Filesystems

Inodes

The data on the disk is in *disk blocks*, fixed size areas on the disk, e.g., 512 or 1024 bytes

Filesystems

Inodes

The data on the disk is in *disk blocks*, fixed size areas on the disk, e.g., 512 or 1024 bytes

Having a fixed size allows for easy and fast allocation and deallocation

Filesystems

Inodes

The data on the disk is in *disk blocks*, fixed size areas on the disk, e.g., 512 or 1024 bytes

Having a fixed size allows for easy and fast allocation and deallocation

This is similar to pages in memory; but now physical location of blocks *is* important as discs are mechanical devices

Filesystems

Inodes

The data on the disk is in *disk blocks*, fixed size areas on the disk, e.g., 512 or 1024 bytes

Having a fixed size allows for easy and fast allocation and deallocation

This is similar to pages in memory; but now physical location of blocks *is* important as discs are mechanical devices

Whole numbers of blocks are always allocated to files

Filesystems

Inodes

The data on the disk is in *disk blocks*, fixed size areas on the disk, e.g., 512 or 1024 bytes

Having a fixed size allows for easy and fast allocation and deallocation

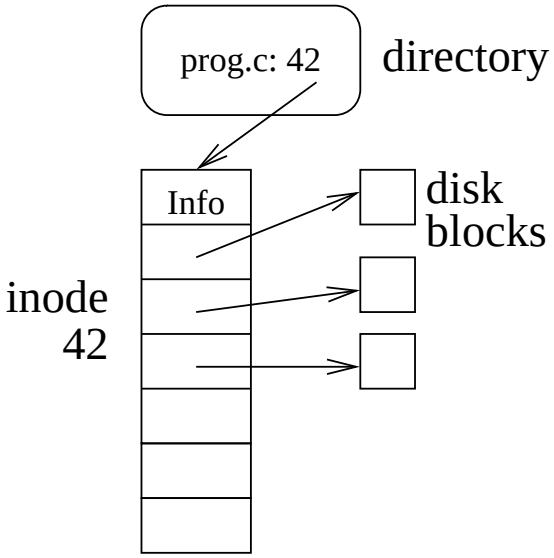
This is similar to pages in memory; but now physical location of blocks *is* important as discs are mechanical devices

Whole numbers of blocks are always allocated to files

This can lead to wastage, e.g., a 1025 byte file might need two blocks, but uses just over half of the space. Though there are lot of tricks in real filesystems to avoid the worst of this

Filesystems

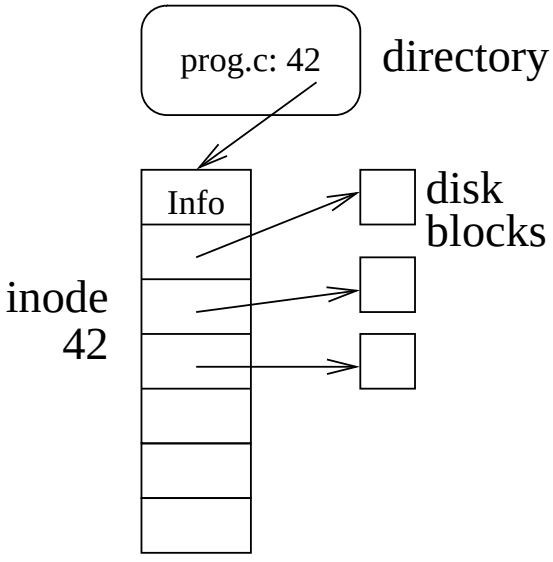
Inodes



An inode is of fixed size and has space for, say, 10 block pointers

Filesystems

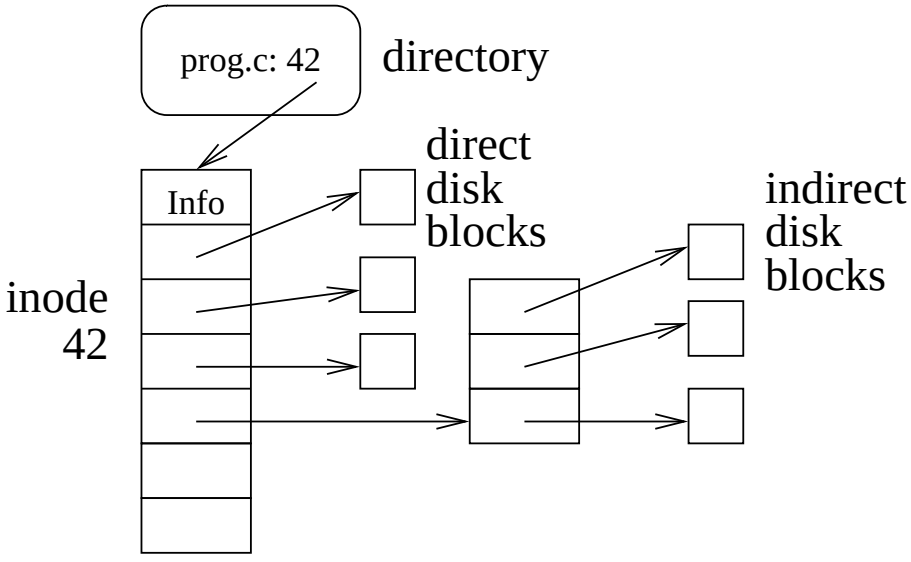
Inodes



But then you can't have files bigger than $10 \times 1024 = 10$KB

Filesystems

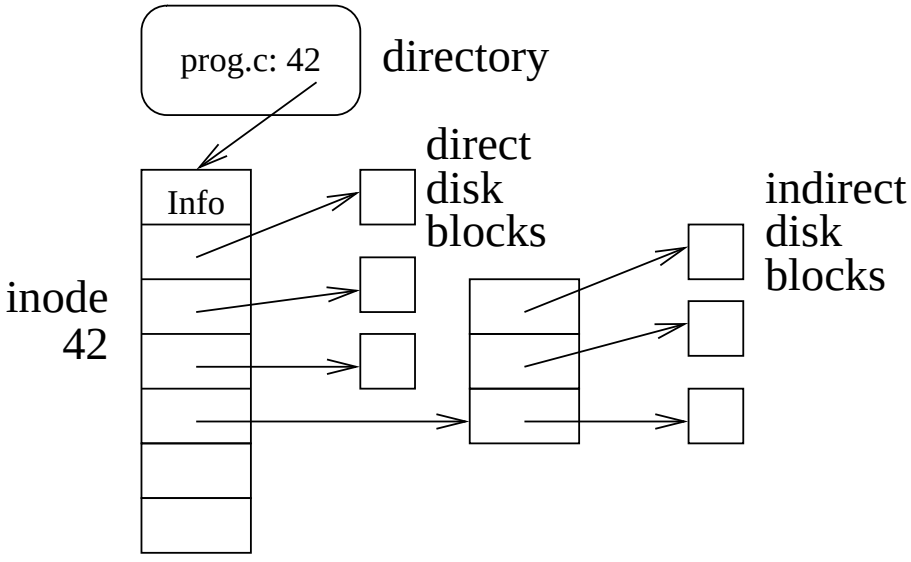
Inodes



So for such files we have an *indirect block*, that contains a pointer to an array of 256, say, block pointers

Filesystems

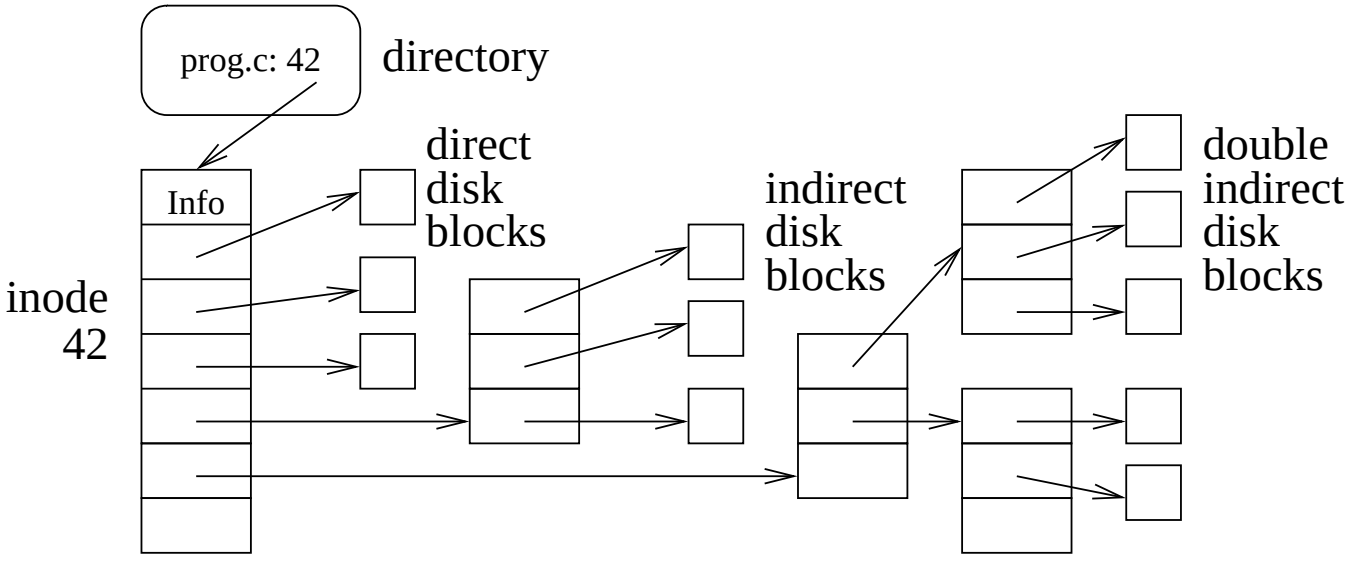
Inodes



This gives us 256 more blocks, which is 256KB more space

Filesystems

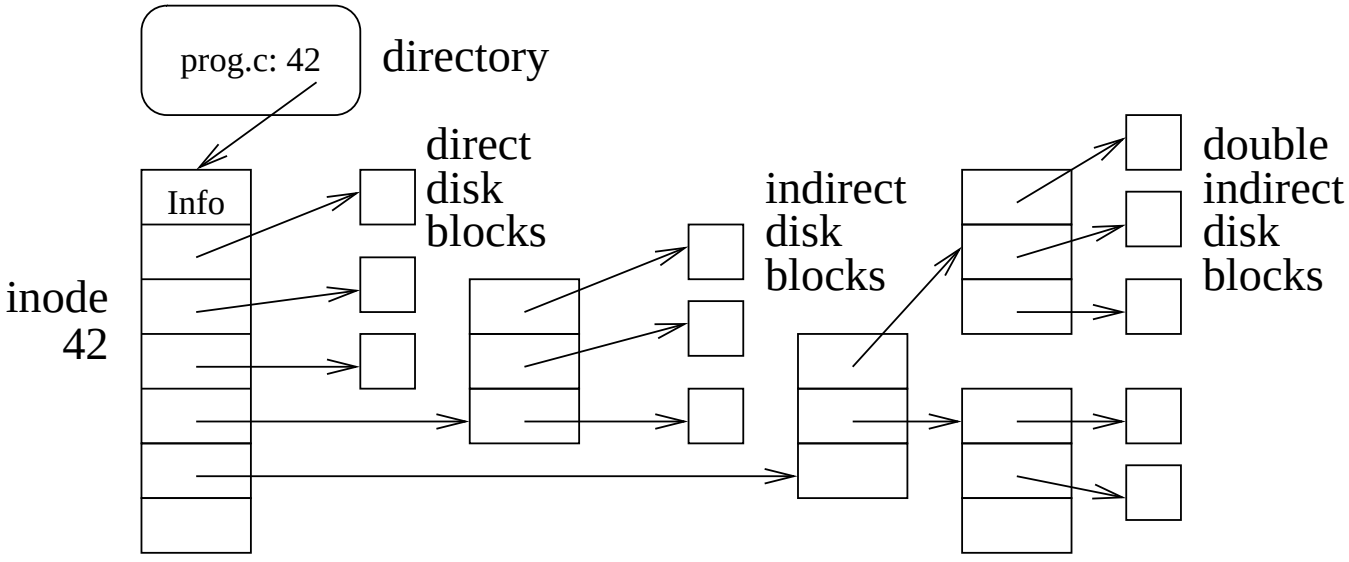
Inodes



Bigger files have a *double indirect block*

Filesystems

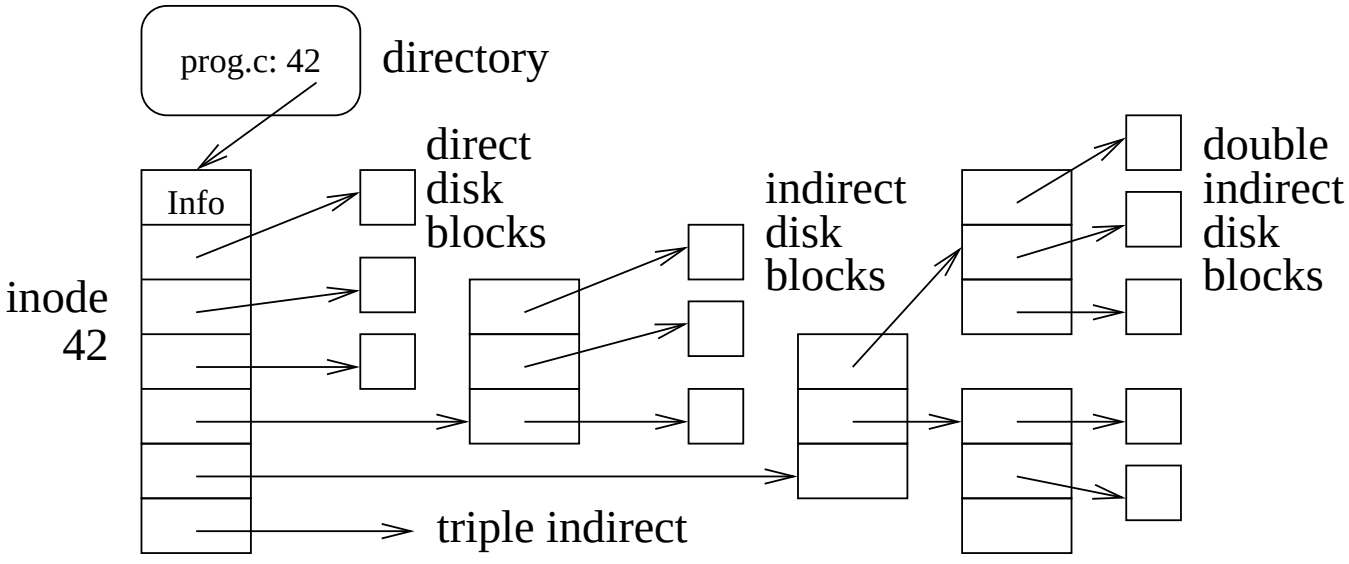
Inodes



This gives us $256 \times 256 = 65536$ more blocks, 65MB more space

Filesystems

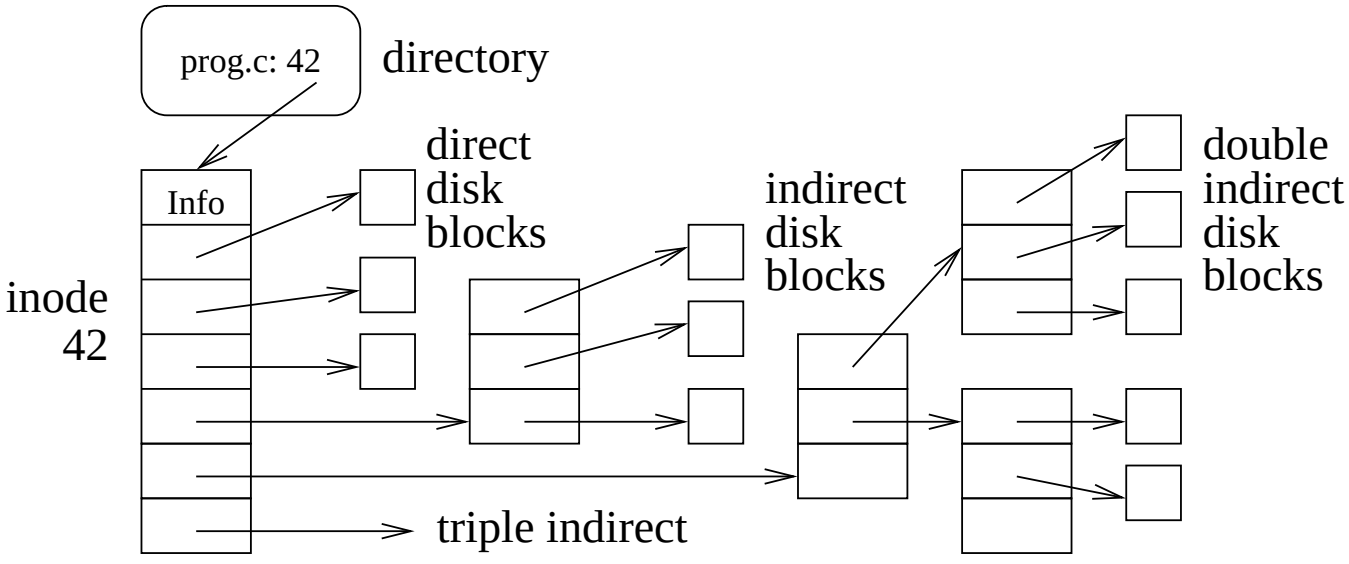
Inodes



Extreme files need a *triple indirect block*

Filesystems

Inodes



This takes us to 16 million blocks, 16GB more space

Filesystems

Inodes

Now every indirect block is overhead occupying space on the disk that could otherwise be storing data

Filesystems

Inodes

Now every indirect block is overhead occupying space on the disk that could otherwise be storing data

But this is not so wasteful as you might think as most files are quite small; the overhead for large files is relatively small, too

Filesystems

Inodes

Caching the inode and the indirect blocks in memory helps reduce the lookup overhead

Filesystems

Inodes

Caching the inode and the indirect blocks in memory helps reduce the lookup overhead

The space for the pointers is used for various other things when the inode refers to something other than a disk file

Filesystems

Inodes

For example, a *soft link* (similar to a Windows *shortcut*) to a file or directory

Filesystems

Inodes

For example, a *soft link* (similar to a Windows *shortcut*) to a file or directory

This is a special inode whose purpose is to say “don’t look at me, look at this file instead”

Filesystems

Inodes

For example, a *soft link* (similar to a Windows *shortcut*) to a file or directory

This is a special inode whose purpose is to say “don’t look at me, look at this file instead”

If you had a soft link named `foo` that linked to `bar` its content would be just the name “`bar`”

Filesystems

Inodes

For example, a *soft link* (similar to a Windows *shortcut*) to a file or directory

This is a special inode whose purpose is to say “don’t look at me, look at this file instead”

If you had a soft link named `foo` that linked to `bar` its content would be just the name “`bar`”

But the action of the OS when a program opens `foo` is not to present the data “bar”, but to close inode `foo` and open inode named by `bar` instead

Filesystems

Inodes

For example, a *soft link* (similar to a Windows *shortcut*) to a file or directory

This is a special inode whose purpose is to say “don’t look at me, look at this file instead”

If you had a soft link named `foo` that linked to `bar` its content would be just the name “`bar`”

But the action of the OS when a program opens `foo` is not to present the data “bar”, but to close inode `foo` and open inode named by `bar` instead

In effect, this is another way for files to have multiple names, but it is very different from normal multiple names, called *hard links*

Filesystems

Inodes

A hard link is the normal reference to (the inode of) a file; a soft link is (a reference to an inode containing) a signpost saying “look over there”

Filesystems

Inodes

A hard link is the normal reference to (the inode of) a file; a soft link is (a reference to an inode containing) a signpost saying “look over there”

The soft link might point to a place where there is no file; a hard link *is* the file

Filesystems

Inodes

A hard link is the normal reference to (the inode of) a file; a soft link is (a reference to an inode containing) a signpost saying “look over there”

The soft link might point to a place where there is no file; a hard link *is* the file

And, as there are no inode references involved in a soft link, it can be the name of any file on any filesystem in the machine

Filesystems

Inodes

A hard link is the normal reference to (the inode of) a file; a soft link is (a reference to an inode containing) a signpost saying “look over there”

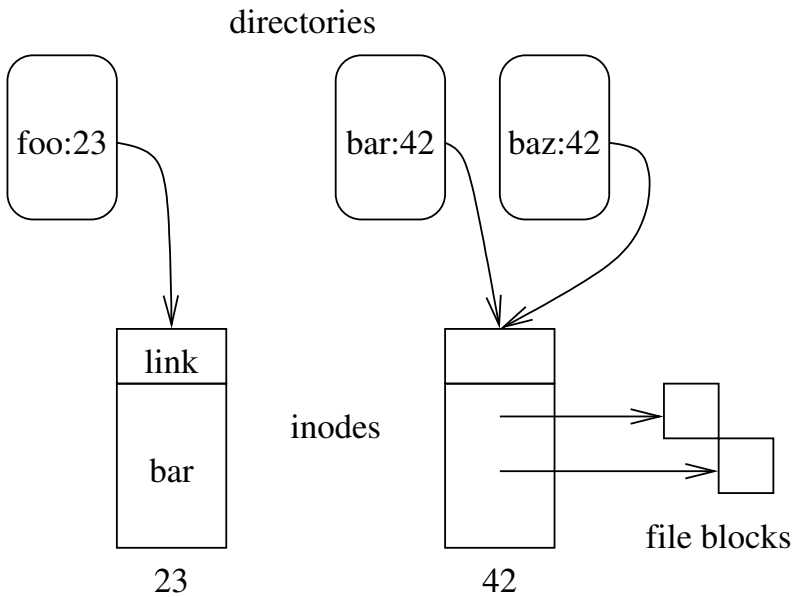
The soft link might point to a place where there is no file; a hard link *is* the file

And, as there are no inode references involved in a soft link, it can be the name of any file on any filesystem in the machine

Note: a hard link refers to the file, while a soft link refers to a *name* of the file. So a hard link is a name, while a soft link is a name of a name

Filesystems

Inodes



Filesystems

Inodes

Use `ls -li` to see the link details and inode number of a file under Unix

```
% ln -s somefile link1
% ls -li link1
3154340 lrwxrwxrwx 2 rjb users 6 2010-04-22 10:38 lin
k1 -> somefile
% ln link1 link2
% ls -li link*
3154340 lrwxrwxrwx 2 rjb users 6 2010-04-22 10:38 lin
k1 -> somefile
3154340 lrwxrwxrwx 2 rjb users 6 2010-04-22 10:38 lin
k2 -> somefile
```