

# PROGRAMMING ASSIGNMENT № 6

---

Steven Seppala Talia Garcia

13 November 2014

## Programming assignment six

### 1. File Suffix

---

```
1 #include <stdio.h>
2 #include <dirent.h>
3 #include <string.h>
4
5 int main(int argc, char *argv[])
6 {
7     struct dirent
8         *directory_entry;
9
10    DIR *directory_to_open;
11
12    if(argc<3)
13    {
14        fprintf(stderr, "ERROR\r\nUsage: ./PROGRAM_NAME <directory_name> ←
15                         <.extention>\n");
16        return 1;
17    }
18
19    directory_to_open = opendir (argv[1]);
20    if(directory_to_open == NULL)
21    {
22        fprintf(stderr, "ERROR\r\nCannot open directory '%s'\nDoes it ←
23                         exist?\r\n", argv[1]);
24        return 1;
25    }
26
27    while ((directory_entry = readdir(directory_to_open)) != NULL)
28    {
29        if(strstr(directory_entry->d_name, argv[2])!=NULL)
30        {
31            printf("\t%s\r\n", directory_entry -> d_name);
32        }
33    }
34    closedir(directory_to_open);
35    return 0;
```

34

35 }



## 2. Daemon

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <signal.h>
5 #include <fcntl.h>
6 #include <time.h>
7 #include <sys/types.h>
8 #include <sys/stat.h>
9 #include <syslog.h>
10
11 long daemon_init(char* logfile_to_write_to, char* file_to_monitor,←
12   char* daemon_name)
13 {
14     if(!daemon_name)
15         daemon_name = "TEST_DAEMON";
16     if(!file_to_monitor)
17         exit(-1);
18     if(!logfile_to_write_to)
19         exit(-1);
20
21     pid_t pid;
22     /* Fork off the parent process */
23     pid = fork();
24
25     /* An error occurred */
26     if (pid < 0)
27         exit(-1);
28     /* Success: Let the parent terminate */
29     if (pid > 0)
30         exit(EXIT_SUCCESS);
31     /* On success: The child process becomes session leader */
32     if (setsid() < 0)
33         exit(-1);
34     /* Catch, ignore and handle signals */

```

```

34     //TODO: Implement a working signal handler */
35     signal(SIGCHLD, SIG_IGN);
36     signal(SIGHUP, SIG_IGN);
37
38     /* Fork off for the second time*/
39     pid = fork();
40
41     /* An error occurred */
42     if (pid < 0)
43         exit(-1);
44     /* Success: Let the parent terminate */
45     if (pid > 0)
46         exit(EXIT_SUCCESS);
47
48     /* Set new file permissions */
49     umask(0);
50
51     printf("File to monitor: %s\n",file_to_monitor);
52     int new_in = open(file_to_monitor, O_CREAT | O_RDONLY);
53     dup2(new_in, 0);
54     close(new_in);
55
56     if((freopen(logfile_to_write_to,"w+", stdout))==NULL)
57         return -1;           //fd=1
58
59     /* Open the log file */
60     openlog(daemon_name, LOG_PID, LOG_DAEMON);
61
62     printf("File to monitor: %s\n",file_to_monitor);
63
64     fseek(stdin, 0, SEEK_END);
65     long starting_file_size = ftell(stdin) - 1;
66
67
68     return(starting_file_size);
69 }
70
71 /*#####
72 *#Time Print Function #
73 *#####*/
74
75 long Print_String(char* file_name, long previous_size)
76 {
77     /* This function gets the time of day
78      * and prints out a file that has been passed
79      * to it. Then it measures how big the file is
80      * in bytes and prints out the two. */

```

```

81     time_t rawtime;
82     struct tm * timeinfo;
83     time (&rawtime);
84     timeinfo = localtime (&rawtime);
85
86     long file_size;
87     printf( "Monitoring file:: %s at %s", file_name, asctime(timeinfo)←
88             );
88     fseek(stdin, 0, SEEK_END);
89     file_size = ftell(stdin) - 1;
90     printf("Size is :: %ld ||", file_size);
91     printf("Previous size was :: %ld\n", previous_size);
92
93     return(file_size);
94 }
95
96
97 int main(int argc, char* argv[])
98 {
99     long file_size;
100    if((file_size=(daemon_init(argv[1],argv[2],argv[3])))== -1)
101    {
102        fprintf(stderr, "Error setting up daemon.\r\n"
103                  "Please run with <Log_File> <File_To_Watch> {OPTIONAL}←
104                  <Daemon_Name>\r\n");
105        return EXIT_FAILURE;
106    }
107    for(int execute_time = 0; execute_time <= 180; execute_time++)
108    {
109        if((execute_time%10)==0)
110            file_size = Print_String(argv[2], file_size);
111            fflush(stdout);
112            sleep(1);
113    }
114    printf("\nDone monitoring file, goodbye.\n");
115    syslog (LOG_NOTICE, "DAEMON_TEST_KILLED");
116    closelog();
117
118    return EXIT_SUCCESS;
119 }
```

---

```
16:23:40 @ - - -
clang tns_daemon.c
16:31:01 @ - - -
./a.out size.txt size.log
File to monitor: size.log
16:31:18 @ - - -
./a.out size.log size.txt
File to monitor: size.txt
16:31:33 @ - - -
cat size.log
File to monitor: size.txt
Monitoring file:: size.txt at Fri Nov 14 16:31:33 2014
Size is :: 215 ||Previous size was :: 215
16:31:41 @ - - -
cat size.txt
File to monitor: size.log
Monitoring file:: size.log at Fri Nov 14 16:31:18 2014
Size is :: 45 ||Previous size was :: 45
Monitoring file:: size.log at Fri Nov 14 16:31:28 2014
Size is :: 45 ||Previous size was :: 45
Monitoring file:: size.log at Fri Nov 14 16:31:38 2014
Size is :: 122 ||Previous size was :: 45
16:31:46 @ - - -
cat size.log
File to monitor: size.txt
Monitoring file:: size.txt at Fri Nov 14 16:31:33 2014
Size is :: 215 ||Previous size was :: 215
Monitoring file:: size.txt at Fri Nov 14 16:31:43 2014
Size is :: 311 ||Previous size was :: 215
Monitoring file:: size.txt at Fri Nov 14 16:31:53 2014
Size is :: 408 ||Previous size was :: 311
16:31:57 @ - - -
cat size.txt
File to monitor: size.log
Monitoring file:: size.log at Fri Nov 14 16:31:18 2014
Size is :: 45 ||Previous size was :: 45
Monitoring file:: size.log at Fri Nov 14 16:31:28 2014
Size is :: 45 ||Previous size was :: 45
Monitoring file:: size.log at Fri Nov 14 16:31:38 2014
Size is :: 122 ||Previous size was :: 45
Monitoring file:: size.log at Fri Nov 14 16:31:48 2014
Size is :: 219 ||Previous size was :: 122
Monitoring file:: size.log at Fri Nov 14 16:31:58 2014
Size is :: 316 ||Previous size was :: 219
16:32:02 @ - - -
```

### 3. File Image

```
16:45:13 | @ [REDACTED]
sudo dd if=/dev/zero of=./seppala_garcia.img bs=1024 count=64
64+0 records in
64+0 records out
65536 bytes (66 kB) copied, 0.000540758 s, 121 MB/s
16:45:15 | @ [REDACTED]
sudo mkfs.ext2 -b 1024 -N 16 -i 1024 seppala_garcia.img
mke2fs 1.42.9 (4-Feb-2014)
seppala_garcia.img is not a block special device.
Proceed anyway? (y,n) y
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
16 inodes, 64 blocks
3 blocks (4.69%) reserved for the super user
First data block=1
1 block group
8192 blocks per group, 8192 fragments per group
16 inodes per group

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

## (a) File system specs

```
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: dc761934-d31e-423d-aa14-d17d5504b9f3
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: ext_attr resize_inode dir_index filetype sparse_super
                     signed_directory_hash
Filesystem flags: user_xattr acl
Default mount options: clean
Filesystem state: Continue
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 16
Block count: 64
Reserved block count: 3
Free blocks: 43
Free inodes: 5
First block: 1
Block size: 1024
Fragment size: 1024
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 16
Inode blocks per group: 2
Filesystem created: Thu Nov 13 15:15:30 2014
Last mount time: n/a
Last write time: Thu Nov 13 15:15:30 2014
Mount count: 0
Maximum mount count: -1
Last checked: Thu Nov 13 15:15:30 2014
Check interval: 0 (<none>)
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
First inode: 11
Inode size: 128
Default directory hash: half_md4
Directory Hash Seed: de84c901-90ef-466b-8956-ae266b40436f
Directories: 2
Group 0: block bitmap at 3, inode bitmap at 4, inode table at 5
          43 free blocks, 5 free inodes, 2 used directories
(END)
```

## (b) Root Directory

```
debugfs: imap ./
Inode 2 is part of block group 0
    located at block 5, offset 0x0080
debugfs: bd -f ./ 0
0000 0200 0000 0c00 0102 2e00 0000 0200 0000 .....
0020 0c00 0202 2e2e 0000 0b00 0000 1400 0a02 .....
0040 6c6f 7374 2b66 6f75 6e64 0000 0c00 0000 lost+found.....
0060 d403 0801 6162 635f 7669 7274 0000 0000 ....abc_virt....
0100 0000 0000 0000 0000 0000 0000 0000 0000 .....
*

```

## (c) ABC's

```
debugfs: imap abc_virt
Inode 12 is part of block group 0
    located at block 6, offset 0x0180
debugfs: bd -f abc_virt 1
0000 6262 6262 6262 6262 6262 6262 6262 6262 bbbbbbbbbb.....
*
```

## (d) Making Directories

Each time a new directory is created it uses an inode. When the file system is created it only has 5 open inodes, since we created it with 16. Writing the 'abc' file uses a node. Thus we have 4 inodes left. Each time a directory is made it will use a inode as well. Once these are used, we will have no inodes left to create any new files.