Operating Systems
Lecture 31: Filesystems 2

Nipun Batra
Nov 15, 2018

Deleting Files

There is no system call for deleting files!

Inode (and associated file) is garbage collected when there are no references (from paths or fds).

Paths are deleted when: `unlink()` is called.

Let's Learn About Link Before Unlink
Hard Link

```
prompt> echo hello > file
prompt> cat file
hello
prompt> ln file file2  // create a hard link, link file to file2
prompt> cat file2
hello
```

Let's Learn About Link Before Unlink
Hard Link

```
prompt> echo hello > file
prompt> cat file
hello
prompt> ln file file2  // create a hard link, link file to file2
prompt> cat file2
hello
```

link(old pathname, new one)

Let's Learn About Link Before Unlink
Hard Link

```
prompt> echo hello > file
prompt> cat file
hello
prompt> ln file file2  // create a hard link, link file to file2
prompt> cat file2
hello
```

link(old pathname, new one)

- Link a new file name to an old one

Let's Learn About Link Before Unlink
Hard Link

```
prompt> echo hello > file
prompt> cat file
hello
prompt> ln file file2  // create a hard link, link file to file2
prompt> cat file2
hello
```

link(old pathname, new one)

- Link a new file name to an old one
- Create another way to refer to the same file

Let's Learn About Link Before Unlink
Hard Link

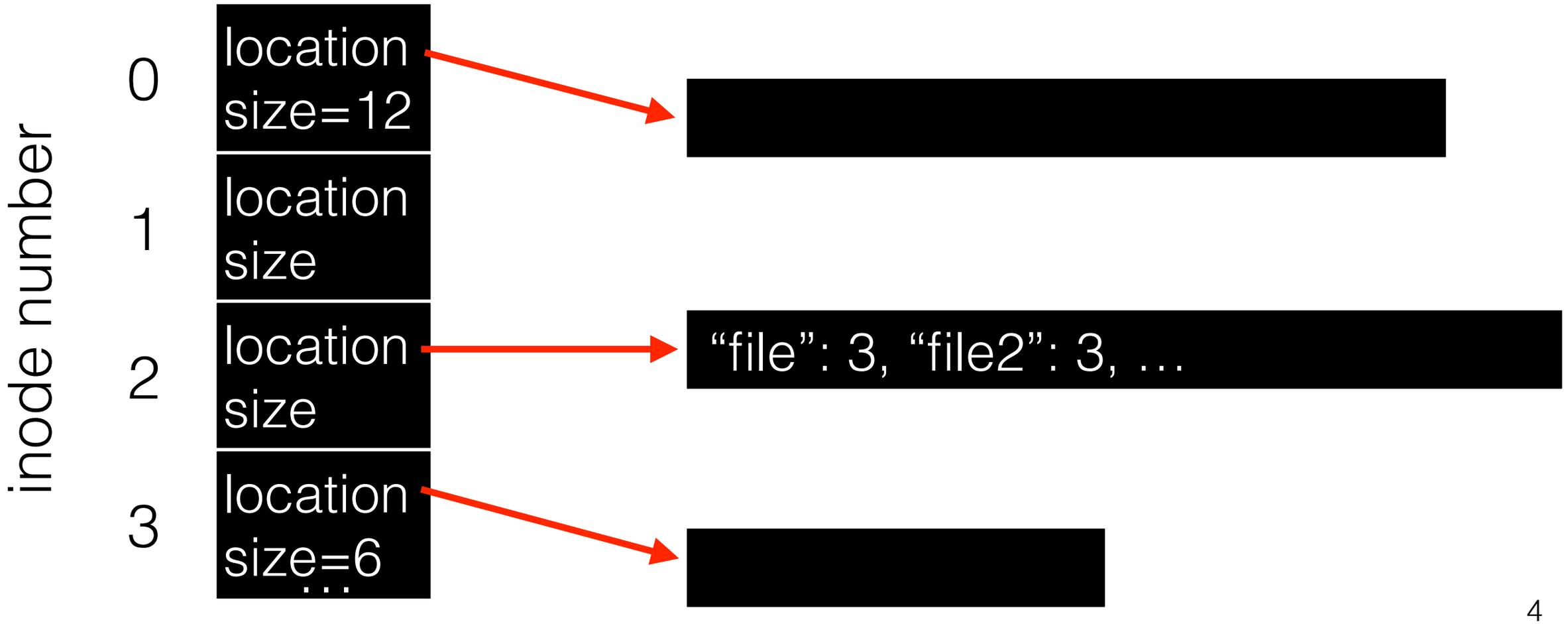
```
prompt> echo hello > file
prompt> cat file
hello
prompt> ln file file2  // create a hard link, link file to file2
prompt> cat file2
hello
```

link(old pathname, new one)

- Link a new file name to an old one
- Create another way to refer to the same file
- The command-line link program : ln

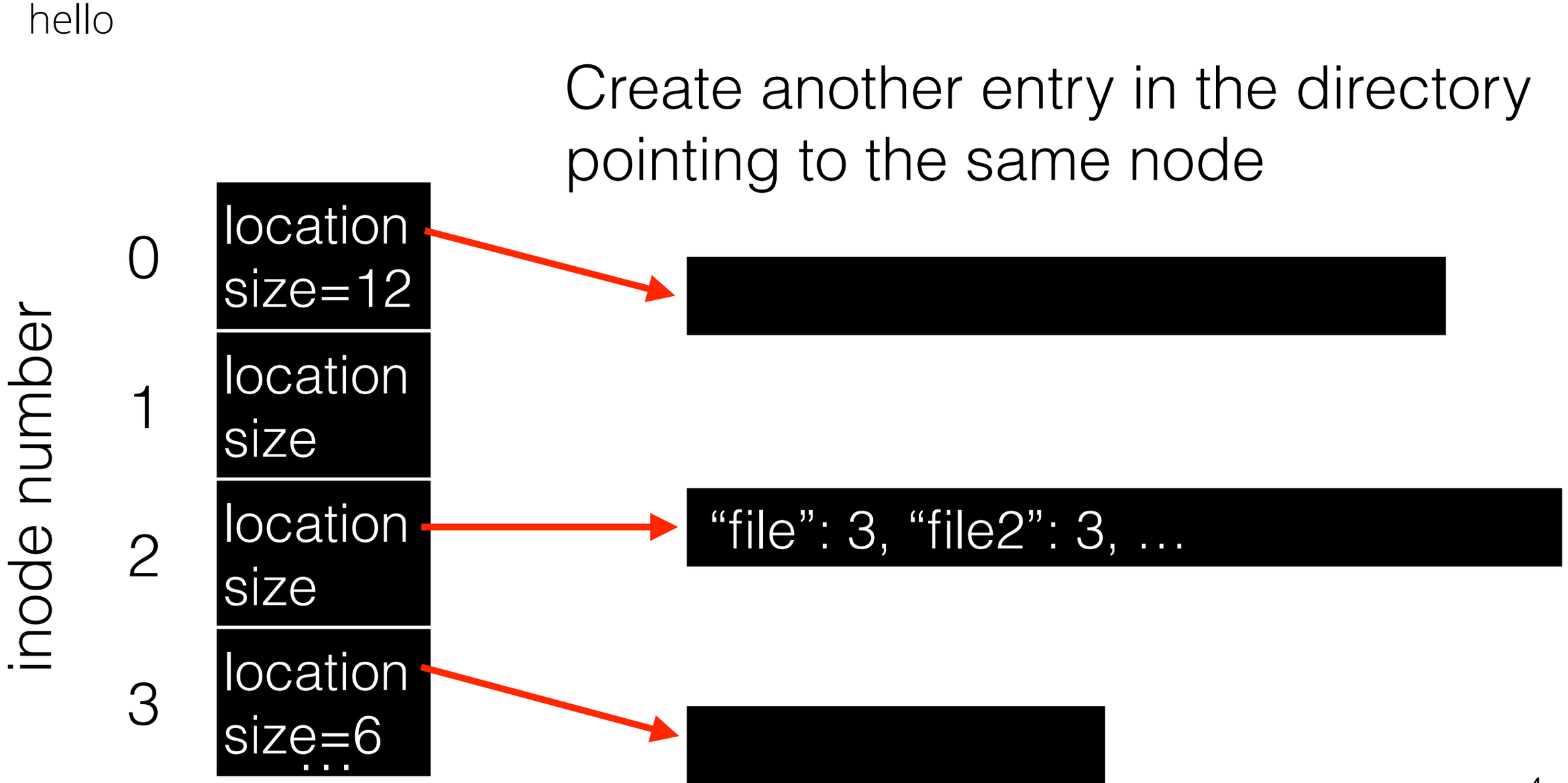
Let's Learn About Link Before Unlink
Hard Link

```
prompt> echo hello > file
prompt> cat file
hello
prompt> ln file file2  // create a hard link, link file to file2
prompt> cat file2
hello
```

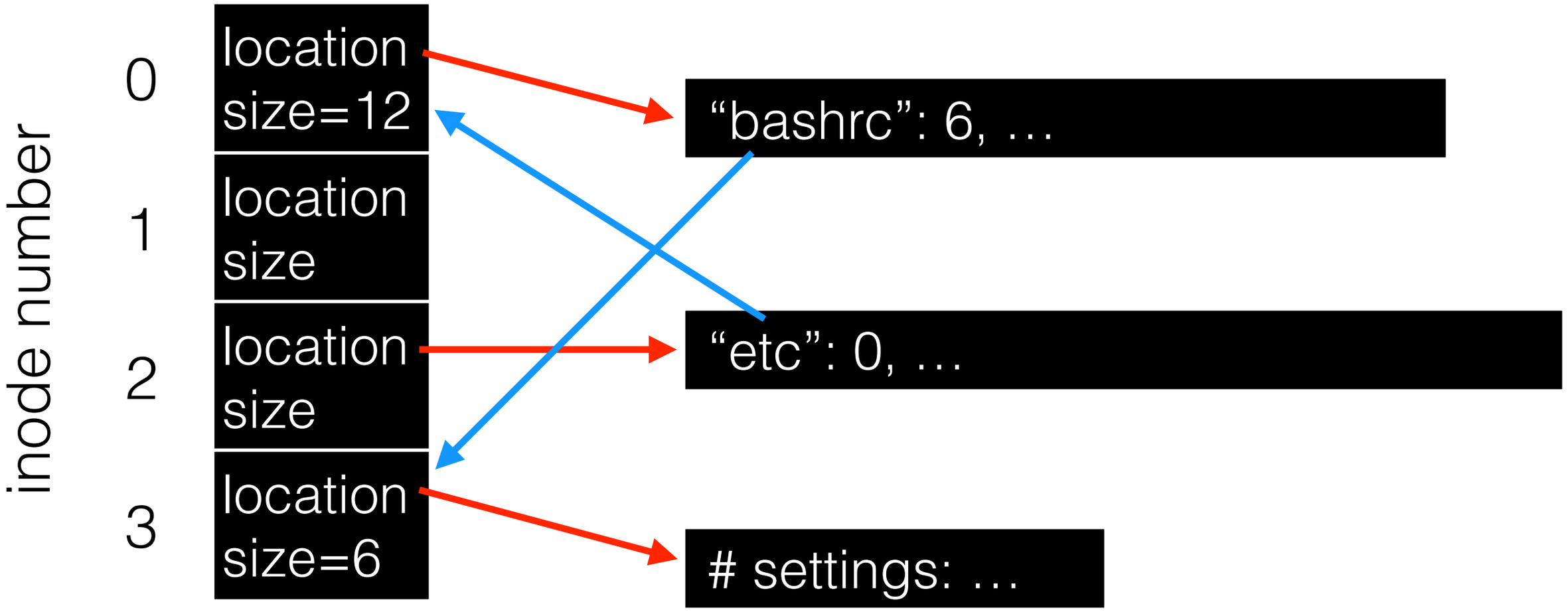


Let's Learn About Link Before Unlink
Hard Link

```
prompt> echo hello > file
prompt> cat file
hello
prompt> ln file file2  // create a hard link, link file to file2
prompt> cat file2
hello
```

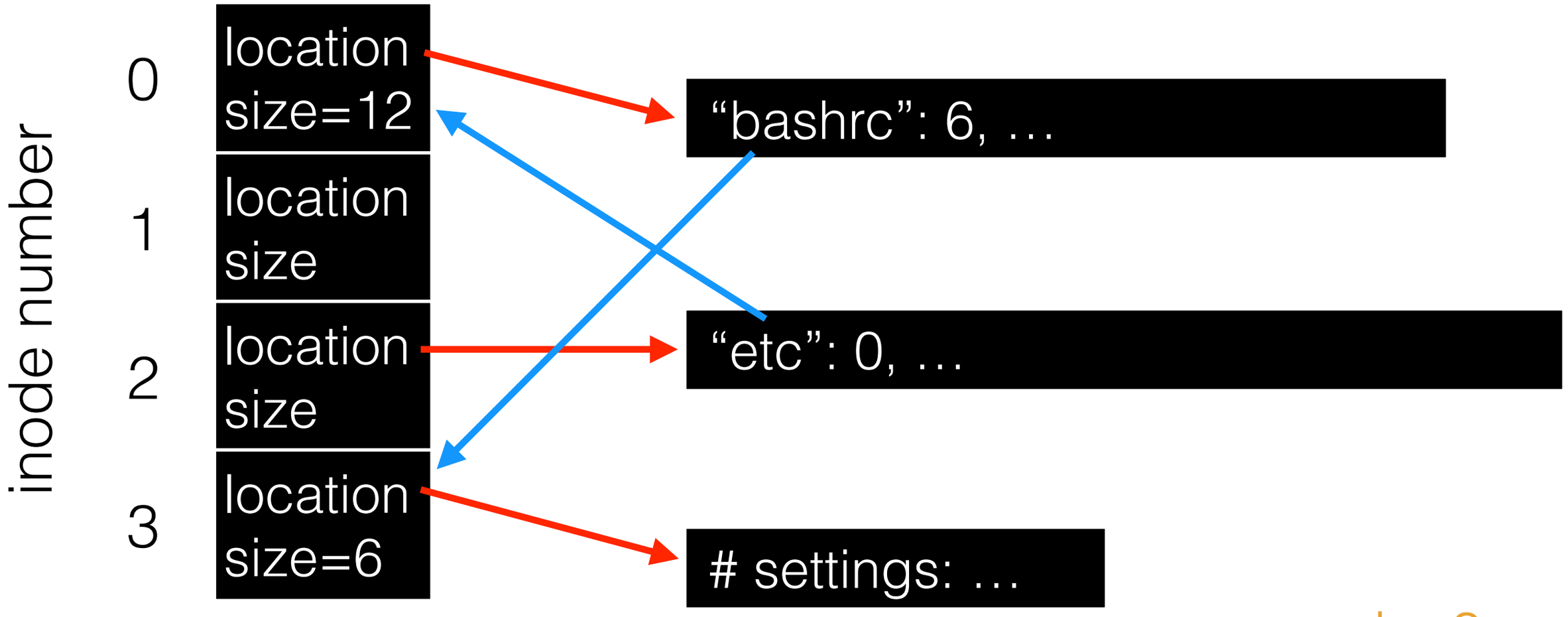


Directories and Files



Directories and Files

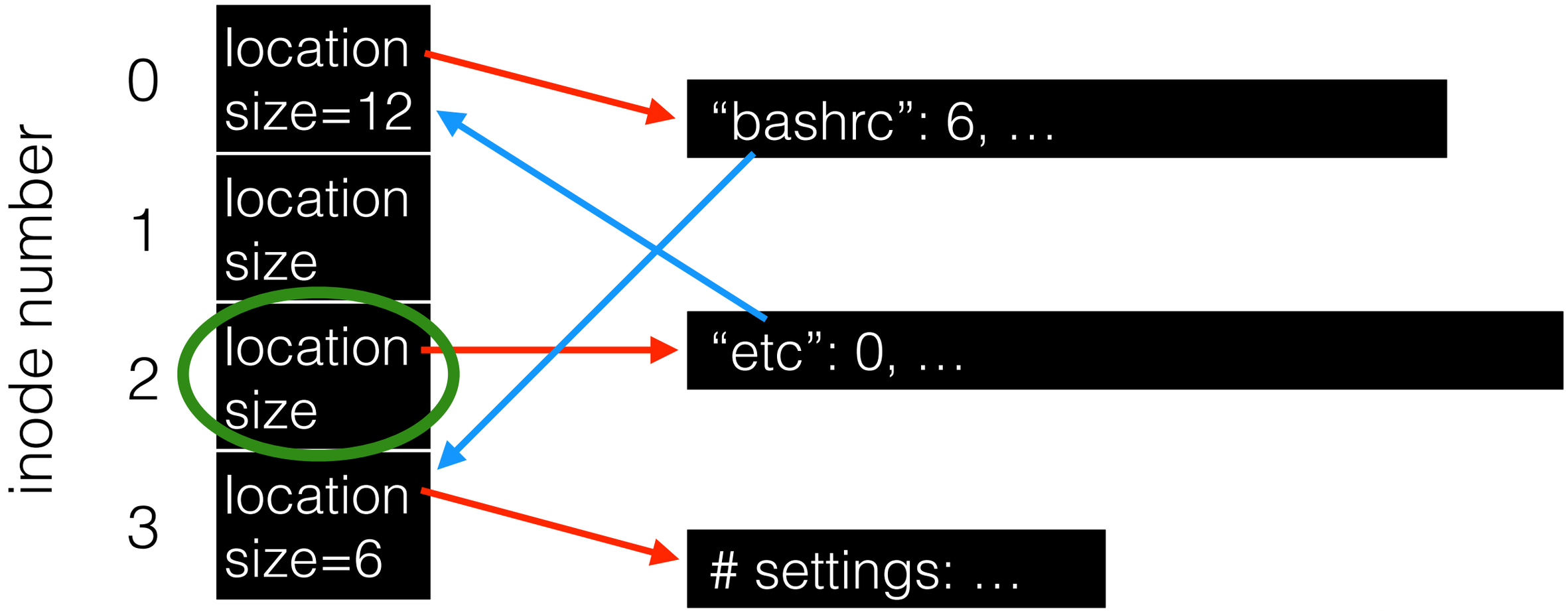
read **/etc/bashrc**



reads: 0

Directories and Files

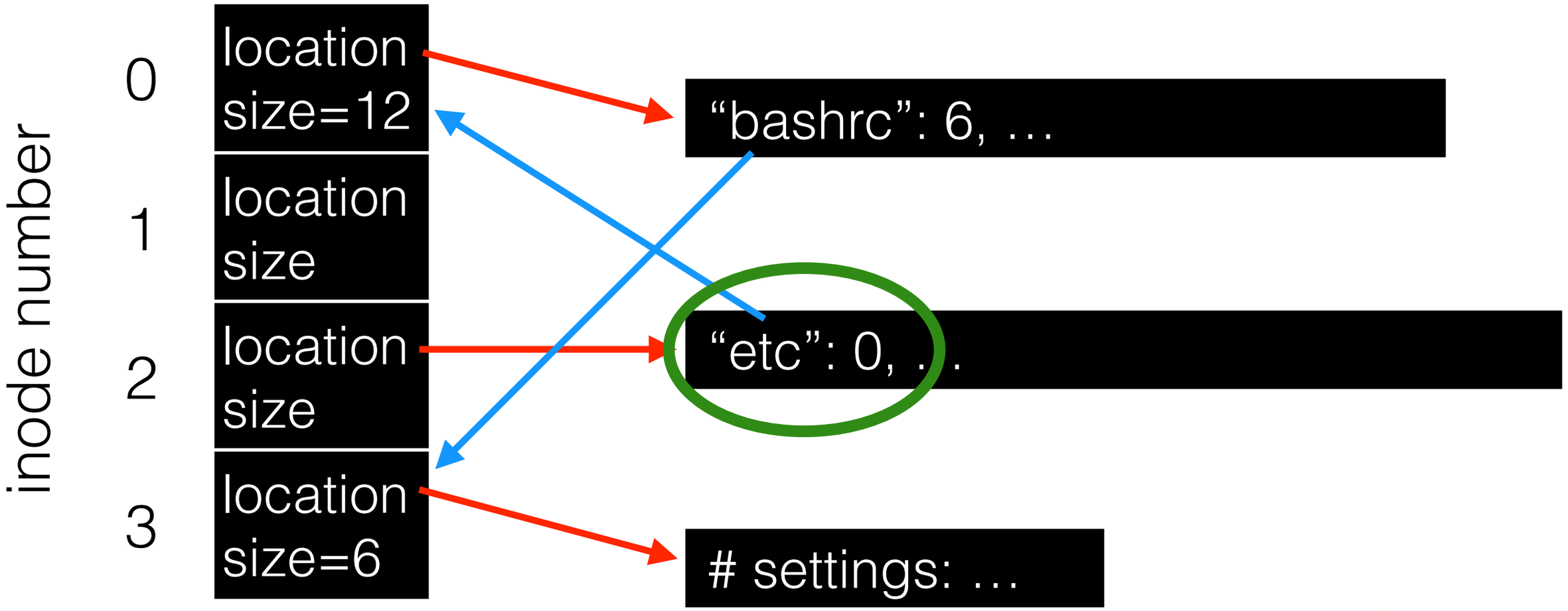
read **/etc/bashrc**



reads: 1

Directories and Files

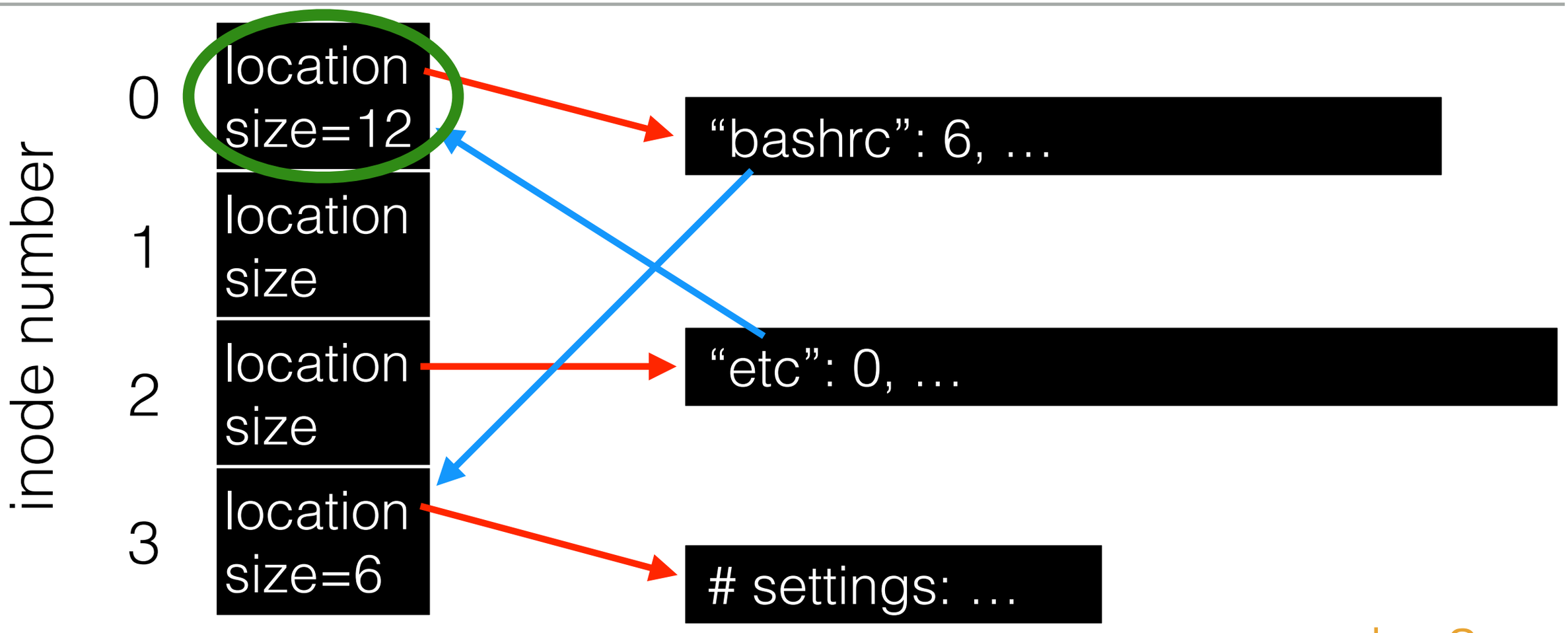
read **/etc/bashrc**



reads: 2

Directories and Files

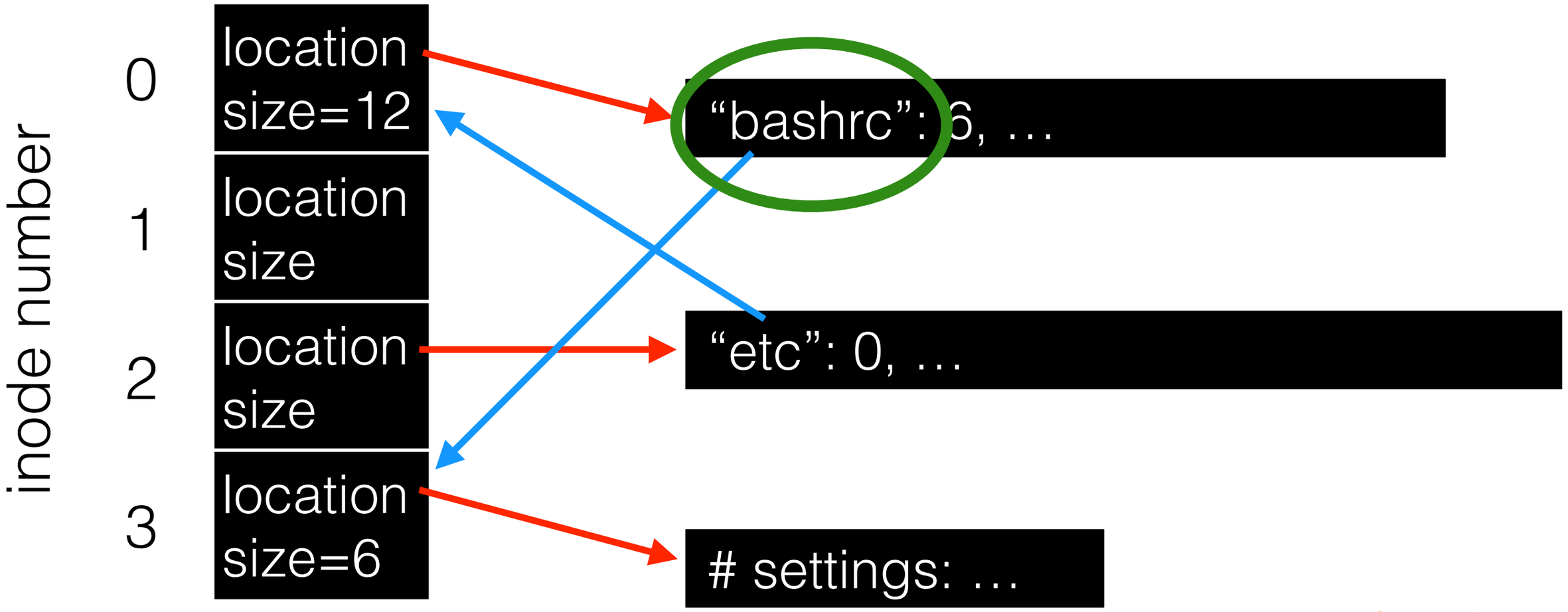
read **/etc/bashrc**



reads: 3

Directories and Files

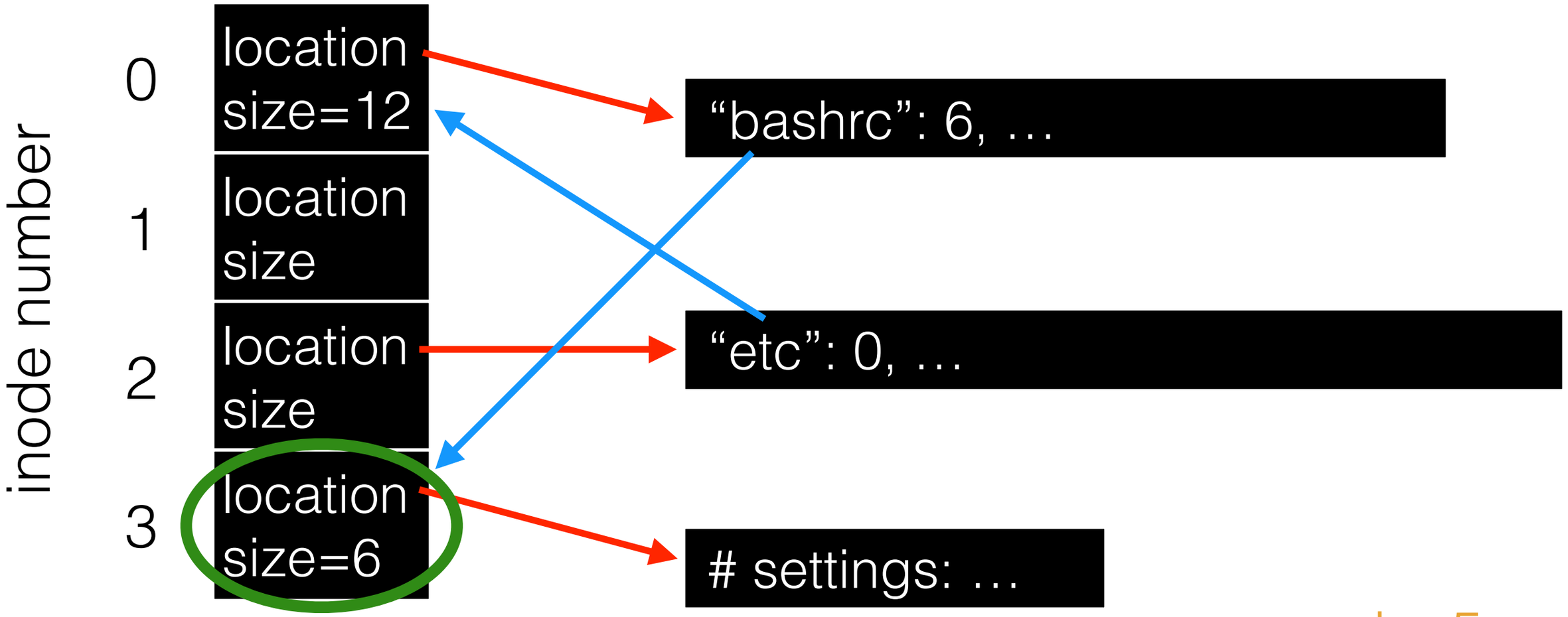
read **/etc/bashrc**



reads: 4

Directories and Files

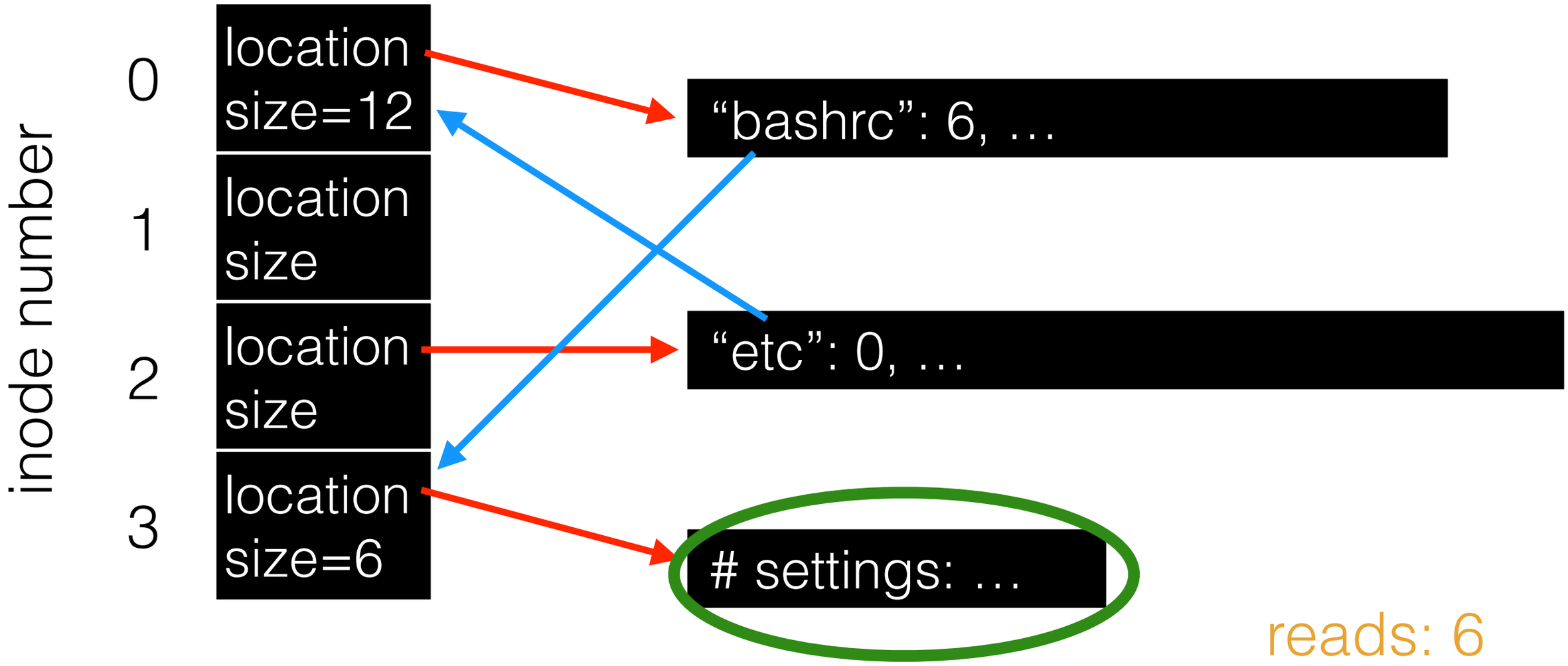
read **/etc/bashrc**



reads: 5

Directories and Files

read **/etc/bashrc**



reads: 6

Creating Directories

prompt> strace mkdir foo

…

mkdir(“foo”, 0777) = 0

prompt>

```
▶ ls -la
total 9033248
drwxr-xr-x+  90 nipun  staff        2880 Nov 15 05:28 .
drwxr-xr-x    6 root   admin         192 Sep 25 06:35 ..
-r--------    1 nipun  staff           7 Nov 30  2017 .CFUserTextEncoding
-rw-r--r--@   1 nipun  staff       34820 Nov 14 16:56 .DS_Store
drwx------    3 nipun  staff          96 Oct 29 19:17 .Trash
drwxr-xr-x    3 nipun  staff          96 Nov 30  2017 .astropy
drwxr-xr-x   15 nipun  staff         480 Sep 17 10:07 .atom
-rw-------    1 nipun  staff         191 Nov 30  2017 .bash_history
-rw-r--r--    1 nipun  staff         172 Jun 25 17:01 .bash_profile
-rw-r--r--    1 nipun  staff          87 Nov 30  2017 .bash_profile-anaconda3.bak
drwx------    5 nipun  staff         160 Nov 30  2017 .bash_sessions
drwxr-xr-x    3 nipun  staff          96 Jun 26 09:46 .cache
drwxr-xr-x    4 nipun  staff         128 Nov 30  2017 .conda
-rw-r--r--    1 nipun  staff          39 Jun 25 17:21 .condarc
drwxr-xr-x    3 nipun  staff          96 Jan 27  2018 .config
drwx------    3 nipun  staff          96 Dec  1  2017 .cups
drwx------   14 nipun  staff         448 Mar 30  2018 .dropbox
-rw-r--r--    1 nipun  staff         322 Feb 10  2018 .floydconfig
drwxr-xr-x    5 nipun  staff         160 Dec  9  2017 .fxhome-helper
```

Directory Entry

```
struct dirent {
    char  d_name[256];   /* filename */
    ino_t      d_ino;  /* inode number */
    off_t      d_off;  /* offset to the next direct */
    unsigned short  d_reclen;  /* length of this record */
    unsigned char  d_type;  /* type of file */
}
```

Directory Entry

```
struct dirent {
    char  d_name[256];   /* filename */
    ino_t      d_ino;  /* inode number */
    off_t      d_off;  /* offset to the next direct */
    unsigned short  d_reclen;  /* length of this record */
    unsigned char  d_type;  /* type of file */
}
```

`unsigned char d_type`

This is the type of the file, possibly unknown. The following constants are defined for its value:

`DT_UNKNOWN`

The type is unknown. Only some filesystems have full support to return the type of the file, others might always return this value.

`DT_REG`

A regular file.

`DT_DIR`

A directory.

`DT_FIFO`

A named pipe, or FIFO. See FIFO Special Files.

How would **ls** work on a Directory?

How would **ls** work on a Directory?

```
int main(int argc, char *argv[]) {
    DIR *dp = opendir(".");   // open current directory
    assert(dp != NULL);
    struct dirent *d;
    while ((d = readdir(dp)) != NULL)  // read one directory entry
    {
        // print outthe name and inode number of each file
        printf("%d %s\n", (int) d->d_ino, d->d_name);
    }
    closedir(dp);   // close current directory
    return 0;
}
```

Removing a Directory

Removing a Directory

- rmdir(): Delete a directory.
  - Require that the directory be empty.
  - If you call rmdir() to a non-empty directory, it will fail.
  - I.e., Only has “.” and “..” entries.

Removing a Directory

- rmdir(): Delete a directory.
  - Require that the directory be empty.
  - If you call rmdir() to a non-empty directory, it will fail.
  - I.e., Only has “.” and “..” entries.

```
nipun@nipun-VirtualBox:~/demo$ mkdir test
nipun@nipun-VirtualBox:~/demo$ touch test/1.txt
nipun@nipun-VirtualBox:~/demo$ touch test/2.txt
nipun@nipun-VirtualBox:~/demo$ strace rm -rf test
```

Removing a Directory

- rmdir(): Delete a directory.
  - Require that the directory be empty.
  - If you call rmdir() to a non-empty directory, it will fail.
  - I.e., Only has “.” and “..” entries.

```
nipun@nipun-VirtualBox:~/demo$ mkdir test
nipun@nipun-VirtualBox:~/demo$ touch test/1.txt
nipun@nipun-VirtualBox:~/demo$ touch test/2.txt
nipun@nipun-VirtualBox:~/demo$ strace rm -rf test
```

```
unlinkat(4, "2.txt", 0)                 = 0
unlinkat(4, "1.txt", 0)                 = 0
close(4)                                = 0
unlinkat(AT_FDCWD, "test", AT_REMOVEDIR) = 0
```

Permission Bits

Permission Bits

- demo_permission.sh

Many File Systems

Users often want to use many file systems.

For example:
- main disk
- backup disk
- NFS
- thumb drives

What is the most elegant way to support this?

Many File Systems

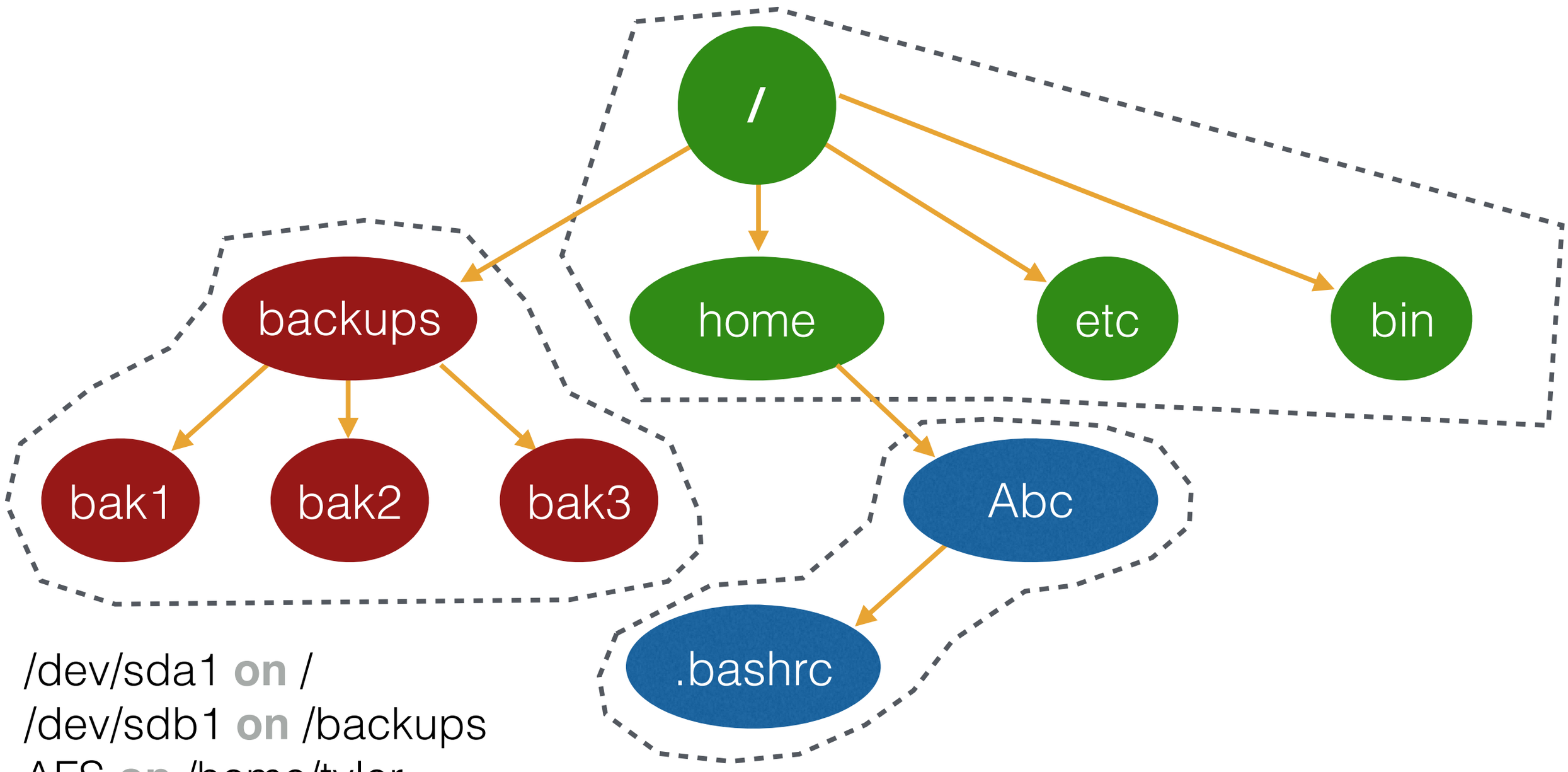
Idea: stitch all the file systems together into a super file system!

Many File Systems

Idea: stitch all the file systems together into a super file system!

sh> **mount**
/dev/sda1 on / type ext4 (rw)
/dev/sdb1 on /backups type ext4 (rw)
NFS on /home/abc type nfs

Many File Systems



/dev/sda1 **on** /
/dev/sdb1 **on** /backups
AFS **on** /home/tyler
harter@galap-1:… **on** /home/tyler/537

Mounting Video

https://www.youtube.com/watch?v=A8lTr5ZpzvA

Mounting Video

https://www.youtube.com/watch?v=A8lTr5ZpzvA