Systems and Internet Infrastructure Security

Institute for Networking and Security Research
Department of Computer Science and Engineering
Pennsylvania State University, University Park, PA

I/O (Part 2)

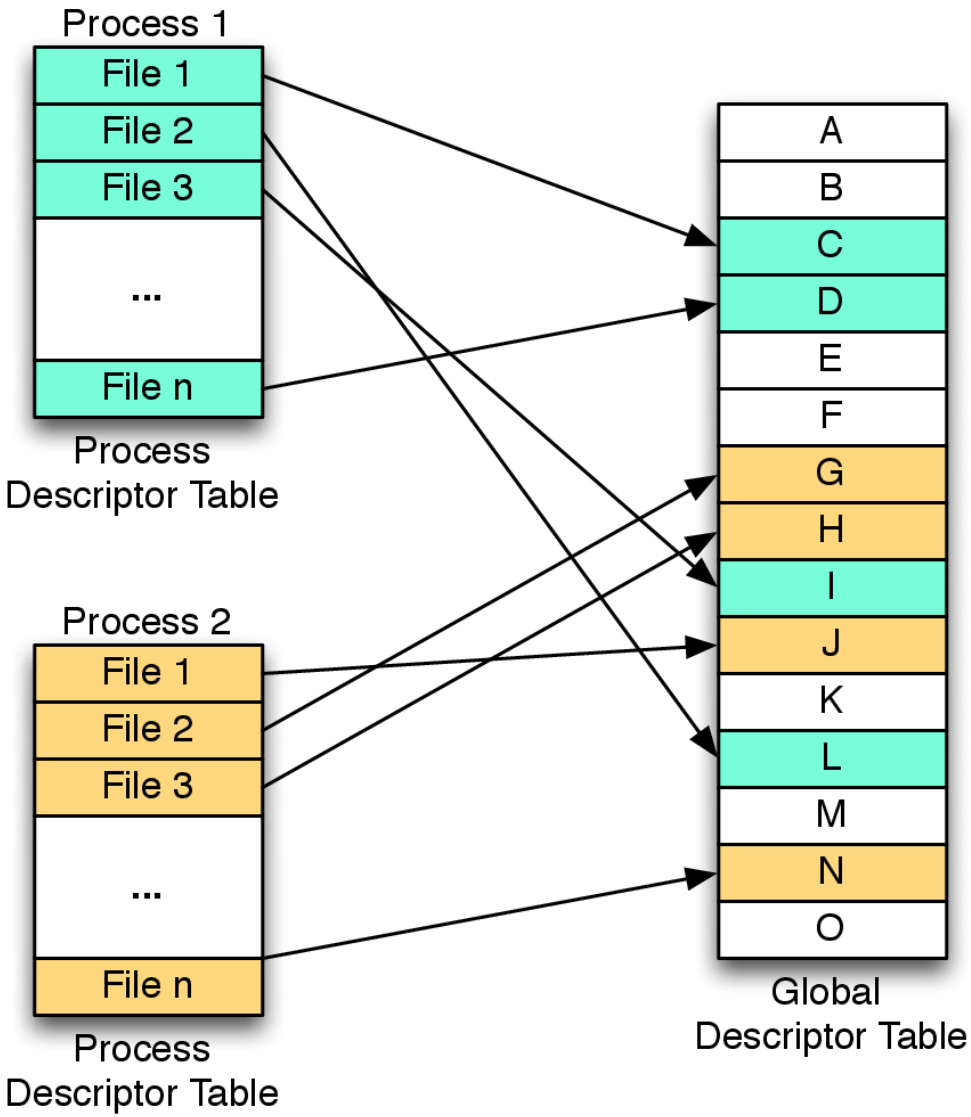
Devin J. Pohly <djpohly@cse.psu.edu>

Low-level I/O

- Defined in POSIX standard
  - I.e., should work on any Unix-like system
  - Stream I/O functions were part of C standard
- Good for working with bytes and binary information
  - Much more control
  - Cleaner interface

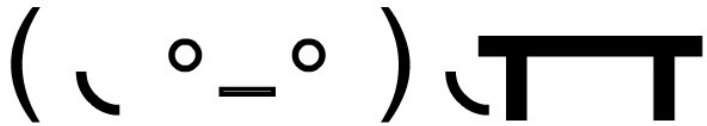
Global file table

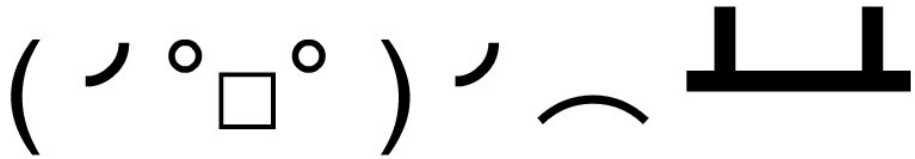
- The kernel keeps track of all the files currently open on the system
  - Stored in a table called the *global file table*
  - Each time you open a file, a new entry is created
  - Tracks information like current offset, read/write permissions, file mode, etc.
  - The same file can be open many times
    - Even by the same process!
- All I/O functions must go through the kernel
  - Why?



File descriptors

- Each process has its own table of open files
  - Called the *(process) file table*
  - On Linux: array of pointers to the global table
  - Also managed by the kernel
- A *file descriptor* is an integer that identifies a specific open file
  - Simplest possibility: index in process file table
  - Acts as a reference to that file
  - Analogous to `FILE *` from stream I/O functions
    - "File handle" can mean either





What is this “mode?”

- A file’s *mode* is just the Unix way of referring to its permissions
  - Who can read it?
  - Who can change it?
  - Who can execute it?
  - Who can list directories?
  - Remember: Unix was designed to be a multi-user OS
- We’ve seen this already:

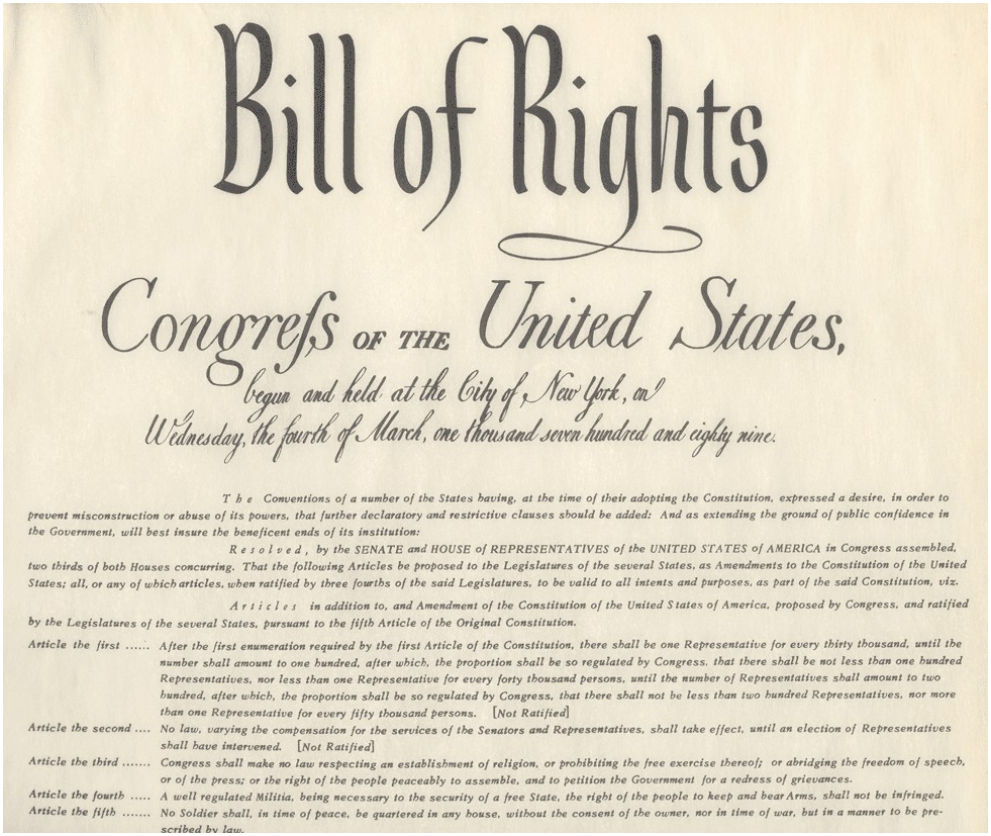
```
chmod +x shello
./shello
```

Access control in Unix

- The Unix filesystem implements *discretionary access control* through file permissions set by the file's owner
  - The permissions are set at the user's discretion
- Every file in the filesystem has a set of bits which determine who has access to it
  - User: the owner of the file, usually the person who created it or "root" (the administrator) for system files
  - Group: a specific set of users on the system, set up by the administrator
  - Other (or "world"): everyone on the system
- Note: the root user has full access to everything, and can change file permissions, owners, etc. at will.

Unix filesystem rights

- There are three rights in a Unix filesystem:
  - Read: allows the subject (user/process) to read the contents of the file
  - Write: allows the subject to alter the contents of the file
  - eXecute: allows the subject to execute the contents of the file (e.g., shell program, binary executable)
  - If you see "RWX" anywhere, that is referring to these
- Why is execute a right?



Bill of Rights

Congress of the United States,

begun and held at the City of New York, on

Wednesday, the fourth of March, one thousand seven hundred and eighty nine.

The Conventions of a number of the States having, at the time of their adopting the Constitution, expressed a desire, in order to prevent misconstruction or abuse of its powers, that further declaratory and restrictive clauses should be added: And as extending the ground of public confidence in the Government, will best insure the beneficent ends of its institution:

Resolved, by the SENATE and HOUSE of REPRESENTATIVES of the UNITED STATES of AMERICA in Congress assembled, two thirds of both Houses concurring. That the following Articles be proposed to the Legislatures of the several States, as Amendments to the Constitution of the United States; all, or any of which articles, when ratified by three fourths of the said Legislatures, to be valid to all intents and purposes, as part of the said Constitution, viz.

Articles in addition to, and Amendment of the Constitution of the United States of America, proposed by Congress, and ratified by the Legislatures of the several States, pursuant to the fifth Article of the Original Constitution.

Article the first ..... After the first enumeration required by the first Article of the Constitution, there shall be one Representative for every thirty thousand, until the number shall amount to one hundred, after which, the proportion shall be so regulated by Congress, that there shall be not less than one hundred Representatives, nor less than one Representative for every forty thousand persons, until the number of Representatives shall amount to two hundred, after which, the proportion shall be so regulated by Congress, that there shall not be less than two hundred Representatives, nor more than one Representative for every fifty thousand persons. [Not Ratified]

Article the second .... No law, varying the compensation for the services of the Senators and Representatives, shall take effect, until an election of Representatives shall have intervened. [Not Ratified]

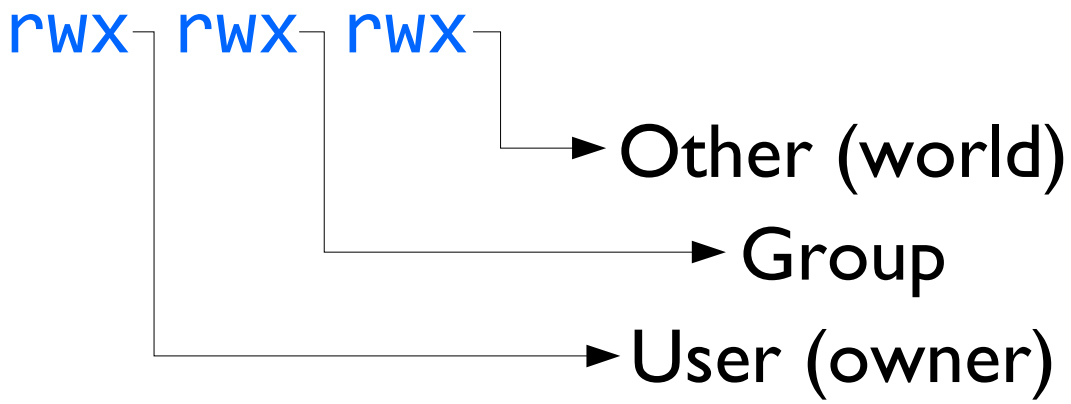
Article the third ..... Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the Government for a redress of grievances.

Article the fourth ..... A well regulated Militia, being necessary to the security of a free State, the right of the people to keep and bear Arms, shall not be infringed.

Article the fifth ..... No Soldier shall, in time of peace, be quartered in any house, without the consent of the owner, nor in time of war, but in a manner to be prescribed by law.

Unix access policy

- Essentially, the mode of a file is a bit string encoding an access policy:



- This is a common shorthand: `r`/`w`/`x` if permission is granted, and `-` if permission is denied, e.g.:

```
rwxr-xr--
```

  - This mode would give permission to the owning user to read, write, and execute the file, the owning group to read and execute, and everyone else only to read.

Mode bits

- Here is a common file mode:

```
rwxr-xr-x
```

  - What does this mode mean?
- These are just bits, where a letter means the bit is set, and a dash means the bit is clear:

```
111101101
```

- Since the bits are grouped in threes, what would be a useful number base for expressing file modes?

Mode bits

- Here is a common file mode:

```
rwxr-xr-x
```

  - What does this mode mean?

- These are just bits, where a letter means the bit is set, and a dash means the bit is clear:

```
111101101
```

- Since the bits are grouped in threes, what would be a useful number base for expressing file modes?

  - File modes are often expressed in octal:

```
0755
```

The chmod command

- To set a file's mode from the command line, use the chmod command:

  `chmod MODE FILE(S)`

- The mode argument is:
  - Some combination of `[ugo]*[-+=][rwx]*`
  - Or a mode in octal
- By the way, `chmod 777` is almost never a good idea...
  - What would this do?

Opening files

- The `open` function opens a file for low-level I/O and returns an integer file descriptor:

```
int open(const char *path, int flags, mode_t mode);
```

  - `path`: a string containing the absolute or relative path to the file to open
  - `flags`: bit field specifying how to open the file and what features you want to enable
  - `mode`: if creating a file, this specifies the permission bits
  - If there is an error when opening the file, this function will return a negative number (often -1)

- Example:

```
int fd = open("/tmp/foo", O_RDWR | O_CREAT, 0644);
```

  - This opens the file /tmp/foo for reading and writing. If it doesn't exist, it is created with the mode `rw-r--r--`.

Flags for open

- Access flags:
  - `O_RDONLY`: read-only
  - `O_WRONLY`: write-only
  - `O_RDWR`: read and write
- Other flags:
  - `O_CREAT`: create the file if it does not exist
  - `O_EXCL`: (used with O_CREAT) ensure the file *doesn't* exist
    - Used to make sure you don't overwrite another file
  - `O_TRUNC`: truncate the file to zero-length if it already exists
  - Lots of others, see `man open`
- As with any bit flags, you bitwise-OR them together with the | operator.

Constants for file modes

- You can use standard constants for mode bits too:
  - `S_IRUSR`: user may read
  - `S_IWUSR`: user may write
  - `S_IXUSR`: user may execute
  - `S_IRWXU`: user may read, write, and execute
  - `S_IRGRP`: group may read
  - `S_IWGRP`: group may write
  - `S_IXGRP`: group may execute
  - `S_IRWXG`: group may read, write, and execute
  - `S_IROTH`: anyone may read
  - `S_IWOTH`: anyone may write
  - `S_IXOTH`: anyone may execute
  - `S_IRWXO`: anyone may read, write, and execute

Error handling

- Many low-level functions specify errors by setting a variable called `errno`
  - Always mentioned in the man page
  - The function returns nonzero to indicate failure
  - `errno` indicates what the specific error was
  - Need to include `<errno.h>`
- Some common errors:
  - `EINVAL`: invalid argument
  - `ENOENT`: file not found
  - `ENOMEM`: out of memory
  - `EACCES`: access denied (mode does not permit the action)

Aside: printing errors

- The `perror` function will check `errno` and print a nicely formatted error message to `stderr`:

```
void perror(const char *s);
```

  - The `s` parameter is just a prefix to give more information about the error (often this is the name of the function that failed).

- Example:

```
int fd = open("foo.txt", O_RDWR, 0644);
if (fd < 0) {
    perror("open");
    return 1;
}
```

Closing files

- The `close` function will close an open file:

```
int close(int fildes);
```

  - `fildes`: the file descriptor of the file to close
  - This function can fail! The return value is 0 for success, or nonzero for failure.