Linux Directory Structure and Basic Bash Commands

COMPSCI 215

Linux Tree Hierarchy

- Linux file system is based on a tree hierarchy.
  - *There is a top-level with other sublevels branching beneath it*.
- The tree hierarchy offers storage and quick access.
- Unlike the *Windows* file system, where it uses DRIVE LETTERS, *Linux* stores everything within a single directory structure called a VIRTUAL DIRECTORY.

Linux File System

- **WINDOWS**

  `C:\Users\Rich\Documents\test.doc`

  \ → Backslash (Windows)

- Indicates that **`test.doc`** is located in **`Documents`**, which itself is located in directory **`Rich`**. **`Rich`** is contained in directory **`Users`**, which is located on the hard drive partition assigned letter **`C`**.
- **`C`** is usually the first hard drive on the PC.

- **LINUX**

  `/home/rich/Documents/test.doc`

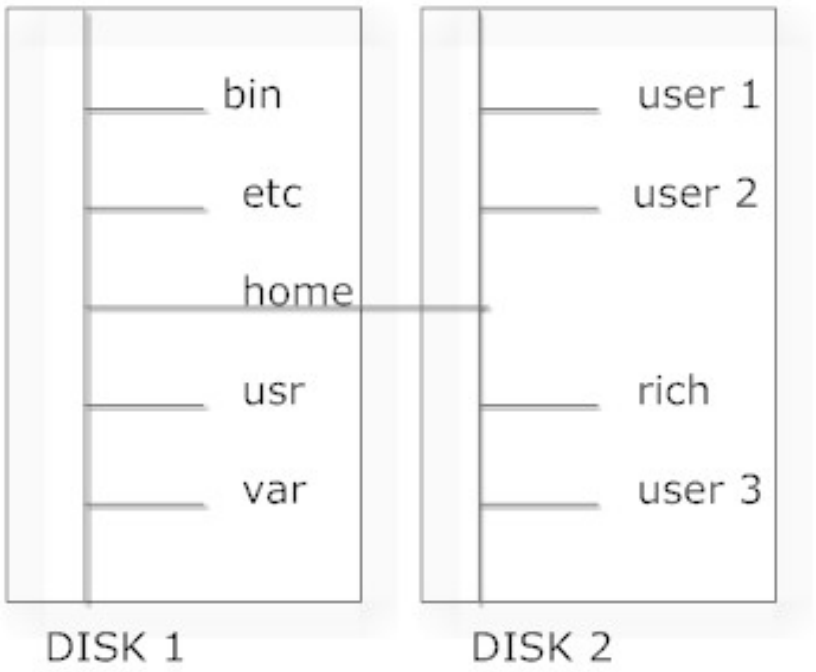
  / → Forward slash (Linux)

- Indicates only that the file **`test.doc`** is in directory **`Documents`**, under the directory **`rich`**, which is contained in the directory **`home`**.
- It does not provide information as to which physical disk on the PC the file is stored.

How Does Linux Incorporate Each Storage Drive?

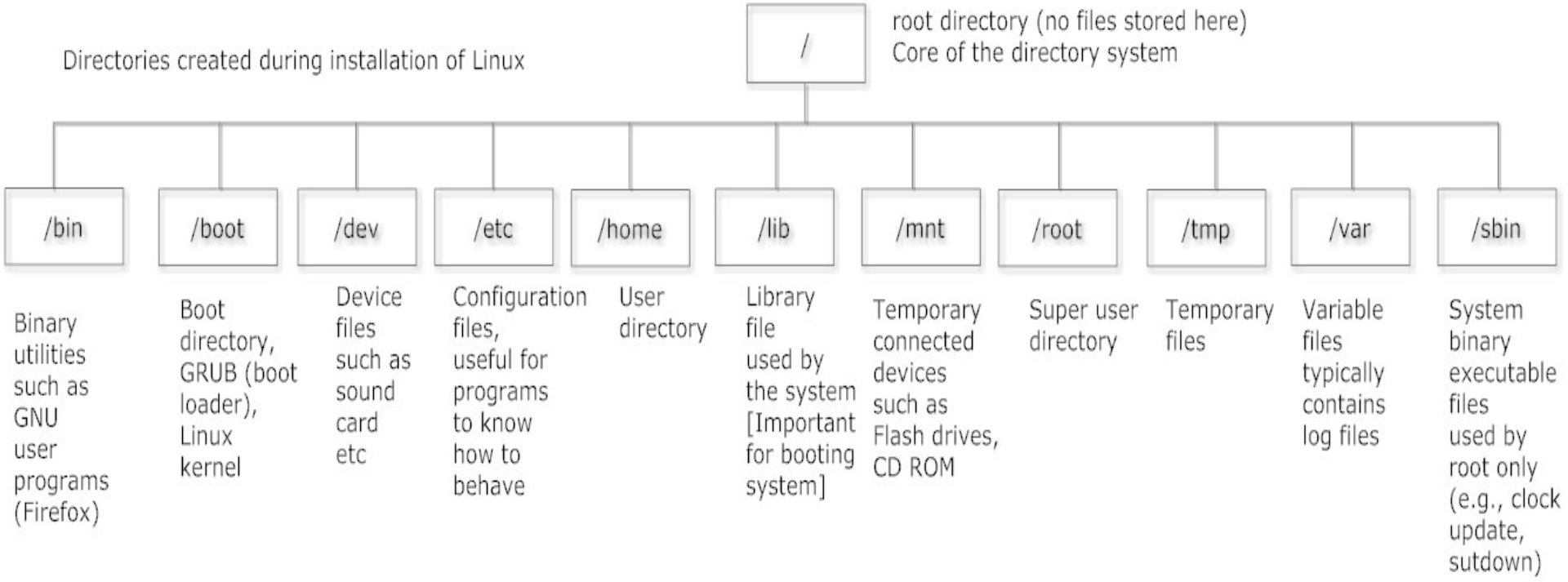
- First hard drive installed in a Linux PC is called **root drive.**
  - **Root drive** contains core of the virtual directory. Everything else builds from there.
- On the **root drive**, Linux creates **mount points** (these are special directories where you assign additional storage devices).
- *The virtual directory causes files and directories to appear within these mount points*, even though they are physically stored on a different drive.

The Linux File Structure



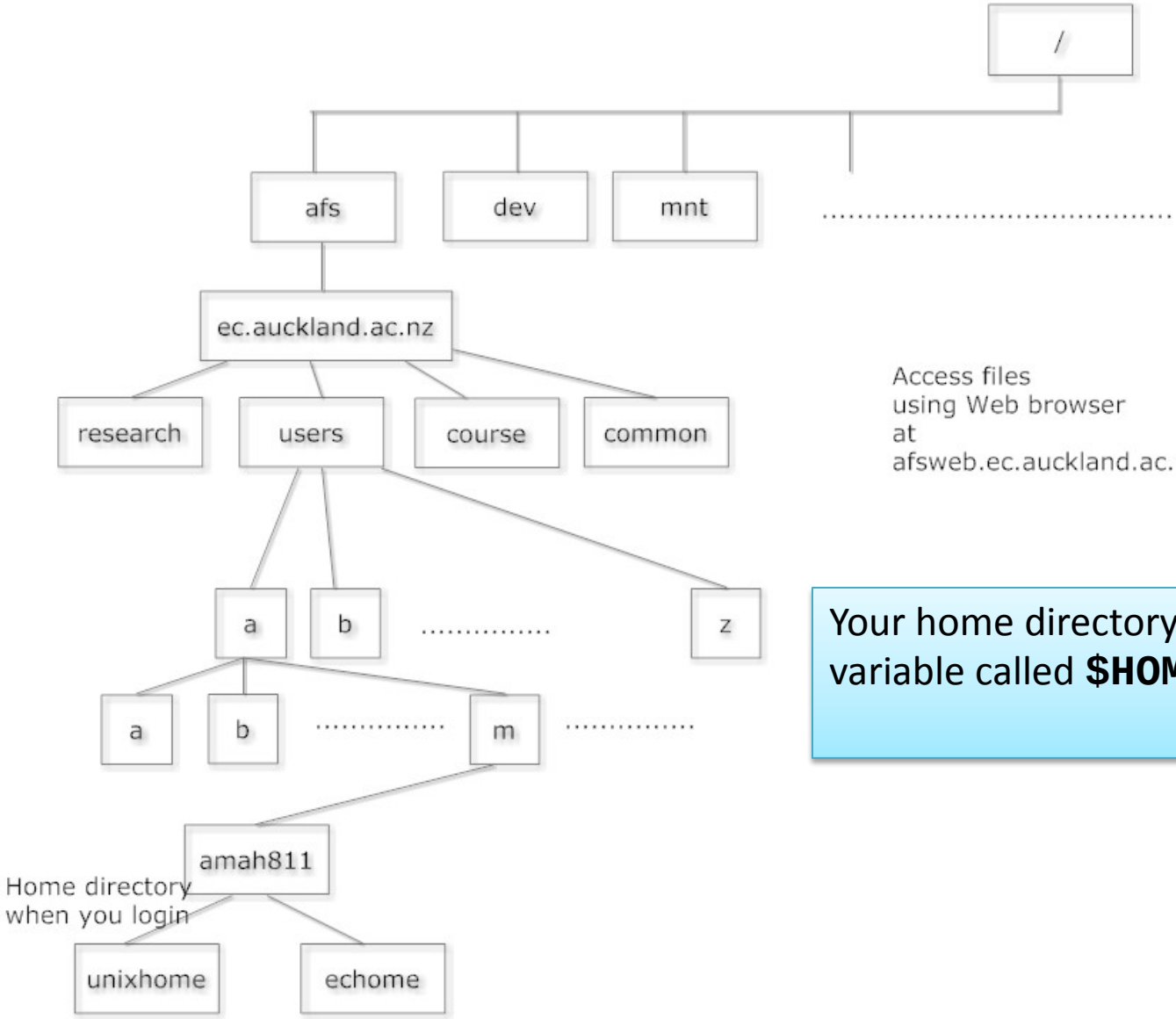
- One hard drive is associated with the root directory.
- **Other hard drives can be mounted anywhere in the directory structure.**
- Second hard drive is mounted on `/home` where user directories are located.

Linux System Directories



- The Linux filesystem structure has evolved from Unix file structure.
- The file structure has been somewhat convoluted over the years by different flavours of Unix.
- There are few common directory names that are used for common functions.

University Directory Structure



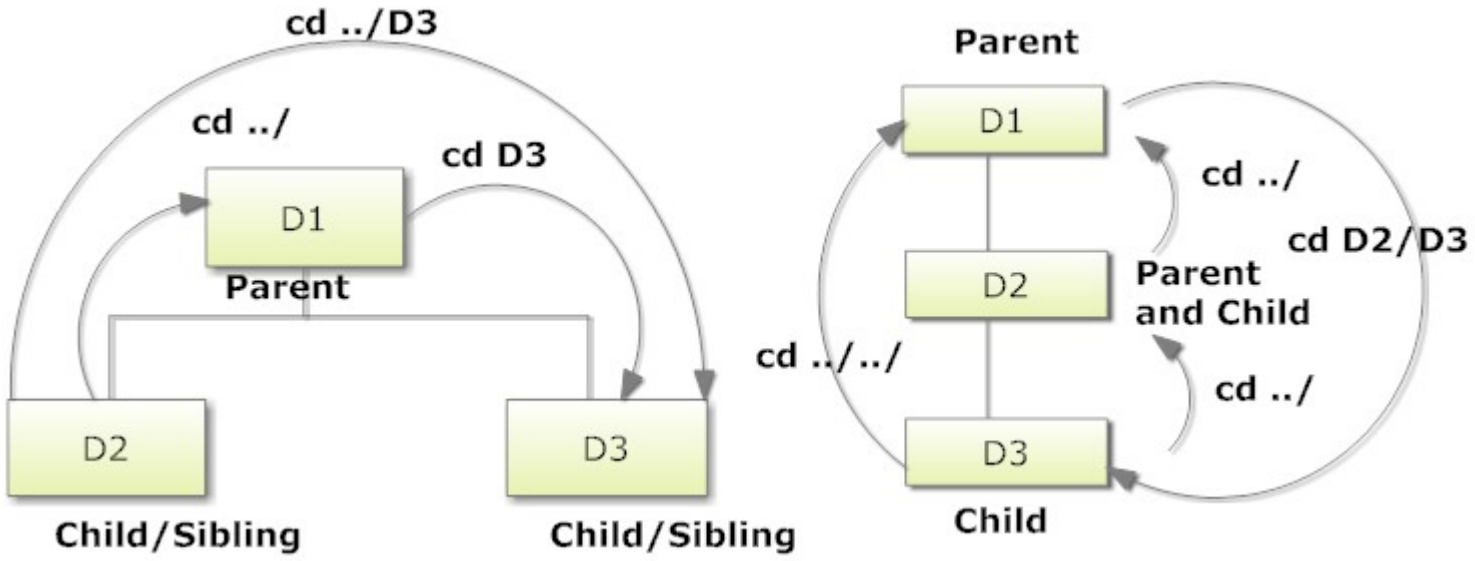
Your home directory path is stored in a variable called **$HOME**.

Traversing Directories

- We can refer to a file by **absolute** or **relative path** names.
- Absolute pathname: Specifies full path from the root to the desired directory or file
  - Absolute pathname to my home directory is **`/afs/ec.auckland.ac.nz/users/a/m/amah811/unixhome`**
  - To refer to file1 in my home directory, the absolute pathname: **`/afs/ec.auckland.ac.nz/users/a/m/amah811/unixhome/file1.txt`**
- Relative pathname: Path from current directory to a file or directory
  - If my current directory is **`/afs/ec.auckland.ac.nz/users/a/m/amah811`**
  - Relative path to file1.txt in my home directory is **`unixhome/file1.txt`**
  - And relative path to my home directory is **`unixhome`**

Changing Directories (cd)

- A relative file path starts with either a directory name or a special character indicating a relative location to your current directory location.
  - The dot (.) represents current directory
  - The dot dot (..) represents the parent directory



Linux Bash Commands

- The Linux commands could be divided into three categories:
  - Directory commands: Commands that work with directories (`pwd, ls, cd, mkdir, rmdir`)
  - Editor commands: Commands that allow you to manipulate files (Editors such as `vi` or `gedit`)
  - File commands: Commands that allow you to manage files (`cp, mv, sort, cut, paste, diff, rm, uniq`)

pwd command

- Displays the current working directory.
- *Does not have many practical options, but one of the most important commands as it lets you know where you are in the tree hierarchy*.

```
amah811@login01:~$ pwd
/afs/ec.auckland.ac.nz/users/a/m/amah811/unixhome
```

File and Directory Listing (`ls`)

- Basic listing (`ls`): Displays files and directories in your current directory.
- *It produces listing in alphabetic order (columns)*.
- To distinguish between files and directories use the `-F` parameter.

```
amah811@login01:~$ ls
12APR2013       4APR2013      D1          dir2        exitcodetest.sh  first.dat        iftest.sh   nine        test1.sh    test22.sh
12APR2013.tgz   5APR2013      ZipCode.sh  dirtree.sh  file             foo              index.html  script2.sh  test123.sh  test3.sh
215A1_clu034    5APR2013.tgz  assign.tgz  dtree.sh    fileA            forloopshift.sh  last.dat    test.sh     test2       test4.sh
4APR.tgz        A1            dir         echo        fileB            hat              myclasses   test1       test2.sh    testdir
amah811@login01:~$ ls -F
12APR2013/      4APR2013/     D1/         dir2/       exitcodetest.sh* first.dat       iftest.sh*  nine        test1.sh*   test22.sh
12APR2013.tgz   5APR2013/     ZipCode.sh* dirtree.sh* file             foo              index.html  script2.sh* test123.sh* test3.sh*
215A1_clu034/   5APR2013.tgz  assign.tgz  dtree.sh*   fileA            forloopshift.sh* last.dat    test.sh     test2/      test4.sh*
4APR.tgz        A1/           dir/        echo*       fileB            hat/             myclasses/  test1/      test2.sh*   testdir/
```

/ → Directory, * → Executable

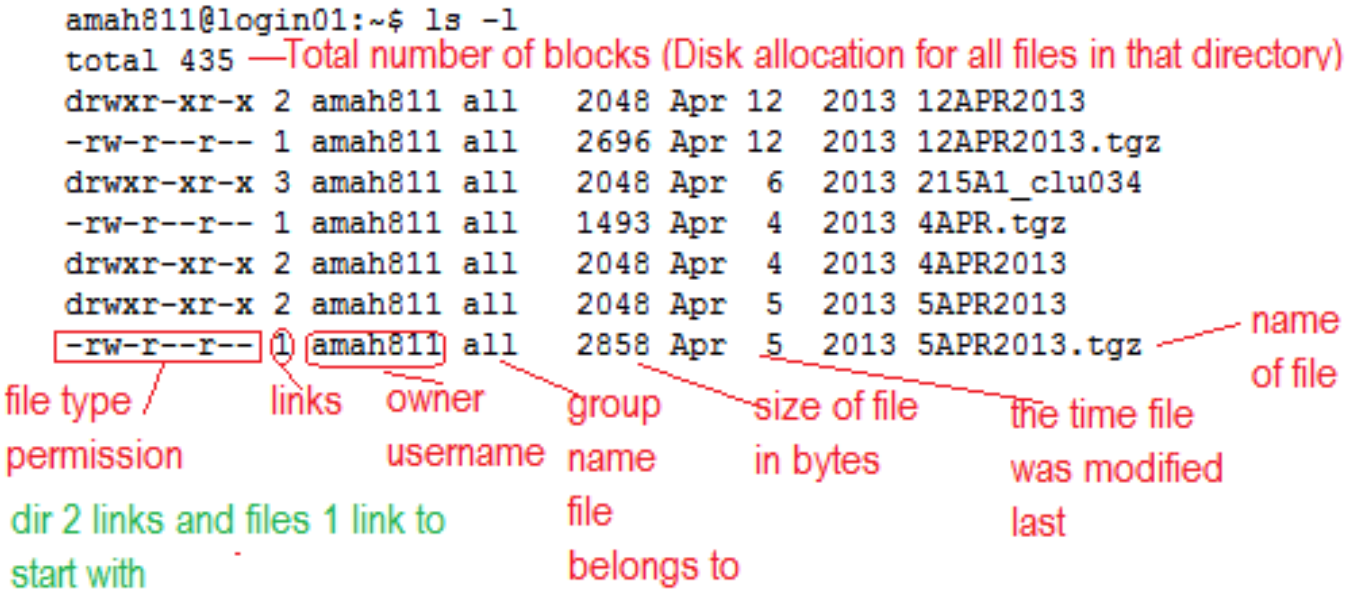
ls parameters

- Use the -a parameter to show *hidden files* starting with a dot (.). Notice the (.) and (..) directories.
- Use the –F along with –R parameter (ls –FR) to recursively show the contents of all directories contained in the directory where you do the listing.

```
amah811@login01:~$ ls -aF
./    .Xauthority     12APR2013/     5APR2013/     assign.tgz     echo*            first.dat          index.html    test.sh      test2.sh*
../   .bash_history   12APR2013.tgz  5APR2013.tgz  dir/           exitcodetest.sh* foo                last.dat      test1/       test22.sh
.123  .bash_profile*  215A1_clu034/  A1/           dir2/          file             forloopshift.sh*   myclasses/    test1.sh*    test3.sh*
.124  .emacs.d/       4APR.tgz       D1/           dirtree.sh*    fileA            hat/               nine          test123.sh*  test4.sh*
.???  .viminfo        4APR2013/      ZipCode.sh*   dtree.sh*      fileB            iftest.sh*         script2.sh*   test2/       testdir/
```

More useful `ls` parameters

- The basic listing does not produce much information.
- Use the `-l` parameter to produce a long listing.



More `ls` parameters

- Use the `-i` option for *inode* information.
  - *inode* provides a unique identification number the kernel assigns to each object in the file system.
- Use the `-s` parameter to print block size of each file.

```
amah811@login01:~$ ls -sail
total 470
1251606533   4 drwxr-xr-x 16 amah811 root   4096 Mar  5 08:58 .
 911212902   2 drwxrwxrwx  4 amah811 root   2048 Aug 18  2012 ..
1251606650   0 -rw-r--r--  1 amah811 all       0 Apr  9  2013 .123
1251606664   0 -rw-r--r--  1 amah811 all       0 Apr  9  2013 .124
1251606666   0 -rw-r--r--  1 amah811 all       0 Apr  9  2013 .???
1251606556   1 -rw-------  1 amah811 all     288 Mar  5 08:57 .Xauthority
1251606530  13 -rw-------  1 amah811 all   12719 Mar  2 13:29 .bash_history
1251606748   1 -rwxr-xr-x  1 amah811 all     427 Mar  2 13:27 .bash_profile
```

inode number

block size

cp command (Copying)

- Allows you to make copy of a file.
  - `cp [option] source destination`
  - `-i` option for *interactive mode to give you a warning* before overwriting an already existing file.
  - `-p` option to preserve file access or modification times of the original file for the copied file.
- Allows any combination of full or partial path
- Copy from current directory to parent directory:
  - `cp source ../`
- Copy from current directory to sibling directory:
  - `cp source ../sibling`
- Copy file from parent to current directory:
  - `cp ../source .`
- Recursively copy contents of a directory to another
  - `cp –R source_dir destination_dir`

`mv` command (Moving)

- Allows you to move or rename a file or directory
  - `mv source destination`
  - `-i` option for interactive
- Source and destination can have same name only when moving file to another directory.

rm/rmdir command (Removing)

- Allows deleting of files:
  - `rm –i file`
  - Use the `–i` to get the warning as there is no Recycle Bin to recover the file.
  - Use `-f` option for forcible deletion (no warnings).
- `rmdir` allows deleting of empty directories.
- `rm –rf` directory recursively deletes all contents of a file.
  - The `-f` option is to suppress warnings about descending into sub-directories and deleting contents.

`touch` command

- Used to update the modification date and time of a file; does not modify the contents.
- If a file does not exists, then touch can be used to create an empty file.
- `touch filename`
- Use the `-t` parameter to specify the time.
  - `touch -t 201411251200 testfile`

Viewing File Contents

- `stat` provides a complete rundown of a file

```
amah811@login01:~/myclasses$ stat dtree.sh
  File: `dtree.sh'
  Size: 437             Blocks: 2          IO Block: 4096   regular file
Device: 10h/16d Inode: 1251606550  Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (2375829/ amah811)   Gid: (62215/     all)
Access: 2013-03-17 00:29:00.000000000 +1300 (last accessed)
Modify: 2013-03-17 00:29:00.000000000 +1300 (last modified)
Change: 2013-03-17 00:29:00.000000000 +1300 (last time meta data of file changed, e.g. file permission)
```

- Does not provide file type. Use `file` command, which classifies files into:
  - Text files (printable characters)
  - Executable files (can run on the system)
  - Data files (non-printable binary characters, but can't run on the system)