Linux utilities

Unit objectives

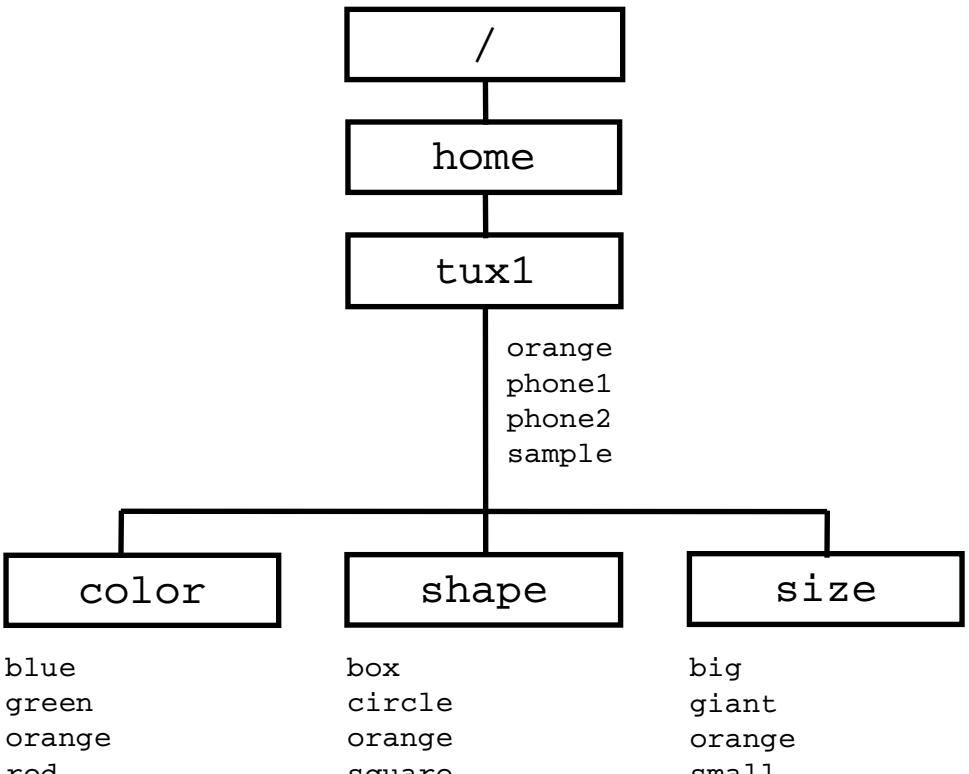
After completing this unit, you should be able to:

- Use the **find** and **locate** commands to search for files
- Use the **grep** command to search text files for patterns
- Use the **cut** command to list specific columns of a file
- Use the **sort** command to sort the contents of a file
- Use the **head** and **tail** commands to view specific lines in a file
- Use the **type**, **which**, and **whereis** commands to find commands
- Use the **file** command to find out the content of a file
- Use the **join** and **paste** commands to combine files
- Compress and uncompress files with **gzip**, **gunzip**, **zcat**, **bzip2**, **bunzip2**, and **bzcat**

The find command

- With the **find** command, you can search one or more directory structures for files that meet specified criteria.
- You can display the names of matching files or execute commands against those files.
- Syntax:
  - `$ find path expression`

Sample directory structure



Using find

- Generally, you want to search a directory structure for files with certain names and list the names found.

```
$ cd /home/tux1
$ find . -name orange
./orange
./color/orange
./shape/orange
./size/orange
```

Executing commands with find

- The -exec option executes a command on each of the file names found.

```
$ find . -name 's*' -exec ls -i {} \;
187787    ./sample
187792    ./shape/square
202083    ./size/small
```

- A set of curly brackets ({ }) is a placeholder for each file name. The backslash (\) escapes the following semicolon (;).

Interactive command execution

- The -ok option also causes command execution but on an interactive basis.

```
$ find  .  -name  o\*  -ok  rm {}  \;
< rm ... ./orange > ? y
< rm ... ./color/orange > ? y
< rm ... ./shape/orange > ? n
< rm ... ./size/orange > ? y
```

Additional find options

`-type`	`f` `d`	Ordinary file Directory
`-size`	`+`*`n` `-`*`n`* *`n`*`c`	Larger than *n* blocks Smaller than *n* blocks Equal to *n* characters
`-mtime`	`+`*`n` `-`*`n`* *`n`*	Modified more than *n* days ago Modified less than *n* days ago Modified *n* days ago
`-perm`	*`onum`* *`mode`*	Access permissions match *onum* Access permissions match *mode*
`-user`	*`user`*	Finds files owned by *user*
`-newer`	*`ref.file`*	File was modified more recently than *ref.file*
`-o` `-a`		Logical OR Logical AND

find examples


```
$ find . -perm 777
./size/giant

File matches expr 1 and expr 2:
$ find . -name 's*' -type f -a -size +2\
>-exec ls -i {} \;
187787    ./sample
187792    ./shape/square
202083    ./size/small

File matches expr 1 or expr 2:
$ find . -name circle -o -name 'b*'
./color/blue
./size/big
./shape/box
./shape/circle
```

locate command

- **locate** allows you to quickly find a file on the system based on simple criteria.

```
$ locate passwd
/usr/share/man/man1/passwd.1.gz
/usr/share/man/man5/passwd.5.gz
/etc/passwd
/usr/bin/passwd
```

- This requires that the superuser runs updatedb regularly.
  - Most distributions run updatedb automatically.
  - SuSE does not install locate/updatedb by default.

The grep command

- Searches one or more files or standard input for lines matching a pattern
- Simple match or regular expression
- Syntax
  - `grep [options] pattern [file1 ...]`

grep sample data files

- Phone 1:

Chris	10300	internal
Jan	20500	internal
Lee	30500	external
Pat	40599	external
Robin	50599	external
Terry	60300	internal

- Phone 2:

Chris	1342	internal
Jan	2083	internal
Lee	3139	external
Pat	4200	internal
Robin	5200	internal
Terry	6342	external

Basic grep


```
$ grep 20 phone1
Jan       20500          internal

$ grep 20 phone*
phone1:Jan     20500          internal
phone2:Jan     2083       internal
phone2:Pat     4200       internal
phone2:Robin   5200       internal
$ grep -v Jan phone2
Chris          1342       internal
Lee       3139       external
Pat       4200       internal
Robin          5200       internal
Terry          6342       external
```

grep with regular expressions

- Patterns with metacharacters should be in single quotation marks (') so that the shell will leave it alone.
- Valid metacharacters with grep include $. * ^ [-].
  - **.** Any single character
  - ***** Zero or more occurrences of the preceding character
  - **[a-f]** Any *one* of the characters in the range *a* through *f*
  - **^a** Any line that starts with *a*
  - **z$** Any line that ends with *z*

Regular expression

1. Display all processes that belong to user tux1:
   $ _____

2. Display all lines of the phone1 file (blank and non-blank):
   $ grep _____ phone1

3. Display only the lines of phone1 that contain an *e* and end in a *0*:
   $ grep _____ phone1

grep options

- **-v** Print lines that do not match
- **-c** Print only a count of matching lines
- **-l** Print only the names of files with matching lines
- **-n** Number the matching lines
- **-i** Ignore case of letters when making comparisons
- **-w** Do a whole word search
- **-f <file>** Read expressions from file instead command line

Other greps

- **fgrep** allows only fixed strings (no regular expressions).
- **egrep** allows for multiple (alternate) patterns.

```
$ egrep '20500|40599|50599' phone1
Jan       20500     internal
Pat       40599     external
Robin     50599     external
```

- What does the following command do?

```
$ grep 30 phone1 | grep intern
????????
```

The cut command

- Pull selected columns or fields from one or more files
- Syntax:

```
cut -f(ields) -d(elimiter) file(s)
cut -c(haracters) file(s)
```

cut example (1 of 2)


```
$ cat /etc/passwd
root:x:0:0:Big Brother:/root:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
tux1:x:500:500:Tux the Penguin(1):/home/tux1:/bin/bash
tux2:x:501:501:Tux the Penguin(2):/home/tux2:/bin/bash

$ cut -f1,3,6,7 -d: /etc/passwd
root:0:/root:/bin/bash
shutdown:6:/sbin:/sbin/shutdown
tux1:500:/home/tux1:/bin/bash
tux2:501:/home/tux2:/bin/bash
```

cut example (2 of 2)


```
$ ps
  PID TTY STAT TIME COMMAND
 9374 p0  S    0:00 -bash
14460 p0  R    0:00 ps

$ ps | cut -c-5,20-
  PID COMMAND
 9374 -bash
14471 ps
```

The sort command

- The **sort** command sorts the lines in the file specified and writes the result to standard output.

```
sort -tdelimiter -kfield -options file
```

```
$ cat animals
dog.2
cat.4
penguin.10

$ sort animals
cat.4
dog.2
penguin.10
```

sort examples


```
$ sort -k1.2 animals
cat.4
penguin.10
dog.2
$ sort -t. -k2 animals
penguin.10
dog.2
cat.4
$ sort -t. -n -k2 animals
dog.2
cat.4
penguin.10
```

- Options
  - **-d** Sorts in dictionary order (only letters, digits, spaces considered in comparisons)
  - **-r** Reverses the order of the specified sort
  - **-n** Sorts numeric fields in arithmetic value

The head and tail commands

- The **head** command can be used to view the first few lines of a file or files. The command syntax is:

```
$ head  [-lines] file(s)
```

```
$ head -5 myfile
$ ls -l | head -12
```

- The **tail** command displays the last few lines of a file or files. The command syntax is:

```
$ tail [{-lines|-n lines|-n +lines|-f}] file(s)
```

```
$ tail -20 file
$ tail -n +20 file
$ tail -f file
```

The type, which, and whereis commands

- To find out what the path to a command is, use **type**.

```
$ type find echo
find is /usr/bin/find
echo is a shell builtin
```

- To find out where the binary is located, use **which**.

```
$ which find echo
/usr/bin/find
/bin/echo
```

- To locate the binary, source, and manual page files of a command, use whereis.

```
$ whereis find echo
find: /usr/bin/find /usr/share/man/man1/find.1.gz
echo: /bin/echo /usr/share/man/man1/echo.1.gz
```

The file command

- With the **file** command, you can find out what the type of data is in the file.

```
$ file /etc/passwd /bin/ls /home/tux1 /tmp/fake.jpg
/etc/passwd:   ASCII text
/bin/ls:       ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), dynamically linked (uses shared libs
), for GNU/Linux 2.6.15, stripped
/home/tux1:     directory
/tmp/fake.jpg: PDF document, version 1.5
```

The join and paste commands

- The **join** and **paste** commands combine files.

```
$ cat one
a apple another
b bee beast
c cat
$ cat two
a ape
b broken
d dog
$ join one two
a apple another ape
b bee beast broken
$ paste one two
a apple another a ape
b bee beast     b broken
c cat   d dog
```

Compressing and uncompressing Files

- **gzip**, **gunzip**, **zcat**; **bzip2**, **bunzip2**, **bzcat**

```
$ ls -l file1
-rw-r--r-- [...] 34833 2009-05-13 12:33 file1
$ gzip -v file1
file1:     69.6% -- replaced with file1.gz
$ ls -l file1.gz
-rw-r--r-- [...] 10615 2009-05-13 12:33 file1.gz
$ zcat file1
(original contents of file1 displayed)
$ gunzip file1.gz
$ ls -l file1
-rw-r--r-- [...] 34833 2009-05-13 12:33 file1
$ bzip2 file1
$ ls -l file1.bz2
-rw-r--r-- [...] 10335 2009-05-13 12:33 file1.bz2
$ bzcat file1.bz2 | wc -c
34833
```

Unit review

- The following commands were considered:
  - The **find** command is used to recursively search directories for files with particular characteristics.
  - The **grep** command is used to select entire lines containing a particular pattern.
  - The **head** and **tail** commands are used to view specific lines in a file.
  - The **sort** command sorts the contents of a file by the options specified.
  - Find out where you can find commands with **type**, **where**, and **whereis**.
  - The **gzip**, **gunzip**, **zcat**, **bzip2**, **bunzip2**, and **bzcat** commands can be used to create and work with compressed files.

Checkpoint

1. True or False: The command `ps -aux | grep tux | grep firefox` lists all Firefox processes of a user named tux.

2. Which command would best be used to locate all files in your system that begin with the string *team*?
   a. `find / -name "^team"`
   b. `find / -name "team*"`
   c. `find / -name "*team*"`
   d. `find / -type f -name "team"`

3. Translate the following command into your native language:

```
ls -lR|egrep "txt$|tab$"|sort -rn -k5|tail -n
+4|head -5
```

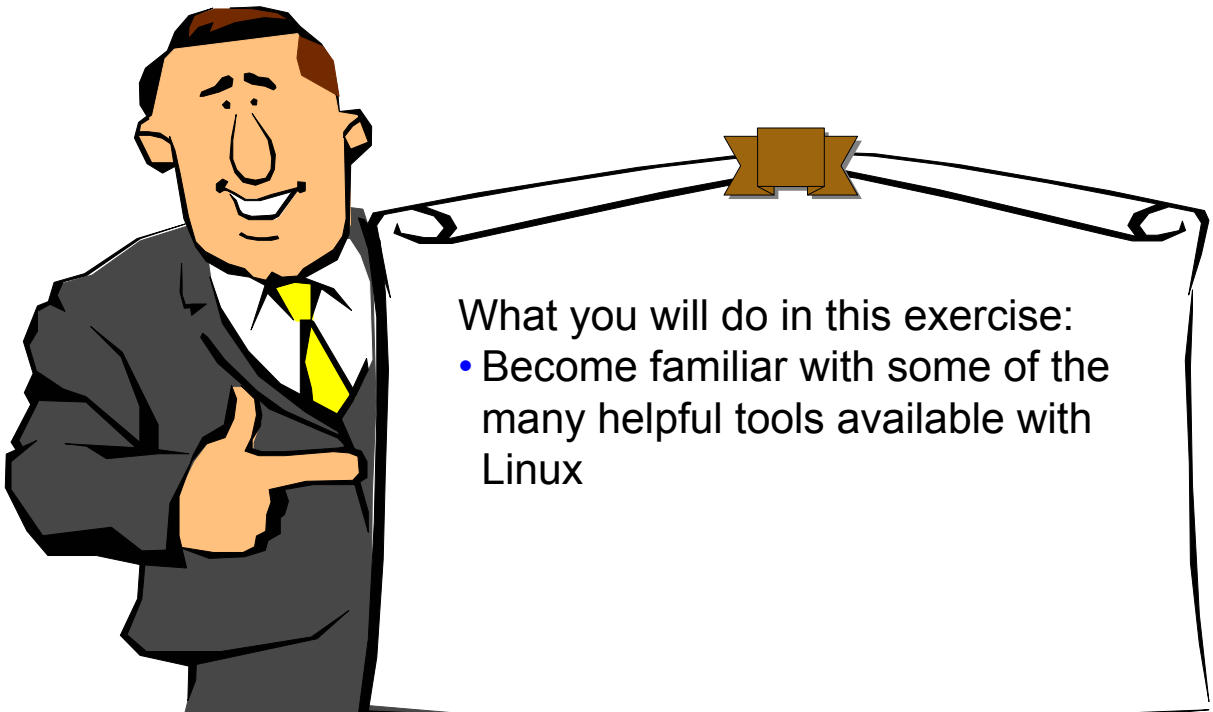
Checkpoint solutions

1. True or False: The command `ps -aux | grep tux | grep firefox` lists all Firefox processes of a user named tux.
   The answer is true. Technically, the command lists all processes for which the **ps** command output includes the *tux* and *firefox* strings. Note that this command will also list all processes of a user named tux01, for example, and all processes named tux or tuxedo, for example. Note also that *all Firefox processes* is nebulous in that a Firefox application might include a process called plug-in container that would not appear in the command output.

2. Which command would best be used to locate all files in your system that begin with the string *team*?
   a. `find / -name "^team"`
   b. `find / -name "team*"`
   c. `find / -name "*team*"`
   d. `find / -type f -name "team"`
   The answer is `find / -name "team*"`. Note that the **find** command uses the same metacharacters as the shell.

3. Translate the following command into your native language:
   `ls -lR|egrep "txt$|tab$"|sort -rn -k5|tail -n +4|head -5`
   The answer is find the fourth through eighth largest files in the current directory tree whose file names end with either *txt* or *tab*.

Exercise: Linux utilities



What you will do in this exercise:

- Become familiar with some of the many helpful tools available with Linux

Unit summary

Having completed this unit, you should be able to:

- Use the **find** and **locate** commands to search for files
- Use the **grep** command to search text files for patterns
- Use the **cut** command to list specific columns of a file
- Use the **sort** command to sort the contents of a file
- Use the **head** and **tail** commands to view specific lines in a file
- Use the **type**, **which**, and **whereis** commands to find commands
- Use the **file** command to find out the content of a file
- Use the **join** and **paste** commands to combine files
- Compress and uncompress files with **gzip**, **gunzip**, **zcat**, **bzip2**, **bunzip2**, and **bzcat**