IT Scripting and Automation

Unix/Linux Filesystem

Lecturer: Art Ó Coileáin

The Filesystem

- The file system is responsible for representing and organising the system's storage resources.
- The file system can be thought of as comprising four main components:
  - **A namespace**: a way to name things and organise them in a hierarchy.
  - **An API**: a set of system calls for navigating and manipulating objects.
  - **A security model**: a scheme for protecting, hiding, and sharing things.
  - **An implementation**: software to tie the logical model to the hardware.
- The file system is presented as a single unified hierarchy that starts at the directory /and continue downward through an arbitrary number of subdirectories.
- / is also called the root directory.

Pathnames

- The list of directories that must traversed to locate a particular file plus that file's filename form a pathname.
- Pathnames can be:
  - **absolute** such as '/tmp/foo' **or**
  - **relative** such as 'bookN1/topic'.
- Relative pathnames are interpreted **starting** at the current directory.
- Usually the terms: pathname and path are used interchangeably.

Limitations:

- The filesystem can be arbitrarily deep however, each component of a pathname (each directory name) must have no more than 255 characters long.
- There is also a limit on the path length, you can pass into the kernel as system call argument: 4095 bytes on Linux, 1023 bytes in older systems. To access a file with a pathname longer than this, you must cd to an intermediate directory and use relative pathnames.

The Naming of Files & Mount

- The naming of files and directories is basically unrestricted, except that:
  - names are limited in length and
  - must not contain slash characters or nulls.

Mount:

- Filesystem are attached to the ‘file tree’ with the command **mount**.
- Mount maps a directory with the existing ‘file tree’, called the mount point, to the root of the newly attached filesystem.
- **NB**: The previous contents of the mount point become inaccessible as long as another filesystem is mounted there.
- Mount points are usually empty directories.

Mount

Usage:

- $ su -c mount /dev/sda4 /users
- This command installs the filesystem stored on the disk partition represented by /dev/sda4 under the path /users.
- 'ls' command can be used to see the filesystem's content.
- To detach filesystems, the umount command is used : **umount/mnt/cdrom**

File systems mounted at Boot-time:

- A list of filesystems that are customary mounted on a particular system is kept in the **/etc/fstab** or **/etc/filesystems** file. These filesystems are automatically mounted at boot time.

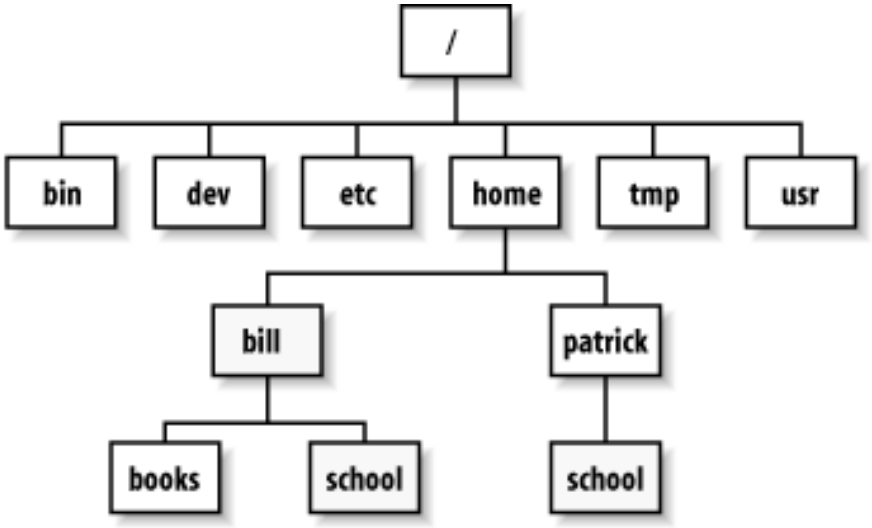
Organisation of the 'File Tree'

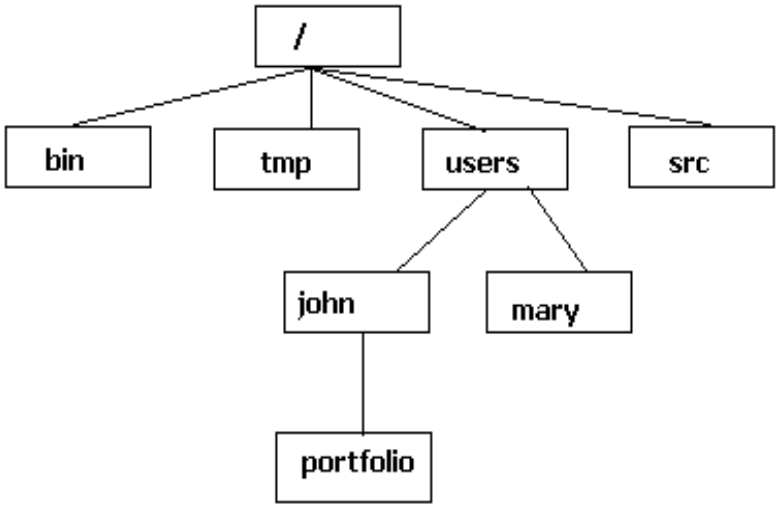
The main parts of the root filesystem are:

- **/etc** for critical systems and configuration files,
- **/sbin** holds commands needed to boot the system
- **/bin** for important utilities which are needed in single user mode
- **/tmp** for temporary files
- **/dev** contains special device files that control access to peripheral devices
- **/usr** where most of the standard programs are kept and most libraries. It is usually separate filesystem for convenience in administration
- **/lib** or /**usr/lib** keep shared library files
- **/var** for log files, accounting information, and various other items that grow or change rapidly and vary on each host. It is also a good idea to put it on its own filesystem
- **/home** or home directory of users are also kept on a separate filesystem
- **/proc** provides information about running processes and the kernel

Unix-like ‘File trees’

- On many systems, the **hier** man page (**filesystem** on Solaris) outlines some general guidelines for the layout of the filesystem.





Disk partitions

AIX	*/dev/hdisk2* (refers to the entire disk)
FreeBSD	*/dev/da0s1e* (short form: */dev/da1c*)
HP-UX	*dev/rdsk/c0t4d0*
Linux	*/dev/sdc1*
Solaris	*/dev/rdsk/c0t4d0s7*
Tru64	*/dev/rdisk/dsk2c*

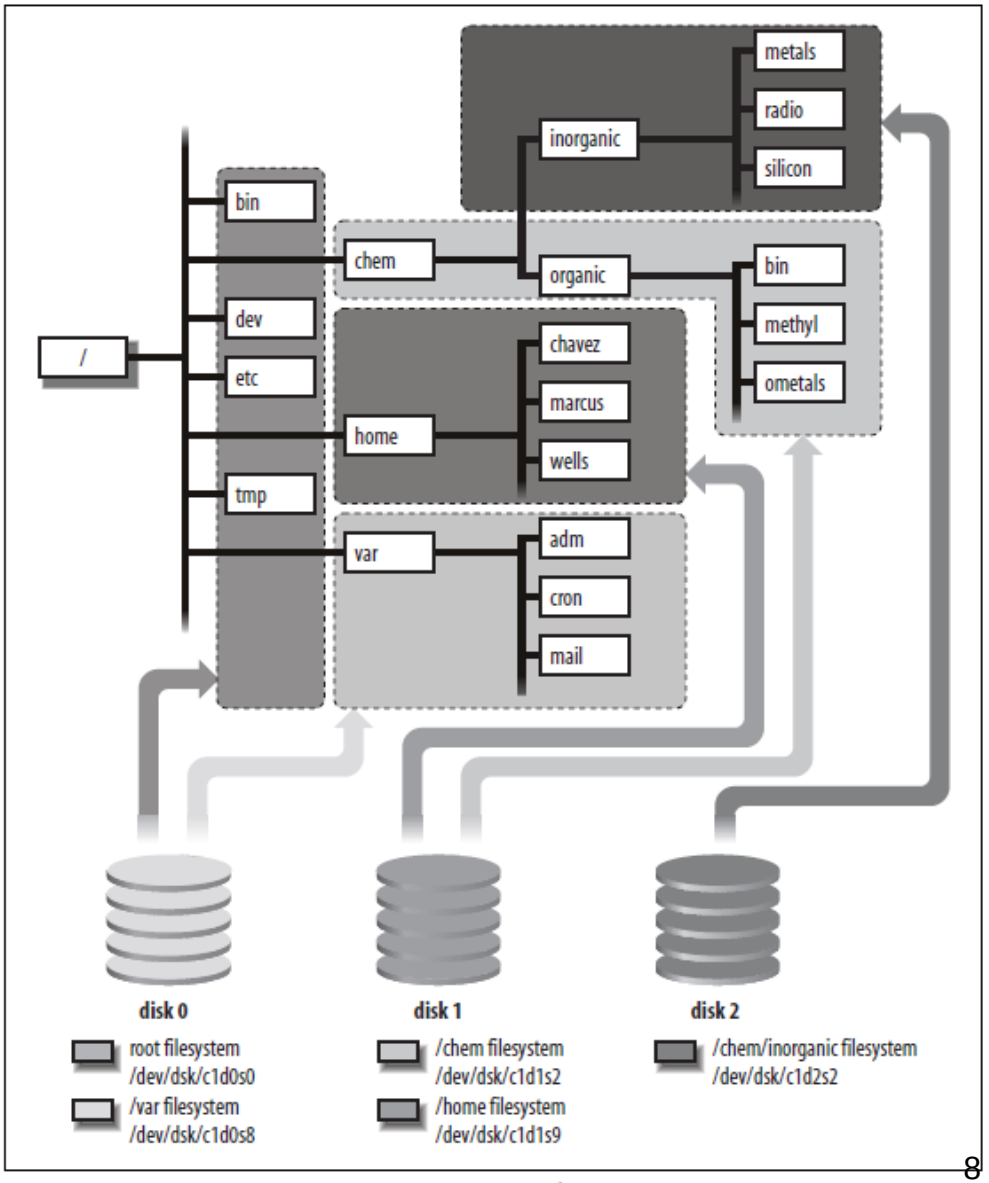
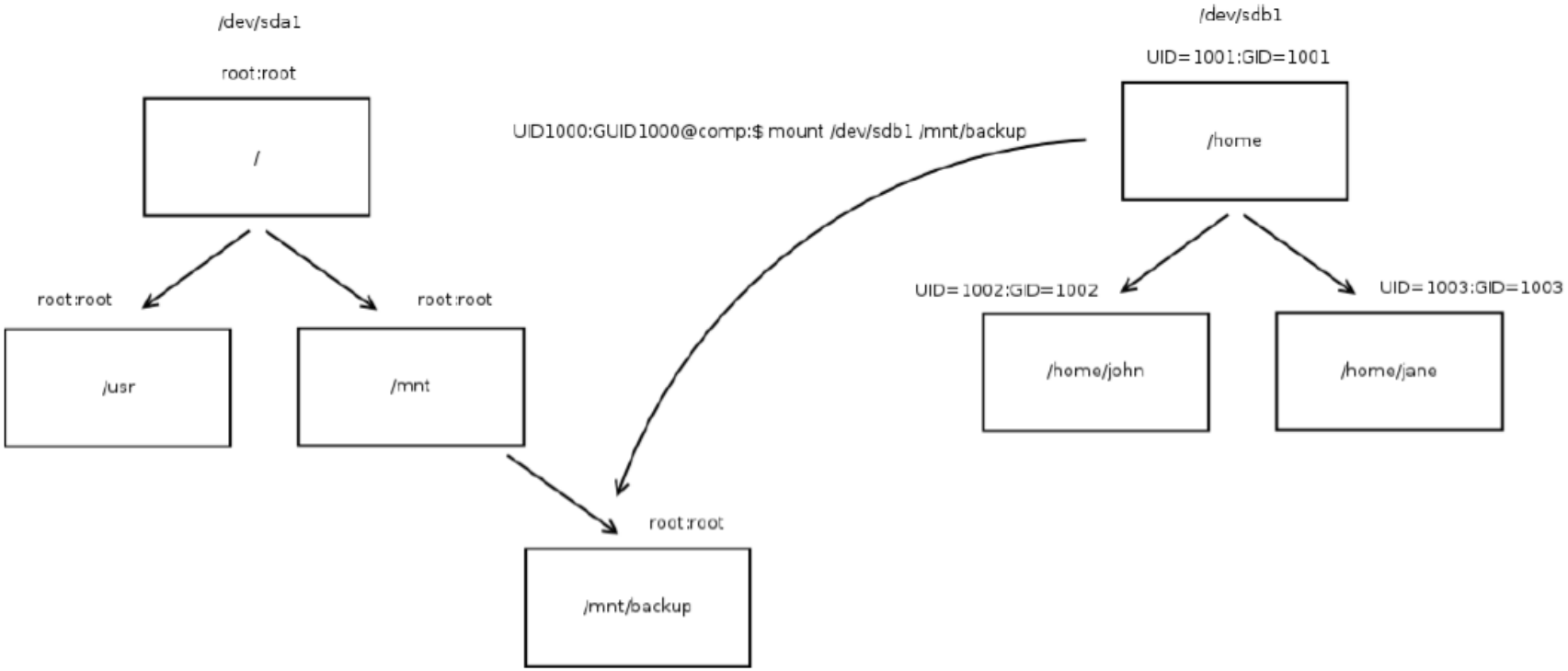


Figure 10-1. Mounting disk partitions within the Unix filesystem

Access to files on other partitions…

- Consider a system with 2 hard drives, having:
  - root filesystem at /dev/sda1 and
  - /home filesystem at /dev/sdb1.
- Now, a process with UID=1000:GID=1000 wants to mount /dev/sdb1 file system at /mnt/backup.



File Types

- The file type can be determined by using the command: ls -ld or ls -l

```
$ ls -ld /usr/include
drwxr-xr-x  27 root  root  4096 Jul 15 20:57 /usr/include
```

File-type encoding used by ls

File type	Symbol	Created by	Removed by
Regular file	-	editors, **cp**, etc.	**rm**
Directory	d	**mkdir**	**rmdir, rm -r**
Character device file	c	**mknod**	**rm**
Block device file	b	**mknod**	**rm**
Local domain socket	s	**socket**(2)	**rm**
Named pipe	p	**mknod**	**rm**
Symbolic link	l	**ln -s**	**rm**

- It is a good idea to use **rm -i file.txt** when deleting files (to confirm the deletion)

Chmod: change permissions

- **chmod 711 myprogram.sh** gives all permissions to the owner and execute only permissions to everyone else.

Permission encoding for chmod

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

Chown and chgrp: change ownership/group

- The **chown** command changes a file's ownership, and the **chgrp** command changes its group ownership.
- To change a file's group, you must either be:
  - the **owner of the file** and belong to the group you are changing to or
  - be the **superuser**.

Recursive:

- Like **chmod**, **chown** and **chgrp** offer the recursive **–R** flag to change the setting of a directory and all the files underneath it.
  - **$ su -c 'chown -R seanfilename.txt'**
  - **$ su -c 'chgrp -R students2015 filename.txt'**

- chown command can change both the owner and group of a file at one with the syntax: **chown user:groupfile** …
  - **$ su -c 'chown –R sean:students2015 filename.txt'**

Symbolic links

- A symbolic or soft link points to a file by name. When the kernel comes upon a symbolic link in the course of looking up a pathname, it redirects its attention to the pathname stored as the contents of the link.

Hard Vs Soft:

- The difference between hard links and symbolic links is that a hard link is a direct reference, whereas a symbolic link is a reference by name.

Usage:

- Symbolic links can be created with ln –s and be removed with rm.
- A symbolic link can contain either an absolute or a relative path.
  - **$su -c ln -s MyDirectory/secure.txt /tmp/secure.txt**
- It links **MyDirectory/secure.txt** to **/tmp/secure.txt** with. It creates the symbolic link **MyDirectory/secure.txt** with a target of **/tmp/secure.txt**