File Properties

- File attributes
- The “touch” command
- Shell meta characters

Command "ls"

- "ls": list content of a directory
- "ls" can take many options, some are listed below
  - Options are prefixed with a short dash "–"
  - Options can be combined: `ls -al` ←→ `ls -a -l`

Option	Function
`-l`	Long listing format, list detail information
`-a`	List all files, include hidden files (file name starts with a dot), and the two default directories `.` & `..`
`-A`	Almost all, everything but the `.` & `..`
`-t`	Sort by modification time
`-1`	List one file (one column) per line
`-R`	List subdirectories recursively
`-r`	Reverse order while sorting
`-h`	Human-readable size when using with "-l" option
`-s`	Print size of each file by block
`-S`	Sort by file size

`ls` and pipe

- “`ls -l`”
  - List the content of the directory in alphabetic order, from a to z
- “`ls -lr`”
  - List the content of the directory in reversed alphabetic order, from z to a
- “`ls -lS`”
  - List the content of the directory in the order of size, from large to small
- “`ls -lSr`”
  - List the content of the directory in the order of size, from small to large
- With pipe and wc
  - Q: How to count the number of files contained in a directory (do not count the two default special directories)?
  - `ls -A | wc`

File Properties

```
hlin@linux:~> ls -l
drwxrwxrwx  2  hlin   csuser 4096 2006-07-17 14:38 public_html
-rw-r--r--  1  hlin   csuser   14   2006-07-17 15:35 test.txt
```

- `ls -l` returns 8 columns for each entry in the directory
  - 1st - file mode (type and permissions)
  - 2nd -number of links associated with the file”
  - 3rd - ownership of the file (user name)
  - 4th - group assigned to the user
  - 5th - size in bytes (by default)
  - 6th - date of last modification
  - 7th - time of last modification
  - 8th - file name

Types of UNIX Files

- Regular file (-)
- Directory file (d)
- Symbolic link: a type of file that points to another. (l)
- Character special file, providing unbuffered I/O access. (c)
- Block special file, such as hard drive, proving buffered I/O (b)
- Pipe, also called FIFO: a type of file used for communication between processes. (p)
- Socket: a type of file used for network communication between processes. (s)

inode

From the Linux Information Project:

- An inode is a data structure on a filesystem on Linux and other Unix-like operating systems that stores all the information about a file except its name and its actual data.
- A file is a named collection of related information that appears to the user as a single, contiguous block of data and that is retained in storage.
  - Storage refers to computer devices or media that can hold data for relatively long periods of time.
- A directory in Unix-like OS is merely a special type of file that associates file names with a collection of inodes. A file name is just an entry in a table with inode numbers, rather than being associated directly with a file.
- When a file is created, it is assigned both a name and an inode number, both are stored as entries in the directory that appears to the user to contain the files.
- Space for inodes must be set aside when an OS is installed and that system does its initial structuring of the filesystem.
  - Within any filesystem, the max # of inodes (files) is set when the filesystem is created.
  - Run out of space:
    - consume all the space,
    - use up all the inodes with many very small files

Symbolic Link File

- Create a symbolic file pointing to another file (target file)

  `ln -s <target_file> <link_file>`

  - The TARGET can be a file or directory (the source)
  - Both the TARGET and the LINK_FILE include path information (??)
  - Check: `ls -l /usr/bin/sh`

- Symbolic link file provides convenience for file or directory access

**Note about "hard link"

  - `ln <target_file> <link_file>`
  - Its just another entry in the directory pointing to the same data (inode), it's like the data file has more than one name.
  - If you remove one of the hard links, the file still exist.
  - "hardlink" is not for "directory" and cannot across filesystems.

- Command to list inode info:
  - `ls -i`

File Permissions

- Three types of permissions: rwx
  - For directory:

r = permission to read directory entry, get list of the directory contents

w = permission to create or remove files or directories under it

x = examine the directory (cd into it)

- = no permission

You cannot browse a directory which has no “x” permission to you

- File access is controlled by three groups
  - User (u): the owner of the file
  - Group (g): the group the user is assigned to
  - Other (o): those not in the group user assigned to

9 regular permissions (9 bits) for each file

```
drwxr-xr-x  2 hlin csuser 4096 2015-07-17 14:38 public_html
```

```
--- | --- | ---
```

Octal	**Binary**	**Permissions**
0	000	---
1	**001**	**--x**
2	**010**	**-w-**
3	011	-wx
4	**100**	**r--**
5	101	r-x
6	110	rw-
7	111	rwx

Modify File Permissions -- chmod

`chmod` : change any permissions for a file you have ownership

- Use the octal numbers:

```
chmod 655 filename
chmod –R 655 dirname
```

- Use the following special symbols
  - u: user (the owner); g: group; o: others; a: all
  - add permissions (+); remove permissions (-); set permissions (=)

```
chmod u+x filename
chmod u+a filename
chmod a=rwx filename
chmod g=x filename
chmod go=x filename
Chmod a-x filename
```

Q: What are the file permissions after running the following cmd? (assuming file currently has `rw-r--r--`)

- “`chmod a+x file`”
  - “`rwxr-xr-x (755)`
- “`chmod a=r file`”
  - “`r--r--r--`” `(444)`
- “`chmod o-rw file`”
  - “`rw-r-----`” `(640)`
- “`chmod g+x file`”
  - “`rw-r-xr--`” `(654)`
- “`chmod ug+x fname`”
  - “`rwxr-xr--`” `(754)`

Questions:

- *Explain permissions “755”, “444” and “666” for a regular file.*
- *Can you “cd” a directory which has permissions 766? How about 755? How about 744?*

755	rwx|r-x|r-x
444	r--|r--|r--
666	rw-|rw-|rw-

Two Special Permissions (s & t)

- `ls -l /usr/bin/passwd`, you get the `-rwsr-xr-x`
- `ls -l /`, you will see `drwxrwxrwt" for "tmp"`
- What are the "`s`" and "`t`" ?
  - "s" is for SUID and SGID (4 for SUID, 2 for GUID) for special programs (executable files), meaning: `Set-User-ID or Set-Group-ID`
    - If the `SUID bit` is set, when the program is executed, the effective UID is set to the owner of the file, not the UID of the person who runs the program
    - This is useful when the user needs a special permission, such as to write to the password file in the case of changing password
  - Sticky bit "t", a special bit for directory (1 for the sticky bit)
    - If sticky bit is set, files under the directory can be removed or renamed only by its owner. This is commonly used for public temporary directories, such as `/tmp`
    - `/tmp` directory will be cleaned up after rebooting the system (normally)
- The 3 special bits or special group(`sst`), a file permission mode can be represented as: 0755, 4755, 1755
- Normally, if none of sst is set (for regular files/directories), the "0" is dropped, just use 3-octave.

- What permissions the file (666) has afterwards? (represent with letters and octave numbers)
  - `chmod a+s filename`
  - `chmod u+s filename`
  - `chmod 4755 filename`
  - `chmod 6755 filename`
  - `chmod 1755 filename`
  - `chmod 1755 dirname`
  - `chmod a+t dirname`
  - `chmod g+t dirname`
  - `chmod u+t dirname`
  - `chmod o+t dirname`

`umask` – set file mode creation mask

- When a file is created, it is given a set of default permissions which are determined by the program creating the file
  - Initially the program gives 666 for regular file and 777 for directory
- By setting different "umask", the default permissions for regular files and directories can be different
  - By default, umask is normally set to `0022` (check the current setting with "umask") `(000|000|010|010)`
    - The created regular file has permissions 644 (666 subtracts 022)
    - The created directory file has permissions 755 (777 subtracts 022)
  - "umask" can be reset: "umask 0002", the default permissions
    - For regular file: 0664
    - For directory: 0775

The “touch” Command

- Changes file timestamp
- Creates an empty file if the file does not exist
  - Use “touch” to create a set of files to practice the file permissions, etc
- File timestamp is not reliable, why?
  - Since it can be changed with “touch” command!!

Shell Metacharacters

- What are metacharacters?
  - Special characters used to represent something other than themselves
- Shell metacharacters used by the shell for file name matching
  - * - matches zero or more characters of any type
  - ? – matches for a single character of any type
  - { } – matches for any of a list of comma-separated strings, normally the strings are file names, different suffixes, separated with “,”, NO WHITE SPACE
  - [ ] – matches any one character in the set
  - [!abc] or [^abc]– not matches any character in the set, (not a,b or c)
- Use back slash (\) to disable metacharacters
- In UNIX, letters are case sensitive

A directory contains the following files:
ab, abc, a1, a2, a3, all, a12, ba ba.1, ba.2, filex, filey, AbC, ABC, ABc2, abc, abc123, file1, file1.bak, abc122, file2, file2.bak, none, nobody, nothing, one, nowhere, nobody, nonsense

ls command	Files Listed
*ls a**	List all files starting with a
*ls *[0-9]*	List all files ending in at least one digit
*ls [aA]**	List all files starting with a or A
ls [a-zA-Z][a-zA-Z]	List all files containing just two alphabetic characters
ls [A-Z][A-Z][A-Z]	List three character files where all letters are uppercase
*ls *.cpp *.h*	List files ending in cpp
*ls *.{cpp,h}*	List all .cpp and .h file
*ls no{ne,th,n}**	List files starting with none, noth, non, followed by anything
*ls *[0-9A-Za-z]*	List all files ending in a digit, an uppercase letter, or a lowercase letter
ls a?c?	List files starting with “a”, followed by a single character, followed by “c”, and another single character
echo ? ls ?	The shell treats “?” as literal question mark if it cannot find a match
ls [^abB] ls [!abB]*	List all files not starting with a, b, or B

Review Questions

- Your current directory contains the following files:

```
ab abc a1 f.cpp time.h a2 a3 all a12 ba ba.1 ba.2
filex filey AbC ABC ABc2 abc a.c a.cpp alex
```

- Give the command that will do the following:
  - list all files starting with **a**.
  - list all files ending in at least one digit
  - list files ending in a **x** or **y**
  - list all files whose name contains two characters only
  - **remove** two character files starting with **a** or **A**
  - **Create a tarball of all files starting with "a"**
- What will be listed with the following command?
  - `ls *[0-9]`
  - `ls a?c?`
  - `ls *.{cpp,h}`