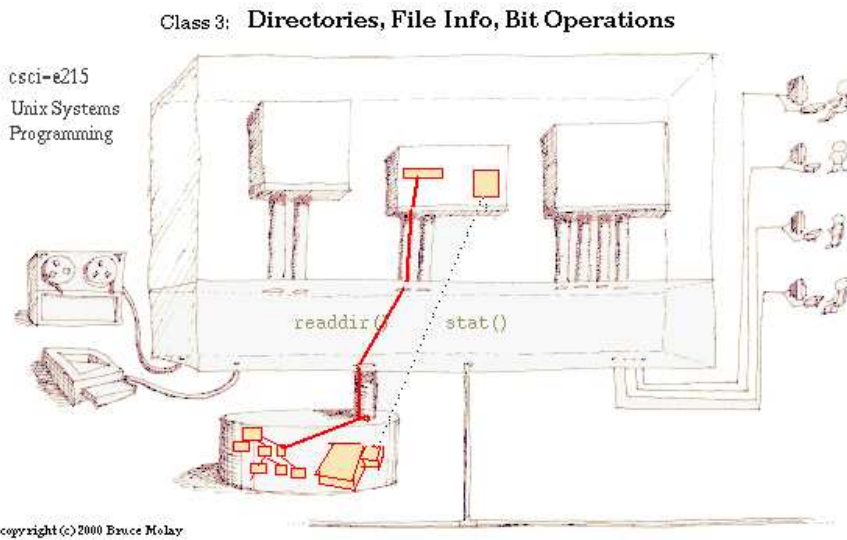


page000.html

000

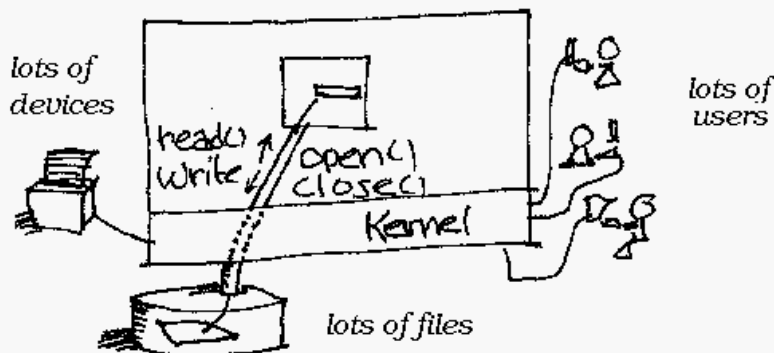


page001.html

001

Class 3: Directories, File Info, Bit Ops

Recall:



Last time we explored 'who' (list current users) and saw:

- how to process the contents of files using
 open, read, write, close, lseek
- the importance of buffering to make I/O more efficient

Today, we explore 'ls' (list current files) and shall see:

- 1) How to read a directory
- 2) Properties of files (size, owner, permission, type..)
- 3) How file properties are represented and how to process those representations

But first -- One last comment about buffering

What is the difference between open/fopen, getc/read, putc/write, and close/fclose?

Answer: fopen, getc, ... are library functions that provide buffering for open, read, write, close.

Somewhere in /usr/include are the details of a *FILE* *
 /usr/include/x86_64-linux-gnu/bits/types/struct_FILE.h

Writing ls (with our 3-step method)

1) What does ls DO?

ls lists the contents of directories
 AND displays information about files.

Examples

ls

dir

ls /tmp

ls hello.c

file

ls -l

lists and shows info

ls -l /etc

ls -l hello.c

type

```
-rw-r--r-- 1 lib215 extlib 8415 2010-02-08 10:23 hello.c
permisss owner group size modtime name
```

Other ls options

<code>ls -a</code>	<i>shows "." files</i>
<code>ls -a /</code>	
<code>ls -lu</code>	<i>shows access time</i>
<code>ls -s</code>	<i>shows size in blocks</i>

Note: arguments to ls may be:

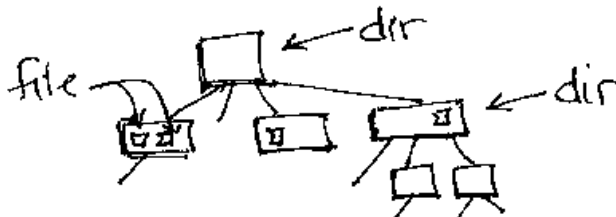
<i>none</i>	<i>=> show current directory</i>
<i>filenames</i>	<i>=> show info about files</i>
<i>dirs</i>	<i>=> show directory contents</i>

so, We need to learn:

- How to read a list of files from a directory*
- How to get info about files*
- How to present info about files*

3 Brief Review of the File System Tree

The disk is organized as a tree of directories, each of which contains files and/or directories.



The commands `cd`, `pwd`, `ls` allow you to explore a FILE SYSTEM.

4 So... writing *ls* should be easy:

```

    open directory
  --> read entry      -----+ EOD?
  --- display entry   |
    close <-----+

```

Maybe it is just like writing the who command.

Is a directory like a file?

5 Q. What IS A Directory?

A: A special kind of file that contains a list of files and or other directories.

Note: Every directory contains the items "." and ".."

6 Q: Can We Use *open*, *read*, *close* to Read This List?

A: yes. The *open(2)* system call can open a dir
But...

A: no. The *read(2)* system call returns -1 (see man)

Let's Try: *cat / ; more /tmp ; od -c /etc*

Code demo: *open_a_dir.c*

Ans2: Even if you could, you don't want to, because different types of disks have different directory formats.

For example, Unix can read CDROMS, VFAT floppy disks, USB flash drives, each with a different format of directory content. It is better to rely on the kernel to know about formats.



OK, How DO I Read a Directory ?

```
man -k direct  
man -k direct | grep read
```

Ans: `opendir()`, `readdir()`, `closedir()`.

Using these Directory Functions

```
#include <dirent.h>
```

```
opendir (name)
```

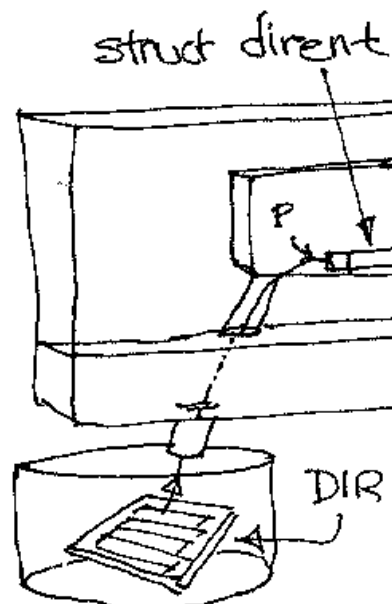
*Creates a connection.
Returns a DIR **

```
readdir (DIR *)
```

*Gets next record from
dir. Returns a
struct dirent **

```
closedir (DIR *)
```

Closes the connection.



Writing ls1.c

```
main()
    opendir
    while ( readdir )
        print d_name

    closedir
    return 0
```

Let's compile and run the code...

Note how permission bits (rwx) affect opendir/readdir

How Well Does ls1.c Work?

a) No columns

solution: doable, interesting

b) Not sorted

solution: use qsort() on array

c) No -a option (lists all files)

solution: simple string handling
write ls1-nodots. to see how easy it is

d) No -l option

solution: hmmm..
is this as easy as -a ?

9 How Do We Add the -l Option ?

Note: The other info is NOT in the directory

Q: What Does -l Display?

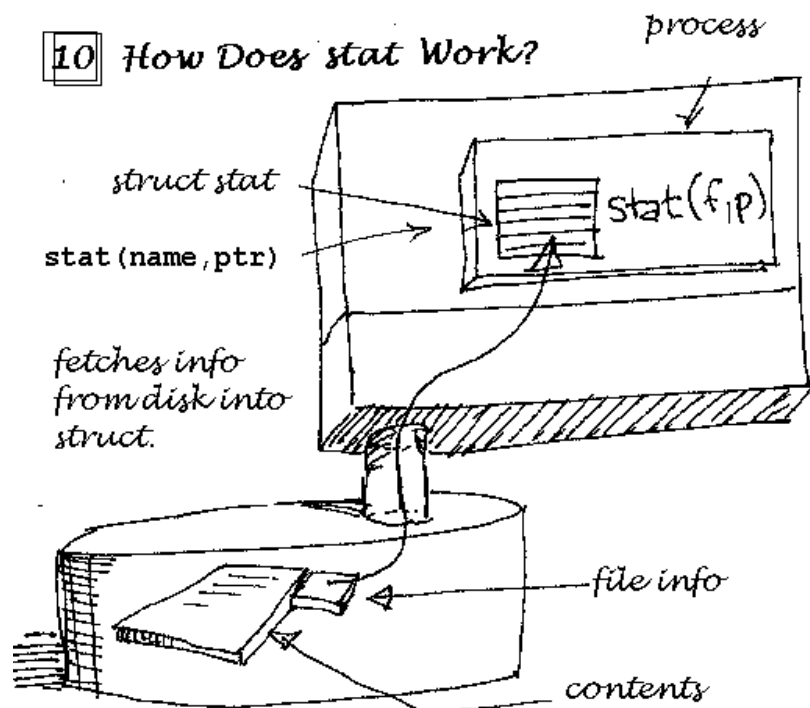
A: modtime, size, owner, group, links,
type, permission,...

Q: Where do we learn about reading
these properties?

A: Search the manual

Result: the stat() system call

10 How Does stat Work?





What Do We Get From `stat` ?

The struct contains:

```
st_mode,  
st_uid,  
st_gid,  
st_size,  
st_links,  
st_mtime, ...
```

So.. We Just:

- 1) Read a `dirent` for the filename,
- 2) Use `stat` to get file info for that name
- 3) Display the items in the struct

(code: `stat1.c`)



How'd We Do with `stat1.c`?

filename? Perfect!

filesize? Perfect!

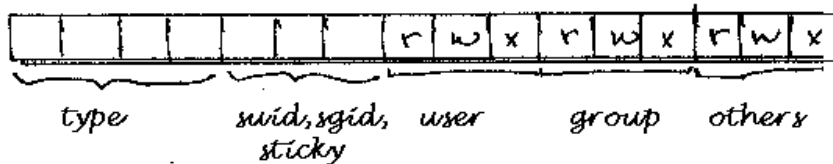
mod time: Use `ctime()` or `localtime()`
to convert

owner, group: These are stored as
numbers. We need
to map numbers to
names.

type? ??
permission? ?? in the mode

13 Where Are Type and Permissions Stored?

A: `st_mode` is a 16-bit value. Fields are encoded in sections of this value:



Four bits
means 16
patterns:

Each file
type has a
pattern.

Each permission is
on or off, so a
single 1 or 0 will
do.

Programming for File Types and File Permissions -- Subfield Coding

* Subfield coding is not magic:

617-222-3333 phone
011-22-4567 social security #
102.102.34.34 IP number

* A computer stores integers as a string of bits:

215 =

0	0	1	1	0	1	0	1	1	1
---	---	---	---	---	---	---	---	---	---

? =

0	0	0	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---	---	---

* As with telephone numbers, people define the size and coding in each region.

Programming for File Types and File Permissions -- Masking to Read Fields

Q: How do I examine a bit or subfield?

A: Use "bitwise AND: &" to MASK the rest

Ex:

1	0	1	1	0	1	0	0	1
0	0	0	1	1	1	0	0	0

 value 551
 &

0	0	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---	---

 mask 070

0	0	0	1	0	1	0	0	0
---	---	---	---	---	---	---	---	---

 result 050 — octal base 8

- * The & operator compares values bit by bit. Its result contains 1's only where both numbers have a 1.
- * 1's in the mask allow a value to 'show through'
- * skill: expressing binary numbers in base 8: group by 3s

Testing Permission Bits with Masks:

Picture:

user			group			others		
1	0	1	1	0	1	0	0	1
0	0	0	0	0	0	1	0	0

 = 004

Code:

```
if ( st_mode & 004 )
    printf("readable by others");
if ( st_mode & 002 )
    printf("writable by others");
```

Testing File Type with Masks

	type			user			group			others		
mode												
& mask	1	1	1	1	0	0	0	0	0	0	0	0
octal value	1	7		0			0		0		0	

- * Masks are in <sys/stat.h> and/or <bits/stat.h> or <linux/stat.h> or somewhere else in /usr/include

```
-rwxr-wr-w 2 dce-lib215 Domain Users 787 Feb 12 2020 ls1.c
```

[14] How Do We Convert UID to Name?

Check the manual. We learn about:

`/etc/passwd` -- traditional Unix user database
but not good a network of machines

`ldapsearch -x "(uid=lib215)" | more`

Asks a server on the network for user information

Fact: Unix identifies users by *UID* (*userID*) and stores that value. The `passwd` and `ldap` databases associate a string to a *UID* number.

`getpwuid(3)` looks up user information given the *UID*.

[15] How Do We Convert GID to Name?

Use `getgrgid(3)` passing the *GID* as an argument.
Get back a pointer to a struct group.

[16] `stat2.c` Now Has Correct Format

type, permissions, links, owner, group,
size, mod time, name!

[17] Writing `ls2.c`

Combine `ls1.c` to get names with `stat2.c`
to get info.