|  |  |
| --- | --- |
|  | **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 №1*

*По курсу: «Операционные системы»*

*На тему: «Процессы-демоны»*

Студент ИУ7-66Б

Турсунов Ж.Р.

Преподаватель

Рязанова Н.Ю.

*Москва, 2021 г.*

# Листинг

#include <syslog.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/resource.h>

#include <sys/stat.h> //umask

#include <unistd.h> //setsid

#include <stdio.h> //perror

#include <signal.h> //sidaction

#include <string.h>

#include <errno.h>

#include <sys/file.h>

#define LOCKFILE "/var/run/daemon.pid" //чтобы создавать в этой директории файлы нужны права суперпользователя

#define LOCKMODE (S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH)

int lockfile(int fd)

{

struct flock fl;

fl.l\_type = F\_WRLCK;

fl.l\_start = 0;

fl.l\_whence = SEEK\_SET;

fl.l\_len = 0;

return(fcntl(fd, F\_SETLK, &fl));

}

int already\_running(void)

{

syslog(LOG\_ERR, "Проверка на многократный запуск!");

int fd;

char buf[16];

fd = open(LOCKFILE, O\_RDWR | O\_CREAT, LOCKMODE);

if (fd < 0)

{

syslog(LOG\_ERR, "невозможно открыть %s: %s!", LOCKFILE, strerror(errno));

exit(1);

}

syslog(LOG\_WARNING, "Lock-файл открыт!");

flock(fd, LOCK\_EX | LOCK\_UN);

if (errno == EWOULDBLOCK) {

syslog(LOG\_ERR, "невозможно установить блокировку на %s: %s!", LOCKFILE, strerror(errno));

close(fd);

exit(1);

}

syslog(LOG\_WARNING, "Записываем PID!");

ftruncate(fd, 0);

sprintf(buf, "%ld", (long)getpid());

write(fd, buf, strlen(buf) + 1);

syslog(LOG\_WARNING, "Записали PID!");

return 0;

}

void daemonize(const char \*cmd)

{

int fd0, fd1, fd2;

pid\_t pid;

struct rlimit rl;

struct sigaction sa;

// 1. Сбрасывание маски режима создания файла

umask(0);

// 2. Получение максимального возможного номера дискриптора

if (getrlimit(RLIMIT\_NOFILE, &rl) < 0)

perror("Невозможно получить максимальный номер дискриптора!\n");

// 3. Стать лидером новой сессии, чтобы утратить управляющий терминал

if ((pid = fork()) < 0)

perror("Ошибка функции fork!\n");

else if (pid != 0) //родительский процесс

exit(0);

setsid();

// 4. Обеспечение невозможности обретения терминала в будущем

sa.sa\_handler = SIG\_IGN;

sigemptyset(&sa.sa\_mask);

sa.sa\_flags = 0;

if (sigaction(SIGHUP, &sa, NULL) < 0)

perror("Невозможно игнорировать сигнал SIGHUP!\n");

// if ((pid = fork()) < 0)

// perror("Ошибка функции fork!\n");

// else if (pid != 0) //родительский процесс

// exit(0);

// 5. Назначить корневой каталог текущим рабочим каталогом,

// чтобы впоследствии можно было отмонтировать файловую систему

if (chdir("/") < 0)

perror("Невозможно назначить корневой каталог текущим рабочим каталогом!\n");

// 6. Зактрыть все файловые дескрипторы

if (rl.rlim\_max == RLIM\_INFINITY)

rl.rlim\_max = 1024;

for (int i = 0; i < rl.rlim\_max; i++)

close(i);

// 7. Присоеденить файловые дескрипторы 0, 1, 2 к /dev/null

fd0 = open("/dev/null", O\_RDWR);

fd1 = dup(0); //копируем файловый дискриптор

fd2 = dup(0);

// 8. Инициализировать файл журнала

openlog(cmd, LOG\_CONS, LOG\_DAEMON);

if (fd0 != 0 || fd1 != 1 || fd2 != 2)

{

syslog(LOG\_ERR, "ошибочные файловые дескрипторы %d %d %d\n", fd0, fd1, fd2);

exit(1);

}

syslog(LOG\_WARNING, "Демон запущен!");

}

int main()

{

daemonize("tursunovJr");

// 9. Блокировка файла для одной существующей копии демона

if (already\_running() != 0)

{

syslog(LOG\_ERR, "Демон уже запущен!\n");

exit(1);

}

syslog(LOG\_WARNING, "Проверка пройдена!");

while(1)

{

syslog(LOG\_INFO, "Демон!");

sleep(5);

}

}

# Результат работы программы

# Запуск демона

# **Проверка что демон запустился**

# 

# **Прекращение работы демона**

# 

# **Проверка что демон прекратил свое существование**

# 

# **Вывод syslog**

# 