Lecture: File management

Finding files

1) **locate** command

   - The **locate** command searches a database of filenames
   - The database is updated automatically on a daily basis
   - Manual database update can be performed using **updatedb** command

Case:

- Try to locate the *fstab* file in your system. Try also to locate *Fstab*; in this case you should find nothing.

```
$locate fstab
```

- The -i option makes the search case-insensitive; try again to search for *Fstab* file

```
$locate –i Fstab
```

- Create a new file and name it *file1*; try to find the newly created *file1* using **locate**
  - You should find nothing, because the locate database was not updated
  - Run **updatedb** then search again

2) **find** command

- using this command you can find files by any combination of a wide number of criteria, including name, size, type and so on; locate can only finds files by name
- Syntax: find <location> <search criteria> <file name>

Example:

- Go to */usr*, and type:

```
$find . -name manual.html
```

  - It will search by name for *manual.html* file under current directory (using .)
  - This identical with:

```
$find /usr -name manual.html
```

- Other search criteria:
  - -type d: will search for directories
  - -type f: will search just for files

- -iname: will search with case insensitive

Exercise

1. Use **locate** to find files whose name contains the string 'bashbug'. Try the same search with **find**, looking over all files on the system.
   - Why you can't find anything with **find?**
   - What can you do in order to have same result as searching with **locate?**

2. Check by your self this two commands: **apropos**, **whatis**, **which**, **type** and use them in a couple of examples.

3.
   a. Find out whether the **ls** command runs a program directly, or is a shell alias or function.
   b. Locate the binary of the **traceroute** program.
   c. Use **whatis** to find out what the watch command does.
   d. Use **apropos** to find programs for editing the partition table of disks.
   e. See if the Linux installation you are using has an *updatedb.conf*, and look at the current configuration.

Editing files

1) Sed

- In general, sed operates on a stream of text that it reads from either standard input or from a file.
- Syntax: sed [options] commands [file-to-edit]
- We can simply read a file's content like this:

```
$sed '' /usr/share/gnupg/gpg-conf.skel
```

- We use the single quotes to pass commands in sed – we passe nothing, so it just printed each line it received
- Using **1,5p** between single quotes will print the text between line 1 and 5. Also, **-n** should be used as an option, to clean up the results.

```
$sed -n '1,5p' /usr/share/gnupg/gpg-conf.skel
```

- Sed is very used when we want to perform search and replace text operations
  - Let's suppose we want to replace the string *lines* we found in the file above with *12345*

```
$sed 's/lines/12345/' gpg-conf.skel
```

- We are using the s/pattern/replacement/ command to substitute text matching the pattern with the replacement. The **s** command

operates on the first match in a line and then moves to the next line

- Add the /g modifier to replace every occurrence on each line, rather than just the first occurrence

2) Grep

- The **grep** command is used to search text or searches the given file for lines containing a match to the given strings or words. By default, grep displays the matching lines.

- Create **file1** and fill it with the following:

```
THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.


Two lines above this line is empty.
And this is the last line.
```

- Checking for the given string in **file1**

```
$ grep "line" file1
```

- Checking for the given string in multiple files

```
$cp file1 file2
$grep line file*
```

```
file1:this line is the 1st lower case line in this file.

file1:Two lines above this line is empty.

file1:And this is the last line.

file1.symlink:this line is the 1st lower case line in this file.

file1.symlink:Two lines above this line is empty.

file1.symlink:And this is the last line.

file2:this line is the 1st lower case line in this file.

file2:Two lines above this line is empty.

file2:And this is the last line.
```

- Case insensitive search using grep –i

```
$grep –i "the" file1
```

- Match regular expression in files
    - searches for all the pattern that starts with “line” and ends with “empty” with anything in-between.

```
$ grep "line.*empty" file1
```

- Checking for full words, not for sub-strings using grep –w
    - The following example is the regular grep where it is searching for “is”. When you search for “is”, without any option it will show out “is”, “his”, “this” and everything which has the substring “is”.

```
$ grep -i "is" file1
```

    - The following example is the word grep where it is searching only for the word “is”. Please note that this output does not contain the line “This Line Has All Its First Character Of The Word With Upper Case”, even though “is” is there in the “This”, as the following is looking only for the word “is” and not for “this”.

```
$ grep -iw "is" file1
```

- Highlighting the search using GREP_OPTIONS

```
$ export GREP_OPTIONS='--color=auto' GREP_COLOR='100;30'
$ grep this file1
```

- Displaying lines after the match using grep –A
    - Syntax: grep -A <N> "string" FILENAME, where N stands for the number of lines
    - Create the following:

```
$cat file_text


Line 1

Line 2

Line 3

Example to show the difference between WORD and word

Line 4

Line 5

Line 6
```

```
$ grep -A 3 -i "example" file_text
```

- Invert match using grep –v

- When you want to display the lines which does not matches the given string/pattern, use the option -v as shown below. This example will display all the lines that did not match the word "go".

```
$ grep -v "lines" file1
```

- display the lines which does not matches all the given pattern.
  - Syntax: grep -v -e "pattern" -e "pattern"

```
$ cat file3.txt
a
b
c
d
$ grep -v -e "a" -e "b" -e "c" file3.txt
```

- Counting the number of matches using grep -c

```
$ grep -c line file1
```

- find out how many lines does not match the pattern:

```
$ grep -v -c this file1
```

Exercises

a. Use **grep** to find information about the HTTP protocol in the file */etc/servimces*.
b. Usually this file contains some comments, starting with the '#' symbol. Use **grep** with the **-v** option to ignore lines starting with '#' and look at the rest of the file in less.
c. Add another use of **grep -v** to your pipeline to remove blank lines (which match the pattern **^$**).
d. Use **sed** (also in the same pipeline) to remove the information after the **'/'** symbol on each line, leavimng just the names of the protocols and their port numbers.

3) Vim

- Vim editor is a full screen editor and has two modes of operation:
  a) *Command mode* commands which cause action to be taken on the file
  b) *Insert mode* in which entered text is inserted into the file.

- Start vim

```
$vim file1
```

- Start vim and recover a file that was being edit before a system crash

```
$vim –r file1
```

- Exit vim
  - quit (or exit) vim

:q <Enter>

  - quit vim even though latest changes have not been saved for this vim call

:q! <Enter>

  - quit vim, writing out modified file to file named in original invocation

:wq <Enter>

- Moving the Cursor (open again **file1** and try the following)

  - move cursor down one line

j or <Return> [or down-arrow]

  - move cursor up one line

k [or up-arrow]

  - move cursor left one character

h or <Backspace> [or left-arrow]

  - move cursor right one character

l or <Space> [or right-arrow]

  - move cursor to start of current line

0 (zero)

  - move cursor to end of current line

$

  - move cursor to beginning of next word

w

  - move cursor back to beginning of preceding word

b

- move cursor to first line in file

:0<Return> or 1G

- move cursor to line n

:n<Return> or nG

- move cursor to last line in file

:$<Return> or G

- Screen Manipulation

CTRL+f move forward one screen

CTRL+b move backward one screen

CTRL+d move down (forward) one half screen

CTRL+u move up (back) one half screen

- Inserting or Adding Text

i insert text before cursor, until <Esc> hit
I insert text at beginning of current line, until <Esc> hit
a append text after cursor, until <Esc> hit
A append text to end of current line, until <Esc> hit
o open and put text in a new line below current line, until <Esc> hit
O open and put text in a new line above current line, until <Esc> hit

- Changing Text

r replace single character under cursor (no <Esc> needed)
R replace characters, starting with current cursor position, until <Esc> hit

- Deleting Text

x delete single character under cursor
dw delete the single word beginning with character under cursor
D delete the remainder of the line, starting with current cursor position
dd delete entire current line
u undo whatever you just did; a simple toggle, it's going back just one single step

- Cutting and Pasting Text

  yy copy (yank, cut) the current line into the buffer
  Nyy copy (yank, cut) the next N lines, including the current line, into the buffer
  p put (paste) the line(s) in the buffer into the text after the current line

- Searching Text

  /string search forward for occurrence of string in text
  ?string search backward for occurrence of string in text
  n move to next occurrence of search string
  N move to next occurrence of search string in opposite direction

- Pattern substitution – sed usage –
  - **c** – confirm each substitution.
  - **g** – replace all occurrences in the line.
  - **i** – ignore case for the pattern.
  - **%s** – specifies all lines
  - Examples:
    - Substitute all occurrences of a text with another text in the whole file

```
:%s/old-text/new-text/g
```

    - Substitution of a text with another text within a range of lines

```
:1,10s/old-text/new-text/g
```

    - Substitute only the whole word and not partial match
      - Standard substitution

```
Original Text: This is his idea

:s/his/her/g

Translated Text: Ther is her idea
```

      - Whole word substitution

```
Original Text: This is his idea

:s/\<his\>/her/

Translated Text: This is her idea
```

- Determining Line Numbers

  :.= returns line number of current line at bottom of screen
  := returns the total number of lines at bottom of screen
  ^g provimdes the current line number, along with the total number of lines, in the file at the bottom of the screen

- Saving and Reading Files

:r filename<Return>	read file named filename and insert after current line (the line with cursor)
:w<Return>	write current contents to file named in original vim call
:w newfile<Return>	write current contents to a new file named newfile
:12,35w new_file<Return>	write the contents of the lines numbered 12 through 35 to a new file named new_file
:w! prevfile<Return>	write current contents over a pre-existing file named prevfile

Exercises

1. Create in your home directory the file temp.txt
    a. Type „I use Vim for this excercise."
    b. Save and close
2. Copy */etc/sysctl.conf* file to your home directory. Name it as */etc/sysctl.conf.bak.*
    a. Go to the last line of the file
    b. Go to line 10, add a new line, and add in this text:

```
##This is the text on line 10.
##This is the text on line 11.
##I wrote two lines.
```

    c. Delete the three lines you just created
    d. Save the file, but don't exit.
    e. Go to line 15, copy 10 lines of text. Go to the bottom of the file and place the text there.
    f. Undo the last step
    g. Go to the top of the file, replace all occurrences of "ipv4" with "ipv6", and prompt for each change
    h. Go to line 1, search for "kernel", move to the end of the line, add this text:

```
##some text
```

    i. exit from the file and don't save the changes

Lecture: User management

- Add a regular user to your system

```
$useradd user1
```

- Set a password for user1

```
$passwd user1
```

- Running the `useradd user1` command creates an account named `user1`. If you run `cat /etc/passwd` to view the content of the `/etc/passwd` file, you can learn more about the new user from the line displayed to you:

```
user1:x:2002:2002::/home/user1:/bin/bash
```

- `user1` has been assigned a UID of 2002, which reflects the rule that the default UID values from 0 to 1000 are typically reserved for system accounts. GID, group ID of `User Private Group`, equals to UID. The home directory is set to `/home/user1` and login shell to `/bin/bash`. The letter `x` signals that shadow passwords are used and that the hashed password is stored in `/etc/shadow`.

```
$cat /etc/shadow
```

- Specifying a User's Full Name when Creating a User

```
#useradd –c „User Linux" user1
```

- Adding a User with non-default Home Directory (*user1_dir* should exist)

```
#useradd -d /home/user1_dir user1
```

- Creating a user while copying specific contents to its home directory
  - Usually, when a user is created using defaults, the home directory is populated with files required for environment settings from /etc/skel directory
  - You can see those by listing hidden files in your home directory or in /etc/skel.
  - We can step over the defaults and specify our own directory from where future users home directories are populated.
  - In the first place you'll have to create folder dir1 on /, for example and fill it with two files at your choice (and also the hidden profile files from /etc/skel – otherwise your user profile will look weird ☺)
  - There two ways you can create a custom home directory:
    I. Using options for **useradd** command

```
#useradd -m -k /dir1 user5
```

II. The location of /etc/skel can be changed by editing the line that begins with SKEL= in the configuration file /etc/default/useradd. By default this line says SKEL=/etc/skel. You can put here your path to dir1. Then you can create users as you did before, without any options.

- Setting the Account Expiration Date

```
#useradd -e 2015-11-05 user3
```

- Adding a User with Non-default Shell

```
#useradd -s /bin/ksh user4
```

- Modify user data - Lock/Unlock user account (a password is required to be set before)

  - Lock the account

```
#usermod –L user2
```

  - Unlock the account

```
#usermod –U user2
```

- Delete user account and its files

```
#userdel --remove user3
```

- Creating a Group with Default Settings

```
#groupadd friends
```

  - The **groupadd** command creates a new group called friends. You can read more information about the group from the newly-created line in the /etc/group file

- Creating a group with specified GID

```
#groupadd -g 60002 coleagues
```

- Add existing user to a group

```
# usermod -G user1 friends
# cat /etc/group | grep friends
```

User rights management

- Changing the ownership of a directory and its contents
  - Syntax: chgrp [option] group object

```
#chgrp -R friends dir1
```

- Change object owner
  - Syntax: chown [options] new_owner object(s)

```
#chown user1 file2
```

- Change object group
  - Syntax: chown [options] :new_group object(s)

```
#chown :colleagues file2
```

Exercise

Find a way to change both owner and group for an object using a single command

- Preserving permissions when copying files

```
$cp –p file1 file1.original
```

- Changing file and directory permissions
  - Set read/execute permissions for all categories on **file1**

```
$chmod 555 file1
```

  - Set read/write/execute permissions for owner, read/execute for group and none for others for a directory and its contents (use -R option for doing the operation recursive)

```
#chmod -R 760 dir1
```

  - Test file creation on **dir1** by trying to create files as owner, as **user3** (which belong to **friends** group) and as **user4** (which belong to **colleagues** group)
  - Try reading the file you created previously, as **user3** and as **user4**

Case:

- Special Directory Permissions: 'Sticky'
  - Enable 'sticky' permission on /tmp directory

```
#chmod +t /tmp
#ls -l -d /tmp
```

```
drwxrwxrwt. 17 root root 4096 Oct 26 10:14 /tmp/
```

  - If a file have sticky bit set, only the file's owner may modify it, no matter other permissions that file is having.

- Examine default behavior of newly created files

  - Log in as **user1** and create a subdirectory in **/tmp** with the same name as your username.

  - Create and edit a new file with some lines in it. Store the file in your newly created directory, as /tmp/user1/file.txt.

```
#ls -l /tmp/student/file.txt
```

```
-rw-rw-r-- 1 user1 user1 77 Oct 15 07:12 /tmp/user1/file.txt
```

  - Add **user2** to group **user1** then log in as **user2**
  - Try to modify the file1.txt content.

- Although the **user2** is part of group **user1** (group who have write acces on file1.txt), the permission is denied because the sticky bit was set. Only **user1** can modify the file.
- You can try the same example on a folder without sticky bit, to test the write access on a file as **user2**

Umask

- Set a default set of permissions for a file or a directory on its creation
- When seting a mask, you specifie what permissions the object will not have
- See the current default mask

```
#umask

0022
```

- Change the mask so the newly created objects will have read and write permissions for owner, read permissions for group and none for others

```
#umask 037
```

- Create a new file and check the results

```
-rw-r-----. 1 root root    0 May 30 06:03 file2
```

Exercises

1.

a. Find out who owns the file */bin/ls* and who owns your home directory (in */home*).
b. Log on as root, and create an empty file with **touch**. The user and group owners should be 'root' – check with ls.
c. Change the owner of the file to be 'users'.
d. Change the group owner to be any non-root user.
e. Change both of the owners back to being 'root' with a single command.

2.

a. Find out what permissions are set on your home directory (as a normal user). Can other users access files inside it?
b. If your home directory is only accessible to you, then change the permissions to allow other people to read files inside it, otherwise change it so that they can't.
c. Check the permissions on */bin* and */bin/ls.* Are they reasonable?
d. Check the permissions available on */etc/passwd* and */etc/shadow*.
e. Write one command which would allow people to browse through your home directory and any subdirectories inside it and read all the files.

3.

a. Do the conversion between symbolic mode and octal mode for the following:

Row	Symbolic mode	Octal mode	User/Owner	Group	Other
1	rwxrw-r--				
2	r-xr--r--				
3	rwxr-xr--				
4	rwx------				

b. Do the conversion between octal mode and symbolic mode for the following:

Row	Octal mode	Symbolic mode	User/Owner	Group	Other
1	600				
2	711				
3	724				
4	622				

c. Do the conversion between symbolic mode and octal mode for the following (umask permissions):

Row	Symbolic mode	Octal mode (umask)
1	rwxrw-r--	
2	r-xr--r--	
3	rwxr-xr--	
4	rwx------	