ITF22519: Introduction to Operating Systems

Fall Semester, 2021

Lab13: File Systems

Submission Deadline: Novermber 26th, 2021 23:59

A file is a logical unit of information created by a process. File Systems are parts of the OS that manages files. File Systems define how a file is structured, named, accessed, used, protected, implemented, and managed. In this lab, you will do some practice with some of the most common file system calls to make input-output operations on files as well as operations to handle files. In addition, you will also learn how to set up file permissions.

Before you start, remember to `commit` and `push` your previous lab to your git repository. Then, try to `pull` the new lab to have all necessary materials:

```
$ cd Introduction2OS/labs
$ git pull upstream main
$ cd lab13
```

1 Common System Calls

- ```
  #include <sys/types.h>
  #include <sys/stat.h>
  #include <fcntl.h>

  int open(const char* path, int flags)
  ```

  Opens a file at path and returns a file descriptor. For example, `open("/tmp/myfile", O_RDONLY)` opens an existing file called `myfile` in the `temp` directory in read-only mode. It then returns a file descriptor that can be used like a handle to the open file.

- ```
  #include <sys/types.h>
  #include <sys/stat.h>
  #include <fcntl.h>

  int creat(const char* path, mode_t mod).
  ```

  This system call is equivalent with:

  ```
  open(path, O_WRONLY | O_CREAT | O_TRUNC, mod);
  ```

- ```
  #include <unistd.h>

  ssize_t read(int fd, void *buf, size_t count)
  ```

  Reads `count` bytes from the file descriptor `fd` into the memory location starting at `buf`. It starts reading from the current offset into the file. The offset is set to zero if the file has just been opened.
  ```

- `#include <unistd.h>`

  `ssize_t write(int fd, const void* buf, size_t noct)`

  Writes `noct` bytes from the buffer `buf` into the file descriptor `fd`. The function returns the number of bytes written and the value -1 in case of an error.

- `off_t lseek(int fd, off_t offset, int whence)`

  Sets the offset of the file descriptor `fd` to `offset`, depending on `whence`. If `whence` is `SEEK_SET`, then the offset is figured from the start of the file. If `whence` is `SEEK_CUR`, then the offset is relative to the current offset.

- `#include <unistd.h>`

  `int close(int fd);`

Closes a file and thus eliminating the assigned descriptor. The function returns 0 in case of success and -1 in case of an error. At the termination of a process an open file is closed anyway.

Task 1

Compile and run the C program given below and explain how it works.

```
#include<stdio.h>
#include<stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

char buf1[]="Introduction to Opearting Systems ";
char buf2[]="Lab13 File System";

int main( void){
     int fd;
     if ((fd=creat("TestFile.out", 0666)) < 0){
          perror("Error creating file");
          exit (1);
     }

     if (write(fd, buf1, sizeof(buf1)) < 0){
         perror("Error writing buffer 1");
         exit(2);
     }

     if (lseek(fd, 8192, SEEK_SET) < 0){
         perror("Error Positioning file");
         exit(3);
     }
     if (write(fd, buf2, sizeof(buf2)) < 0){
         perror("Error writing buffer 2");
         exit(2);
      }
}
```

2 Hidden Directories

A `hidden directory` is a file or folder which is not displayed by default in directory listing. It is usually used to store user preferences. Hidden directories are not a security mechanism kind of because access is not restricted.
Lets do simple practice with hidden directories. Do the following steps:

- Make a directory named `Test` by running `mkdir Test` command.
- Use `ls` command to view the files and directories in your current folder.
- Rename that directory to `.Test` by running `mv Test/ .Test/` command.
- Use `ls` command to view the files and directories in your current folder.
- Use `ls` command to view the files and directories in your current folder. Note that at this step, the `.Test` directory is not listed.
- Go to the hidden directory `Test` by running `cd .Test` command. You are now in a hidden directory.
- Go to the parent directory by running `cd ..` command.

Now, run the command `ls -la`. If you recall from lab1, you should be aware that the `-l` flag lists item line-by-line and that the `-a` flag lists all files and directories including hidden ones. Again, ***hidden*** directories or folders is not a security mechanism because it is still accessible. However, it can cheat naive users.

Task 2

When you run the command `ls -la` in any directory, you will see two directories named "." and "..". What are these directories and why are they in every directory? (Hints: Refer to lab1).

3 File Permissions

Type the command `ls -l` in your current directory. The option `-l` means that the `ls` command will list each file or sub directory as a line. You should see something like:

```
total 55
drwxr-xr-x 2 ttdinh ttdinh 4096 Aug 31 06:16 lab1
drwxr-xr-x 2 ttdinh ttdinh 4096 Nov  3 12:56 lab10
drwxr-xr-x 2 ttdinh ttdinh 4096 Nov 17 13:44 lab11
drwxr-xr-x 2 ttdinh ttdinh 4096 Nov 17 13:45 lab12
drwxr-xr-x 2 ttdinh ttdinh 4096 Oct  1 14:47 lab2
drwxr-xr-x 4 ttdinh ttdinh 4096 Sep 16 14:16 lab3
drwxr-xr-x 2 ttdinh ttdinh 4096 Oct 27 14:52 lab4
drwxr-xr-x 3 ttdinh ttdinh 4096 Sep 25 12:21 lab5
drwxr-xr-x 3 ttdinh ttdinh 4096 Oct  7 09:13 lab6
drwxr-xr-x 3 ttdinh ttdinh 4096 Nov  5 11:21 lab7
drwxr-xr-x 2 ttdinh ttdinh 4096 Oct 21 10:53 lab8
drwxr-xr-x 2 ttdinh ttdinh 4096 Oct 27 14:58 lab9
drwxr-xr-x 2 ttdinh ttdinh 4096 Oct  6 02:35 solution
-rwxr--r-- 1 ttdinh ttdinh    9 Sep  3 16:10 Testcase.txt
```

You can now see the file permissions associated with each item. Each file and directory has three kinds of user based permission groups: owner, group, and other (all users). In addition, each file or directory has three basic permission types: read, write, and execute.

Each line output of the command `ls -l` is displayed as `-rwxrwxrwx 1 owner:group`. The `-` is a special permission flag used to indicate whether the item is a directory (presented with the character `d`) or a file (presented with the character `-`). The three sets of `rwx` stand for read, write, and execute permissions being granted for the owner, group, and all users, respectively. If the permission is not granted, a `-` character is substituted in place of the permission type. The last piece of information is the owner and group assignment.

Again, man page is your best friend in Linux. Before moving the the rest of this lab, you may want to read the man pages for the following programs `chmod`, `chown` and `chgrp`.

Task 3

Suppose that the file *myscript.sh* is in your current directory.

- What command would you type to find the owner of the file *myscript.sh*?
- What command would you type to change the owner of the file *myscript.sh* to your username?
- What would be the side effect of `chmod -R 777`?

4 Exercises

4.1 Exercise 1 (30 pts)

Using file system calls in this lab to write a C program *Printing.c* that prints the last 10 characters of a file input to standard output. Assume that the input file is named `InputFile.txt` and that it contains more than 10 characters. You are not supposed to use `printf()` and any variants to print the expected outputs.

4.2 Exercise 2 (40 pts)

Using file system calls in this lab to write a C program *copy.c* that does the following:

- The program takes `source_file` and `dest_file` as two input parameters. Then, the command to be run in Bash should be: `./copy source_file dest_file`.
- If the number of input parameters is not two, the program should print out usage message and exit.
- If the `source_file` exists, copy its contents to the `dest_file`.
- If the `source_file` does not exist, print out the error message and exit.

Expected output of the script is showed below:

```
$ ./copy test.txt
Usage: ./copy source_file dest_file

$ ./copy source_file dest_file
The file source_file does not exist

$ ./copy source_file dest_file
Copying the file source_file to dest_file...
```

4.3 Exercise 3 (30 pts)

Consider the following command for the permission of file *myscript.sh*

```
chmod 650 myscript.sh
```

Answer the following questions:

- Who (owner, group, other) can execute the file?.
- Who can write to the file?
- What can the owner do?

(Note: Owner is the one who creates the file, group is the users from group that owner is a member, and other are all other users).

5 What To Submit

Complete this lab and put your files into the lab13 directory of your repository. Run git `add .` and `git status` to ensure the files have been added and commit the changes by running `git commit -m Commit Message`. Finally, submit your files to GitHub by running `git push`. Check the GitHub website to make sure all files have been submitted.