УО «Белорусский Государственный Технологический Университет»

**ФИТ**

Отчет по Лабораторной работе №5 на тему «Работа с процессами и потоками. Часть 1»

Выполнили студенты

8-й группы:

Куницкий Н. Д.

Сакович А. С.

Проверила:

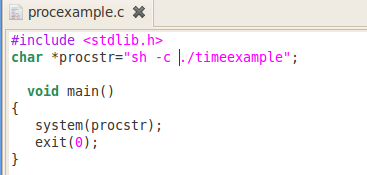
Герман Ю. О

Минск 2016

**Цель**. Освоить системное программирование работы с процессами и потоками в Linux.

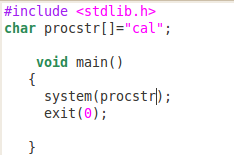
**Краткие теоретические сведения.**

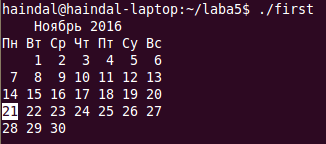
Запустим редактор и наберем следующий код



Мы используем команду system для запуска внешней программы. Вместо procstr нужно указать имя запускаемого процесса.

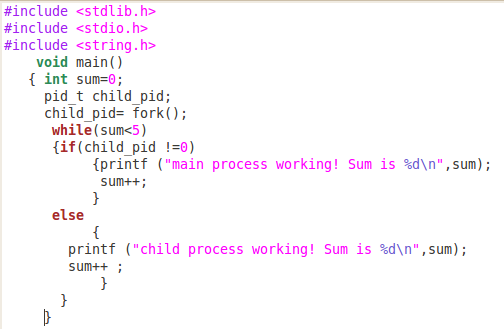
**Пример.** Запускаем календарь:



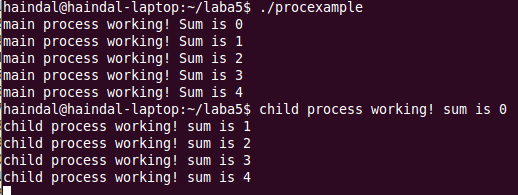


Команда fork.

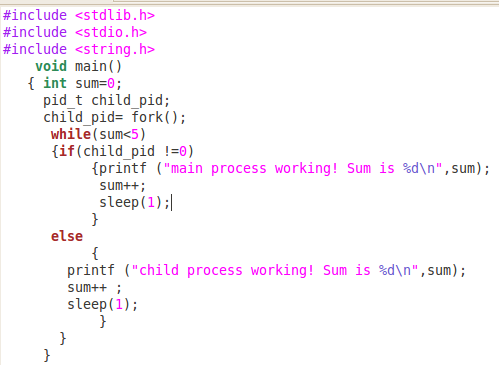
Мы используем команду system для запуска внешней программы. Имеется возможность породить параллельную ветвь программы с помощью команды fork(). Рассмотрим следующий пример:

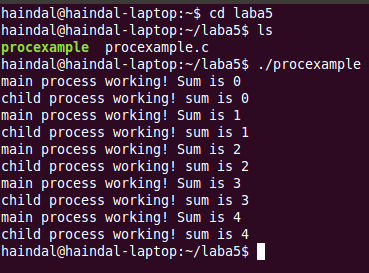


Наберите его и выполните.



В этой программе один процесс (дочерний или родительский) выполняется, пока не закончится. Значит, чтобы дать возможность смены процессов, нужно как-то приостановить выполняющийся процесс. Этого можно добиться с помощью команды sleep(число\_секунд). Вот, как мы изменили программу и что из этого вышло:





**Задание.№1** Теперь нужно сообразить, как осуществить взаимодействие процессов. Стандартным образом для этой цели используется канал (pipe). Мы осуществим взаимодействие через файл. Один процесс будет писать в файл какую-то строку, а второй будет считывать.

**Листинг:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

void main()

{

pid\_t child\_pid;

child\_pid=fork();

if(child\_pid !=0)

{system("echo 'FILE'>file\_1.txt");

sleep(1);

}

else{

system("cat file\_1.txt");

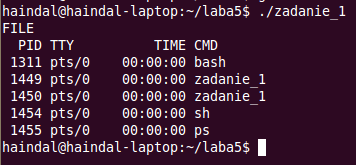
system("ps");

sleep(1);

}

}

**Скриншот:**

****

**Задание №2** Запустить дочерний процесс через fork и пусть он работает в бесконечном цикле (или ограничьте счетчиком проходов по циклу). Через пять секунд родительский процесс должен убить себя, а дочерний процесс выдать команду ps.

**Листинг:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <signal.h>

int main()

{pid\_t child\_pid;

child\_pid=fork();

if(child\_pid!=0)

{printf("Main process:\n");

system("ps");

child\_pid=getpid();

kill(child\_pid,SIGTERM);

}

else{

printf("Child process:\n");

for(;;)

{

sleep(5);

system("ps");

kill(child\_pid,SIGTERM);

}

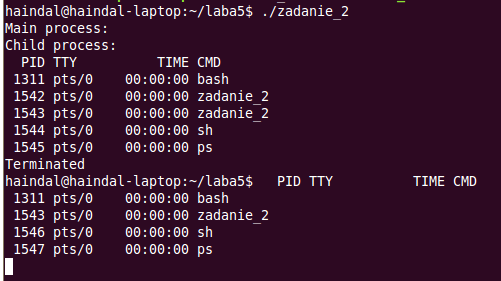
}

system("ps");

return 0;

}

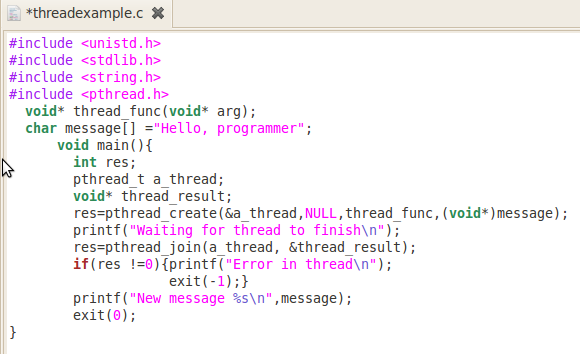
**Скриншот:**

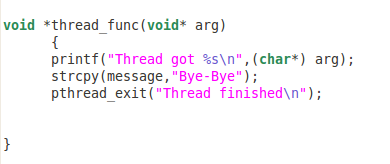
****

**Внимание**. Список выполняющихся процессов можно получить с помощью команды

$ps

Теперь перейдем к потокам. Вот пример простейшего потока, запускаемого из метода main:





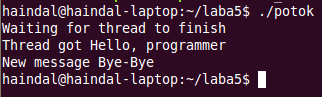
Объявлена функция потока – thread\_func. В методе main объявлена потоковая переменная (собственно поток) pthread\_t a\_thread. Поток создаем командой pthread\_create с указанием выполняемой функции – thread\_func.

Завершение потока реализуется командой pthread\_exit.

Для компиляции нужно указать командную строку особым образом:

$cc –D\_REENTRANT file\_potok.c –o potok -lpthread

**Скриншот:**

****

**Задание№3**. Запустить поток, который возвращает текущую дату и время.

**Листинг:**

#include <time.h>

#include <stdlib.h>

#include <stdio.h>

void main()

{struct tm\* timing;

time\_t t\_time;

(void) time(&t\_time);

timing=localtime(&t\_time);

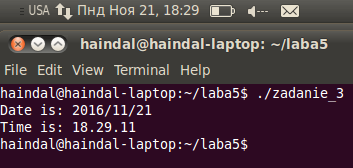
printf("Date is: %02d/%02d/%02d\n",timing->tm\_year+1900,timing->tm\_mon+1,timing->tm\_mday);

printf("Time is: %02d.%02d.%02d\n",timing->tm\_hour, timing->tm\_min, timing->tm\_sec);

exit(0);

}

**Скриншот:**

****

**Вывод:** освоили системное программирование работы с процессами и потоками в Linux.