COMP1521

W9 - System Programming

Overview

Admin

File permissions (Q2, Q3, Q4)

Environment variables (Q1)

UTF-8

Misc (Q5)

Admin

Assignment 2 due W10 friday.

W8 test due today.

W9 test released today.

W9 labs due next monday.

(Optional) practice exam running in W10 lab (I think).

- There are also lab questions for that week

My experience!!!!!!

Pretty please fill it out.

I’ll read all feedback given.

Course staff may offer some incentives 👀.

I will be sad if no one does it.

File Permissions

File permissions

File permissions are stored in a file's metadata.

They are displayed to the terminal when using commands like `ls`:

```
-rw-r--r-- 1 z5420273 z5420273  534 Sep 19  2022 count.s
-rw-rw-r-- 1 z5420273 z5420273  856 Sep 19  2022 dynamic_load.s
```

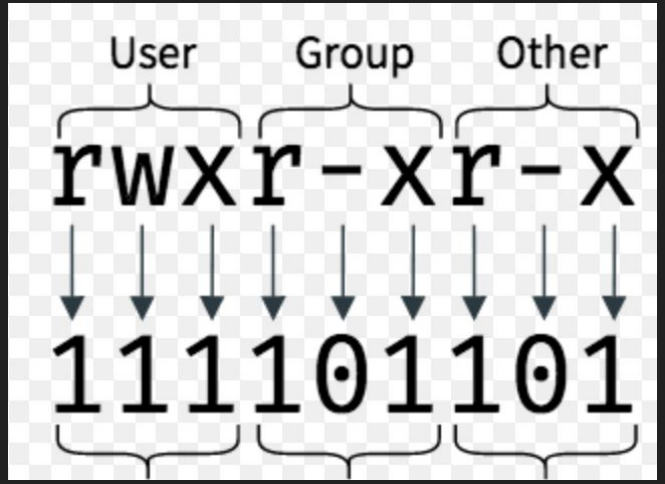
They are stored in a bitwise representation:

: ttttrwxrwxrwx

(first 4 bits represent file type)

File permissions can be read using `stat`.

File permissions can be modified using `chmod`.



File permissions

User = you

Group = probably just you, but maybe close friends

Others = everyone else

Stat

`stat` takes two arguments: a pathname of the file to stat, and a stat struct object pointer to store the results in.

It will return an error code (non zero) on failure, or zero on success.

The information about file permissions and type can be accessed in the st_mode field.

There exist already defined bitmasks which can be used to inspect the object:
run “man inode”

The format of storage is the same as on the previous slide.

```c
int main(int argc, char **argv) {
    struct stat s;
    char *pathname = argv[1];
    int code = stat(pathname, &s);
    if (code != 0) {
        perror("couldn't stat file");
        exit(1);
    }
    mode_t mode = s.st_mode;
}
```

Stat

The object returned by stat also contains information:

- User id of owner
- Group id of owner
- File size
- Time of last access, last change, creation
- More!!!

Chmod

`chmod` takes two arguments: a `path` : the pathname of a file and a `mode`: the `mode_t` (just a number) which is formatted the same as the `s.st_mode` field from stat.

```
int main(int argc, char **argv) {
    char *pathname = argv[1];
    mode_t mode = S_IRUSR | S_IWUSR | S_IXUSR | S_IRGRP | S_IROTH;
    int code = chmod(pathname, mode);
    if (code != 0) {
        perror("couldn't chmod file");
        exit(1);
    }
}
```

Tutorial Q2

2. The `stat()` and `lstat()` functions both take an argument which is a pointer to a `struct stat` object, and fill it with the meta-data for a named file.

   On Linux, a `struct stat` contains the following fields (among others, which have omitted for simplicity):

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

   Explain what each of the fields represents (in more detail than given in the comment!) and give a typical value for a regular file which appears as follows:

```
$ ls -ls stat.c
8 -rw-r--r--  1 jas  cs1521  1855  Sep  9 14:24 stat.c
```

   Assume that `jas` has user id 516, and the `cs1521` group has group id 36820.

Tutorial Q3

3. Consider the following (edited) output from the command `ls -l ~cs1521`:

```
drwxr-x--- 11 cs1521 cs1521 4096 Aug 27 11:59 17s2.work
drwxr-xr-x  2 cs1521 cs1521 4096 Aug 20 13:20 bin
-rw-r-----  1 cs1521 cs1521   38 Jul 20 14:28 give.spec
drwxr-xr-x  3 cs1521 cs1521 4096 Aug 20 13:20 lib
drwxr-x--x  3 cs1521 cs1521 4096 Jul 20 10:58 public_html
drwxr-xr-x 12 cs1521 cs1521 4096 Aug 13 17:31 spim
drwxr-x---  2 cs1521 cs1521 4096 Sep  4 15:18 tmp
lrwxrwxrwx  1 cs1521 cs1521   11 Jul 16 18:33 web -> public_html
```

a. Who can access the `17s2.work` directory?

b. What operations can a typical user perform on the `public_html` directory?

c. What is the file `web`?

d. What is the difference between `stat("web", &info)` and `lstat("web", &info)`? (where `info` is an object of type `(struct stat)`)

Tutorial Q4

4. Write a C program, chmod_if_public_write.c, which is given 1+ command-line arguments which are the pathnames of files or directories
   If the file or directory is publically-writeable, it should change it to be not publically-writeable, leaving other permissions unchanged.

   It also should print a line to stdout as in the example below

```
$ dcc chmod_if_public_write.c -o chmod_if_public_write
$ ls -ld file_modes.c file_modes file_sizes.c file_sizes
-rwxr-xrwx 1 z5555555 z5555555 116744 Nov  2 13:00 file_sizes
-rw-r--r-- 1 z5555555 z5555555    604 Nov  2 12:58 file_sizes.c
-rwxr-xr-x 1 z5555555 z5555555 222672 Nov  2 13:00 file_modes
-rw-r--rw- 1 z5555555 z5555555   2934 Nov  2 12:59 file_modes.c
$ ./file_modes file_modes file_modes.c file_sizes file_sizes.c
removing public write from file_sizes
file_sizes.c is not publically writable
file_modes is not publically writable
removing public write from file_modes.c
$ ls -ld file_modes.c file_modes file_sizes.c file_sizes
-rwxr-xr-x 1 z5555555 z5555555 116744 Nov  2 13:00 file_sizes
-rw-r--r-- 1 z5555555 z5555555    604 Nov  2 12:58 file_sizes.c
-rwxr-xr-x 1 z5555555 z5555555 222672 Nov  2 13:00 file_modes
-rw-r--r-- 1 z5555555 z5555555   2934 Nov  2 12:59 file_modes.c
```

   Make sure you handle errors.

Environment variables

Environment variables

ENV variables are variables shared between processes.

**environ is the secret third argument to main (although, this method of accessing environment variables is now obsolete).

Environment variables can be more easily accessed using `getenv(char *name)`.

One very common environment variable is `PATH`, which contains a list of `:` separated directories where your computer can expect to find executable command line programs (like `ls`, `cat`, `git`, etc).

Cue example!

Tutorial Q1

1. Write a C program, `print_diary.c`, which prints the contents of the file `$HOME/.diary` to stdout

   The lecture example getstatus.c shows how to get the value of an environment variable.

   snprintf is a convenient function for constructing the pathname of the diary file.

UTF-8

UTF-8

A way to represent more characters than ASCII

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+010000	[b]U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

UTF-8 example

f0 9f a4 93 (4 byte character)

11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

11110000 10011111 10100100 10010011

= https://www.cogsci.ed.ac.uk/~richard/utf-8.cgi?input=f0+9f+a4+93&mode=bytes

Fun extra question

Tutorial Q5

5. Write a C program, `print_file_bits.c`, which given as a command line arguments the name of a file contain 32-bit hexadecimal numbers, one per line, prints the low (least significant) bytes of each number as a signed decimal number (-128..127).