



Western  
UNIVERSITY • CANADA

# Files and Directories

Winter 2022



# File and Directory Types

- What is a file?
  - A container for data
  - Persistent (stays around)
  - Accessible by a name
- In Unix, EVERYTHING is a file

# File and Directory Types

- Unix file types  
([https://en.wikipedia.org/wiki/Unix\\_file\\_types](https://en.wikipedia.org/wiki/Unix_file_types))
  - Regular
  - Directory
  - Device
  - Link
  - Others

# File and Directory Types

- Behind the scenes, the name in a directory is translated to a specific location on disk (disk #  $\rightarrow$  cylinder #  $\rightarrow$  track #  $\rightarrow$  sector #  $\rightarrow$  block #). This is known as an "inode number" (index node number).
- This is why it is called a "directory" instead of a "folder". The directory contains an index of files (like the index in a book). The files do not physically "live" inside a folder.

# File and Directory Types

- Regular file
  - Text file – Contain "printable" characters
  - Binary file – Contain any ASCII characters from 0 to 255

# File and Directory Types

- Directory ("file")
  - Contains the names and inode numbers for all files and directories in this directory
  - We treat it like a folder

# File and Directory Types

- Device file
  - Allows I/O from a device (e.g. Soundcard, mouse, etc.)
- Links
  - Hard links
  - Soft links
    - Like a shortcut



# File and Directory Types

- Links (continued)
  - Soft links are popular and are often called "symbolic links" or "symlinks"
  - `ln -s <target directory> <link name>`

```
[wbeldman@compute ~]$ ln -s /tmp ~/global-temp-directory
[wbeldman@compute ~]$ ls -l ~/global-temp-directory
lrwxrwxrwx 1 wbeldman wbeldman 4 Jan 14 01:50 /home/wbeldman/global-temp-directory -> /tmp
```

# File and Directory Types

- How do you find out what type a file is?  
Three suggestions:
  - `ls -F`
  - `ls -l`
  - `ls --color=auto`
  - (or all of them: `ls -F -l --color=auto`)

# Pathnames and links

- Unix has directories and subdirectories. When referring to a path (e.g. the `cd` command), you will use one of two types
  - Absolute path
  - Relative path

# Pathnames and links

- Absolute path
  - Begins with a / (the root)
  - Explicitly uses the entire path from root all the way to the subdirectory
  - E.g. /home/wbeldman

# Pathnames and links

- Relative path
  - Dependent on what your current working directory is. E.g.
  - `cd tmp` - means change directory to tmp inside my current working directory. This is not the same as `cd /tmp`

# Pathnames and links

- When referring to a location in the system, your command will check in the following order
  - / - The root
  - ~ - The home directory
  - . – The current directory (./ to be more explicit)
  - .. – The parent directory (../ to be more explicit)
  - Otherwise try the current working directory

# Pathnames and links

- E.g. If I am in /home/wbeldman, the following are all equivalent
  - /home/wbeldman/cs2211/readme.txt
  - ~/cs2211/readme.txt
  - cs2211/readme.txt
  - ~/cs2211/otherfolder/../readme.txt

# Pathnames and links

- Another way to combine shortcuts are like this:
  - `cd ../../../../` - Go up three directories in the tree
- When running a command found in the current working directory (e.g. your compiled C program), use the `./` shortcut
  - `./myProgram`



# Wildcarding

- We can use special characters to represent a sequence of other characters
- When using a wildcard to match multiple files, this is known as "globbing"
  - \* – matches 0 or more characters
  - ? – matches exactly one character
  - [...] – matches any one character in the list

# Wildcarding

- E.g.
  - `a*.c*` matches **abc.c** and **abra.cpp**
  - `a?.c` matches **ab.c**, **ax.c**, but not **abc.c**
  - `b[aei]t` matches **bat**, **bet**, or **bit**, but not **but** or **baet**
  - `b[!aei]t` matches **but** but not **bat**, **bet**, **bit**, or **baet**

# Wildcarding

- E.g. combining sequences
  - `mv a*[ch] cfiles/` – move all files beginning with a and ending in .c or .h into cfiles
  - `ls [abc]*.*` – list all files beginning with a, b, or c, followed by (possibly) anything, followed by a dot, followed by a single character

# Wildcarding

- Wildcards do not traverse directories. It only matches in the current directory
  - E.g. `csnow*c` does NOT match `csnow/codec`
- Wildcards do not match "hidden files".  
Hidden files are files that start with a . (dot)
  - E.g. `"cat *profile"` will not find `".bash_profile"`

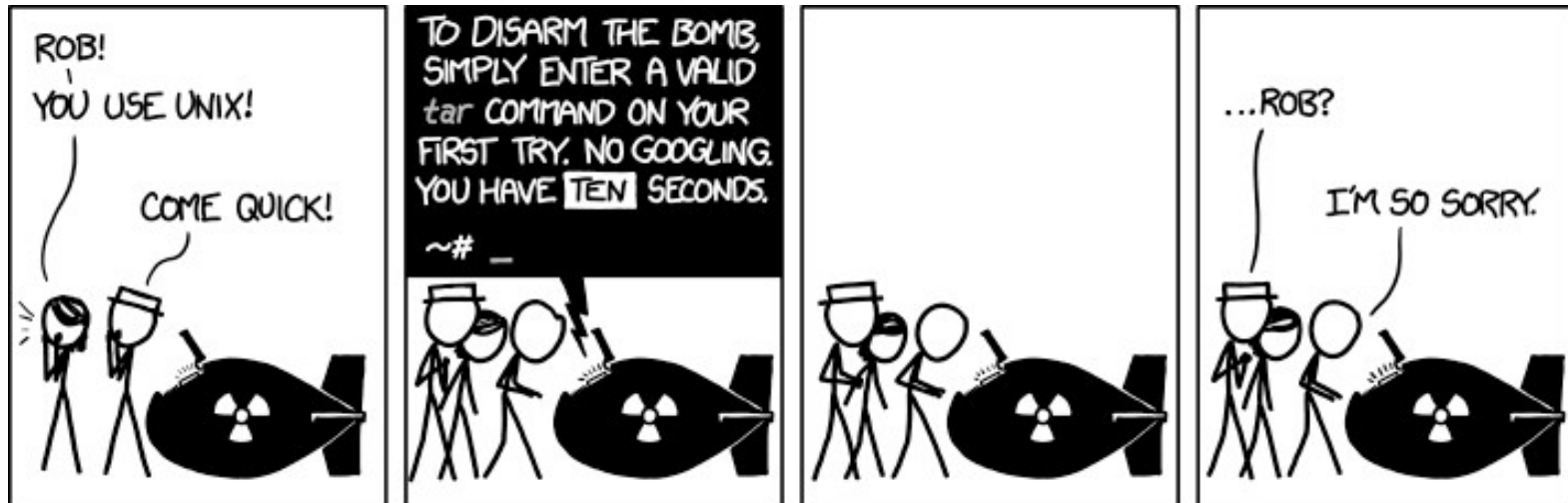
# Wildcarding

- [...] allows a range of characters instead of explicitly listing each one
  - E.g. `ls [a-z]*` matches all files beginning with a-z
- The wildcard expansion is done by your shell, NOT by the program you are running
  - E.g. When `ls *` is run, it is actually doing `ls file1.txt file2.txt file3.txt ...`

# Wildcarding

- Other advanced examples to think about
  - `ls /bin/*[-_]*`
  - What's the difference between `ls *` and `ls`
  - `mv *.bat *.bit` – This does not rename all .bat files as .bit files as you might expect

# Tar



# Tar

- An archiving command
- Used to "bundle up" a directory and make a single file out of it, or "unpack" a file into a new directory
- The resulting file by convention has a .tar extension and is called a "tarball"
- Great for backups/snapshots, submitting your code in the assignments ;)



# Tar

- To create a tarball out of a directory called Assignment2, use
  - `tar cvf Assignment2.tar Assignment2`
  - `c == create, v == verbose, f == filename`
- To create Assignment2 out of a tarball, use
  - `tar xvf Assignment2.tar`
  - `x == extract, v == verbose, f == filename`

# Tar

- You might recognize this as zipping or unzipping a directory
- Technically this is not exactly the same because the tarball is not compressed/uncompressed
- If you supply `z` as in `cvzf/xvzf`, this will create an archive file and compress it. By convention, we use `.tar.gz` as the extension

# Finding files

- Use find to recursively locate files in a large directory structure
- A very powerful tool that can (among many things)
  - Match wildcards in file names
  - Match based on file size, permissions, creation time, etc.
  - Execute commands on each file found

# Finding files

- To use find, the syntax follows the format:  
find <path> <expression>  
e.g.
  - find ./ -name "README" – Find all files and directories under the current directory called "README"
  - find /usr/include -name "\*.h" – Find all files and directories under /usr/include ending in .h

# Finding files

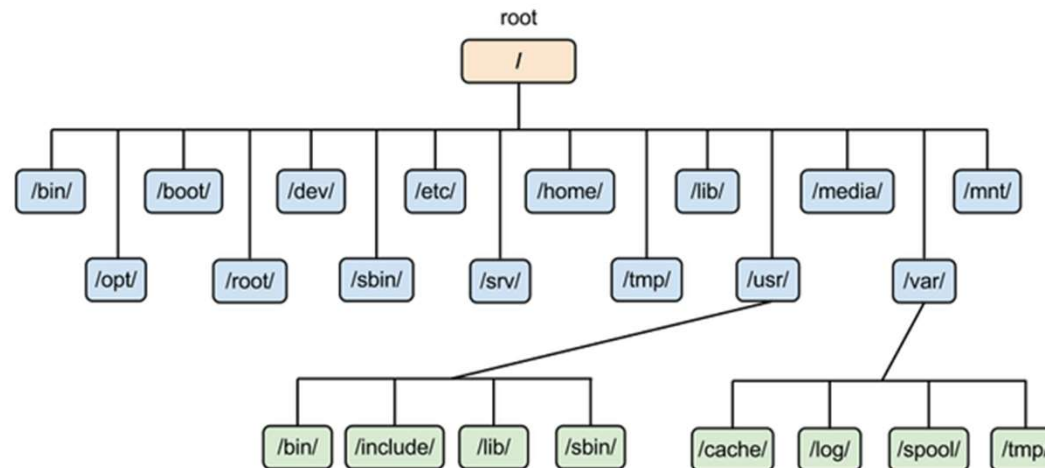
- More examples
  - `find ./ -type f -name "README"` – Find all files (but not directories) under the current directory called "README"
  - `find ./ -type d -name "README"` – Find all directories (but not files) under the current directory called "README"

# Finding files

- Use "man find" to read the manual and discover more options

# The Unix File System

- A Unix filesystem is organized like an upside down tree



# The Unix File System

- The top of the file system is known as the "root" directory and is identified with a single slash ( / )
- This is NOT the same as a backslash ( \ ) which is commonly used in Windows
- What kind of files are stored in which directory are done by convention only.



# The Unix File System

- The Unix file system can transparently span multiple disks (including network attached disks).
- As a regular user, you don't actually need to know this detail, nor should you care
- You can use the `df` command to see other disks

```
filestore.csd.uwo.ca:/data/cs_homes/wbeldman 38T 3.9T 34T 11% /home/wbeldman
```

# The Unix File System

- Some common directories in Unix

- / the root
- /bin **bin**aries (executables)
- /dev **dev**ices (peripherals)
- /devices where the **dev**ices really live
- /etc startup and control files
- /lib **lib**raries (really in /usr)
- /opt **opt**ional software packages
- /proc access to **proc**esses
- /sbin **s**tandalone **bin**aries
- /tmp place for **temp**orary files

- /usr **user** stuff
- /usr/bin **bin**aries again (user)
- /usr/include **include** files for compilers
- /usr/lib **lib**raries of functions etc.
- /usr/local **local** stuff
- /usr/local/bin local **bin**aries
- /usr/local/lib local **lib**raries
- /usr/openwin X11 stuff
- /usr/sbin sysadmin stuff
- /usr/tmp place for more **temp**orary files
- /usr/ucb **ucb** binaries
- /var **var**iable stuff

# The Unix File System

- `/bin` – contains small executable programs (binaries). This is where you find the common commands you are used to (e.g. `ls`, `cd`, `mkdir`, etc.)
- `/sbin` – contains small executable programs (binaries) but are only used by the system administrator.
- `/lib` (and `/usr/lib`) – contains binary library files that other programs might call (e.g. `stdio.h`)

# The Unix File System

- /dev – Everything in Unix is a file – even devices. This directory contains device files (see slide 8) e.g. disk drives, input devices
- /boot – The OS kernel lives here. The heart of the operating system. If the rest of the Unix system is broken, you at least need /boot to be available so the OS can boot up

# The Unix File System

- /etc – Contains system configuration files (regular files). The OS and other programs typically store their configuration information in this directory
- /proc – Virtual files that represent the current state of the kernel. Processes can refer to files here to retrieve information about the system or other processes

# The Unix File System

- /mnt – Commonly used to hold sub-directories that are temporarily mounted. This could be something like /mnt/cdrom or /mnt/windows for dual-boot systems
- /usr – Stores programs and files used by end-users. Non default stuff usually goes here. Think of this like C:\Program Files
- /var – Variable data files. Typically log files

# The Unix File System

- /home – Contains the home directories for any user with a login to the system (except root). A home directory is the user's personal space
- /root – The home directory of the "root" user

# The Unix File System

- /tmp – Contains temporary files and directories. Accessible by everyone. Many systems periodically purge this directory so DON'T store important files here!
- /opt – Some large applications will choose to bundle all their files and directories here instead of /usr, /etc, /bin, etc.



# Working with directories

- We covered making and removing directories already. Let's look at moving and copying directories

# Working with directories

- A directory is just a file, so you move it the same as you would any other file:
- `mv <directory1> <directory2>` - Moves <directory1> into <directory2>

# Working with directories

- Copying is a little different. You have to explicitly tell `cp` that you want to copy the directory and any files below it.
- Use the `-r` argument to copy recursively. E.g.
  - `cp -r <directory1> <directory2>` - Copy directory 1 and everything below it into directory 2


# Working with directories

- pushd and popd can be used to maintain a list (a stack) of directories
  - pushd <directory1> - change directory to directory1 and put directory1 on top of the stack
  - popd - change directory to the top of the stack and remove it from the stack
- The current stack is also printed for you

# Working with directories

- Use the dirs command to display the current stack

```
[wbeldman@compute ~]$ pushd /tmp
/tmp ~
[wbeldman@compute tmp]$ pushd /
/ /tmp ~
[wbeldman@compute /]$ pushd ~
~ / /tmp ~
[wbeldman@compute ~]$ dirs
~ / /tmp ~
[wbeldman@compute ~]$ popd
/ /tmp ~
[wbeldman@compute /]$ popd
/tmp ~
[wbeldman@compute tmp]$ popd
~
```



# File names

- Almost any character can be in a filename except / and the "null" character
- Still, don't use these special characters:  
? \* [ ] " ' ( ) & : ; !
- Don't use these as the first character:  
- ~
- If you do, you're gonna have a bad time!

# File names

- Unix is case sensitive. Upper and lower case are different. A.txt and a.txt are different files
- Unix doesn't use extensions.
  - a
  - a.
  - .a
  - ...
  - a.b.cAre all valid filenames

# File names

- Unix is case sensitive. Upper and lower case are different. A.txt and a.txt are different files
- Unix doesn't use extensions.
  - a
  - a.
  - .a
  - ...
  - a.b.care all valid filenames



# File names

- Extensions are still useful to the user so in practice they are still used. E.g.
  - .c means a C program
  - .jpg for a JPEG image file
  - .txt means a text file
  - .mp3 for a music file

# File names

- Executable files do not have an extension either. Windows usually uses the .exe extension.
- There are limits to the length of the name
  - Typically 255 characters for a file name and 4096 characters for the entire path

# File names

- Use the "file" command to gather info on a file

```
[wbeldman@compute Lecture-2]$ file a.out
a.out: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=14f113374b1fc6f8a009211d78f48bf3cb4f86ad, for GNU/Linux 3.2.0, not stripped
[wbeldman@compute Lecture-2]$ file pun.c
pun.c: C source, ASCII text
```

# File names

- Files beginning with a . (dot) are "hidden" files. ls will not list them by default. You have to use "ls -a"
- You have a bunch of these in your home directory already. These typically hold personal configuration files rather than storing them in /etc for all to see

# Unix Quoting

- Use quotations marks to stop the shell from interpreting special characters (e.g. whitespace, \*, or ~)
- Here's an example with "

```
[wbeldman@compute ~]$ echo Here are some words
Here are some words
[wbeldman@compute ~]$ echo "Here are some words"
Here are some words
```

# Unix Quoting

- Here's an example with ‘

```
[wbeldman@compute ~]$ echo Welcome to $HOME
Welcome to /home/wbeldman
[wbeldman@compute ~]$ echo Welcome to '$HOME'
Welcome to $HOME
```

# Unix Quoting

- You can use ` (backtick) to "insert the results of a command". E.g.

```
[wbeldman@compute Lecture-2]$ file `ls pun*`  
pun:      ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically l  
inked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=d8af98c618c  
59bdae46edec6c27f661848f4e085, for GNU/Linux 3.2.0, not stripped  
pun.c: C source, ASCII text
```

- Note that a backtick (`) and single quote (') are not the same!

# Unix Quoting

- The backslash "quotes" or "escapes" the next character (like a newline or another special character). E.g.

```
[wbeldman@compute Lecture-2]$ echo "This is a long line \  
> that continues \  
> over \  
> several lines"  
This is a long line that continues over several lines  
[wbeldman@compute Lecture-2]$ echo \  
*  
[wbeldman@compute Lecture-2]$ echo "Escaping \" character"  
Escaping " character  
[wbeldman@compute Lecture-2]$ _
```



# Miscellaneous

- A few other useful commands to know
  - `head <filename>` - View the first few lines of a file
  - `tail <filename>` - View the last few lines of a file

# Miscellaneous

- A few other useful commands to know
  - `alias <mycommand>=<another command>`

```
[wbeldman@compute Lecture-2]$ ll
-bash: ll: command not found
[wbeldman@compute Lecture-2]$ alias ll='ls -l'
[wbeldman@compute Lecture-2]$ ll
total 66
-rwx----- 1 wbeldman wbeldman 24352 Jan 11 23:22 a.out
-rwx----- 1 wbeldman wbeldman 24352 Jan 11 23:49 pun
-rwxr-x--- 1 wbeldman wbeldman  115 Jan 11 23:49 pun.c
```

# Miscellaneous

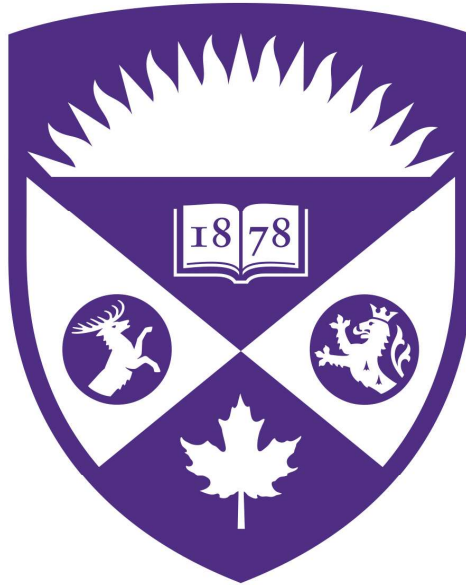
- A few other useful commands to know
  - `which <command>` - Where is a command located. The command `whereis` works similarly. This can help you find out if a command is installed or not.
  - `whatis <command>` - A one-liner description of `<command>` (This is drawn from the man page)

# Miscellaneous

- A few other useful commands to know
  - clear – clear all the text off of the screen
  - history – a running history of all the commands you have run
  - touch <filename> - Updates the "update time" on a file. If <filename> does not exist, this is a useful way to create a new empty file

# Miscellaneous

- A few other useful commands to know
  - `echo "Some Text"` – Write "Some Text" to the screen. This will be very useful when writing shell scripts
  - `grep "Some Text" <filename>` - Search for "Some Text" inside <filename>



Western  
UNIVERSITY • CANADA