Module 10
Working With Text

Exam Objective

3.2 Searching and Extracting Data from Files

Objective Description

Search and extract data from files in the home directory

Text Files

Text Files

- A large number of the files in a typical file system are *text files*.
- Text files only contain text, no formatting features that you might see in a word processing file.
- A significant number of commands exist to help users manipulate text files.
- There are also features available for the shell to control the output of commands.
  - Instead of the output displaying in the terminal window, the output can be *redirected* into another file or another command.

Viewing Files in the Terminal

- The `cat` command is a useful command that can be used to create and display text files, as well as combining copies of text files.

- To display a file using the `cat` command type the command followed by the filename:

```
sysadmin@localhost:~/Documents$ cat food.txt
Food is good.
```

- The default output of the `cat` command is the terminal

- The `cat` command can also be used to redirect file content to other files or input to another command by using redirection characters.

Viewing Files Using a Pager

- The `cat` command is good for viewing small files but is not ideal for large files.
- To view larger files, use a *pager* command.
- There are two commonly used pager commands:
    - The `less` command provides advanced paging capability but is not included with all Linux distributions.
    - The `more` command has fewer features than `less`, but is available in Linux distributions.
- The `more` and `less` commands allow users to move around a document using *keystroke commands*.

Pager Movement Commands

- To view a file with the `less` command, pass the filename as an argument:

```
sysadmin@localhost:~/Documents$ less words
```

- When viewing a file with the `less` command, use the **H** key or **Shift + H** to display a help screen
- Below are the most commonly used movement commands (identical in `more` and `less`):

Key	Movement
Spacebar	Window forward
B	Window backward
Enter	Line forward
Q	Exit
H	Help

Pager Searching Commands

- There are two ways to search in the `less` command:
  - Searching forward or backward from your current position.
  - To start a search forward from your current position, use the slash / key then type the text to match and press **Enter**. For example:

```
/frog
```

```
bullfrog
bullfrog's
bullfrogs
bullheaded
```

  - If no matches forward from your current position can be found, then the last line of the screen will report "`Pattern not found`":
  - If more than one match can be found by a search, then use the **N** key to move the *next* match and use the **Shift**+**N** key combination to go to a *previous* match.

Head and Tail

- The `head` and `tail` commands are used to display only the first or last few lines of a file.
- Passing a number as an option will cause both the `head` and `tail` commands to output the specified number of lines instead of default ten.
- The `-n` option can be used to indicate how many lines to output.
- The negative value option (i.e., `-3`) means show -# lines
- With the positive value option, if the `-n` option is used with a number prefixed by the plus sign, then the `tail` command recognizes this to mean to display the contents starting at the specified line and continuing all the way to the end.

Command Line Pipes

Command Line Pipes

- The *pipe* | character can be used to send the output of one command to another.
- The pipe character allows you to utilize the `head` and `tail` commands not only on files, but on the output of other commands.
- For example, some commands have large amount of output. To more easily view the beginning of the output, pipe it to the `head` command. The following example displays only the first ten lines:

```
sysadmin@localhost:~$ ls /etc | head
```

- Multiple pipes can be used consecutively to link multiple commands together. Each command only sees input from the previous command.

Input/Output Redirection

Input/Output Redirection

- *Input/Output (I/O) redirection* allows for command line information to be passed to different *streams*.
- The streams are:
  - STDIN: *Standard input*, or STDIN, is information entered normally by the user via the keyboard.
  - STDOUT: *Standard output*, or STDOUT, is the normal output of commands.
  - STDERR: Standard error, or STDERR, are error messages generated by commands.
- I/O redirection allows the user to redirect STDIN so that data comes from a file and STDOUT/STDERR so that output goes to a file.
- Redirection is achieved by using the arrow < > characters.

STDOUT

- The standard output of a command will display to the screen:

```
sysadmin@localhost:~$ echo "Line 1"
Line 1
```

- Using the > character, the standard output of a command can be redirected to a file instead:

```
sysadmin@localhost:~$ echo "Line 1" > example.txt
```

- The `example.txt` file contains the output of the `echo` command, which can be viewed with the `cat` command:

```
sysadmin@localhost:~$ cat example.txt
Line 1
```

STDOUT

- The single arrow overwrites any contents of an existing file.
- It is possible to preserve the contents of an existing file using the >> characters, which append to a file instead of overwriting it.

```
sysadmin@localhost:~$ echo "New line 1" > example.txt
sysadmin@localhost:~$ cat example.txt
Line 1
sysadmin@localhost:~$ echo "Another line" >> example.txt
sysadmin@localhost:~$ cat example.txt
New line 1
Another line
```

STDERR

- STDERR can be redirected similarly to STDOUT.
- Using the `>` character to redirect, stream #1 is assumed by default.
- Stream #2 must be specified when redirecting STDERR by placing the number `2` before the arrow `>` character.
- In the example, the `2>` indicates that all error messages should be sent to the file `error.txt`:

```
sysadmin@localhost:~$ ls /fake
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$ ls /fake 2> error.txt
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
```

Redirecting Multiple Streams

- It is possible to direct both the STDOUT and STDERR of a command at the same time
- Both STDOUT and STDERR can be sent to a file by using the ampersand `&` character in front of the arrow `>` character. The `&>` character set that means both `1>` and `2>`:

```
sysadmin@localhost:~$ ls /fake /etc/ppp
ls: cannot access /fake: No such file or directory
/etc/ppp:
ip-down.d  ip-up.d
sysadmin@localhost:~$ ls /fake /etc/ppp &> all.txt
sysadmin@localhost:~$ cat all.txt
ls: cannot access /fake: No such file or directory
/etc/ppp:
ip-down.d
ip-up.d
```

- If you don't want STDERR and STDOUT to both go to the same file, they can be redirected to different files by using both `>` and `2>`.

```
sysadmin@localhost:~$ ls /fake /etc/ppp > example.txt 2> error.txt
```

STDIN

- With most commands, if you want to read data from a file into a command, you can specify the filename as an argument to the command.
- However, for some commands, if you don't specify a filename as an argument, they revert to using STDIN to get data (i.e., the `cat` and `tr` commands)
- Using STDIN, it is possible to tell the shell to get STDIN from a file instead of from the keyboard by using the < character:

```
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
ip-down.d
ip-up.d
sysadmin@localhost:~$ tr 'a-z' 'A-Z' < example.txt
/ETC/PPP:
IP-DOWN.D
IP-UP.D
```

Sorting Files or Input

Sorting Files for Input

- The `sort` command can be used to rearrange the lines of files or input in either dictionary or numeric order.

```
sysadmin@localhost:~$ cat mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
sysadmin@localhost:~$ sort mypasswd
bin:x:2:2:bin:/bin:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/bin/sh
```

Fields and Sort Options

- The `sort` command can rearrange the output based on the contents of one or more fields.
- Fields are determined by a *field delimiter* contained on each line.

```
bin:x:2:2:bin:/bbin:x:2:2:bin:/bin:/bin/sh
```

- For example, the following command can be used to sort the third field of the `mypasswd` file numerically.
  - The `-t` option will allow for another field separator to be specified
  - To specify which field to `sort` by, use the `-k` option with an argument to indicate the field number
  - The `-n` option is used to perform a numeric sort.

```
sysadmin@localhost:~$ sort -t: -n -k3 mypasswd
```

Viewing File Statistics

Viewing File Statistics

- The `wc` command provides the number of lines, words and bytes (1 byte = 1 character in a text file) for a file, and a total line count if more than one file is specified.

```
sysadmin@localhost:~$ wc /etc/passwd /etc/passwd-
  35   56 1710 /etc/passwd
  34   55 1665 /etc/passwd-
  69  111 3375 total
```

- It is also possible to view only specific statistics:
  - Use the `-l` option to show just the number of lines
  - The `-w` option to show just the number of words
  - The `-c` option to show just the number of bytes, or any combination of these options

Filtering Files

Filter File Sections

- The `cut` command can extract columns of text from a file or standard input.
- The `cut` command is used for working with delimited files—which contain columns separated by a delimiter.
    - By default, the `cut` command expects its input to be separated by the **Tab** character, but the `-d` option can specify alternative delimiters such as the colon or comma.
    - The `-f` option can specify which fields to display.
    - The `-c` option is used to extract columns of text based upon character position.
    - In the following example, the first, fifth, sixth and seventh fields from `mypasswd` database file are displayed:

```
sysadmin@localhost:~$ cut -d: -f1,5-7 mypasswd
```

Filter File Contents

- The `grep` command can be used to filter lines in a file or the output of another command that matches a specified pattern:

```
sysadmin@localhost:~$ grep --color bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin:/bin/bash
```

- The `grep` command can be used with several options to filter lines:
  - The `-d` option can specify alternative delimiters such as the colon or comma
    - by default, the `cut` command expects its input to be separated by the **Tab** character
  - The `-f` option can specify which fields to display.
  - The `-c` option is used to extract columns of text based upon character position.

Basic Regular Expressions

Basic Regular Expressions

- *Regular expressions*, also referred to as *regex*, are a collection of *normal* and *special* characters that are used to find simple or complex patterns, respectively, in files.

- *Normal characters* are alphanumeric characters which match themselves.
    - For example, an `a` character would match an `a`

- *Special characters* have special meanings when used within patterns by commands like the `grep` command.

- There are both *Basic Regular Expressions* (available to a wide variety of Linux commands) and *Extended Regular Expressions* (available to more advanced Linux commands).

Basic Regular Expressions

- Basic Regular Expressions include the following:

Character	Matches
.	Any single character
[ ]	A list or range of characters to match one character. If the first character is the caret ^, it means any character *not* in the list
*	The previous character repeated zero or more times
^	If the first character in the pattern, the pattern must be at the beginning of the line to match, otherwise just a literal ^
$	If the last character in the pattern, the pattern must be at the end of the line to match, otherwise just a literal $

The `grep` command is just one of the many commands that support regular expressions. Some other commands include the `more` and `less` commands.

Basic Regular Expressions- The . Character

- The . character matches any character except for the new line character.
- The pattern `r..f` would find any line that contained the letter `r` followed by exactly two characters and then the letter `f`:

```
sysadmin@localhost:~/Documents$ grep 'r..f' red.txt
reef
roof
```

- The . character can be used any number of times. To find all words that have at least four characters, the following pattern can be used:

```
sysadmin@localhost:~/Documents$ grep '....' red.txt
reef
reeed
roof
```

Basic Regular Expressions- The [ ] Character

- In some cases, you want to specify exactly which characters you want to match, such as a lower-case alphabet character or a number character.
- The square brackets `[]` match a single character from the list or range of possible characters contained within the brackets:

```
sysadmin@localhost:~/Documents$ grep '[0-9]' profile.txt
I am 37 years old.
3121991
I have 2 dogs.
123456789101112
```

- Each possible character can be listed out `[abcd]` or provided as a range `[a-d]`, as long as the range is in the correct order.
- To match a character that is *not* one of the listed characters, start the `[]` set with a `^` symbol.

Basic Regular Expressions- The * Character

- The asterisk `*` character is used to match zero or more occurrences of a character or pattern preceding it.

- For example, `e*` would match zero or more occurrences of the letter `e`:

```
sysadmin@localhost:~/Documents$ grep 're*d' red.txt
red
reeed
rd
```

- It is also possible to match zero or more occurrences of a list of characters by utilizing square brackets. the following example matches zero or more occurrences of the `o` character or the `e` character:

```
sysadmin@localhost:~/Documents$ grep 'r[oe]*d' red.txt
```

- To make the asterisk `*` character useful, create a pattern which includes more than just the one character preceding it (i.e., `ee*`).

Basic Regular Expressions- Anchor Characters

- When performing a pattern match, the match could occur anywhere on the line.
- *Anchor characters* specify whether the match occurs at the beginning of the line or the end of the line.
- The circumflex `^` character is used to ensure that a pattern appears at the beginning of the line:

```
sysadmin@localhost:~/Documents$ grep '^root' passwd
root:x:0:0:root:/root:/bin/bash
```

- The second anchor character, `$`, can be used to ensure a pattern appears at the end of the line:

```
sysadmin@localhost:~/Documents$ grep 'r$' alpha-first.txt
B is for Bear
F is for Flower
```

Basic Regular Expressions- The \ Character

- Use the backslash character \ to match a character that happens to be a special Regular Expression character:

- In the example below, the pattern re* matches an `r` (followed by zero or more of the letter `e`):

```
sysadmin@localhost:~/Documents$ grep 're*' newhome.txt
**Beware** of the ghost in the bedroom.
```

- To look for an actual asterisk `*` character, place a backslash `\` character before the `*` character:

```
sysadmin@localhost:~/Documents$ grep 're\*' newhome.txt
**Beware** of the ghost in the bedroom.
```

Extended Regular Expressions

- The use of extended regular expressions often requires a special option be provided to the command to recognize them.
    - The `-E` option to the `grep` command can understand extended regular expressions.
- The following regular expressions are considered extended:

Character	Matches
?	Matches previous character zero or one time, so it is an optional character
+	Matches previous character repeated one or more times
|	Alternation or like a logical or operator

Extended Regular Expressions

- Use the `?` character to match `colo` followed by zero or one `u` character followed by an `r` character:

```
sysadmin@localhost:~/Documents$ grep -E 'colou?r' spelling.txt
American English: Do you consider gray to be a color or a shade?
British English: Do you consider grey to be a colour or a shade?
```

- Use the `+` character to match one or more `e` characters:

```
sysadmin@localhost:~/Documents$ grep -E 'e+' red.txt
red
reef
reeed
```

- Use the `|` character to match either `gray` or `grey`:

```
sysadmin@localhost:~/Documents$ grep -E 'gray|grey' spelling.txt
American English: Do you consider gray to be a color or a shade?
British English: Do you consider grey to be a colour or a shade?
```