CDOT
SICT AR Meeting Area
People

get involved with CDOT
as a Student
as an Open Source Community Member
as a Company

courses
BTC640
BTH740
BTP300
DPI908
DPS901
DPS905
DPS909
DPS911
DPS914
DPS915
DPS924
DPS931
EAC234
ECL500
GAM531
GAM666
GAM670
GPU610
LUX Program
MAP524
OOP344
OPS235
OPS245
OPS335
OPS345
OPS435
OPS445
OPS535
OPS635
OSD600
OSD700
OSL640
OSL740
OSL840

ULI101 Week 2

Contents [hide]

Linux File System [edit]

Unix File System. [edit]

- The Unix/Linux file system is hierarchical, similar to other operating systems today
  - Files are organized in directories
  - Directories may contain sub-directories
- What is different (from Windows) is that there are no drive letters (such as C:, or D:)? All files and directories appear under a single root, even if multiple storage devices are used.
- Learning command-line navigation of the file system is essential for efficient system usage

Hierarchical File System? [edit]

- In the Linux (Unix) OS, the *root directory* `/` is the starting directory, and other *child directories*, *grandchild directories*, etc. are created.

Real World Mozilla
RHT524
SBR600
SEC520
SPO600
SRT210
ULI101

course projects

Course Project List
Potential Course Projects
Contrib Opportunities

links

CDOT
Planet CDOT
FSOSS

Tools

What links here
Related changes
Upload file
Special pages
Printable version
Permanent link
Page information

The hierarchical structure resembles an *upside-down tree*. There is actually a command called `tree` that can display a *tree diagram*!

```
# display files and directories below root (/) with the tree
command
$ tree /
/
|- home
|   |- user1
|   |- user2
|   |- user3
|   ...
|
|- public
|   |- ipc144
|   |- nled
|
....
```

In the code shown above, the `$` refers to the prompt that waits for you to type something. That's where you type the command `tree`; some sample output of the command `tree` is shown below (notice the leading root directory, i.e. `/` at the top of the output following the `tree` command).

This course will teach you the skills you require to navigate the directory tree in a Linux file system because learning command-line navigation of the file system is essential for efficient system usage and administration.

Typical Unix/Linux Directories [edit]

A sample of some of the commonly found subdirectories that lie below the root directory (`/`) is shown in the table below. The table includes a brief description of the purpose of that subdirectory.

`/`	Root directory (ancestor to all directories).
`/home`	Used to store usersf home directories.
`/bin`	Common system binaries (commands).
`/usr/bin`	Common utilities (commands) for users.
`/usr/sbin`	Common utilities for user administration.
`/etc`	General System Admin. Files (eg passwd).
`/var`	Dynamic files (log files).
`/tmp`, `/var/tmp`	Temporary files for programs.
`/dev`	Device files (terminals, printers, etc.).

Home directory [edit]

- Every user when receiving an account has a *home* directory created.
- This is where you keep your personal files
- `~` represents your home
  - You can use the `~` symbol in pathnames

- A `cd` command without any argument will get you directly to your home directory
- Remember to keep your files private

Types of Files [edit]

On a Unix/Linux file system a *file* can be anything. To an average computer user a file is a container for: a text document, video, music, photo etc.

A directory is a special type of file (index file) containing references to other file locations on the physical disc or to other file related information. Devices like a terminal, a scanner, or a printer are also files. You will learn more details about these types of files later in this course. Any file (or directory) name starting with a period (such as `.` or `..`) is considered a hidden file (or directory). You can use the `ls -l` command to determine the type of file.

For Example:

```
$ ls -l /dev/tty
crw-rw-rw- 1 root root 5, 0 2003-03-14 08:07 /dev/tty

$ ls -l monday.txt w1.c
-rw-r--r-- 1 someuser users 214 2006-01-23 14:20 monday.txt
-rw-r--r-- 1 someuser users 248 2005-10-12 13:36 w1.c

$ ls -ld uli101
drwxr-xr-x 2 someuser users 4096 2006-01-17 16:43 uli101
```

Notes for listing above:

- Use the `-l` option of the `ls` command to get detailed information about a file.
- Use the `-ld` option of the `ls` command to get detailed information for just the directory itself, not the filenames within it
- The first character in detailed listing determines the file type, so:
  - `-` indicates a regular file
  - b or c indicates a device file
  - d indicates a directory

Hidden Files [edit]

A filename that begins with a '`.`' is a hidden file. So, if a filename name starts with a '`.`' as its first character, like `.profile` for example, it is a considered to be a hidden file by Linux commands and is suppressed from the normal listing of files. See the examples shown below:

```
$ pwd
/home/murray

$ ls
uli101.txt

$ ls -a
```

```
. .. .profile uli101.txt

$ ls -A
.profile uli101.txt
```

In the file listing shown above

- `pwd` displays the present working directory. This command is used to display where on in the Linux file system the logged in user is presently working in. In the example shown above, that location is `/home/murray`
- `ls` displays the normal listing of file, i.e. all non-hidden files in the present working directory (this is usually abbreviated to pwd in these notes).
- `ls -a` displays all files including hidden.
- '`.`' and '`..`' directories are hidden.
- `ls -A` displays 'Almost' all files not including `.` and `..`

- Why make files hidden?
  - To clean up directories.
  - To hide backups.
  - To protect important files from accidental deletion.
- Remember: directories are really files, you can hide them as well.

Working With The File System [edit]

- Be very careful when working with files on the command line, as there is no undo command or a Trash/Recycling Bin
  - A single command can wipe out your entire account
  - Changes are instant and permanent
- Make backups of important files, preferably outside of your account - USB storage is a good option
- You will learn later additional ways to control file access through file permissions which will help you prevent accidental file damage or deletion

Basic Commands [edit]

`pwd`

Used to display the user's present working directory. A user may need to know where they are located on the computer system in order to build directories, copy files, etc.

`cd` ***directorypath***

Used to change to a directory. Entering the cd command without a directory name will change to the user's home directory.

`ls`

Used to display the contents of a directory (eg. regular files or sub-directories). By default, the ls command displays non-hidden filenames only.

The following are common options available with the `ls` command

-
- `-a` short display of hidden & non-hidden files
- `-l` detailed display of files (excl. hidden files)

- `-d` combined with `-l` option, displays info about the directory itself instead of the files within it
- Options can be combined, for example: `ls -la` (or `ls -l -a`)

`mkdir` ***directorypath***
: Used to create a directory. Multiple arguments can be used to create multiple directories. The option `-p` (parent) allows multiple directory levels to be created.

`rmdir` ***directorypath***
: Used to remove only empty directories (i.e. directories that contain no subdirectories or regular files). A user cannot remove a directory from within the directory itself.

`mv` ***sourcepath directorypath***
: Used to move a file from one location to another and/or rename the file. The mv command can be used to move directories as well as files. The `-i` option asks for confirmation if the destination filename already exists.

`cp` ***sourcepath directorypath***
: Used to copy a file from one location to another. The cp command can be used to backup important files. – The `-i` option asks for confirmation if the destination filename already exists. – The `-r` (recursive) option allows copying of directories and their contents.

`rm` ***filepath***
: Used to remove a regular file.

`rm -r` ***filepath***
: Used to recursively remove a directory and it's contents. Recursive means to descend to lower levels, which in this case, indicates that subdirectories and their contents are also removed. Note: it is a good idea to include the `-i` option to confirm deletion of subdirectories and their contents!

`cat` ***filepath***
: To display contents of one or more files (i.e. to catenate files). For example, `cat file1 file2 file3` will display the contents of `file1` and `file2` and `file3` on the screen one after the other. To display the contents of small files (files longer than the screen will scroll to the end). For example, issuing the command `cat .bash_profile` in your home directory would display the contents of your setup file.

`more` ***filepath***
: **Used to display the contents of large regular files one screen at a time. The user can navigate throughout the file by pressing keys such as:**

<SPACEBAR>	Move to next screen
b	Move to previous screen
<ENTER>	Move to next line
/car<ENTER>	Search for pattern "car"
q	Exit to shell

`less` ***filepath***
: Works like more command, but contains more navigation features.

`touch` ***path***
: Used to update the date and time of existing files. The `touch` command is also used to

create empty files. You will be using the touch command to create empty files when you practice the file management on-line tutorial

`file` ***path***

Determines a file type. Useful when a particular file has no file extension or the extension is unknown or incorrect.

The find Command [edit]

The `find` command allows searching for files by file name, size, and file attributes recursively throughout the file system. An optional action can be performed on matches

```
#Search for a file named bob:
find / -name bob

# Delete empty files belonging to user alice:
find / -user alice -empty -delete

# Find all files modified less than 5 minutes ago:
find / -mmin -5

# Find large files
find . -size +100M
```

File Naming [edit]

- Unix/Linux is case sensitive!
- Adopt a consistent file naming scheme. this will help you find your files later
- Make your file and directory names meaningful
- Avoid non alphanumeric characters, as they have a special meaning to the system and will make your work more difficult
- Avoid using spaces in file names - consider periods, hyphens and underscores instead
- Feel free to use file name extensions to describe the file purpose

Getting Help with Commands [edit]

- A comprehensive online manual for common UNIX/Linux commands exists on your server
- The online manual is a command called `man`

```
# show man page of ls command
$ man ls
```

```
man [options] command

Options:

-k provides short (one-line) explanation relating to the
commands matching the character string.
This can be used if user doesn't know name of command, eg. man -
k calendar
```

Text Editing [edit]

Editing text files is an everyday activity for both users as well as administrators on a Unix and Linux system

- System configuration files
- Scripts and programs
- Documentation
- Web pages

As the GUI may not always be available, knowing command-line text editors is a very valuable skill.

Please note that although both Unix/Linux and Windows use ASCII to encode text files, there are small differences that may cause problems (particularly with scripts) when copying files between different systems:

- If needed, use the `unix2dos` and `dos2unix` utilities to convert between the two systems
- A specific system may have many editors available and as you work with one for a while you will probably pick a favourite one
- A traditional fall-back is the `vi` editor, as it is most likely to be present on all Unix-like systems, especially when installed with a minimum software complement

Consider knowing `vi` as one of many badges of a competent Unix/Linux user

- `vi` has a relatively steep learning curve and is not user friendly, but it offers nice advanced features which will be introduced later in the course (Visual) Editor

`vi` is a powerful, interactive, visually-oriented text editor with these features:

- Efficient editing by using keystrokes instead of mouse.
- Use of regular expressions
- Possibility to recover files after accidental loss of connection
- Features for programmers (eg. line numbering, auto-indent, etc)
- Although you may prefer to use other editors (such as `nano` or `nled`), knowing `vi` is very useful, as this is one editor that is present on all Unix-like systems

Starting vi Session [edit]

There are two ways to start an editing session with `vi`:

– Enter `vi` *filename* -recommended since *filename* has already been assigned and changes will be saved to that *filename* when saving within `vi`, for example `:w<ENTER>` – If the *filename* exists, it will be edited. If the *filename* doesn't exist, it will be created. – Enter `vi` - filename is not assigned, therefore user has to type `:w filename<ENTER>` in order to save the file.

Modes [edit]

There are three operational modes while using the vi editor:

– Command Mode (default mode when starting) :: User presses letter(s) for a command

– for example to input text, delete text, append text, etc. Does NOT require `<ENTER>` key, the keystrokes are used individually.

– Input Mode :: Input Mode allows user to enter or edit text. Press `<ESC>` to return to command mode.

– Last-line Mode :: Pressing colon ":" opens a prompt at the bottom of the screen to enter more complex commands, such as search and replace. Requires `<ENTER>` key to execute command.

Moving in Command Mode [edit]

You can move around to text in the screen by using the following keys:

h	left
j	down
k	up
l	right
w	right one word to special character
W	right one word including special characters
b	left one word to special character
B	left one word including special characters
0 (zero)	beginning of line
$	end of line
G	go to last line in file
237G	go to line 237 in file

- You may be able to move around by using the arrow keys (depends on version of vi).

Getting into Input Mode [edit]

While in command mode, you can issue the following commands to input text:

i	insert to left of cursor
I	insert at beginning of line
o	insert line below current line
O	insert line above current line
a	append to right of cursor
A	append at end of current line
r	replace character under cursor
R	overwrite text character-by-character

Don't forget to hit <ESC> to return to command mode.

Common Editing Commands [edit]

x	Delete single character under the cursor
d	Delete

eg. dw - delete from the current position to the next word or special character

eg. d$ - delete from the current position to the end of the line

eg. dd - delete the entire current line

c	Change eg. cw - change from the current position to the next word or special character eg. c$ - change from the current position to the end of the line eg. cc - change the entire current line
y	Yank (copy) eg. yw - copy from the current position to the next word or special character eg. y$ - copy from the current position to the end of the line eg. yy - copy the entire current line
p	paste deleted or copied text after or below cursor
P	paste deleted or copied text before or above cursor
u	undo previous edit
.	repeat previous edit

Editing commands can be preceded with a number, for example: 3x = delete the next three characters 2u = undo the last two edits 12dd = delete 12 lines

Searching [edit]

Search for text (in command mode)

/pattern	Search forward for pattern
?pattern	Search backwards for pattern
n	Display next match

Saving Edited File [edit]

- Work performed during vi session is stored in a Work Buffer (temporary storage) until the user saves their work.
- To save your vi session, make sure you are in command mode by pressing `<ESC>`
- To save your changes and exit, type `ZZ` (two capital z's). You can also use either `:x<ENTER>` or `:wq<ENTER>`
- You can save without exiting by typing `:w<ENTER>`

Aborting Editing Session [edit]

If you make a mistake in your editing session that undo cannot easily solve, you can abort your session without modifying the contents of your file by using the following last-line command: `:q!<ENTER>`

Categories: Pages with syntax highlighting errors | ULI101 | ULI101-2018