Introduction to Computer Systems COMP2300/6300

Systems Programming and Files

Eric McCreath

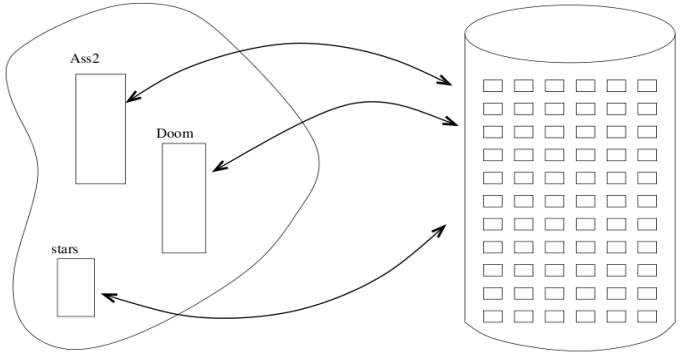
Research School of Computer Science
The Australian National University

Files

- Information used by a computer system may be stored on a variety of storage mediums (magnetic disks, magnetic tapes, optical disks, flash disks etc).
- A file system provides a uniform logical view of this information.

Files

- The operating system provides a mapping between the abstract logical units of storage, that is a file, and the physical storage device.



Files

- A layered approach is often used in implementing this mapping. The layers are:
  - lower levels — physical properties of the storage device,
  - intermediate levels — map the logical file concepts into physical device properties, and
  - upper levels — symbolic file names and logical properties of files.

Files

- A file is a named collection of related information.
- Files consists of a sequence of bits, bytes, lines, or records.
- The information in a file is generally defined by its creator.
- There is numerous types of files. These include: text, source, object, executable, binary, compressed, graphics image, video, data base, etc...
- Typically the operating system will keep track of attributes for each file. These attributes include: name, type, location, size, protection,temporal information about creation and use.

Unix:
•The 'stat' command can be used to obtain information about a file. There is also a 'stat' system call that provides information about a file.
•The 'file' command will attempt to determine the type of a file. This will use the 'magic' number at the beginning of the file.

File

- A file is an abstract data type. The operating system will provide a set of routines to uniformly view/manipulate this data type. A basic set of routines includes:
  - creation,
  - writing,
  - reading,
  - repositioning,
  - deletion,
  - truncating,
  - appending, and
  - renaming.
- From these basic operations other operations such as copying or printing may be performed.
- Rather than the operating system constantly searching the directory for each operation that is performed on file. Files are generally opened and information about the file is stored in the open file table. Then all file operation are performed using an index to this table. The process will close the file once the interaction with the file is complete.

open - man page

OPEN(2) Linux Programmer's Manual OPEN(2)

NAME
open, creat - open and possibly create a file or device

SYNOPSIS
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

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

int creat(const char *pathname, mode_t mode);

DESCRIPTION
Given a pathname for a file, open() returns a file descriptor, a small, nonnegative integer for use in subsequent system calls (read(2), write(2), lseek(2), fcntl(2), etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

read – man page

READ(2) Linux Programmer's Manual READ(2)

NAME
read - read from a file descriptor

SYNOPSIS
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.

write – man page

WRITE(2) Linux Programmer's Manual WRITE(2)

NAME
write - write to a file descriptor

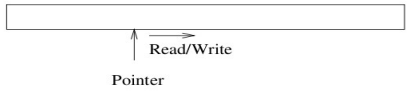
SYNOPSIS
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);

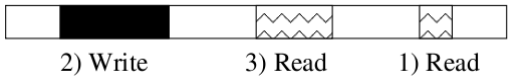
DESCRIPTION
write() writes up to count bytes from the buffer pointed buf to the file referred to by the file descriptor fd.

Accessing Files

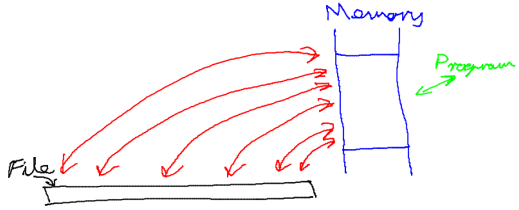
- There are three main approaches for accessing files:
  - Sequential access,



  - Direct access (or random), and



  - Memory mapped access.



Directories

- Directories store information such as name, size, and location of a file.
- Operations performed on a directory include:
  - search for a file,
  - create a file,
  - delete a file,
  - list a directory,
  - rename a file, and
  - traverse the file system.
- There are a number graphs the directories of a file system can form. Such as: single directory, two level tree, tree, acyclic graph, general graph.
- Given the size of these beasts and the time taken to traverse filesystems it can be a difficult task maintaining consistency.

links

- A single file can be referred to via multiple names.
- In UNIX the file name is contained in the directory, which is a special file, rather than the file itself.
- In UNIX the 'ln' command can be used to create either symbolic links, or hard links to files.
  - Symbolic links are like little files which contain the name of the file they point to, they can cross into different filesystem. The file does not also need to be there. When a symbolic links is deleted the file it points to is never deleted.
  - Hard links are just like another file name for the same file. They must be on the same filesystem. When the last of these hard links is deleted the file is also deleted. Generally a filesystem will not permit multiple hard links to directories.

Protection

- The file system must be protected from improper access.
- The types of access a user may be either granted or denied includes:
  - read,
  - write,
  - execute,
  - append,
  - delete, and
  - list.
- In most cases access to a file is controlled at these low-level functions. This in tern controls access to higher level functions.

Some Useful Unix Commands/Files

- /dev/null – a file that is like a black hole (content can go in but it does not get stored or fill up).
- /dev/zero – a 'file' that is a source of zeros.
- /dev/random – a source of random numbers.
- /dev/mem – a device file which gives access to physical memory.
- /dev/sda1 (or similar) – the raw blocks of the hard disk.
- dd – a command for converting and coping files.
- du – a command estimate the space used by a file.
- touch – a command to change a file's timestamp (also often used for creating an empty file).
- od – view the file's data in formats such as octal, hex, or as characters.

Sparse Files

- Files can have 'gaps' in the middle of them. These 'gaps' are considered zero and do not need to be stores by file system.
- The below creates a big empty file that does not take up much actual hard disk space:

dd if=/dev/zero of=sparcefile bs=1 count=0 seek=1G

File Names

- Tricky!
- Old file systems had limited length names, this can sometimes cause problems when files from a newer file systems are stored on old file systems.
- In Unix systems case is important in filenames, whereas, in windows it is not. This can cause problems as we move files and programs between different system. Also there is the slash and sloch difference.
- Spaces in file names can create havoc in scripts.

File Locking

- If multiple processes are using one file then locking may be useful for in preventing race conditions.
- In Unix file locking is advisory and can be done via the “flock” and “fcnlt” functions. Locking is not universally implemented in file systems.
- Some programs will use the creation of a file with just the process id in it to create a simple lock.

Some Special 'files'

- In Unix 'everything' is a file. So there is variety of things that look like files but don't have the backing storage of regular files.
- /proc or /sys entries are kernel generated 'files' for obtaining information about the running OS and setting preferences for the running OS.
- Device nodes (created with the command 'mknod') are either block or character device 'references'.
- Named pipes (creaded with the command 'mkfifo') are 'files' that can act as connecting pipes between two programs. So one program can write to the pipe while the other reads from it, the named pipe acts as a buffer (there is also unnamed versions of these that a program can create with 'pipe').

mmap

- A file can be mapped into memory, this can be done with the 'mmap' function. 'mmap' enables the file's contents to be view and modified in normal memory. There is two basic versions of mmap:
  - SHARED - where modifications to the memory of the mapped file are written back to the actual file, also other processes that map the same file see the same modifications.
  - PRIVATE – where the process has its own private copy of the file. Modifications are not written back to the file and other processes do not see any changes made to the memory by the process that made the private mmap (uses copy on write).

Systems Programming

- Take an action and then if a problem occurs ask for forgiveness, rather, than ask for permission and then when permission is granted take the action.
- When using a provided service attempt to use that service in a minimal and normal way. When providing a service attempt to get your code working for all the unusual and boundary cases.
- Take the time to fix all the warnings a compiler provides.
- 'valgrid' is your friend.
- Add code that checks for error conditions (i.e check that files open properly rather than just assuming they will).