UF UNIVERSITY of FLORIDA
Herbert Wertheim College of Engineering

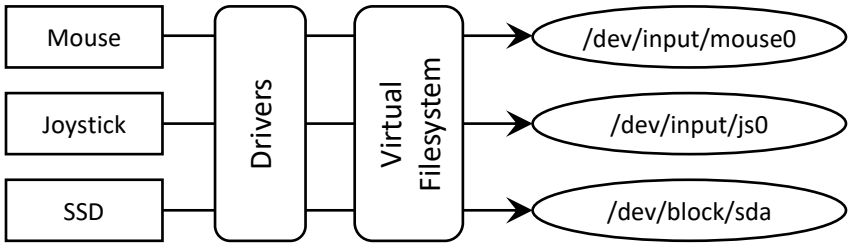
COP4600

Ex6: IO Device Files

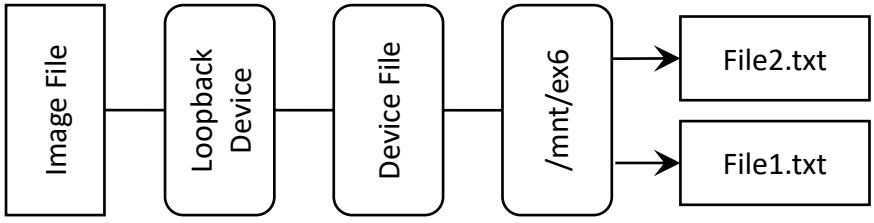
Overview

This exercise should begin to help you understand the Linux File System and how to interact with 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, your kernel's file system is using the ext4 file system (fourth extended file system).

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 do not present data records on a storage medium (as traditional files do). Instead, device files act as handles to read from and write 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.



In this exercise, you will create and attach a virtual storage drive via a ***loopback device***, which allows a regular file to masquerade as a device… which will then to represented using a virtual file (*Unix be cray like that*). Loopback devices are commonly used on Unix systems, especially 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 exercise is broken into these main parts:

1) Create a storage device image file.
2) Connect the file to a loopback device, format it, and mount a filesystem on the device.
3) Write data to the virtual device by creating files on it after mounting
4) Unmount the virtual device and disconnect the loopback device
5) Verify the data has been written to the image file

Creating the Image

Follow these steps to complete the exercise:

1. Create an `ex6.img` file, filled with zeroes, of size one mebibyte (**1 MiB**) 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 using `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.*
   1. You will need to use `sudo` for the `losetup` command.

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:
   1. one empty,
   2. 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 `hexedit` (`sudo apt-get install hexedit`) and use it to examine the image – you should be able to find your **filenames** and **text**. *Take a screenshot of the file text.*

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

Submissions

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

- Screenshot of loopback setup, formatting, and mounting of the image file (`step2.png`)
- Screenshot of the text files' contents (using `cat`) in the mounted directory (`step3.png`)
- Screenshot of image file in the `hexedit` program showing your file text (`step5.png`)