File Deletion

The goal of this lab is to familiarize students with some issues related to file deletion.

Introduction

As an introduction to this lab, you need to understand that there are different designs for how data and metadata could be organized and managed on a disk to create a file system. Linux has been using what is called the *extended file system* (ext) for many years. This design has undergone some changes that have been called ext2, ext3, and now ext4. In DOS, Microsoft started out with what is called the *file allocation table* (FAT), which continued on with its early Windows products (i.e., Windows version 1, 2, 3, 95, 98 and ME). When Microsoft designed Windows NT[1] they did a total redesign of the file system and called it the *NT File System* (NTFS), which has also undergone a few redesigns since it was first introduced around 1995.

In an ideal world the instructor would hand each student a small external USB hard disk for this exercise, but at least two things get in the way of doing that: 1) money, and 2) the current DoD restrictions on the use of external USB drives. Another option would be to examine the results of deleting things on your own hard drive, but that is tricky and risky. So instead of attempting any of those options, in this lab you use what is called a *virtual hard disk* or a *disk image*, which is a file that is used to emulate a disk.

The Windows OS uses the idea of "drive letters" to differentiate different places to store data, such as "C:" and "D:" and "H:". Unix does not use such an abstraction. Instead, **all** data is accessed off some path from the root of the file system. When a disk is added, then the OS must be told where to "mount" it in the file system; sometimes this is configured and performed automatically, and sometimes it must be done manually. Today you will manually need to "mount" your virtual disks to a spot in your home directory hierarchy.

As an aside, the phrase "mounting a disk" is a leftover from older computing days when large things called *disk packs* had to be physically mounted in place. Today, the term "mounting a disk" usually means making the contents of a connected disk available to users as a file system.

There is an appendix of basic Unix commands at the end of these instructions.

[1] NT stands for *New Technology*.

Task 1: EXT2 Virtual Disk Mounting

In this task you will mount a virtual disk.

1. Work within the home directory of the "file-deletion" computer, i.e., where the virtual terminal named "file-deletion" starts.

2. Use the 'll' command to display your home directory content. Notice the "mnt" directory, and use 'll mnt/' to view the content of that directory.
   Also notice the "myfs.img" file, which is a virtual EXT2 file system, and "ntfs.img", which is a virtual NTFS file system. We will first work with the EXT2 file system.

3. Disks are mounted on "mount points", which are directories. We will use the "mnt" directory as our mount point. You can mount your virtual disk by entering the following:

   ```
   sudo mount -o loop myfs.img mnt
   ```

   Your file system is now mounted under the `mnt` directory.

4. Use the command 'll mnt/' to display the size (in bytes) of the files under the directory mnt. The file size is displayed before the Month.

   In item #1 of the report, record the size of each file, as displayed in the output.

5. Un-mount the "disk" by doing the following3:

   ```
   sudo umount mnt
   ```

   This un-mounting is *similar* to removing a USB drive.

[3] Note that the `umount` command only has a 'u' in front of "mount".

Task 2: Deleting a File on Unix

1. Try to display the contents of the "disk" `myfs.img` using the `cat` command:

   ```
   cat myfs.img
   ```

2. The output from the `cat` command may have been a lot of gibberish and junk. The terminal is not equipped to display arbitrary data; it only displays ASCII data well. Display the contents of `myfs.img` as raw data in hexadecimal notation, using the command:

   ```
   hexdump -C myfs.img
   ```

   The output of this command is in three columns: 1) the offset (i.e., location) in the disk image where the data is located, 2) the raw data (in hex), and 3) an ASCII representation of the same data (where possible).

offset	data (hex)	data (ASCII)
00000000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	|................|
*		
00000400	80 00 00 00 00 04 00 00 33 00 00 00 d7 03 00 00	|........3.......|

   Notice: the '*' in the offset column replaces repeated data. For example, in the above output, the data at offsets `00000000`, `00000010`, … `000003f0` are all the same: 16 bytes, where each byte has the value `0x00`. Thus, the output is abbreviated with the asterisk in place of repeated data. Further, this data does not have an ASCII representation, so the third column displays the data as dots. Also in the example, at offset 0x408 is hex value 0x33, which is ASCII value '3'.

3. If we are interested only in the ASCII data in the file, we can extract this using the `strings` command. Use the following command to display the ASCII data and the offset of that data in `myfs.img`:

   ```
   strings -td myfs.img
   ```

   In item #2 of the report, record the offsets for the file names and file data, as displayed in the output.

4. Re-mount your file system:

   ```
   sudo mount -o loop myfs.img mnt
   ```

5. Use the below command to delete `file2`:

   ```
   rm mnt/file2
   ```

6. Use the '`ls mnt`' command to verify that the deleted file is no longer present.

7. Once again, un-mount your file system:

   ```
   sudo umount mnt
   ```

8. Again, display all the ASCII text in the "disk" by entering the following:

   ```
   strings -td myfs.img
   ```

   In item #3 of the report, record the offsets for the file names and file data, as displayed in the output.

   Note that item #4 asks a follow-up question.

Task 3: Undeleting a File on Unix

In this task you will attempt to undelete the file you deleted earlier. In Unix this can be a tricky and difficult task, reserved for knowledgeable system administrators. Even then, when attempting to delete a file there should not be anyone on the system creating new files, or the data may be lost permanently anyway.

Later, we will undelete files using tools that know how to interpret the file system layout. These are especially useful when the file systems grow large and when the formats are complex. For this task, however, we will undelete files "manually" by looking at the raw bytes of the drive and without the assistance of any special file recovery tools.

1. Refer to item #3 of the report. Find the offset to the content of file 2 (i.e., where "Second file created" is located). You will need this number in the next step.

2. Use the `dd` command (shown below) to copy the data from the location on the disk to a new file (`rfile2`) that will hold the recovered data. Below, replace SKIPNUMBER with the offset you found above, and replace FILESIZE with the size of `file2` (from item #1 of the report):

   This command yanked the data out of the virtual disk into another file, called rfile2. To fully restore the file, it may need to be put back into the file system, but for now we will leave it where it is.

3. Display the contents of the file you just recovered:

   ```
   cat rfile2
   ```

Task 4: Securely Deleting a File on Unix

For some, it is comforting to know that it may be possible to undelete files. For others, it is frightening to know that something that was deleted may still be there. For the latter group, this task will show one way to securely delete a file on Ubuntu. You will be using a command called `shred`, which may not be installed on all Linux distributions; although, most operating systems give you *some* utility or operation that will allow you to securely delete files.

1. Use the `strings` command to verify the existence of `file3` and its data:

   ```
   strings myfs.img
   ```

2. Re-mount your file system:

   ```
   sudo mount -o loop myfs.img mnt
   ```

3. Use the following commands to view the files, securely delete `file3`, and then to confirm the deletion:

   ```
   ls mnt
   shred -uxz mnt/file3
   ls mnt
   ```

4. Once again, un-mount your file system:

   ```
   sudo umount mnt
   ```

5. Repeat the use of the `strings` command:

   ```
   strings -td myfs.img
   ```

   In item #5 of the report, record the offsets for the file names and file data, as displayed in the output.

Task 5: NTFS Virtual Disk

As mentioned in the Introduction section, NTFS is the file system used by the professional versions of Windows. Because of the way NTFS manages files, it is much easier to undelete them, as long as new files have neither overwritten the metadata nor the "deleted" data on the disk.

1. Mount the virtual disk:

   ```
   sudo mount -o loop ntfs.img mnt
   ```

2. Use the command '`ll mnt/`' to display the size (in bytes) of the files under the directory mnt. The file size is displayed before the Month.

   In item #6 of the report, record the size of the files, as reported by `ll`.

3. Delete `file1` and **securely** delete `file3`, using the commands:

   ```
   rm mnt/file1
   shred -uxz mnt/file3
   ```

4. Un-mount the "disk" by doing the following:

   ```
   sudo umount mnt
   ```

5. Verify that the data still exists on the virtual disk by entering the following command:

   ```
   strings -td ntfs.img | grep file
   ```

   In item #7 of the report, record the offsets for the file names and file data, as displayed in the output.

6. Use the `ntfsundelete` command to find information about deleted files, as shown below:

   ```
   ntfsundelete -p 100 ntfs.img
   ```

   Note that the inode number is the left-most number in the output.

   Record in item #8 of the report the inode numbers associated with file1.

7. Undelete file1 using the following command (replacing INODE with the number of the recoverable file):

   ```
   ntfsundelete --undelete --inodes INODE --output rfile1 ntfs.img
   ```

8. Use `ll` to list the contents of the current directory. You should see the deleted file. Once again, this utility yanks the file out of the file system.

9. Use the `cat` command to display the content of `rfile1`.

   Note that there are some additional follow-up questions in the Report.

Submission

Post the completed Report to Sakai by the deadline.

Appendix – Some Unix Commands

`cd` Change the current directory.

```
cd destination
```

With no "destination," the current directory will be changed to your home directory. If "destination" is "..", then the current directory will be changed to the parent of your current directory.

`cp` Copy a file.

```
cp source destination
```

This will copy the file with the name "source" to a new file with the name "destination." The "destination" can include a path to new directory.

`clear` Erase all the output on the current terminal and place the shell prompt at the top of the terminal.

`less` Display a page of a text file at a time in the terminal. (Also see `more`).

```
less file
```

To see another page press the space bar. To see one more line press the Enter key. To quit at any time press 'q' to quit.

`ls` List the contents and/or attributes of a directory or file

```
ls  directory
ls  file
```

With no "directory" or "file," this will display the contents of the current working directory.

`ll` This is a shortcut for "ls –l" to provide a more detailed (i.e., "long") printout of the information in the current directory.

`man` Manual

```
man command
```

Displays the manual page for "command". To see another page, press the space bar. To see one more lines, press the Enter key. To quit before reaching the end of the file, enter 'q'.

`more` Display a text file, a page at a time using the terminal. (Also, see `less`).

```
more file
```

To the next page, press the space bar. To see one more line, press the Enter key. At any time, press 'q' to quit.

`mv` Move and/or Rename a file/directory

```
mv source destination
```

The "source" file will be moved and/or renamed to the "destination."

File Deletion

`pwd` Display the present working directory

```
pwd
```