Volume Access

Introduction and/or Background

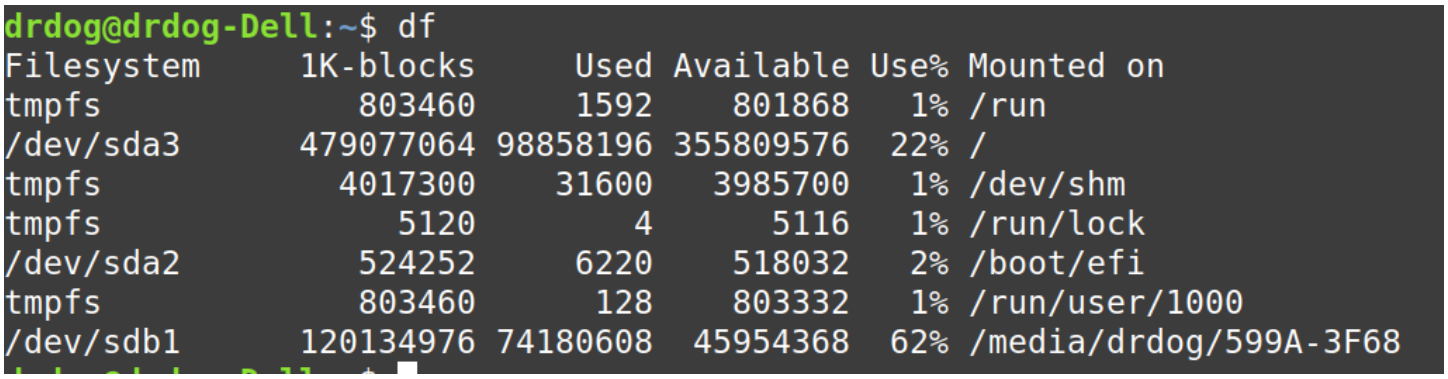
Employ common Linux sysadmin tools to create and format primary and logical partitions This activity covers the lsblk, fdisk, gparted, and mkfs commands

History

In the early development of Unix the skunk works at AT&T Bell Labs were poor boys and girls indeed. Their lab consisted of the cast offs that other units within AT&T were no longer using. Another words a junkpile. That presented a unique challenge to the developers -- How does one unify a file system when the hardware that would store it lacked any sort of standard at the time? There were no ATAPI, ATA, SCSI, SATA interface standards in those early days as every computer manafacturer had their own set of standards to go by for each machine they sold.

As a result the team developed the concept of the Unix BLOCK. A BLOCK was a given sector length x given number of heads on the storage device. That was later updated to also include number of cylinders. As a consequence of this arrangement it was possible regardless of the device at the time, if the signalling could be acquired, then the data could be retrieved/stored. It was then possible to connect a Honeywell or DEC disk pack to a IBM main frame and make it work. With the arrival of standards the idea of structured BLOCKS dictated by harware was replaced with Logical Block Addressing (LBA).

One can still see block addressing using the df command:



```
drdog@drdog-Dell:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
tmpfs             803460     1592    801868   1% /run
/dev/sda3      479077064 98858196 355809576  22% /
tmpfs            4017300    31600   3985700   1% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
/dev/sda2         524252     6220    518032   2% /boot/efi
tmpfs             803460      128    803332   1% /run/user/1000
/dev/sdb1      120134976 74180608  45954368  62% /media/drdog/599A-3F68
```

BLOCKS are now uniform thanks to LBA schema. With df we can discern the physical (/dev) and virtual storage (tmpfs) being addressed in the system.

Utilization

Keep in mind that BLOCK is merely a unit of storage and by itself if not of much use. Any file larger than BLOCK could not be stored for example. Building on top of BLOCK is inodes which is an index of multiple BLOCKs in a daisy chained fashion. Hence an inode can be a pointer to bundle of BLOCKs used to store a large file. An index of inodes is maintained in a Superblcok. The Superblock kept the inode list, state of the metadata on the drive, size of the device and filetype.

In the early days the entire Superblock was stored in memory. As the system ran only the inode table was updated. The entire Superblock was not stored back to disk till either the system was issued a shutdown order or the admin requested the system perform a forced write to disk. As a result Unix Admins were very wary of the power backup to their systems. A power outage could scramable a drive as none of the updates would have been written to disk. This is no longer an issue for most systems and Linux in particular has adopted the NTFS style of update on write so that the Superblock is current at all times.

As hardware has improved the Unix file system has kept pace. On a Linux system the native file format is ext2/ext3/ext4, ext4 being the most current. But Linux is not limited to just those 3. There is Reiser4, ZFS, BtrFS, and XFS. In addition Linux is capable of accessing some 80+ file systems of other types. Yes Linux can read a NTFS volume. One minor thing to keep in mind -- Just because Linux can access a volume does not mean it can run the programs contained on it. For example, Linux could see an Adobe Illustrator .EXE file but it cannot run it as it is a Windows program.

To use a file system Linux ultimately reads the /etc/fstab file. That file outlines where the disk is located and in what state it should be addressed for use. At a minimum Linux uses two paritions on a volume. A root system where all the files are located and a swap space for virtual memory.

Devices and Mount Points

A device in a Linux system can be physical(serial ports, USB, disk drive, keyboard, etc) or virtual (terminals, vhost). All devices are defined in the /dev directory. All devices are mounted, i.e. recognized by the Linux kernel. Devices are addressed in a round robin fashion as needed by the kernel. The interface between the device and the kernel is called a mount point. The preferred mount point for devices is the /dev directory. These are all local system type. Another mont pont is /mnt. This is intended as the remote or network for remote file systems that are to be added to the local file structure. Progams like Samba and NFS are programs that are used to facilitate remote mounting.

Objectives
In this project/lab the student will:

- Gain familiarity with volume access tools

Equipment/Supplies Needed

- As outlined in Lab 0.0.1.

Procedure
Perform the steps in this lab in the order they are presented to you. Answer all questions and record the requested information. Use the Linux Virtual Machine to perform lab activities as directed. Unless otherwise stated, all tasks done as a non-root user. If root access is needed use the sudo command.

Assignment

Volume Management

Create two 20GB virtual drives for your VM. (See VMWare settings)
Power on the VM, open terminal and switch to root.

1 `fdisk -l`

You should see three Disks: sda, sdb, and sdc

2 `fdisk /dev/sdb`

This will start the fdisk program. View all options: `m`
Create a partition: `n`
Create first partition as primary: `p`
Set partition `1`
Set first sector as `2048`
Set last sector as `+10G`

3 Create a partition: `n`
Create second partition as extended: `e`
Set partition `2`
Leave first sector value as default: `Enter`
Leave last sector value as default: `Enter`
Write partition table to disk: `w`

4 `fdisk -l`

Record a screenshot

Partition using gparted

5 su -
6 apt-get install gparted
7 gparted

Confirm your work by starting gparted. You can use the gui or invoke it from the command line.

8 Select /dev/sdc from the drop-down menu
- 8.a Select Device -> Partition Table; confirm.
- 8.b Select Partition > New from the menu
- 8.c New Size = 10240
- 8.d Create as = Primary Partition
- 8.e File system = ntfs
- 8.f Click Add
- 8.g Partition -> New from the menu to create another partition
- 8.h Create as Primary Partition
- 8.i New Size: don’t change
- 8.j File System: ext4
- 8.k Click Add
- 8.l Click the Green check to commit

Open gparted again and Record screenshots of both virtual drives in gparted.

Format sdb1 with mkfs

Still as root user, issue Command:

1 sudo mkfs.ntfs /dev/sdb1
2 sudo mkfs.ext4 /dev/sdc2

Check results at the CLI with:

3 lsblk -f

Record a screenshot

Record the change with

4 `lsblk -f > ~/blockdevsMMDDYY`

Lab Submissions Proof: Provide screenshots as indicated in the lab; upload your proof to Canvas for grading.

Rubric

Checklist/Single Point Mastery

Concerns Working Towards Proficiency	Criteria Standards for This Competency	Accomplished Evidence of Mastering Competency
	Criteria #1 fdisk: (25 points)	Screen Shot #1
	Criteria #2 gparted disk2 (25 points)	Screen Shot #2
	Criteria #3 gparted disk3 (25 points)	Screen Shot #3
	Criteria #4 lsblk -f: (25 points)	Screen Shot #4