|  |  |
| --- | --- |
| 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

**Лабораторная работа №4**

**по курсу “Операционные системы”**

**по теме “Файловая система /proc”**

|  |  |
| --- | --- |
| Студент: | Уласик Е.А. |
| Группа: | ИУ7-61 |
| Преподаватель: | Рязанова Н.Ю. |

1. *г.*

Задание 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 программы

1. #include <stdio.h>
2. #include <string.h>
3. #include <unistd.h>
4. #include <stdlib.h>
5. #include <sys/types.h>
6. #include <dirent.h>
7. #include <limits.h>
9. #define BUF\_SIZE 0x200
11. **char** \*outputNames[] = {
12. "pid", "filename", "state", "ppid", "gid", "session", "tty\_nr", "tp\_gid", "flags", "minflt",
13. "cminflt", "majflt", "cmajflt", "utime", "stime", "cutime", "cstime", "priority", "nice", "num\_threads",
14. "itrealvalue", "start\_tiime", "vsize", "rss", "rsslim", "startcode", "endcode", "startstack", "kstkesp", "kstkeip",
15. "signal", "blocked", "sigignore", "sigcatch", "wchan", "nswap", "cnswap", "exit\_signal", "processor", "rt\_priority",
16. "policy", "delayacct\_blkio\_tics", "cquest\_time", "start\_data", "end\_data", "start\_brk", "arg\_start", "arg\_end", "env\_start", "env\_end",
17. "exit\_code"
18. };
20. **void** print\_stat(**char** \*buf)
21. {
22. **int** i;
23. **char** \*pch = strtok(buf, " ");
25. **while** (pch != NULL && i < 51)
26. {
27. printf("\n%15s:\t %s", outputNames[i], pch);
28. pch = strtok(NULL, " ");
29. i++;
30. }
31. }
33. **void** print\_environ(**char** \*buf)
34. {
35. printf("%s\n", buf);
36. }
38. **void** print\_cmdline(**char** \*buf)
39. {
40. printf("%s\n", buf);
41. }
43. **void** (\*func) (**char**\*);

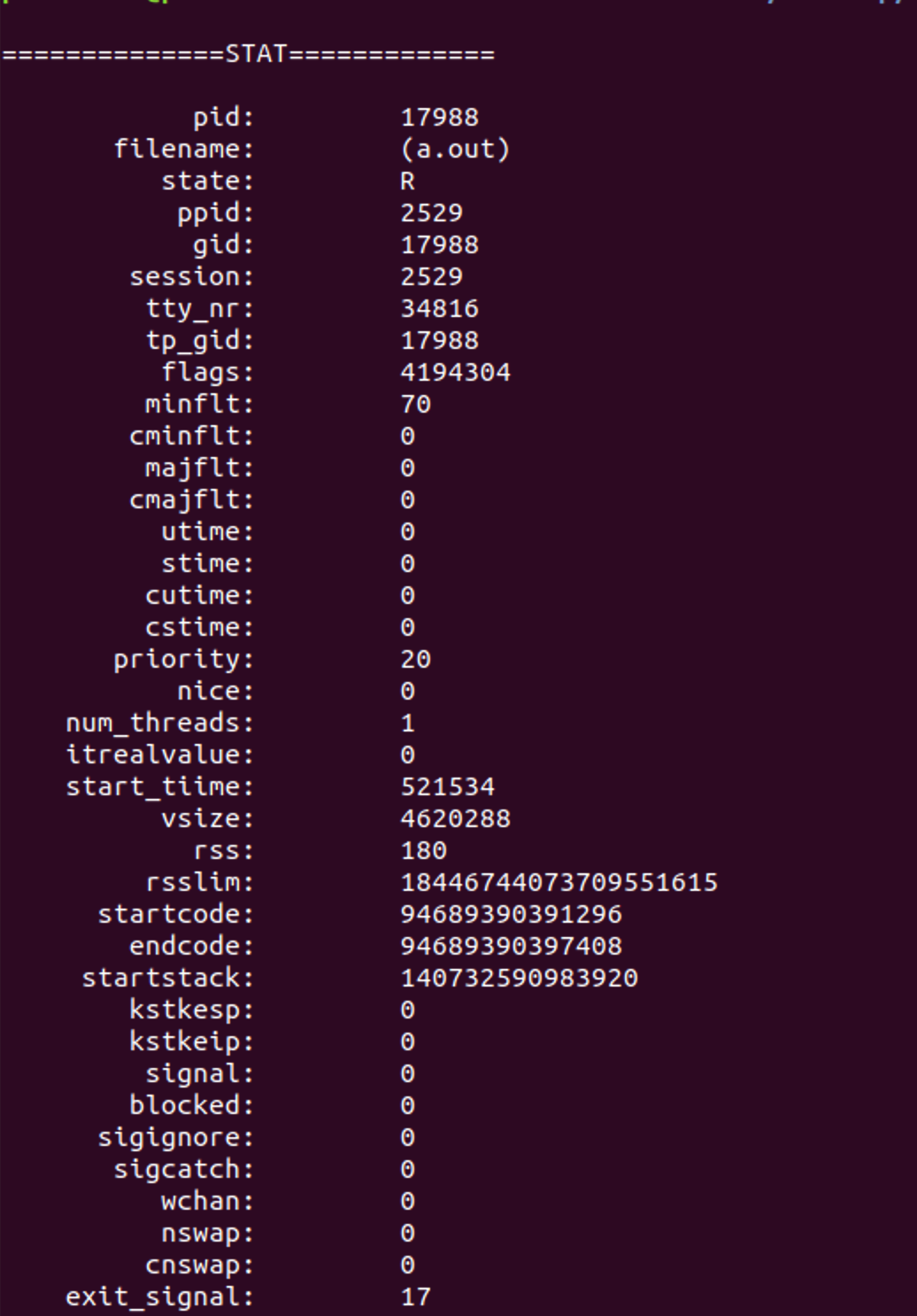
Листинг 1. Код программы задания 1 (часть 1)

Листинг 2. Код программы задания 1 (часть 2)

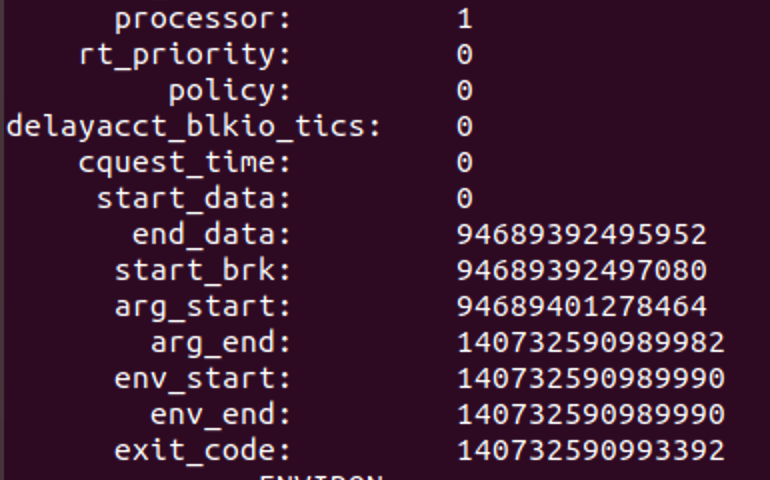
1. **void** read\_file(**char**\* filePath, **void** (\*func)(**char**\*))
2. {
3. **char** buf[BUF\_SIZE];
4. **int** i, len;
5. **FILE** \*f = fopen(filePath, "r");
6. **while** ((len = fread(buf, 1, BUF\_SIZE, f)) > 0)
7. {
8. **for** (i = 0; i < len; i++)
9. **if**( buf[i] == 0)
10. buf[i] = 10;
11. buf[len - 1] = 0;
12. func(buf);
13. }
14. fclose(f);
15. }
17. **void** read\_fd()
18. {
19. **struct** dirent \*dirp;
20. DIR \*dp;
21. **char** str[BUF\_SIZE];
22. **char** path[BUF\_SIZE];
23. dp = opendir("/proc/self/fd");
24. **while** ((dirp = readdir(dp)) != NULL)
25. {
26. **if** ((strcmp(dirp->d\_name, ".") != 0) &&
27. (strcmp(dirp->d\_name, "..") != 0))
28. {
29. sprintf(path, "%s%s", "/proc/self/fd/", dirp->d\_name);
30. readlink(path, str, BUF\_SIZE);
31. printf("%s -> %s\n", dirp->d\_name, str);
32. }
33. }
34. closedir(dp);
35. }

38. **int** main(**int** argc, **char** \*argv[])
39. {
40. printf("\n==============STAT=============\n");
41. read\_file("/proc/self/stat", print\_stat);
42. printf("\n==============ENVIRON=============\n");
43. read\_file("/proc/self/environ", print\_environ);
44. printf("\n==============CMDLINE=============\n");
45. read\_file("/proc/self/cmdline", print\_environ);
46. printf("\n=============FD=============\n");
47. read\_fd();
48. **return** 0;
49. }

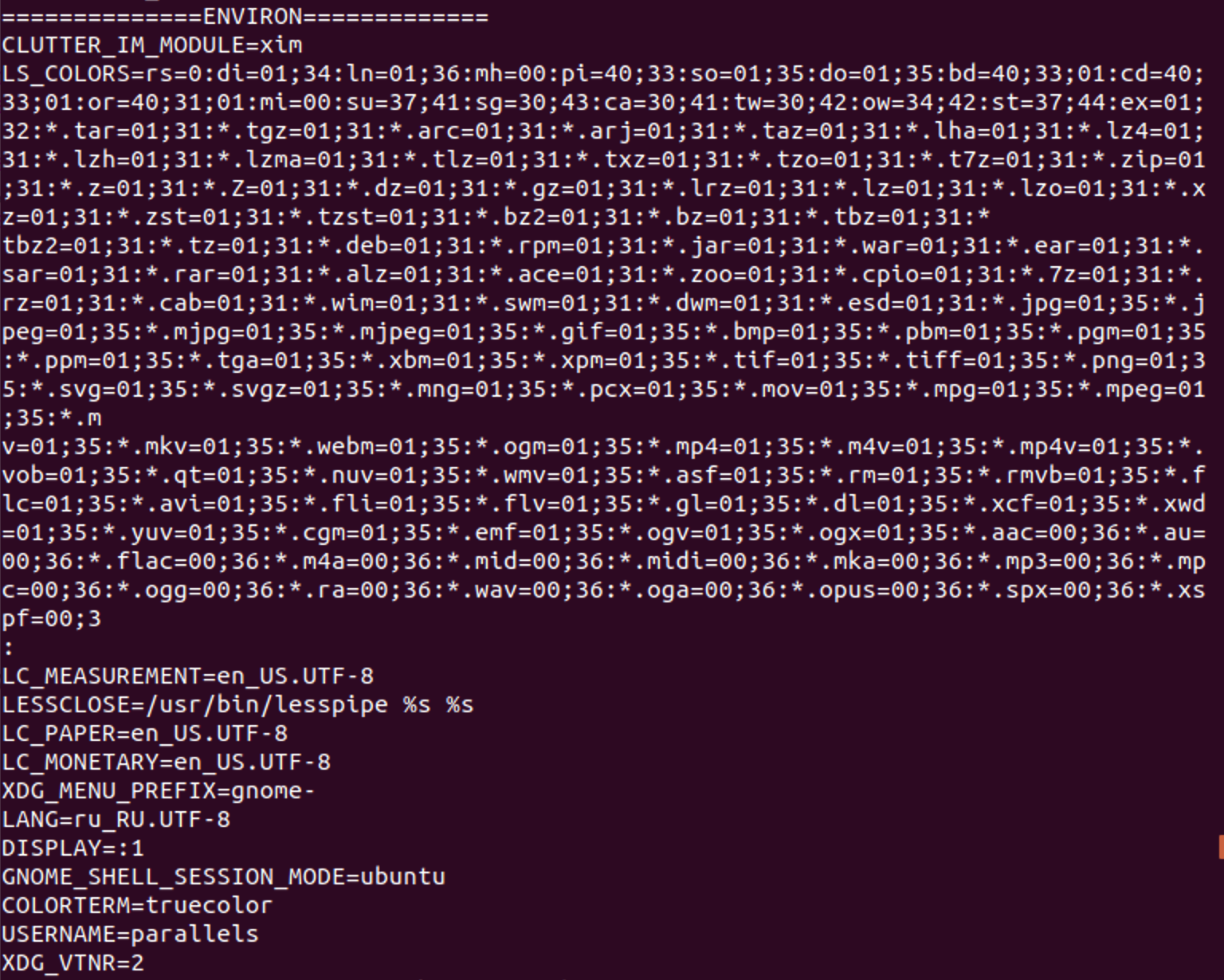
Результат:



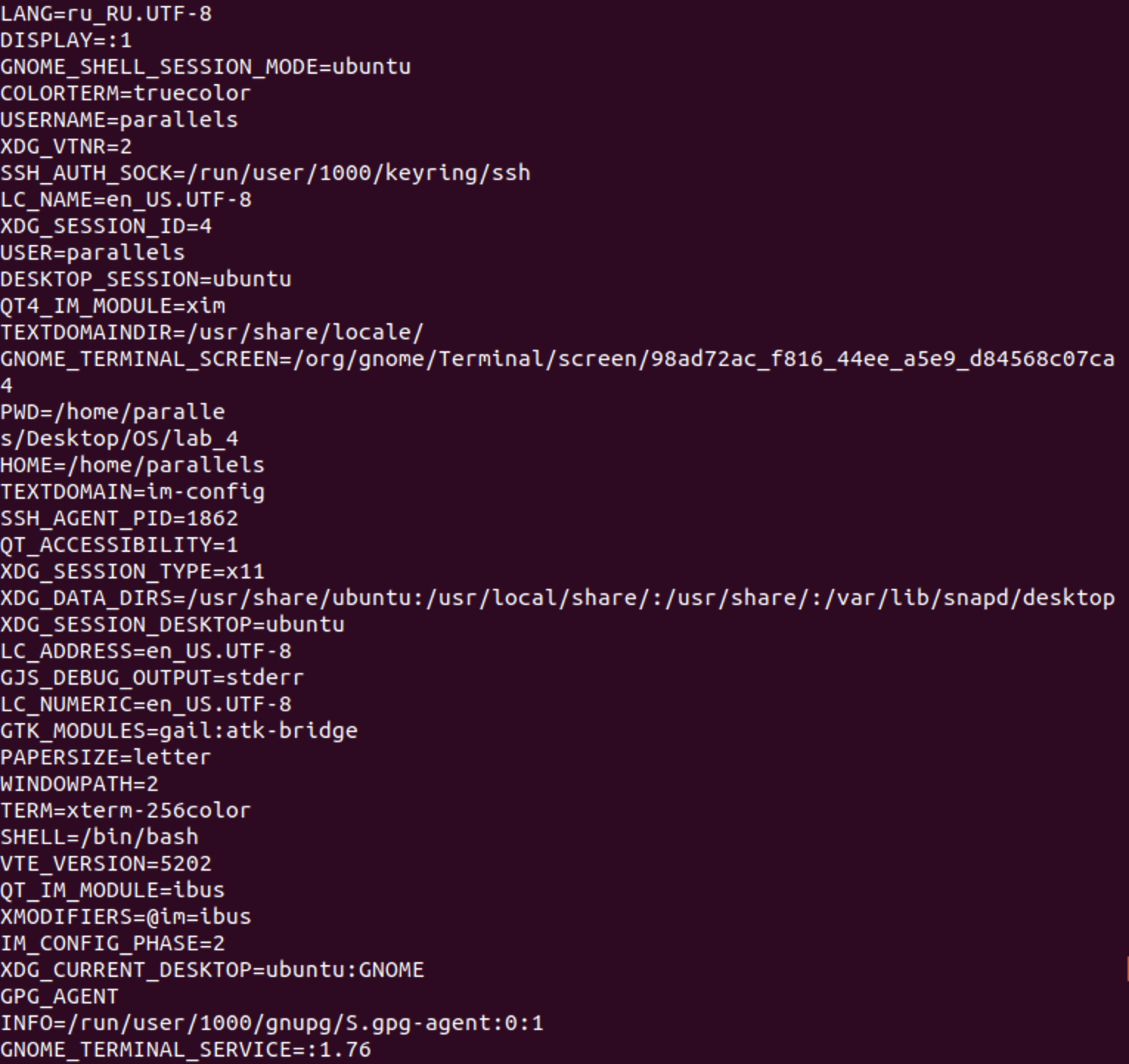
Картинка 1. Вывод на экран файла stat (часть 1)



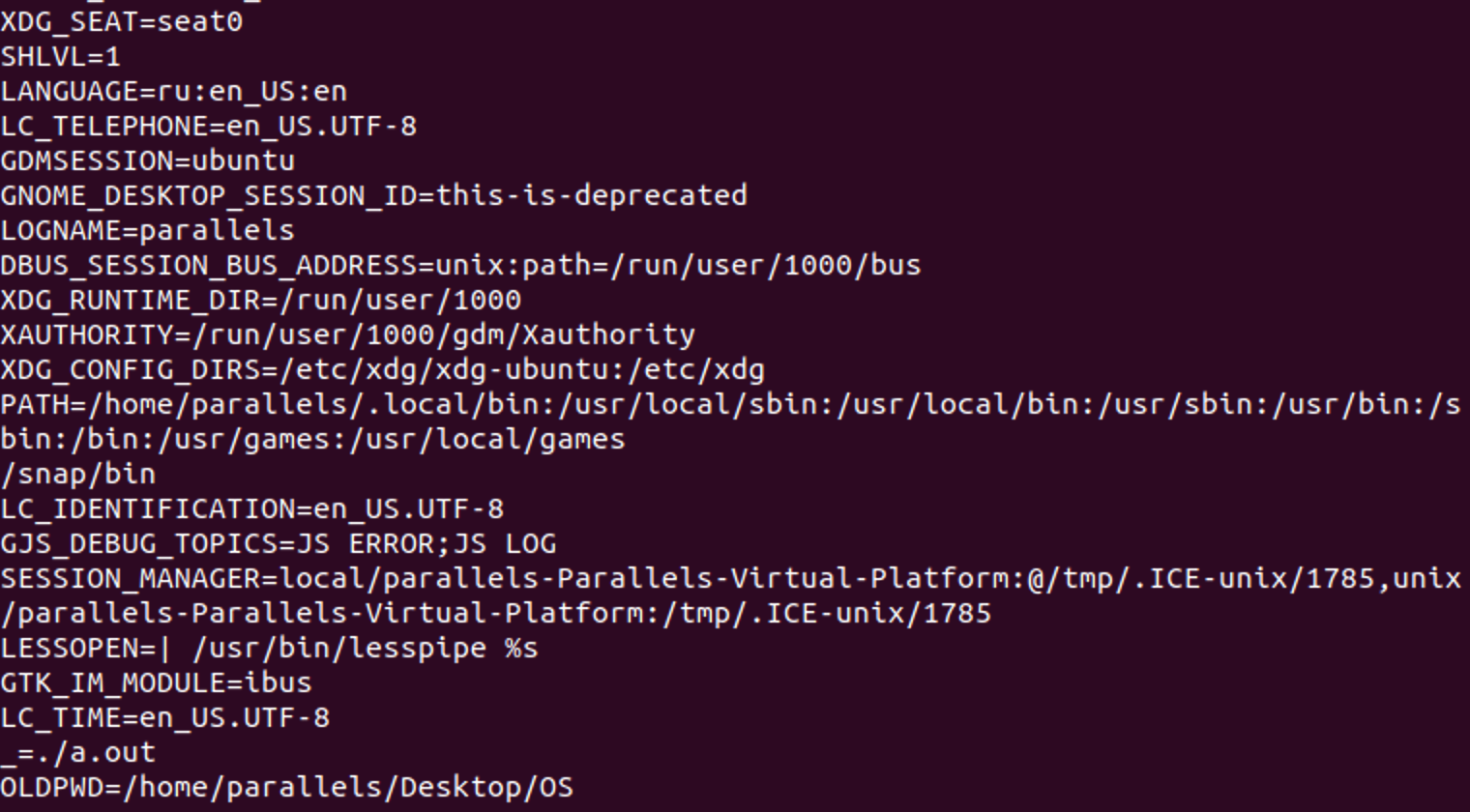
Картинка 2. Вывод на экран файла stat (часть 2)



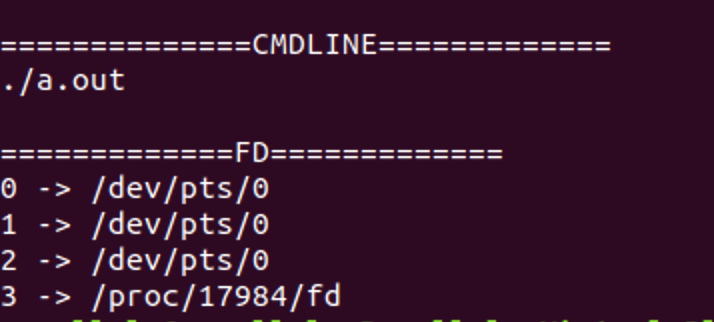
Картинка 3. Вывод на экран содержания файла environ для текущего процесса (часть 1)



Картинка 4. Вывод на экран содержания файла environ для текущего процесса (часть 2)



Картинка 5. Вывод на экран содержания файла environ для текущего процесса (часть 3)



Картинка 6. Вывод на экран информации из директории fd и содержание файла cmdline

Задание 2

* Продемонстрировать передачу данных из режима пользователя в режим ядра и из режима ядра в режим пользователя с помощью скриншота.
* Показать в каталоге созданный файл, директорию и поддиректорию.

Листинг программы

1. #include <linux/module.h>
2. #include <linux/init.h>
3. #include <linux/kernel.h>
4. #include <linux/proc\_fs.h>
5. #include <linux/string.h>
6. #include <linux/vmalloc.h>
7. #include <linux/uaccess.h>
8. #include <linux/sched.h>
9. #include <linux/init\_task.h>
11. MODULE\_LICENSE("GPL");
12. MODULE\_DESCRIPTION("Module for fortune cookies.");
13. MODULE\_AUTHOR("Ulasik Evgeniy");
15. #define MAX\_COOKIE\_LENGTH PAGE\_SIZE
16. #define TEMP\_BUF\_SIZE 512
18. **static** **char**\* cookie\_pot;
19. **static** **int** cookie\_index;
20. **static** **int** next\_fortune;
22. **static** **struct** proc\_dir\_entry \*proc\_entry;
23. **static** **struct** proc\_dir\_entry \*proc\_slink;
24. **static** **struct** proc\_dir\_entry \*proc\_dir;
26. **char** temp[TEMP\_BUF\_SIZE];
28. ssize\_t fortune\_write(**struct** file \*filp, **const** **char** \_\_user \*buff, **size\_t** count, loff\_t \*off)
29. {
30. **int** space\_available = (MAX\_COOKIE\_LENGTH - cookie\_index) + 1;
31. **if** (count > space\_available)
32. {
33. printk(KERN\_INFO "+ Fortune: Cookie pot is full!\n");
34. **return** -ENOSPC;
35. }
36. **if** (copy\_from\_user(&cookie\_pot[cookie\_index], buff, count))
37. **return** -ENOSPC;
38. cookie\_index += count;
39. cookie\_pot[cookie\_index - 1] = 0;
40. **return** count;
41. }
43. ssize\_t fortune\_read(**struct** file \*filp, **char** \_\_user \*buff, **size\_t** count, loff\_t \*off)
44. {
45. **int** len;
46. **if** (cookie\_index == 0 || \*off > 0)
47. **return** 0;
48. **if** (next\_fortune >= cookie\_index)
49. next\_fortune = 0;
50. len = sprintf(temp, "%s\n", &cookie\_pot[next\_fortune]);

Листинг 3. Код программы задания 2 (Часть 1)

Листинг 4. Код программы задания 2 (Часть 2)

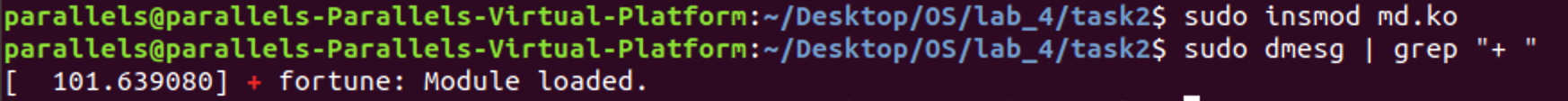
1. copy\_to\_user(buff, temp, len);
2. next\_fortune += len;
3. \*off += len;
4. **return** len;
5. }
7. **static** **const** **struct** file\_operations proc\_file\_fops =
8. {
9. .owner = THIS\_MODULE,
10. .write  = fortune\_write,
11. .read  = fortune\_read,
12. };
14. **static** **int** \_\_init my\_module\_init(**void**)
15. {
16. **int** ret = 0;
17. cookie\_pot = (**char** \*)vmalloc(MAX\_COOKIE\_LENGTH );
18. **if** (!cookie\_pot)
19. ret = -ENOMEM;
20. **else**
21. {
22. memset(cookie\_pot, 0, MAX\_COOKIE\_LENGTH );
23. proc\_dir = proc\_mkdir("fortune", NULL);
24. proc\_entry = proc\_create("fortune", 0666, proc\_dir, &proc\_file\_fops);
25. proc\_slink = proc\_symlink("fortune\_symlink", proc\_dir, "/proc/fortune");
26. **if** (proc\_entry == NULL || proc\_dir == NULL || proc\_slink == NULL)
27. {
28. ret = -ENOMEM;
29. vfree(cookie\_pot);
30. printk(KERN\_INFO "+ fortune: Couldn't create proc entry\n");
31. }
32. **else**
33. {
34. cookie\_index = 0;
35. next\_fortune = 0;
36. printk(KERN\_INFO "+ fortune: Module loaded.\n");
37. }
38. }
39. **return** ret;
40. }
42. **static** **void** \_\_exit my\_module\_cleanup(**void**)
43. {
44. remove\_proc\_entry("fortune", NULL);
45. remove\_proc\_entry("fortune", NULL);
46. remove\_proc\_entry("fortune\_symlink", NULL);
47. **if** (cookie\_pot)
48. vfree(cookie\_pot);
49. printk(KERN\_INFO "+ fortune: Module unloaded.\n");
50. }
52. module\_init(my\_module\_init);
53. module\_exit(my\_module\_cleanup);

Листинг makefile

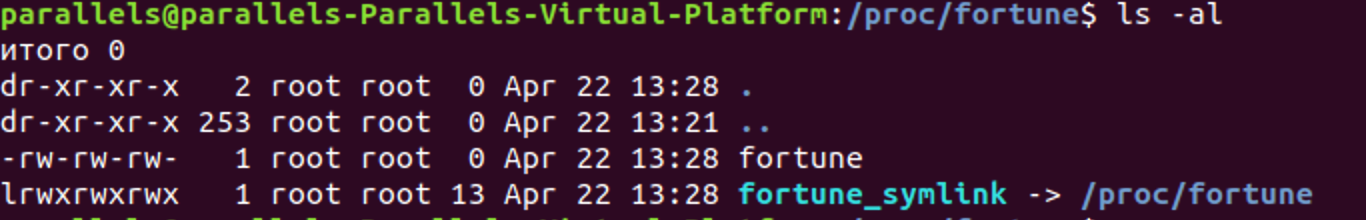
1. KBUILD\_EXTRA\_SYMBOLS = $(shell pwd)/Module.symvers
3. ifneq ($(KERNELRELEASE),)
4. obj-m := md.o
6. else
7. CURRENT = $(shell uname -r)
8. KDIR = /lib/modules/$(CURRENT)/build
9. PWD = $(shell pwd)
11. **default**:
12. sudo $(MAKE) -C $(KDIR) M=$(PWD) modules
13. sudo make clean
15. **clean**:
16. rm \*.o
17. rm \*.mod.c
18. rm \*.order
19. rm \*.ko
20. rm \*.symvers
21. endif

Листинг 5. Код makefile

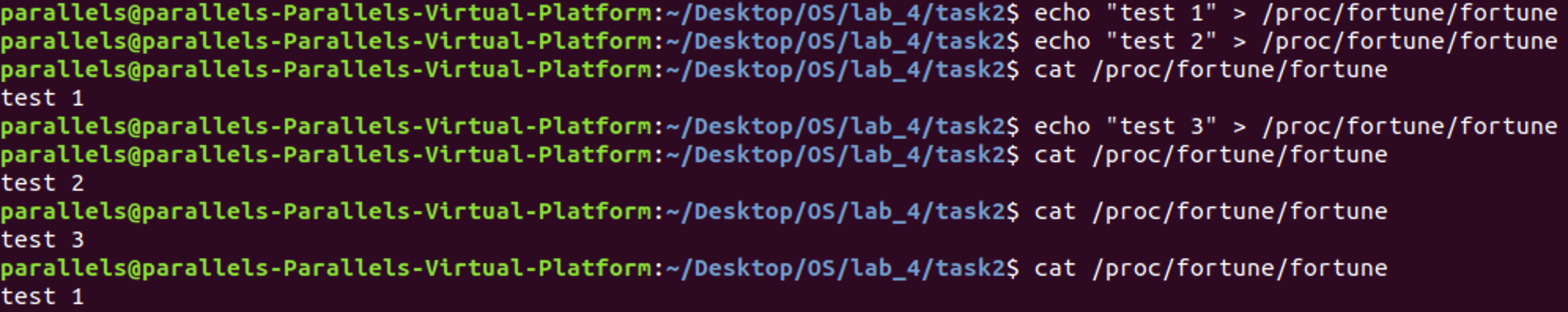
Результат:



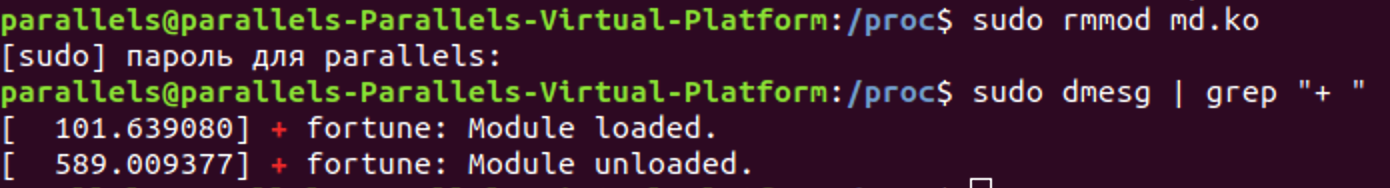
Картинка 7. Загрузка модуля и вывод в системном журнале



Картинка 8. Проверка того, что поддиректория, файл и символьная ссылка были созданы



Картинка 9. Передача данных из режима пользователя в режим ядра и из режима ядра в режим пользователя



Картинка 10. Выгрузка модуля