Lab 7: Teeny Tiny File System

CS 3841 - OS
Mitchell Johnstone

---

1. View the man page for free with "man free". What does each column mean? In your own words, what is the difference between free memory and available memory?

total -> total amount of memory on system

used -> how much memory is used

free -> How much available?

shared -> How much shared mem?

Buffers -> How much mem in the buffers

Cache -> How much mem in cache?

buff/cache -> buffers + cache memory

available -> Total bytes free

2. View the man page for mount with "man mount". Use the information in the man page to record which file system types your Linux installation supports.

Some of the filesystem types we've got going on in our Linux installation:

- Binfmt_misc
- Tmpfs
- 9p
- Cgroup2
- Cgroup

3. Pick one of the file systems supported by our Linux installation and learn more about how it works (use Google to help). Give a summary of the file system. Document your resources.

Tmpfs is a temporary file system, which loads and unloads data into RAM. This makes accessing the files fairly efficient and fast, since it doesn't need to pull the data from main memory, instead being able to pull straight from RAM.

4. Create the RAM disk as outlined above. What do each of the parameters to "mount" mean?

-t type -> Specifices the filesystem type (in our case, tmpfs)

-o output -> Where to output the RAM

Size=512m -> our filesystem has a max of 512 Mb.

5. Record the output of free -h after creation of the RAM disk. What is different? Why?

Before:

	total	used	free	shared	buff/cache	available
Mem:	12Gi	83Mi	12Gi	0.0Ki	57Mi	12Gi
Swap:	4.0Gi	0B	4.0Gi			

After:

	total	used	free	shared	buff/cache	available
Mem:	12Gi	84Mi	12Gi	0.0Ki	59Mi	12Gi
Swap:	4.0Gi	0B	4.0Gi			

There's less available space now since we made the RAM disk!

Why? Because we just mounted a new file system in RAM! So there's less available RAM since some is dedicated to the file system.

6. Based on the observation of the free command before adding the ram disk, and after adding it and adding files to it, in which category of memory does the ram drive appear to reside?

The used is a bit more, as well as the buff. As well, the used space is a bit less too.

7. Record the status of the RAM disk with the command df -h. What can you learn about the file system by running df?

Running df-h got the following result:

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sdb	251G	6.6G	232G	3%	/
tmpfs	6.2G	0	6.2G	0%	/mnt/wsl
tools	472G	122G	351G	26%	/init
none	6.2G	0	6.2G	0%	/dev
none	6.2G	8.0K	6.2G	1%	/run
none	6.2G	0	6.2G	0%	/run/lock
none	6.2G	0	6.2G	0%	/run/shm
none	6.2G	0	6.2G	0%	/run/user
tmpfs	6.2G	0	6.2G	0%	/sys/fs/cgroup

drivers 472G 122G 351G 26% /usr/lib/wsl/drivers

lib 472G 122G 351G 26% /usr/lib/wsl/lib

C:\ 472G 122G 351G 26% /mnt/c

tmpfs 512M 0 512M 0% /media/ramdisk

As can be seen in the last item, our ramdisk file a tmpfs filesystem, which is a temporary file system

8. Copy or download a large file into the RAM disk. I suggest a large pdf file (maybe around 25M). Interact with this file (such as opening a pdf with a viewer) in the RAM disk as well as a copy on the normal file system. What differences do you notice (e.g. performance, etc.)? Why do you think these difference happen or not?

We made a very large file by doing: Yes > large.txt

Eventaully, the tmpfs ran out of memory, but the file was written.

It was much faster opening files since the contents were all in cache/ram.

9. Record the output of free -h and df -h now that the RAM disk has a file in it. How does the output compare to what you recorded before? Does the output make sense? Why?

We can now see the following changes from free -h and f -h:

	total	used	free	shared	buff/cache	available
Mem:	12Gi	83Mi	11Gi	512Mi	573Mi	11Gi

1.

Running df-h got the following result:

Filesystem Size Used Avail Use% Mounted on

/dev/sdb 251G 6.6G 232G 3% /

tmpfs 6.2G 0 6.2G 0% /mnt/wsl

tools 472G 122G 351G 26% /init

none 6.2G 0 6.2G 0% /dev

none 6.2G 8.0K 6.2G 1% /run

none 6.2G 0 6.2G 0% /run/lock

none 6.2G 0 6.2G 0% /run/shm

none 6.2G 0 6.2G 0% /run/user

tmpfs 6.2G 0 6.2G 0% /sys/fs/cgroup

drivers 472G 122G 351G 26% /usr/lib/wsl/drivers

lib 472G 122G 351G 26% /usr/lib/wsl/lib

C:\ 472G 122G 351G 26% /mnt/c

tmpfs 512M 512M 0 100% /media/ramdisk

Now, we've taken up all space in the ramdisk. Ooof.

2. Record the contents /etc/fstab. What does each entry mean? Consult "man fstab" for help.

Contents:

LABEL=cloudimg-rootfs / ext4 defaults 0 0

This means that there's a default disk at /.

3. What would an entry in /etc/fstab need to look like to create the 512MB tmpfs RAM disk when the system boots?

Name location file system type options 0 0

tmpfs /media/ramdisk tmpfs size=512m

- contents of flash.img:

johnstonem@MSOE-PF3TQD7W:/media/myimage$ hexdump –C flash.img

hexdump: –C: No such file or directory

0000000 3ceb 6d90 666b 2e73 6166 0074 0102 0001

0000010 0002 1002 f827 0027 0020 0040 0000 0000

0000020 0000 0000 0080 3929 a206 4e84 204f 414e

0000030 454d 2020 2020 4146 3154 2036 2020 1f0e

0000040 5bbe ac7c c022 0b74 b456 bb0e 0007 10cd

0000050 eb5e 32f0 cde4 cd16 eb19 54fe 6968 2073

0000060 7369 6e20 746f 6120 6220 6f6f 6174 6c62

0000070 2065 6964 6b73 202e 5020 656c 7361 2065

0000080 6e69 6573 7472 6120 6220 6f6f 6174 6c62

0000090 2065 6c66 706f 7970 6120 646e 0a0d 7270

00000a0 7365 2073 6e61 2079 656b 2079 6f74 7420

00000b0 7972 6120 6167 6e69 2e20 2e2e 0d20 000a

00000c0 0000 0000 0000 0000 0000 0000 0000 0000

*

00001f0 0000 0000 0000 0000 0000 0000 0000 aa55

0000200 fff8 ffff 0000 0000 0000 0000 0000 0000

0000210 0000 0000 0000 0000 0000 0000 0000 0000

*

0005000 fff8 ffff 0000 0000 0000 0000 0000 0000

0005010 0000 0000 0000 0000 0000 0000 0000 0000

*

04e2000

1. Where are the mounted partitions of '/dev/sda' mounted? Why do suspect that two partitions are used? What is an advantage of partitioning the disk in this way?
    1. One for boot, one for the rest of the data. This section is reserved for the boot up sequence ,to allow for a very small boot up section.
2. Look at the image file with hexdump (hexdump –C flash.img | more). NOTE: make sure the image is **unmounted** before you do this. Can you find your files? How about the ones you deleted? How did you find them or why couldn't you?
    1. Yes! We can find some of my files!

2. Each file is a FAT Directory entry. Files get created, and they put it into memory. This puts the file details in memory. Even when the files are deleted, we can see them, since we didn't fully wipe it out..

3. We can read the FAT entry cluster details to follow the cluster chain. Using those indices, we can get the data.

3. Show a capture of the hexdump (of the flash.img) containing the directory entries for the files you added to the image as well as the ones you added and then deleted. What is different?

Hexdumps of the flash.img was as follows:

```
0000000 3ceb 6d90 666b 2e73 6166 0074 0102 0001
0000010 0002 1002 f827 0027 0020 0040 0000 0000
0000020 0000 0000 0080 3929 a206 4e84 204f 414e
0000030 454d 2020 2020 4146 3154 2036 2020 1f0e
0000040 5bbe ac7c c022 0b74 b456 bb0e 0007 10cd
0000050 eb5e 32f0 cde4 cd16 eb19 54fe 6968 2073
0000060 7369 6e20 746f 6120 6220 6f6f 6174 6c62
0000070 2065 6964 6b73 202e 5020 656c 7361 2065
0000080 6e69 6573 7472 6120 6220 6f6f 6174 6c62
0000090 2065 6c66 706f 7970 6120 646e 0a0d 7270
00000a0 7365 2073 6e61 2079 656b 2079 6f74 7420
00000b0 7972 6120 6167 6e69 2e20 2e2e 0d20 000a
00000c0 0000 0000 0000 0000 0000 0000 0000 0000
*
00001f0 0000 0000 0000 0000 0000 0000 0000 aa55
0000200 fff8 ffff 0000 0000 0000 0000 0000 0000
0000210 0000 0000 0000 0000 0000 0000 0000 0000
*
0005000 fff8 ffff 0000 0000 0000 0000 0000 0000
0005010 0000 0000 0000 0000 0000 0000 0000 0000
```

4. Remount the image, change directory to the directory where the file system is mounted. Now try to unmount the file system with "umount /media/myimage". Did it work successfully? Why or why not?

    1. No, because we count unmount from the mount, or else it wouldn't know where to put the user. This creates an error, so it simply doesn't unmount.