

[REPORT]



■과 목 명: 운영체제 (SWE3004-42)

■담 당 교 수: 신동균 교수님

■제 출 일: 2022년 5월 19일

■학 과: 수학과

■학 번: 2016314786

■성 명: 김호진

Chap 39. FS APIs

수학과 김호진 (2016314786)

1. Stat: Write your own version of the command line program stat, which simply calls the stat () system call on a given file or directory. Print out file size, number of blocks allocated, reference (link) count, and so forth. What is the link count of a directory, as the number of entries in the directory changes? Useful interfaces: stat (), naturally.

특정 파일 또는 디렉토리에 대해 stat() system call을 호출하는 command line 프로그램 mystat.c를 다음과 같이 작성하였다.

```
home > borussen > ostep_ch39 > C mystat > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4  #include <sys/stat.h>
5  #include <time.h>
6
7  int main(int argc, char **argv) {
8
9      if (argc != 2) {
10         fprintf(stderr, "Usage: ./stat <filename>\n");
11         exit(EXIT_FAILURE);
12     }
13
14     struct stat file_status;
15     char *filename = argv[1];
16
17     if (stat(filename, &file_status) != 0) {
18         fprintf(stderr, "Failed to access file status\n");
19         exit(EXIT_FAILURE);
20     }
21
22     printf("File: %s\n", argv[1]);
23     printf("Size: %ld    Blocks: %ld    IO block: %ld\n", (long)file_status.st_size, (long)file_status.st_blocks, (long)file_status.st_blksize);
24     printf("Mode: %lo    Inode: %lu    Links: %ld\n", (unsigned long)file_status.st_mode, file_status.st_ino, file_status.st_nlink);
25     printf("Uid: %ld    Gid: %ld\n", (long)file_status.st_uid, (long)file_status.st_gid);
26     printf("Access: %s", ctime(&file_status.st_atime));
27     printf("Modify: %s", ctime(&file_status.st_mtime));
28     printf("Change: %s", ctime(&file_status.st_ctime));
29
30     exit(EXIT_SUCCESS);
31 }
```

디렉토리 foo가 다음과 같이 하나의 하위 디렉토리와 4개의 파일로 구성되어 있다고 하자.

Linux > Ubuntu-20.04 > home > borussen > ostep_ch39 > foo					foo 검색
이름	수정한 날짜	유형	크기		
sub1	2022-05-16 오후 6:27	파일 폴더			
foo1.txt	2022-05-15 오전 1:44	텍스트 문서	1KB		
foo2.txt	2022-05-15 오전 2:42	텍스트 문서	1KB		
foo3.txt	2022-05-15 오전 2:38	텍스트 문서	1KB		
foo4.txt	2022-05-15 오전 2:42	텍스트 문서	0KB		

디렉토리 foo에 대한 mystat의 실행결과는 다음과 같다. 문제가 요구한대로 mystat은 파일 크기, 할당된 블록 수, 링크 개수 등을 출력하며 각 항목에 대해 stat의 실행결과와 같은 값을 가진다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./mystat foo
File: foo
Size: 4096      Blocks: 8      IO block: 4096
Mode: 40755     Inode: 73550    Links: 3
Uid: 1000      Gid: 1000
Access: Thu May 19 01:00:50 2022
Modify: Mon May 16 18:31:27 2022
Change: Mon May 16 18:31:27 2022
borussen@DESKTOP-L834KLC:~/ostep_ch39$ stat foo
  File: foo
  Size: 4096          Blocks: 8          IO Block: 4096   directory
Device: 810h/2064d   Inode: 73550        Links: 3
Access: (0755/drwxr-xr-x)  Uid: ( 1000/borussen)  Gid: ( 1000/borussen)
Access: 2022-05-19 01:00:50.120000000 +0900
Modify: 2022-05-16 18:31:27.157300100 +0900
Change: 2022-05-16 18:31:27.157300100 +0900
Birth: -
```

파일의 링크 수는 파일에 있는 하드 링크의 수를 의미하며, 디렉토리에서는 하위 디렉토리가 생성될 때마다 링크 수가 증가한다. 이때 디렉토리의 링크 수에서 2를 뺀 값이 디렉토리에 존재하는 하위 디렉토리의 총 개수를 나타낸다. 디렉토리의 링크 수는 하위 디렉토리가 이동되거나 삭제될 때마다 하나씩 감소한다.

- 디렉토리 foo에 두 개의 하위 디렉토리가 존재하는 경우 → $\text{Links} = 2 + 2 = 4$

```
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./mystat foo
File: foo
Size: 4096      Blocks: 8      IO block: 4096
Mode: 40755     Inode: 73550    Links: 4
Uid: 1000      Gid: 1000
Access: Thu May 19 01:00:50 2022
Modify: Thu May 19 01:07:25 2022
Change: Thu May 19 01:07:25 2022
```

- 디렉토리 foo에 세 개의 하위 디렉토리가 존재하는 경우 → $\text{Links} = 2 + 3 = 5$

```
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./mystat foo
File: foo
Size: 4096      Blocks: 8      IO block: 4096
Mode: 40755     Inode: 73550    Links: 5
Uid: 1000      Gid: 1000
Access: Thu May 19 01:00:50 2022
Modify: Thu May 19 01:08:21 2022
Change: Thu May 19 01:08:21 2022
```

2. List Files: Write a program that lists files in the given directory. When called without any arguments, the program should just print the file names. When invoked with the `-l` flag, the program should print out information about each file, such as the owner, group, permissions, and other information obtained from the `stat ()` system call. The program should take one additional argument, which is the directory to read, e.g., `mysl -l directory`. If no directory is given, the program should just use the current working directory. Useful interfaces: `stat ()`, `opendir ()`, `readdir ()`, `getcwd ()`.

`stat`, `opendir`, `readdir`, `getcwd` 등을 활용하여 지정된 디렉토리에 있는 파일들을 나열하는 프로그램 `mysl.c`를 다음과 같이 작성하였다. 프로그램은 인수 없이 호출될 때 파일의 이름만을 출력하며, `-l` 플래그를 사용하여 호출될 경우에는 `stat()` system call로부터 얻은 정보를 함께 출력한다.

```
home > borussen > ostep_ch39 > C mysl.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/stat.h>
6  #include <sys/types.h>
7  #include <dirent.h>
8  #include <time.h>
9
10 void print_stats(char *path) {
11     struct stat file_status;
12     if (stat(path, &file_status) < 0) {
13         fprintf(stderr, "Failed to get stats\n");
14         exit(EXIT_FAILURE);
15     }
16     printf("%5u %5u %6ld %6ld %8ld %6lu %5lu %s", file_status.st_uid, file_status.st_gid, file_status.st_size,
17           file_status.st_blocks, file_status.st_blksize, file_status.st_ino, file_status.st_nlink, ctime(&file_status.st_mtime));
18 }
19
20
21 int main(int argc, char **argv) {
22     int detail = 0, opt;
23
24     while ((opt = getopt(argc, argv, "l")) != -1) {
25         if (opt == 'l') {
26             detail = 1;
27         }
28     }
29
30     char *dir = optind < argc ? argv[optind] : ".";
31     char path[1024];
32     struct dirent *pDirent;
33     DIR *pDir;
34
35     if ((pDir = opendir(dir)) == NULL) {
36         fprintf(stderr, "Failed to open directory\n");
37         exit(EXIT_FAILURE);
38     }
39
40     if (detail) {
41         printf("  File  Uid  Gid  Size  Block IO_block  Inode Links          Time\n");
42     }
43     while ((pDirent = readdir(pDir)) != NULL) {
44         if (detail) {
45             sprintf(path, "%s%s", dir, dir[strlen(dir)-1] == '/' ? "" : "/", pDirent->d_name);
46             printf("%8s", pDirent->d_name);
47             print_stats(path);
48         }
49         else {
50             printf("%s\n", pDirent->d_name);
51         }
52     }
53
54     exit(EXIT_SUCCESS);
55 }
```

다음의 디렉토리에 대해 `mysls`와 `mysls -l`을 차례대로 실행하면, 두 경우 모두 문제에서 의도한대로 결과가 출력되는 것을 확인할 수 있다.

Linux > Ubuntu-20.04 > home > borussen > ostep_ch39				ostep_ch39 검색
이름	수정한 날짜	유형	크기	
foo	2022-05-19 오전 1:08	파일 폴더		
foo.txt	2022-05-15 오전 2:13	텍스트 문서	1KB	
myfind	2022-05-15 오전 2:51	파일	18KB	
myfind.c	2022-05-17 오전 1:02	C 파일	3KB	
mysls	2022-05-19 오전 1:36	파일	17KB	
mysls.c	2022-05-19 오전 1:38	C 파일	2KB	
mystat	2022-05-19 오전 1:04	파일	17KB	
mystat.c	2022-05-19 오전 1:20	C 파일	2KB	
mytail	2022-05-15 오전 2:15	파일	17KB	
mytail.c	2022-05-17 오전 1:06	C 파일	2KB	

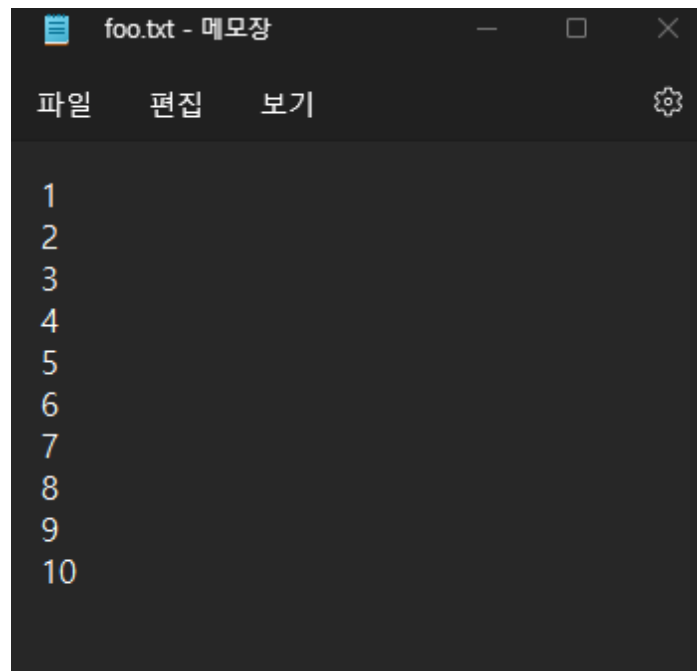
```
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./mysls
mystat
mystat.c
foo
foo.txt
myfind.c
mytail
mysls
myfind
mytail.c
mysls.c
..
.
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./mysls -l
File Uid  Gid  Size  Block  IO_block  Inode  Links      Time
mystat 1000  1000 16976   40    4096  53690     1  Thu May 19 01:04:20 2022
mystat.c 1000  1000  1064     8    4096  73509     1  Thu May 19 01:20:25 2022
foo 1000  1000  4096     8    4096  73550     5  Thu May 19 01:08:21 2022
foo.txt 1000  1000    31     8    4096  73557     1  Sun May 15 02:13:13 2022
myfind.c 1000  1000  2807     8    4096  48892     1  Tue May 17 01:02:00 2022
mytail 1000  1000 17256   40    4096  73512     1  Sun May 15 02:15:33 2022
mysls 1000  1000 17344   40    4096  53700     1  Thu May 19 01:36:15 2022
myfind 1000  1000 17736   40    4096  73574     1  Sun May 15 02:51:13 2022
mytail.c 1000  1000  1437     8    4096  73496     1  Tue May 17 01:06:37 2022
mysls.c 1000  1000  1499     8    4096  73567     1  Thu May 19 01:38:00 2022
.. 1000  1000  4096     8    4096    670    18  Tue May 10 22:43:49 2022
. 1000  1000  4096     8    4096  48956     3  Thu May 19 01:36:15 2022
```

3. Tail: Write a program that prints out the last few lines of a file. The program should be efficient, in that it seeks to near the end of the file, reads in a block of data, and then goes backwards until it finds the requested number of lines; at this point, it should print out those lines from beginning to the end of the file. To invoke the program, one should type: `mytail -n file`, where `n` is the number of lines at the end of the file to print. Useful interfaces: `stat ()`, `lseek ()`, `open ()`, `read ()`, `close ()`.

`stat`, `lseek`, `open`, `read`, `close` 등을 소스코드에 활용하여 파일의 마지막 몇 줄을 출력하는 프로그램 `mytail.c`를 다음과 같이 작성하였다.

```
home > borussen > ostep_ch39 > C mytail.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <unistd.h>
7  #include <fcntl.h>
8
9  #define handle_error(msg) \
10     do { perror(msg); exit(EXIT_FAILURE); } while (0)
11
12  int main(int argc, char **argv) {
13
14     if (argc != 3 || strlen(argv[1]) <= 1 || argv[1][0] != '-') {
15         fprintf(stderr, "Usage: %s -<offset> <filename>\n", argv[0]);
16         exit(EXIT_FAILURE);
17     }
18
19     int lines = atoi(argv[1]);
20     lines *= -1;
21     lines++;
22
23     char *filename = argv[2];
24     struct stat file_status;
25     int fd;
26
27     if (stat(filename, &file_status) == -1) { handle_error("stat"); }
28
29     if ((fd = open(filename, O_RDONLY)) == -1) { handle_error("open"); }
30
31     if (lseek(fd, -1, SEEK_END) == -1) { handle_error("lseek"); }
32
33     char buf[file_status.st_size];
34     int offset;
35
36     while (lines > 0) {
37         if (read(fd, buf, 1) == -1) { handle_error("read"); }
38
39         if (buf[0] == '\n') { lines--; }
40
41         offset = lseek(fd, -2, SEEK_CUR);
42         if (offset == -1) { break; }
43     }
44
45     if (offset > 0 || lines == 0) {
46         if (lseek(fd, 2, SEEK_CUR) == -1) { handle_error("lseek"); }
47     }
48     else {
49         if (lseek(fd, 0, SEEK_SET) == -1) { handle_error("lseek"); }
50     }
51
52     memset(buf, 0, file_status.st_size);
53     if (read(fd, buf, file_status.st_size) == -1) { handle_error("read"); }
54
55     printf("%s", buf);
56     close(fd);
57
58     exit(EXIT_SUCCESS);
59 }
```

다음의 foo.txt에 대해 mytail -n file을 실행하면, 결과가 의도대로 출력되는 것을 확인할 수 있다.



```
foo.txt - 메모장

파일 편집 보기

1
2
3
4
5
6
7
8
9
10
```

```
borussen@DESKTOP-L834KLC:~/ostep_ch39$ gcc -o mytail mytail.c
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./mytail -3 foo.txt
8
9
10
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./mytail -6 foo.txt
5
6
7
8
9
10
```

4. Recursive Search: Write a program that prints out the names of each file and directory in the file system tree, starting at a given point in the tree. For example, when run without arguments, the program should start with the current working directory and print its contents, as well as the contents of any sub-directories, etc., until the entire tree, root at the CWD, is printed. If given a single argument (of a directory name), use that as the root of the tree instead. Refine your recursive search with more fun options, similar to the powerful find command line tool. Useful interfaces: figure it out.

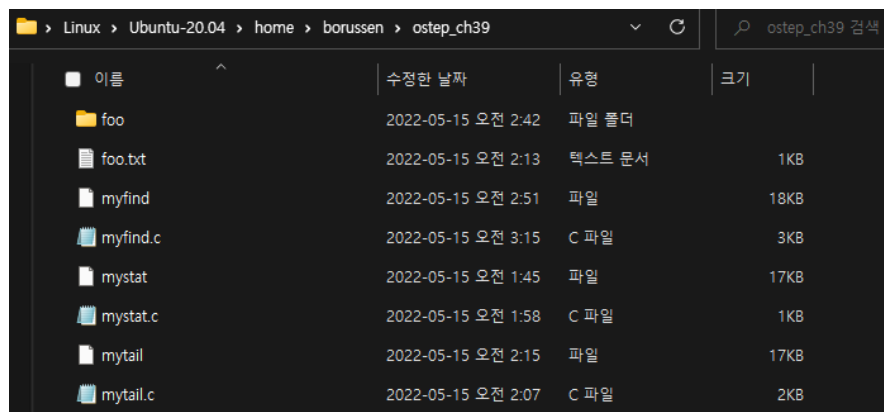
파일 시스템 트리의 지정된 지점으로부터 시작하여 각 파일과 디렉토리 이름을 출력하는 프로그램 myfind.c를 다음과 같이 작성하였다.

```
home > borussen > ostep_ch39 > C myfind.c > main(int, char **)
1  < ∨ #include <stdio.h>
2  < ∨ #include <stdlib.h>
3  < ∨ #include <string.h>
4  < ∨ #include <sys/types.h>
5  < ∨ #include <sys/stat.h>
6  < ∨ #include <unistd.h>
7  < ∨ #include <dirent.h>
8  < ∨ #include <errno.h>
9  < ∨ #include <limits.h>
10 < ∨ #include <regex.h>
11
12 #define STRINGSIZE 1024
13 #define handle_error(msg) \
14     do { perror(msg); exit(EXIT_FAILURE); } while (0)
15
16 < ∨ void find_dir(char *pathname, int currentDepth, int maxDepth, regex_t *preg) {
17
18     DIR *dp;
19     struct dirent *d;
20     errno = 0;
21
22     if (currentDepth++ > maxDepth) { return; }
23
24 < ∨ if ((dp = opendir(pathname)) == NULL) {
25 < ∨     if (errno == EACCES) {
26         fprintf(stderr, "myfind: '%s': Permission denied\n", pathname);
27         return;
28     }
29 < ∨     else {
30         handle_error("opendir");
31     }
32 }
33
34 < ∨ while ((d = readdir(dp)) != NULL) {
35     char filePath[STRINGSIZE] = "";
36     strncpy(filePath, pathname, strlen(pathname));
37
38     if (filePath[strlen(filePath) - 1] != '/') { strncat(filePath, "/", 1); }
39     strncat(filePath, d->d_name, strlen(d->d_name));
40
41 < ∨     if (strcmp(d->d_name, ".") != 0 && strcmp(d->d_name, "..", 2) != 0) {
42         if (preg == NULL || regex(preg, d->d_name, 0, NULL, 0) != REG_NOMATCH) { printf("%s\n", filePath); }
43         if (d->d_type == DT_DIR) { find_dir(filePath, currentDepth, maxDepth, preg); }
44     }
45 }
46     closedir(dp);
47 }
```


home > borussen > ostep_ch39 > C myfind.c > ...

```
49 int main(int argc, char **argv) {
50
51     char *pathname = ".";
52     char *pattern = "";
53     struct stat sb;
54     int maxDepth = INT_MAX, opt, currentDepth = 1, enable_pattern = 0;
55     regex_t preg;
56
57     while ((opt = getopt(argc, argv, "d:n:")) != -1) {
58         switch (opt) {
59             case 'd':
60                 maxDepth = atoi(optarg);
61                 if (maxDepth < 0) {
62                     fprintf(stderr, "Max depth must be positive\n");
63                     exit(EXIT_FAILURE);
64                 }
65                 break;
66             case 'n':
67                 pattern = optarg;
68                 enable_pattern = 1;
69                 break;
70             default:
71                 break;
72         }
73     }
74
75     if (argc > 3 && optind == 1) {
76         fprintf(stderr, "Usage: %s -d [max depth] -n [pattern] [filepath]\n", argv[0]);
77         exit(EXIT_FAILURE);
78     }
79
80     if (optind == argc - 1) { pathname = argv[optind]; }
81
82     if (enable_pattern && regcomp(&preg, pattern, 0) != 0) { handle_error("regcomp"); }
83
84     if (stat(pathname, &sb) == -1) { handle_error("stat"); }
85
86     if (!enable_pattern || (enable_pattern && regexec(&preg, pathname, 0, NULL, 0) != REG_NOMATCH)) { printf("%s\n", pathname); }
87
88     if (S_ISDIR(sb.st_mode)) {
89         if (enable_pattern) { find_dir(pathname, currentDepth, maxDepth, &preg); }
90         else { find_dir(pathname, currentDepth, maxDepth, NULL); }
91     }
92 }
93
94 exit(EXIT_SUCCESS);
95 }
```

myfind.c가 존재하는 디렉토리에서 별다른 인수 없이 프로그램을 실행하면, 다음과 같이 현재 작업 디렉토리에서 시작하여 하위 디렉토리의 내용까지 전부 print 하는 것을 확인할 수 있다.



이름	수정된 날짜	유형	크기
foo	2022-05-15 오전 2:42	파일 폴더	
foo.txt	2022-05-15 오전 2:13	텍스트 문서	1KB
myfind	2022-05-15 오전 2:51	파일	18KB
myfind.c	2022-05-15 오전 3:15	C 파일	3KB
mystat	2022-05-15 오전 1:45	파일	17KB
mystat.c	2022-05-15 오전 1:58	C 파일	1KB
mytail	2022-05-15 오전 2:15	파일	17KB
mytail.c	2022-05-15 오전 2:07	C 파일	2KB

```
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./myfind
.
./mystat
./mystat.c
./foo
./foo/foo1.txt
./foo/foo4.txt
./foo/foo2.txt
./foo/foo3.txt
./foo.txt
./myfind.c
./mytail
./myfind
./mytail.c
```

단일 인수로 디렉토리 이름이 지정된 경우에는 해당 인수를 트리의 root로 사용하여 결과를 출력하는 것을 확인할 수 있다.

```
borussen@DESKTOP-L834KLC:~/ostep_ch39$ ./myfind foo
foo
foo/foo1.txt
foo/foo4.txt
foo/foo2.txt
foo/foo3.txt
```