

COMP1521 25T3

Week 7 Lecture 2

File Systems

Announcements

Test 5 and Test 6: due tomorrow

Assignment 2:

- Released today
- Walkthrough video coming Thu
- Mini assignment hackathon Friday 9am-11am
 - CSE Help (K17 G05)
 - Hosted by: Jennifer Yu, Alex Blackmore, Alex Kroh

Today's Lecture

- Recap
 - File Operations
- Filesystems
 - File metadata
 - Permissions
 - "stat" system call
 - Hard Links and Symbolic Links
 - Working with directories



Recap Files

Question 1: What is better to use to read in a file? `fgetc` or `fgets` or `fscanf`?

Question 2: If I successfully open a file using `FILE *f = fopen("data", "w");`

- A. What will happen if the file already exists? What if it doesn't?
- B. What is the difference between mode "a" and "w"?

Question 3: How many bytes would the following print to the file `f`:

- A. `fprintf(f, "%d", 255);`
- B. `fputc(f, 255);`

Recap: dup2

```
// What will be printed to terminal?  
// What will be inside data.txt?  
int main(void) {  
    FILE* fp = fopen("data.txt", "a");  
    int fd = fileno(fp); // Retrieve file descriptor from FILE*  
    close(STDOUT_FILENO);  
    dup2(fd, STDOUT_FILENO);  
  
    printf("Hello!\n");  
  
    return 0;  
}
```

Filesystems

File Systems

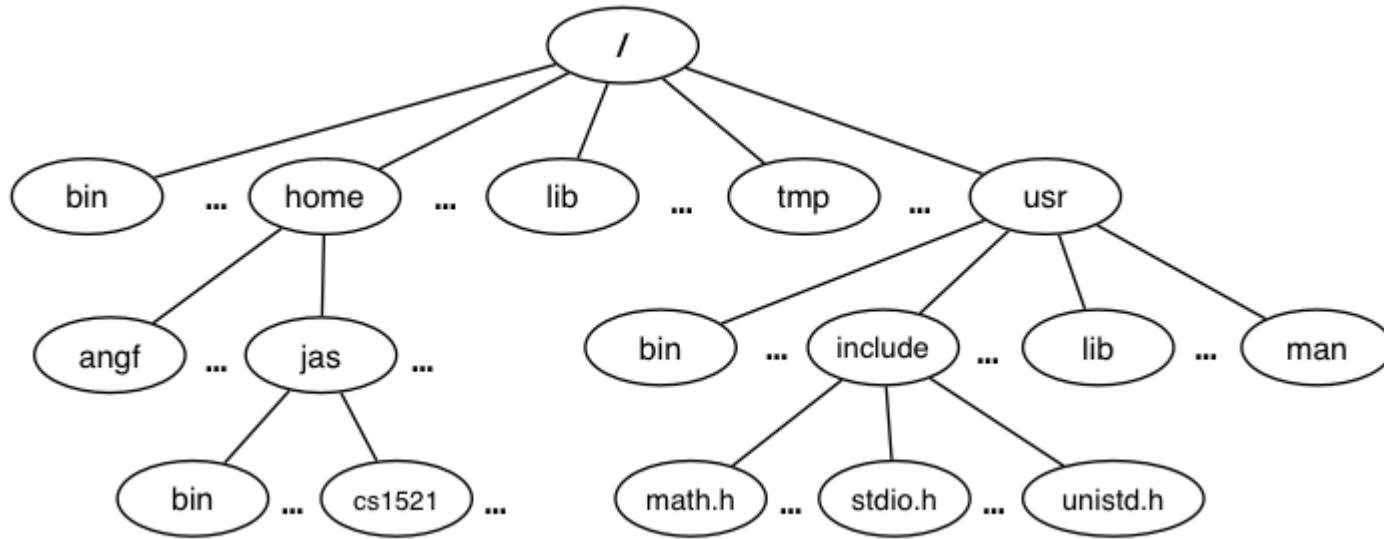
File systems manage stored data (e.g. on disk, SSD)

File = named sequence of bytes stored on device

Directory = named set of **files** or **Directories**

- File system maps name to location(s) on device
- File system maintains meta-data (e.g. permissions, timestamps)

Unix/Linux File System



Unix/Linux file system is tree-like

- symlinks actually make it a graph
- when traversing you may find infinite loops

Unix-like File Names

Sequences of 1 or more bytes

- filenames can contain any byte except
- **0x00** bytes (ASCII '\0') used to terminate filenames
- **0x2F** bytes (ASCII '/') used to separate components of pathnames.
- maximum filename length, depends on file system, typically 255

Two filenames have a special meaning:

- . current directory
- .. parent directory

Some programs (shell, ls) treat filenames starting with . specially.

Demo .bashrc

Paths and directories

Absolute pathnames start with a leading `/` and give full path from root e.g.
`/usr/include/stdio.h`

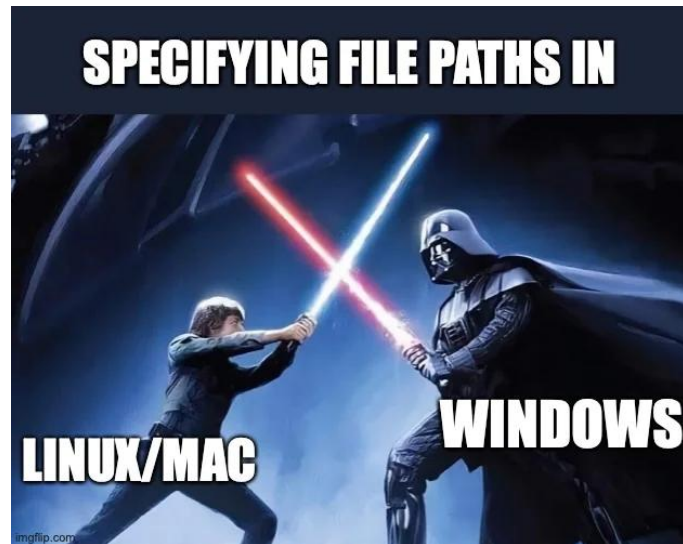
Relative pathnames do **not** start with a leading `/` e.g.

`../../another/path/prog.c`

`./a.out`

`main.c`

Demo `a.out`; `./a.out`



Current Working Directory

Every process (running program) has a **current working directory (CWD)**

- This is an absolute pathname
- This is the directory from where the process was run from
- Shell command **pwd** prints the **CWD**

Relative pathnames appended to CWD of process e.g.

- If **CWD** is `/home/z5555555/lab07/`
- And **relative path** is `main.c`
- **Absolute path** would be `/home/z5555555/lab07/main.c`

Everything is a file

- Originally, files only managed data stored on a magnetic disk
- Unix philosophy: *Everything is a file*
- File systems used to access
 - Files
 - Directories
 - Devices
 - System information
 - Other filesystems

Demo: everything is a file

Unix-like File Metadata

Metadata for file system objects is stored in **inodes** within the file system.

They hold:

- The **location of file contents** in the filesystem
- File **type** (regular file, directory, ...)
- File **size** in bytes
- File **ownership**
- File access **permissions** - who can read, write, execute the file
- **Timestamps** - times of file was created, last accessed, last updated

File system implementations often add complexity to improve performance

- e.g. data of small files might be stored in the **inode** itself.

Inodes

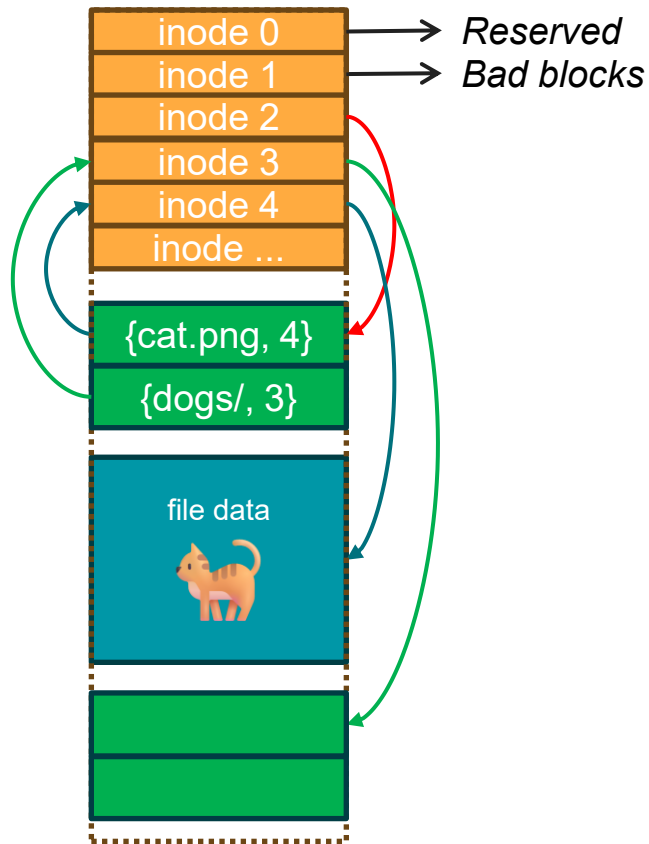
Files system has large table of inodes containing metadata

- inode number is the inodes id
 - Unique to file system (like student id within UNSW)
 - Other file systems may use same inode id to identify a different file
(At another university, your student id would refer to another student)

Directories are effectively a collection of (name, inode-number) pairs

- **ls -li** prints **inode-numbers**

Filesystem layout



 Disk Bytes

 inode entry stores file metadata and location of file data on disk

 Directory entry data maps file name to file inode number. Special files . and .. not shown.

 File bytes

File Access: Behind the scenes

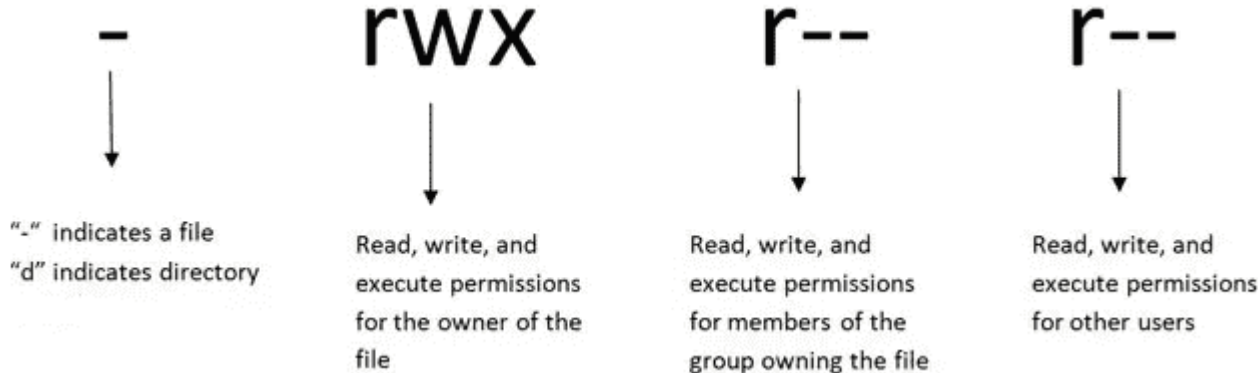
Access to files by name proceeds (roughly) as...

- **Open directory** and scan for **name**
 - if not found, "No such file or directory"
- If found as {**name**, **inumber**}, **access inode table**: inodes[**inumber**]
- Collect **file metadata** and...
 - Check file access permissions given current user/group
 - If don't have required access, "Permission denied"
 - Update access timestamp
- Use data in inode (size, location of bytes) to **access file** contents

File Permissions

Every file and directory in linux has read, write and execute permissions (access rights) for each of the following user groups:

- **user**: the file's owner
- **group**: the members of the file's group
- **other**: everyone else
- type **ls -l** on command line to see



Demo permissions

File Permissions: read, write, execute

	Read	Write	Execute
File	View contents of file	Modify file	Run as executable
Directory	View names of file e.g. use ls	Create, delete, rename files	Can cd into it. Also needed to access (read, write, execute) items in directory

Modifying Permissions

You can think of permissions as a set of bits and then each 3 bits as an octal digit. e.g.

<code>rwX</code>	<code>r-X</code>	<code>r-X</code>
<code>111</code>	<code>101</code>	<code>101</code>
<code>7</code>	<code>5</code>	<code>5</code>

You can use the **chmod** command to set the permissions of a file or directory using the desired 3 digit octal code. e.g.

```
$ chmod 755 f.txt
```

Demo permissions

Hard Links and Symbolic Links

File system **links** allow multiple paths to access the same file

- Hard links

- Multiple names referencing the same inode number (file)
- Hard links must be on the same filesystem (inodes unique to filesystem)
- Can not create a [extra] hard link to directories
- All hard links to a file have equal status (metadata stored in inode)
- File destroyed when last hard link removed (link count stored in inode)

Assuming 'fileA' already exists, this creates a hard link named 'fileB'

```
ln fileA fileB
```

Hard Links and Symbolic Links

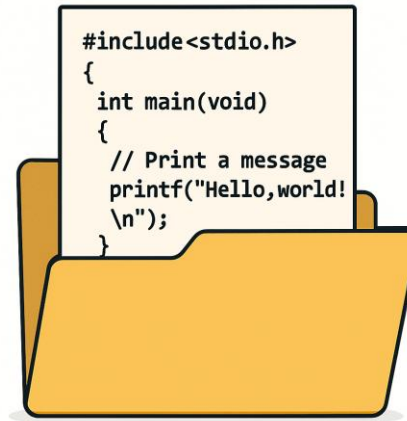
File system **links** allow multiple paths to access the same file

- Symbolic links (symlinks)
 - Point to another path name
 - Accessing the symlink (by default) accesses the file being pointed to
 - Symbolic link can point to a directory
 - Symbolic link can point to a pathname on another filesystems
 - Symbolic links don't have permissions (not needed - they are just a pointer)

Assuming 'fileA' already exists, this creates a symbolic link named 'fileB'

```
ln -s fileA fileB
```

Show me the code!



```
#include<stdio.h>
{
  int main(void)
  {
    // Print a message
    printf("Hello,world!
    \n");
  }
}
```

C library wrapper for stat system call

```
int stat(const char *pathname, struct stat *statbuf);
```

- Returns metadata associated with **pathname** in **statbuf**
- Metadata returned includes:
 - inode number
 - type (file, directory, symbolic link, device)
 - size of file in bytes (if it is a file)
 - permissions (read, write, execute)
 - times of last access/modification/status-change
- **returns -1 and sets errno** if metadata not accessible

C library wrapper for stat system call

```
int lstat(const char *pathname, struct stat *statbuf);
```

- same as stat() but doesn't follow symbolic links
 - in other words gives you metadata about the symbolic link
 - and not the file it links to
 - important not to get stuck in infinite loops

```
int fstat(int fd, struct stat *statbuf);
```

- same as stat() but gets data via an open file descriptor

See **man 2 stat** # For stat syscall with example

man 3 stat # For struct stat

man 7 inode # For masks and macros and inodes in general

definition of struct stat

man 3 stat

```
struct stat {  
    dev_t      st_dev;      /* ID of device containing file */  
    ino_t      st_ino;      /* Inode number */  
    mode_t     st_mode;     /* File type and mode */  
    nlink_t    st_nlink;    /* Number of hard links */  
    uid_t      st_uid;      /* User ID of owner */  
    gid_t      st_gid;      /* Group ID of owner */  
    dev_t      st_rdev;     /* Device ID (if special file) */  
    off_t      st_size;     /* Total size, in bytes */  
    ...  
};
```

st_mode field of struct stat

man 7 inode

st_mode is a bitwise-or of these values (and others):

S_IFLNK	0120000	symbolic link
S_IFREG	0100000	regular file
S_IFDIR	0040000	directory
S_IRUSR	0000400	owner has read permission
S_IWUSR	0000200	owner has write permission
S_IXUSR	0000100	owner has execute permission
S_IRGRP	0000040	group has read permission
S_IWGRP	0000020	group has write permission
S_IXGRP	0000010	group has execute permission
S_IROTH	0000004	others have read permission
S_IWOTH	0000002	others have write permission
S_IXOTH	0000001	others have execute permission

Code demos stat.c

stat.c

Another good sample program at bottom of man 2 stat

Making a directory

```
int mkdir(const char *pathname, mode_t mode);
```

- Returns 0 if successful, **returns -1 and sets `errno`** otherwise
 - for example: `mkdir("newDir", 0755)`
- If **pathname** is e.g. ``a/b/c/d``
 - All of the directories ``a``, ``b`` and ``c`` must exist
 - directory ``c`` must be writable to the caller
 - directory ``d`` must not already exist
- the new directory contains two initial entries
 - ``.`` is a reference to itself
 - ``.`` is a reference to its parent directory

Demo: `mkdir.c`

Opening and Reading directories

// open a directory stream for directory name

```
DIR *opendir(const char *name);
```

// return a pointer to next directory entry

```
struct dirent *readdir(DIR *dirp);
```

// close a directory stream

```
int closedir(DIR *dirp);
```

Found in man 3

Demo list_directory.c

Useful Linux (POSIX) functions

`chmod(char *pathname, mode_t mode)` // change permission of file/...

`unlink(char *pathname)` // remove a file...

`rename(char *oldpath, char *newpath)` // rename a file/directory

`chdir(char *path)` // change current working directory

`getcwd(char *buf, size_t size)` // get current working directory

`link(char *oldpath, char *newpath)` // create hard link to a file

`symlink(char *target, char *linkpath)` // create a symbolic link

Demo: `chmod.c` `rm.c` `rename.c` `my_cd.c` `getcwd.c` `nest_directories.c` `many_links.c`
`chain_links.c`

Reach Out

Content Related Questions:

[Forum](#)

Admin related Questions email:

cs1521@cse.unsw.edu.au



Student Support | I Need Help With...

My Feelings and Mental Health

Managing Low Mood, Unusual Feelings & Depression



Mental Health Connect

student.unsw.edu.au/counselling
Telehealth



In Australia Call Afterhours UNSW Mental Health Support Line

1 300 787 026
5pm-9am



Mind HUB

student.unsw.edu.au/mind-hub
Online Self-Help Resources



Outside Australia Afterhours 24-hour Medibank Hotline

+61 (2) 8905 0307

Uni and Life Pressures

Stress, Financial, Visas, Accommodation & More



Student Support Indigenous Student Support

— student.unsw.edu.au/advisors

Reporting Sexual Assault/Harassment



Equity Diversity and Inclusion (EDI)

— edi.unsw.edu.au/sexual-misconduct

Educational Adjustments

To Manage my Studies and Disability / Health Condition



Equitable Learning Service (ELS)

— student.unsw.edu.au/els

Academic and Study Skills



Academic Language Skills

— student.unsw.edu.au/skills

Special Consideration

Because Life Impacts our Studies and Exams



Special Consideration

— student.unsw.edu.au/special-consideration