UNIX file system

1. **Ordinary files**
   Ordinary files can contain text, data, or program information. Files cannot contain other files or directories. Unlike other operating systems, UNIX filenames are not broken into a name part and an extension part (although extensions are still frequently used as a means to classify files). Instead they can contain any keyboard character except for '/' and be up to 256 characters long (note however that characters such as *,?,# and & have special meaning in most shells and should not therefore be used in filenames). Putting spaces in filenames also makes them difficult to manipulate - rather use the underscore '_'.

2. **Directories**
   Directories are containers or folders that hold files, and other directories.

3. **Devices**
   To provide applications with easy access to hardware devices, UNIX allows them to be used in much the same way as ordinary files. There are two types of devices in UNIX - block-oriented devices which transfer data in blocks (e.g. hard disks) and character-oriented devices that transfer data on a byte-by-byte basis (e.g. modems and dumb terminals).

4. **Links**
   A link is a pointer to another file. There are two types of links - a hard link to a file is indistinguishable from the file itself. A soft link (or symbolic link) provides an indirect pointer or shortcut to a file. A soft link is implemented as a directory file entry containing a pathname.

William Knottenbelt (2001) http://www.doc.ic.ac.uk/~wjk/UnixIntro

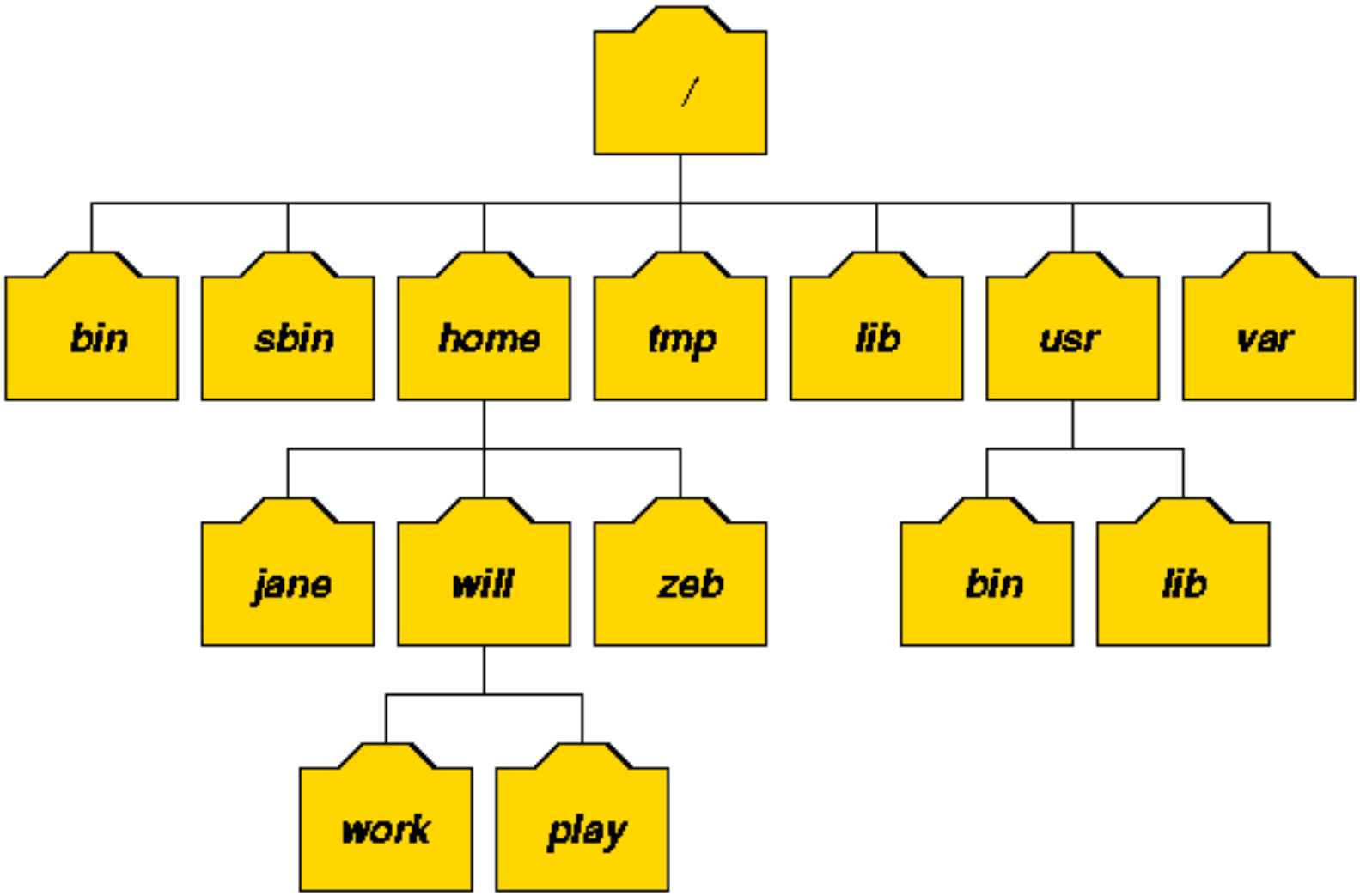


Fig. 2.1: Part of a typical UNIX filesystem tree

William Knottenbelt (2001) http://www.doc.ic.ac.uk/~wjk/UnixIntro

Unix file system

/	The "root" directory
/bin	Essential low-level system utilities
/usr/bin	Higher-level system utilities and application programs
/sbin	Superuser system utilities (for performing system administration tasks)
/lib	Program libraries (collections of system calls that can be included in programs by a compiler) for low-level system utilities
/usr/lib	Program libraries for higher-level user programs
/tmp	Temporary file storage space (can be used by any user)
/home or /homes or /Users	User home directories containing personal file space for each user. Each directory is named after the login of the user.
/etc	UNIX system configuration and information files
/dev	Hardware devices
/proc	A pseudo-filesystem which is used as an interface to the kernel.Includes a sub-directory for each active program (or process).

William Knottenbelt (2001) http://www.doc.ic.ac.uk/~wjk/UnixIntro

Commands that manipulate the file system

- `pwd` (print [current] working directory)

  `pwd` displays the full absolute path to the your current location in the filesystem. So

```
$ pwd↵
/usr/bin
```

  implies that `/usr/bin` is the current working directory.

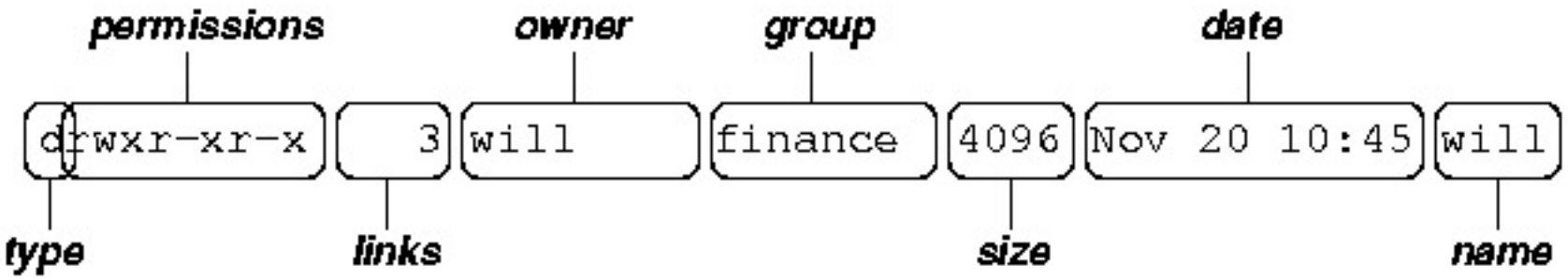
- `ls` (list directory)

  `ls` lists the contents of a directory. If no target directory is given, then the contents of the current working directory are displayed. So, if the current working directory is `/`,

```
$ ls↵
bin   dev  home  mnt   share  usr  var
boot  etc  lib   proc  sbin   tmp  vol
```

  Actually, `ls` doesn't show you *all* the entries in a directory - files and directories that begin with a dot (.) are hidden (this includes the directories '.' and '..' which are always present). The reason for this is that files that begin with a . usually contain important configuration information and should not be changed under normal circumstances. If you want to see all files, `ls` supports the `-a` option:

ls -a *and also* ls -a -l *or* ls -al *or* ls -la



- *type* is a single character which is either 'd' (directory), '-' (ordinary file), 'l' (symbolic link), 'b' (block-oriented device) or 'c' (character-oriented device).
- *permissions* is a set of characters describing access rights. There are 9 permission characters, describing 3 access types given to 3 user categories. The three access types are read ('r'), write ('w') and execute ('x'), and the three users categories are the user who owns the file, users in the group that the file belongs to and other users (the general public). An 'r', 'w' or 'x' character means the corresponding permission is present; a '-' means it is absent.
- *links* refers to the number of filesystem links pointing to the file/directory (see the discussion on hard/soft links in the next section).
- *owner* is usually the user who created the file or directory.
- *group* denotes a collection of users who are allowed to access the file according to the group access rights specified in the permissions field.
- *size* is the length of a file, or the number of bytes used by the operating system to store the list of files in a directory.
- *date* is the date when the file or directory was last modified (written to). The `-u` option display the time when the file was last accessed (read).
- *name* is the name of the file or directory.

William Knottenbelt (2001) http://www.doc.ic.ac.uk/~wjk/UnixIntro

- man (man ls)
- info (info ls)
- cd path
- mkdir *directory*
- rmdir *directory*
- cp *source-file(s) destination*
- cp -rd *source-directories destination-directory*
- mv *source destination*
- rm *target-file(s)* [try using the -i option]
- cat *target-file(s)*
- more *target-file(s)* or less *target-file(s)*

Making soft and hard links to files

Direct (hard) and indirect (soft or symbolic) links from one file or directory to another can be created using the `ln` command.

`$ ln` *filename linkname*

creates another directory entry for *filename* called *linkname* (i.e. *linkname* is a hard link). Both directory entries appear identical (and both now have a link count of 2). If either *filename* or *linkname* is modified, the change will be reflected in the other file (since they are in fact just two different directory entries pointing to the same file).

`$ ln -s` *filename linkname*

creates a shortcut called *linkname* (i.e. *linkname* is a soft link). The shortcut appears as an entry with a special type ('`l`'):

```
$ ln -s hello.txt bye.txt
$ ls -l bye.txt
lrwxrwxrwx   1 will finance 13 bye.txt -> hello.txt
$
```

The link count of the source file remains unaffected. Notice that the permission bits on a symbolic link are not used (always appearing as `rwxrwxrwx`). Instead the permissions on the link are determined by the permissions on the target (`hello.txt` in this case).

[I suggest to minimize the use of hard links]

Specifying multiple files

Multiple filenames can be specified using special pattern-matching characters. The rules are:

- '`?`' matches any single character in that position in the filename.
- '`*`' matches zero or more characters in the filename. A '`*`' on its own will match all files. '`*.*`' matches all files with containing a '`.`'.
- Characters enclosed in square brackets ('[' and ']') will match any filename that has one of those characters in that position.
- A list of comma separated strings enclosed in curly braces ("{" and "}") will be expanded as a Cartesian product with the surrounding characters.

For example:

1. `???` matches all three-character filenames.
2. `?ell?` matches any five-character filenames with '`ell`' in the middle.
3. `he*` matches any filename beginning with '`he`'.
4. `[m-z]*[a-l]` matches any filename that begins with a letter from '`m`' to '`z`' and ends in a letter from '`a`' to '`l`'.
5. `{/usr,}{/bin,/lib}/file` expands to `/usr/bin/file /usr/lib/file /bin/file` and `/lib/file`.

Note that the UNIX shell performs these expansions (including any filename matching) on a command's arguments *before* the command is executed.

Exercises

1. Try the following command sequence:
   - `cd`
   - `pwd`
   - `ls -al`
   - `cd .`
   - `pwd` (where did that get you?)
   - `cd ..`
   - `pwd`
   - `ls -al`
   - `cd ..`
   - `pwd`
   - `ls -al`
   - `cd ..`
   - `pwd` (what happens now)
   - `cd /etc`
   - `ls -al | more`
   - `cat passwd`
   - `cd -`
   - `pwd`
2. Continue to explore the filesystem tree using `cd`, `ls`, `pwd` and `cat`. Look in `/bin`, `/usr/bin`, `/sbin`, `/tmp` and `/boot`. What do you see?

3. Change to the home directory of another user directly, using `cd ~username`.
4. Change back into your home directory.
5. Make subdirectories called `work` and `play`.
6. Delete the subdirectory called `work`.
7. Copy the file `/etc/passwd` into your home directory.
8. Move it into the subdirectory `play`.
9. What is the difference between listing the contents of directory play with `ls -l` and `ls -L`?
10. What is the output of the command: `echo {con,pre}{sent,fer}{s,ed}`? Now, from your home directory, copy `/etc/passwd` and `/etc/group` into your home directory in one command given that you can only type `/etc` once.
11. Still in your home directory, copy the entire directory `play` to a directory called `work`, preserving the symbolic link.
12. Delete the `work` directory and its contents with one command. Accept no complaints or queries.
13. Experiment with the options on the `ls` command. What do the `d`, `i`, `R` and `F` options do?