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

Дисциплина Операционные системы

Тема Процессы-демоны

Студент Набиев Ф.М.
Группа ИУ7–63Б
Оценка (баллы)
Преподаватель Рязанова Н.Ю.

Москва, 2020 г.

1 Задание

1.1 Условие

Запрограммировать демона и проанализировать информацию об этом процессе.

1.2 Реализация

```
1  #include <sys/resource.h>
2  #include <sys/types.h>
3  #include <sys/file.h>
4  #include <sys/stat.h>
5
6  #include <unistd.h>
7  #include <signal.h>
8  #include <syslog.h>
9  #include <fcntl.h>
10
11 #include <stdlib.h>
12 #include <string.h>
13 #include <stdio.h>
14 #include <errno.h>
15 #include <time.h>
16
17 #define LOCK_FILE "/var/run/daemon.pid"
18 #define LOCK_MODE (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)
19
20 int already_running(void);
21 void daemonize(const char *cmd);
22
23 int main(int argc, const char **argv)
24 {
25     daemonize(argv[0]);
26
27     if (already_running())
28     {
29         syslog(LOG_ERR, "Daemon is already running");
30         exit(1);
31     }
32
33     time_t time_var;
34
35     for (;;)
```

```
36      {
37          time_var = time(NULL);
38          syslog(LOG_INFO, "Time: %s\n", ctime(&time_var));
39
40          sleep(5);
41      }
42  }
43
44  void daemonize(const char *cmd)
45  {
46      // 1 правило:
47      // для возможности создания файлов с любыми правами доступа
48      // сброс маски режима создания файла
49      umask(0);
50
51      pid_t pid;
52      struct rlimit rl;
53      struct sigaction sa;
54
55      // Получить максимально возможный номер дескриптора файла
56      if (getrlimit(RLIMIT_NOFILE, &rl) < 0)
57      {
58          fprintf(stderr, "%s: getrlimit, %s\n", cmd, strerror(errno));
59          exit(1);
60      }
61
62      // 2 правило:
63      // создание дочернего процесса и завершение процесса-предка
64      if ((pid = fork()) < 0)
65      {
66          fprintf(stderr, "%s: fork, %s\n", cmd, strerror(errno));
67          exit(1);
68      }
69      else if (pid != 0)
70      {
71          exit(0);
72      }
73
74      // 3 правило:
75      // обеспечение невозможности обретения
76      // управляющего терминала в будущем
77      setsid();
78
79      sa.sa_handler = SIG_IGN;
80      sigemptyset(&sa.sa_mask);
81      sa.sa_flags = 0;
82
83      if (sigaction(SIGHUP, &sa, NULL) < 0)
```

```
84      {
85          fprintf(stderr, "%s: can not ignore SIGHUP, %s\n",
86                  cmd, strerror(errno));
87          exit(1);
88      }
89
90      // 4 правило:
91      // Назначение корневого каталога текущим рабочим каталогом
92      // (на случай, если демон был запущен с  подмонтированной ФС)
93      if (chdir("/") < 0)
94      {
95          fprintf(stderr, "%s: can not cd to \"/\", %s\n",
96                  cmd, strerror(errno));
97          exit(1);
98      }
99
100     // 5 правило:
101     // Закрытие всех открытх файловых дескрипторов
102     if (rl.rlim_max == RLIM_INFINITY)
103     {
104         rl.rlim_max = 1024;
105     }
106
107     for (int i = 0; i < rl.rlim_max; ++i)
108     {
109         close(i);
110     }
111
112     // 6 правило:
113     // Присоединение файловых дескрипторов 1, 2, 3 к /deb/null
114     int fd0 = open("/dev/null", O_RDWR);
115     int fd1 = dup(0);
116     int fd2 = dup(0);
117
118     // Инициализация файла журнала
119     openlog(cmd, LOG_CONS, LOG_DAEMON);
120
121     if (fd0 != 0 || fd1 != 1 || fd2 != 2)
122     {
123         syslog(LOG_ERR, "Wrong file descriptors %d %d %d", fd0, fd1, fd2);
124         exit(1);
125     }
126 }
127
128 int already_running(void)
129 {
130     int fd;
131     char buf[256];
```

```
132
133     fd = open(LOCK_FILE, O_RDWR | O_CREAT, LOCK_MODE);
134
135     if (fd < 0)
136     {
137         syslog(LOG_ERR, "Can not be open %s: %s",
138                LOCK_FILE, strerror(errno));
139         exit(1);
140     }
141
142     // Если файл уже заблокирован
143     if (flock(fd, LOCK_EX | LOCK_NB) < 0)
144     {
145         if (errno == EWOULDBLOCK)
146         {
147             close(fd);
148             return 1;
149         }
150
151         syslog(LOG_ERR, "Can not block %s: %s",
152                LOCK_FILE, strerror(errno));
153         exit(1);
154     }
155
156     ftruncate(fd, 0);
157     sprintf(buf, "%d", getpid());
158     write(fd, buf, strlen(buf) + 1);
159
160     return 0;
161 }
```

1.3 Анализ информации о процессе-демоне

Запустим программу с правами супер-пользователя и выведем информацию о процессах в системе, используя команду ps с ключами —a (вывод процессов, которыми владеют другие пользователи), —j (вывод идентификаторов сессии, группы процессов, управляющего терминала и группы процессов терминала) и —x (вывод процессов, не имеющих управляющего терминала).

Рис. 1.1

```
 ~/Documents/Repositories/bmstu/OperatingSystems/lab_11  master ! ?
# gcc mydaemon.c

 ~/Documents/Repositories/bmstu/OperatingSystems/lab_11  master ! ?
# sudo ./a.out

 ~/Documents/Repositories/bmstu/OperatingSystems/lab_11  master ! ?
# ps -ajx
PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
   0     1     1     1 ?           -1 Ss       0   1:28 /sbin/init
   0     2     0     0 ?           -1 S        0   0:00 [kthreadd]
   2     3     0     0 ?           -1 I<       0   0:00 [rcu_gp]
   2     4     0     0 ?           -1 I<       0   0:00 [rcu_par_gp]
```

Колонки, слева направо: идентификатор родительского процесса, идентификатор процесса, идентификатор группы процессов, идентификатор сессии, имя терминала, идентификатор группы процессов терминала, состояние процесса, идентификатор пользователя, совокупное время использования процессора для конкретного процесса и строка команды.

Рис. 1.2

```
    1  1453  1453  1453 ?           -1 Ssl    115   0:00 /usr/lib/colord/colord
    2  1491     0     0 ?           -1 I        0   0:00 [kworker/7:0]
    1  1529  1529  1529 ?           -1 Ss       0   0:00 ./a.out
24374  1538  1538 24374 pts/1     1538 R+    1000   0:00 ps -ajx
    1  1722  1722  1722 ?           -1 Ss     120   0:00 /usr/sbin/exim4 -bd -q30
 1021  1841  1021  1021 ?           -1 Sl       0   0:00 gdm-session-worker [pam]
```

Процесс демон имеет идентификатор 1529. Кроме того, у него совпадают идентификаторы процесса, лидера сессии, лидера группы, потому что процесс-демон является лидером группы и лидером сессии, единственным в своей группе и сессии. В колонке имени терминала стоит знак вопроса, что означает, у процесса-демона нет управляющего терминала. Состояние процесса — Ss. S означает прерываемый сон, а s — процесс является лидером сессии.

Процесс-демон не может выводить сообщения на стандартное устройство вывода сообщений об ошибках, так как он не имеет управляющего терминала. Поэтому для вывода сообщений используется механизм syslog. Эта функция отправляет сообщения через сокет домена UNIX — /dev/log.

Используя команду sudo tail -f /var/log/syslog, получим последнее содержимое syslog. Затем попробуем запустить демона еще раз —

в syslog выведется предупреждение, что этот демон уже запущен (так как он должен быть единственным в своей группе и сессии).

Рис. 1.3

```
 ~/Documents/Repositories/bmstu/OperatingSystems/lab_11  master ! ?
# sudo tail -f /var/log/syslog
Jun 29 12:20:44 rfl ./a.out: Time: Mon Jun 29 12:20:44 2020#012
Jun 29 12:20:49 rfl ./a.out: Time: Mon Jun 29 12:20:49 2020#012
Jun 29 12:20:54 rfl ./a.out: Time: Mon Jun 29 12:20:54 2020#012
Jun 29 12:20:59 rfl ./a.out: Time: Mon Jun 29 12:20:59 2020#012
Jun 29 12:21:04 rfl ./a.out: Time: Mon Jun 29 12:21:04 2020#012
Jun 29 12:21:09 rfl ./a.out: Time: Mon Jun 29 12:21:09 2020#012
Jun 29 12:21:14 rfl ./a.out: Time: Mon Jun 29 12:21:14 2020#012
Jun 29 12:21:19 rfl ./a.out: Time: Mon Jun 29 12:21:19 2020#012
Jun 29 12:21:24 rfl ./a.out: Time: Mon Jun 29 12:21:24 2020#012
Jun 29 12:21:29 rfl ./a.out: Time: Mon Jun 29 12:21:29 2020#012
Jun 29 12:21:34 rfl ./a.out: Time: Mon Jun 29 12:21:34 2020#012
Jun 29 12:21:34 rfl ./a.out: Daemon is already running
Jun 29 12:21:39 rfl ./a.out: Time: Mon Jun 29 12:21:39 2020#012
Jun 29 12:21:44 rfl ./a.out: Time: Mon Jun 29 12:21:44 2020#012
Jun 29 12:21:49 rfl ./a.out: Time: Mon Jun 29 12:21:49 2020#012
```

Убьем демона при помощи команды sudo kill и увидим, что он пропал из списка процессов:

Рис. 1.4

```
 ~/Documents/Repositories/bmstu/OperatingSystems/lab_11  master ! ?
# sudo kill 1529

 ~/Documents/Repositories/bmstu/OperatingSystems/lab_11  master ! ?
# ps -ajx | grep a.out
24374  2611  2610 24374 pts/1     2610 S+    1000   0:00 grep --color=auto --ex
clude-dir=.hg --exclude-dir=.svn --exclude-dir=.idea --exclude-dir=.tox a.out
```