Operating Systems
Semester 1, 2015
Eike Ritter

The University of Birmingham
24 November, 2015

Assessed Exercise 4

Deadline: Friday 11 December, 4pm

The Task

Write a kernel module called `minix.ko` which provides a way to change the uid for all files in a given minix file system to a specific value.

More precisely, the kernel module should operate as follows:

- The kernel module provides a proc-file `/proc/minix` which accepts input of the form `<uid><ino><path>`, where `uid` is string denoting an integer between 0 and 65535 inclusive, `ino` is a string denote a non-negative integer and path is a string denoting a fully qualified pathname pointing to a file in a minix file system. `uid`, `ino` and `path` are separated by exactly one space each. Writing invalid input to the proc-file returns `-EFAULT`. If `ino` is not zero you may assume that `ino` is a valid inode number, and the kernel changes the uid of this inode to `uid`. If `ino` is 0, the kernel changes the uid of **all** files in the minix-file system to `uid`.

- The file system should not be corrupted after your changes. To test this, execute the following steps in order: unmount the file system, then do a file system check via `fsck.minix -f` and finally remove the module when no minix-file system is mounted.

Hints

The solution to this exercise is not that long but you will have to understand the code implementing the minix-file system in order to write your own code.

- Please use the code provided on the module website and fill the spaces marked with TODO. Please submit a zip-file with the whole minix file-system implementation including the Makefile.

- The minix file system keeps a map of all valid inodes. This map is an array of little-endian 16-bit short integers. To see how this is used, look at the functions `minix_count_free_inodes` and `count_free` in the file `bitmap.c`. In a 16-bit short integer the least significant bit corresponds to the lowest inode number.

- You will need to consult the kernel sources, in particular the definitions for super blocks, dentries, VFS-inodes and raw inodes (inodes of the minix file system).

- It is probably best to start this exercise by implementing the case when `ino` is not 0.

- The `-i` option of the `ls` command gives you the inode number for any given file.
- You will need a minix file system for testing. The best way to create such a file system is to use the command

  ```
  dd if=/dev/zero of=<filename> bs=1024 count=<n>
  ```

  where n is the size of the file system in KB, to create a file of the given size, then to use

  ```
  mkfs.minix <filename>
  ```

  to create a file system on this file and then use

  ```
  mount -o loop <filename> <mountpoint>
  ```

  to mount this file system.
- You need to be careful about freeing inodes, dentries and paths.
- You are allowed to use or call any part of the kernel except for the `stat` or `chown` system calls or their implementation in the kernel for this exercise. You must use the inode map of the minix file system to find out which inodes are valid and implement chown by changing the inode directly.