Files and Directories

COSC-361
Stephen Marz

TICKLE COLLEGE OF ENGINEERING
MIN H. KAO DEPARTMENT OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE

1

What Is a File

- Information on a file system
- CPU instructions generally deal with RAM and registers.
  - RAM and registers are relatively small
  - Need some way to access larger memory
    - FILES!

2

Extensions and Magic

- How do we identify the *type* of a file?
- Two ways:
  - Extension (typically three letters after a .)
    - mypicture.jpg
    - myworddocument.docx
  - Magic (usually the first few bytes of a file)
    - ELF (Executable and Linkable Format)

3

ELF Magic

```
smarz@DOMO:~ $ hexdump -C /bin/ls | head
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  03 00 3e 00 01 00 00 00  e0 5a 00 00 00 00 00 00  |..>......Z......|
00000020  40 00 00 00 00 00 00 00  68 13 02 00 00 00 00 00  |@.......h.......|
00000030  00 00 00 00 40 00 38 00  0b 00 40 00 19 00 18 00  |....@.8...@.....|
00000040  06 00 00 00 04 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000050  40 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |@.......@.......|
00000060  68 02 00 00 00 00 00 00  68 02 00 00 00 00 00 00  |h.......h.......|
00000070  08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00  |................|
00000080  a8 02 00 00 00 00 00 00  a8 02 00 00 00 00 00 00  |................|
00000090  a8 02 00 00 00 00 00 00  1c 00 00 00 00 00 00 00  |................|
```

Linux treats nearly all files generically.

The only file type that is actually understood by Linux and UNIX flavors is ELF (Executable and Linkable Format). The OS will actually read this format directly via an exec* system call.

4

Linux ELF Execution

```
smarz@DOMO:~ $ strace ls | head
execve("/usr/bin/ls", ["ls"], 0x7ffe4c6b9010 /* 22 vars */) = 0
brk(NULL)                               = 0x55f1137fc000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffacc58aee0) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=148991, ...}) = 0
mmap(NULL, 148991, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe7cb09f000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libcap.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0@  \0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=22456, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fe7cb09d000
mmap(NULL, 24920, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe7cb096000
mmap(0x7fe7cb098000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7fe7cb098000
mmap(0x7fe7cb09a000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7fe7cb09a000
mmap(0x7fe7cb09b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7fe7cb09b000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000C\2\0\0\0\0"..., 832) = 832
lseek(3, 792, SEEK_SET)                 = 792
read(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\0\201\336\t\36\251c\324\233E\3715oK\5H\334"..., 68) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2136840, ...}) = 0
lseek(3, 792, SEEK_SET)                 = 792
read(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\0\201\336\t\36\251c\324\233E\3715oK\5H\334"..., 68) = 68
lseek(3, 864, SEEK_SET)                 = 864
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
```


5

Secondary Storage

- Can only be accessed via device driver.
  - Disk drivers
    - ATA
    - SCSI
    - NVME
- In OS - user application accesses via special system calls, such as open, read, write, lseek, close.

6

Type Enforcement

- Strong
  - Windows style
    - Files have an attached "handler", such as an application.
- Weak
  - Linux style
    - Files are mainly generic, without much OS intervention.
    - It is up to the application to decide how to read/use the file.

7

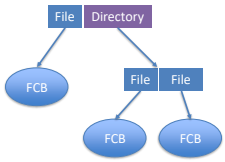
Directories

- Directories is a logical separation of files for organizational and efficiency purposes.
- Approaches to implement directories:
  - Tree-structured directory
  - Directed, acyclic directory
  - Unstructured

8

Tree-Structured Directories

- Root node is the root directory
  - Root node is path '/'
- Branches point to:
  - Other directories
  - File control blocks



9

Tree-Structured Directories

- Only allows one parent per file/directory
- No symbolic links
- No direct directories
  - You must traverse to the root to get down a different branch.

10

Directed-Acyclic Directory

- Allows multiple parents per file/directory.
- Allows links
  - Hard links - share the same FCB (inode in Linux).
    - Hard links may NOT span across file systems.
  - Soft links - a special file with its own FCB.
    - Soft links may span across file systems.

11

Path

- A string interpretation of traversing a tree to get to a certain file.
- Absolute: start with / (the root). All directories must be specified.
  - Example: /home/smarz/hello/world.txt
- Relative: start with CWD (current working directory). Only directories AFTER CWD must be specified.
  - Example: hello/world.txt

12

Practice

- Is mydir/home.txt relative or absolute?
  - Relative
- Symbolic links have their own inodes?
  - True

13

14