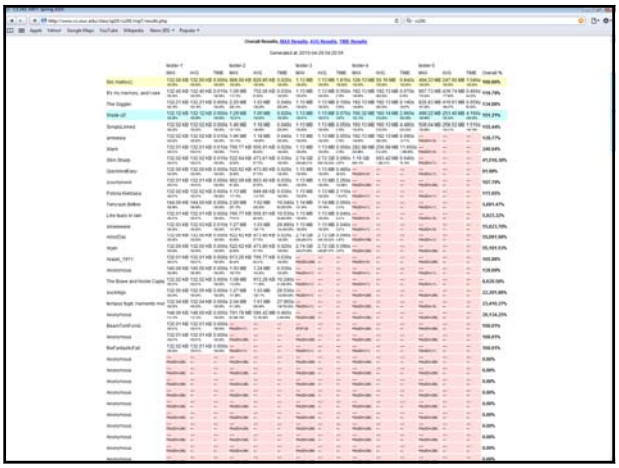
CS 241 Section
(04/29/2010)

In Section Today…

- MP7
- HW3 Clarifications
- File System Topics

MP7

- In MP7, the task is simple: reimplement malloc(), calloc(), realloc(), and free().
- As part of MP7, a contest is running comparing all submissions.



HW3 Clarifications

- The professors have announced a few clarifications on HW3…
- *(All these clarifications are also posted online on the "MP & Homeworks" page of the course website.)*

HW3 Clarifications

- Question 1(B):
  - When implementing alarm(20) without the alarm() call, you can just use sleep(20) call to wait 20 seconds.

HW3 Clarifications

- Question 3(C):
  - When calculating statistics regarding a page table (eg: given a 64-bit logical address, 16 KB pages, …), you may want to refer to the lecture on April 15 in the section "Multilevel Page Tables"
- Quesiton 3(C), Part 3:
  - Instead of just the first level of the page table needing to fit into the page table, all levels must.

HW3 Clarifications

- Question 4(C):
  - When evaluating the FIFO, SSTF, SCAN, and C-SCAN algorithms, the "total time" to serve the requests should include only the seek times (5ms /cylinder moved).

HW3 Clarifications

- Question 5(A):
  - When writing the bitmap overview after a series of file write/delete requests, remember that:
    - The search for free blocks always begin at the lowest number blocks
    - Files to not have to be sequentially ordered/contiguous.
    - This bitmap should not include space needed for i-nodes, we only care about data blocks in this question.

HW3 Clarifications

- Question 5(C):
  - For [Part 1] and [Part 2], we care only about the data blocks allocated and not about the i-nodes allocated.
  - One of the lines has a typo, where a file4 should be created inside the folder “subdir”

HW3 Due Tomorrow

- Homework #3 is due in your svn at 11:00am, the beginning of class.
- No late submissions.

UNIX File Systems

inode: per-file data structure

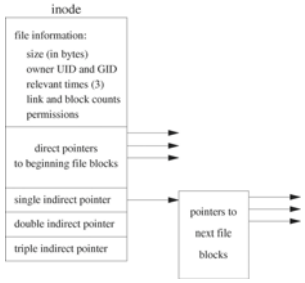
Advantage

Efficient for small files

Flexible if the size changes

Disadvantage

File must fit in a single disk partition

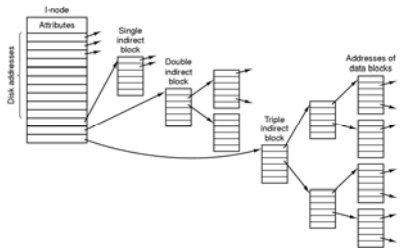


inodes

- Stores almost all “metadata” about a file:
  - Size, owner, permissions, creation time, access time, etc, etc
  - Also stores a “link count”, which indicates how many **hard links** exist to the file. (We’ll look at what a hard link is later.)
  - Also stores the “block index table” that was talked about extensively in lecture.

inodes

Block index table, illustrated:

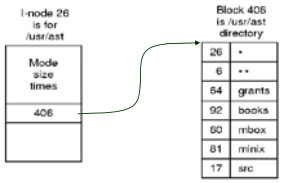


Directory Files

- The one major attribute that inodes do NOT store is the **name of the file**.
- Instead, the names of the file inside a directory are stored in a file that describes the contents of a directory.
  - Remember, a directory is a file too!

Directory Files

- Instead of storing data, directory files store a table that associates a name of the file with the files inode.



Directory Files

- The command "ls –i" lists the contents of the directory file. This is "ls –i /" on a csil-linux-ts box:
  - 3407873 bin/
    2 boot/
    1377 dev/
    3964929 etc/
    3735553 home/
    173712 jawt.h@
    173713 jawt_md.h@
    173710 jni.h@
    …

…back to inodes.

- In the inode files, the "metadata" about the file, there is a series of bytes that describe the permissions of the file.
  - Who can read the file?
  - Who can write to the file?
  - Who can execute the file?

File Permissions

- In UNIX, the file permissions system is relatively basic.
  - Each file has a single owner and a single group associated with it.
  - Each file also has permissions associated with itself for the owner, members of the group the file is in, and for everyone else.

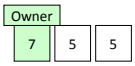
File Permissions

- These permissions are stored as a three-octal-digit number (000 to 777).

7	5	5

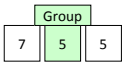
File Permissions

- The most-significant number is the owner's permission.



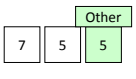
File Permissions

- The middle number is the group's permission.



File Permissions

- The least-significant number is everyone else's permission.



File Permissions

- Each octal number is simply three bits: a read bit, a write bit, and an execute bit.

	7	5	5
Read:	1	1	1
Write:	1	0	0
Execute:	1	1	1

File Permissions

- Thus:
  - 755 means “everyone can read and execute by file, but only the owner can write to (edit) my file”
  - 644 means “everyone can read my file, only the owner can write to my file, and no one can execute it”
  - 660 means “only members of the file’s group and the file’s owner may read or edit the file; others cannot even read it”

File Permissions on Directory Files

- If the inode contains “metadata” about a directory file, the execute permission takes on a different meaning:
  - The execute bit determines if a user/group/everyone can get a list of the contents of the directory.
  - If you don’t have “execute access” on a directory, you cannot “ls” it or “cd” into it, since you’re unable to access the list of files in the directory.

File Permissions

- The command “ls –l” will list various permissions information about a file.

```
lrwxrwxrwx   1 root root    27 Oct 22  2009 local -> …
drwx------   2 root root 16384 Oct 22  2009 lost+found/
drwxr-xr-x   2 root root  4096 Mar 11  2009 media
drwxr-xr-x   2 root root  4096 Oct 26  2009 misc/
drwxr-xr-x   2 root root  4096 Mar 11  2009 mnt/
drwxr-xr-x   6 root root  4096 Jan 15 17:04 opt/
dr-xr-xr-x 717 root root     0 Apr 23 05:08 proc/
drwxr-xr-x   2 root root  4096 Oct 22  2009 reserved/
drwxr-x---  16 root root  4096 Apr 27 12:09 root/
```

Hard Links

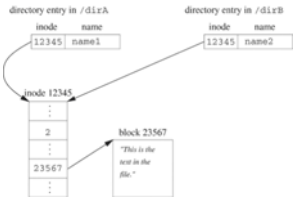
- You know about inodes, you know about permissions, but how do we “link” an inode to a file?
- A **hard link** is simply a direct directory file reference to an inode.
  - Remember, directory files simply list (file name <-> inode) pairs.

Hard Links

- To create a hard link, you simple use:
  ln <target> <link name>

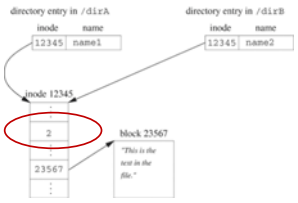
Hard Links

- If you're at "/", "/dirA/name1" exists, and you want to create a hard link to that file at location "/dirB/name2":
  - ln /dirA/name1 /dirB/name2



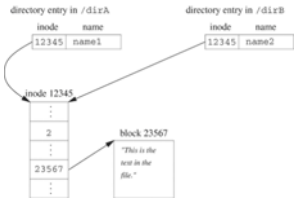
Hard Links

- If you're at "/", "/dirA/name1" exists, and you want to create a hard link to that file at location "/dirB/name2":
  - ln /dirA/name1 /dirB/name2
  - Remember: inodes store a "link count"



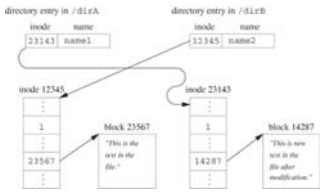
Hard Link Example (contd)

Q: What happens if `/dirA/name1` is deleted and recreated?



Hard Link Example (contd)

A: `/dirA/name1` and `/dirB/name2` are now two distinct files.



Symbolic Links ("Soft Links")

- **Q:** What if we want the file to only be in one single directory? Having pointers in different directories to the exact same file seems like it could get messy…

Symbolic Links ("Soft Links")

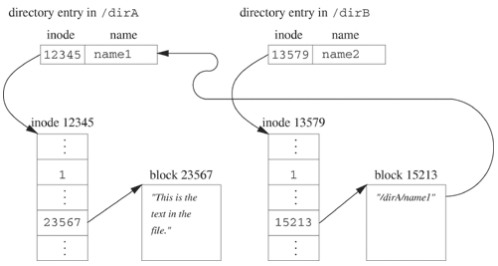
- **A:** Symbolic links (aka "soft link") are "shortcuts", which references an existing file's path and NOT the existing file's inode number.
- A soft link is a special file that simply stores a path to another file/directory.
  - Eg: "/home/class/sp10/cs241/"

Symbolic Links ("Soft Links")

- The command "ln" also creates soft links:
  ln -s <existing file> <name of soft link>

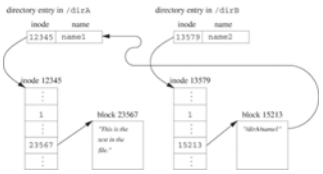
Symbolic Links ("Soft Links")

- ln -s /dirA/name1 /dirB/name2



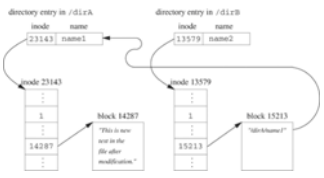
Soft Link Example (contd)

Q: What happens if `/dirA/name1` to is deleted and recreated?



Soft Link Example (contd)

A: `/dirA/name1` has a different inode, but `/dir/name2` still links to it.



Key Differences

- If you delete a file:
  - With hard links, since the reference count is still greater than 0, the file itself is not deleted and still can be referenced to at a different location.
  - With soft links, if you delete the file that's being pointed to, that file is now deleted and the link will be unable to resolve (eg: "unable to fild /path/to/file").

File Permissions

- The command “ls –l” will show the link count in the second column:

```
– lrwxrwxrwx   1 root root    27 Oct 22  2009 local -> …
  drwx------   2 root root 16384 Oct 22  2009 lost+found/
  drwxr-xr-x   2 root root  4096 Mar 11  2009 media
  drwxr-xr-x   2 root root  4096 Oct 26  2009 misc/
  drwxr-xr-x   2 root root  4096 Mar 11  2009 mnt/
  drwxr-xr-x   6 root root  4096 Jan 15 17:04 opt/
  dr-xr-xr-x 717 root root     0 Apr 23 05:08 proc/
  drwxr-xr-x   2 root root  4096 Oct 22  2009 reserved/
  drwxr-x---  16 root root  4096 Apr 27 12:09 root/
```

File Permissions

- For most user storage space, it makes sense that the link count (number of hard links) is only 1:

```
– -rw------- 1 wfagen2 csGrads 8254 Apr 27 19:11 alloc.c
– -rw------- 1 wfagen2 csGrads    9 Apr 27 19:11 ALLOCNAME.txt
– -rw------- 1 wfagen2 csGrads 3581 Apr 29 06:26 contest-alloc.c
– -rw------- 1 wfagen2 csGrads  253 Apr 29 06:27 contest.h
– -rw------- 1 wfagen2 csGrads  842 Apr 27 19:03 Makefile
– -rw------- 1 wfagen2 csGrads 3427 Apr 29 06:27 mcontest.c
– -rw------- 1 wfagen2 csGrads 1387 Apr 27 16:57 mreplace.c
– -rw------- 1 wfagen2 csGrads  129 Apr 27 15:51 README.txt
– drwx------ 2 wfagen2 csGrads   64 Apr 27 15:51 testers/
```

Overview of Topics as a PDF…

- The course professors have also prepared an overview of these topics, attached with the HW #3 clarifications.