, WSTOPSIG(status));  
 }  
   
 printf("\nПредок:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор группы = %d\  
 \nИдентификатор потомка child\_1 (pid) = %d\  
 \nИдентификатор потомка child\_2 (pid) = %d\n",  
 getpid(), getpgrp(), child\_1, child\_2);  
   
 close(fd[1]);  
 **char** message[100];  
 read(fd[0], message, **sizeof** message);  
 printf("\nПредок прочитал: %s\n", message);  
  
 **return** 0;  
}

**Результат работы программы:**

Потомок child\_1:   
Собственный идентификатор (pid) = 8938   
Идентификатор предка (ppid) = 8937   
Идентификатор группы = 8937  
  
Потомок child\_1 написал: (Сообщение от child\_1)  
  
Потомок child\_2:   
Собственный идентификатор (pid) = 8939   
Идентификатор предка (ppid) = 8937   
Идентификатор группы = 8937  
  
Потомок child\_2 написал: (Сообщение от child\_2)  
  
Дочерний процесс (8938) завершён нормально c кодом (0).  
  
Дочерний процесс (8939) завершён нормально c кодом (0).  
  
Предок:   
Собственный идентификатор (pid) = 8937   
Идентификатор группы = 8937   
Идентификатор потомка child\_1 = 8938   
Идентификатор потомка child\_2 = 8939  
  
Предок прочитал: (Сообщение от child\_1)(Сообщение от child\_2)

**Задание 5:** в программу с программным каналом включить собственный обработчик сигнала. Использовать сигнал для изменения хода выполнения программы.

**Листинг программы:**

#**include** <stdio.h>  
#**include** <stdlib.h>  
#**include** <unistd.h>  
#**include** <signal.h>  
  
**int** child\_1\_write\_flag = 0;  
**int** child\_2\_write\_flag = 0;  
  
**void** **quit\_signal\_handler**(**int** signum)  
{  
 child\_1\_write\_flag = 1;  
 child\_2\_write\_flag = 1;  
}  
  
**int** **main**()  
{  
 **int** val, status, fd[2];  
 **pid\_t** child\_1, child\_2;  
  
 **if** (pipe(fd) == -1)  
 {  
 perror("Pipe failed");  
 exit(1);  
 }  
 signal(SIGQUIT, quit\_signal\_handler);  
 **if** ((child\_1 = fork()) == -1)  
 {  
 perror("Fork failed");  
 exit(2);  
 }  
 **else** **if** (child\_1 == 0)  
 {  
 printf("\nПотомок child\_1:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 signal(SIGQUIT, quit\_signal\_handler);  
 sleep(6);  
 **if** (child\_1\_write\_flag == 1)  
 {  
 **char** message[] = "(Сообщение от child\_1)";  
 close(fd[0]);  
 write(fd[1], message, **sizeof** message - 1);  
 printf("\nСигнал (Ctrl-\\) пришёл.");  
 printf("\nПотомок child\_1 написал: %s\n", message);  
 }  
 **else**  
 {  
 printf("\nСигнал (Ctrl-\\) не пришёл.\n");  
 }  
 exit(0);  
 }  
  
 **if** ((child\_2 = fork()) == -1)  
 {  
 perror("Fork failed");  
 exit(3);  
 }  
 **else** **if** (child\_2 == 0)  
 {  
 printf("\nПотомок child\_2:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор предка (ppid) = %d\  
 \nИдентификатор группы = %d\n",  
 getpid(), getppid(), getpgrp());  
 signal(SIGQUIT, quit\_signal\_handler);  
 sleep(6);  
 **if** (child\_2\_write\_flag == 1)  
 {  
 **char** message[] = "(Сообщение от child\_2)";  
 close(fd[0]);  
 write(fd[1], message, **sizeof** message - 1);  
 printf("\nСигнал (Ctrl-\\) пришёл.");  
 printf("\nПотомок child\_2 написал: %s\n", message);  
 }  
 **else**  
 {  
 printf("\nСигнал (Ctrl-\\) не пришёл.\n");  
 }  
 exit(0);  
 }  
  
 **while** ((val = wait(&status)) != -1) {  
 **if** (WIFEXITED(status))  
 printf("\nДочерний процесс (%d) завершён нормально c кодом (%d).\n", val, WEXITSTATUS(status));  
 **else** **if** (WIFSIGNALED(status))  
 printf("\nДочерний процесс (%d) завершён неперехватываемым сигналом №(%d)\n", val, WTERMSIG(status));  
 **else** **if** (WIFSTOPPED(status))  
 printf("\nДочерний процесс (%d) остановился, номер сигнала: (%d)\n", val, WSTOPSIG(status));  
 }  
   
 printf("\nПредок:\  
 \nСобственный идентификатор (pid) = %d\  
 \nИдентификатор группы = %d\  
 \nИдентификатор потомка child\_1 (pid) = %d\  
 \nИдентификатор потомка child\_2 (pid) = %d\n",  
 getpid(), getpgrp(), child\_1, child\_2);  
   
 close(fd[1]);  
 **char** message[100];  
 read(fd[0], message, **sizeof** message);  
 printf("\nПредок прочитал: %s\n", message);  
  
 **return** 0;  
}

**Результат работы программы, если сигнал был получен:**

Потомок child\_2:   
Собственный идентификатор (pid) = 3937   
Идентификатор предка (ppid) = 3935   
Идентификатор группы = 3935  
  
Потомок child\_1:   
Собственный идентификатор (pid) = 3936   
Идентификатор предка (ppid) = 3935   
Идентификатор группы = 3935

^\

Сигнал (Ctrl-\) пришёл.  
Потомок child\_1 написал: (Сообщение от child\_1)

Сигнал (Ctrl-\) пришёл.  
Потомок child\_2 написал: (Сообщение от child\_2)  
  
Дочерний процесс (3937) завершён нормально c кодом (0).  
  
Дочерний процесс (3936) завершён нормально c кодом (0).  
  
Предок:   
Собственный идентификатор (pid) = 3935   
Идентификатор группы = 3935   
Идентификатор потомка child\_1 (pid) = 3936   
Идентификатор потомка child\_2 (pid) = 3937  
  
Предок прочитал: (Сообщение от child\_1)(Сообщение от child\_2)

При получении сигнала (Ctrl-\) дочерние процессы обмениваются сообщениями с предком, пользователю выводятся соответствующие сообщения.

**Результат работы программы, если сигнал не был получен:**

Потомок child\_1:   
Собственный идентификатор (pid) = 3967   
Идентификатор предка (ppid) = 3966   
Идентификатор группы = 3966  
  
Потомок child\_2:   
Собственный идентификатор (pid) = 3968   
Идентификатор предка (ppid) = 3966   
Идентификатор группы = 3966  
  
  
Сигнал (Ctrl-\) не пришёл.  
Сигнал (Ctrl-\) не пришёл.  
  
Дочерний процесс (3968) завершён нормально c кодом (0).  
  
Дочерний процесс (3967) завершён нормально c кодом (0).  
  
Предок:   
Собственный идентификатор (pid) = 3966   
Идентификатор группы = 3966   
Идентификатор потомка child\_1 (pid) = 3967   
Идентификатор потомка child\_2 (pid) = 3968  
  
Предок прочитал:

При отсутствии сигнала дочерние процессы завершается без обмена сообщениями с предком и выводят сообщения пользователю об отсутствии сигнала.