



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 ИУ, Информатика и системы управления

КАФЕДРА ИУ7, Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ

к лабораторной работе №1

по курсу

“Операционные системы”
На тему: “Демоны”

Студент ИУ7-64Б
(Группа)

(Подпись, дата) **А.А. Андреев**
(И.О.Фамилия)

Преподаватель

(Подпись, дата) **Н.Ю. Рязанова**
(И.О.Фамилия)

2022 г.

Оглавление

Задание	3
Реализация	4
Результат работы программы	7

Задание

1. Структурировать исходный код программы в листинге 4.7.
2. Изменить программу так, чтобы она выводила на экран дерево каталогов в наглядной форме при помощи символов *, ../, | и тд.
3. Изменить функцию myfwt() так, чтобы она передавала функции lstat() не полный путь к файлу, а только его имя каждый раз, когда встречается каталог.

Реализация

В листингах 1-5 приведен код программы.

Листинг 1: Реализация программы, Часть 1

```
1. #include <errno.h>
2. #include <fcntl.h>
3. #include <signal.h>
4. #include <stdarg.h>
5. #include <stdio.h>
6. #include <stdlib.h>
7. #include <string.h>
8. #include <syslog.h>
9. #include <sys/resource.h>
10. #include <sys/stat.h>
11. #include <time.h>
12. #include <unistd.h>
13.
14. #define SLEEP_TIME 5
15.
16. /*
17.  * ps -ajx
18.  *
19.  * Демон — лидер группы и сессии, не имеет управляющего
    терминала.
20.  * Пользователь не должен влиять на него из командной строки.
21.  *
22.  * PROCESS STATE CODES:
23.  * D — uninterruptible sleep (usually IO)
24.  * I — Idle kernel thread
25.  * R — running or runnable (on run queue)
26.  * S — interruptible sleep (waiting for an event to complete)
27.  * T — stopped by job control signal
28.  * t — stopped by debugger during the tracing
29.  * W — paging (not valid since the 2.6.xx kernel)
30.  * X — dead (should never be seen)
31.  * Z — defunct ("zombie") process, terminated but not reaped
    by its parent
32.  *
33.  * Какой сон можно прервать?
34.  * Если процесс блокирован на каком-то событии, такой сон
    можно прервать.
35.  * (Нельзя прервать, если процесс блокирован на в/в)
36.  */
37.
38. void syslog_quit(const char *prompt) {
39.     syslog(LOG_ERR, "Unable to %s: %m", prompt);
40.     exit(EXIT_FAILURE);
41. }
42.
43. void fsyslog_quit(const char *format, ...) {
44.     char prompt[256];
```

Листинг 2: Реализация программы, Часть 2

```
45.     va_list ap;
46.     va_start(ap, format);
47.     vsnprintf(prompt, sizeof prompt, format, ap);
48.     va_end(ap);
49.
50.     syslog_quit(prompt);
51. }
52.
53. #define LOCKFILE "/var/run/daemon.pid"
54. #define LOCKMODE (S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH)
55. // S_IRUSR – владелец может читать
56. // S_IWUSR – владелец может писать
57. // S_IRGRP – группа-владелец может читать
58. // S_IROTH – все остальные могут читать
59.
60. int lockfile(int fd) {
61.     struct flock fl = {
62.         .l_type = F_WRLCK,           // Режим блокировки (F_WRLCK
        – блокировка записи)
63.         .l_start = 0,               // Относительное смещение в
        байтах, зависит от l_whence
64.         .l_whence = SEEK_SET,       // Считать смещение от начала
        файла
65.         .l_len = 0                   // Длина блокируемой области
        в байтах (0 – до конца файла)
66.     };
67.     return fcntl(fd, F_SETLK, &fl); // Установить блокировку
68. }
69.
70. int already_running(void) {
71.     const int fd = open(LOCKFILE, O_RDWR|O_CREAT, LOCKMODE);
72.     if (fd == -1) {
73.         fsyslog_quit("open %s", LOCKFILE);
74.     }
75.     if (lockfile(fd) == -1) {
76.         if (errno == EACCES || errno == EAGAIN) {
77.             close(fd);
78.             return EXIT_FAILURE;
79.         }
80.         fsyslog_quit("lock %s", LOCKFILE);
81.     }
82.     char buf[16];
83.     ftruncate(fd, 0);
84.     sprintf(buf, "%ld", (long) getpid());
85.     write(fd, buf, strlen(buf) + 1);
86.     return EXIT_SUCCESS;
87. }
88.
89. void daemonize(const char *cmd) {
90.     // Инициализировать файл журнала
91.     // (ALERT: У Раго это сделано в конце функции)
92.     openlog(cmd, LOG_CONS, LOG_DAEMON);
```

Листинг 3: Реализация программы, Часть 3

```
93.      // 1. Сбросить маску режима создания файлов. Маска
        наследуется и может
94.      // маскировать некоторые биты прав доступа.
95.      umask(0);
96.
97.      // Получить максимально возможный номер дескриптора файла.
98.      // (ALERT: Ещё раз, ничего перемещать нельзя, Н. Ю. Банит)
99.      struct rlimit rl;
100.     if (getrlimit(RLIMIT_NOFILE, &rl) == -1) {
101.         syslog_quit("getrlimit");
102.     }
103.
104.     // 2. Вызвать функцию fork и завершить родительский
        процесс. Этим самым
105.     // мы гарантируем, что дочерний процесс не будет являться
        лидером
106.     // группы, а это необходимое условие для вызова функции
        setsid
107.     const pid_t pid = fork();
108.     if (pid == -1) {
109.         syslog_quit("fork");
110.     } else if (pid != 0) {
111.         exit(EXIT_SUCCESS);
112.     }
113.
114.     // Обеспечить невозможность обретения управляющего
        терминала в будущем.
115.     // SIGHUP – сигнал, посылаемый процессу для уведомления о
        потере
116.     // соединения с управляющим терминалом пользователя.
117.     struct sigaction sa;
118.     sa.sa_handler = SIG_IGN;
119.     sigemptyset(&sa.sa_mask);
120.     sa.sa_flags = 0;
121.     if (sigaction(SIGHUP, &sa, NULL) == -1) {
122.         syslog_quit("ignore SIGHUP");
123.     }
124.
125.     // 3. Создать новую сессию, при этом процесс становится
        (а) лидером
126.     // новой сессии (б) лидером новой группы процессов и (в)
        лишается
127.     // управляющего терминала
128.     if (setsid() == -1) {
129.         syslog_quit("setsid");
130.     }
131.
132.     // 4. Сделать корневой каталог текущим рабочим каталогом
133.     if (chdir("/") == -1) {
134.         syslog_quit("chdir");
135.     }
```

Листинг 4: Реализация программы, Часть 4

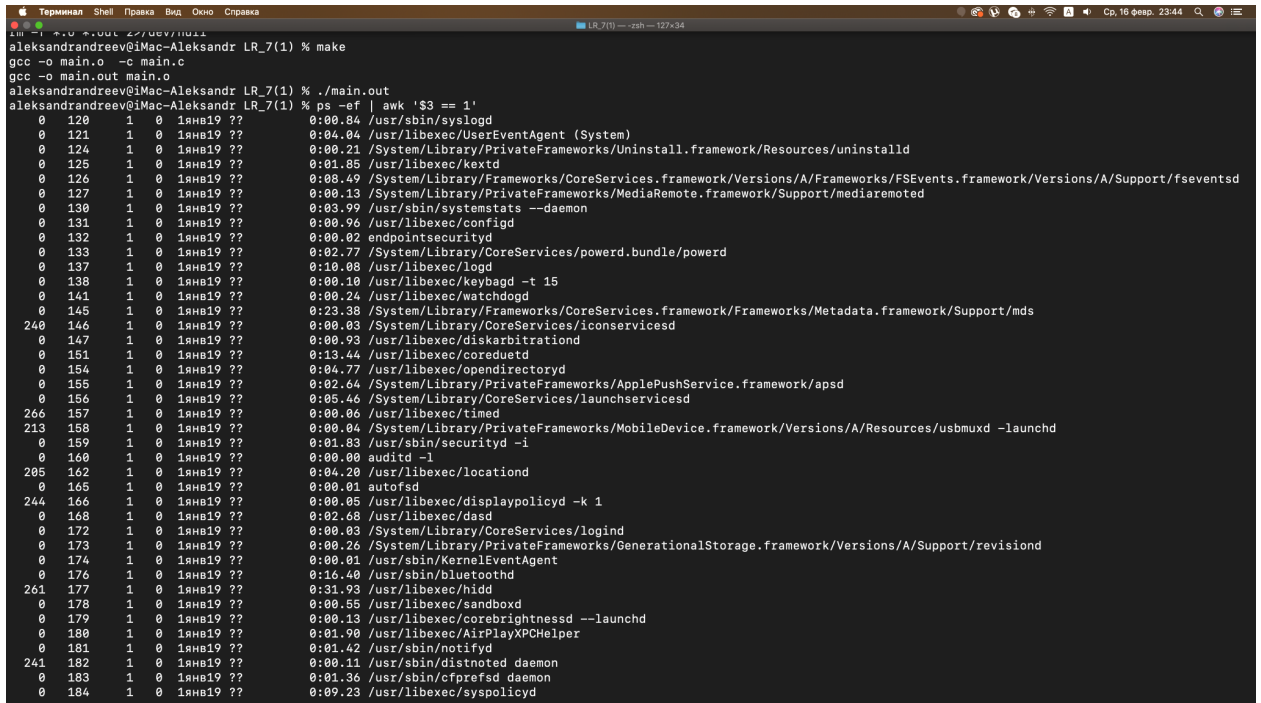
```
136. // 5. Закрыть все открытые файловые дескрипторы. Это ненужные
    процессу-
137. // демону файловые дескрипторы, закрытию которых он может
    препятствовать
138. //
139. // The value RLIM_INFINITY, defined in <sys/resource.h>,
    is considered
140. // to be larger than any other limit value. If a call to
    getrlimit()
141. // returns RLIM_INFINITY for a resource, it means the
    implementation
142. // does not enforce limits on that resource. Specifying
    RLIM_INFINITY as
143. // any resource limit value on a successful call to
    setrlimit() inhibits
144. // enforcement of that resource limit.
145. //
    (https://pubs.opengroup.org/onlinepubs/7908799/xsh/getrlimit.ht
    ml)
146. if (rl.rlim_max == RLIM_INFINITY) {
147.     rlim_max = 1024;
148. }
149. for (rlim_t fd = 0; fd < rlim_max; ++fd) {
150.     close(fd);
151. }
152.
153. // 6. Присоединить файловые дескрипторы 0, 1 и 2 к
    /dev/null.
154. //
155. // Для того, чтобы можно было использовать функции
    стандартных библиотек
156. // и они не выдавали ошибки.
157. if (open("/dev/null", O_RDWR) != 0) {
158.     syslog_quit("open /dev/null");
159. }
160. (void) dup(0);
161. (void) dup(1);
162. }
163.
164. int main(void) {
165.     daemonize("DAEMON");
166.
167.     if (already_running() != EXIT_SUCCESS) {
168.         syslog(LOG_ERR, "ALREADY RUNNING");
169.         exit(EXIT_FAILURE);
170.     }
171.
172.     time_t t = time(NULL);
173.     syslog(LOG_WARNING, "STARTS %s",
        asctime(localtime(&t)));
174.
```

Листинг 5: Реализация программы, Часть 5

```
175.     for (;;) {  
176.         t = time(NULL);  
177.         syslog(LOG_INFO, "current time is %s",  
178.             asctime(localtime(&t)));  
178.         sleep(SLEEP_TIME);  
179.     }  
180. }
```


Результат работы программы

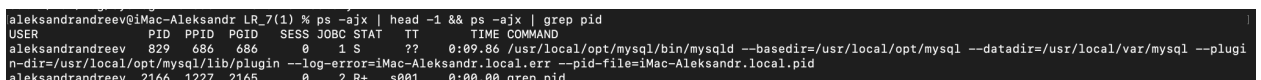
На Рисунке 1 приведен просмотр всех демонов при помощи команды “*ps -ef | awk '\$3 == 1'*”.



```
aleksandrareev@iMac-Aleksandr LR_7(1) % make
gcc -o main.o -c main.c
gcc -o main.out main.o
aleksandrareev@iMac-Aleksandr LR_7(1) % ./main.out
aleksandrareev@iMac-Aleksandr LR_7(1) % ps -ef | awk '$3 == 1'
0 120 1 0 1яна19 ?? 0:00.04 /usr/sbin/syslogd
0 121 1 0 1яна19 ?? 0:04.04 /usr/libexec/UserEventAgent (System)
0 124 1 0 1яна19 ?? 0:00.21 /System/Library/PrivateFrameworks/Uninstall.framework/Resources/uninstalld
0 125 1 0 1яна19 ?? 0:01.85 /usr/libexec/kextd
0 126 1 0 1яна19 ?? 0:08.49 /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/FSEvents.framework/Versions/A/Support/fsevents
0 127 1 0 1яна19 ?? 0:00.13 /System/Library/PrivateFrameworks/MediaRemote.framework/Support/mediaremoted
0 130 1 0 1яна19 ?? 0:03.99 /usr/sbin/systemstats --daemon
0 131 1 0 1яна19 ?? 0:00.96 /usr/libexec/configd
0 132 1 0 1яна19 ?? 0:00.02 endpointsecurityd
0 133 1 0 1яна19 ?? 0:02.77 /System/Library/CoreServices/powerd.bundle/powerd
0 137 1 0 1яна19 ?? 0:10.08 /usr/libexec/logd
0 138 1 0 1яна19 ?? 0:00.18 /usr/libexec/keybagd -t 15
0 141 1 0 1яна19 ?? 0:00.24 /usr/libexec/watchdogd
0 145 1 0 1яна19 ?? 0:23.38 /System/Library/Frameworks/CoreServices.framework/Frameworks/Metadata.framework/Support/mds
240 146 1 0 1яна19 ?? 0:00.03 /System/Library/CoreServices/iconservicesd
0 147 1 0 1яна19 ?? 0:00.93 /usr/libexec/diskarbitrationd
0 151 1 0 1яна19 ?? 0:13.44 /usr/libexec/coreduetd
0 154 1 0 1яна19 ?? 0:04.77 /usr/libexec/opendirectoryd
0 155 1 0 1яна19 ?? 0:02.64 /System/Library/PrivateFrameworks/ApplePushService.framework/apsd
0 156 1 0 1яна19 ?? 0:05.46 /System/Library/CoreServices/launchservicesd
266 157 1 0 1яна19 ?? 0:00.06 /usr/libexec/timed
213 158 1 0 1яна19 ?? 0:00.04 /System/Library/PrivateFrameworks/MobileDevice.framework/Versions/A/Resources/usbmuxd --launchd
0 159 1 0 1яна19 ?? 0:01.83 /usr/sbin/securityd -i
0 160 1 0 1яна19 ?? 0:00.00 auditd -l
205 162 1 0 1яна19 ?? 0:04.20 /usr/libexec/locationd
0 165 1 0 1яна19 ?? 0:00.01 autofsd
244 166 1 0 1яна19 ?? 0:00.05 /usr/libexec/displaypolicyd -k 1
0 168 1 0 1яна19 ?? 0:02.68 /usr/libexec/dasd
0 172 1 0 1яна19 ?? 0:00.03 /System/Library/CoreServices/logind
0 173 1 0 1яна19 ?? 0:00.26 /System/Library/PrivateFrameworks/GenerationalStorage.framework/Versions/A/Support/revisioid
0 174 1 0 1яна19 ?? 0:00.01 /usr/sbin/KernelEventAgent
0 176 1 0 1яна19 ?? 0:16.40 /usr/sbin/bluetoothd
261 177 1 0 1яна19 ?? 0:31.93 /usr/libexec/hidd
0 178 1 0 1яна19 ?? 0:00.55 /usr/libexec/sandboxd
0 179 1 0 1яна19 ?? 0:00.13 /usr/libexec/corebrightnessd --launchd
0 180 1 0 1яна19 ?? 0:01.90 /usr/libexec/AirPlayXPCHelper
0 181 1 0 1яна19 ?? 0:01.42 /usr/sbin/notifyd
241 182 1 0 1яна19 ?? 0:00.11 /usr/sbin/distnoted daemon
0 183 1 0 1яна19 ?? 0:01.36 /usr/sbin/cfprefsd daemon
0 184 1 0 1яна19 ?? 0:09.23 /usr/libexec/syspolicyd
```

Рисунок 1: Просмотр всех демонов

На Рисунке 1 приведен вывод при помощи команды “*ps -ajx | head -1 && ps -ajx | grep pid*”.



```
aleksandrareev@iMac-Aleksandr LR_7(1) % ps -ajx | head -1 && ps -ajx | grep pid
USER      PID  PPID  PGID  SESS  JOBC  STAT  TT    TIME COMMAND
aleksandrareev 829 686 686 0 1 S ?? 0:09.86 /usr/local/opt/mysql/bin/mysqld --basedir=/usr/local/opt/mysql --datadir=/usr/local/var/mysql --plugi
n-dir=/usr/local/opt/mysql/lib/plugin --log-error=iMac-Aleksandr.local.err --pid-file=iMac-Aleksandr.local.pid
aleksandrareev 2166 1227 2165 0 2 R+ s001 0:00.00 grep pid
```

Рисунок 2: Вывод с pid