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»

Отчёт по лабораторной работе №4 по курсу «Операционные системы»

Тема Файловая система \proc

Студент Якуба Д.В.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Рязанова Н.Ю.

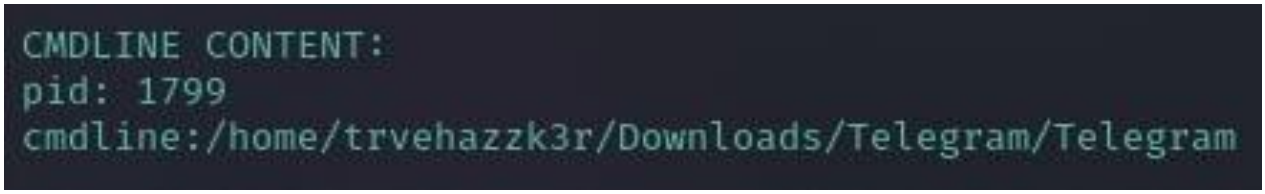
Москва, 2021 г

/proc/pid/cmdline

```
void printCMDLINE()
{
    char pathToOpen[PATH_MAX];
    snprintf(pathToOpen, PATH_MAX, "/proc/%d/cmdline", PID);
    FILE *file = fopen(pathToOpen, "r");

    char buf[BUFFSIZE];
    int len = fread(buf, 1, BUFFSIZE, file);
    buf[len - 1] = 0;

    printf("\nCMDLINE CONTENT:\n");
    printf("pid: %d\ncmdline:%s\n", PID, buf);

    fclose(file);
}
```

Данный файл содержит полную командную строку процесса, если он полностью не выгружен или убит. В любом из последних двух случаев файл пуст и чтение его приводит к тому же результату, что и чтение пустой строки.

Результат выполнения:



/proc/pid/environ

```
void printENVIRON()
{
    char pathToOpen[PATH_MAX];
    snprintf(pathToOpen, PATH_MAX, "/proc/%d/environ", PID);
    FILE *file = fopen(pathToOpen, "r");

    int len;
    char buf[BUFFSIZE];
    printf("\nENVIRON CONTENT:\n");
    while ((len = fread(buf, 1, BUFFSIZE, file)) > 0)
    {
        for (int i = 0; i < len; i++)
            if (!buf[i])
                buf[i] = '\n';
        buf[len - 1] = '\n';
        printf("%s", buf);
    }

    fclose(file);
}
```

Файл содержит набор пар “переменная=значение”, доступный каждому пользовательскому процессу. Такой набор называется набором переменных окружения.

Результат выполнения:

```
ENVIRON CONTENT:
POWERSHELL_TELEMETRY_OPTOUT=1
PAM_KWALLET5_LOGIN=/run/user/1000/kwallet5.socket
USER=trvehazzk3r
XDG_SEAT=seat0
XDG_SESSION_TYPE=x11
HOME=/home/trvehazzk3r
DESKTOP_SESSION=plasma
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COMMAND_NOT_FOUND_INSTALL_PROMPT=1
LOGNAME=trvehazzk3r
XDG_SESSION_CLASS=user
XDG_SESSION_ID=3
PATH=/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
XDG_RUNTIME_DIR=/run/user/1000
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session1
DISPLAY=:0
LANG=myv_RU.UTF-8
XDG_CURRENT_DESKTOP=KDE
XAUTHORITY=/home/trvehazzk3r/.Xauthority
XDG_SESSION_DESKTOP=KDE
SHELL=/usr/bin/zsh
XDG_VTNR=7
PWD=/home/trvehazzk3r
SHLVL=0
OLDPWD=/home/trvehazzk3r
GTK_MODULES=gail:atk-bridge
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
QT_QPA_PLATFORMTHEME=
QT_AUTO_SCREEN_SCALE_FACTOR=0
_JAVA_OPTIONS=-Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
QT_ACCESSIBILITY=1
_=/usr/bin/ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-vEKMgqDnX0cU/agent.909
SSH_AGENT_PID=957
XCURSOR_THEME=breeze_cursors
XCURSOR_SIZE=24
KDE_FULL_SESSION=true
KDE_SESSION_VERSION=5
KDE_SESSION_UID=1000
KDE_APPLICATIONS_AS_SCOPE=1
GTK_RC_FILES=/etc/gtk/gtkrc:/home/trvehazzk3r/.gtkrc:/home/trvehazzk3r/.config/gtkrc
GTK2_RC_FILES=/etc/gtk-2.0/gtkrc:/home/trvehazzk3r/.gtkrc-2.0:/home/trvehazzk3r/.config/gtkrc-2.0
SESSION_MANAGER=local/TrveHazzk3r:@/tmp/.ICE-unix/1040,unix/TrveHazzk3r:/tmp/.ICE-unix/1040
DESKTOP_STARTUP_ID=TrveHazzk3r;1617819540;71774;1070_TIME169535
```

Приведённые переменные окружения:

USER – пользователь, запустивший процесс.

HOME – домашний каталог текущего пользователя.

LOGNAME – имя текущего пользователя.

PATH – список каталогов, в которых находятся исполняемые файлы.

LANG – язык и кодировка текущего пользователя.

SHELL – путь к оболочке командной строки.

PWD – путь к рабочей директории.

SHLVL – уровень текущей командной оболочки.

OLDPWD – путь к предыдущему рабочему каталогу.

/proc/pid/fd

```
void printFD()
{
    char pathToOpen[PATH_MAX];
    snprintf(pathToOpen, PATH_MAX, "/proc/%d/fd/", PID);
    DIR *dir = opendir(pathToOpen);

    printf("\nFD CONTENT:\n");

    struct dirent *readDir;
    char string[PATH_MAX];
    char path[BUFFSIZE] = {'\0'};
    while ((readDir = readdir(dir)) != NULL)
    {
        if ((strcmp(readDir->d_name, ".") != 0) && (strcmp(readDir->d_name,
"..") != 0))
        {
            sprintf(path, "%s%s", pathToOpen, readDir->d_name);
            readlink(path, string, PATH_MAX);
            printf("{%s} -- %s\n", readDir->d_name, string);
        }
    }

    closedir(dir);
}
```

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.

Результат выполнения представлен на следующей странице.

```
FD CONTENT:
{0} -- pipe:[24174]
{1} -- /home/trvehazzk3r/.xsession-errors
{2} -- /home/trvehazzk3r/.xsession-errors
{3} -- /home/trvehazzk3r/.local/share/TelegramDesktop/tdata/working
{4} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/working
{5} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/working
{6} -- socket:[24181]entfd]ocal/share/TelegramDesktop/tdata/working
{7} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/working
{8} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/working
{9} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/working
{10} -- socket:[24183]entfd]ocal/share/TelegramDesktop/tdata/working
{11} -- /home/trvehazzk3r/.local/share/TelegramDesktop/log.txtorking
{12} -- socket:[16380]k3r/.local/share/TelegramDesktop/log.txtorking
{13} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{14} -- socket:[24186]entfd]ocal/share/TelegramDesktop/log.txtorking
{15} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{16} -- socket:[24842]entfd]ocal/share/TelegramDesktop/log.txtorking
{17} -- socket:[23054]entfd]ocal/share/TelegramDesktop/log.txtorking
{18} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{19} -- anon_inode:inotifyd]ocal/share/TelegramDesktop/log.txtorking
{20} -- socket:[23055]tifyd]ocal/share/TelegramDesktop/log.txtorking
{21} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{22} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{23} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{24} -- pipe:[24187]eventfd]ocal/share/TelegramDesktop/log.txtorking
{25} -- pipe:[24187]eventfd]ocal/share/TelegramDesktop/log.txtorking
{26} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{27} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{28} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/log.txtorking
{29} -- /usr/share/mime/mime.cachehare/TelegramDesktop/log.txtorking
{30} -- anon_inode:[eventfd].cachehare/TelegramDesktop/log.txtorking
{31} -- /dev/dri/card0entfd].cachehare/TelegramDesktop/log.txtorking
{32} -- /dev/dri/card0entfd].cachehare/TelegramDesktop/log.txtorking
{33} -- /dev/dri/card0entfd].cachehare/TelegramDesktop/log.txtorking
{34} -- /dev/dri/card0entfd].cachehare/TelegramDesktop/log.txtorking
{35} -- /usr/share/icons/Flat-Remix-Blue-Dark/icon-theme.cacheorking
{36} -- anon_inode:[eventfd]t-Remix-Blue-Dark/icon-theme.cacheorking
{37} -- anon_inode:[eventfd]t-Remix-Blue-Dark/icon-theme.cacheorking
{38} -- socket:[24188]entfd]t-Remix-Blue-Dark/icon-theme.cacheorking
{39} -- /home/trvehazzk3r/.local/share/TelegramDesktop/tdata/user_data/media_cache/0/binlog
{40} -- /home/trvehazzk3r/.local/share/TelegramDesktop/tdata/user_data/cache/0/binlogbinlog
{41} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/user_data/cache/0/binlogbinlog
{42} -- socket:[26633]entfd]ocal/share/TelegramDesktop/tdata/user_data/cache/0/binlogbinlog
{44} -- socket:[25861]entfd]ocal/share/TelegramDesktop/tdata/user_data/cache/0/binlogbinlog
{45} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/user_data/cache/0/binlogbinlog
{49} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/user_data/cache/0/binlogbinlog
{50} -- anon_inode:[eventfd]ocal/share/TelegramDesktop/tdata/user_data/cache/0/binlogbinlog
```

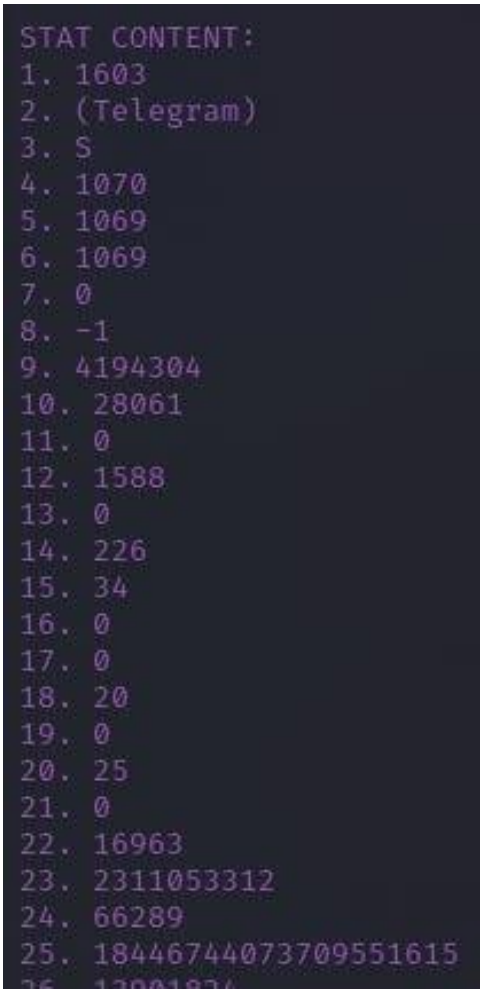
/proc/pid/stat

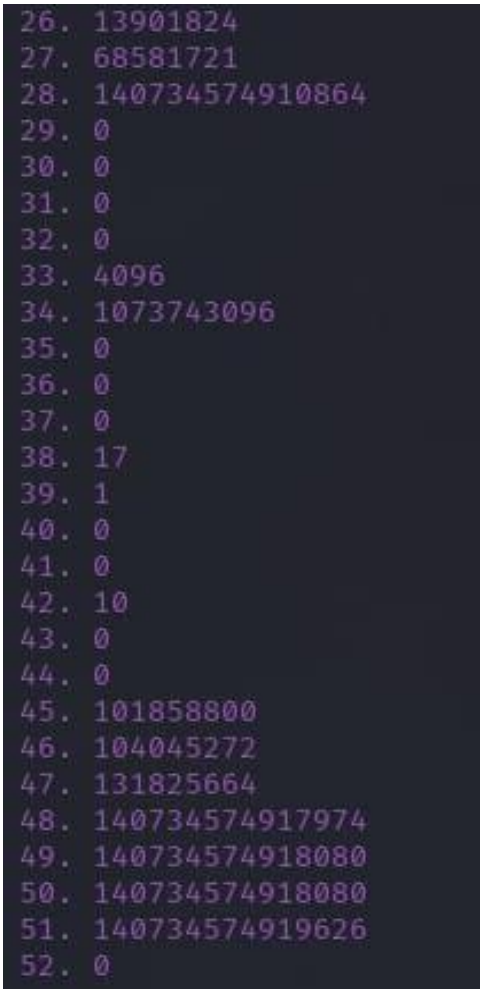
```c
void printSTAT()
{
    char pathToOpen[PATH_MAX];
    snprintf(pathToOpen, PATH_MAX, "/proc/%d/stat", PID);
    char buf[BUFFSIZE];

    FILE *file = fopen(pathToOpen, "r");
    fread(buf, 1, BUFFSIZE, file);
    char *tokens = strtok(buf, " ");

    printf("\nSTAT CONTENT: \n");
```

```
    for (int i = 1; tokens != NULL; i++)
    {
        printf("%d. %s \n", i, tokens);
        tokens = strtok(NULL, " ");
    }

    fclose(file);
}
```

Данный файл содержит в себе статусную информацию о процессе.

Результат выполнения:





Описание содержимого файла:

1) pid - уникальный идентификатор процесса.
2) comm - имя исполняемого файла в круглых скобках.
3) state - состояние процесса.
4) ppid - уникальный идентификатор процесса-предка.
5) pgrp - уникальный идентификатор группы.
6) session - уникальный идентификатор сессии.
7) tty_nr – управляющий терминал.
8) tpgid – уникальный идентификатор группы управляющего терминала.
9) flags – флаги.

10) minflt - Количество незначительных сбоев, которые возникли при выполнении процесса, и которые не требуют загрузки страницы памяти с диска.
11) cminflt - количество незначительных сбоев, которые возникли при ожидании окончания работы процессов-потомков.
12) majflt - количество значительных сбоев, которые возникли при работе процесса, и которые потребовали загрузки страницы памяти с диска.
13) cmajflt - количество значительных сбоев, которые возникли при ожидании окончания работы процессов-потомков.
14) utime - количество тиков, которые данный процесс провел в режиме пользователя.
15) stime - количество тиков, которые данный процесс провел в режиме ядра.
16) cutime - количество тиков, которые процесс, ожидающий завершения процессов-потомков, провёл в режиме пользователя.
17) cstime - количество тиков, которые процесс, ожидающий завершения процессов-потомков, провёл в режиме ядра.
18) priority – 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 - значение для nice в диапазоне от 19 (наиболее низкий приоритет) до -20 (наивысший приоритет).
20) num_threads – число потоков в данном процессе.
21) itrealvalue – 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 - время в тиках запуска процесса после начальной загрузки системы.
23) vsize - размер виртуальной памяти в байтах.
24) rss - 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 - текущий лимит в байтах на резидентный размер процесса. 26) startcode - адрес, выше которого может выполняться код программы.
27) endcode - адрес, ниже которого может выполняться код программ.
28) startstack - адрес начала стека.
29) kstkesp - текущее значение ESP (указателя стека).
30) kstkeip - текущее значение EIP (указатель команд).
31) signal - битовая карта ожидающих сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.
32) blocked - битовая карта блокируемых сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.
33) sigignore - битовая карта игнорируемых сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.
34) sigcatch - битовая карта перехватываемых сигналов. Устарела, потому что не предоставляет информацию о сигналах реального времени, необходимо использовать /proc/[pid]/status.
35) wchan - "канал", в котором ожидает процесс.
36) nswap - количество страниц на своппинге (не обслуживается).
37) cnswap - суммарное nswap для процессов-потомков (не обслуживается).
38) exit_signal - сигнал, который будет послан предку, когда процесс завершится.
39) processor - номер процессора, на котором последний раз выполнялся процесс.

40) rt_priority - приоритет планирования реального времени, число в диапазоне от 1 до 99 для процессов реального времени, 0 для остальных.
41) policy - политика планирования.
42) delayacct_blkio_ticks - суммарные задержки ввода/вывода в тиках.
43) guest_time – гостевое время процесса (время, потраченное на выполнение виртуального процессора на гостевой операционной системе) в тиках.
44) cguest_time - гостевое время для потомков процесса в тиках.
45) start_data - адрес, выше которого размещаются инициализированные и неинициализированные (BSS) данные программы.
46) end_data - адрес, ниже которого размещаются инициализированные и неинициализированные (BSS) данные программы.
47) start_brk - адрес, выше которого куча программы может быть расширена с использованием brk().
48) arg_start - адрес, выше которого размещаются аргументы командной строки (argv).
49) arg_end - адрес, ниже которого размещаются аргументы командной строки (argv).
50) env_start - адрес, выше которого размещается окружение программы.
51) env_end - адрес, ниже которого размещается окружение программы.
52) exit_code – статус завершения потока в форме, возвращаемой waitpid().