**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»

# Дисциплина: «Операционные системы»

# Лабораторная работа №4

# Тема работы: «Виртуальная файловая система /proc»

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватель: Рязанова Н. Ю.

Москва, 2020 г.

# Задание 1.

**Используя виртуальную файловую систему /proc, вывести информацию об окружении процесса, информацию, характеризующую состояние процесса, содержание директории fd и cmdline.**

Ниже приведена программа для вывода информации об окружении процесса:

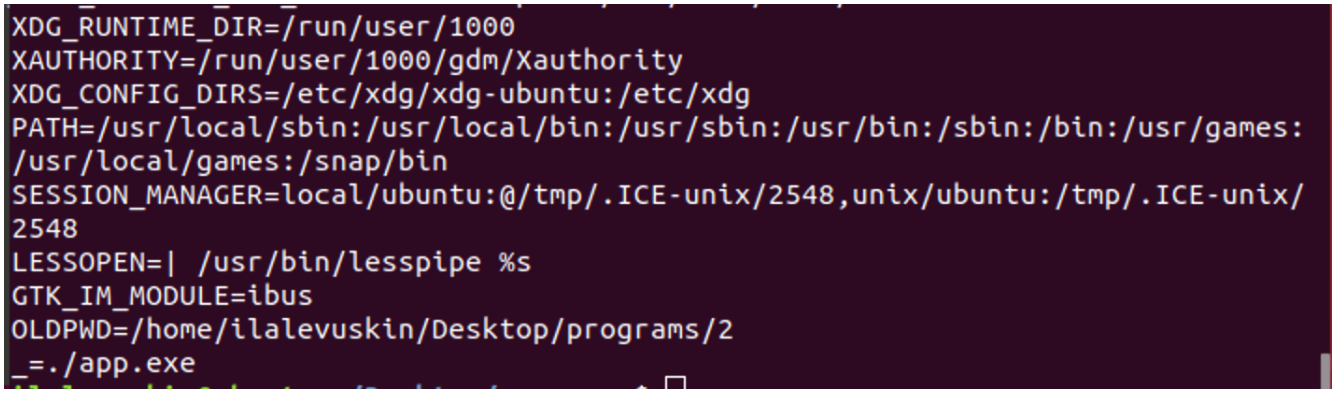
```c
#include <stdio.h>
#define BUF_SIZE 0x100

int main(int argc, char *argv)
{
    char buf[BUF_SIZE];
    int len, i;
    FILE *f;
    f = fopen("/proc/self/environ", "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] = 0;
        printf("%s", buf);
    }
    fclose(f);
    return 0;
}
```

Результат выполнения программы:

```
ilalevuskin@ubuntu:~/Desktop/programs$ ./app.exe
CLUTTER_IM_MODULE=xim
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01
:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:s
t=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:
*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tz
o=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.l
rz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;
31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.j
ar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01
;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.
swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjp
eg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01
;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:
*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m
2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=
01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*
.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=0
1;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:
*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.mid
i=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;3
6:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_MENU_PREFIX=gnome-
LANG=en_US.UTF-8
DISPLAY=:0
GNOME_SHELL_SESSION_MODE=ubuntu
COLORTERM=truecolor
DESKTOP_AUTOSTART_ID=10d0d283ecc2a06a4a158548254429142200000025480007
USERNAME=ilalevuskin
XDG_VTNR=2
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XDG_SESSION_ID=3
```

```
USER=ilalevuskin
DESKTOP_SESSION=ubuntu
QT4_IM_MODULE=xim
TEXTDOMAINDIR=/usr/share/locale/
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/dc5527c9_7925_49aa_9397_4443
1408acf2
PWD=/home/ilalevuskin/Desktop/programs
HOME=/home/ilalevuskin
TEXTDOMAIN=im-config
SSH_AGENT_PID=2625
QT_ACCESSIBILITY=1
XDG_SESSION_TYPE=x11
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/
desktop
XDG_SESSION_DESKTOP=ubuntu
GTK_MODULES=gail:atk-bridge
WINDOWPATH=2
TERM=xterm-256color
SHELL=/bin/bash
VTE_VERSION=5202
QT_IM_MODULE=xim
XMODIFIERS=@im=ibus
IM_CONFIG_PHASE=2
XDG_CURRENT_DESKTOP=ubuntu:GNOME
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
GNOME_TERMINAL_SERVICE=:1.62
XDG_SEAT=seat0
SHLVL=1
GDMSESSION=ubuntu
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=ilalevuskin
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
```




Окружение (environment) или среда — это набор пар ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ доступный каждому пользовательскому процессу. Иными словами, окружение — это набор переменных окружения.

Некоторые переменные окружения:

1. LS_COLORS=rs=0

   используется для определения цветов, с которыми будут выведены имена файлов при вызове ls.

2. LESSCLOSE=/usr/bin/lesspipe %s %s

   LESSOPEN=| /usr/bin/lesspipe %s

   определяют пре- и пост- обработчики файла, который открывается при вызове less.

3. XDG_MENU_PREFIX=gnome-

   XDG_VTNR=2

   XDG_SESSION_ID=3

   XDG_SESSION_TYPE=x11

   XDG_SESSION_DESKTOP=ubuntu

   XDG_CURRENT_DESKTOP=ubuntu:GNOME

   XDG_RUNTIME_DIR=/run/user/1000

   XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg

   переменные, необходимые для вызова xdg-open, использующейся для открытия файла или URL в пользовательском приложении.

4. LANG=en_US.UTF-8

   язык и кодировка пользователя.

5. DISPLAY=:0

   указывает приложениям, куда отобразить графический пользовательский интерфейс.

6. GNOME_SHELL_SESSION_MODE=ubuntu
   GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/scree
   GNOME_DESKTOP_SESSION_ID=this-is-deprecated
   GNOME_TERMINAL_SERVICE=:1.62
   переменные среды рабочего стола GNOME.
7. COLORTERM=truecolor
   определяет поддержку 24-битного цвета.
8. USER=ilalevushkin
   имя пользователя, от чьего имени запущен процесс.
9. USERNAME=ilalevishkin
   имя пользователя, кто инициировал запуск процесса
10. SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
    путь к сокету, который агент использует для коммуникации с другими процессами.
11. TEXTDOMAINDIR=/usr/share/locale/
    TEXTDOMAIN=im-config23
    директория и имя объекта сообщения, получаемого при вызове gettext.
12. PWD=/home/ilalevushkin/Desktop/programs
    путь к рабочей директории.
13. HOME=/home/ilalevushkin
    путь к домашнему каталогу текущего пользователя.
14. SSH_AGENT_PID=2625
    идентификатор процесса ssh-agent.
15. TERM=xterm-256color
    тип запущенного терминала.
16. SHELL=/bin/bash
    путь к предпочтительной оболочке командной строки.
17. SHLVL=1
    уровень текущей командной оболочки.
18. LOGNAME=ilalevushkin
    имя текущего пользователя.

19. PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
    список каталогов, в которых система ищет исполняемые файлы.

20. _=./app.exe - полная командная строка процесса.

Ниже приведена программа для вывода информации, характеризующей состояние процесса:

```c
#include <stdio.h>
#include <string.h>
#define BUF_SIZE 0x100

int main(int argc, char *argv)
{
    char buf[BUF_SIZE];
    int len, i;
    FILE *f;
    f = fopen("/proc/self/stat", "r");
    fread(buf, 1, BUF_SIZE, f);
    char *pch = strtok(buf, " ");

    printf("stat: \n");

    while (pch != NULL)
    {
        printf("%s\n", pch);
        pch = strtok(NULL, " ");
    }
    fclose(f);
    return 0;
}
```

Результат выполнения программы:

```
ilalevuskin@ubuntu:~/Desktop/programs$ ./app.exe
stat:
16616
(app.exe)
R
3354
16616
3354
34816
16616
4194304
70
0
0
0
0
0
0
0
20
0
1
0
471066
4612096
199
18446744073709551615
94767234691072
94767234693800
```

```
140723132907472
0
0
0
0
0
0
0
0
0
17
0
0
0
0
0
0
94767236791696
94767236792336
94767242297344
140723132912◆?Y◆◆
```

Содержимое файла /proc/[pid]/stat:

1. pid=16616 - уникальный идентификатор процесса.
2. comm=(app.exe) - имя исполняемого файла в круглых скобках.
3. state=R - состояние процесса.
4. ppid=3354 - уникальный идентификатор процесса-предка.
5. pgrp=16616 - уникальный идентификатор группы.
6. session=3354 - уникальный идентификатор сессии.
7. tty_nr=34816 – управляющий терминал.
8. tpgid=16616 – уникальный идентификатор группы управляющего терминала.
9. flags=4194304 – флаги.
10. minflt=70 - Количество незначительных сбоев, которые возникли при выполнении процесса, и которые не требуют загрузки страницы памяти с диска.
11. cminflt=0 - количество незначительных сбоев, которые возникли при ожидании окончания работы процессов-потомков.
12. majflt=0 - количество значительных сбоев, которые возникли при работе процесса, и которые потребовали загрузки страницы памяти с диска.
13. cmajflt=0 - количество значительных сбоев, которые возникли при ожидании окончания работы процессов-потомков.
14. utime=0 - количество тиков, которые данный процесс провел в режиме пользователя.
15. stime=0 - количество тиков, которые данный процесс провел в режиме ядра.
16. cutime=0 - количество тиков, которые процесс, ожидающий завершения процессов-потомков, провёл в режиме пользователя.
17. cstime=0 - количество тиков, которые процесс, ожидающий завершения процессов-потомков, провёл в режиме ядра.
18. priority=20 – для процессов реального времени это отрицательный приоритет планирования минус один, то есть число в диапазоне от -2 до -100, соответствующее приоритетам в реальном времени от 1 до 99. Для остальных процессов это необработанное значение nice, представленное в ядре.

Ядро хранит значения nice в виде чисел в диапазоне от 0 (высокий) до 39 (низкий), соответствующих видимому пользователю диапазону от -20 до 19.

19. nice=0 - значение для nice в диапазоне от 19 (наиболее низкий приоритет) до -20 (наивысший приоритет).

20. num_threads=1 – число потоков в данном процессе.

21. itrealvalue=0 – количество мигов до того, как следующий SIGALARM будет послан процессу интервальным таймером. С ядра версии 2.6.17 больше не поддерживается и установлено в 0.

22. starttime=471066 - время в тиках запуска процесса после начальной загрузки системы.

23. vsize=4612096 - размер виртуальной памяти в байтах.

24. rss=199 - резидентный размер: количество страниц, которые занимает процесс в памяти. Это те страницы, которые заняты кодом, данными и пространством стека.

    Сюда не включаются страницы, которые не были загружены по требованию или которые находятся в своппинге.

25. rsslim=18446744073709551615 - текущий лимит в байтах на резидентный размер процесса.

26. startcode=94767234691072 - адрес, выше которого может выполняться код программы.

27. endcode=94767234693800 - адрес, ниже которого может выполняться код программ.

28. startstack=140723132907472 - адрес начала стека.

29. kstkesp=0 - текущее значение ESP (указателя стека).

30. kstkeip=0 - текущее значение EIP (указатель команд).

31. signal=0 - битовая карта ожидающих сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.

32. blocked=0 - битовая карта блокируемых сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.

33. sigignore=0 - битовая карта игнорируемых сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.

34. sigcatch=0 - битовая карта перехватываемых сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.

35. wchan=0 - "канал в котором ожидает процесс.

36. nswap=0 - количество страниц на своппинге (не обслуживается).

37. cnswap=0 - суммарное nswap для процессов-потомков (не обслуживается).

38. exit_signal=17 - сигнал, который будет послан предку, когда процесс завершится.

39. processor=0 - номер процессора, на котором последний раз выполнялся процесс.

40. rt_priority=0 - приоритет планирования реального времени, число в диапазоне от 1 до 99 для процессов реального времени, 0 для остальных.

41. policy=0 - политика планирования.

42. delayacct_blkio_ticks=0 - суммарные задержки ввода/вывода в тиках.

43. guest_time=0 – гостевое время процесса (время, потраченное на выполнение виртуального процессора на гостевой операционной системе) в тиках.

44. cguest_time=0 - гостевое время для потомков процесса в тиках.

45. start_data=94767236791696 - адрес, выше которого размещаются инициализированные и неинициализированные (BSS) данные программы.

46. end_data=94767236791696 - адрес, ниже которого размещаются инициализированные и неинициализированные (BSS) данные программы.

47. start_brk=94767242297344 - адрес, выше которого куча программы может быть расширена с использованием brk().

48. arg_start=140723132912 - адрес, выше которого размещаются аргументы командной строки (argv).

Листинг ссылка программы для вывода содержания директории fd:

```c
#include <stdio.h>
#include <string.h>
#include <dirent.h>

#include <unistd.h>
#define BUF_SIZE 0x1000

void read_fd()
{
    printf("\nfd:\n");
    struct dirent *dirp;
    DIR *dp;

    char str[BUF_SIZE];
    char path[BUF_SIZE];

    dp = opendir("/proc/self/fd");
    while ((dirp = readdir(dp)) != NULL)
    {
        if ((strcmp(dirp->d_name, ".") != 0) && (strcmp(dirp->d_name, "..") != 0))
        {
            sprintf(path, "%s%s", "/proc/self/fd/", dirp->d_name);
            readlink(path, str, BUF_SIZE);
            path[BUF_SIZE] = '\0';
            printf("%s -> %s\n", dirp->d_name, str);
        }
    }
    closedir(dp);
}

int main(int argc, char *argv)
{
    read_fd();
    return 0;
}
```

Результат выполнения программы

```
ilalevuskin@ubuntu:~/Desktop/programs$ ./app.exe

fd:
0 -> /dev/pts/0
1 -> /dev/pts/0
2 -> /dev/pts/0
3 -> /proc/17365/fd
```

Ниже приведена программа для вывода содержания директории cmdline:

```c
#include <stdio.h>
#include <unistd.h>
#define BUF_SIZE 0x1000

void read_cmdline()
{
    char buf[BUF_SIZE];
    FILE *f;
    int len;
    f = fopen("/proc/self/cmdline", "r");
    len = fread(buf, 1, BUF_SIZE, f);
    buf[len - 1] = 0;
    printf("cmdline for %d process = %s\n", getpid(), buf);
    fclose(f);
}

int main()
{
    read_cmdline();
    return 0;
}
```

Результат выполнения программы:

```
ilalevuskin@ubuntu:~/Desktop/programs$ ./app.exe
cmdline for 17559 process = ./app.exe
```

## Задание 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 программы:

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <linux/string.h>
#include <linux/vmalloc.h>
#include <asm/uaccess.h>
#include <linux/uaccess.h>

#define COOKIE_BUF_SIZE PAGE_SIZE

ssize_t fortune_read(struct file *file, char *buf, size_t count, loff_t *f_pos);
ssize_t fortune_write(struct file *file, const char *buf, size_t count, loff_t *f_pos);
int fortune_init(void);
void fortune_exit(void);

struct file_operations fops = {
    .owner = THIS_MODULE,
    .read = fortune_read,
    .write = fortune_write,
};

char *cookie_buf;
struct proc_dir_entry *proc_file;
unsigned int read_index;
unsigned int write_index;

ssize_t fortune_read(struct file *file, char *buf, size_t count, loff_t *f_pos)
{
    int len;
    if (write_index == 0 || *f_pos > 0)
    {
        return 0;
    }

    if (read_index >= write_index)
    {
        read_index = 0;
    }
```

```
    len = sprintf(buf, "%s\n", &cookie_buf[read_index]);
    read_index += len;
    *f_pos += len;

    return len;
}

ssize_t fortune_write(struct file *file, const char *buf, size_t count, loff_t *f_pos)
{
    int free_space = (COOKIE_BUF_SIZE - write_index) + 1;

    if (count > free_space)
    {
        printk(KERN_INFO "Cookie pot fill.\n");
        return -ENOSPC;
    }

    if (copy_from_user(&cookie_buf[write_index], buf, count))
    {
        return -EFAULT;
    }

    write_index += count;
    cookie_buf[write_index - 1] = 0;

    return count;
}

int fortune_init(void)
{
    cookie_buf = vmalloc(COOKIE_BUF_SIZE);

    if (!cookie_buf)
    {
        printk(KERN_INFO "Not enough memory for the cookie pot.\n");
        return -ENOMEM;
    }
```

```
    memset(cookie_buf, 0, COOKIE_BUF_SIZE);
    proc_file = proc_create("fortune", 0666, NULL, &fops);

    if (!proc_file)
    {
        vfree(cookie_buf);
        printk(KERN_INFO "Cannot create fortune file.\n");
        return -ENOMEM;
    }

    read_index = 0;
    write_index = 0;

    proc_mkdir("Dir_in_proc", NULL);
    proc_symlink("Symbolic_in_proc", NULL, "/proc/fortune");

    printk(KERN_INFO "Fortune module loaded.\n");
    return 0;
}

void fortune_exit(void)
{
    remove_proc_entry("fortune", NULL);

    if (cookie_buf)
    {
        vfree(cookie_buf);
    }

    printk(KERN_INFO "Fortune module unloaded.\n");
}

module_init(fortune_init);
module_exit(fortune_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Ilya Levushkin");
```

Скриншоты:

```
ilalevuskin@ubuntu:~/Desktop/programs$ make
make -C /lib/modules/5.3.0-40-generic/build M=/home/ilalevuskin/Desktop/progr
ams modules
make[1]: Entering directory '/usr/src/linux-headers-5.3.0-40-generic'
  CC [M]  /home/ilalevuskin/Desktop/programs/fortune.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/ilalevuskin/Desktop/programs/fortune.mod.o
  LD [M]  /home/ilalevuskin/Desktop/programs/fortune.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.3.0-40-generic'
sudo make clean
rm -rf .tmp_versions
rm .fortune.*
rm *.o
rm *.mod.c
rm *.symvers
rm *.order
ilalevuskin@ubuntu:~/Desktop/programs$ sudo insmod ./fortune.ko
ilalevuskin@ubuntu:~/Desktop/programs$ dmesg | tail -1
[ 5703.429785] Fortune module loaded.
ilalevuskin@ubuntu:~/Desktop/programs$ lsmod | grep fortune
fortune                16384  0
```

```
ilalevuskin@ubuntu:~/Desktop/programs$ echo "Hello!" > /proc/fortune
ilalevuskin@ubuntu:~/Desktop/programs$ echo "It's me, Ilya!" > /proc/fortune
ilalevuskin@ubuntu:~/Desktop/programs$ cat /proc/fortune
Hello!
ilalevuskin@ubuntu:~/Desktop/programs$ cat /proc/fortune
It's me, Ilya!
ilalevuskin@ubuntu:~/Desktop/programs$ sudo rmmod ./fortune.ko
ilalevuskin@ubuntu:~/Desktop/programs$ dmesg | tail -1
[ 5797.773319] Fortune module unloaded.
ilalevuskin@ubuntu:~/Desktop/programs$ lsmod | grep fortune
```

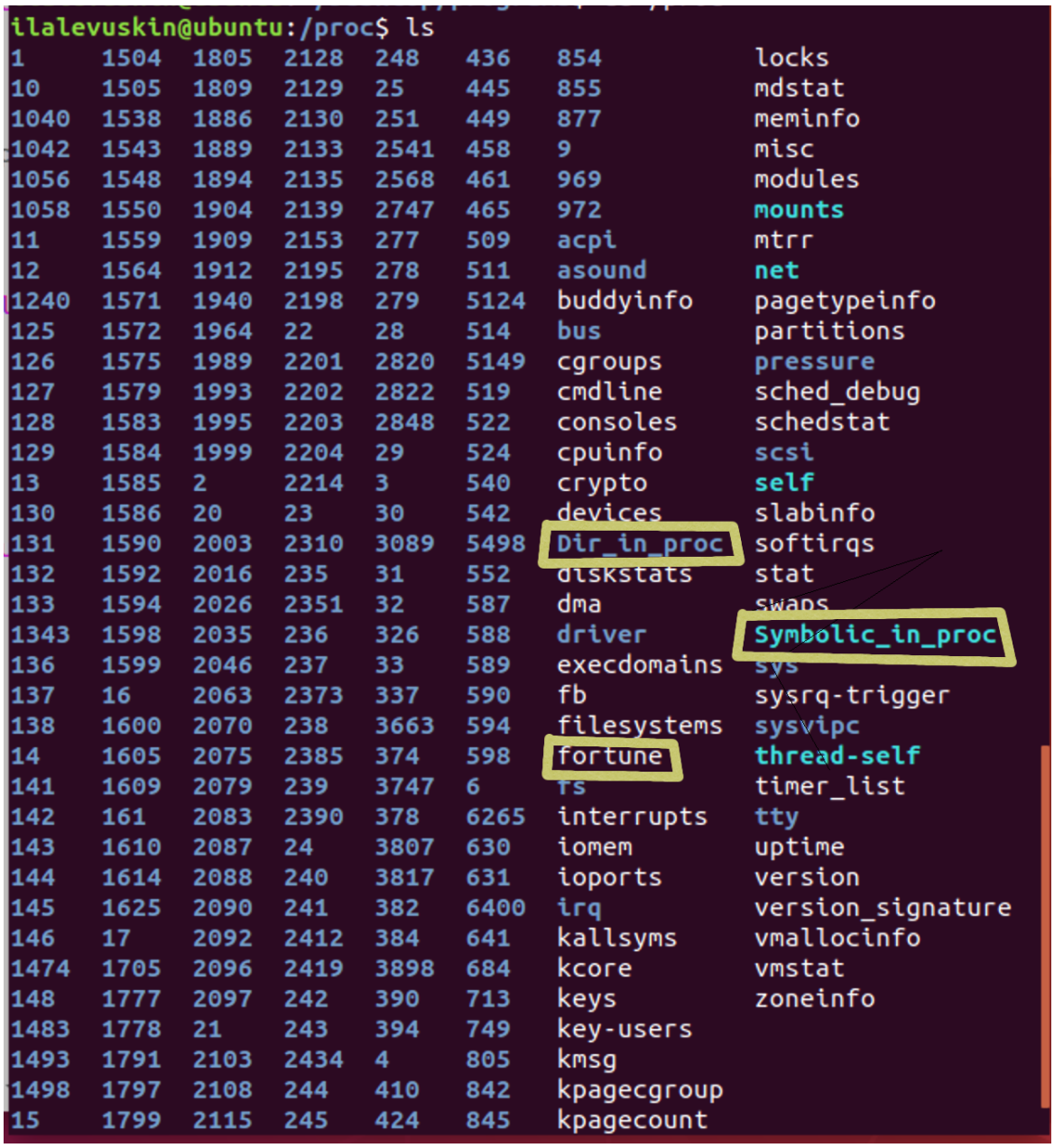



Рис. 1: Созданные файл, поддиректория и символьная ссылка в директории /proc