Revised: 1/27/2019

Files and Filesystem Navigation

Michael C. Hackett

Computer Science Department

Community College of Philadelphia

Lecture Topics

- Linux Directory Structure
  - Filesystem Hierarchy Standard
  - Navigating Directories
- Files and Directories
  - File Types
  - Filenames
  - Listing Files and their Types
  - Wildcard Metacharacters
- Displaying text files
- Displaying binary files
- Searching for text in files
- Editing text files
  - The vi Editor
- Additional text processing tools

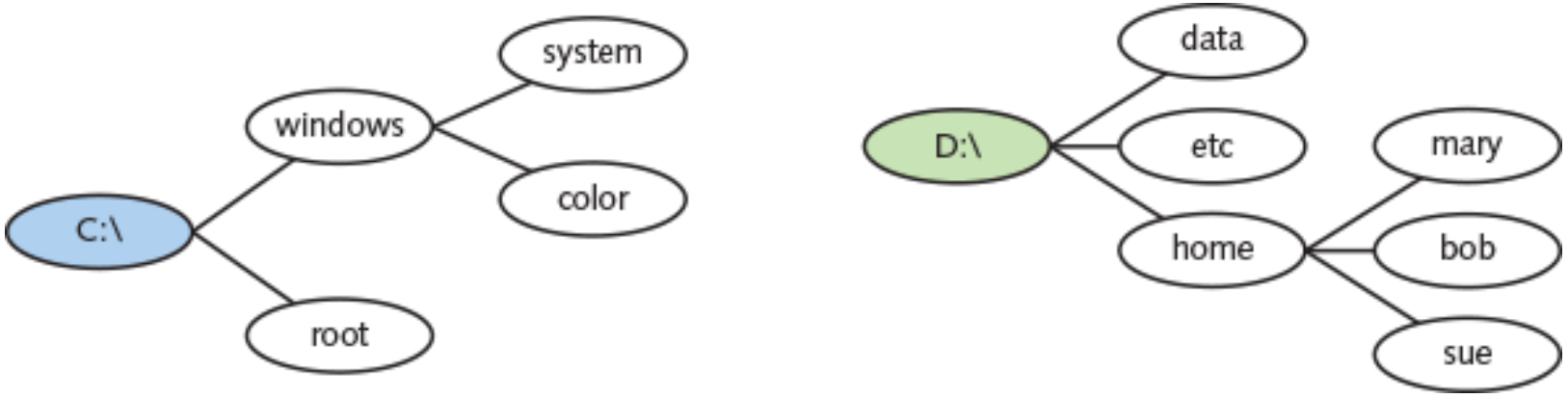
Linux Directory Structure

- Linux uses a logical directory tree to organize files into ***directories*** (or ***folders***)
  - Directories are special types of files used by the filesystem to organize other files in the logical directory tree.

- When files are stored into a directory, the file is stored in the filesystem of a hard drive partition.

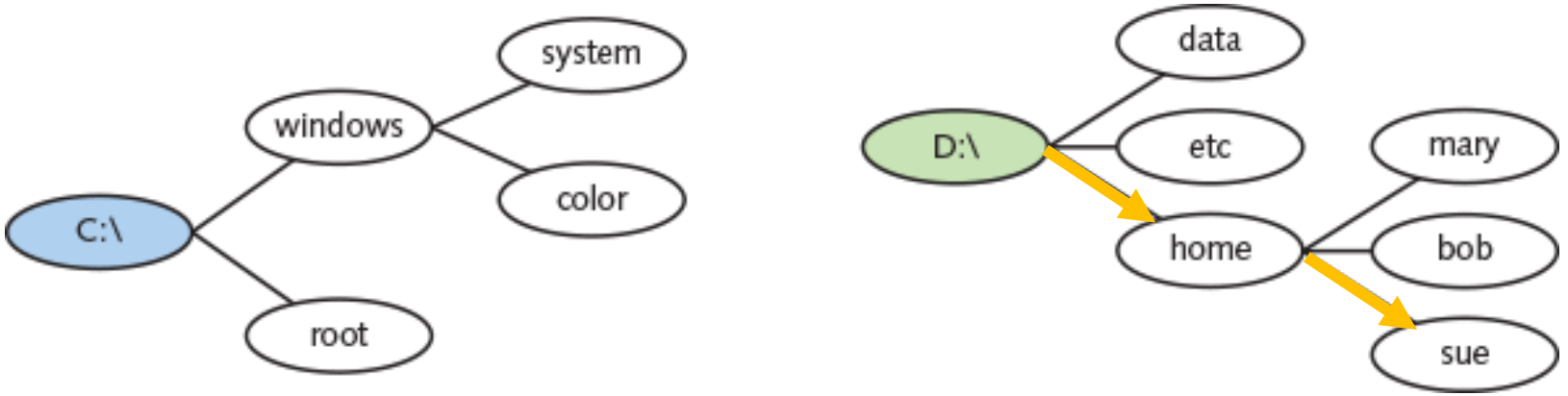
Linux Directory Structure

- You may be familiar with the Windows directory tree structure.
  - Each partition is given a drive letter like C: or D:
  - Has a root directory indicated by the \ character



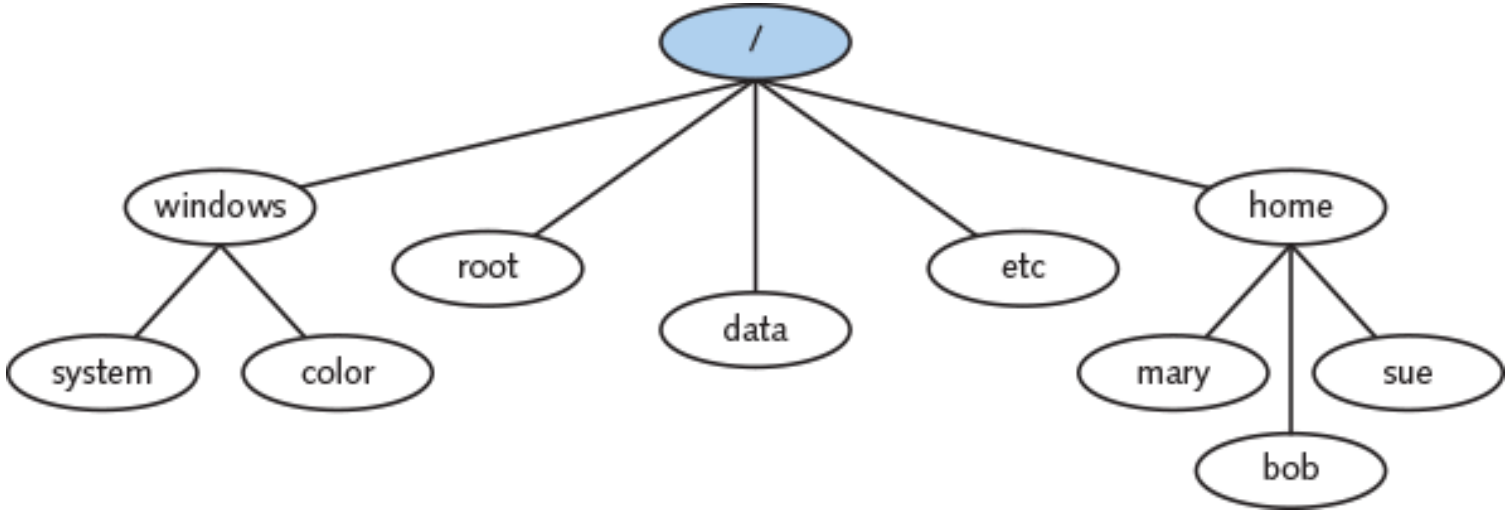
Linux Directory Structure

- The ***absolute pathname*** is the full path to a file or directory, beginning at the root directory.
  - **D:\home\sue**



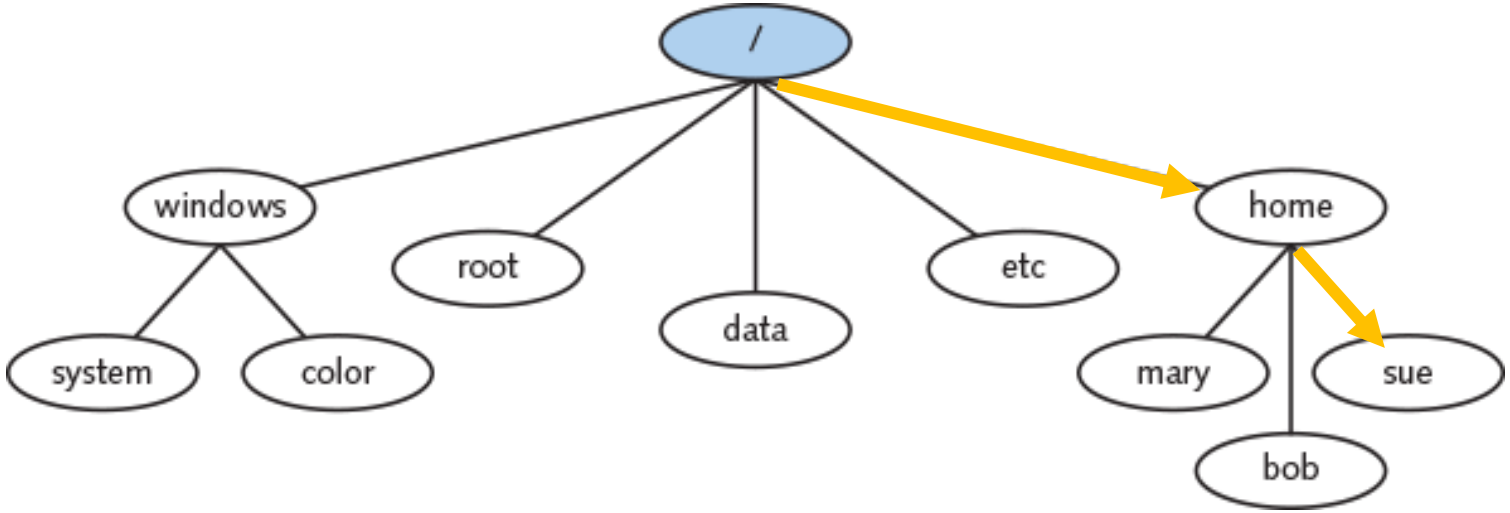
Linux Directory Structure

- Linux uses a similar structure, but without drive letters.
  - Contains a single root directory, identified by the character / (not \ like in Windows)
  - Partitions and drives are mounted (or “attached”) to directories in the directory tree.



Linux Directory Structure

- Absolute path to **/home/sue**
  - **Use forward slash / in Linux**; (Backslash \ on Windows)



Filesystem Hierarchy Standard

- The Filesystem Hierarchy Standard (FHS) specifies the names of files and directories (and their locations) on Linux systems.

- Makes systems compatible with each other, since they'll know where certain files and directories should be

https://refspecs.linuxfoundation.org/fhs.shtml

Filesystem Hierarchy Standard

- The top-most directory in a Linux filesystem, as shown in previous slides) is the root directory, represented by a single forward slash **/**

- Below the root directory are numerous, standardized folders
  - Consistent across almost every Linux distribution
  - Some distributions many include additional folders

Filesystem Hierarchy Standard

- **/bin** – Executable command-line (binary) programs and utilities
  - The **`ls`** command (seen in the previous lecture) is stored here: **`/bin/ls`**
- **/boot** – Files used to boot the operating system
- **/dev** – Hardware and software **dev**ice drivers; files that represent devices connected to the system
  - The second partition of the first SATA hard disk would be represented in this directory as: **`/dev/sda2`**

Filesystem Hierarchy Standard

- **/etc** – Configuration files
  - For example, the file that contains DNS Server information to the system is contained in **`/etc/resolv.conf`**
- **/home** – User home directories
  - A user named **`student`** would (typically) have the following home directory: **`/home/student`**
- **/lib** – Software/program libraries used by the kernel, utilities, and other programs.

Filesystem Hierarchy Standard

- **/media** – Mount points for removable media like CD-ROMs
- **/mnt** – Mount point for temporarily mounted filesystems
- **/opt** – Additional/optional files required by programs installed on the system
- **/proc** – A virtual filesystem that contains files that are continually updated with kernel information
- **/root** – Home directory of the **root user**
  - Normal users have home directories as individual subdirectories in **/home**

Filesystem Hierarchy Standard

- **/sbin** – Programs used in the boot process and the root user
- **/sys** – A virtual filesystem that stores information about devices
- **/tmp** – Temporary files
- **/usr** – Small programs and files accessible by all users
  - Contents are read-only
- **/var** – Files that constantly change as the system runs.
  - Log files, for the most part

Filesystem Hierarchy Standard

- A listing of the root directory on Fedora 31

```
[mhackett@localhost ~]$ ls /
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

Navigation

- After logging in, you are normally placed in your home folder in the directory tree.
  - Regular users: **`/home/`*`username`***
  - Root (superuser) account: **`/root`**

- The **~** metacharacter is used by the shell to refer to the current user's home directory.
  - Works for both regular users and root

Navigation

- Logging in as a normal user:

```
localhost login: mhackett
Password:
Last login: Sat Nov 30 15:13:49 on tty2
[mhackett@localhost ~]$
```

- Logging in as root:

```
localhost login: root
Password:
Last login: Sat Nov 30 15:42:30 on tty2
[root@localhost ~]# _
```

Navigation – Current Location

- To print the current directory you are in, enter the **pwd** command.
  - "**P**rint **W**orking **D**irectory"
  - No options or arguments needed.

```
localhost login: mhackett
Password:
Last login: Sun Dec  1 13:49:47 on tty2
[mhackett@localhost ~]$ pwd
/home/mhackett
[mhackett@localhost ~]$ _
```

```
localhost login: root
Password:
Last login: Sun Dec  1 13:54:33 on tty2
[root@localhost ~]# pwd
/root
[root@localhost ~]# _
```

Navigation – Changing Directories

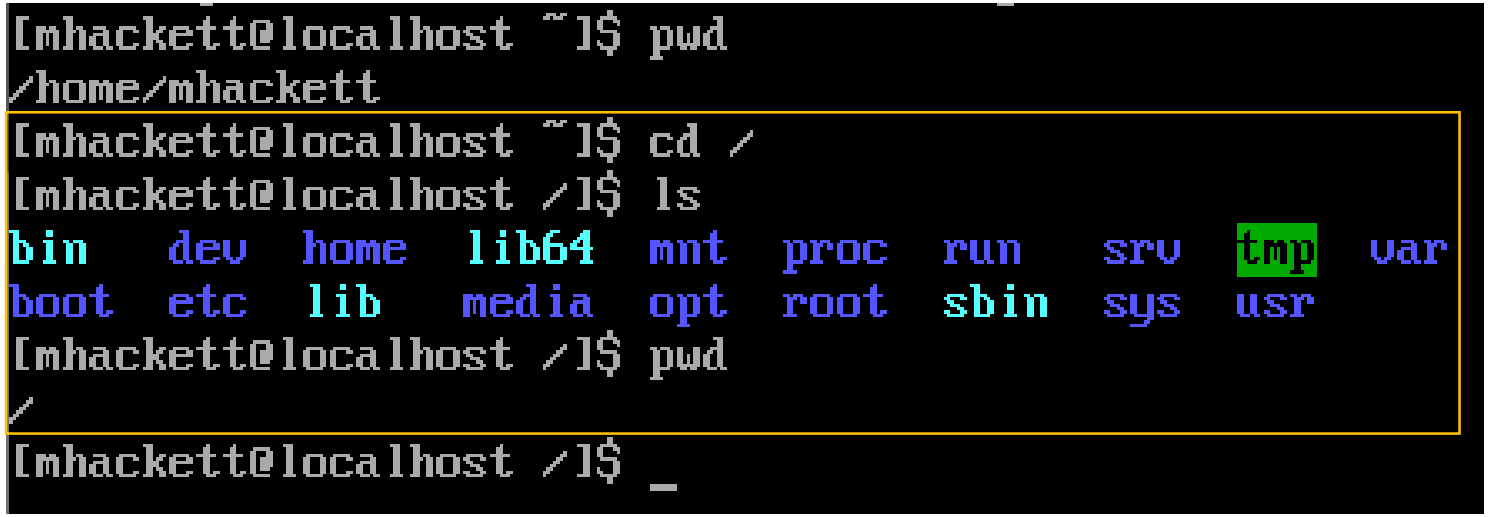
- The command used to navigate into a different directory is the **cd** command.
  - "**C**hange **D**irectory"

cd [argument]

- Argument is the directory you wish to navigate to.

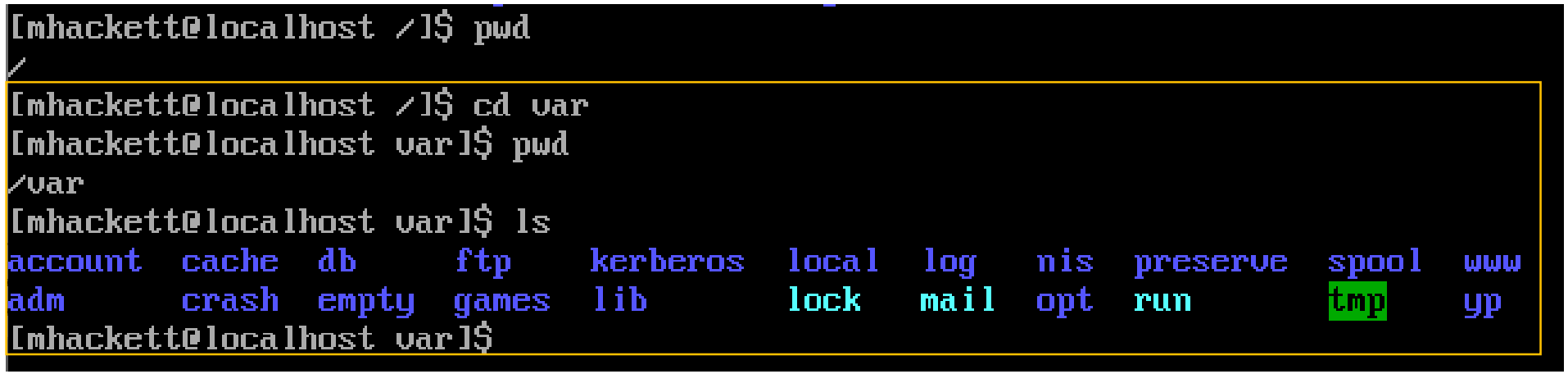
Navigation – Changing Directories

- The command **cd /** would navigate you to the filesystem hierarchy's root directory.



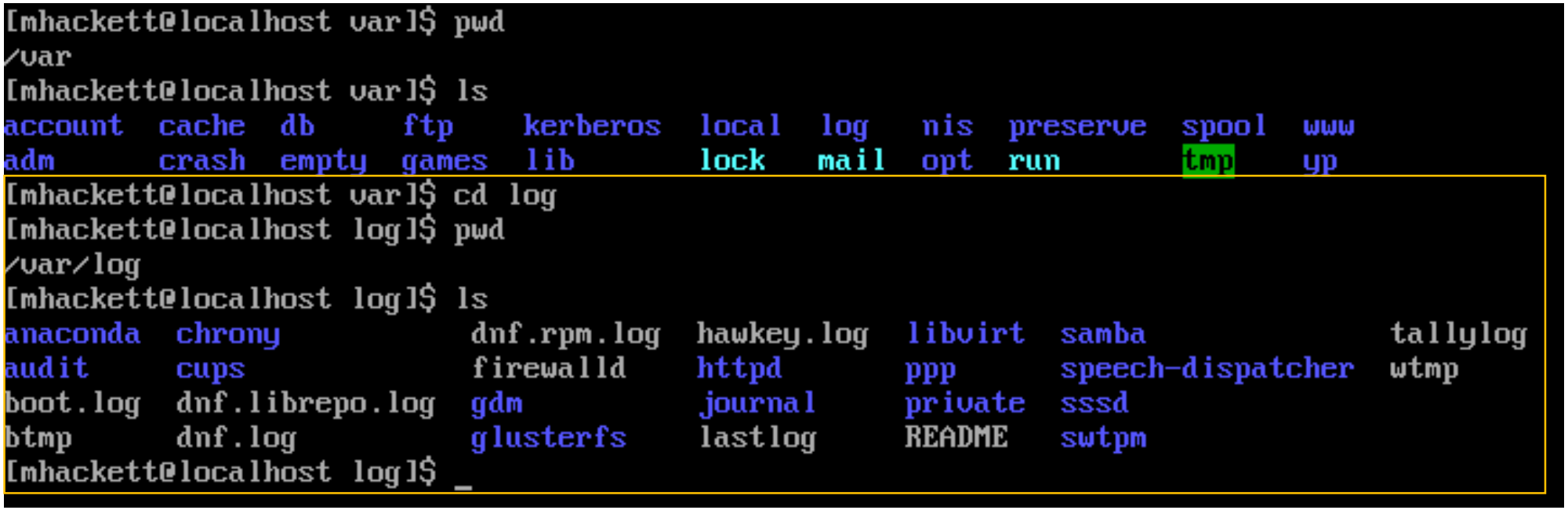
Navigation – Changing Directories

- From here, we could navigate into the var subdirectory with the command **`cd var`**



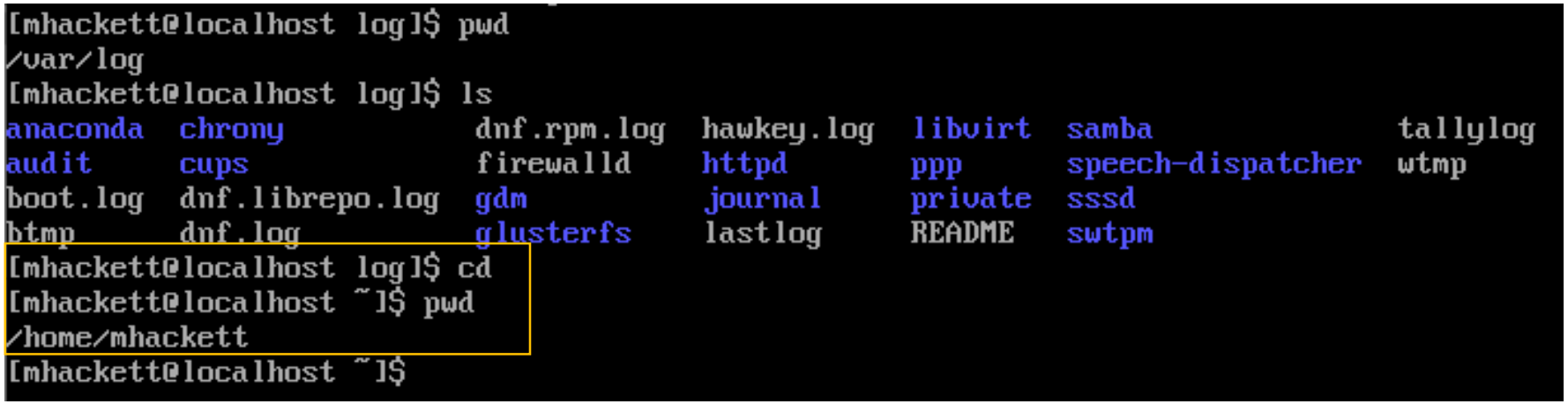
Navigation – Changing Directories

- From here, we could navigate into the log subdirectory with the command **`cd log`**



Navigation – Changing Directories

- To navigate quickly back to your home directory, enter the **cd** command without any arguments.



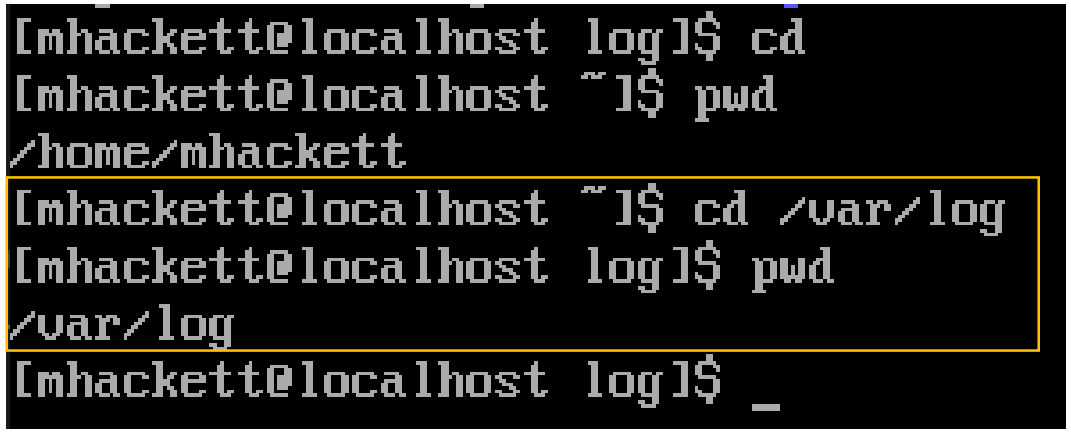
- The following would also work to return to your home directory:
  - **cd ~**

Navigation – Changing Directories

- We can move back to the /var/log directory, by entering

  `cd /var/log`

  - We provide the **`cd`** command with the *absolute* path.
  - The previous examples used *relative* paths; We moved based on where we currently were in the filesystem.

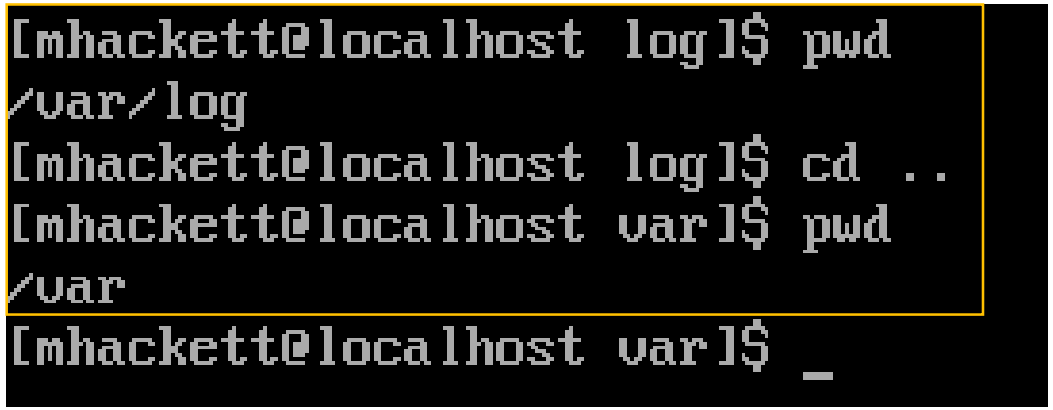


Navigation – Changing Directories

- To move up one directory, enter

  `cd ..`

  - The **`..`** metacharacter is used to refer to the current directory's parent.



Navigation – Changing Directories

- From this location, (/var) we could move to the Downloads directory in our home directory using the following commands

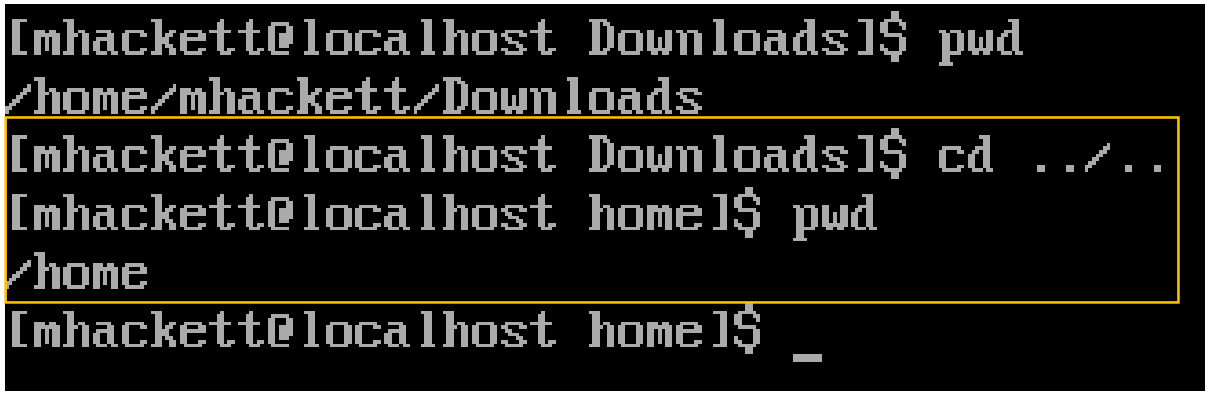
  cd ~/Downloads

```
[mhackett@localhost var]$ pwd
/var
[mhackett@localhost var]$ cd ~/Downloads
[mhackett@localhost Downloads]$ pwd
/home/mhackett/Downloads
[mhackett@localhost Downloads]$
```

Navigation – Changing Directories

- From here, we could move up two directories with the following command

  `cd ../..`



Types of Files

- Text Files
  - Most configuration files are text files
  - Data in the file is stored in plaintext, human-readable format

- Binary Files
  - Data in the file is stored in machine language (binary)
  - Not a human-readable format
  - Used by executable programs

Types of Files

- Executable Programs
  - Compiled programs that can be executed/run as processes in the system
- Directory Files
  - Each directory is a special file that organizes the text, binary, executable, and other files stored in the filesystem

Types of Files

- Linked Files
  - Files that are associated with each other
  - Could represent the same data or act as a shortcut to a different file

- Device Files
  - Files that represent system devices
  - Used by commands that interact with the physical device they represent

Types of Files

- Named Pipe Files
  - Identify channels that pass information from one process to another
- Socket Files
  - Allows a process on another computer to write to a file on the local computer while another process reads that file

Filenames

- Filenames in Linux usually consist of
  - Alphanumeric characters (a-z, A-Z, 0-9)
  - Underscores (_)
  - Dashes (-)
  - Periods (.)
- May be up to 255 characters in length
- Filenames that begin with a period are **hidden files**, and are only seen in a directory listing when using the **`-a`** option with the **`ls`** command.

Filename Extensions

- Filenames typically end with an extension
  - Characters that indicate the type of file

- Example: **`myfile.txt`**
  - **`.txt`** is the file's extension

- A file is not required to have an extension

Common File Extensions

Extension	Description	Type
.bin	Binary executable programs (akin to .exe programs in Windows)	Binary
.c	C source code files	Text
.cc, .cpp	C++ source code files	Text
.html, .htm	Hypertext Markup Language (web page) files	Text
.txt	Text files	Text

Common File Extensions

Extension	Description	Type
.tar	Archive files	Binary
.gz, .bz2, .xz, .Z	Compressed files	Binary
.tar.gz, .tgz, .tar.bz2, .tar.xz, .tar.Z	Compressed archives	Binary
.conf, .cfg	Configuration files	Text
.so	Shared object libraries	Binary
.o, .ko	Compiled objects	Binary

Common File Extensions

Extension	Description	Type
.pl	PERL programs	Text
.tcl	Tool Command Language programs	Text
.jpg, .jpeg, .png, .gif	Graphical images	Binary
.sh	Shell scripts	Text

Listing Files and their Type

- Using the **`-F`** option with the **`ls`** command will display a listing of files in a directory, along with a special symbol that denotes each file's type.

No Symbol	Text file, Binary file, or Device file
*	Executable file
/	Directory file
@	Linked file
=	Socket file
|	Named Pipe file

Listing Files and their Type

- Executing **`ls /etc`** produces a listing of the /etc directory

```
[mhackett@localhost home]$ ls /etc
abrt                            fonts           magic                       request-key.d
adjtime                         foomatic        mailcap                     resolv.conf
aliases                         fprintd.conf    makedumpfile.conf.sample    rpc
alsa                            fstab           man_db.conf                 rpm
alternatives                    fuse.conf       mcelog                      rsyncd.conf
anacrontab                      fwupd           mime.types                  rwtab.d
asound.conf                     gconf           mke2fs.conf                 rygel.conf
at.deny                         gcrypt          modprobe.d                  samba
audit                           gdbinit         modulefiles                 sane.d
authselect                      gdbinit.d       modules-load.d              sasl2
avahi                           gdm             motd                        scl
bash_completion.d               geoclue         motd.d                      security
bashrc                          glvnd           mtab                        selinux
bindresvport.blacklist          gnupg           mtools.conf                 services
binfmt.d                        GREP_COLORS     my.cnf                      sestatus.conf
bluetooth                       groff           my.cnf.d                    sgml
brlapi.key                      group           netconfig                   shadow
```

Listing Files and their Type

- Executing **`ls -F /etc`** produces a listing of the /etc directory including file type symbols

```
[mhackett@localhost home]$ ls -F /etc
abrt/                           fonts/          magic                       request-key.d/
adjtime                         foomatic/       mailcap                     resolv.conf
aliases                         fprintd.conf    makedumpfile.conf.sample    rpc
alsa/                           fstab           man_db.conf                 rpm/
alternatives/                   fuse.conf       mcelog/                     rsyncd.conf
anacrontab                      fwupd/          mime.types                  rwtab.d/
asound.conf                     gconf/          mke2fs.conf                 rygel.conf
at.deny                         gcrypt/         modprobe.d/                 samba/
audit/                          gdbinit         modulefiles/                sane.d/
authselect/                     gdbinit.d/      modules-load.d/             sasl2/
avahi/                          gdm/            motd                        scl/
bash_completion.d/              geoclue/        motd.d/                     security/
bashrc                          glvnd/          mtab@                       selinux/
bindresvport.blacklist          gnupg/          mtools.conf                 services
binfmt.d/                       GREP_COLORS     my.cnf                      sestatus.conf
bluetooth/                      groff/          my.cnf.d/                   sgml/
brlapi.key                      group           netconfig                   shadow
```

Listing Files and their Type

- We've seen the **`-l`** option with the **`ls`** command will display a long listing of files in a directory.

- We'll now take a closer look at what is displayed for each file.

Listing Files and their Type

- Executing **`ls -la ~`** to display a long listing of the current user's home directory including any hidden files.

```
[mhackett@localhost home]$ ls -la ~
total 28
drwx------. 14 mhackett mhackett 4096 Nov 30 14:35 .
drwxr-xr-x.  3 root     root       22 Nov 30 13:57 ..
-rw-------.  1 mhackett mhackett   84 Dec  1 13:54 .bash_history
-rw-r--r--.  1 mhackett mhackett   18 Aug  5 06:19 .bash_logout
-rw-r--r--.  1 mhackett mhackett  141 Aug  5 06:19 .bash_profile
-rw-r--r--.  1 mhackett mhackett  376 Aug  5 06:19 .bashrc
drwx------. 11 mhackett mhackett  226 Nov 30 14:22 .cache
drwx------. 12 mhackett mhackett  227 Nov 30 14:35 .config
drwxr-xr-x.  2 mhackett mhackett    6 Nov 30 14:19 Desktop
drwxr-xr-x.  2 mhackett mhackett    6 Nov 30 14:19 Documents
drwxr-xr-x.  2 mhackett mhackett    6 Nov 30 14:19 Downloads
-rw-------.  1 mhackett mhackett   16 Nov 30 14:19 .esd_auth
drwx------.  3 mhackett mhackett   19 Nov 30 14:19 .local
drwxr-xr-x.  4 mhackett mhackett   39 Nov 30 13:47 .mozilla
drwxr-xr-x.  2 mhackett mhackett    6 Nov 30 14:19 Music
drwxr-xr-x.  2 mhackett mhackett  100 Nov 30 14:34 Pictures
drwxr-xr-x.  2 mhackett mhackett    6 Nov 30 14:19 Public
drwxr-xr-x.  2 mhackett mhackett    6 Nov 30 14:19 Templates
-rw-r-----.  1 mhackett mhackett    5 Nov 30 14:20 .vboxclient-draganddrop.pid
drwxr-xr-x.  2 mhackett mhackett    6 Nov 30 14:19 Videos
[mhackett@localhost home]$
```

Listing Files and their Type

- We'll focus on the .bashrc file and .cache directory, because they appear next to each other

```
-rw-r--r--.  1 mhackett mhackett  376 Aug  5 06:19 .bashrc
drwx------. 11 mhackett mhackett  226 Nov 30 14:22 .cache
```

Listing Files and their Type

- The first character indicates the file's type

-	Text or binary file
d	Directory
l	Linked file (lowercase L)
n	Named Pipe file
s	Socket file
b	Device file (block device)
c	Device file (character device)

Listing Files and their Type

- The next ten characters indicates the file's **permissions** or *mode*

- Explained in a later lecture

Listing Files and their Type

- The next value indicates the number of *hard links* to a file

- Explained in a later lecture

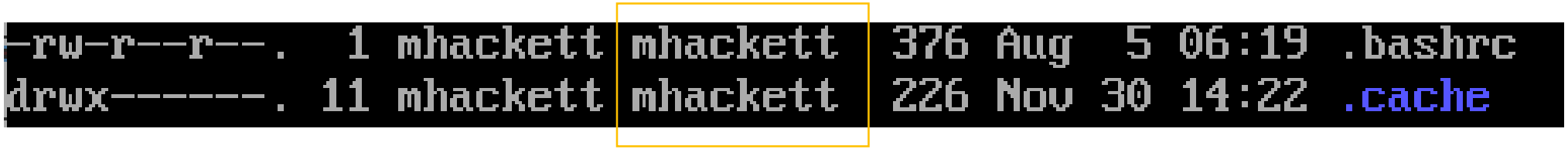
Listing Files and their Type

- The next value indicates the owner of the file

- Explained in a later lecture

Listing Files and their Type

- The next value indicates the name of the group of users that own the file



- Explained in a later lecture

Listing Files and their Type

- The next value indicates the size (in bytes) of the file

Listing Files and their Type

- The next value indicates the date the file was last accessed.

Listing Files and their Type

- The last value is the file's name.

```
-rw-r--r--.  1 mhackett mhackett  376 Aug  5 06:19 .bashrc
drwx------. 11 mhackett mhackett  226 Nov 30 14:22 .cache
```

The file command

- The **`file`** command provides information about a file's type
  - Syntax: **`file`** ***`filename`***

```
[mhackett@localhost ~]$ pwd
/home/mhackett
[mhackett@localhost ~]$ file .bashrc
.bashrc: ASCII text
[mhackett@localhost ~]$ file .cache
.cache: directory
[mhackett@localhost ~]$ file /dev/sda2
/dev/sda2: block special (8/2)
[mhackett@localhost ~]$ file /dev/tty1
/dev/tty1: character special (4/1)
[mhackett@localhost ~]$
```

Wildcard Metacharacters

- To filter the files and directories displayed by the ls command, we can use wildcard metacharacters.

- These characters allow us to specify file and directory names that match a certain pattern.

Wildcard Metacharacters

Wildcard Metacharacters	Description
*	Matches 0 or more characters in the name
?	Matches 1 character in the name
[abd]	Matches 1 character in the name, which must be a, b, or d
[a-d]	Matches 1 character in the name, which must be a, b, c, or d
[a-dA-D]	Matches 1 character in the name, which must be a, b, c, d, A, B, C, or D
[0-5]	Matches 1 character in the name, which must be 0, 1, 2, 3, 4, or 5
[!a-d]	Matches 1 character in the name, which must NOT be a, b, c, or d

Wildcard Metacharacters

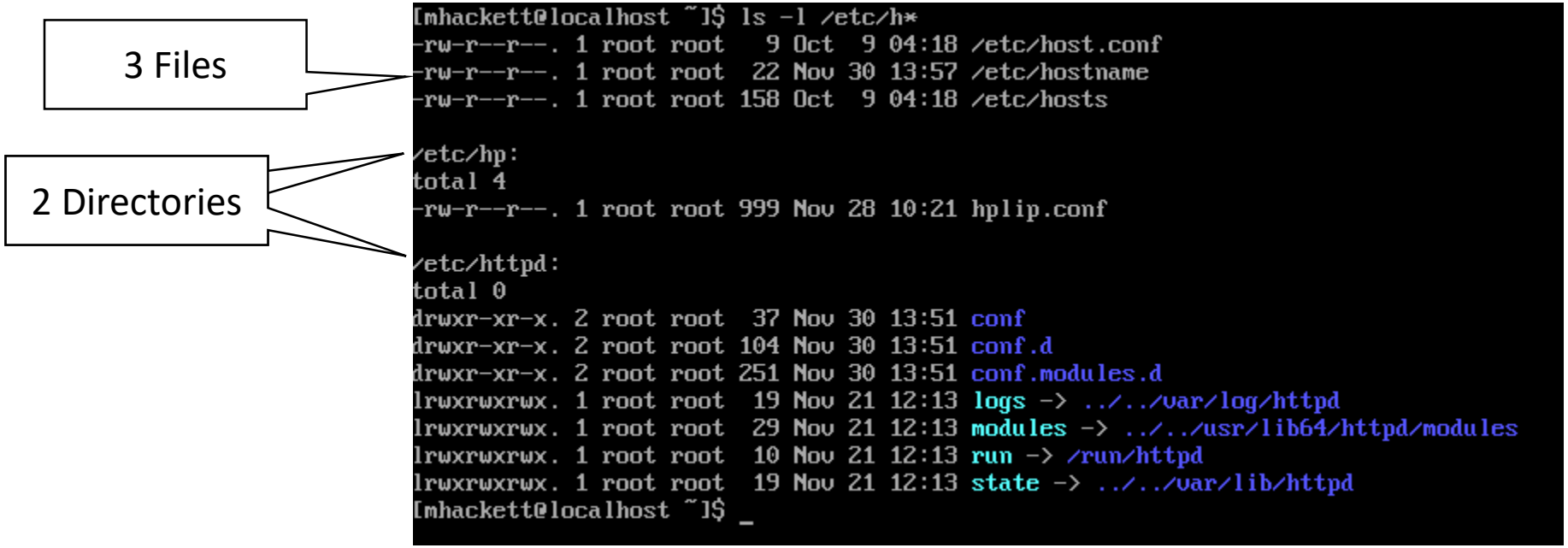
- We'll first try to do a long listing all files and directories in the /etc folder that begin with the letter h

- Executing **`ls -l h`** will return no results because there is no file named h in /etc

```
[mhackett@localhost ~]$ ls -l /etc/h
ls: cannot access '/etc/h': No such file or directory
[mhackett@localhost ~]$ _
```

Wildcard Metacharacters

- Executing **`ls -l /etc/h*`** will return:
  - Any files and directories in /etc that begin with h, followed by any series of characters (including no characters)



Wildcard Metacharacters

- Executing **`ls -l /etc/ho*`** will return:
  - Any files and directories in /etc that begin with ho, followed by any series of characters (including no characters)

```
[mhackett@localhost ~]$ ls -l /etc/ho*
-rw-r--r--. 1 root root   9 Oct  9 04:18 /etc/host.conf
-rw-r--r--. 1 root root  22 Nov 30 13:57 /etc/hostname
-rw-r--r--. 1 root root 158 Oct  9 04:18 /etc/hosts
[mhackett@localhost ~]$ _
```

Wildcard Metacharacters

- Executing **`ls -l /bin/l?`** will return:
  - Any files and directories in /bin that begin with l, followed by any one character

```
[mhackett@localhost ~]$ ls -l /bin/l?
-rwxr-xr-x. 1 root root  85912 Oct 17 03:37 /bin/ln
lrwxrwxrwx. 1 root root     26 Nov 30 13:50 /bin/lp -> /etc/alternatives/print-lp
-rwxr-xr-x. 1 root root 157984 Oct 17 03:37 /bin/ls
lrwxrwxrwx. 1 root root      2 Jul 26 07:40 /bin/lz -> uz
```

Wildcard Metacharacters

- Executing **`ls -l /bin/l??`** will return:
  - Any files and directories in /bin that begin with l, followed by any one character, followed by any one character

```
[mhackett@localhost ~]$ ls -l /bin/l??
-rwxr-xr-x. 1 root root   5441 Nov 19 12:33 /bin/ldd
lrwxrwxrwx. 1 root root     27 Nov 30 13:50 /bin/lpq -> /etc/alternatives/print-lpq
lrwxrwxrwx. 1 root root     23 Nov 30 13:50 /bin/lpr -> /etc/alternatives/print
-rwxr-xr-x. 1 root root  24984 Jul 25 20:25 /bin/lua
-rwxr-xr-x. 1 root root 188272 Aug 18 04:49 /bin/lz4
```

Wildcard Metacharacters

- Executing **`ls -l /bin/l[pn]`** will return:
  - Any files and directories in /bin that begin with l, followed by either p or n

```
[mhackett@localhost ~]$ ls -l /bin/l[pn]
-rwxr-xr-x. 1 root root 85912 Oct 17 03:37 /bin/ln
lrwxrwxrwx. 1 root root    26 Nov 30 13:50 /bin/lp -> /etc/alternatives/print-lp
```

Wildcard Metacharacters

- Executing **`ls -l /bin/l?[dq]`** will return:
  - Any files and directories in /bin that begin with l, followed by any one character, followed by either d or q

```
[mhackett@localhost ~]$ ls -l /bin/l?[dq]
-rwxr-xr-x. 1 root root 5441 Nov 19 12:33 /bin/ldd
lrwxrwxrwx. 1 root root   27 Nov 30 13:50 /bin/lpq -> /etc/alternatives/print-lpq
```

Wildcard Metacharacters

- Executing **`ls -l /bin/l?[d-z]`** will return:
  - Any files and directories in /bin that begin with l, followed by any one character, followed by a character between d and z

```
[mhackett@localhost ~]$ ls -l /bin/l?[d-z]
-rwxr-xr-x. 1 root root 5441 Nov 19 12:33 /bin/ldd
lrwxrwxrwx. 1 root root   27 Nov 30 13:50 /bin/lpq -> /etc/alternatives/print-lpq
lrwxrwxrwx. 1 root root   23 Nov 30 13:50 /bin/lpr -> /etc/alternatives/print
```

Wildcard Metacharacters

- Executing **`ls -l /bin/l?*[!c-z]`** will return:
  - Any files and directories in /bin that begin with l, followed by any one character, followed by any number of characters, and ends with a character that is not in c through z

```
[mhackett@localhost ~]$ ls -l /bin/l?*[!c-z]
lrwxrwxrwx. 1 root root      4 Sep 12 05:37 /bin/lastb -> last
-rwxr-xr-x. 1 root root  34544 Jul 28 16:19 /bin/libgtop_daemon2
-rwxr-xr-x. 1 root root  22424 Jul 28 16:19 /bin/libgtop_server2
lrwxrwxrwx. 1 root root      7 Sep 12 05:37 /bin/linux32 -> setarch
lrwxrwxrwx. 1 root root      7 Sep 12 05:37 /bin/linux64 -> setarch
-rwxr-xr-x. 1 root root 252320 Jul 27 09:32 /bin/lsusb
-rwxr-xr-x. 1 root root  24984 Jul 25 20:25 /bin/lua
-rwxr-xr-x. 1 root root 188272 Aug 18 04:49 /bin/lz4
```

Displaying Text Files

- The **`cat`** command is used to display the contents of a text file in the terminal.
- Syntax: **`cat [options] argument(file)`**

```
[mhackett@localhost ~]$ cat /etc/networks
default 0.0.0.0
loopback 127.0.0.0
link-local 169.254.0.0
```

Displaying Text Files

- The **`cat`** command's **`-n`** option will display line numbers.

```
[mhackett@localhost ~]$ cat -n /etc/networks
     1  default 0.0.0.0
     2  loopback 127.0.0.0
     3  link-local 169.254.0.0
[mhackett@localhost ~]$
```

Displaying Text Files

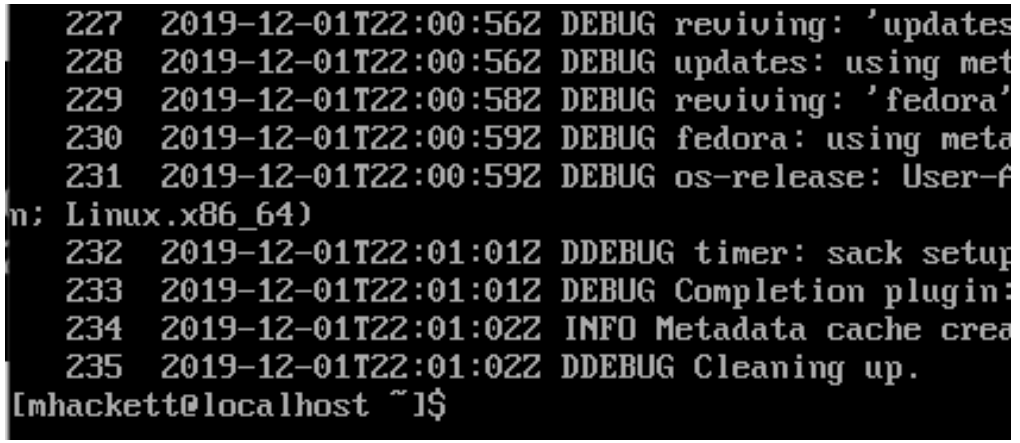
- The **`tac`** command is used to display the contents of a text file (in reverse) in the terminal.
- Syntax: **`tac [options] argument(file)`**

```
[mhackett@localhost ~]$ tac /etc/networks
link-local 169.254.0.0
loopback 127.0.0.0
default 0.0.0.0
[mhackett@localhost ~]$
```

Displaying Text Files

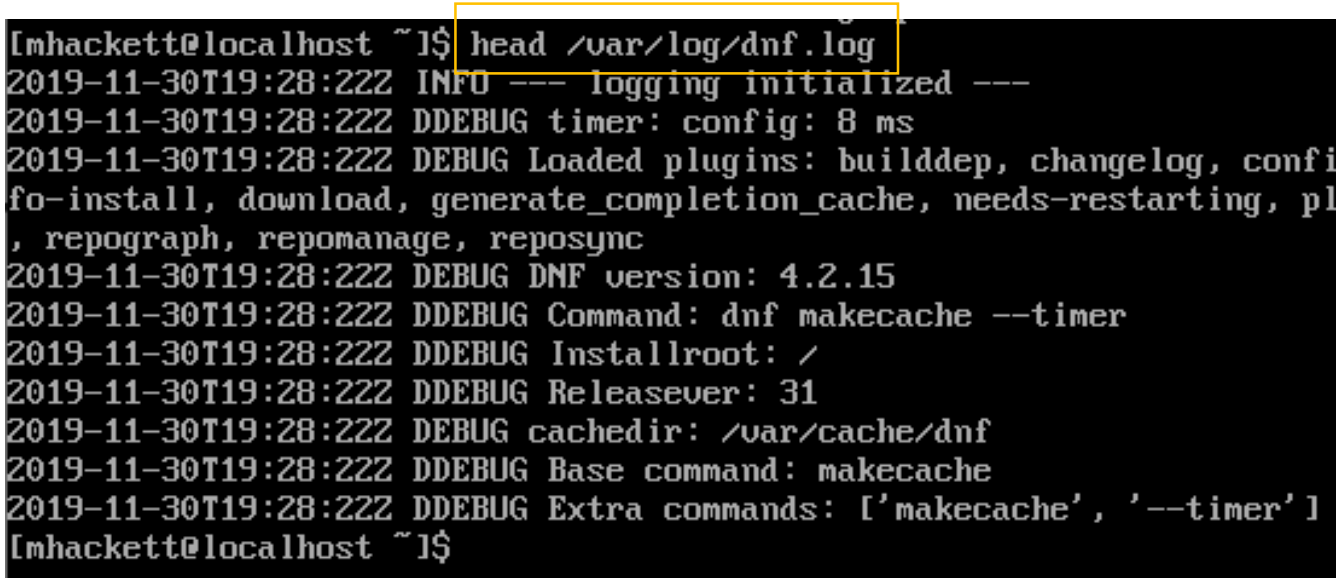
- Some text files are very big and we may wish to step through the text file more slowly.
  - For example, the file /var/log/dnf.log (currently on my system) has 235 lines which all get displayed at once

`cat -n /var/log/dnf.log`



Displaying Text Files

- The **head** command is used to display the first ten lines of a text file in the terminal.
- Syntax: **head [options] argument(file)**



Displaying Text Files

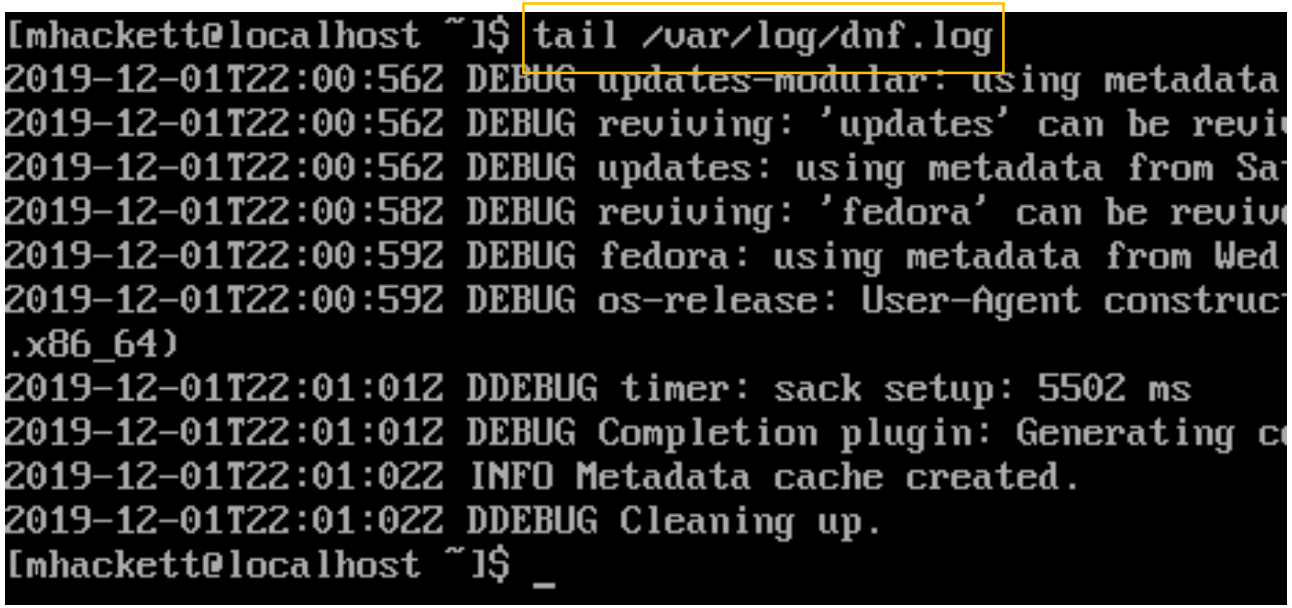
- A numeric option is available to specify the number of lines to be displayed.
  - If you want more or less than the default 10 lines

head -3 /var/log/dnf.log

```
[mhackett@localhost ~]$ head -3 /var/log/dnf.log
2019-11-30T19:28:22Z INFO --- logging initialized ---
2019-11-30T19:28:22Z DDEBUG timer: config: 8 ms
2019-11-30T19:28:22Z DEBUG Loaded plugins: builddep, chan
fo-install, download, generate_completion_cache, needs-re
, repograph, repomanage, reposync
[mhackett@localhost ~]$ _
```

Displaying Text Files

- The **`tail`** command is used to display the last ten lines of a text file in the terminal.
- Syntax: **`tail [options] argument(file)`**



Displaying Text Files

- A numeric option is available to specify the number of lines to be displayed.
  - If you want more or less than the default 10 lines

`tail -5 /var/log/dnf.log`

```
[mhackett@localhost ~]$ tail -5 /var/log/dnf.log
2019-12-01T22:00:59Z DEBUG os-release: User-Agent construc
.x86_64)
2019-12-01T22:01:01Z DDEBUG timer: sack setup: 5502 ms
2019-12-01T22:01:01Z DEBUG Completion plugin: Generating c
2019-12-01T22:01:02Z INFO Metadata cache created.
2019-12-01T22:01:02Z DDEBUG Cleaning up.
[mhackett@localhost ~]$ _
```

Displaying Text Files

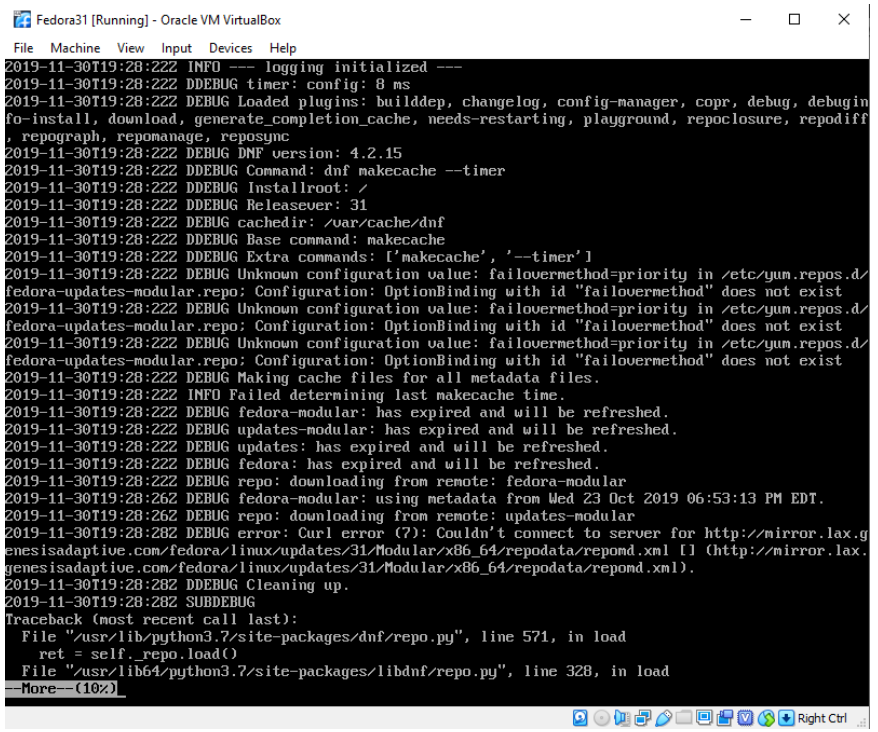
- The **`more`** command is used to display a text file in pages in the terminal.

- Syntax: **`more [options] argument(file)`**

Displaying Text Files

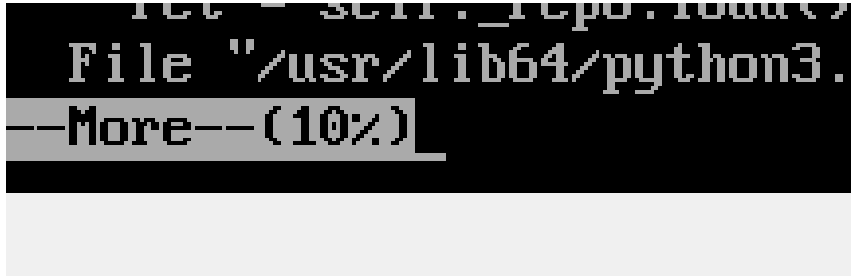
- When opening a text file with the **more** command, the first page will fill the entire terminal.

  more /var/log/dnf.log



Displaying Text Files

- The percentage at the bottom will tell you how far you have progressed through the file



Displaying Text Files

- The following keys are used to advance through the file:

Spacebar	Advance one page forward
Enter	Advance one line forward

- Other key commands:

q	Quits/Exits the more command
h	Help Menu

Displaying Text Files

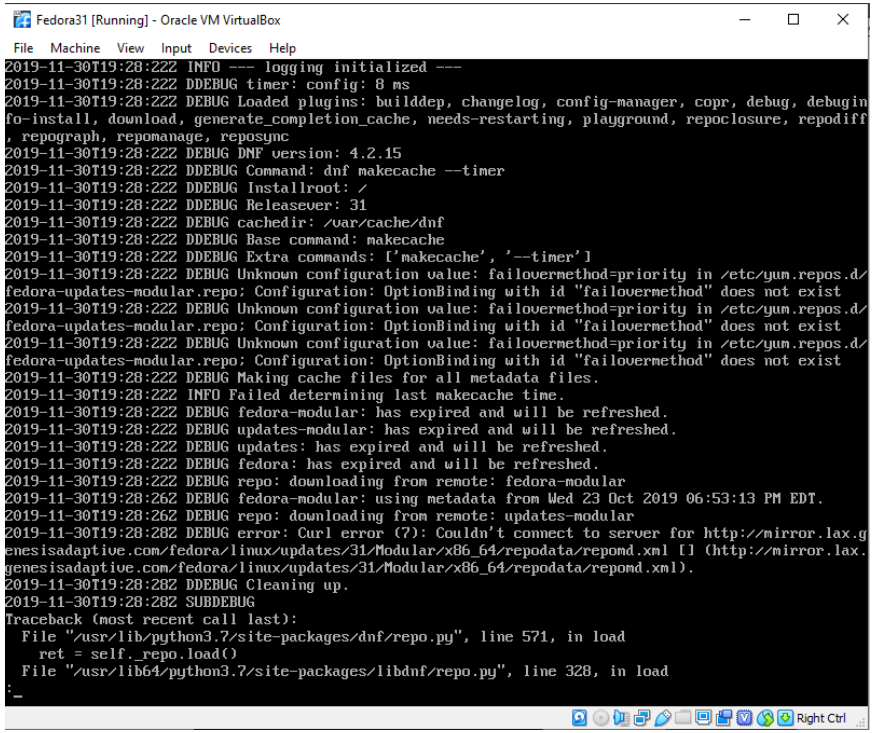
- The **`less`** command is used to display a text file in pages in the terminal.
  - Very similar to the more command

- Syntax: **`less [options] argument(file)`**

Displaying Text Files

- When opening a text file with the **`less`** command, the first page will fill the entire terminal.

  `less /var/log/dnf.log`



Displaying Text Files

- The less command does not show a percentage at the bottom.

- The following keys are used to advance through the file:

Spacebar	Advance one page forward
Enter	Advance one line forward
Up Arrow	**Advance one line forward**
Down Arrow	**Advance one line backward**
Page Up	**Advance one page forward**
Page Down	**Advance one page backward**

- Other key commands:

q	Quits/Exits the less command
h	Help Menu

Displaying Binary Files

- To view a binary file, you typically need to open the file with a program that understands the binary information contained in the file.
  - It is not safe to try to read a binary file with text display programs like cat and less.

- The **`strings`** command takes a binary file and tries to retrieve any human-readable text from the file's data.
  - The text is displayed in the terminal

Displaying Binary Files

- Syntax: **`strings [options] argument(file)`**

- For example, the command **`strings /bin/ls`** would display any text contained inside of the binary executable **`ls`** program

Displaying Binary Files

- Another way to view a binary files is with the od command.
- The **od** command takes a binary file and displays it in octal (base 8)
- Syntax: **`od [options] argument(file)`**

Displaying Binary Files

- For example, the command **`od /bin/cd`** would display the binary executable **`cd`** program's data in base 8

```
[root@localhost ~]# od /bin/cd
0000000 020443 072457 071163 061057 067151 071457 005150 072542
0000020 066151 064564 020156 062143 021040 040044 005042
0000036
[root@localhost ~]#
```

Displaying Binary Files

- The **`-x`** option will display the data in hexadecimal

  `od -x /bin/cd`

```
[root@localhost ~]# od -x /bin/cd
0000000 2123 752f 7273 622f 6e69 732f 0a68 7562
0000020 6c69 6974 206e 6463 2220 4024 0a22
0000036
[root@localhost ~]# _
```

Searching for Text in Files

- The most common tool for searching for text in text files is the **`grep`** command

  Syntax: **`grep [options]`** ***`expression file(s)`***

Searching for Text in Files

- The command **`grep hackett /etc/passwd`** would search the /etc/passwd file for any lines that contain the literal text “hackett” and display the lines to the terminal

```
[root@localhost ~]# grep hackett /etc/passwd
mhackett:x:1000:1000:Michael Hackett:/home/mhackett:/bin/bash
[root@localhost ~]# _
```

Searching for Text in Files

- The **grep** command makes use of **regular expressions** (patterns of text) in its searching.
  - **G**lobal **r**egular **e**xpression **p**rint

- Regular expressions (*regexp*) are similar to wildcard metacharacters, except they are interpreted by text tools and not the shell.
  - **Common regular expressions** – available to most text tools
  - **Extended regular expressions** – less common and not available to all text tools

Searching for Text in Files

Regexp	Description	Example	Type
*	Matches 0 or more occurrences of previous character	test* matches tes, test, testt, testtt, testttt, and so on	Common
?	Matches 0 or 1 occurrences of the previous character	test? matches tes, test	Extended
+	Matches 1 or more occurrences of the previous character	test+ matches test, testt, testtt, testttt, and so on	Extended
.	Matches any 1 character	test. matches test1, test2, test3, and so on; testa, testb, testc, and so on	Common
[...]	Matches any 1 character in the range specified	test[2468] matches test2, test4, test6, test8 test[g-j] matches testg, testh, testi, testj	Common

Searching for Text in Files

Regexp	Description	Example	Type
[^…]	Matches any 1 character NOT specified in the range	test[^2468] matches test1, test3, test5, test7, test9, test0, testa, testb, testc, and so on	Common
{ }	Matches a specific number or range of the previous character	test{3} matches testtt test{1,3} matches test, testt, testtt	Extended
^	Matches the following characters if they are the first characters of a line	^test matches any line that starts with test	Common
$	Matches the previous characters if they are the last characters of a line	test$ matches any line the ends with test	Common
(…|…)	Matches either of two sets of characters	(test1|test2) matches either test1 or test2	Common

Searching for Text in Files

- **`grep`**'s **`-s`** option suppresses any warnings.
  - Useful for ignoring grep operations on directory files
- The command **`grep -s hackett /etc/*`** will search through all files and directories in the /etc directory and print any lines from text files that contain the text "hackett"

```
[root@localhost ~]# grep -s hackett /etc/*
/etc/group:mhackett:x:1000:
/etc/gshadow:mhackett:!::
/etc/passwd:mhackett:x:1000:1000:Michael Hackett:/home/mhackett:/bin/bash
/etc/shadow:mhackett:$6$3/RDoshAeMsJFoFm$jaPfUWIrvauViwtFspg1K/IxlsifLKCEGZNGNF
FxStImOkRrisPo6UvMHTLcWONo/::0:99999:7:::
/etc/shadow-:mhackett:$6$3/RDoshAeMsJFoFm$jaPfUWIrvauViwtFspg1K/IxlsifLKCEGZNGN
HFxStImOkRrisPo6UvMHTLcWONo/::0:99999:7:::
/etc/subgid:mhackett:100000:65536
/etc/subuid:mhackett:100000:65536
[root@localhost ~]# _
```

Searching for Text in Files

`grep -s hackett /etc/sub*`

- Any lines that contain the pattern “hackett” in files that start with “sub” in the /etc directory

```
[root@localhost ~]# grep -s hackett /etc/sub*
/etc/subgid:mhackett:100000:65536
/etc/subuid:mhackett:100000:65536
[root@localhost ~]# _
```

Searching for Text in Files

- **`grep`**'s **`-c`** option prints the number of times a pattern appears in a file.
- The command **`grep -c bash /etc/passwd`** will print the total number of lines in the passwd file that contains the pattern bash

```
[root@localhost ~]# grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
mhackett:x:1000:1000:Michael Hackett:/home/mhackett:/bin/bash
[root@localhost ~]# grep -c bash /etc/passwd
2
[root@localhost ~]# _
```

Searching for Text in Files

- **grep**'s **-i** option ignores casing.
- The command **grep -i hackett /etc/passwd** will print the lines in the passwd file that contains the pattern hackett, regardless of casing

```
[root@localhost ~]# grep -i hackett /etc/passwd
mhackett:x:1000:1000:Michael Hackett:/home/mhackett:/bin/bash
[root@localhost ~]# _
```

Searching for Text in Files

- If your pattern contains a space, be sure to put it in double quotes.

```
grep -s "Michael Hackett" /etc/*
```

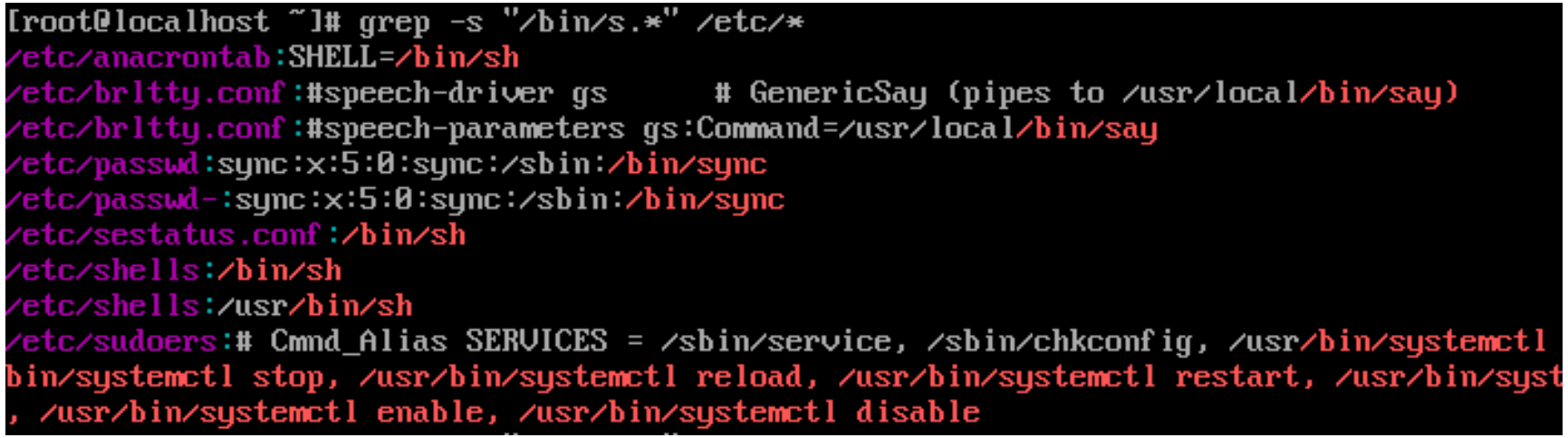
```
[root@localhost ~]# grep -s "Michael Hackett" /etc/*
/etc/passwd:mhackett:x:1000:1000:Michael Hackett:/home/mhackett:/bin/bash
[root@localhost ~]# _
```

Searching for Text in Files

- When searching using a regular expression, quotes are also required.

  `grep -s "/bin/s.*" /etc/*`

  Searches every file in the /etc directory for lines that contain the pattern: **/bin/** followed by 1 **s** and followed by **any** number of characters.



Searching for Text in Files

`grep -s "/bin/s[^y]" /etc/*`

Searches every file in the /etc directory for lines that contain the pattern: **/bin/s** followed by any character that is **not y**.

```
[root@localhost ~]# grep -s "/bin/s[^y]" /etc/*
/etc/anacrontab:SHELL=/bin/sh
/etc/brltty.conf:#speech-driver gs      # GenericSay (pipes to /usr/local/bin/say)
/etc/brltty.conf:#speech-parameters gs:Command=/usr/local/bin/say
/etc/sestatus.conf:/bin/sh
/etc/shells:/bin/sh
/etc/shells:/usr/bin/sh
[root@localhost ~]#
```

Searching for Text in Files

- For extended regular expressions, the **-E** option is required.

  grep -sE "/bin/s?" /etc/shells

  Searches the shells file in the /etc directory for lines that contain the pattern: **/bin/** followed by 0 or more **s's**

```
[root@localhost ~]# grep -sE "/bin/s?" /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
[root@localhost ~]#
```

Searching for Text in Files

- Other common grep options:
  - **`-v`** Output the lines that do NOT match the pattern
  - **`-l`** Only print the files that have matching lines
  - **`-o`** Only print the matching part of the line
  - **`-F`** Ignore regular expression characters in the search pattern

Searching for Text in Files

- Other tools commonly used for searching and manipulating text files are **`sed`** and **`awk`**
  - We may see them in later lectures, but an entire 4-credit course could be devoted just to them

- Both are worth you becoming somewhat proficient with if you plan on a possible career in Unix/Linux systems administration

Editing text files

- The most common CLI text editor on Linux and Unix systems is **vi** (“vee eye”)
  - A similar program, **vim** (**v**i **im**proved), is installed on most Linux systems.
  - Entering the vi command will open vim, if vim is installed.

Editing text files

- The following syntax is used to open a file in vi

  `vi` *`file`*

- If the file does not exist, it will be created
  - Provided the user has permission to create new files in the current directory

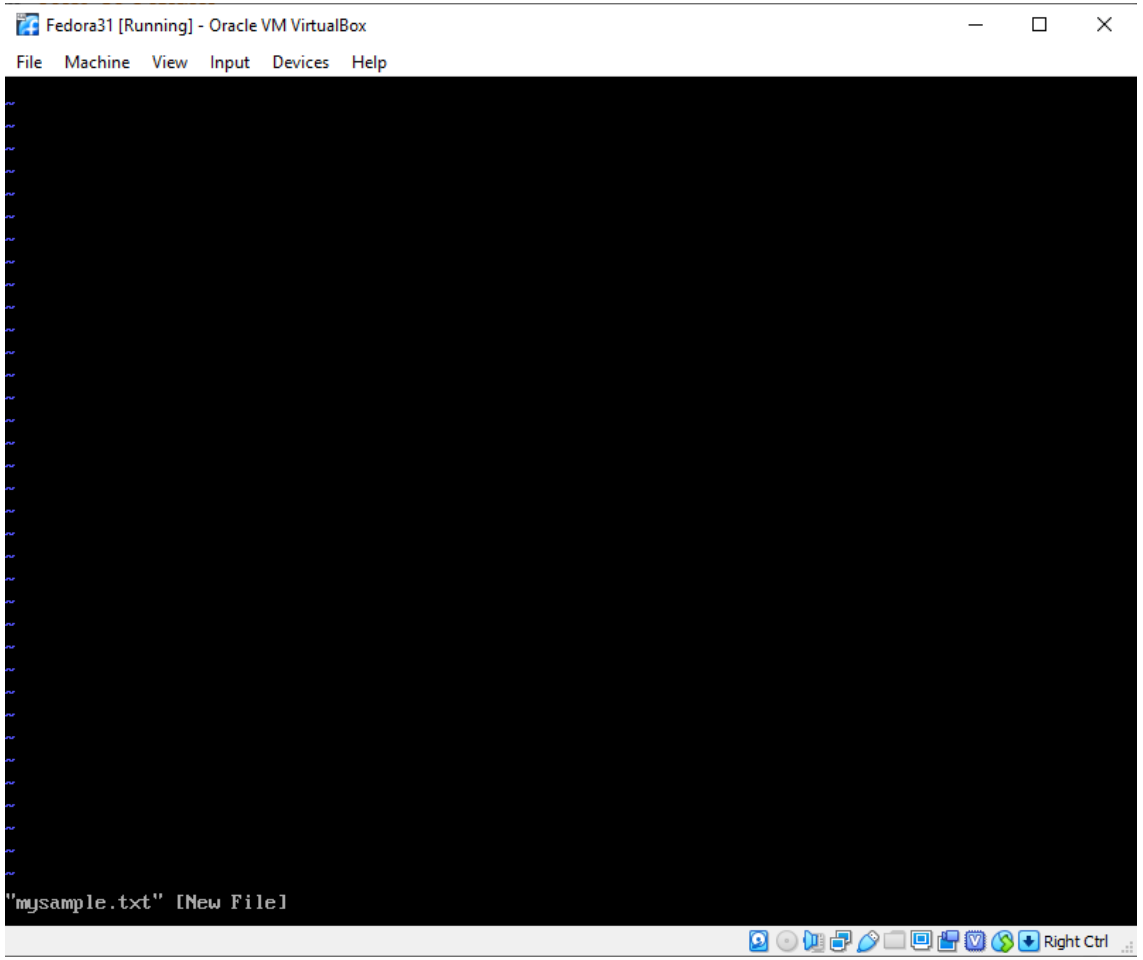
Editing text files

- Entering the command **`vi mysample.txt`** will open this file named mysample.txt (or create it if it doesn't exist)

- This assumes the user is in the same directory as the file.
  - If it is in a different directory, a path must be provided
  - For example, **`vi ~/mysample.txt`** would open/create the file if it was in the user's home directory

Editing text files

```
vi mysample.txt
```



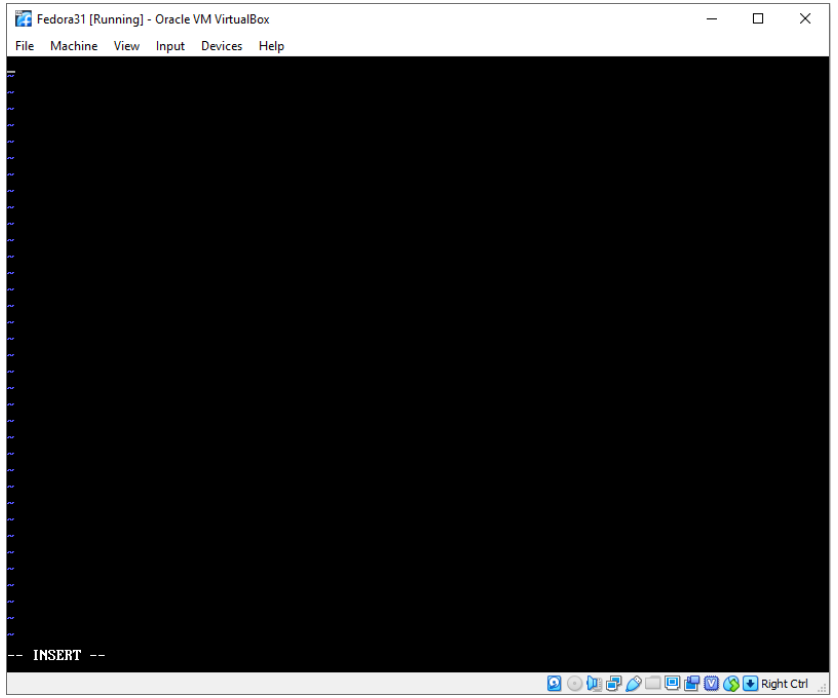
Editing text files

- Vi has four modes:
  - Insert Mode
  - Command Mode
  - Execute Mode
  - Visual Mode

- To enter Command Mode, press the Escape key
  - Vi opens in Command Mode
- To enter Insert Mode, press the i key (while in Command Mode)
- To enter Execute Mode, press the : key (while in Command Mode)
- To enter Visual Mode, press the v or V key (while in Command Mode)

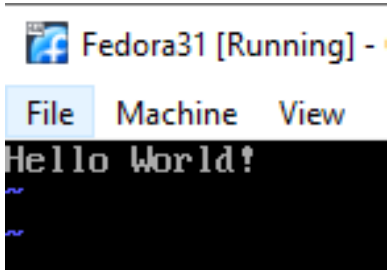
Editing text files

- After pressing the i key, I am put in insert mode.
  - Allowing me to edit the file

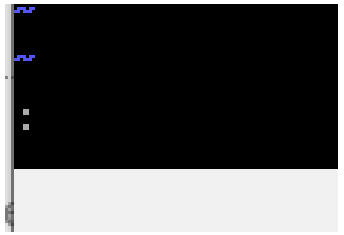


Editing text files

- After making any changes, I can save and exit vi.



- Change to Command Mode (Escape)
  - Press the colon key (Shift+;) to enter Execute Mode



Editing text files

- Common Execute Mode commands (press Enter after each)

:w	Write/save the file
:w *filename*	Writes/saves the file as the specified filename
:q!	Exits vi without saving changes
:wq	Exits vi and saves the file
:set nu	Turns line numbers on
:set nonu	Turns line numbers off
:$	Goes to the last line
:1	Goes to the first line
:*number*	Goes to this line number (:4 would go to the fourth line)

Editing text files

- Common Command Mode Navigation keys

h	Move one character left
l	Move one character right
j	Move one line down
k	Move one line up
^	Move to the beginning of the current line
$	Move to the end of the current line
w	Move to the next word in the line
b	Move to the previous word in the line
gg	Move to the first line of the file
Shift+G	Move to the last line of the file

Editing text files

- Common Command Mode Editing keys

X	Delete the character selected by the cursor
*number*x	Delete this number of characters after (and including) this character
dd	Delete the current line
dnumber	Deletes this number of lines after (and including) the current line
yy	Copies the current line
y*number*	Copies this number of lines after (and including) the current line
p	Pastes copied or deleted line(s) below the current line
P	Pastes copied or deleted line(s) above the current line
u	Undo the last change
U	Undo all changes on the current line
/*text*	Finds occurrences of the specified *text* (n to go forward; N to go backward)
:s/*pattern*/*replacement*/g	Replaces all occurrences of *pattern* with the *replacement* text

Editing text files

- After executing :w

- Then after executing :q, vi closes and I’m placed back at the command prompt

Editing text files

- The command **`cat mysample.txt`** will show the data saved to the file.

```
[mhackett@localhost ~]$ cat mysample.txt
Hello World!
[mhackett@localhost ~]$
```

Editing text files

- Other commonly used text editors
  - (Command Line) GNU nano
  - (Command Line) GNU Emacs (**E**ditor **Mac**ro**s**)
  - (Graphical Text Editor, GNOME desktop) gedit
  - (Graphical Text Editor, KDE desktop) kate

- It’s worth having some exposure to nano and emacs
  - Ultimately, the text editor you use comes down to preference
  - You should absolutely be somewhat proficient with vi, though.