C Files

Introduction to UNIX

External View of a Compiler

Compiling and Running a C program

CS-211 Operating System

- UNIX
  - Not Windows, not Mac OS, not IOS
  - Android is a flavor of UNIX
  - Linux is a flavor of UNIX
    - Linux installed on CS lab machines
- You may work in your own environment, but…
  - We don't teach anything but UNIX
  - We don't support anything but UNIX
  - We don't grade anywhere but UNIX
  - "It works on my system" is NOT a justification.

What is a “File”

Computer File

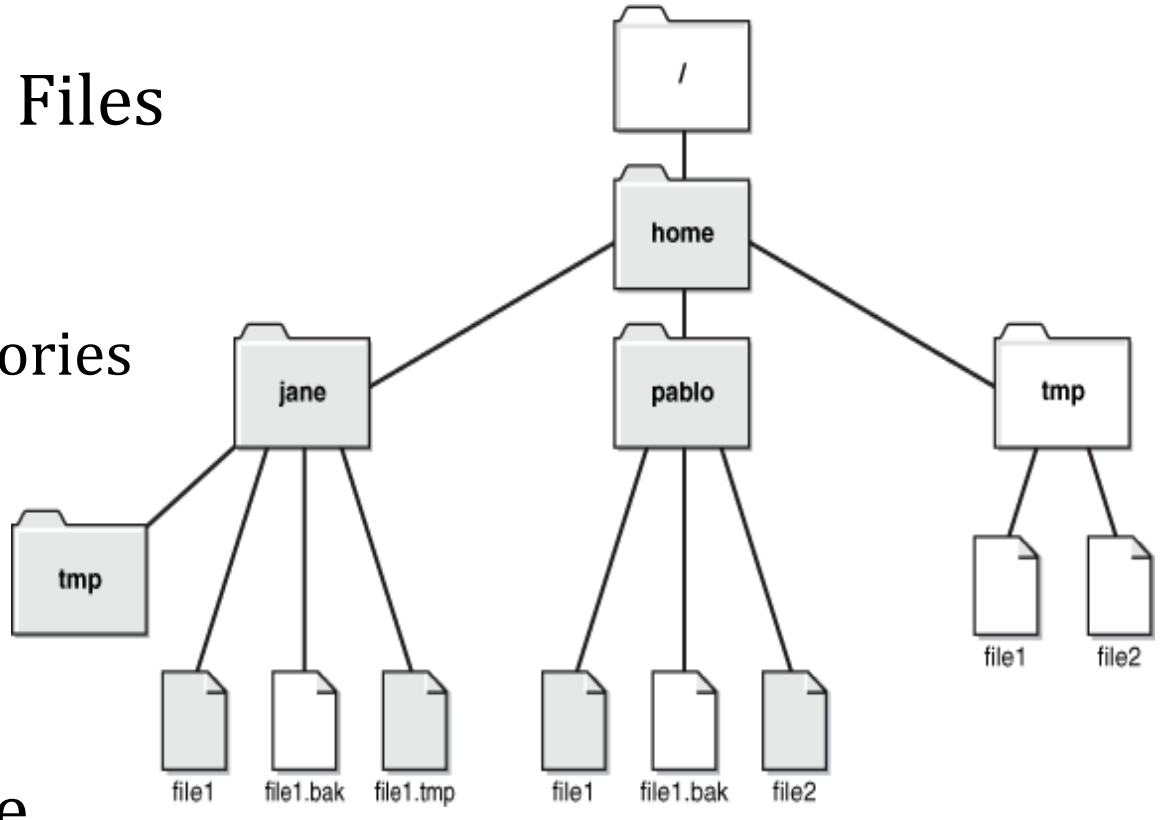
- Collection of Data
- File Name – label that references the entire collection
  - File Type – <label>.xxx
    - Part of the name that tells what program reads the data
    - .txt, .ppt, .pdf, etc.
- Kept in a File System
- File Properties
  - Attributes of the file
  - For instance, who is allowed to read or write the file
  - For instance, time/date when file was last modified

Directory

- File which contains files
- Directory Name
  - Label for files
- Directories read/written by Operating System
- Directories may contain both files and sub-directories

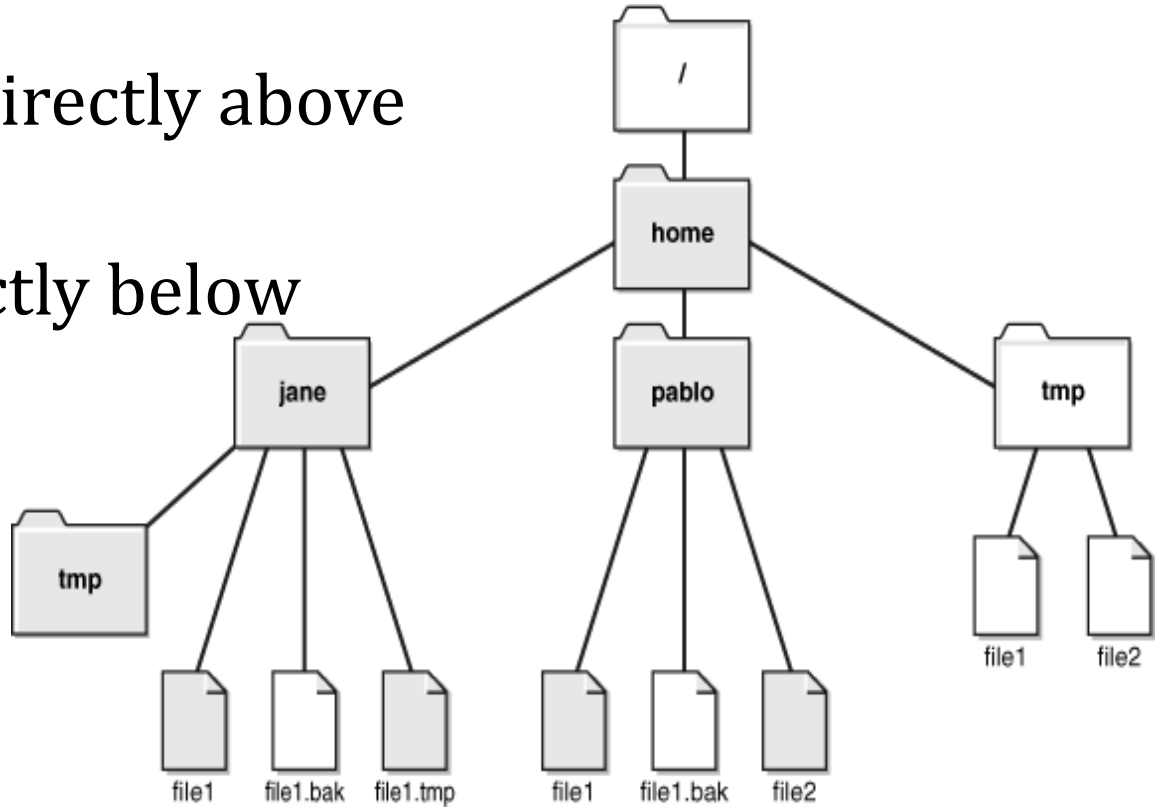
UNIX File System

- Upside down "tree" of Directories & Files
  - "/" is the "root" of the tree
  - "Branches" are directories
  - "Leaves" are data files or empty directories
- Path
  - List of directories from root to file
  - Separated by forward slashes (/)
  - e.g. "/home/pablo"
- Fully Qualified File Name : Path / file
  - .e.g "/home/pablo/file1.bak"



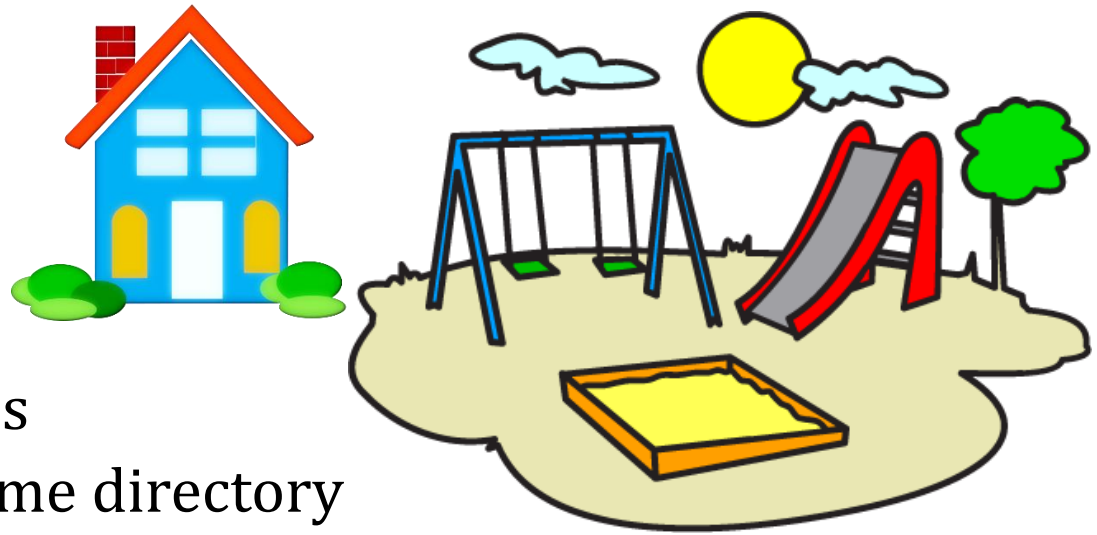
Directory Terminology

- "Parent" directory is the directory directly above
  - e.g. "/home" is the parent of "pablo"
- "Child" directory is a directory directly below
  - e.g. "jane" is a child of "/home"
  - Also called "sub-directory"



"Home Directory"

- When you start UNIX, you are in your "Home" directory
  - .e.g /home/tbarten1
- Shorthand: "~"
- You own your home directory
  - You can create or remove files
  - You can create or remove sub-directories
  - You control attributes of files in your home directory
- You may be able to read outside your home directory
- You probably cannot write outside your home directory



Current Directory

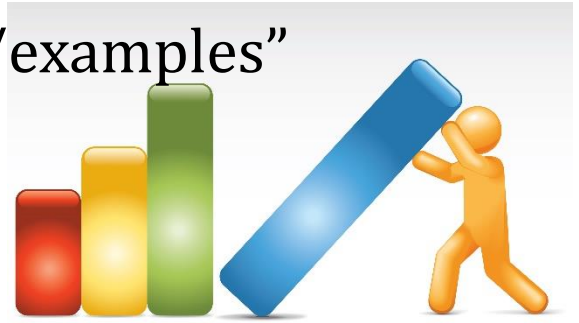
- "Current Directory" is the directory you are working in
  - Starts at your home directory
- Change the current directory with the command "cd <argument>"
  - "cd" with no argument specified, returns to your home directory
  - "cd <label> moves to the <label> subdirectory of the current directory
  - "cd .." moves to the parent of the current directory
  - "cd <fully-qualified directory>" move to the fully qualified directory
- Abbreviated as "."

Listing what's in a directory

- "ls <arguments>"
  - "ls" with no arguments lists what's in the current directory
  - "ls <label>" lists what's in the label sub-directory of the current directory
  - "ls .." lists what's in the parent directory
  - "ls <fully_qualified_path>" lists what's in the fully_qualified_path directory
  - "ls ~" lists what's in the home directory
- "ls –l <arguments>" lists file names and attributes

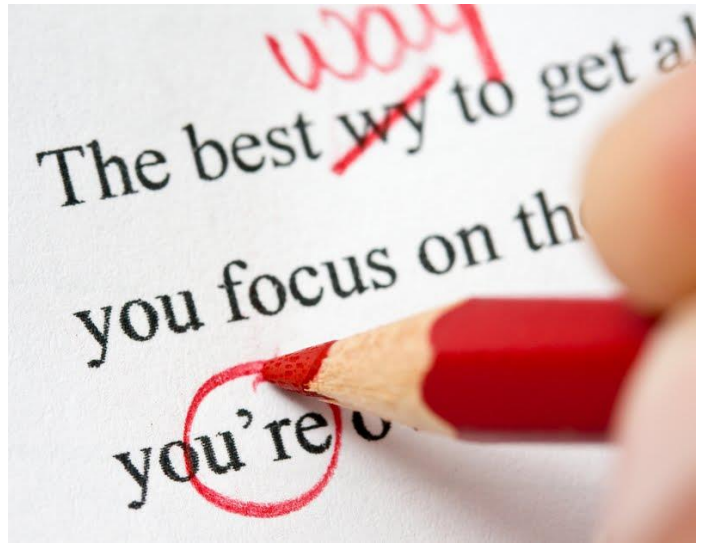
Unix Directory Commands

- Make a new directory using “mkdir <label>”
  - makes a sub-directory of your current directory
  - e.g. “mkdir examples” will create directory “/home/tbarten1/examples”
    - <label> becomes a sub-directory of the current directory
    - The “examples” directory will be empty



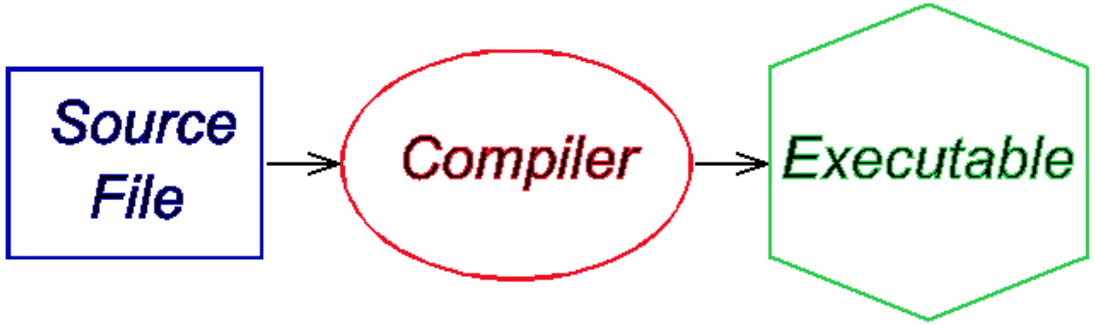
- Remove an empty directory using “rmdir <label>
  - e.g. “rm examples” removes the “examples” subdirectory of the current directory

Editing C code



- Use a file editor, "gedit"
  - Start with "gedit <file_name>&"… e.g. "gedit myfile.c"
  - <file_name> is assumed to be in the current directory
    - If there is already a file with that name present – edit it
    - If no, open an empty file. Create the file when "save" occurs
- C code in a file with type "<label>.c"
  - Can be edited by any ASCII text editor (no special stuff)

Compiling



- Why Compile?
  - "Machine Language" is laboriously low level
  - Want to run the same C code on different machines
  - Don't want to re-translate every time the program is run

- Warning: Changing C code does NOT change the executable command!
  - Must re-compile!

Compiling C Code

- Compile command: “gcc <arguments>”
- Lots of arguments, but start simple…
- Arguments end with the name of the C code file to be compiled
  - e.g. “gcc <arguments> mypgm.c”
- Output file: Executable command file
  - argument “-o <filename>” to create executable file named <filename>
  - argument “-g” to enable debugging
  - e.g. “gcc –o mypgm -g mypgm.c”
    - Compiles mypgm.c from the current directory
    - If there are no compile errors, writes executable file “mypgm” in the current directory

Compiler Errors

- When the compiler gets confused it generates a message
  - Message severity
    - Warning: Something seems wrong, but compiler can live with it
    - Error: Something is wrong that stops the compile – No output generated!
  - Messages contain the <line>.<column> that the compiler was working on when it got confused.
  - Message contains the compiler's description of the problem

```
floatx.c:32:2: warning: statement with no effect [-Wunused-value]
  result << 1;
  ^
floatx.c:71:9: error: invalid operands to binary << (have ‘double’ and ‘int’)
  result << 1;
         ^
```

Running C Code

- When you type a command, UNIX looks for an executable file with that name in a pre-defined list of directories
- Your current directory is not in that list
- Therefore, you need to specify a fully qualified name
- Shorthand is to use "." to represent the current directory
- e.g. "./mycmd" runs the executable file "mycmd" in the current directory

Resources

- Programming in C Chapter 2
- UNIX tutorial (http://www.ee.surrey.ac.uk/Teaching/Unix/)
- gedit wiki (https://wiki.gnome.org/action/show/Apps/Gedit?action=show&redirect=Gedit)
- gedit on-line manual (https://help.gnome.org/users/gedit/stable/)
- gcc on-line manual (https://gcc.gnu.org/onlinedocs/gcc-4.7.4/gcc/)