File Systems Part 1

Karen L. Karavanic
Portland State University CS 532

Winter 2020

Portland State
UNIVERSITY

What is a File?

- [To the User] A linear array of bytes
- [To the OS] A sequence of blocks
- A portable abstraction over the details of physical storage of persistent data
- A virtualization ("myData.txt") of the physical storage (HDD, SDD, tape, etc)
- Is "file" the only approach?
  - No
  - Examples: records, persistent objects, relational database – table rows

Key Unix Concept: Everything is a File

- Goal: Develop an abstraction for stored (persistent) data: The File
- Create a simple API for interacting with files
- Even better: use this API for as many things as possible
  - Ex.s: Directories, devices, pipes, processes (`/proc`), disks (`/dev/disk*`), non-things (`/dev/null`)
  - Common Operations: Read, write
- Needed to manage read and write: Create, Delete, Bookmark

Portland State UNIVERSITY

Key Unix Concept: Everything is a File

Note: The POSIX file interface <> the C stdio library interface

- The C library is a higher level set of functions implemented with POSIX calls
- The C library library uses a FILE* to identify each file

OS Management of Files

- OS must manage conflicts/sharing across processes
- Keep a count of how many processes are accessing a file:
  - open, close
  - Example of Readers/Writers problem
- Assign a unique name to each file: Directory structure
- Supply a token for use in system calls:
  - The file descriptor
    - int fd1 = open(argv[1], O_RDONLY);
    - int retval = close(fd1);

OS Management of Files

- Bookkeeping: File Metadata
  - See `stat` structure (text p. 14 Chapter 39)
  - Access this via `stat`() system call
- System Resilience: Synchronizing writes
  - `fsync()` forces data to be on device before program continues
  - `fcntl` (F_FULLFSYNC) forces device to actually save the data not just hold it in device memory

OS Management of Files

- The Open File Table
  - Each file descriptor is a separate entry, even if they refer to the same file:

```
fd1 = open("file", O RDONLY);
fd2 = open("file", O RDONLY);
```

  - Child process inherits the open file entries of the parent process
  - dup() and dup2() used to duplicate existing file descriptors
    - Example: redirection

OS Management of Files

Redirection: `cat > my.file`

```
fd = open("my.file", CREATE_FLAGS, CREATE_MODE);
if (fd == -1) {
   Printf ("failed to open file \n");
   Return 1;
}
if (dup2(fd, STDOUT_FILENO) == -1) {
   Printf ("failed to redirect stdout \n");
   Return 1;
}
if (close(fd) == -1) {
   Printf ("failed to close the file \n");
   Return 1;
}
```

Portland State UNIVERSITY

OS Management of Files

Redirection

- File descriptor table before redirection:

0	standard input
1	standard output
2	standard error

- File descriptor table after redirection:

0	standard input
1	write access my.file
2	standard error

Portland State UNIVERSITY