Filesystems

ECE598: Advanced Operating Systems – Homework 8

Spring 2018

Due: Thursday, 5 April 2018, 2:00pm

This homework involves filesystems.

You may work on this homework with a group.

1. **Download the homework code template**

   - Download the code from:
     `http://web.eece.maine.edu/~vweaver/classes/ece598/ece598_hw8_code.tar.gz`
   - Uncompress the code. On Linux or Mac you can just
     `tar -xzvf ece598_hw8_code.tar.gz`

2. **Investigate romfs file system support**

   - The filesystem we are using is called ROMFS. You can read the documentation on it here: `https://www.kernel.org/doc/Documentation/filesystems/romfs.txt`
   - We have no disk support, so the romfs is loaded as a ramdisk at boot time (it is included as a binary blob during the kernel build). All block reads/writes are turned into RAM accesses with the proper offset. The code for this is simple, and can be found in:
     `kernel/drivers/block/ramdisk.c`
   - The romfs filesystem support is in `kernel/fs/romfs/romfs.c`
     The important functions are:
     - `romfs_get_inode()` – given a filename, finds the inode for it
     - `romfs_stat()` – returns metadata from a file
     - `romfs_mount()` – mounts the filesystem
     - `romfs_statfs()` – the statfs system call, reports disk usage for tools like "df"
     - `romfs_getdents()` – returns the directory entry info

3. **Modify values returned by the filesystem (5pts)**

   (a) When you run `ls -l` (lowercase L) it lists the files including some of the metadata. You'll notice that romfs does not store file times, so they default to the beginning of the UNIX epoch.

   Modify the `romfs_stat()` call so that it shows a different modify time when doing `ls -l`. By default the buf.ctime, buf.mtime, and buf.atime are all zero. Set them to something more interesting.

   (b) You can run `df` to see how much disk space is available. For our system it's a really small amount.

   Modify the `romfs_statfs()` routine so that it lies and reports the ramdisk size as being 1GB and 50% full.

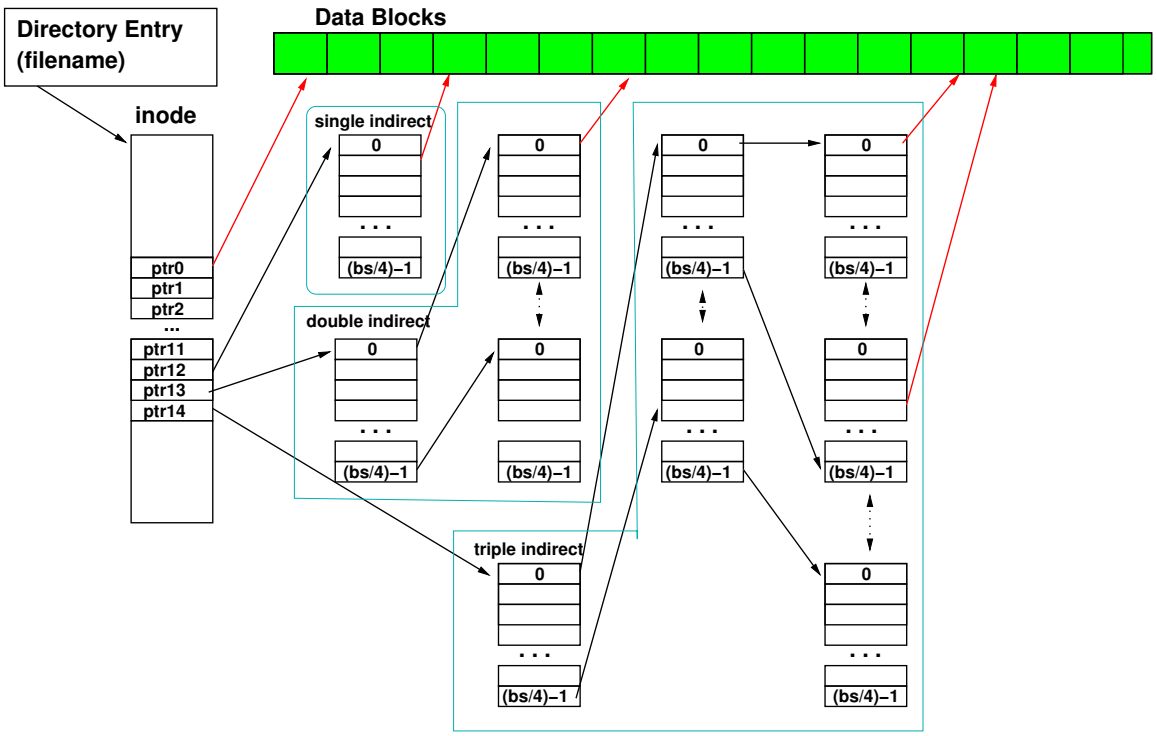

Figure 1: ext2 inode Diagram

4. **Questions (5pts)**

In addition to the data requested above, answer these questions in the README file.

(a) On a Linux ext2 filesystem, is the filename stored in the inode? Explain why or why not.

(b) The ext2 filesystem inode contains 15 block pointers. The first 12 (0-11) point directly to disk blocks. Entry 12 points to an indirect block which contains BLOCKSIZE/4 (divided by 4 because the pointer size is 32-bits) block pointers. Entry 13 points to a double indirect block where each pointer points to an indirect block. And finally, entry 14 points to a triple indirect block. (See the diagram in Figure 1 which may or may not help).

i. Assuming a blocksize of 1kB, what is the maximum file size supported without using indirect blocks?

ii. What is the maximum file size if additionally all the blocks from the first indirect block are used?

iii. What is the maximum file size if additionally the second indirect blocks are used?

iv. What is the maximum size of a file if all possible blocks are used (including the triple indirect)?

v. For the previous answer involving the maximum size, what is the overhead involved (i.e. how much space is spent holding all of the indirect blocks)?

(c) The ext2 and fat filesystems are very different.

i. List one use case where an ext2 filesystem works better than fat.

ii. List one use case where a fat filesystem works better than ext2.

(d) The file `new_file` is shown via the `ls` command to be 41 Megabytes in size

```
rasp-pi:~% ls -lh new_file
-rw-r--r-- 1 vince weaver 41M Mar 30 21:34 new_file
```

But the command `du -h` which shows how many disk blocks a file uses only shows 16k being used.

```
rasp-pi:~% du -h new_file
16K new_file
```

How is this possible? Why might this be a useful feature to have?

(e) Pick a filesystem supported by Linux (that isn't ext2/3/4, btrfs, or fat) and do some brief research on it. Write a few sentences describing where the filesystem originated, its strengths and weaknesses, and why there's a Linux driver for it. You can find a list of Linux filesystems under the `fs` subdirectory of a Linux source tree, which you can find online here: `http://lxr.free-electrons.com/source/fs/`.

5. **Submit your work**

- Run `make submit` in your code directory and it should make a file called `hw8_submit.tar.gz`. E-mail that file to me as well as the document with the answers to the questions.