CENG328 Operating Systems
Laboratory Chapter 7

1 File System

Unix and Linux based operating systems use specific filesystems (ext2 / ext3 / ext4, xfs, btrfs, etc) for storing files, directories and other filesystem objects such as links, pipes and sockets. These filesystems usually store information such as filename, size, inode number, owning user / group info, access modes, timestamps, etc. When listing files with ls command, these different file types can be recognized by the first character next to the permissions:

- **Regular File:** Stores data (text, image, audio...) that can be read/written by programs. Denoted by "-" sign.
- **Directory:** A data structure that stores other files, directories, etc. Denoted by "d" sign.
- **Link:** A softlink (symlink) is a file that acts as a pointer to another file. Denoted by "l" sign.
- **Socket:** A filesystem-level IPC method to establish communication between processes. Denoted by "s" sign.
- **Pipe:** A filesystem-level IPC method to establish communication between processes. Denoted by "p" sign.
- **Character device:** A device which the driver communicates with single characters. Denoted by "c" sign.
- **Block device:** A device which the driver communicates with blocks of data. Denoted by "b" sign.

```
$ ls
drwxrwxr-x 2 user   user     4096 May  3 18:29 backups
lrwxrwxrwx 1 user   user        6 May  3 18:29 list -> list.1
-rw-rw-r-- 1 user   user      132 May  3 18:29 list.0
-rw-rw-r-- 1 user   user      218 May  3 18:29 list.1
-rw-rw-r-- 1 user   user       63 May  3 18:29 list.2
brw-rw---- 1 root   disk    7,  6 Apr 24 10:11 sda1
prw-rw-r-- 1 user   user        0 May  3 18:29 somepipe
srwxrwxr-x 1 user   user        0 May  3 18:28 somesocket
crw------- 1 root   root   10, 55 Apr 24 10:12 vboxdrv
```

Information stored about a file by the filesystem can be queried by "stat" command:

```
$ stat list.0
  File: 'list.0'
  Size: 132             Blocks: 24         IO Block: 4096   regular file
Device: 2fh/47d Inode: 40894950    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    user)   Gid: ( 1000/    user)
Access: 2018-05-03 18:29:09.138309639 +0300
Modify: 2018-05-03 18:29:08.862312844 +0300
Change: 2018-05-03 18:29:08.862312844 +0300
 Birth: -
```

```
 $ stat list
  File: 'list' -> 'list.1'
  Size: 6               Blocks: 8          IO Block: 4096   symbolic link
Device: 2fh/47d Inode: 40894965    Links: 1
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/    user)   Gid: ( 1000/    user)
Access: 2018-05-03 18:29:30.642060416 +0300
Modify: 2018-05-03 18:29:30.642060416 +0300
Change: 2018-05-03 18:29:30.642060416 +0300
 Birth: -
```

2 Programming

You may use stat() system call to obtain info about files in C. Study the following code (fs1.c):

```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>
#include <string.h>

int main(int argc, char **argv) {
  struct stat statbuf;
  if (argc != 2) {
    printf("Usage: %s \e[4mFILENAME\n", argv[0]);
    return -1;
  }
  stat(argv[1], &statbuf);
  printf("Filename: %s\n", argv[1]);
  if (S_ISREG(statbuf.st_mode))
    printf("Type: regular file\n");
  else if (S_ISDIR(statbuf.st_mode))
    printf("Type: directory\n");
  else if (S_ISLNK(statbuf.st_mode))
    printf("Type: link\n");
  else if (S_ISFIFO(statbuf.st_mode))
    printf("Type: pipe\n");
  else if (S_ISSOCK(statbuf.st_mode))
    printf("Type: socket\n");
  else if (S_ISCHR(statbuf.st_mode))
    printf("Type: character device\n");
  else if (S_ISBLK(statbuf.st_mode))
    printf("Type: block device\n");
  printf("Inode: %lu\n", statbuf.st_ino);
  printf("Size: %ld\n", statbuf.st_size);
  printf("UID: %d\n", statbuf.st_uid);
  printf("Last Accessed at: %ld\n", statbuf.st_atim.tv_sec);
  printf("Permissions: %o\n", statbuf.st_mode & 0777);
  if (statbuf.st_mode & S_IRUSR)
    printf("File has user read permission\n");
  if (statbuf.st_mode & S_IWGRP)
    printf("File has group write permission\n");
  if (statbuf.st_mode & S_IXOTH)
    printf("File has others execution permission\n");
  return 0;
}
```

stat() system call queries the file given as the input parameter and fills a struct with information about this file. Some of the info that can be displayed are omitted in this code, you may learn more about them using "man 2 stat".

So far you have used "cd" and "ls" commands to traverse directories and list their contents. Directories can be traversed in C as well. Study the code below (fs2.c):

```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>
#include <string.h>

void traverse(char*, int);

int main(int argc, char **argv) {
  if (argc != 2) {
    printf("Usage: %s \e[4mDIRNAME\n", argv[0]);
    return -1;
  }
  // call the traverse function with the given directory name
  printf("%s/\n", argv[1]);
  traverse(argv[1], 4);
  return 0;
}

void traverse(char* dirname, int depth) {
  DIR* dp;
  struct dirent* entry;
  struct stat statbuf;

  // open the directory structure and go in
  if ((dp = opendir(dirname)) == NULL) {
    printf("No such directory: %s\n", dirname);
    return;
  }
  chdir(dirname);

  // read all files and directories within
  while ((entry = readdir(dp)) != NULL) {
    lstat(entry->d_name, &statbuf);

    // if the chosen item is a directory, print its name
    // and recursively travel into it
    if (S_ISDIR(statbuf.st_mode)) {
      if (strcmp(entry->d_name, ".") != 0
          && strcmp(entry->d_name, "..") != 0) {
        printf("%*s%s/ (%ld bytes, mode %o)\n", depth, " ",
            entry->d_name, statbuf.st_size, statbuf.st_mode & 0777);
        traverse(entry->d_name, depth + 4);
      }
    } else
      printf("%*s%s (%ld bytes, mode %o)\n", depth, " ", entry->d_name,
          statbuf.st_size, statbuf.st_mode & 0777);
  }
```

```
  // fs2.c continued
  // go back to the parent directory and close directory structure
  chdir("..");
  closedir(dp);
}
```

Compile this program and run it like "`./fs1 .`", "`./fs1 /home/`", "`./fs1 /bin/`", etc. A list of all directories and files that reside under the directory you have specified in the command line argument should appear on your screen, along with their sizes and their access modes.

In order to get a list all files and directories in a directory, you must first open the directory structure with **opendir()**, it returns a pointer to the structure. Then, each time **readdir()** is called, a pointer to the next file/directory is returned. At this point, **stat()** or **lstat()** system calls can be used to fill a structure, which contains various information (inode number, mode, user id, size, etc) about the selected file/directory. readdir() returns NULL when there are no more files/directories left in the open directory. When this happens, **closedir()** can be called to close the directory structure and free the directory pointer.

3 Exercises

1. Read man pages for the following library functions: stat (2), opendir, readdir, closedir, chdir.
2. Red man page for stat (3) command.
3. Modify fs2.c such that inode numbers are printed for each file and directory as well. In order to accomplish this, you may refer to man page of stat (2) system call by running "man 2 stat".