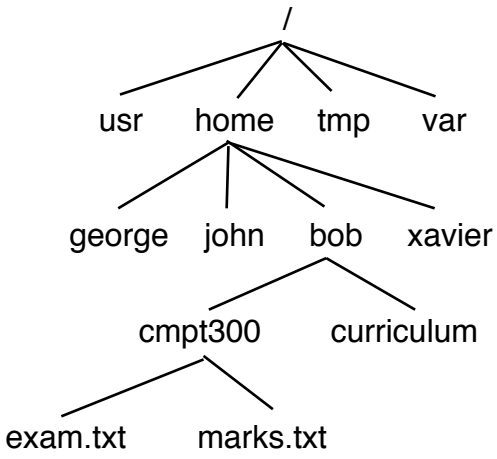
UNIX File System Structure

The Unix file system is a hierarchical structure that allows users to store information by name. At the top of the hierarchy is the root directory, which always has the name /. A typical Unix file system might look like:



The location of a file in the file system is called its path. Since the root directory is at the top of the hierarchy, all paths start from `/`. Using the sample file system illustrated above, let us determine the path for exam.txt:

- All paths start from the root directory, so the path begins with `/`
- We need to go to the `home` subdirectory, and the path becomes `/home`
- In the `home` directory, we go into the `bob` subdirectory. The path is now `/home/bob`
- In the `bob` directory, we go into the `cmpt300` subdirectory. The path is now `/home/bob/cmpt300`
- The file `exam.txt` is in the `cmpt300` directory. Appending the filename to the path, the path becomes `/home/bob/cmpt300/exam.txt`

Unix assigns a special "shorthand" notation for specifying commonly used paths. These special paths and their shorthand names are listed below:

- `/`: A single slash `/` specifies the root directory. As we have seen, all paths begin with `/` because root is at the top of the file system structure.
- `~`: A tilde `~` specifies the current user's home directory. Every Unix user has a home directory, where his/her personal files are stored. When a user logs into Unix, he/she is automatically moved to this directory. This notation is not always available in the Bourne Shell.
- `~user`: A tilde `~` followed by a user-id specifies the home directory of the given user. For example, at SFU, `~jd20` is the home directory of the user who has the user-id `jd20`. This shorthand notation is also not always available in the Bourne Shell.

- `.`: This single dot `.` specifies the current working directory, that is, the directory that the user is currently in.
- `..`: A pair of dots `..` refers to the parent of the current working directory.

Now that we know how Unix organizes files, let us attempt to define what we mean by the term file.

Files and Directories

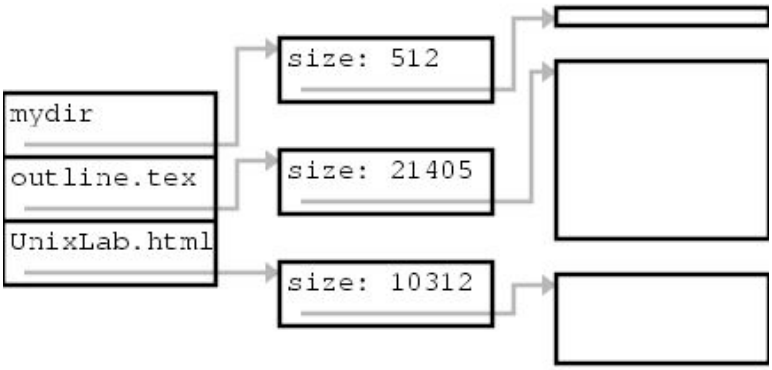
There are 3 kinds of files: ordinary files (files), directory files (directories), and special files.

- *Files*: Files are containers for data. This data can be anything, including the text of a report, an image (picture) of a house, an executable program like a word processor, or any arbitrary data. Files are created by users or programs in order to save data for future use. For example, a user might save the report she wrote using a text editor into a text file, or she might save an image from a drawing program into an image file.

- *Directories*: Each directory contains a number of files. A directory can contain other directories, be contained in another directory, or both. A directory that contains a file is called that file's parent directory. Similarly, if directory A contains directory B, then directory A is directory B's parent directory. A directory that is contained in another directory is called a subdirectory.

- *Special files*: A special file is much like an ordinary file, and shares the same basic interface. However, special files are not stored in the file system because they represent input/output devices. The I/O device provides the data directly. You will not responsible for knowing how special files work or even how to use them. Simply be aware that they are present.

Every file has an associated data structure that contains important information about that file. This data structure is known as an **i-node**. An i-node has information about the length of the file, the time the file was most recently modified or accessed, the time the i-node itself was most recently modified, the owner of the file, the group identifications and access permissions, the number of links to the file (we will discuss these shortly), and pointers to the location on the disk where the data contained in the file is stored. A filename is used to refer to an i-node. A directory is a file that contains a list of names, and for each name there is a pointer to the i-node for the file or directory. The following picture offers a graphical interpretation of files, their i-nodes, and the blocks that they occupy on disk:



Filenames

The name given to a file or directory can contain almost any character that can be typed on the keyboard and be of almost any length. It is common to use only letters and digits for the first character of a filename, and use only letters, digits, periods [`.`], hyphens [`-`] and underscores [`_`] for the remainder of the filename. However, the following characters should never be used in filenames:

- quotes [`"`], [`'`] or [`` ` ``]
- spaces
- question marks [`?`]
- asterisks [`*`]
- slashes [`/`] or [`\`]
- greater than [`>`] or less than [`<`] signs

These characters all have special meaning to Unix. Also, Unix is case sensitive. For example, the files `hello.c`, `Hello.c` and `HELLO.c` are three different filenames representing three different files in Unix.

In the Macintosh and Windows operating systems, an extension is automatically assigned to each file when it is created (an extension is a special string that is added at the end of a filename to indicate its type, and is separated from the rest of the file name by a period, such as `.doc` or `.xls`). By default, Unix filenames do not have any extension. Therefore, Unix users often add this extension themselves to indicate the type of data a file contains. Some examples are `.c` for C program files (e.g., `hello.c`), `.txt` for text files (e.g., `document.txt`), and `.ps` for PostScript documents (e.g., `outline.ps`).