COMP1521

W9 - System Programming

Overview

Admin

File permissions (Q2, Q3, Q4)

Environment variables (Q1)

UTF-8

Reading directories & recursive file system traversal

Misc (Q5)

Admin

Assignment 2 due W10 friday.

W8 test due thursday.

W9 test released thursday.

W9 labs due next monday.

(Optional) practice exam running in W10 lab.

- There are also lab questions for that week

My experience!!!!!!

Do it pls

File Permissions

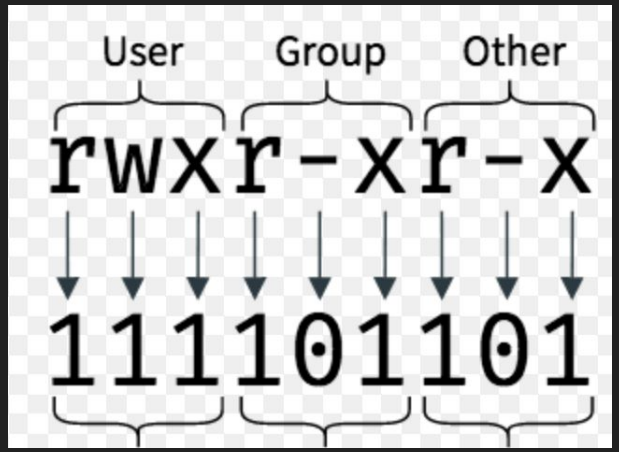
File permissions

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`.

User = you, Group = you (+ maybe friends), Other = everyone on the system.

Reading file permissions

Use the defines in `man inode` to read the file mode.

S_I[R/W/X][USR/GRP/OTH]

And use macros in same man page to check file type

S_IS[REG/DIR/…]

```c
int main(void) {
    char *pathname = "./test.txt";
    struct stat s;
    stat(pathname, &s);
    int mode = s.st_mode;
    // Check if others have read permission
    if (mode & S_IROTH) {
        printf("others have read permission\n");
    }
    // Check if owner has execute permission
    if (mode * S_IXUSR) {
        printf("owner has execute permission\n");
    }
    // Check if file is a directory
    if (S_ISDIR(mode)) {
        printf("file is directory\n");
    }
    // Check if file is a regular file
    // (easier to use S_ISREG(mode))
    if ((mode & S_IFMT) == S_IFREG) {
        printf("file is a regular file\n");
    }
}
```

Modifying file permissions

```c
int main(void) {
    char *pathname = "./test.txt";
    // Set read permission for all users, and write and execute only for file owner
    int mode = S_IROTH | S_IRGRP | S_IRUSR | S_IWUSR | S_IXUSR;
    chmod(pathname, mode);

    // Set the commonly used 0770 permission (read write and execute for user and group)
    chmod(pathname, 0770);

    // Add permission for all users to execute to the existing permission
    struct stat s;
    stat(pathname, &s);
    int new_mode = s.st_mode | S_IXOTH;
    chmod(pathname, new_mode);
}
```

Use chmod. If you need to add/modify permission, it’s often useful to stat first and add permissions using bitwise OR.

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

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    printf("%s\n", getenv("PATH"));
    printf("%s\n", getenv("USER"));
    printf("%s\n", getenv("PWD"));
    printf("%s\n", getenv("HOME"));
}
```

getenv(name) returns the value of an environment variable

Environment variables are special variables managed by your computer that are shared between all processes

PATH

```
z5420273@vx14:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/import/ravel/2/z5420273/bin:/
import/ravel/2/z5420273/bin-pc.amd64.linux
```

A list of `:` separated directories where executables can be found.

Those executables can be invoked by simply writing their name in the terminal

```
z5420273@vx14:~$ ls -l /usr/bin/echo
-rwxr-xr-x 1 root root 43856 Sep 21  2022 /usr/bin/echo
```

```
z5420273@vx14:~$ echo hello
hello
```

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

0xf0 0x9f 0xa4 0x93 (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

Reading directories + traversal

Directories

Open/close just like files.

Read entries from directory using readdir.

See below a full list of the fields in the entry.

```
struct dirent {
    ino_t          d_ino;       /* Inode number */
    off_t          d_off;       /* Not an offset; see below */
    unsigned short d_reclen;    /* Length of this record */
    unsigned char  d_type;      /* Type of file; not supported
                                   by all filesystem types */
    char           d_name[256]; /* Null-terminated filename */
};
```

```c
#include <dirent.h>
#include <stdlib.h>
#include <stdio.h>

int main(void) {
    // Open the directory
    DIR *dir = opendir(".");
    if (!dir) {
        perror("opendir");
        exit(1);
    }

    // Loop through entries
    struct dirent *entry;
    while ((entry = readdir(dir))) {
        printf("%s\n", entry->d_name);
    }

    // Close directory
    closedir(dir);
}
```

file system traversal

Take advantage of recursion to call the “traverse” function on every sub directory and sub sub directory etc.

Sometimes required in final exam questions (and often in assignment 2, but not this term).

```c
void traverse_and_list(char *path) {
    // Open the directory
    DIR *dir = opendir(path);
    if (!dir) {
        perror("opendir");
        exit(1);
    }

    // Loop through entries
    struct dirent *entry;
    while ((entry = readdir(dir))) {
        char this_pathname[MAX_PATHNAME_LEN];
        snprintf(this_pathname, MAX_PATHNAME_LEN, "%s/%s", path, entry->d_name);

        struct stat s;
        stat(this_pathname, &s);

        printf("%s\n", this_pathname);

        if (S_ISDIR(s.st_mode)) {
            // Do not recurse into the current directory or the parent directory
            if (
                strncmp(entry->d_name, ".", MAX_PATHNAME_LEN) == 0
                || strncmp(entry->d_name, "..", MAX_PATHNAME_LEN) == 0) {

                continue;
            }

            // Recurse into directory and print those paths too
            traverse_and_list(this_pathname);
        }
    }

    // Close directory
    closedir(dir);
}
```

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).