Introduction

- Globs, also called “wildcards”, are special characters used in the shell designed to match filenames.
- Glob characters are used for manipulating (listing, copying, moving, etc.) groups of files.
- Glob characters include:
  - `*` = Match zero or more of any characters
  - `?` = Match exactly one character
  - `[]` = Match a range of characters

Asterisk * Character

- The *asterisk* wildcard* can be used to match any string, including the emp
- To display all files in the current directory:

```
sysadmin@localhost:~$ ls *
Desktop Documents Downloads Music Pictures Public Templates Videos
```

- To display all files in the current directory that begin with the letter `D`:

```
sysadmin@localhost:~$ ls D*
Desktop Documents Downloads
```

- To display all files in the current directory that begin with `D` and have an

```
sysadmin@localhost:~$ ls D*n*
Documents Downloads
```

Question Mark ? Character

- The *question mark ? character* in a string will match exactly one character.
- To display all files in the current directory that have exactly one character in file name:

```
sysadmin@localhost:~$ echo /usr/bin/?
/usr/bin/[ /usr/bin/w
```

- To display all files in the current directory that begin with the letter `D` and have three more characters:

```
sysadmin@localhost:~$ echo D???
Desktop Documents Downloads
```

Brackets [ ] Characters

- With the *square bracket [ ] characters*, a set of characters can be enclosed that will be used to match exactly one character.
- To display all files in the current directory that begin with `a`, `b` or `c`:

```
sysadmin@localhost:/usr/bin$ ls [a-c]
a2p apt awk cal cmp col cut dig
```

- To display all files in the current directory that don't begin with `a`, `b` or `c` use the exclamation `!` or caret `^` characters:

```
sysadmin@localhost:/usr/bin$ ls [!a-c]*
```

Complex Globbing Examples

- To display all files in the current directory that begin with `a`, `b` or `c` and are at least 5 characters long:

```
sysadmin@localhost:/usr/bin$ ls [abc]????*
```

- To display all files in the current directory that begin with don't end with `x`, `y` or `z`:

```
sysadmin@localhost:/usr/bin$ ls* [^xyz]
```

Chapter 05
File Manipulation

NDG

Introduction

- Everything is considered a file in Linux.
- Essential file management commands include:
  - `ls` - list files in a directory
  - `cp` - copy files and directories
  - `mv` - move and/or rename files and directories
  - `rm` - remove files and directories
  - `mkdir` - make new directories
  - `rmdir` - remove empty directories

Listing Files

```
ls [OPTION]... FILE...
```

- By default, the `ls` command will list the files in the current directory.

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads Music  Pictures Public Templates  Videos
```

- Common options for the `ls` command include:
  - `-a` - shows all files, including hidden files
  - `-S` - sorts output by file size
  - `-t` - sorts by time stamp
  - `-r` - reverses the sort
  - `-R` - shows files recursively
  - `-d` - displays directories, not their contents

Listing Files

- For details about a file, such as the type of file, permissions, ownership, or t timestamp, perform a *long listing*, using the `-l` option:

```
sysadmin@localhost:~$ ls -l
total 32
drwxr-xr-x 2 sysadmin sysadmin 4096 Apr 24 16:24 Desktop
...
```

  - Viewing the above output as fields that are separated by spaces, they indicate:
    - File type (`d`)
    - Permissions (`rwxr-xr-x`)
    - Hard link count (`2`)
    - User owner (`sysadmin`)
    - Group owner (`sysadmin`)
    - File size (`4096`)
    - Timestamp (`Apr 24 16:24`)
    - Filename (`Desktop`)

Viewing File Types

- The `file` command displays what type of data the file contains:

```
sysadmin@localhost:~$ file Documents/newhome.txt
Documents/newhome.txt: ASCII text
```

- Why do you need to check the file type?
  - Many Linux commands require data that is text-based, not binary
  - Unknowingly opening or viewing a binary file may cause a terminal window to "hang" forcing a reset

Creating and Modifying Files

```
touch [OPTION]... FILE...
```

- The `touch` command performs one of two actions:
  - Creates a new file if a file name argument doesn't exist.
  - Updates the timestamp of an existing file.
    - No options = Updates timestamp
    - `-a` = Updates access timestamp
    - `-c` = Updates attribute timestamp
    - `-t` = Used to specify timestamp value (if you don't want to use default current time
- To create a file with the `touch` command, you must have the *write* and *execute* permission on the parent directory.

Copying Files

```
cp [OPTION]... SOURCE DESTINATION
```

- The `cp` command will copy a file.

```
sysadmin@localhost:~$ cp /var/old_file.txt ~/newfile.txt
```

- To copy multiple files:

```
sysadmin@localhost:~$ cp /data/* ~
```

- To verify files are copied, use the `-v` option for verbose output.

```
sysadmin@localhost:~$ cp -v /etc/b* ~
'/etc/bash.bashrc' -> '/home/sysadmin/bash.bashrc'
```

- To copy a directory, use the `-r` option for recursive functionality.

```
sysadmin@localhost:~$ cp -R /etc/perl ~
'/etc/perl' -> '/home/sysadmin/perl'
```

Moving Files and Directories

```
mv [OPTION]... SOURCE DESTINATION
```

- The `mv` command is used to *move* a file from one path name in the filesystem to another.

```
sysadmin@localhost:~$ mv old_file.txt newfile.txt
```

- To move multiple files:

```
sysadmin@localhost:~$ mv /data/* ~
```

- Moving a file from one directory to another, without specifying a new name for the file, will cause the file to retain its original name.
- When a directory is moved, everything it contains is automatically moved as well.

Deleting Files

```
rm [OPTION]... FILE...
```

- Without any options, the `rm` command is typically used to remove regular files:

```
sysadmin@localhost:~$ rm file.tx
sysadmin@localhost:~$ ls file.txt
ls: cannot access file.txt: No such file or directory
```

- To avoid accidentally deleting files when using globbing characters, use the `-i` option.

```
sysadmin@localhost:~$ rm -i a*
rm: remove regular file 'adjectives.txt'? y
```

- Deleting a file with the `rm` command requires the *write* and *execute* permissions on its parent directory.

Creating Directories

```
mkdir [OPTION]... FILE...
```

- The `mkdir` command allows you to create (make) a directory.

```
sysadmin@localhost:~$ mkdir one two three
sysadmin@localhost:~$ ls
Desktop   Downloads Pictures Templates two
Documents Music     Public   Videos    one three
```

- Use the `-p` option to automatically create the *parent* directories.

```
sysadmin@localhost:~$ mkdir -p /home/sysadmin/red/blue/yellow/green
```


Removing Directories

```
rmdir [OPTION]... FILE...
```

- The `rmdir` command is used to *remove* empty directories:
- Using the `-p` option will remove directory paths, but only if all of the directories contain other empty directories.
- To remove directories that contain files, use the `rm` command with a recursive `-r` option.

```
sysadmin@localhost:~$ rmdir test_directory
rmdir: failed to remove 'test_directory': Directory not empty
sysadmin@localhost:~$ rm -r test_directory
```

Chapter 10
Standard Text Streams and Redirection

Introduction

- Commands that read in text as input, alter that text in some way, and then produce text as output are sometimes known as filters.
- In order to be able to apply filter commands and work with text streams, it is helpful to understand a few forms of *redirection* that can be used with most commands:
  - Pipelines
  - Standard output redirection
  - Error output redirection
  - Standard input redirection

Standard Output

- When a command executes without any errors, the output that is produced is known as *standard out*, also called `stdout` or `STDOUT`.
- It is possible to redirect standard out from a command so it goes to a file instead of the terminal.
- Redirection is achieved by following a command with the greater-than > character and a destination file.

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads Music  Pictures Public Templates  Videos
sysadmin@localhost:~$ ls ~ > /tmp/home.txt
sysadmin@localhost:~$ cat /tmp/home.txt
Desktop
Documents
Downloads
Music
...
```


Standard Output

- Redirecting output using a single greater-than > character will create a new file, or *overwrite* the contents of an existing file.
- When using the >> characters, the output of the command will be *appended* to the end of a file if it does already exist.
- The number associated with the standard output *file descriptor* (the > character) is the number 1 (one) but it can be omitted:

`COMMAND > FILE` `COMMAND 1> FILE`	Create or overwrite `FILE` with the standard output of `COMMAND`
`COMMAND >> FILE` `COMMAND 1>> FILE`	Create or append to `FILE` with the standard output of `COMMAND`

Standard Error

- When a command encounters an error, it will produce output that is known as *standard error*, also called stderr or STDERR.
- The number associated with the standard error file descriptor is 2 (two).
- To redirect error messages:

```
sysadmin@localhost:~$ ls /junk
ls: cannot access /junk: No such file or directory
sysadmin@localhost:~$ ls /junk 2> /tmp/ls.err
```

- Use the double >> characters after the number 2 to append :

```
sysadmin@localhost:~$ ls /junk 2>> /tmp/ls.err
```


Standard Error 2

- If you want standard error and standard out sent to one file there are two techniques to redirect both:

`COMMAND &> FILECOMMAND > FILE 2>&1`	Create or overwrite `FILE` with all output (stdout, stderr) of `COMMAND`
`COMMAND &>> FILECOMMAND >> FILE 2>&1`	Create or append to `FILE` with all output (stdout, stderr) of `COMMAND`

Command Pipelines

- Command pipelines are often used to make effective use of filter commands.
- In a command pipeline, the output of one command is sent to another command as input.

```
sysadmin@localhost:~$ history | less
```

- Another example is to take the output of the `history` command and filter the output by using the `grep` command.

```
sysadmin@localhost:~$ history | grep "ls"
1 ls ~ > /tmp/home.txt
5 ls 1> /tmp/ls.txt
6 ls 1> /tmp/ls.txt
```