МГТУ им. Баумана

Лабораторная работа №4

По курсу: "Операционные системы"

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

Работу выполнил: Мокеев Даниил, ИУ7-56

Преподаватель: Рязанова Н.Ю.

Москва, 2020

0.1 Листинг кода алгоритмов

В данном разделе будут приведены листинги кода реализованных программ.

Листинг 1: Процессы-сироты

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

int main(){
  int child_1, child_2;
  child_1 = fork();
  if(child_1 == -1){
    perror("Coulnd't fork child #1");
    exit(1);
  }
  if (child_1 == 0){
    sleep(1);
    printf("Child #1: pid=%d; group=%d; ppid=%d\n",
    getpid(), getpgrp(), getppid());

    return 0;
  }
  if (child_1 > 0){
    child_2 = fork();
    if(child_2 == -1){
      perror("Coulnd't fork child #2");
      exit(1);
    }
    if (child_2 == 0){
      printf("\nChild #2: pid=%d; group=%d; ppid=%d\n",
      getpid(), getpgrp(), getppid());

      return 0;
    }else{
      printf("Parent: pid=%d; group=%d; ppid=%d\n",
      getpid(), getpgrp(), getppid());

      return 0;
    }
  }
}
```

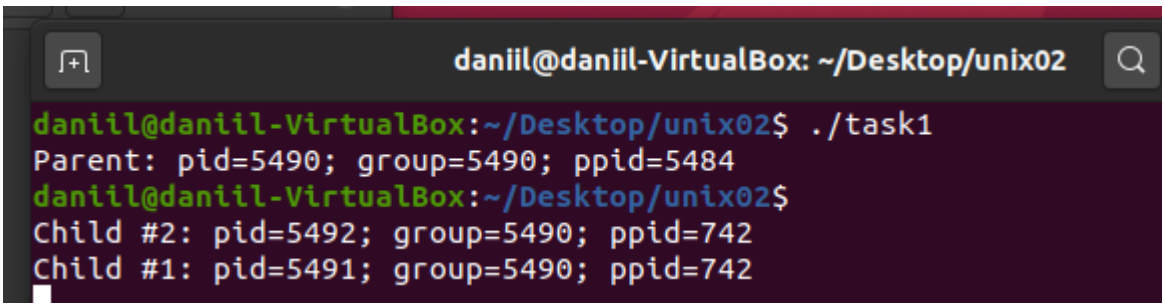

Рис. 1: Пример работы программы №1

Листинг 2: Использование вызова wait()

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

int main(){
  int child_1, child_2;
  child_1 = fork();
  if(child_1 == -1){
    perror("Coulnd't fork child #1");
    exit(1);
  }
  if (child_1 == 0){
    sleep(1);
    printf("Child #1: pid=%d; group=%d; ppid=%d\n",
    getpid(), getpgrp(), getppid());

    return 0;
  }
  if (child_1 > 0){
    child_2 = fork();
    if(child_2 == -1){
      perror("Coulnd't fork child #2");
      exit(1);
    }
    if (child_2 == 0){
      printf("\nChild #2: pid=%d; group=%d; ppid=%d\n",
      getpid(), getpgrp(), getppid());

      return 0;
    }else{
      sleep(2);
      printf("Parent: pid=%d; group=%d; ppid=%d\n",
      getpid(), getpgrp(), getppid());

      pid_t child_pid;
      int status;

      child_pid = wait(&status);
      if (WIFEXITED(status))
      printf("Parent: child %d finished with code %d\n",
      child_pid, WEXITSTATUS(status) );
      else if (WIFSTOPPED(status))
      printf("Parent: child %d finished with code %d\n",
      child_pid, WSTOPSIG(status) );
      return 0;
    }
  }
```

```
49
50      }
```

```
daniil@daniil-VirtualBox:~/Desktop/unix02$ ./task2

Child #2: pid=5891; group=5889; ppid=5889
Child #1: pid=5890; group=5889; ppid=5889
Parent: pid=5889; group=5889; ppid=5879
Parent: child 5890 finished with code 0
daniil@daniil-VirtualBox:~/Desktop/unix02$
```

Рис. 2: Пример работы программы №2

Листинг 3: Использование вызова exec()

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

int main(){
  int child_1, child_2;
  child_1 = fork();
  if(child_1 == -1){
    perror("Coulnd't fork child #1");
    exit(1);
  }
  if (child_1 == 0){
    sleep(1);
    printf("\nChild #1 executes ps -al\n\n");
    if(execlp("ps", "ps", "-al", (char*)NULL) == -1){
      printf("Couldn't exec ps command\n");
      exit(1);
    }

    return 0;
  }
  if (child_1 > 0){
    child_2 = fork();
    if(child_2 == -1){
      perror("Coulnd't fork child #2");
      exit(1);
    }
    if (child_2 == 0){
      printf("\nChild #2 executes ls -l\n\n");
      if(execlp("ls", "ls", "-l", (char*)NULL) == -1){
        printf("Couldn't exec ps command\n");
        exit(1);
      }
      return 0;
    }else{
      printf("\nParent: pid=%d; group=%d; ppid=%d\n",
      getpid(), getpgrp(), getppid());

      pid_t child_pid;
      int status;

      //waiting for the second child to finish
      child_pid = wait(&status);
      if (WIFEXITED(status))
      printf("\nParent: child with pid = %d finished with code %d\n"↩
          ,
      child_pid, WEXITSTATUS(status) );
      else if (WIFSTOPPED(status))
```

```
48            printf("\nParent: child %d finished with code %d\n",
49            child_pid, WSTOPSIG(status) );
50
51            //waiting for the first child to finish
52            child_pid = wait(&status);
53            if (WIFEXITED(status))
54            printf("\nParent: child with pid = %d finished with code %d\n"↩
                  ,
55            child_pid, WEXITSTATUS(status) );
56            else if (WIFSTOPPED(status))
57            printf("\nParent: child %d finished with code %d\n",
58            child_pid, WSTOPSIG(status) );
59
60            return 0;
61          }
62        }
63      }
```

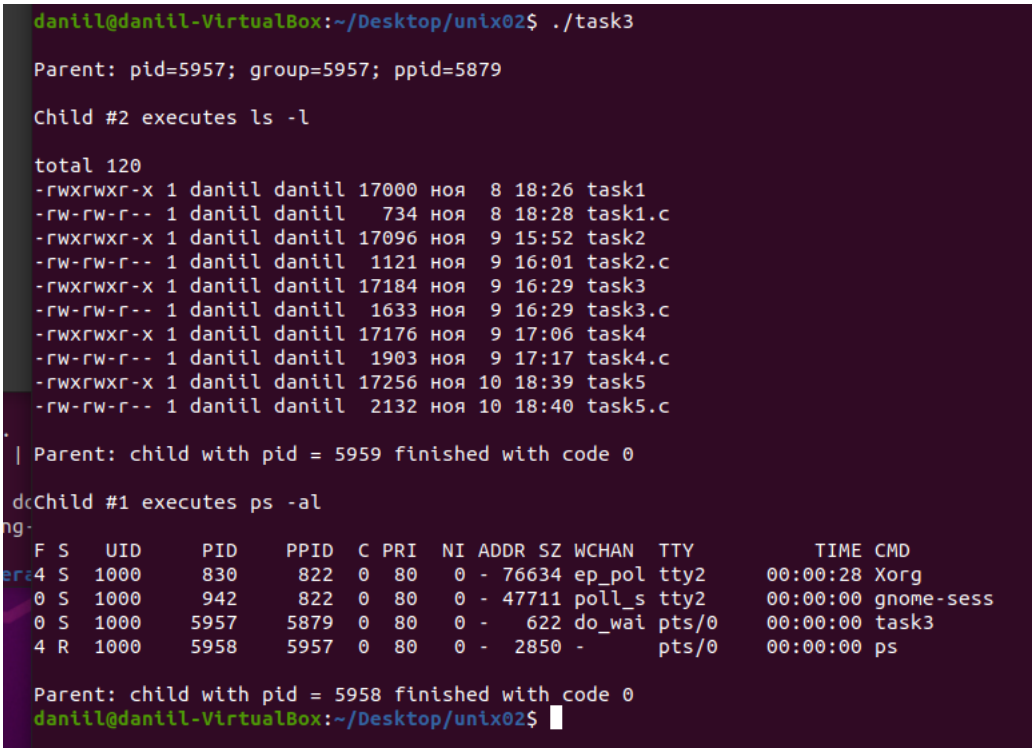

Рис. 3: Пример работы программы №3

Листинг 4: Передача сообщений с помощью программного канала

```
/*
exchanging messages with parent
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define MESSAGE_SIZE 32

int main(){
  int child_1, child_2;

  //initializing pipe
  int fd[2];
  if (pipe(fd) == -1){
    printf("Coundn't create a pipe\n");
    exit(1);
  }

  child_1 = fork();
  if(child_1 == -1){
    perror("Coulnd't fork child #1");
    exit(1);
  }
  if (child_1 == 0){


    close(fd[0]);
    if (write(fd[1], "Hello from child #1, parent!\n",
    MESSAGE_SIZE) > 0)
    printf("Child #1 sent a greeting to the parent\n");
    return 0;
  }
  if (child_1 > 0){
    child_2 = fork();
    if(child_2 == -1){
      perror("Coulnd't fork child #2");
      exit(1);
    }
    if (child_2 == 0){
      sleep(1);
      close(fd[0]);
      if (write(fd[1], "Hello from child #2, parent!\n",
      MESSAGE_SIZE) > 0)
      printf("Child #2 sent a greeting to the parent\n");
      return 0;
    }else{

      char msg1[MESSAGE_SIZE], msg2[MESSAGE_SIZE];
```

```
49            close(fd[1]);
50            read(fd[0], msg1, MESSAGE_SIZE);
51            read(fd[0], msg2, MESSAGE_SIZE);
52
53            printf("Parent read a message from his children: \n%s\n%s",
54            msg1, msg2);
55
56
57
58            pid_t child_pid;
59            int status;
60
61            //waiting for the second child to finish
62            child_pid = wait(&status);
63            if (WIFEXITED(status))
64            printf("\nParent: child with pid = %d finished with code %d\n"↩
                  ,
65            child_pid, WEXITSTATUS(status) );
66            else if (WIFSTOPPED(status))
67            printf("\nParent: child %d finished with code %d\n",
68            child_pid, WSTOPSIG(status) );
69
70            //waiting for the first child to finish
71            child_pid = wait(&status);
72            if (WIFEXITED(status))
73            printf("\nParent: child with pid = %d finished with code %d\n"↩
                  ,
74            child_pid, WEXITSTATUS(status) );
75            else if (WIFSTOPPED(status))
76            printf("\nParent: child %d finished with code %d\n",
77            child_pid, WSTOPSIG(status) );
78
79            return 0;
80          }
81        }
82      }
```

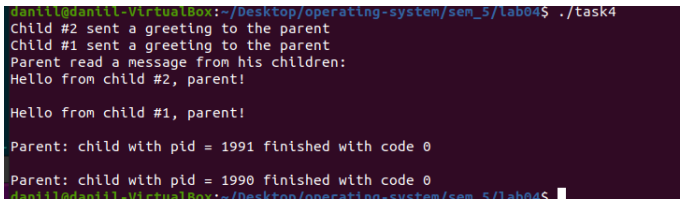

Рис. 4: Пример работы программы №4

Листинг 5: Установка своего обработчика сигнала

```
/*
DIY sig handler
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <stdbool.h>
#include <string.h>

#define MESSAGE_SIZE 64
#define SLEEP_TIME 3

//flag is set to true if signal has been cought
bool flag = false;

void mySignalHandler(int snum){
  printf("\nHandlig signal snum = %d in prosses...\n", snum);
  printf("Done!\n");
  flag = true;
}

int main(){
  int child_1, child_2;
  signal(SIGINT, mySignalHandler);

  //initializing pipe
  int fd[2];
  if (pipe(fd) == -1){
    printf("Coundn't create a pipe\n");
    exit(1);
  }

  child_1 = fork();
  if(child_1 == -1){
    perror("Coulnd't fork child #1");
    exit(1);
  }
  if (child_1 == 0){
    close(fd[1]);
    sleep(SLEEP_TIME);
    if (flag){
      char msg[MESSAGE_SIZE];
      if (read(fd[0], msg, MESSAGE_SIZE) > 0){
        printf("Child #1 read from parent %s\n", msg);
      }
    }

```

```
49          return 0;
50        }
51        if (child_1 > 0){
52          child_2 = fork();
53          if(child_2 == -1){
54            perror("Coulnd't fork child #2");
55            exit(1);
56          }
57          if (child_2 == 0){
58            close(fd[1]);
59
60            sleep(SLEEP_TIME);
61            if (flag){
62              char msg[MESSAGE_SIZE];
63              if (read(fd[0], msg, MESSAGE_SIZE) > 0){
64                printf("Child #2 read from parent %s\n", msg);
65              }
66            }
67
68            return 0;
69          }else{
70            close(fd[0]);
71            printf("Parent's waiting for Ctrl+C being pressed to send ↩
                 messages from children\n");
72            sleep(SLEEP_TIME);
73
74            if (flag){
75              //writing in pipe if we cought the signal
76              if (write(fd[1], "Hello, my child!\n", MESSAGE_SIZE) > 0)
77              printf("Parent sent his first greeting\n");
78              if (write(fd[1], "Hello again, my child!\n", MESSAGE_SIZE) >↩
                    0)
79              printf("Parent sent his second greeting\n");
80            }
81
82            pid_t child_pid;
83            int status;
84
85            //waiting for the second child to finish
86            child_pid = wait(&status);
87            if (WIFEXITED(status))
88            printf("Parent: child with pid = %d finished with code %d\n",
89            child_pid, WEXITSTATUS(status) );
90            else if (WIFSTOPPED(status))
91            printf("Parent: child %d finished with code %d\n",
92            child_pid, WSTOPSIG(status) );
93
94            //waiting for the first child to finish
95            child_pid = wait(&status);
```

```
96            if (WIFEXITED(status))
97            printf("Parent: child with pid = %d finished with code %d\n",
98            child_pid, WEXITSTATUS(status) );
99            else if (WIFSTOPPED(status))
100           printf("Parent: child %d finished with code %d\n",
101           child_pid, WSTOPSIG(status) );
102           return 0;
103         }
104       }
105   }
```

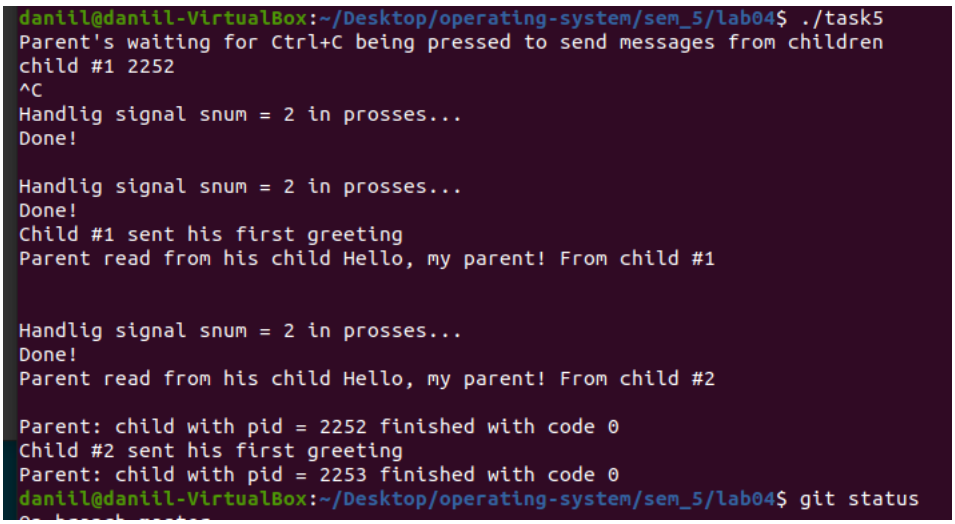

Рис. 5: Пример работы программы №5: был подан сигнал

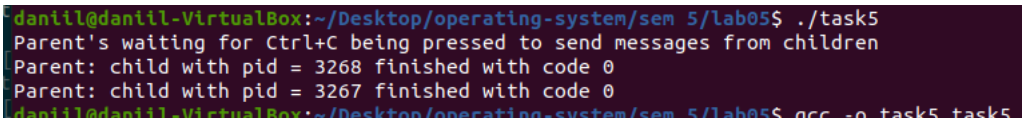

Рис. 6: Пример работы программы №5: сигнал не был подан