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» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

О Т Ч Е Т

по лабораторной работе № 4

Название: Защищенный режим

Дисциплина: Операционные системы

Студент ИУ7-53Б А.С.Саркисов

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель Н.Ю. Рязанова

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

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, кото- рый продолжает выполняться, получает идентификатор предка (PPID), равный 1 или идентификатор процесса-посредника.

Ниже приведён листинг программы для задания №1.

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



int main()
{
  int child_1 = fork();
  if (child_1 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (child_1)
  {
    printf("Parent:\t\tPID=%-5d\tGROUP=%-5d\tCHILD #1=%-5d\n", getpid(), getpgrp(), child_1);
  }
  else
  {
    printf("Child #1:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());
    sleep(1);
    printf("\nChild #1:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());
    return 0;
  }


  int child_2 = fork();
  if (child_2 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (child_2)
  {
    printf("Parent:\t\tPID=%-5d\tGROUP=%-5d\tCHILD #2=%-5d\n", getpid(), getpgrp(), child_2);
    return 0;
  }
  else
  {
    sleep(1);
    printf("Child #2:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d", getpid(), getpgrp(), getppid());
    return 0;
  }
}
```

Ниже приведён пример работы программы для задания №1.

```
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$ ./task_01.out
Parent:         PID=16560       GROUP=16560     CHILD #1=16561
Parent:         PID=16560       GROUP=16560     CHILD #2=16562
Child #1:       PID=16561       GROUP=16560     PPID=16560
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$
Child #1:       PID=16561       GROUP=16560     PPID=2934
Child #2:       PID=16562       GROUP=16560     PPID=2934
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$
```

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.

Ниже приведён листинг программы для задания №2.

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



int main()
{
  int child_1 = fork();
  if (child_1 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (!child_1)
  {
    printf("Child #1:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());
    return 0;
  }

  int child_2 = fork();
  if (child_2 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (!child_2)
  {
    printf("Child #2:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());
    return 0;
  }


  if (child_1 && child_2)
```

```
  {
    printf("Parent:\t\tPID=%-5d\tGROUP=%-5d\tCHILD #1=%-5d\n", getpid(), getpgrp(), child_1);
    printf("Parent:\t\tPID=%-5d\tGROUP=%-5d\tCHILD #2=%-5d\n", getpid(), getpgrp(), child_2);


    int status;
    int return_value;

    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("Parent:\tchild %d finished with code '%d'\n", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("Parent: child %d finished from signal with code '%d'\n", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("Parent:\tchild %d finished from signal with code '%d'\n", return_value, WTERMSIG(
          status));

    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("Parent:\tchild %d finished with code '%d'\n", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("Parent: child %d finished from signal with code '%d'\n", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("Parent:\tchild %d finished from signal with code '%d'\n", return_value, WTERMSIG(
          status));

    return 0;
  }
}
```

Ниже приведён пример работы программы для задания №2.

```
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$ ./task_02.out
Parent:         PID=17091       GROUP=17091     CHILD #1=17092
Parent:         PID=17091       GROUP=17091     CHILD #2=17093
Child #1:       PID=17092       GROUP=17091     PPID=17091
Parent: child 17092 finished with code '0'
Child #2:       PID=17093       GROUP=17091     PPID=17091
Parent: child 17093 finished with code '0'
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$
```

Задание №3

Написать программу, в которой процесс-потомок вызывает систем- ный вызов exec(), а процесс-предок ждет завершения процесса-потомка. Следует создать не менее двух потомков.

В своей программе с помощью системного вызова exec() я запускаю программы ls (ls -a) и ps (ps -al).

Ниже приведён листинг программы для задания №3.

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


int main()
{
  int child_1 = fork();
  if (child_1 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (!child_1)
  {
    printf("Child #1:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());
    if (execlp("ls", "ls", "-a", NULL) == -1)
    {
      printf("Child #1 couldn't exec()");
      exit(1);
    }
  }


  int child_2 = fork();
  if (child_2 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (!child_2)
```

```
  {
    sleep(1);
    printf("Child #2:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());
    if (execlp("ps", "ps", "-al", NULL) == -1)
    {
      printf("Child #2 couldn't exec()");
      exit(1);
    }
  }

  if (child_1 && child_2)
  {
    printf("Parent:\tPID=%-5d\tGROUP=%-5d\tCHILD #1=%-5d\tCHILD #2=%-5d\n\n", getpid(), getpgrp()
        , child_1, child_2);

    int status, return_value;

    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("Parent:\tchild %d finished with code '%d'\n\n", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("Parent: child %d finished from signal with code '%d'\n\n", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("Parent:\tchild %d finished from signal with code '%d'\n\n", return_value, WTERMSIG(
          status));

    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("Parent:\tchild %d finished with code '%d'\n\n", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("Parent: child %d finished from signal with code '%d'\n\n", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("Parent:\tchild %d finished from signal with code '%d'\n\n", return_value, WTERMSIG(
          status));

    return 0;
  }
}
```

Ниже приведён пример работы программы для задания №3.

```
sillyjoe@COMPUKTER:~/Документы/Осн/lab_04$ ./task_03.out
Parent: PID=17153       GROUP=17153     CHILD #1=17154  CHILD #2=17155
Child #1:       PID=17154       GROUP=17153     PPID=17153
.   report     task_01.out  task_02.out  task_03.out  task_04.out  task_05.out
..  task_01.c  task_02.c    task_03.c    task_04.c    task_05.c
Parent: child 17154 finished with code '0'

Child #2:       PID=17155       GROUP=17153     PPID=17153
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 T  1000 13636 13314  0  80   0 -  1127 signal pts/6    00:00:00 task_05.out
1 T  1000 13637 13636  0  80   0 -  1094 signal pts/6    00:00:00 task_05.out
1 Z  1000 13638 13636  0  80   0 -     0 -      pts/6    00:00:00 task <defunct>
0 S  1000 17153 16545  0  80   0 -  1126 wait   pts/7    00:00:00 task_03.out
4 R  1000 17155 17153  2  80   0 -  7541 -      pts/7    00:00:00 ps
Parent: child 17155 finished with code '0'
```

Задание №4

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

Ниже приведён листинг программы для задания №4.

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



int main()
{
  int fd_array[2];
  if (pipe(fd_array) == -1)
  {
    printf("Can't pipe");
    exit(1);
  }

  int child_1 = fork();
  if (child_1 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (!child_1)
  {
    close(fd_array[0]);
    write(fd_array[1], "Hi dad! I am your son!", 30);
    exit(0);
  }

  int child_2 = fork();
  if (child_2 == -1)
  {
    printf("Can't fork");
    exit(1);
  }
  else if (!child_2)
  {
    close(fd_array[0]);
    write(fd_array[1], "Hi dad! I am your daughter!", 30);
```

```
    exit(0);
  }

  if (child_1 && child_2)
  {
    close(fd_array[1]);
    char msg[30];

    read(fd_array[0], msg, 30);
    printf("Parent received:\t'%s'\n", msg);

    read(fd_array[0], msg, 30);
    printf("Parent received:\t'%s'\n", msg);

    int status, return_value;

    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("\nParent:\tchild %d finished with code '%d'", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("\nParent: child %d finished from signal with code '%d'", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("\nParent:\tchild %d finished from signal with code '%d'", return_value, WTERMSIG(
          status));

    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("\nParent:\tchild %d finished with code '%d'\n", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("\nParent: child %d finished from signal with code '%d'\n", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("\nParent:\tchild %d finished from signal with code '%d'\n", return_value, WTERMSIG(
          status));
  }
}
```

Ниже приведён пример работы программы для задания №4.

```
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$ ./task_04.out
Parent received:        'Hi dad! I am your son!'
Parent received:        'Hi dad! I am your daughter!'

Parent: child 17267 finished with code '0'
Parent: child 17268 finished with code '0'
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$
```

Задание №5

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

В своей программе я написал обработчики сигналов SIGINT (сиг- нал прерывания (Ctrl-C) с терминала) и SIGALRM (сигнал истечения времени, заданного alarm()).

Ниже приведён листинг программы для задания №5.

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



int signal_flag = 0;


// ctrl-c
void sigint_handler(int snum)
{
  printf("*** process %d caught signal #%d ***\n", getpid(), snum);
  signal_flag = SIGINT;
}

// alarm()
void sigalrm_handler(int snum)
{
  printf("*** process %d caught signal #%d ***\n", getpid(), snum);
  signal_flag = SIGALRM;
}

int main()
{
  int fd_array[2];
  if (pipe(fd_array) == -1)
  {
    printf("Can't pipe");
    exit(1);
```

```
}

int child_1 = fork();
if (child_1 == -1)
{
  printf("Can't fork");
  exit(1);
}
else if (!child_1)
{
  signal(SIGINT, sigint_handler);

  printf("Child #1:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());

  if (signal_flag == SIGINT)
  {
    close(fd_array[0]);
    write(fd_array[1], "Hi dad! Ctrl-C was pressed!", 30);
  }
  exit(0);
}

int child_2 = fork();
if (child_2 == -1)
{
  printf("Can't fork");
  exit(1);
}
else if (!child_2)
{
  signal(SIGALRM, sigalrm_handler);

  printf("Child #2:\tPID=%-5d\tGROUP=%-5d\tPPID=%-5d\n", getpid(), getpgrp(), getppid());

  alarm(5);

  if (signal_flag == SIGALRM)
  {
    close(fd_array[0]);
    write(fd_array[1], "Hi dad! Time is up!", 30);
  }
  exit(0);
}

if (child_1 && child_2)
{
  signal(SIGINT, sigint_handler);

  printf("NOTE: you have 5 seconds to press Ctrl-C\n");
  printf("Parent:\t\tPID=%-5d\tGROUP=%-5d\tCHILD #1=%-5d\n", getpid(), getpgrp(), child_1);
  printf("Parent:\t\tPID=%-5d\tGROUP=%-5d\tCHILD #2=%-5d\n", getpid(), getpgrp(), child_2);

  close(fd_array[1]);
  char msg[30];

  read(fd_array[0], msg, 30);
  printf("\nParent received:\t'%s'\n", msg);

  int status, return_value;
```

```
    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("\nParent:\tchild %d finished with code '%d'", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("\nParent: child %d finished from signal with code '%d'", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("\nParent:\tchild %d finished from signal with code '%d'", return_value, WTERMSIG(
          status));

    return_value = wait(&status);
    if (WIFEXITED(status))
      printf("\nParent:\tchild %d finished with code '%d'\n", return_value, WEXITSTATUS(status));
    else if (WIFSTOPPED(status))
      printf("\nParent: child %d finished from signal with code '%d'\n", return_value, WSTOPSIG(
          status));
    else if (WIFSIGNALED(status))
      printf("\nParent:\tchild %d finished from signal with code '%d'\n", return_value, WTERMSIG(
          status));
  }
}
```

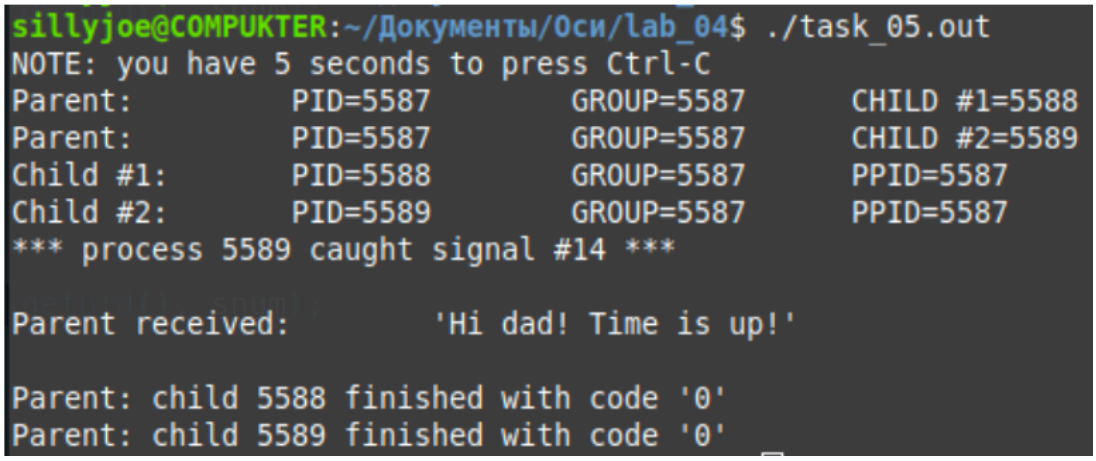
Ниже приведён пример работы программы для задания №5, сигнал прерывания с терминала.

```
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$
sillyjoe@COMPUKTER:~/Документы/Оси/lab_04$ ./task_05.out
NOTE: you have 5 seconds to press Ctrl-C
Parent:         PID=5621        GROUP=5621      CHILD #1=5622
Parent:         PID=5621        GROUP=5621      CHILD #2=5623
Child #1:       PID=5622        GROUP=5621      PPID=5621
Child #2:       PID=5623        GROUP=5621      PPID=5621
^C*** process 5622 caught signal #2 ***
*** process 5621 caught signal #2 ***

Parent received:        'Hi dad! Ctrl-C was pressed!'

Parent: child 5623 finished from signal with code '2'
Parent: child 5622 finished with code '0'
```

Ниже приведён пример работы программы для задания №5, сигнал истечения времени, заданного alarm().



Заключение

Проделав лабораторную работу, я познакомился с такими системны- ми вызовами, как fork(), wait(), exec(), pipe() и signal(); изучил как писать программы, используя эти системные вызовы, а также познако- мился с POSIX сигналами.