Software Systems

Day 8 - Flags, Files

Schedule

- Announcements
- Binary Flags
- File Systems

Announcements

- No class on Friday due to a college-wide long weekend.
- Each class between now and project end will have a small amount of time reserved at the end for a quick standup with your team.
- In general, you can read and start on the homework at any time.
  - Each homework tracks a chapter of Head First C.

Binary Flags

- In Homework 2.5, you may have written code like this:

```
regex_t re;
ret = regcomp(&re, pattern, REG_EXTENDED | REG_NOSUB);
```

- Today, we'll dig into this code a bit more.

Binary Flags

- Exercise (10 minutes):
  - Write a C program that includes `regex.h`.
  - Use the command `gcc -H <file.c>` to see where `regex.h` is on your system. (It's usually in `/usr/include`, but try the command.)
  - In `regex.h`, what are the values of REG_EXTENDED and REG_NOSUB? Do you see other variables that have similar values?
  - What does `REG_EXTENDED | REG_NOSUB` equal? Why might you want to calculate this value in the call to `regcomp`?

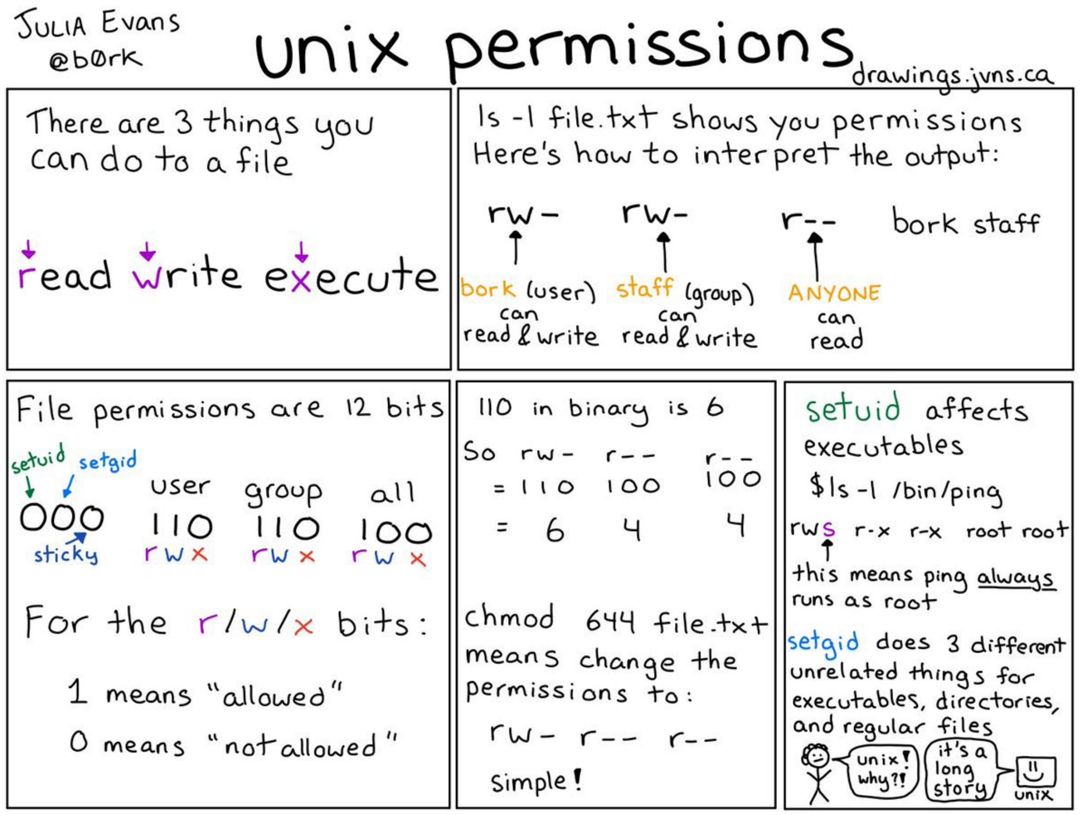
Binary Flags

- On my system:
  - `REG_EXTENDED`: 1
  - `REG_ICASE`: 1 << 1 (equal to 2)
  - `REG_NEWLINE`: 1 << 2 (equal to 4)
  - `REG_NOSUB`: 1 << 3 (equal to 8)
- So `REG_EXTENDED | REG_NOSUB` is (lowest 8 bits only):

```
00000001
00001000
--------
00001001 = 9
```

Binary Flags

- Exercise (15 minutes):
  - Read the cartoon on the right.
  - Answer the questions in the HedgeDoc notes for today.



JULIA EVANS @b0rk
unix permissions
drawings.jvns.ca
There are 3 things you can do to a file
read write execute
ls -l file.txt shows you permissions
Here's how to interpret the output:
rw- rw- r-- bork staff
bork (user) can read & write
staff (group) can read & write
ANYONE can read
File permissions are 12 bits
setuid setgid
000
sticky
user 110 rwx
group 110 rwx
all 100 rwx
For the r/w/x bits:
1 means "allowed"
0 means "not allowed"
110 in binary is 6
So rw- r-- r--
= 110 100 100
= 6 4 4
chmod 644 file.txt
means change the permissions to:
rw- r-- r--
Simple!
setuid affects executables
$ls -l /bin/ping
rws r-x r-x root root
this means ping always runs as root
setgid does 3 different unrelated things for executables, directories, and regular files
unix! why?!
it's a long story
unix

Binary Flags

- The setgid bit:
    - On binary executable files causes the user to inherit the group of the owner
    - On other executable files (usually) does nothing (more on this later)
    - On directories causes new files in the directory to inherit its group ID
- Sticky bit:
    - Collecting files from many users (e.g., a submission system)
    - Shared directory for files (e.g., temporary or public directory)

Binary flags

- Octal flags
  - Sticky but not setuid or setgid: 1
  - Read, write, execute by owner: 7
  - Read, execute by group: 6
  - Read by others: 4
  - All together: 1764
- Symbolically:
  - Lots of ways, but one way is `chmod a=tr,ug+x,u+w <file>`

Binary Flags

- The id command shows you your username and groups.
- In a script, this doesn't change even if you use setuid.
- When you load a script, the OS sees the line `#!/usr/bin/env bash`
- Then it closes the file and starts up bash.
- Then it takes the contents of the file and passes it to bash.
- The file could change in that time window, so most operating systems don't take that risk.

I/O Performance

- A file is a stream of bytes.
- A block is a chunk of memory on a hard disk.
- How do these relate to I/O performance?

I/O Performance

- Exercise (20 minutes):
  - Read about the yes program, in HedgeDoc or on the Web.
  - Answer the questions in the HedgeDoc notes.

For Next Time

- Read Think OS, Chapter 4 and do the reading quiz.
- Read Head First C, Chapter 4.
- Start Homework 3 (due in a week).
- Work on your project (check in with a CA/instructor this/next week).
- Enjoy the long weekend!