|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**ОТЧЕТ**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 3 |

**Название:** Linux

**Дисциплина:** Операционные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Д.М.Блохин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Ю. Рязанова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

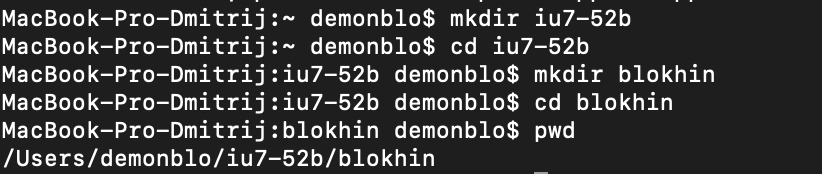
Москва, 2020

# Задание 1

## Условие

Используя команду mkdir cоздайте директорию с имeнем своей группы. В этой директории создать поддиректорию.

## Реализация



# 

# 

# 

# 

# Задание 2

## Условие

Напишите программу, в которой создается дочерний процесс и организуйте как в предке, так и в потомке бесконечные циклы, в которых выводятся идентификаторы процессов с помощью системного вызова getpid().

## 

## Реализация

#include <stdio.h>

#include <unistd.h>

#include <stdbool.h>

#define EXIT\_SUCCESS 0

#define EXIT\_FAILURE 1

int main(void) {

int child = fork();

if (child == -1) {

perror("Error. Can't fork.\n");

return EXIT\_FAILURE;

} else if (child == 0) {

while (true) {

printf("%d\n", getpid());

}

return EXIT\_SUCCESS;

} else {

while (true) {

printf("%d\n", getpid());

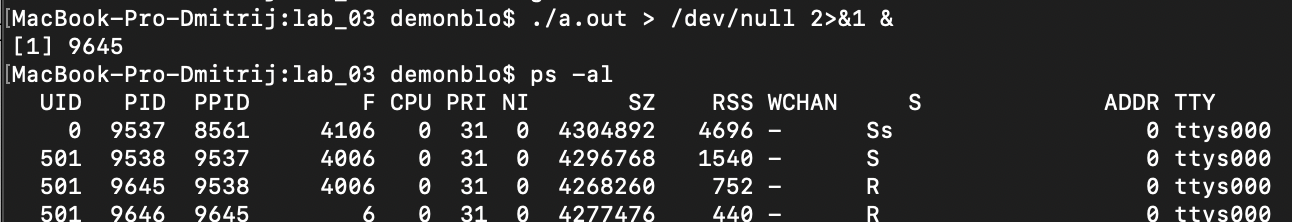
}

return EXIT\_SUCCESS;

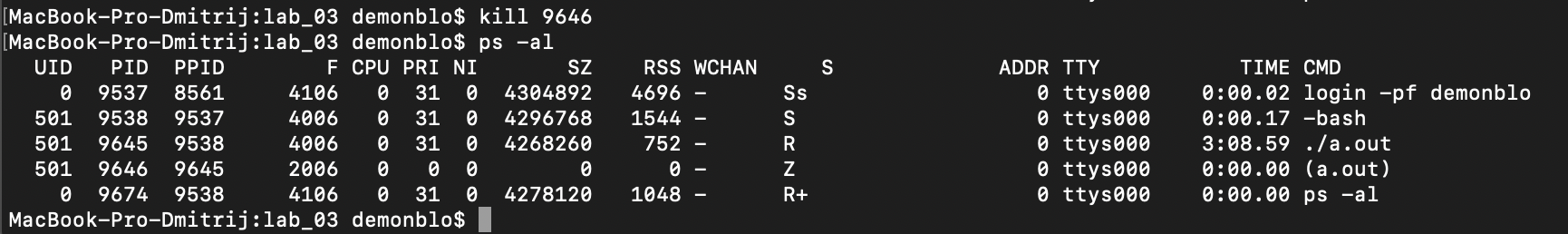
}

}

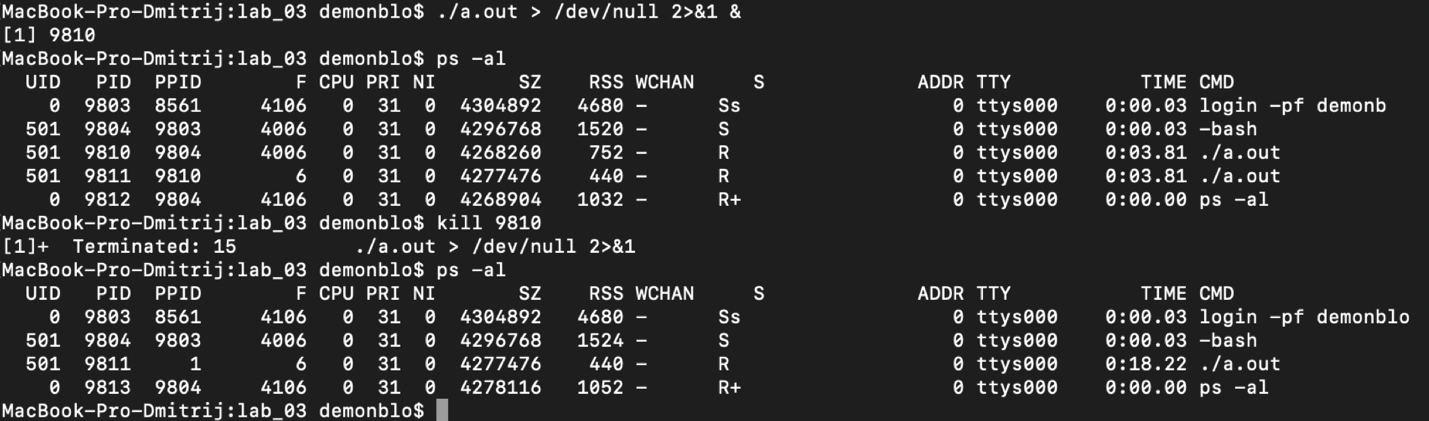
Запустим программу в фоновом режиме и посмотрим идентификаторы созданных процессов:

****

Удаляем дочерний процесс(PID = 9646) и смотрим его статус. Процесс превратился в «зомби».

****

Снова запустим программу. Удалим процесс предка(PID = 9810) и посмотрим идентификатор предка(PPID) у процесса потомка.

****

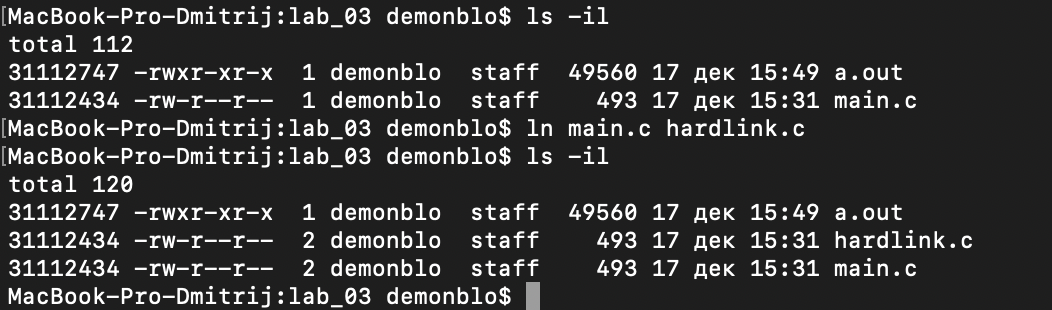
# 

# Задание 3

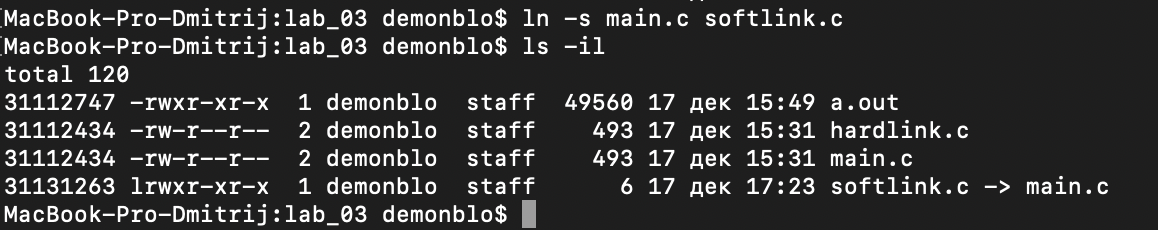
## Условие

Создать жесткую и символическую(мягкую) ссылку на файл.

## Реализация

Создаем жесткую ссылку на файл:

Создаем символическую ссылку на файл:

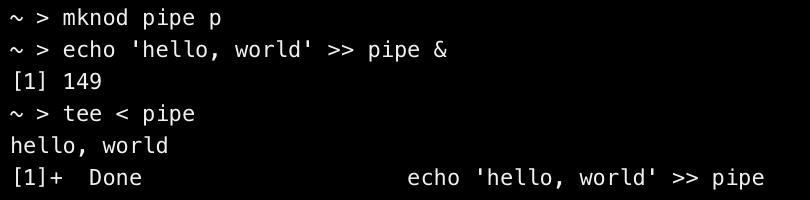


# Задание 4

## Условие

Создать именованный программный канал с помощью команды mknod. Отправить в него сообщение и считать его из канала.

## Реализация

****

# 

# Задание 5

## Условие

Написать программу вывода на экран сообщения «Hello, world!».

## Реализация

#include <stdio.h>

int main(void) {

printf("Hello, World!\n");

return 0;

}

