UNIVERSITY of FLORIDA
Herbert Wertheim
College of Engineering

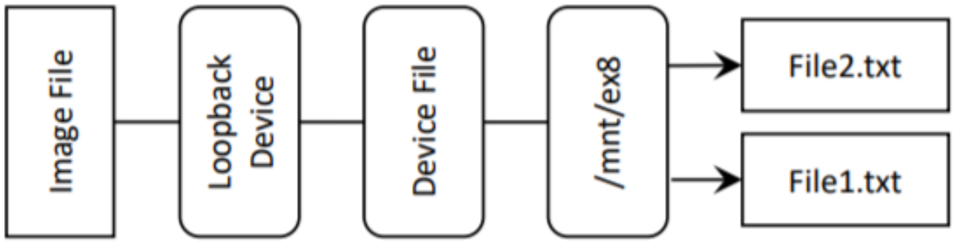
Lab 6: File System Basics & IO Device Files

COP4600 – Operating Systems

Overview

This lab should begin to help you understand the Linux File System and the structs behind it. Linux originally used Minix's file system, but it has gone through four iterations of extended file systems over the course of the kernel's development. The current file system on your kernel is the ext4 file system (fourth extended file system), but you will only be modifying one of the base system calls behind it. The inode is the underlying data structure of the file system in Linux. The inode contains data relevant to when the file system was modified or changed, so that these changes can automatically be recorded. It also contains access information for users and groups along with information about the inodes before and after it connected files.

This lab will also give you an idea of how virtual storage devices work, as you will be creating one. In Unix-based systems (like Linux) and many other operating systems, devices in the system are represented using device files – that is, virtual files present in the virtual filesystem presented by the OS that handle reads from and writes to connected devices. These devices are often physical ones – such as solid-state drives, mice, or keyboards – but they can also be virtual devices, which simulate hardware via a software mechanism. You will create and attach a virtual storage drive via a loopback device. Loopback devices are commonly used on Unix systems for creating, reading, and writing storage images (such disc images, i.e. "ISOs"). You'll connect the storage, mount it, write to it, unmount it, and then verify that the data has been written.



Structure

The lab is broken into two main parts: 1. Understanding file system basics 2. Working with IO Device Files

File Systems

1. Find the system call read in the file system
2. Change read to include printing the inode number of the file being opened (take a screenshot of the changed system call). *Note that due to the nature of this change, the kernel might not be able to finish booting when you restart the VM! This is allowed – you just need to get a screenshot and make a patch file for submitting*

3. Remake the kernel and run in debug mode to view the changes (take a screenshot of the bootup process showing inode numbers being printed)
4. Make a diff of the file containing the edited system call
5. Revert to your clean (post-Lab 1) snapshot

IO Device - Creating the Image

This part of the lab is broken into the following parts, which you should follow:

1. Create a `ex6.img` file, filled with zeroes, of size one megabyte (1MiB) using `dd`:

   ```
   $ dd if=<input file> of=<output file> bs=<block size> count=<how many>
   ```

   The `dd` command's input can come from any source, a regular file or a special (device) file. A special file, `/dev/zero`, will have an infinite number of zeroes available for reading. The output data's size is defined by the block-size and block-count – you could create a 1KiB file by using a size of 512 and a count of 2, for example.

2. Create a loopback device connected to the image file via `losetup`, format the image in **ext4 format** using `mkfs`, and then mount the loopback device on `/mnt/ex6` via `mount`. Take a screenshot of the commands you used in this step.

3. In this step you will be writing data to the virtual device by creating files on it after mounting.
   In the mounted directory (`/mnt/ex6`), create two text files: one empty, and one containing a message. Display the contents of each file using the `cat` command and take a screenshot.

4. Unmount the filesystem (`umount`) and disconnect the loopback device (`losetup`).

5. Lastly, you need to verify if the data has been written to the image file.
   Install the `hexedit` program (`sudo apt-get install hexedit`) and use it to examine the image file – you should be able to find your filenames and text. Take a screenshot.

Just for fun: try the `strings` command and see what happens!

Submissions

You will submit the following at the end of this lab:

- (File Systems) A screenshot of the changed system call
- (File Systems) A screenshot of the bootup process showing inode numbers being printed
- (File Systems) A diff of the file containing the edited system call
- (IO Device) A screenshot of loopback setup, formatting, and mounting of the image file
- (IO Device) A screenshot of text files created on mounted image file
- (IO Device) A screenshot of image in the hexedit program

Helpful Links

File Systems:

http://www.linfo.org/filesystem.html
http://www.linfo.org/inode.html
http://www.linfo.org/ext4fs.html