CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

File Descriptors

File descriptor -- an `int` to let the OS know which file you want to access (read, write, close, etc.)

Open file (or **Open file description**) -- tracks which file was opened, mode of opening (read/write/etc.), *offset into file*

There can be more than one per file, even in the same process

This is created each time open is called

OS keeps *per-process* table linking file descriptors with open files

```
PID 70

| FD  | Open file |
| --- | ---       |
|   0 |  32       |  <-- this is standard input
|   1 |  91       |  <-- this is standard output
|   2 |  91       |  <-- this is standard error
|   3 |           |
|   4 |  99       |
```

```
PID 71

| FD  | Open file |
| --- | ---       |
|   0 |  41       |  <-- this is standard input
|   1 |  97       |  <-- this is standard output
|   2 |  31       |  <-- this is standard error
|   3 |  85       |
|   4 |           |
```

OS keeps *system-wide* table of open files

```
| File                       | Mode     | Location    |
| ---                        | ---      | ---         |
|   /home/student/foo.txt    |  read    |    0        |
|   /usr/include/stdio.h     |  read    |   10        |
|   /usr/include/stdio.h     |  read    |    0        |
|   /home/student/project.c  |  write   |    0        |
|   /home/student/project.c  |  read    |    0        |
|   /home/student/log.txt    |  append  |  110        |
```

Modifying open file (for example, by reading a line) modifies it system-wide

Modifying a file descriptor (for example, by `close` or `dup`) is a per-process change

Note that the ideas of file descriptors, open files, and files (the things stored on disk) should be considered separately

- file descriptors (the `ints`) in one process have nothing to do with file descriptors in another process
- more than one file descriptor (in the same process or different processes) can point to the same open file
- more than one open file can point to the same file

System calls

System calls `dup`, `fork`, and `exec` do not create a new open file (that is the job of `open`)

`close` acts on file descriptors, not open files

`exec` can close a file descriptor depending on the flags associated with the descriptor

If you read about this elsewhere, you will see it is slightly more complicated

More than one file name can be associated with the same file on disk, so the file table actually keeps track of an **inode**

Don't worry about that for now -- we will cover file systems later in the semester