CS241 #29 – Files and Filesystem

Design a file system! What are your design goals?

Paths "." ".." "..." ? "foo1/foo2" vs "/bar1/bar2" ?

Example of relative path?

Can you simplify `a/b/../c/.` ?

What is an absolute path?

What if I need the **Absolute Canonical** Path?

```
int main() {
  char* path = realpath("./../../",NULL);
  puts(path);
  free(path);
  return 0;
}
```

Casestudy: Use realpath to secure file access of a webserver?

```
  char* canonbasepath = realpath(basedir);
  asprintf( &file, "%s/%s", canonbasepath , url );
  puts(file)

  if( strncmp(canonbasepath, file, strlen(____________) ) )  Access denied!
```

File systems are block-based. Why make disk blocks the same size as memory pages?

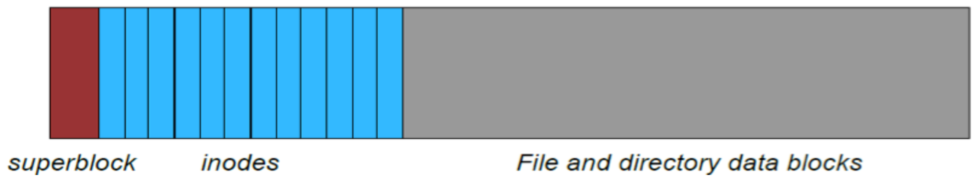
What do we want to store for each file?

What is an inode? Which of the above items is stored in the inode?

Connection between inode & `stat`?

Case study: Disk layout of a ext2 filesystem:

Once formatted, disk blocks are used for i) superblocks, ii) fixed array length of inodes and iii) data blocks:



Ext2 supports 32TB storage.

Ext3 (2001): supports journaling.

Ext4 (2008): Performance + (16TB files) upto 1EB (1024 Peta B) storage.

How does an inode index the file contents?

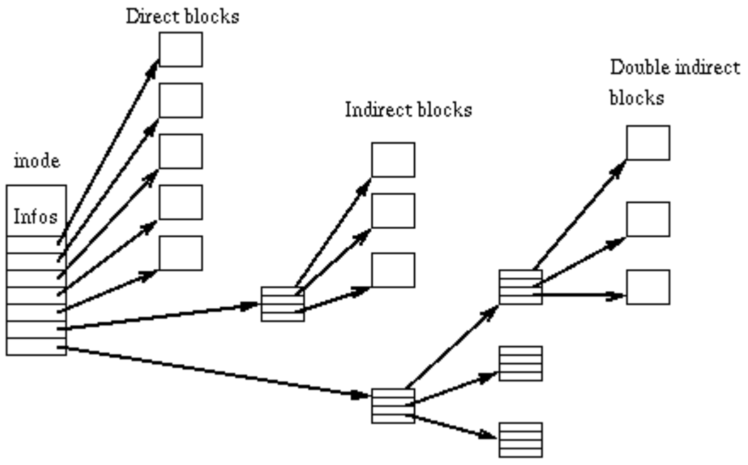
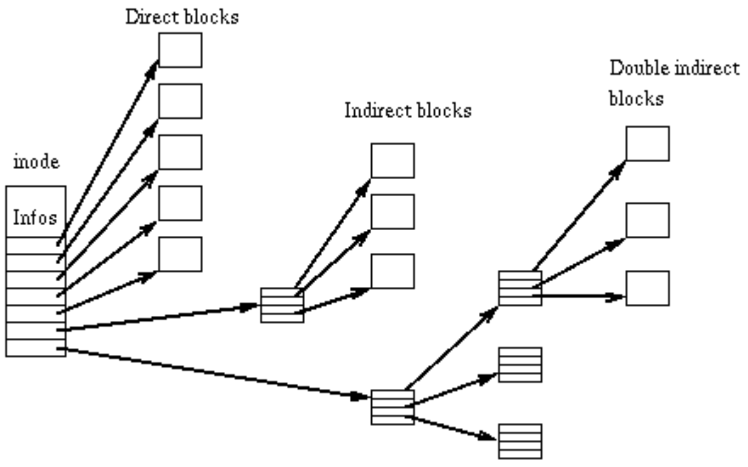


Image source: http://en.wikipedia.org/wiki/Ext2

How many pointers can you store in each indirection table? Assume 64 bit addressing. Each block is 4KB.

In the following examples assume an ext2 filesystem with 4KB disk blocks. Files use 10 direct blocks and 2^32 addressable disk blocks.

1. How many indirect blocks can be referenced?
2. How large is the file (in blocks of data) if the indirect block index is half full?
3. What is the total number of blocks used (ignore the inode entry in the inode array)?



What about huge files? Do we need triple indirect? Quad indirect?