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

*к лабораторной работе №4*

*По курсу: «Операционные системы»*

*На тему: «Файловая система /proc»*

Студент ИУ7-66Б

Турсунов Ж.Р.

Преподаватель

Рязанова Н.Ю.

*Москва, 2021 г.*

**Часть 1.**

* В пользовательском режиме вывести на экран информацию об окружении процесса с комментариями;
* В пользовательском режиме вывести на экран информацию о состоянии процесса с комментариями;
* Вывести информацию из файла cmdline и директории fd.

# Листинг

|  |
| --- |
|  |

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#define BUF\_SIZE 0x200

void statOutput**(**char **\***buf**)**

**{**

int len **=** strlen**(**buf**);**

int currentName **=** 0**;**

char **\***pch **=** strtok**(**buf**,** " "**);**

**while** **(**pch **!=** **NULL** **&&** i **<** 51**)**

**{**

printf**(**"\n%15s:\t %s"**,** outputNames**[**i**],** pch**);**

pch **=** strtok**(NULL,** " "**);**

i**++;**

**}**

**}**

void simpleOutput**(**char **\***buf**)**

**{**

printf**(**"%s\n"**,** buf**);**

**}**

void read\_one\_file**(**char**\*** filename**,** void **(\***print\_func**)(**char**\*))**

**{**

char buf**[**BUF\_SIZE**];**

int i**,** len**;**

FILE **\***f **=** fopen**(**filename**,** "r"**);**

**while** **((**len **=** fread**(**buf**,** 1**,** BUF\_SIZE**,** f**))** **>** 0**)**

**{**

**for** **(**i **=** 0**;** i **<** len**;** i**++)**

**if(** buf**[**i**]** **==** 0**)**

buf**[**i**]** **=** 10**;**

buf**[**len **-** 1**]** **=** 0**;**

print\_func**(**buf**);**

**}**

fclose**(**f**);**

**}**

int main**(**int argc**,** char **\***argv**[])**

**{**

printf**(**"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"**);**

printf**(**"STAT \n\n"**);**

read\_one\_file**(**"/proc/self/stat"**,** statOutput**);**

printf**(**"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"**);**

printf**(**"ENVIRON\n\n"**);**

read\_one\_file**(**"/proc/self/environ"**,** simpleOutput**);**

printf**(**"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"**);**

printf**(**"CMDLINE\n\n"**);**

read\_one\_file**(**"/proc/self/cmdline"**,** simpleOutput**);**

printf**(**"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"**);**

printf**(**"FD\n\n"**);**

chdir**(**"/proc/self/fd"**);**

execlp**(**"/bin/ls"**,**"ls"**,**"-l"**,**0**);**

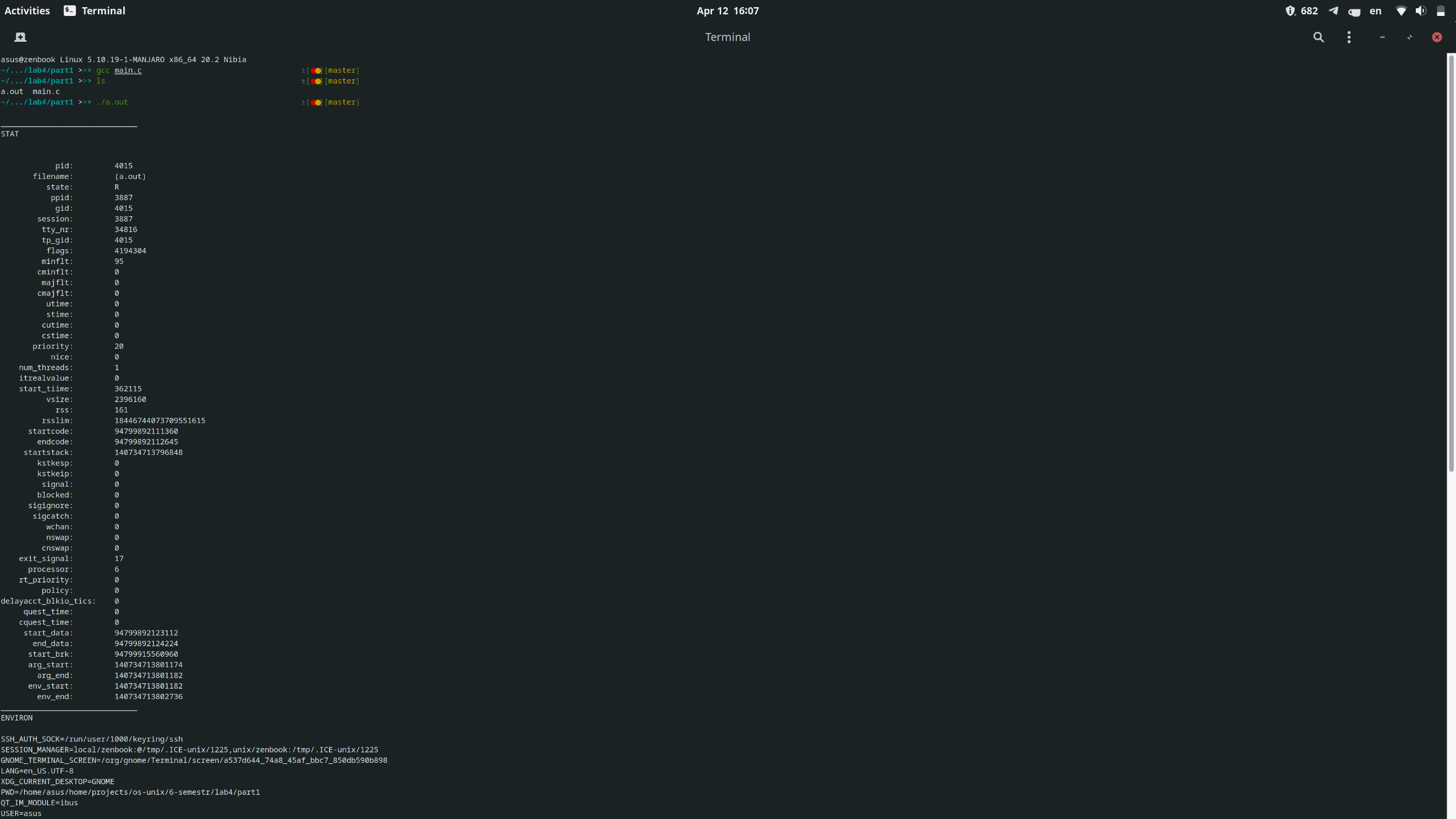
**return** 0**;**

**}**

**Результат работы программы**

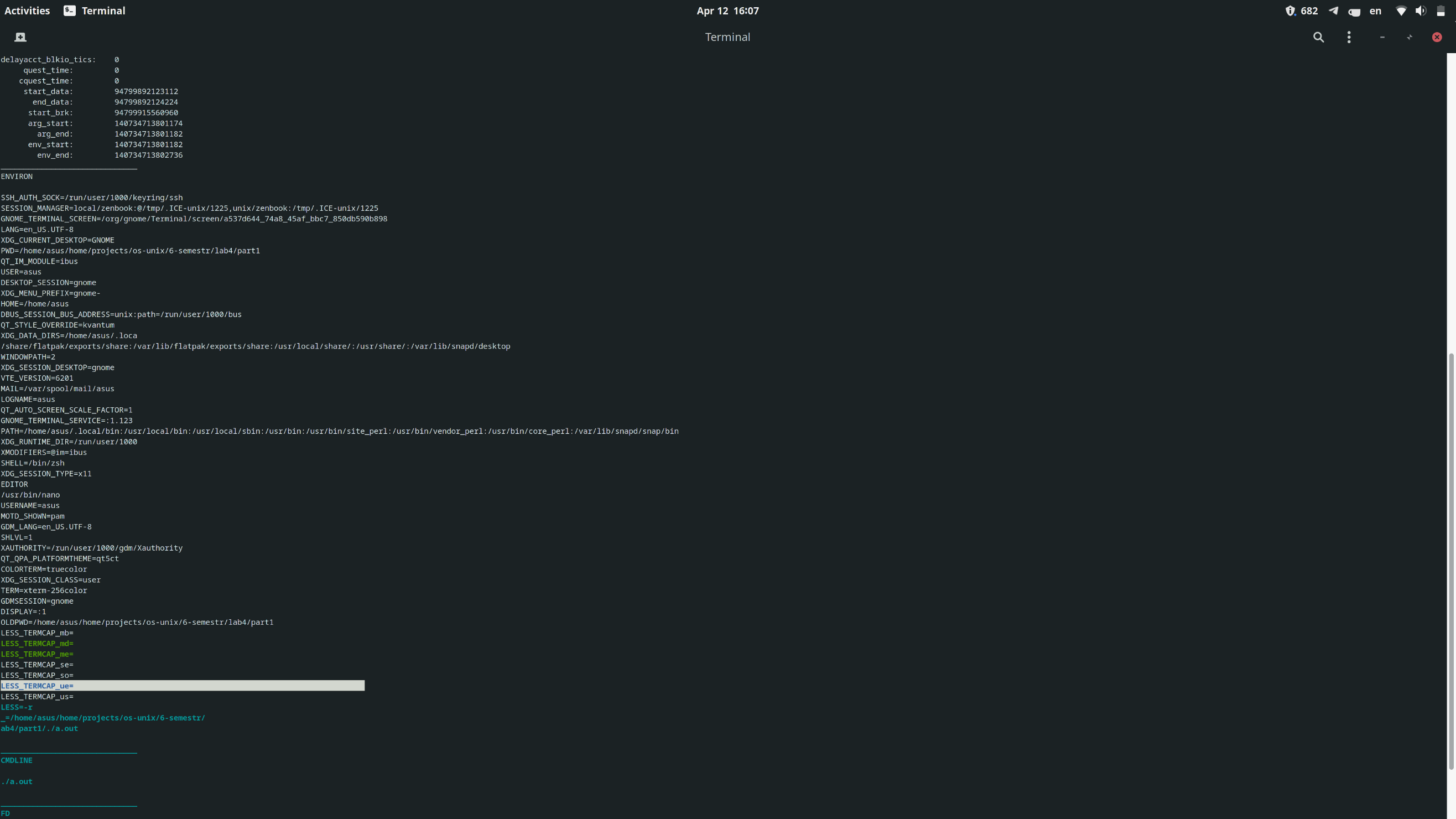
**Содержание файла STAT**

На рисунке представлено содержимое файла stat. В этом файле содержится вся информация о процессе. В данном случае это процесс a.out (написанная мной программа, которая выводит информацию из файла).

****

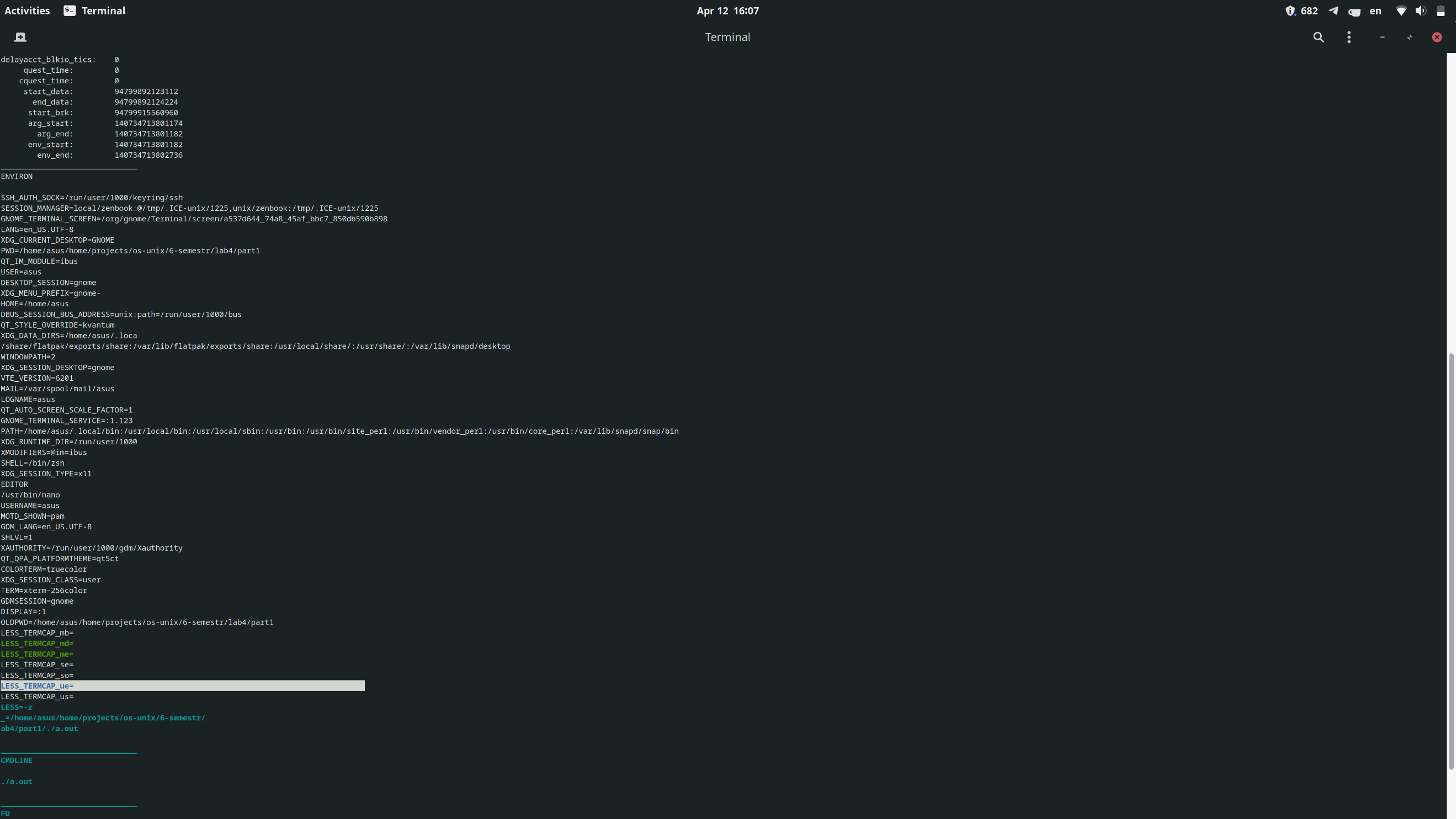
**Содержание файла ENVIRON**

На рисунке представлено содержимое файла environ. Данный файл содержит исходное окружение, которое было установлено при запуске текущего процесса. Переменные окружения разделены символами конца строки (для доступа к поддиректории текущего '\0').



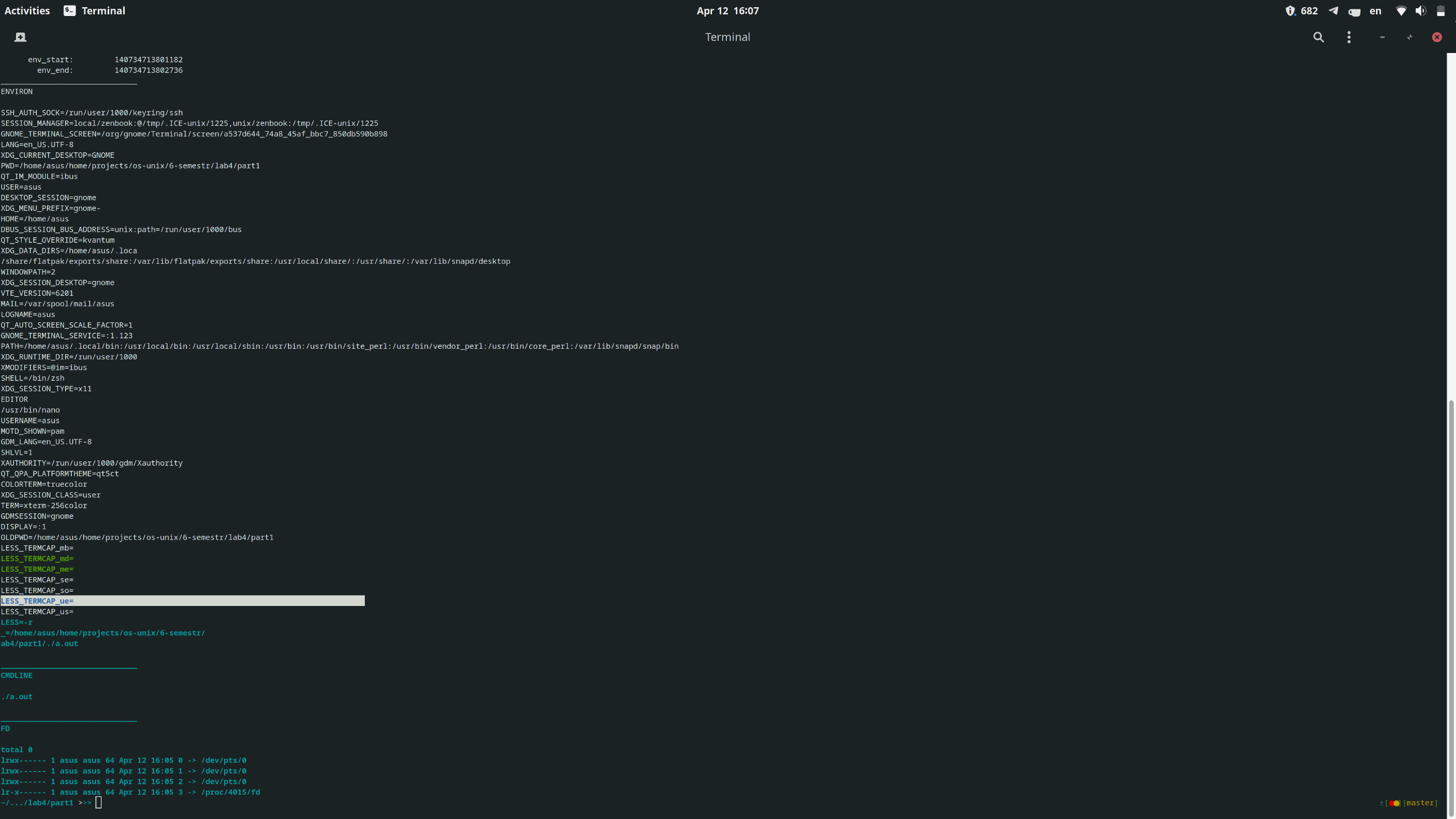
**Содержание файла CMDLINE**

На рисунке представлен вывод файла cmdline. Данный файл содержит полную командную строку процесса, если процесс не находится в состоянии зомби, иначе файл пуст. В данном случае выводит ./a.out, так как этой командой был запущен текущий процесс.

****

**Содержание директории FD**

На рисунке представлено содержимое директории fd. Данная поддиректория содержит одну запись для каждого файла, который открыт процессом. Имя каждой такой записи соответствует номеру файлового дескриптора и является символьной ссылкой на реальный файл. Программа вывела 4 файла, являющиеся символическими ссылками на файлы, которые были открыты процессом



**Часть 2.**

Написать загружаемый модуль ядра, создать файл в файловой системе proc, sysmlink, subdir. Используя соответствующие функции передать данные из пространства пользователя в пространство ядра (введенные данные вывести в файл ядра) и из пространства ядра в пространство пользователя. Продемонстрировать это.

# Листинг

|  |
| --- |
|  |

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/proc\_fs.h>

#include <linux/string.h>

#include <linux/vmalloc.h>

#include <linux/uaccess.h>

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_DESCRIPTION**(**"Message/text Input/Output Kernel Module"**);**

MODULE\_AUTHOR**(**"Tursunov Jasur"**);**

#define MAX\_LENGTH PAGE\_SIZE

static char**\*** storage**;**

static int index**;**

static int next**;**

static struct proc\_dir\_entry **\***proc\_entry**;**

static struct proc\_dir\_entry **\***proc\_slink**;**

static struct proc\_dir\_entry **\***proc\_dir**;**

ssize\_t storage\_write**(**struct file **\***filp**,** const char \_\_user **\***buff**,**

size\_t len**,** loff\_t **\***data**)** **{**

int space\_available **=** **(**MAX\_LENGTH**-**index**)+**1**;**

**if** **(**len **>** space\_available**)** **{**

printk**(**KERN\_INFO "storage: Storage is full!\n"**);**

**return** **-**ENOSPC**;**

**}**

**if** **(**copy\_from\_user**(&**storage**[**index**],** buff**,** len**))**

**return** **-**ENOSPC**;**

index **+=** len**;**

storage**[**index**-**1**]** **=** 0**;**

**return** len**;**

**}**

ssize\_t storage\_read**(**struct file **\***filp**,** char \_\_user **\***buffer**,** size\_t count**,** loff\_t **\***off**)** **{**

int len**;**

**if** **(\***off **>** 0 **||** index **==** 0**)**

**return** 0**;**

**if** **(**next **>=** index**)**

next **=** 0**;**

len **=** copy\_to\_user**(**buffer**,** **&**storage**[**next**],** count**);**

next **+=** len**;**

**\***off **+=** len**;**

**return** len**;**

**}**

static const struct file\_operations proc\_file\_fops **=** **{**

**.**owner **=** THIS\_MODULE**,**

**.**write **=** storage\_write**,**

**.**read **=** storage\_read**,**

**};**

static int \_\_init my\_module\_init**(**void**)** **{**

int ret **=** 0**;**

storage **=** **(**char **\*)**vmalloc**(**MAX\_LENGTH**);**

**if** **(!**storage**)**

ret **=** **-**ENOMEM**;**

**else** **{**

memset**(**storage**,** 0**,** MAX\_LENGTH**);**

proc\_dir **=** proc\_mkdir**(**"directory"**,** **NULL);**

proc\_entry **=** proc\_create**(**"storage"**,** 0644**,** proc\_dir**,** **&**proc\_file\_fops**);**

proc\_slink **=** proc\_symlink**(**"sym\_to\_storage"**,** proc\_dir**,** "/proc/directory/storage"**);**

**if** **(**proc\_entry **==** **NULL)** **{**

ret **=** **-**ENOMEM**;**

vfree**(**storage**);**

printk**(**KERN\_INFO "storage: Couldn't create proc entry\n"**);**

**}**

**else** **{**

index **=** 0**;**

next **=** 0**;**

printk**(**KERN\_INFO "storage: Module loaded.\n"**);**

**}**

**}**

**return** ret**;**

**}**

static void \_\_exit my\_module\_cleanup**(**void**)** **{**

remove\_proc\_entry**(**"storage"**,** **NULL);**

remove\_proc\_entry**(**"directory"**,** **NULL);**

remove\_proc\_entry**(**"sym\_to\_storage"**,** **NULL);**

**if** **(**storage**)**

vfree**(**storage**);**

printk**(**KERN\_INFO "storage: Module exited.\n"**);**

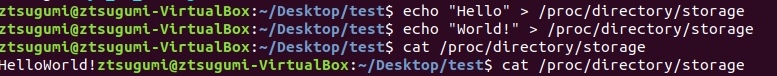
**}**

module\_init**(**my\_module\_init**);**

module\_exit**(**my\_module\_cleanup**);**

**Результат работы программы**

****

****

**Созданная поддиректория в /proc:**



**Результат команды** ls -al:

