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 № 1
Тема: программирование демона
Дисциплина: Операционные системы

Студент Лучина Е.Д

Группа ИУ7-61Б

Преподаватель Рязанова Н.Ю.

Москва.

2020 г.

Демоны - это долгоживущие процессы, не имеющие управляющего терминала, то есть работающие в фоновом режиме. Обычно запускаются во время загрузки системы и завершают работу вместе с ней.

Правила программирования демона.

1) Сброс маску режима создания файлов вызовом umask(). Это позволит создавать файлы с любыми правами доступа. Маска наследуется от запускаемого процесса и может маскировать некоторые биты прав доступа.
2) Вызвать функцию fork() и завершить родительский процесс. Таким образом гарантируется, что дочерний процесс не будет является лидером группы, что необходимо для вызова функции setsid().
3) Вызов setsid() - процесс становится лидером новой сессии, лидером новой группы и лишается управляющего терминала.
4) Сделать корневой каталог текущим рабочим каталогом - chdir("/"). Если текущий каталог работы демона находится в подмонтированной файловой системе (например на флешке), то во время работы демона эту файловую систему нельзя будет отмонтировать.
5) Закрыть все ненужные файловые дескрипторы. Это предотвращает удерживание наследованных дескрипторов в открытом состоянии. С помощью функции getlimit() можно определить максимально возможный номер дескриптора и закрыть все вплоть до этого номера.
6) Открыть файловые дескриптеры 0 (stdin), 1 (stdout), 2 (stderr) на устройстве /dev/null. Таким образом работы с этими потоками ввода-вывода не оказывает влияние на запущенного демона.

Для вывода сообщений демона используется функция syslog().

Использование функции alreadyrunning() гарантирует единственность демона (некоторые демоны допускают работу только одной своей копии). Создает в специальной директории файл, куда записывает id процесса, и осуществляет его блокировку. При попытки копии демона выполнить команду alreadyrunning сработает блокировка файла.

Листинг программы, которая создает процесс-демон с помощью функций Demonize, Already running, Main. Демон выводит время запуска и периодическое (каждые 10 секунд) сообщение о времени работы.

```
#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>
#include "time.h"

// в этот файл already_running запишет pid.
#define LOCKFILE "/var/run/daemon.pid"
#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)
{
    int fd;
    char buf[16];

    fd = open(LOCKFILE, O_RDWR | O_CREAT, LOCKMODE);

    if (fd == -1)
    {
        syslog(LOG_ERR, "невозможно открыть %s: %s!", LOCKFILE,
strerror(errno));
        exit(1);
    }

    if (lockfile(fd) == -1)
    {
        if (errno == EACCES || errno == EAGAIN)
        {
            close(fd);
            exit(1);
        }

        syslog(LOG_ERR,
               "невозможно установить блокировку на %s:
%s!\n", LOCKFILE,
               strerror(errno));
        exit(1);
    }

    ftruncate(fd, 0);
    sprintf(buf, "%ld", (long)getpid());
    write(fd, buf, strlen(buf) + 1);
```

```
    return 0;
}
```

```
void daemonize(const char *cmd)
{
    int i, fd0, fd1, fd2;
    pid_t pid;
    struct rlimit rl;
    struct sigaction sa;

    // 1. Сбросить маску режима файла
    umask(0);

    // 2. Получить максимально возможный номер
дискриптора файла
    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()) == -1)
        perror("Ошибка функции fork!\n");
    else if (pid != 0) //родительский процесс
        exit(0);

    setsid();

    // 4. Обеспечить невозможность обретения
терминала в будущем
    // Игнорируем сигнал о потере терминала.
    // SIGHUP говорит, что терминал был потерян и SIGHUP
завершает процесс

    sa.sa_handler = SIG_IGN;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    if (sigaction(SIGHUP, &sa, NULL) == -1)
        perror("Невозможно игнорировать сигнал SIGHUP!\n");

    // 5. Назначить корневой каталог текущим рабочим
каталогом
    if (chdir("/") < 0)
        perror("Невозможно назначить корневой каталог
текущим рабочим каталогом!\n");

    // 6. Закрыть все файловые дескрипторы
    if (rl.rlim_max == RLIM_INFINITY)
        rl.rlim_max = 1024;
    for (i = 0; i < rl.rlim_max; i++)
        close(i);
```

```
    // 7. Присоединить файловые дескрипторы 0, 1, 2 к
/dev/null
    fd0 = open("/dev/null", O_RDWR);
    fd1 = dup(0);
    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);
    }
}
```

```
int main()
{
    time_t now;
    struct tm *ptr;

    now = time(NULL);
    ptr = localtime(&now);
    // процесс становится демоном
    daemonize("my_daemon");
    syslog(LOG_INFO,
          "Демон запущен: %d:%d:%d", ptr->tm_hour, ptr->tm_min,
          ptr->tm_sec);

    if (already_running())
        exit(1);

    while(1)
    {
        now = time(NULL);
        ptr = localtime(&now);
        syslog(LOG_INFO,
               "Текущее время: %d:%d:%d", ptr->tm_hour, ptr->tm_min,
               ptr->tm_sec);
        sleep(10);
    }
}
```

Работа программы.

Скомпилируем (gcc main.c) и запустим программу с привилегиями суперюзера (sudo ./a.out).

Команда ps -ajx.

- a :процессы, связанные с текущим терминалом, а также процессы других пользователей;
- x : процессы, отсоединенные от терминала.
- J: напечатать следующую информацию: user, pid, ppid, pgid, sess, jobc, state, tt, time, and command.

Запущенный нами демон имеет id=1558, id группы и сессии равно 1558 (является лидером группы и сессии), в качестве родителя терминальный процесс (ppid = 1). Как и ожидалось, он не не имеет управляющего терминала (TTY = ?). Состояние Ss означает следующее

- S - interruptible sleep (waiting for an event to complete) (прерываемый сон)
- s - is a session leader (лидер сессии)

```
myOSlabs/lab_01 $ gcc main.c
myOSlabs/lab_01 $ sudo ./a.out
myOSlabs/lab_01 $ ps -ajx
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
    0     1     1     1 ?           0 Ssl      0   0:00 /init ro
    1    76    76    57 ?           0 T     1000   5:11 /usr/bin/python3.7 /
    1   313   313   313 tty1        0 Ss       0   0:00 /init ro
  313   314   314   313 tty1        0 S     1000   0:05 -zsh
    1   562   562   562 tty3        0 Ss       0   0:00 /init ro
  562   563   563   562 tty3        0 S     1000   0:00 sh -c "$VSCODE_WSL_E
  563   564   563   562 tty3        0 S     1000   0:00 sh /mnt/c/Users/hell
  564   569   563   562 tty3        0 S     1000   0:00 sh /home/Lena/.vscod
  569   571   563   562 tty3        0 Sl    1000   0:02 /home/Lena/.vscode-s
  571   607   563   562 tty3        0 Sl    1000   0:05 /home/Lena/.vscode-s
  607   620   620   620 pts/0       0 Ss    1000   0:01 /usr/bin/zsh
    1   826   826   826 tty4        0 Ss       0   0:00 /init ro
  826   827   827   826 tty4        0 S     1000   0:02 -zsh
    1  1469  1469  1469 ?           0 Ssl    104   0:00 /usr/sbin/rsyslogd
    1  1490  1490  1490 tty5        0 Ss       0   0:00 /init ro
 1490  1491  1491  1490 tty5        0 S     1000   0:01 -zsh
    1  1558  1558  1558 ?           0 Ss       0   0:00 ./a.out
  314  1559  1559   313 tty1        0 R     1000   0:00 ps -ajx
myOSlabs/lab_01 $ sudo kill 1558
```

cat/var/log/syslog:

```
Apr 15 18:37:02 LenaPC my_daemon: Демон запущен: 18:37:2
Apr 15 18:37:02 LenaPC my_daemon: Текущее время: 18:37:2
Apr 15 18:37:12 LenaPC my_daemon: Текущее время: 18:37:12
Apr 15 18:37:22 LenaPC my_daemon: Текущее время: 18:37:22
Apr 15 18:37:32 LenaPC my_daemon: Текущее время: 18:37:32
Apr 15 18:37:42 LenaPC my_daemon: Текущее время: 18:37:42
Apr 15 18:37:52 LenaPC my_daemon: Текущее время: 18:37:52
Apr 15 18:38:02 LenaPC my_daemon: Текущее время: 18:38:2
Apr 15 18:38:12 LenaPC my_daemon: Текущее время: 18:38:12
Apr 15 18:38:22 LenaPC my_daemon: Текущее время: 18:38:22
Apr 15 18:38:32 LenaPC my_daemon: Текущее время: 18:38:32
Apr 15 18:38:42 LenaPC my_daemon: Текущее время: 18:38:42
Apr 15 18:38:52 LenaPC my_daemon: Текущее время: 18:38:52
```