CS2211b

Software Tools and Systems Programming

Week 2a

UNIX File System

Announcements

TA Consulting Hours

Evan Debenham
edebenha@uwo.ca

Consulting Hour:
Thursday 4:30PM to 5:30PM
In MC244
Start on January 25th

Kun Xie
kxie5@uwo.ca

Consulting Hour:
Tuesday 4:30PM to 5:30PM
In MC244
Start on January 16th

Labs Start Today!

Week 1 Q&A on OWL

Week 1 Practice Questions

1.14 Create a directory, and then change to that directory. Next, create another directory in the new directory, and then change to that directory too. Now, run **`cd`** without any arguments followed by **`pwd`**. What do you conclude?

Week 1 Practice Questions

2.23 Can you have the same user-id more then once in the **who** output?

2.24 Both your local and remote machines use identical versions of UNIX. How do you confirm whether you are logged in to a remote machine or not?

Week 1 Practice Questions

2.26 Display the current date in the form dd/mm/yy.

How would you know how to do this?

The UNIX File System

UNIX Files

In UNIX everything is a file!

- Directories, devices and even system properties are expressed as files.
- Allows the same set of tools, utilities and APIs to be used on a wide range of resources.
- Example:

```
echo "Hello terminal 4" > /dev/pts/4
```

"Hello terminal 4" would be displayed on the screen of the user using terminal 4. You need to have permission to write to this terminal for this to work (in most cases you only have permission to write to your own terminal).

UNIX Files

Common file types are:

- **Ordinary Files:**
  - Contains data, text, etc. (regular/normal file).
- **Directory Files:**
  - Represent a directory in the file system.
  - Contain the name of files and subdirectories in the directory, as well as an “inode” number.
  - Like a folder in windows.
- **Device Files:**
  - Represents a device or peripheral in the system.
  - Acts as an interface to that device.
- **Links:**
  - A pointer to another file.
  - Like shortcuts in windows.

UNIX Files

Ordinary Files

- Text file
  - Contains human readable printable characters.
  - Lines terminated by invisible newline character \n.
  - Can see these characters using the **od** command.
  - Files encoded using ASCII or similar standard.

- Binary file
  - Contains both printable and non printable characters.
  - Not human readable.

UNIX Files

Directory Files

- Can not be written to or read directly (managed by the kernel).
- Contain the file names and unique inode numbers for each file in the directory.
- Each inode corresponds to a location on the hard disk.

Filename	Inode Number
.	384555
..	453345
myfile.txt	593234
somedir	532453

Example of directory that contains the file *myfile.txt* and the subdirectory *somedir*.

UNIX Files

Link Files

- Contain a pointer to another file.
- Two types of links:
  - **Hard:** A pointer to an inode that describes the file (including location on the hard disk).
  - **Symbolic**: A new file that contains the path of the file it is pointing to (more similar to a shortcut in windows).
- More on links when we get to the **`ln`** command.

UNIX Files

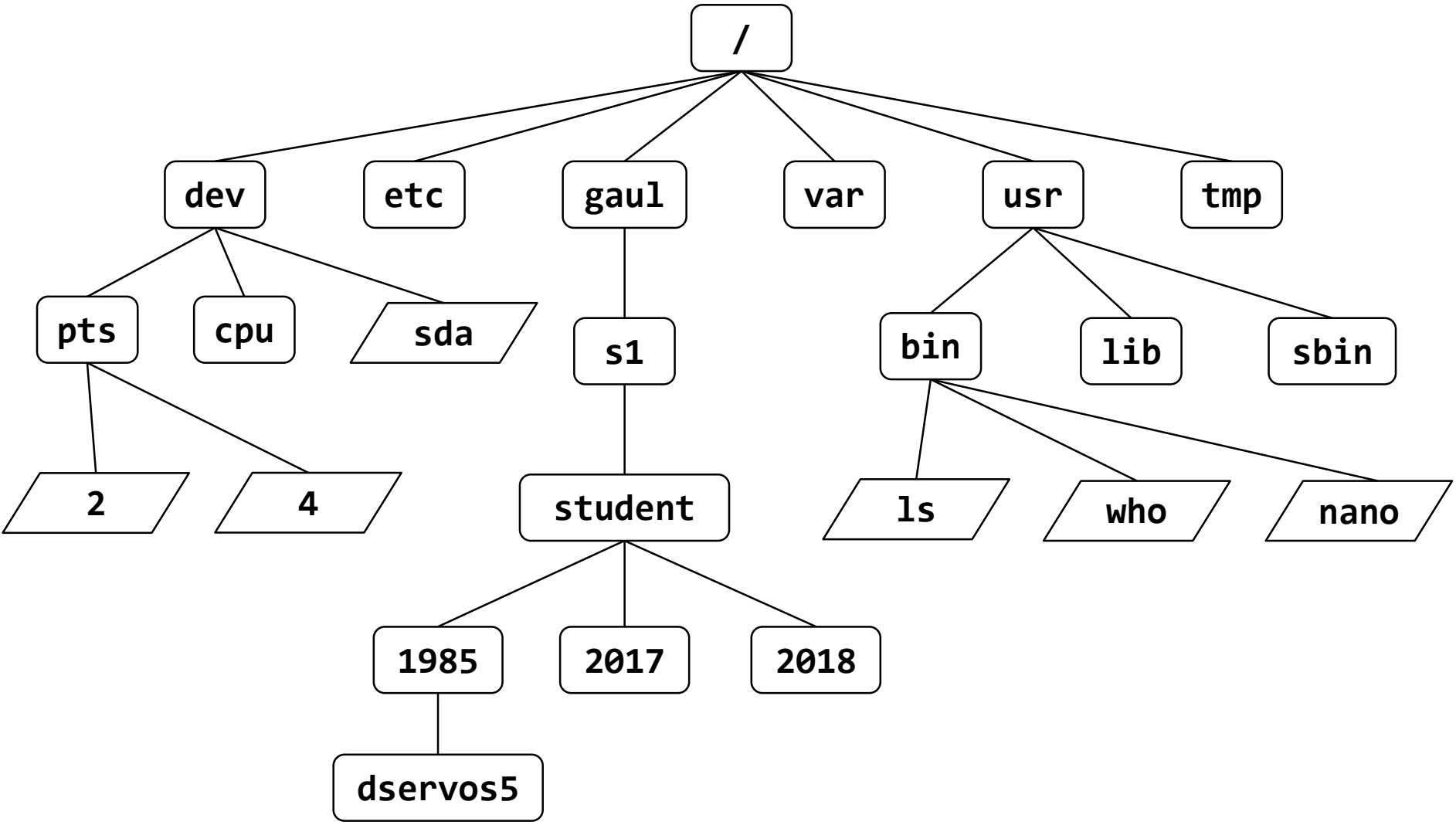
File Names

- Extensions are optional
  - E.g. can have a text file with no extension or a meaningless extension.

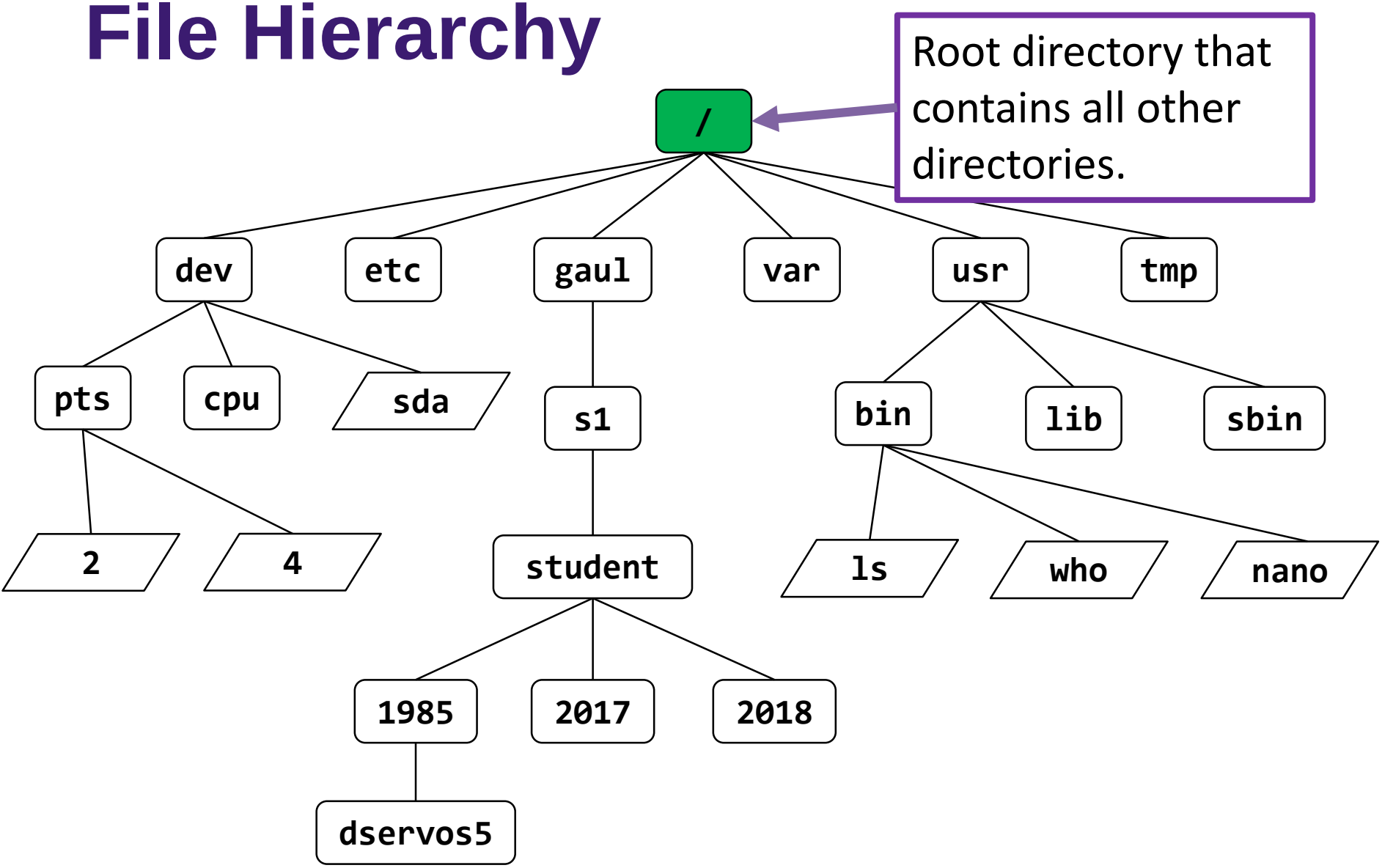
- Case sensitive
  - Can have files in the same directory with same name but different capitalization.

- '.' character at beginning of file name denotes a hidden file (will not show up in `ls`).

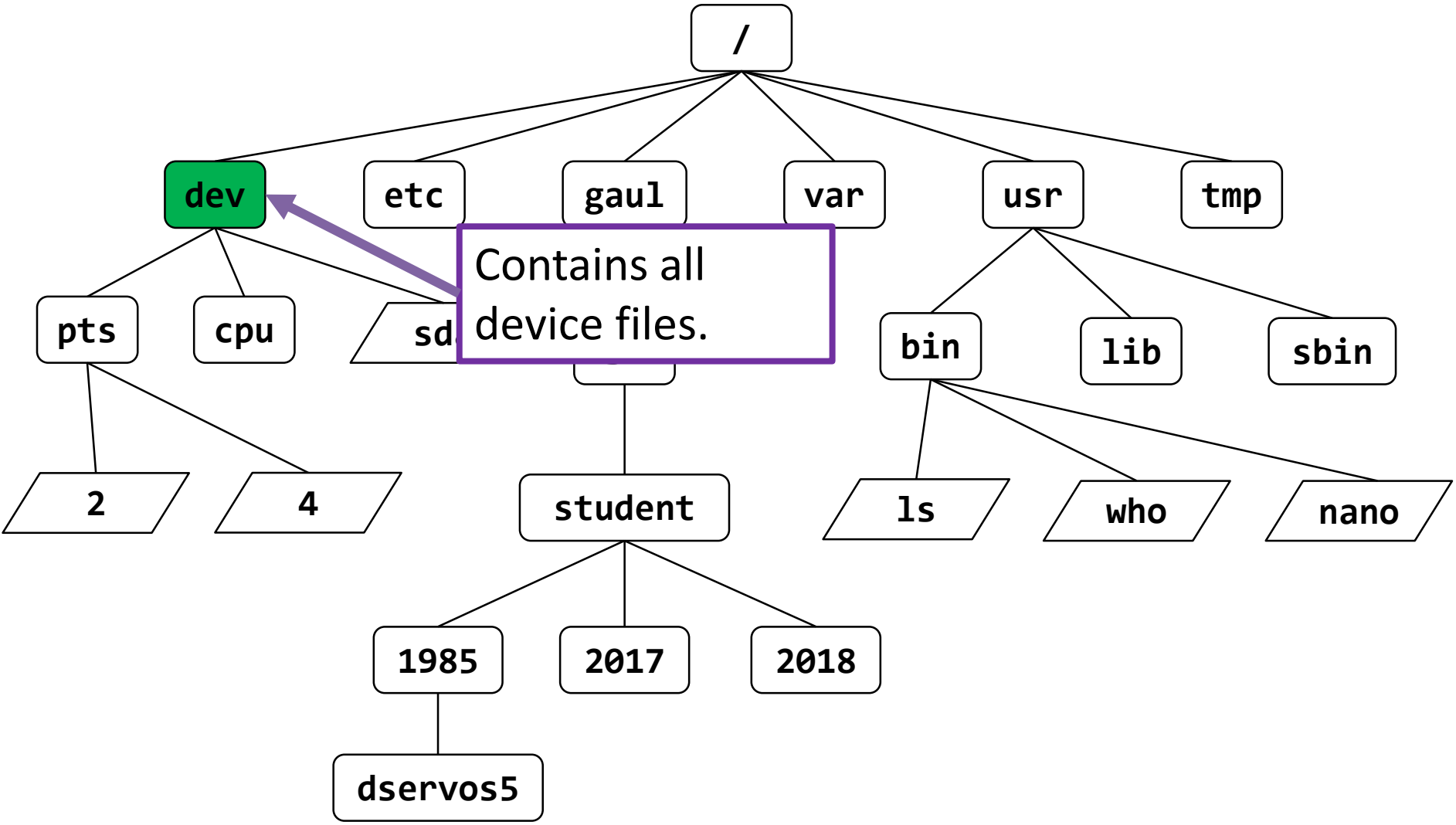
File Hierarchy



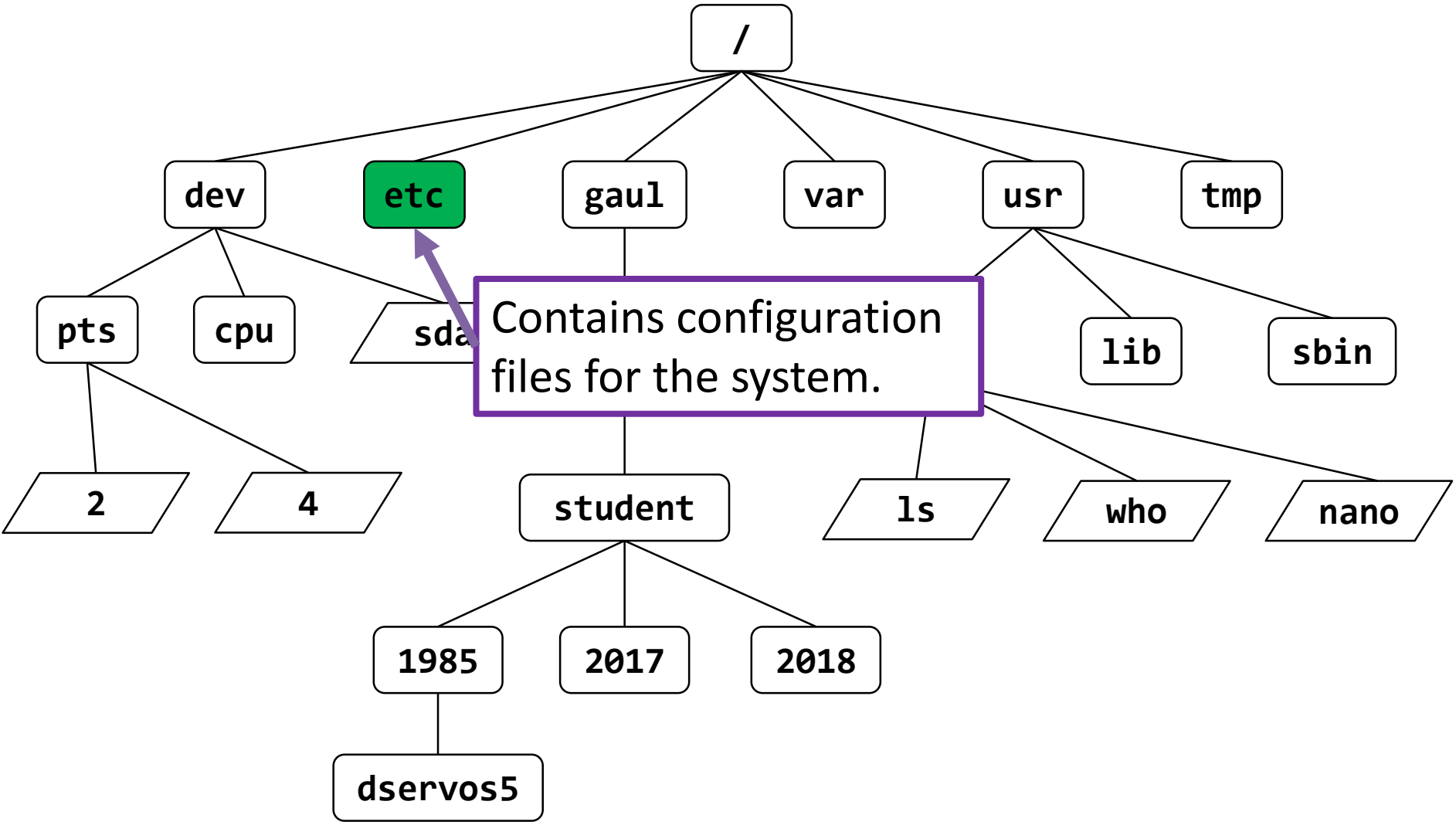
File Hierarchy



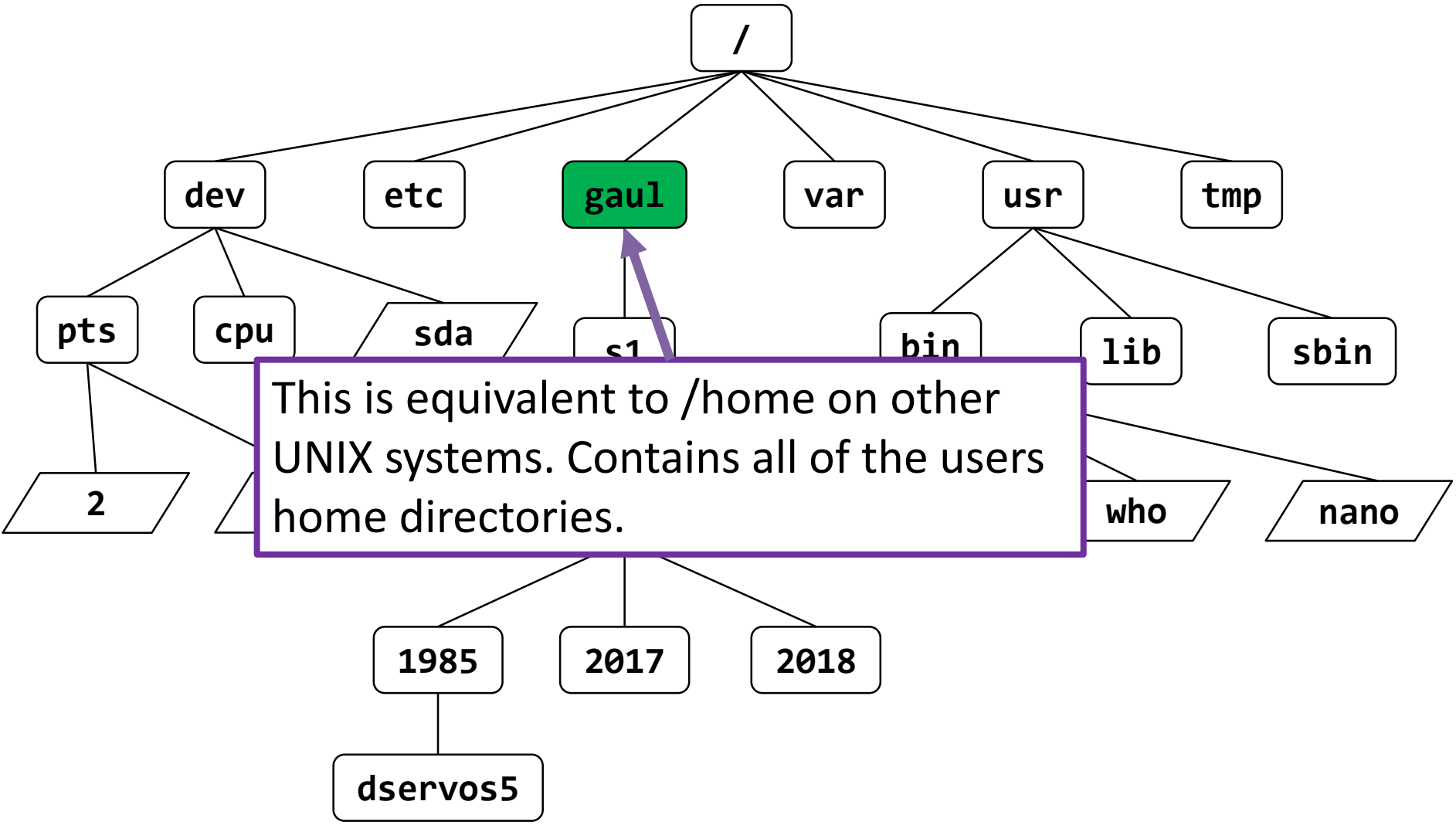
File Hierarchy



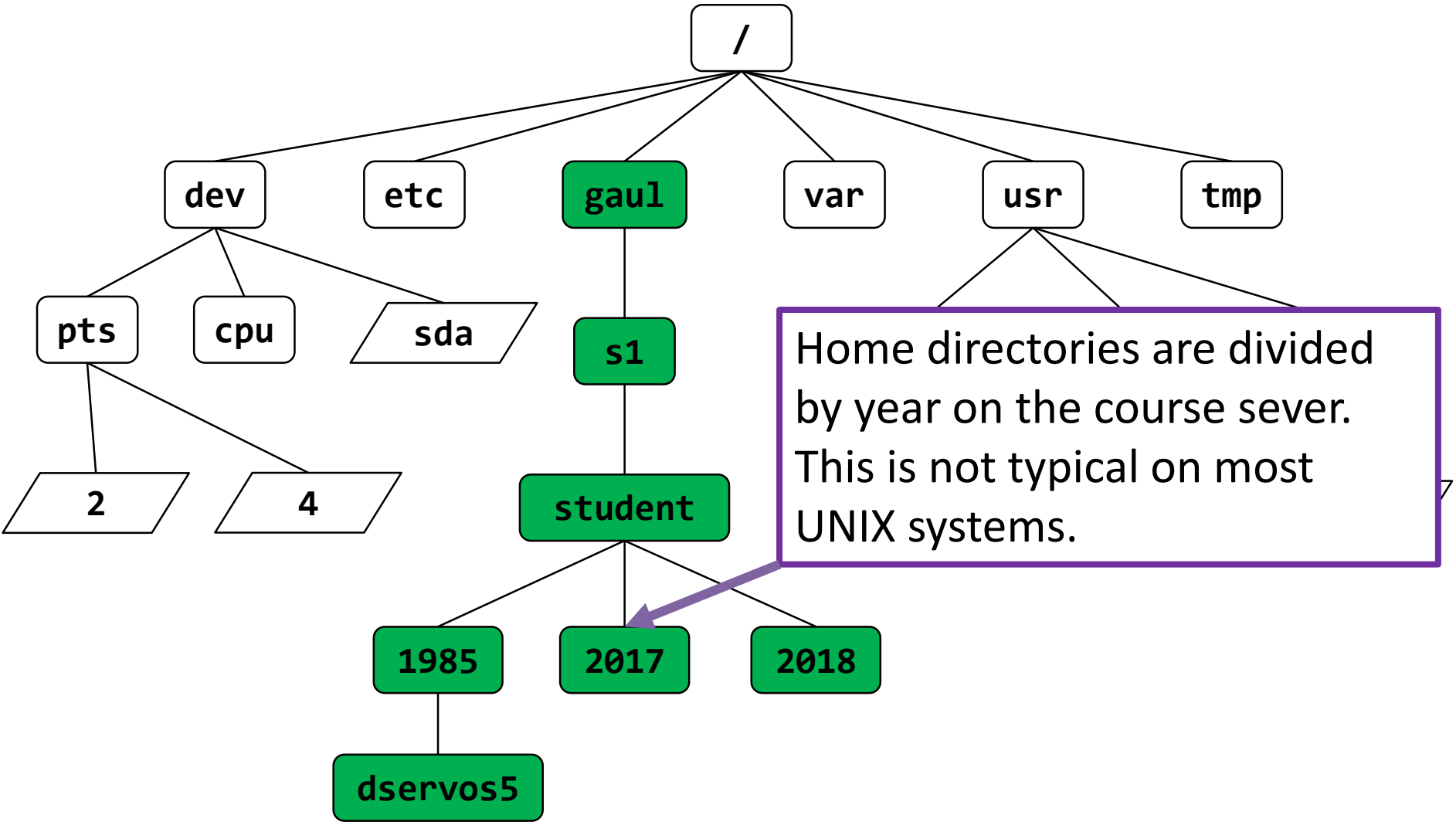
File Hierarchy



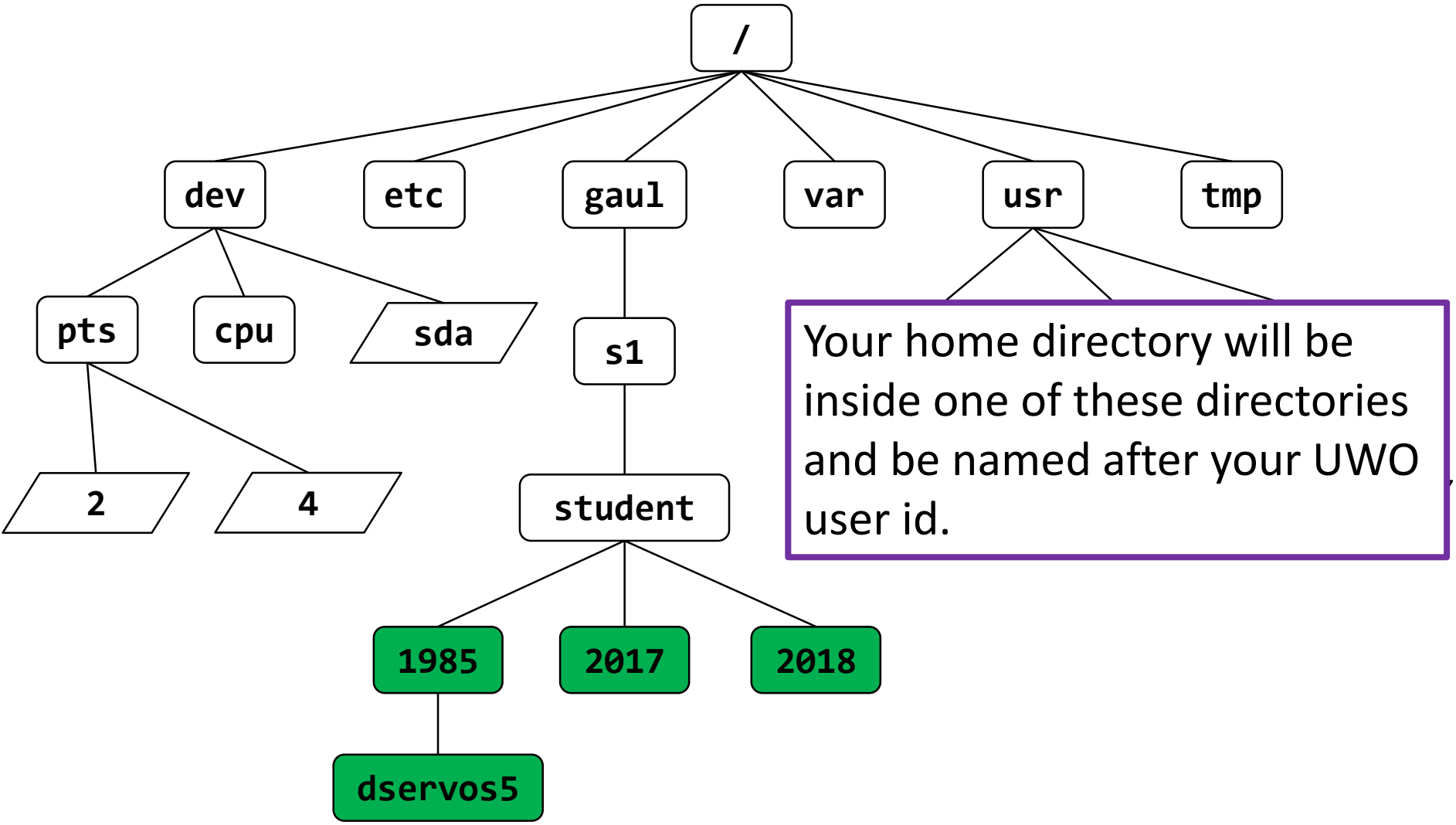
File Hierarchy



File Hierarchy

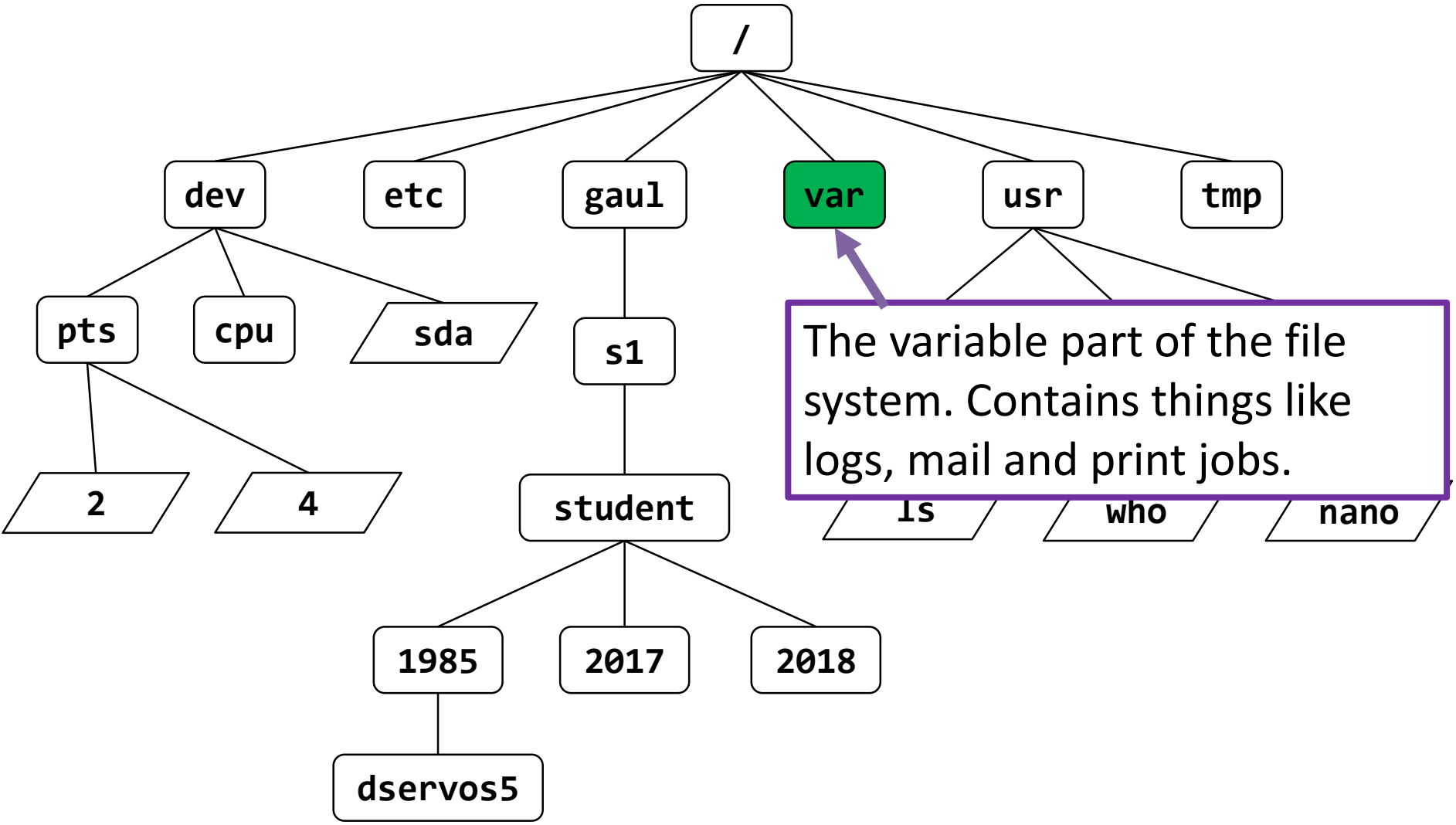


File Hierarchy

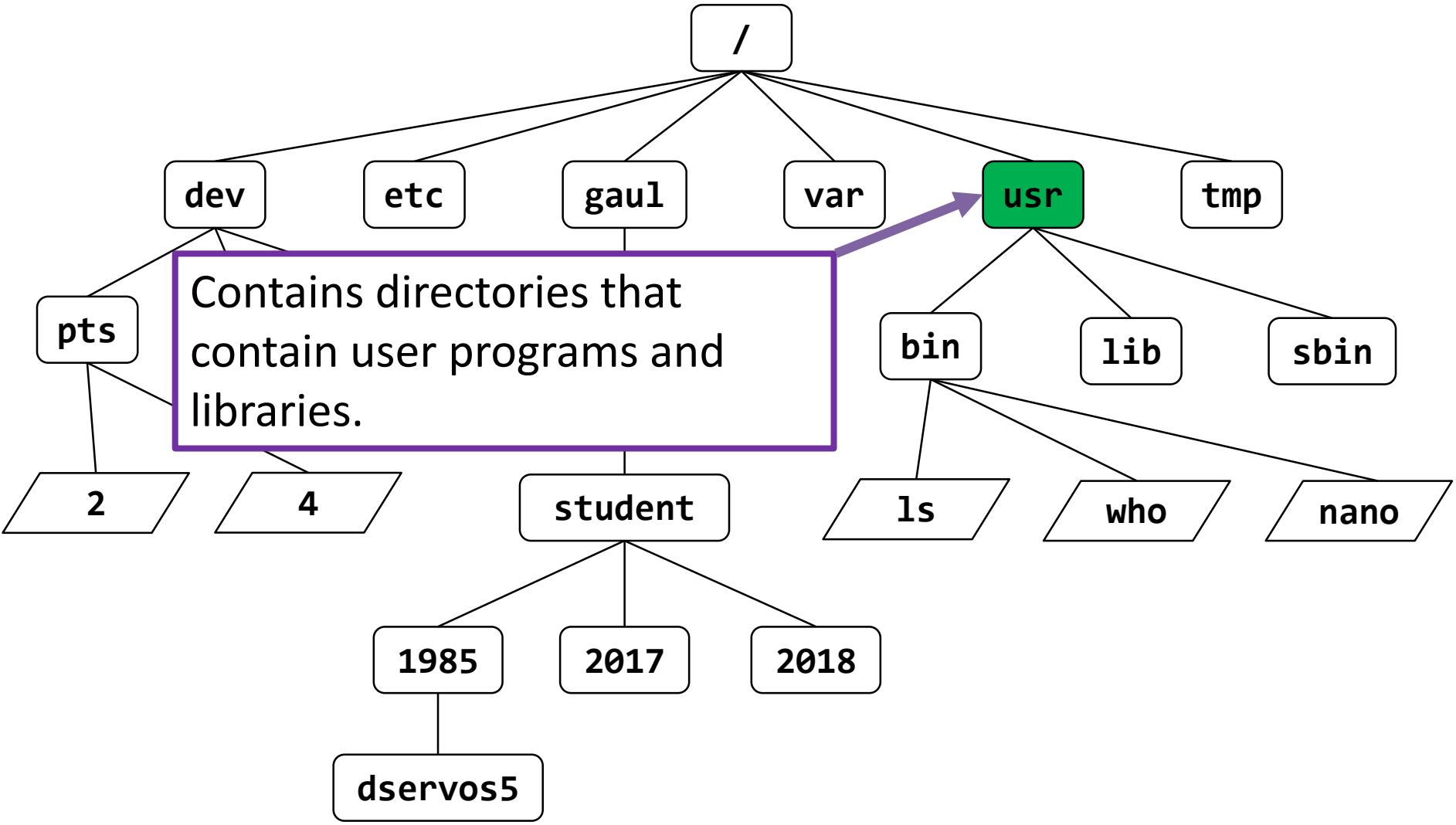


Your home directory will be inside one of these directories and be named after your UWO user id.

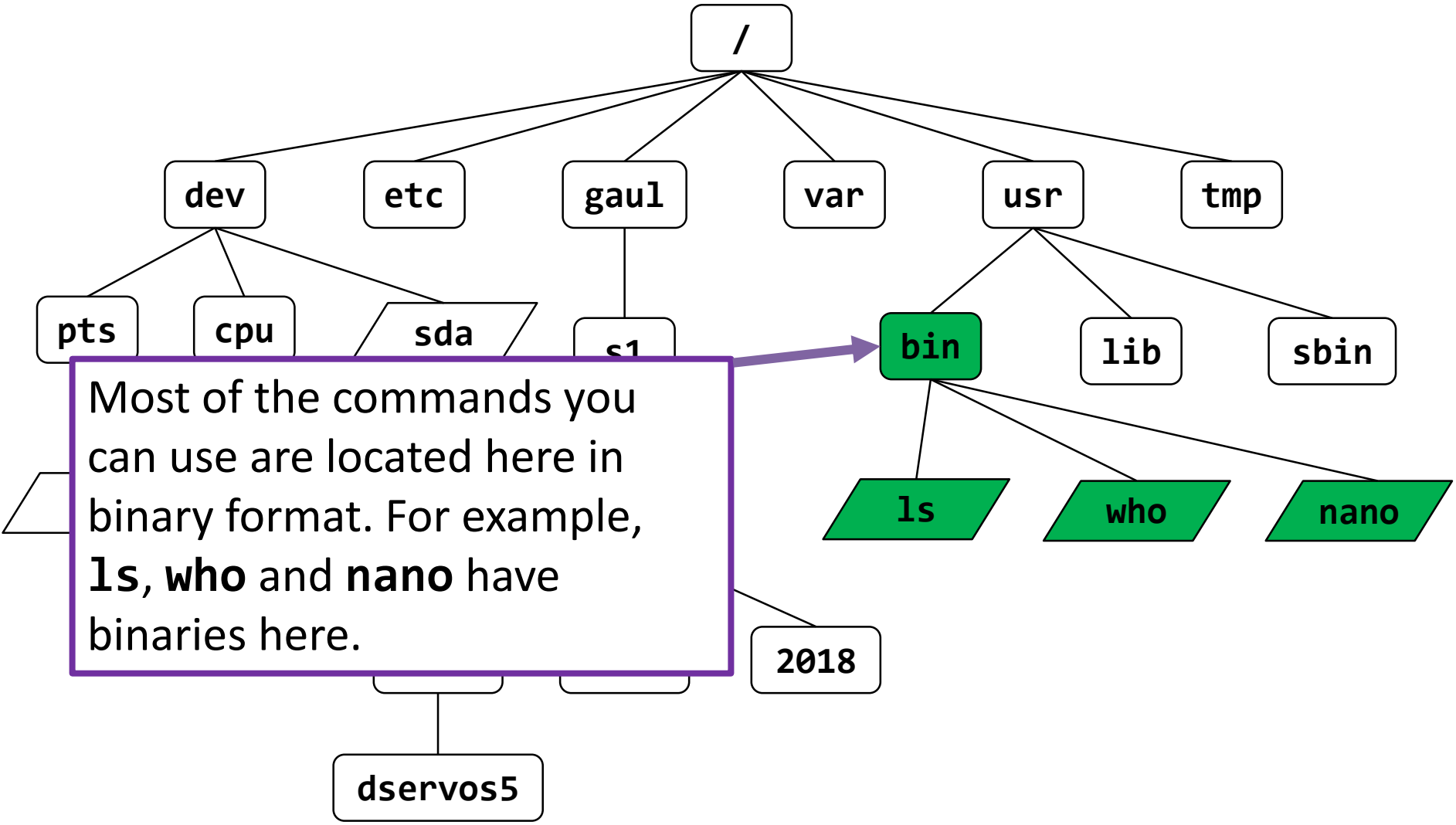
File Hierarchy



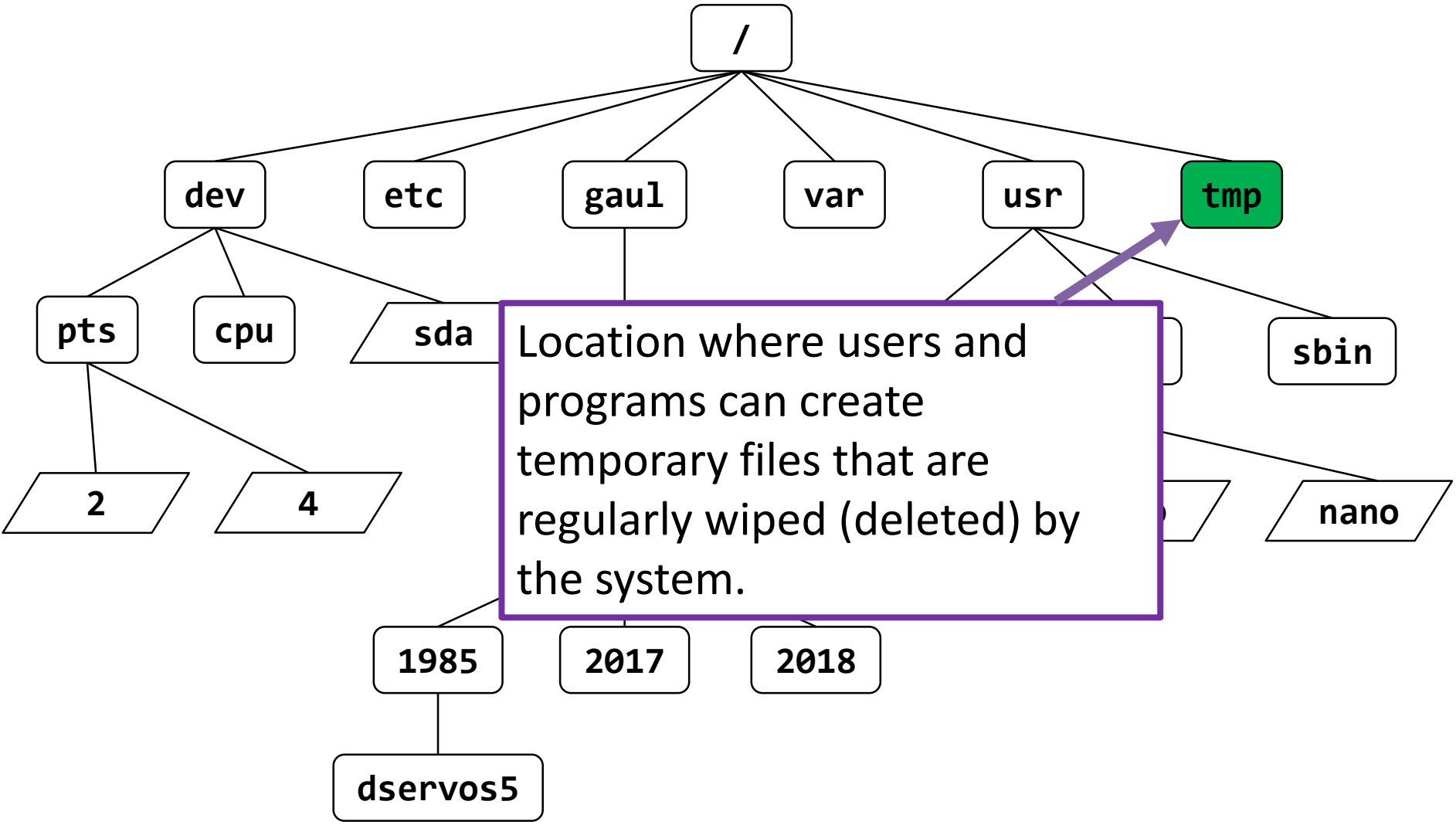
File Hierarchy



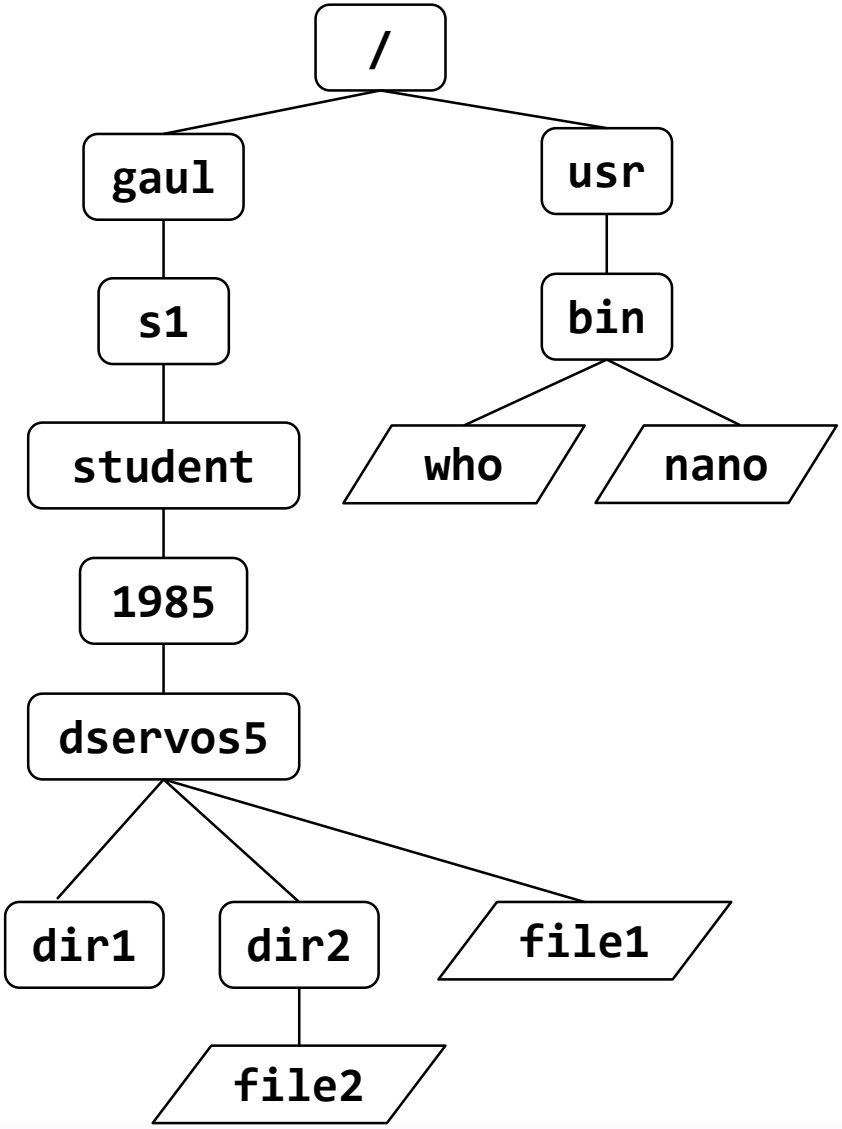
File Hierarchy



File Hierarchy



Absolute Paths

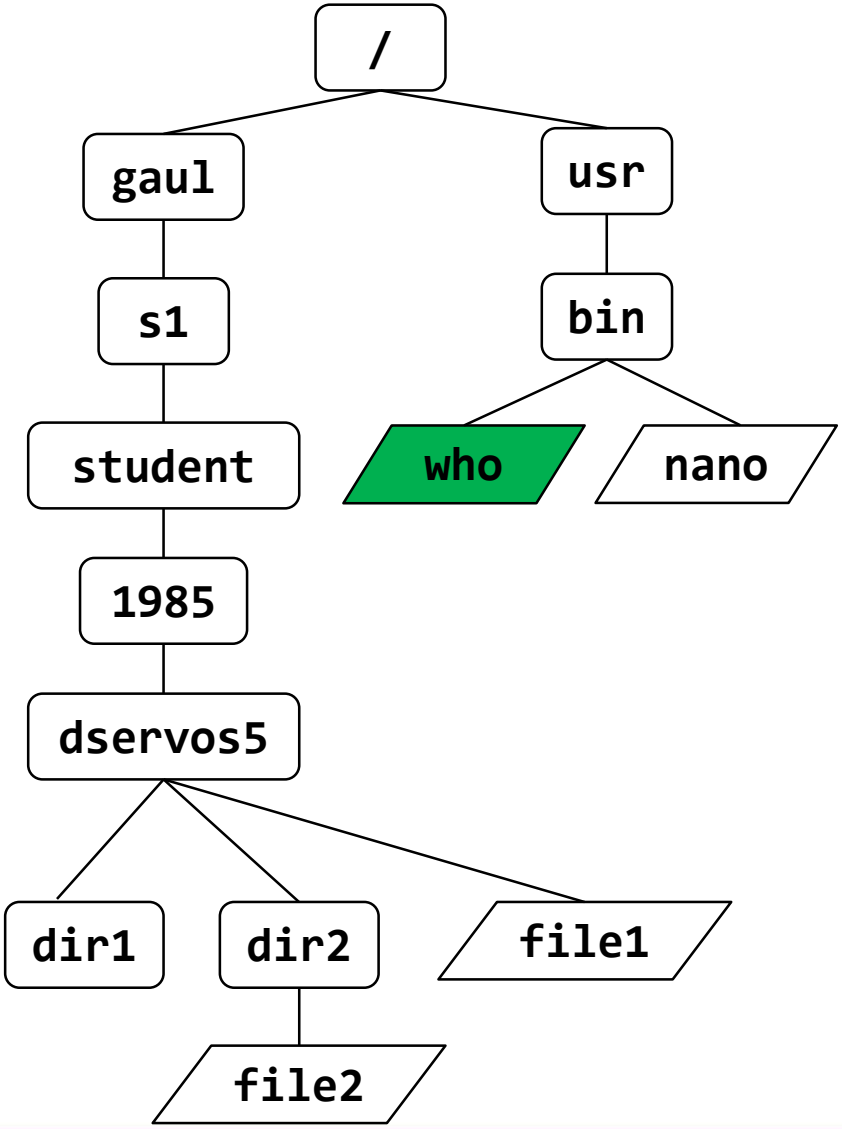


- Every file and directory has a unique absolute path that starts with the root directory /
- They can be used regardless of your current working directory.
- Can be used for both commands and files.
- Example:

```
/usr/bin/nano /var/log/dmesg
```

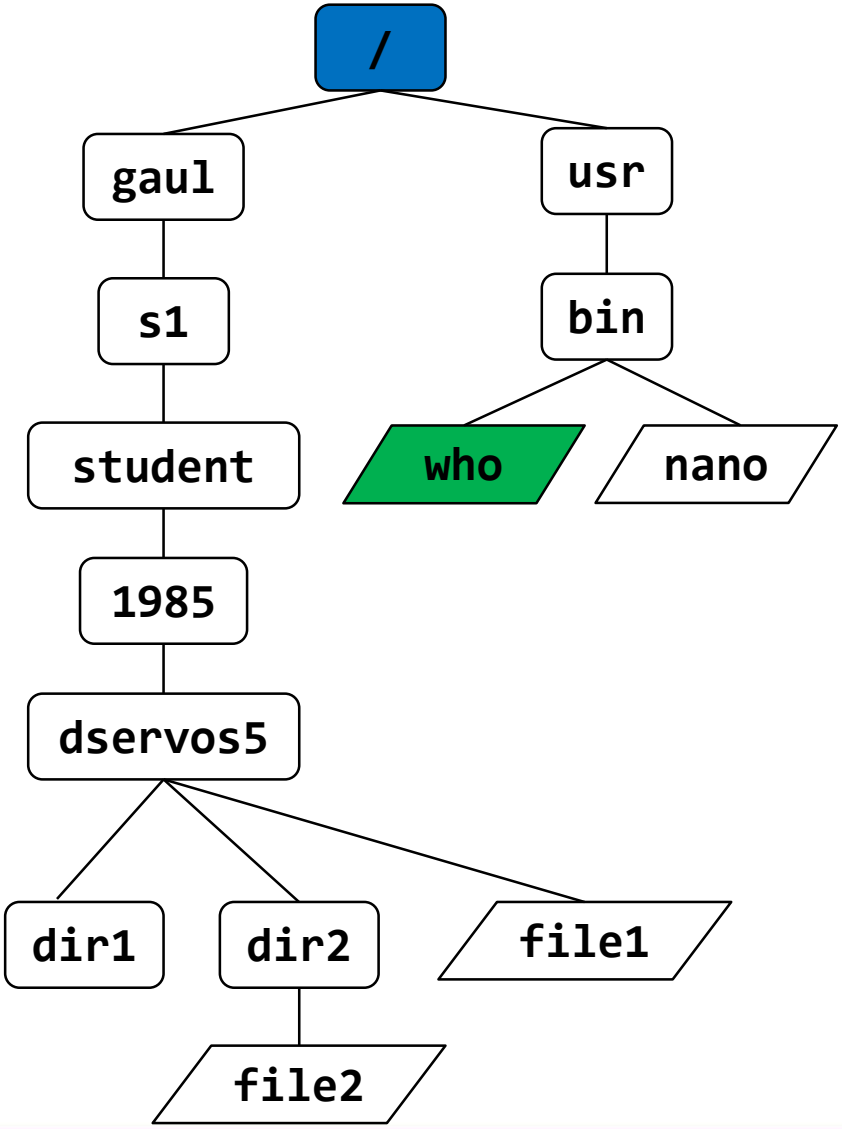
- Command does not have to be in $PATH when using an absolute path.

Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 1:** Find the absolute path for the who command.

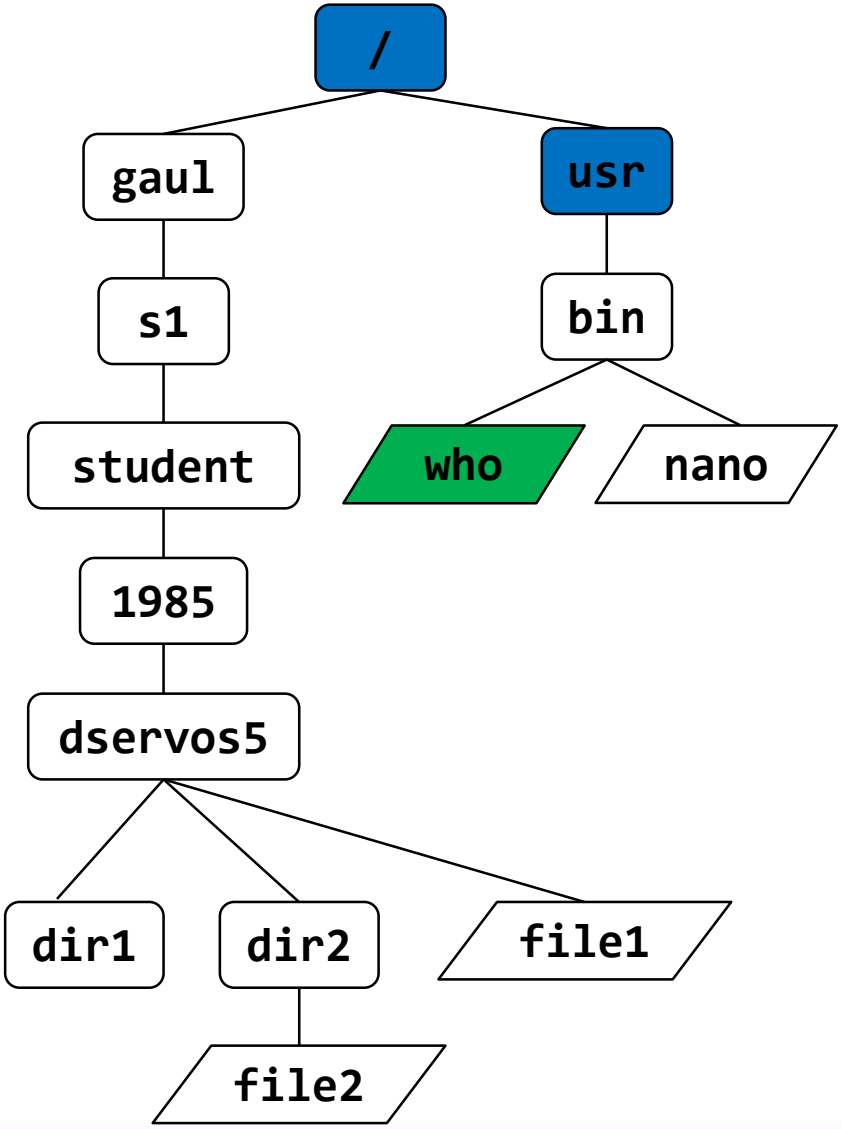
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 1:** Find the absolute path for the who command.

/

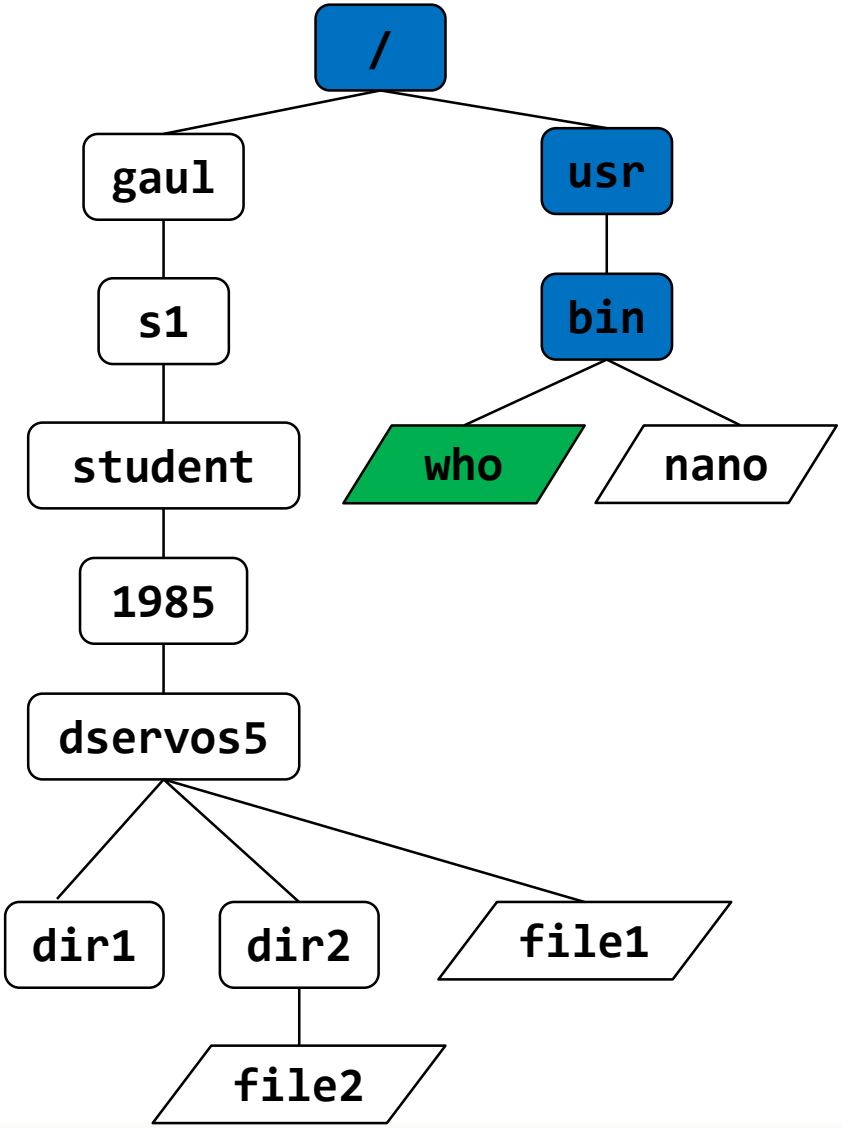
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 1:** Find the absolute path for the who command.

```
/usr
```

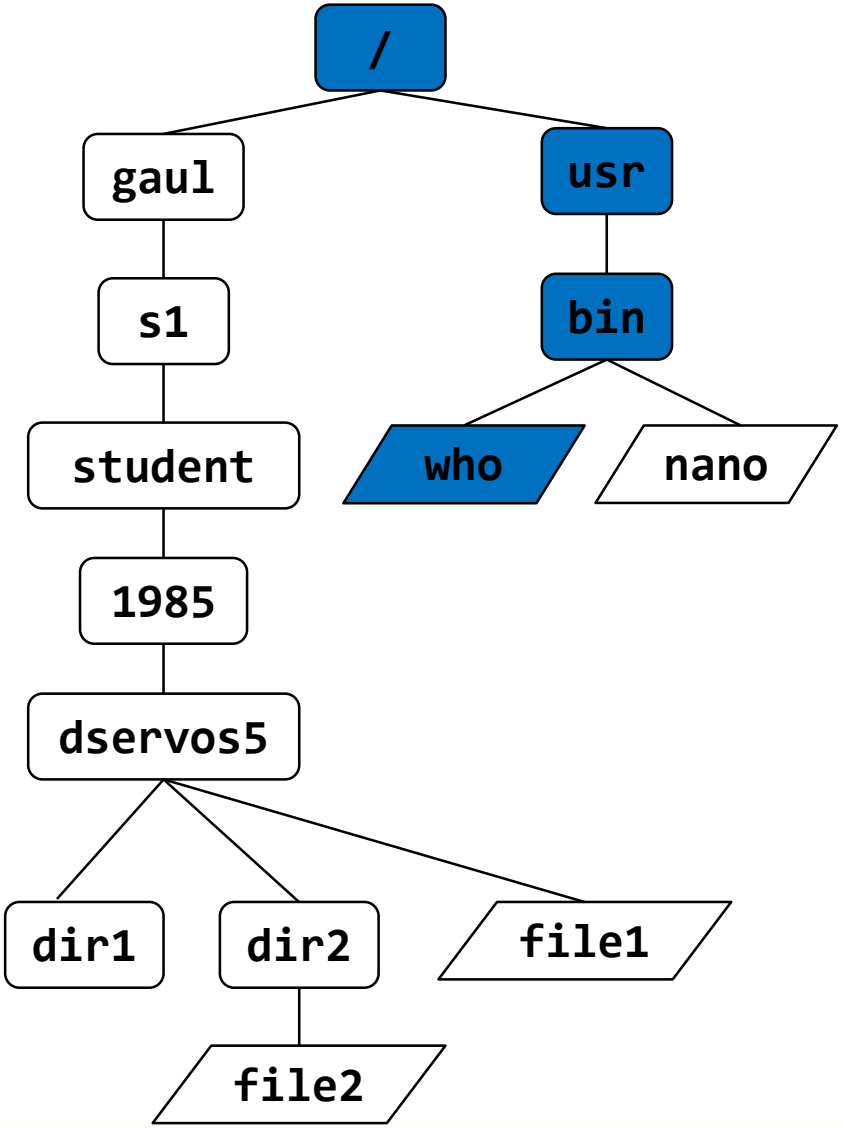
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 1:** Find the absolute path for the who command.

```
/usr/bin
```

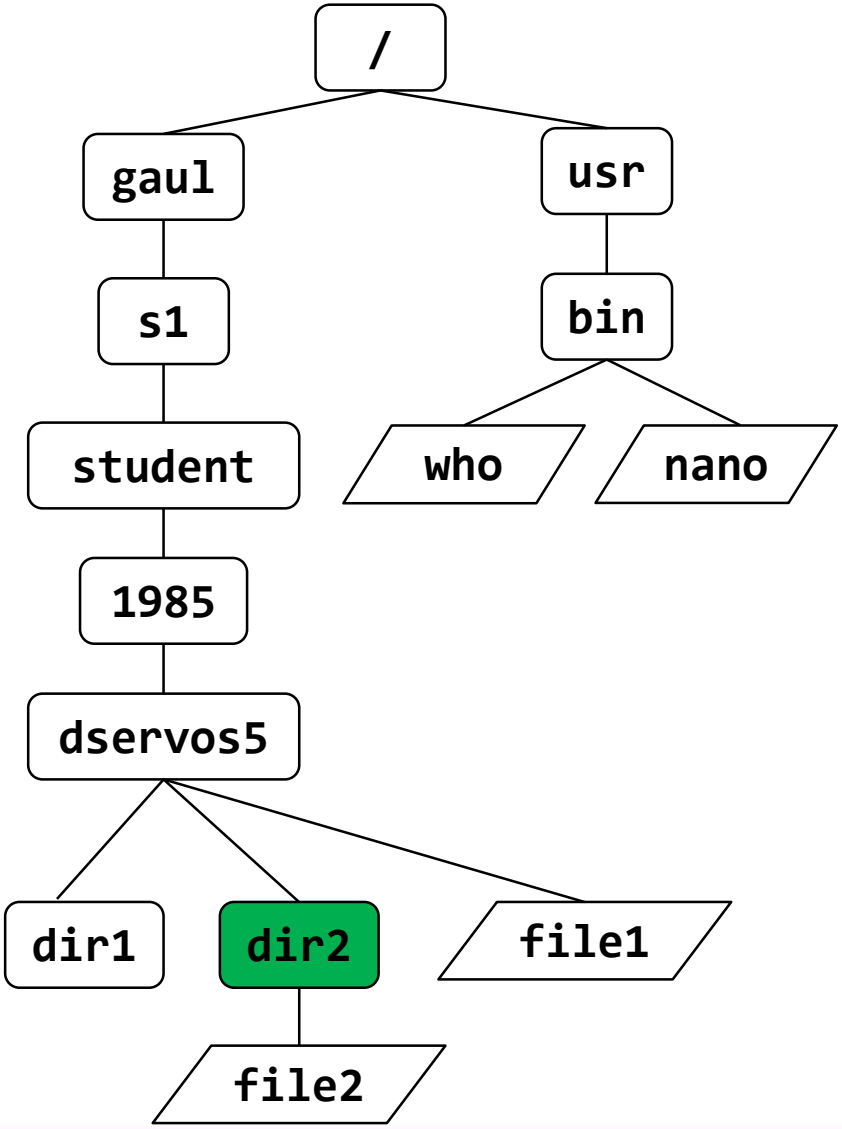
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 1:** Find the absolute path for the who command.

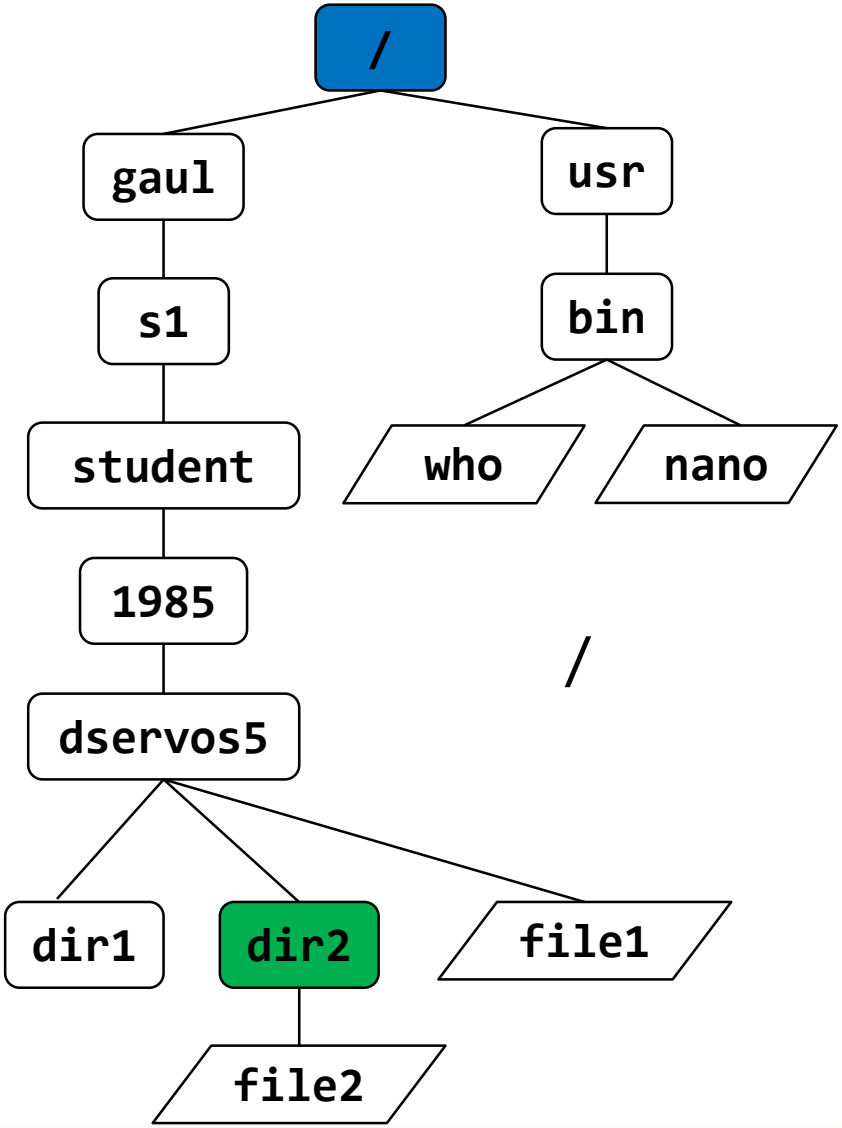
`/usr/bin/who`

Absolute Paths



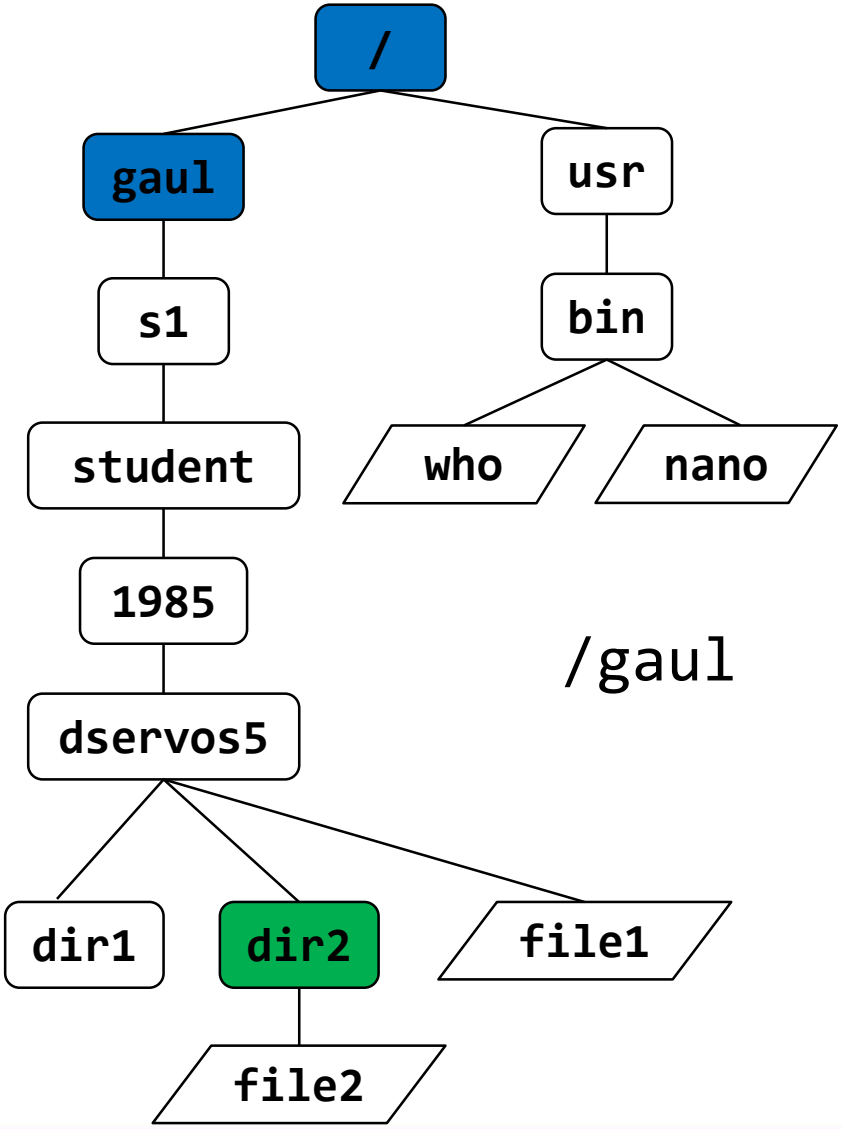
- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

Absolute Paths



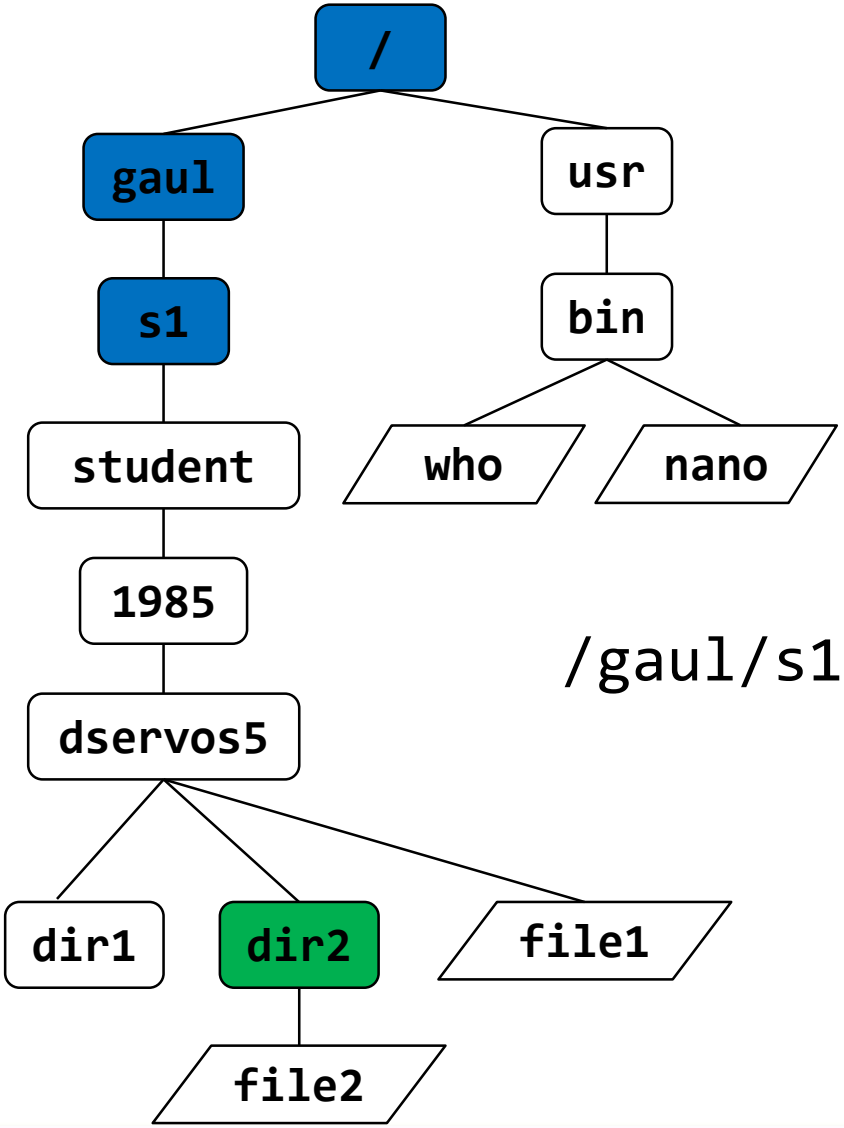
- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

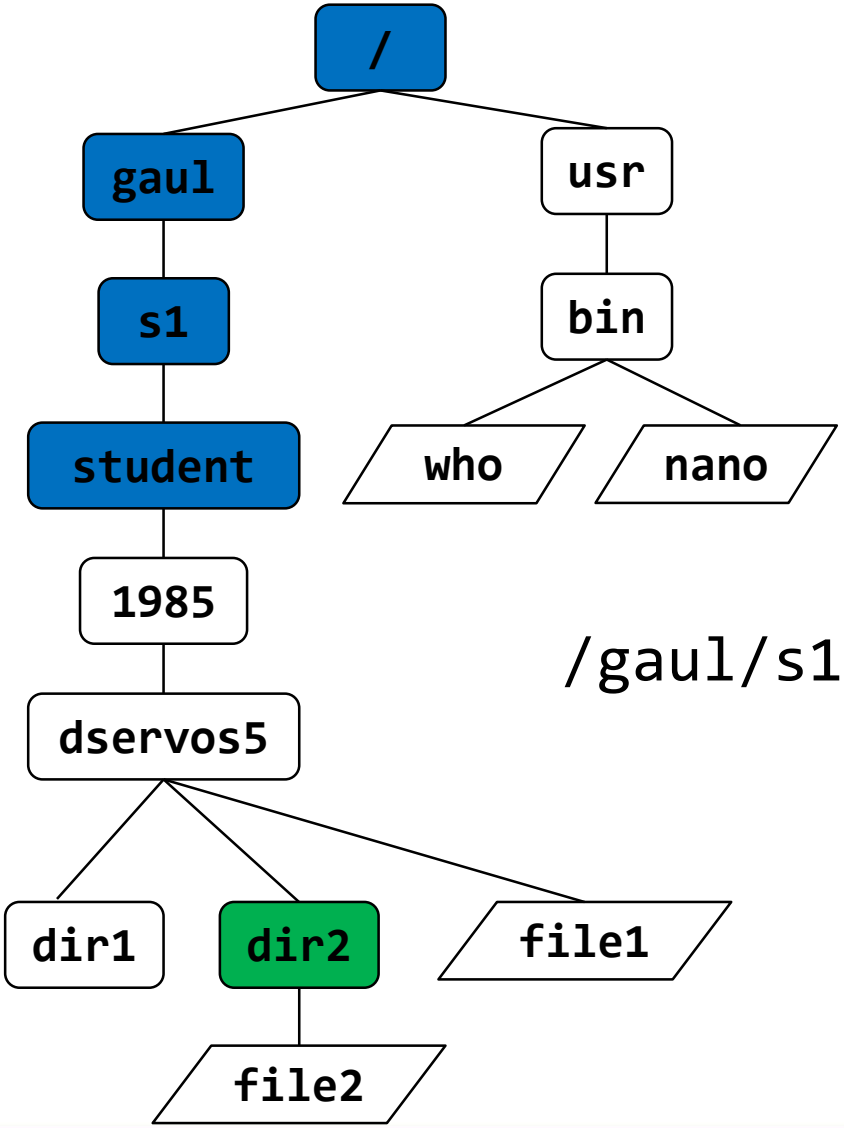
Absolute Paths



/gaul/s1

- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

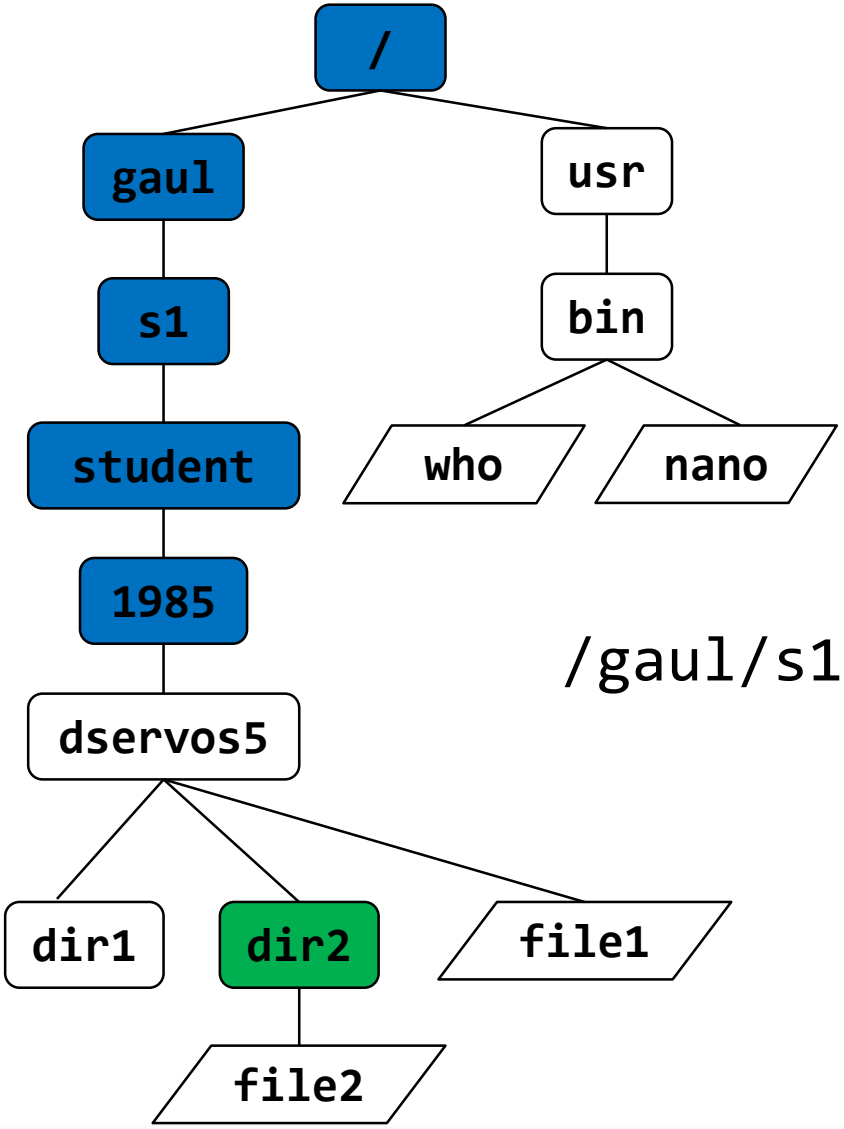
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

```
/gaul/s1/student
```

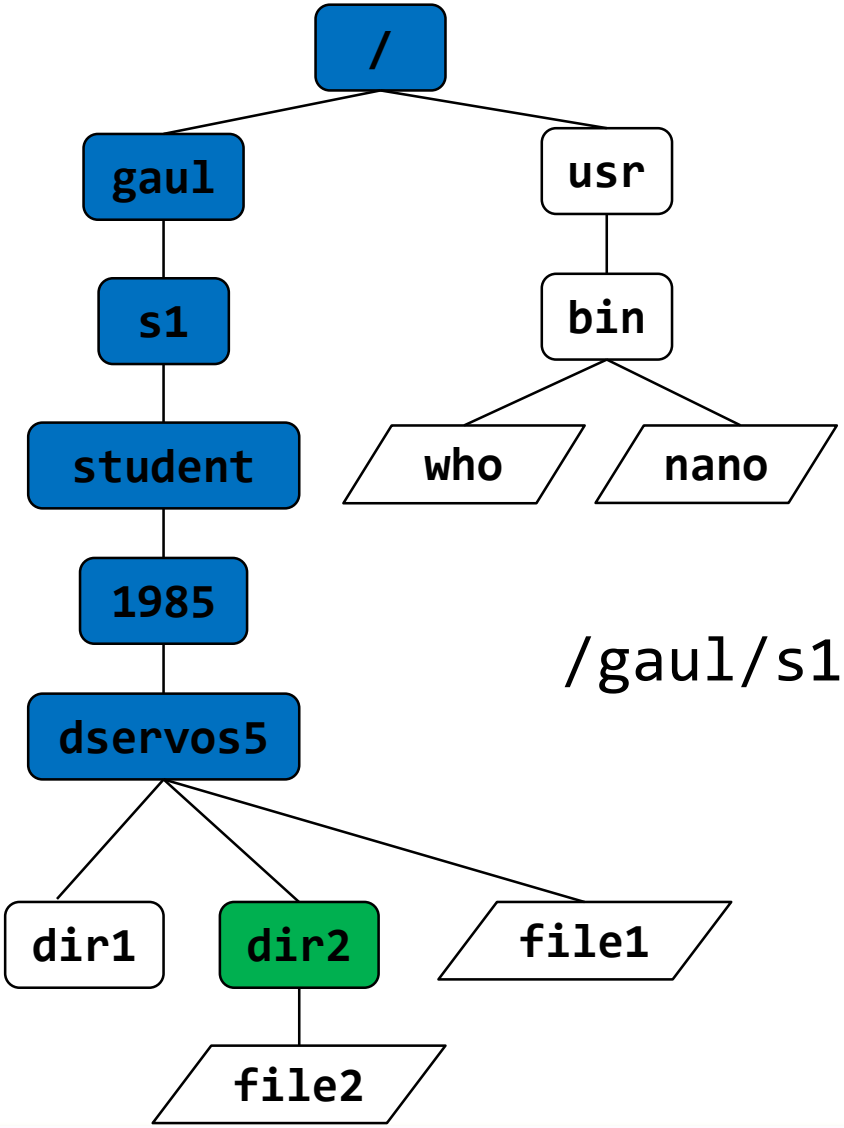
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

```
/gaul/s1/student/1985
```

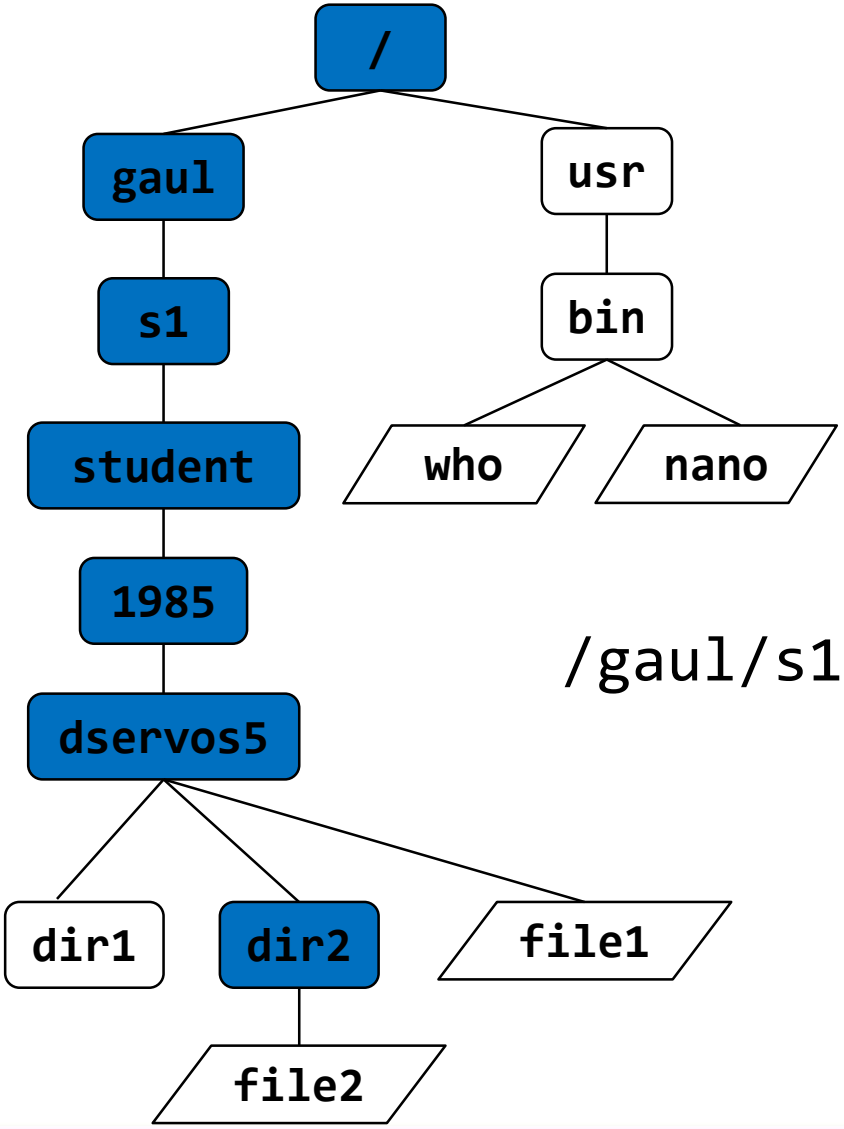
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

```
/gaul/s1/student/1985/dservos5
```

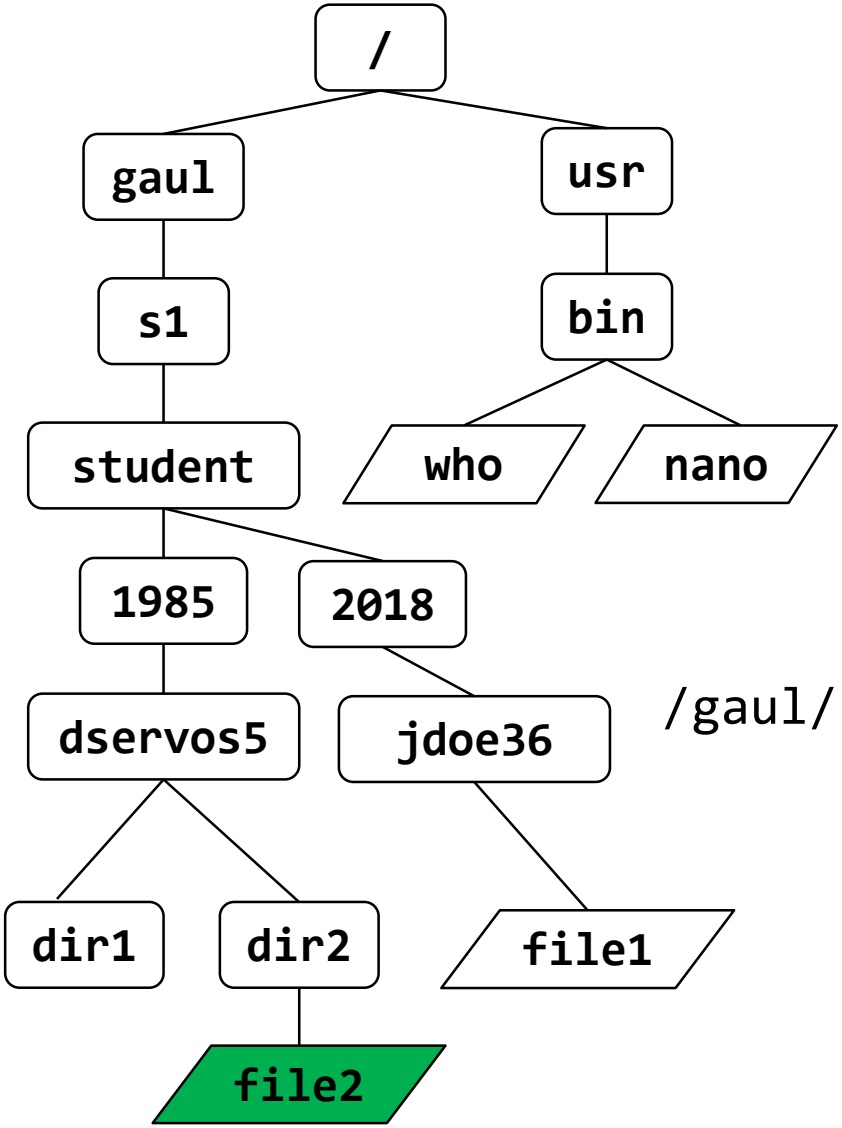
Absolute Paths



- Absolute paths are based on the file hierarchy.
- **Example 2:** Find the absolute path for the dir2 directory in my home directory.

```
/gaul/s1/student/1985/dservos5/dir2
```

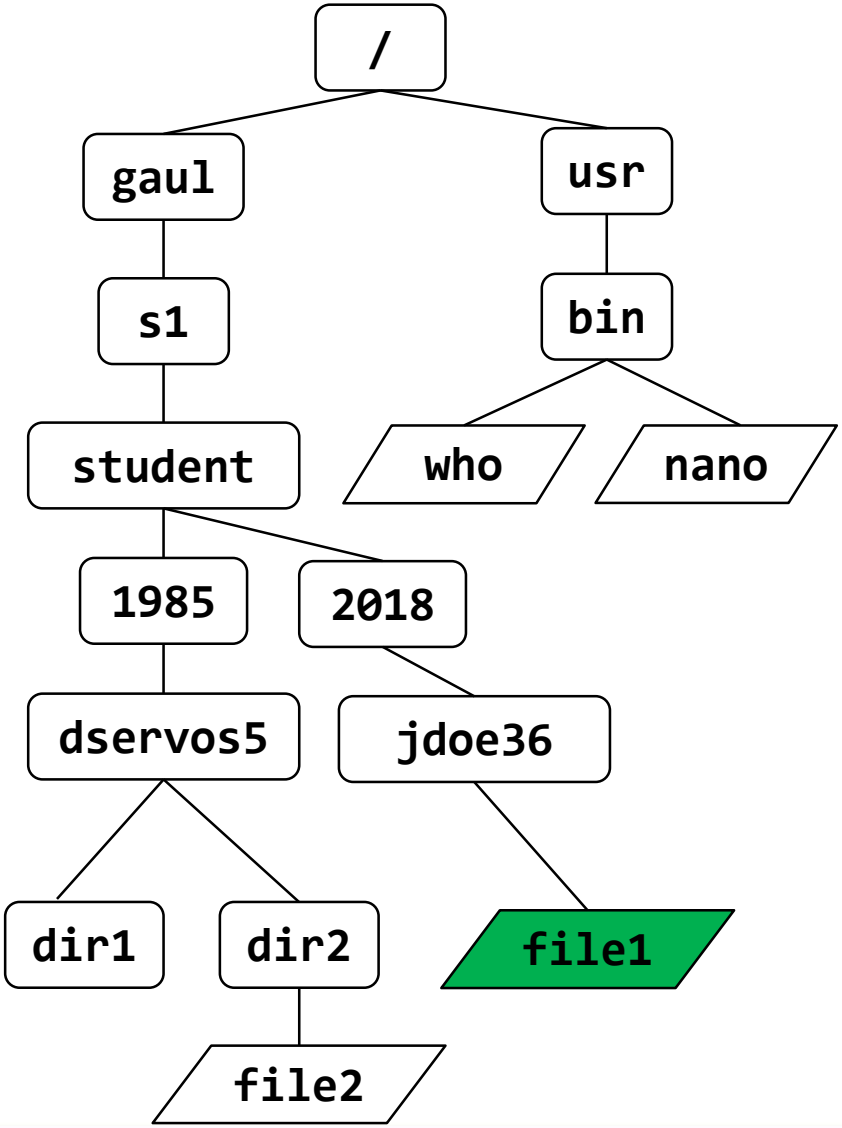
Absolute Paths



- Special shortcut for home directories: ~
- ~ is equivalent to your home directory.
- If you are dservos5, `~/dir2/file2` would be equivalent to:

`/gaul/s1/student/1985/dservos5/dir2/file2`

Absolute Paths

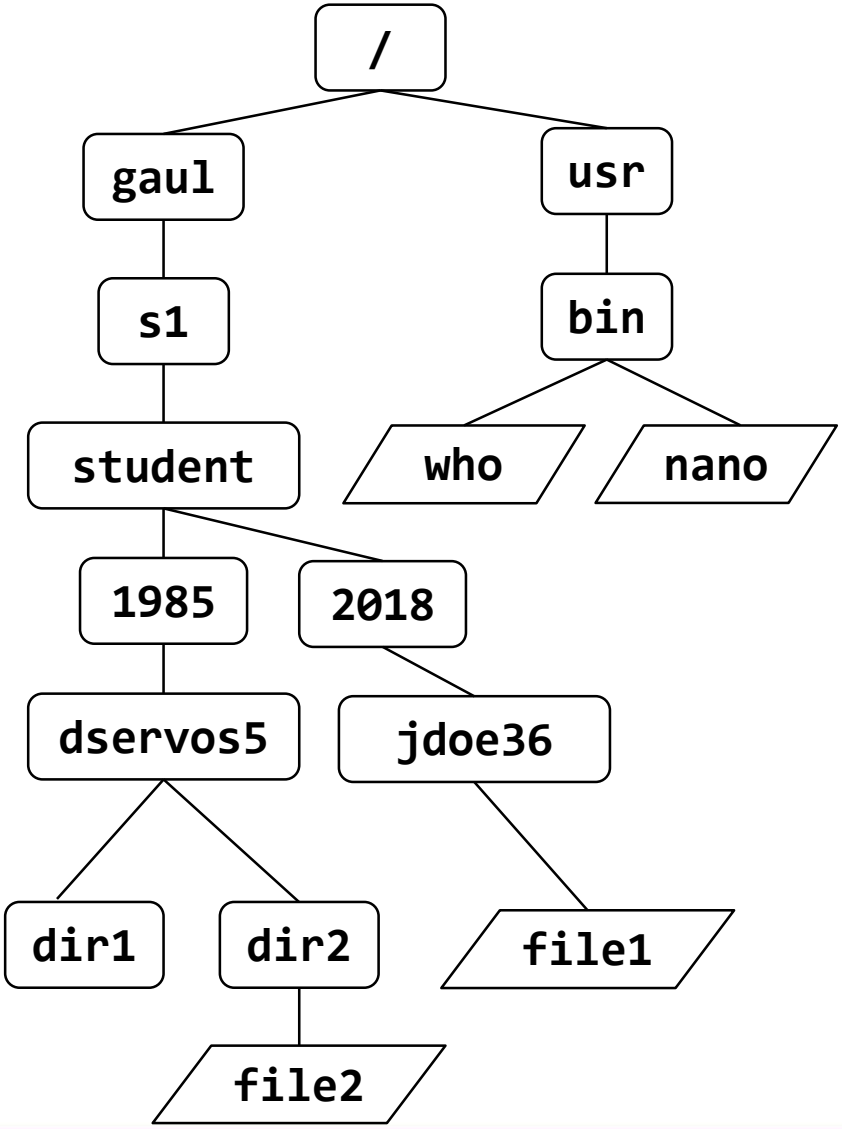


- Can also be used to refer to other user's home directories by putting their user name directly after the ~
- For example, even if you are dservos5, `~jdoe36/file1` would be equivalent to:

`/gaul/s1/student/2018/jdoe36/file1`

- This would be a file name "file1" in the user jdoe36's home directory.

Relative Paths

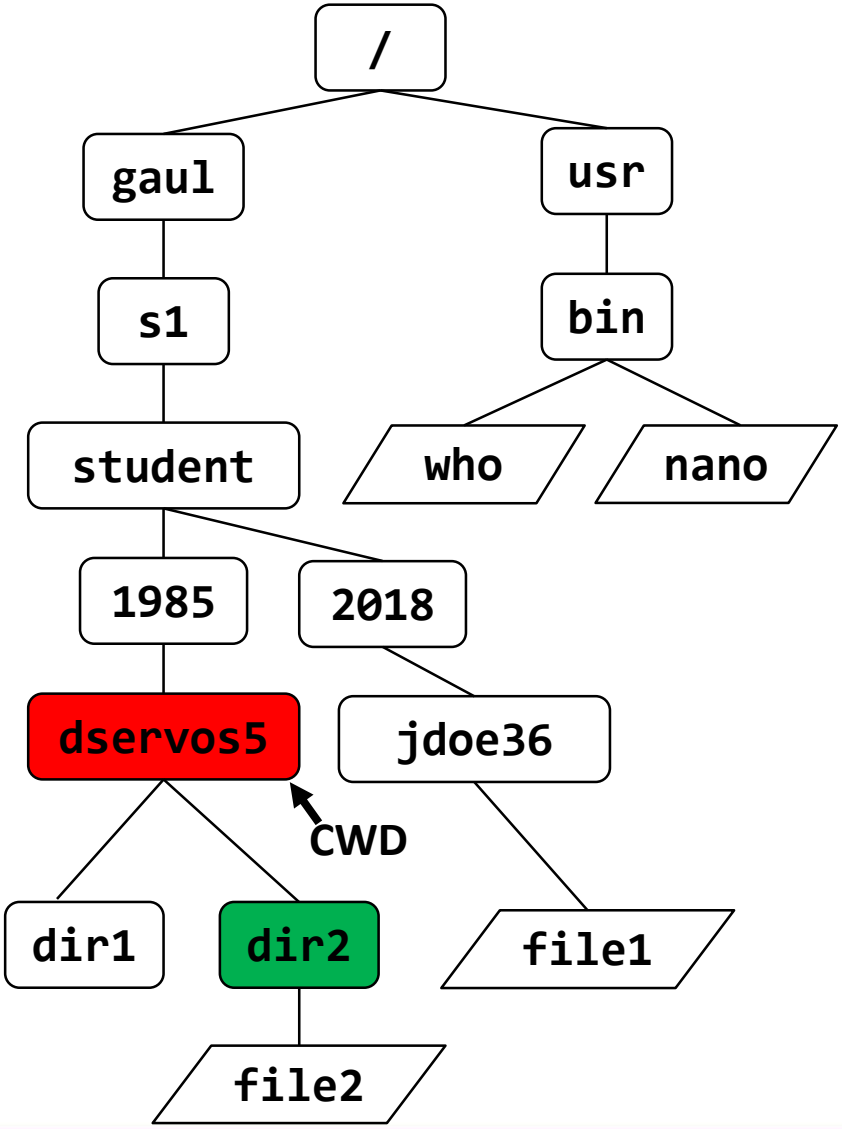


- Relative paths are dependent on your current working directory (the directory you are “in”)
- You can find your current working directory by using the **pwd** command.
- When you first log in, your current working directory is normally set to your home directory.
- **Example:** For user dservos5 **pwd** would output:

  `/gaul/s1/student/1985/dservos5`

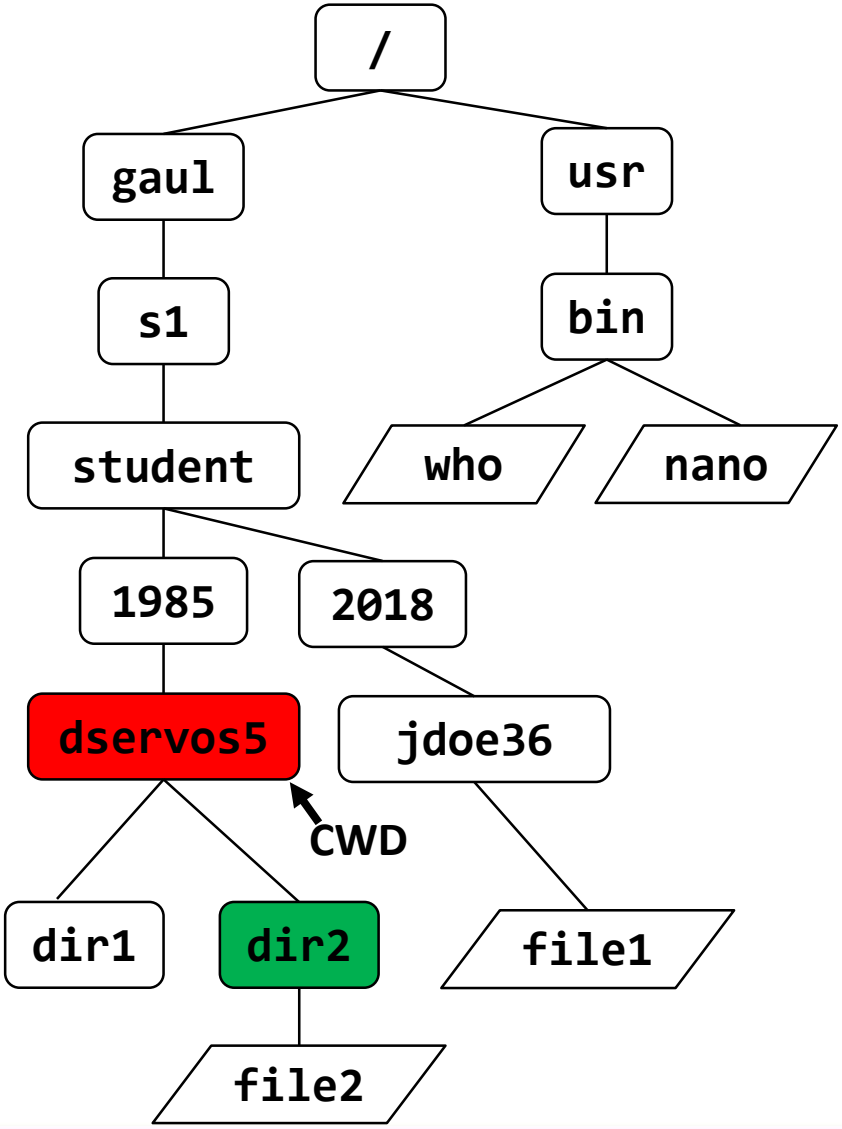
  if their current working directory is their home directory.

Relative Paths



- Relative paths are easier to work with than absolute paths when navigating via the command line.
- Relative paths do not start with / or ~.
- **Example 1:** What is the relative path to dir2 if you are in ~dserovs5?

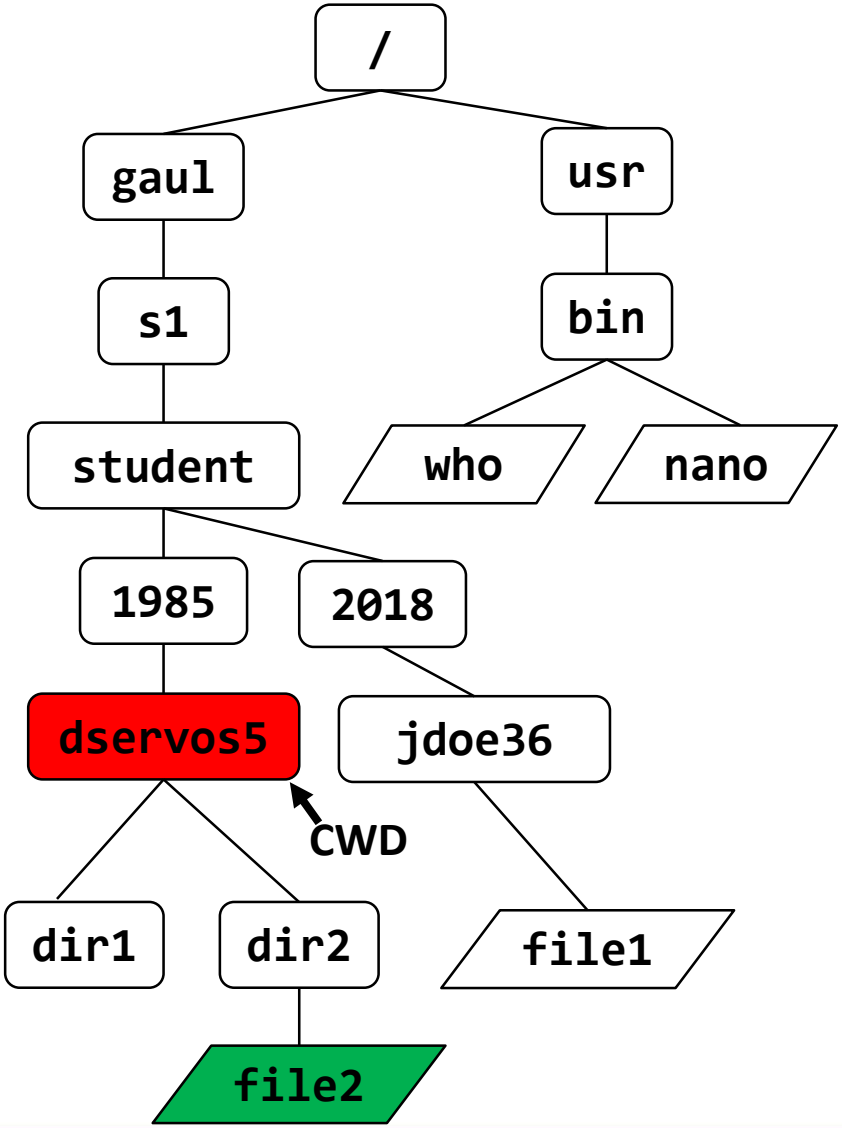
Relative Paths



- Relative paths are easier to work with then absolute paths when navigating via the command line.
- Relative paths do not start with / or ~.
- **Example 1:** What is the relative path to dir2 if you are in ~dserovs5?

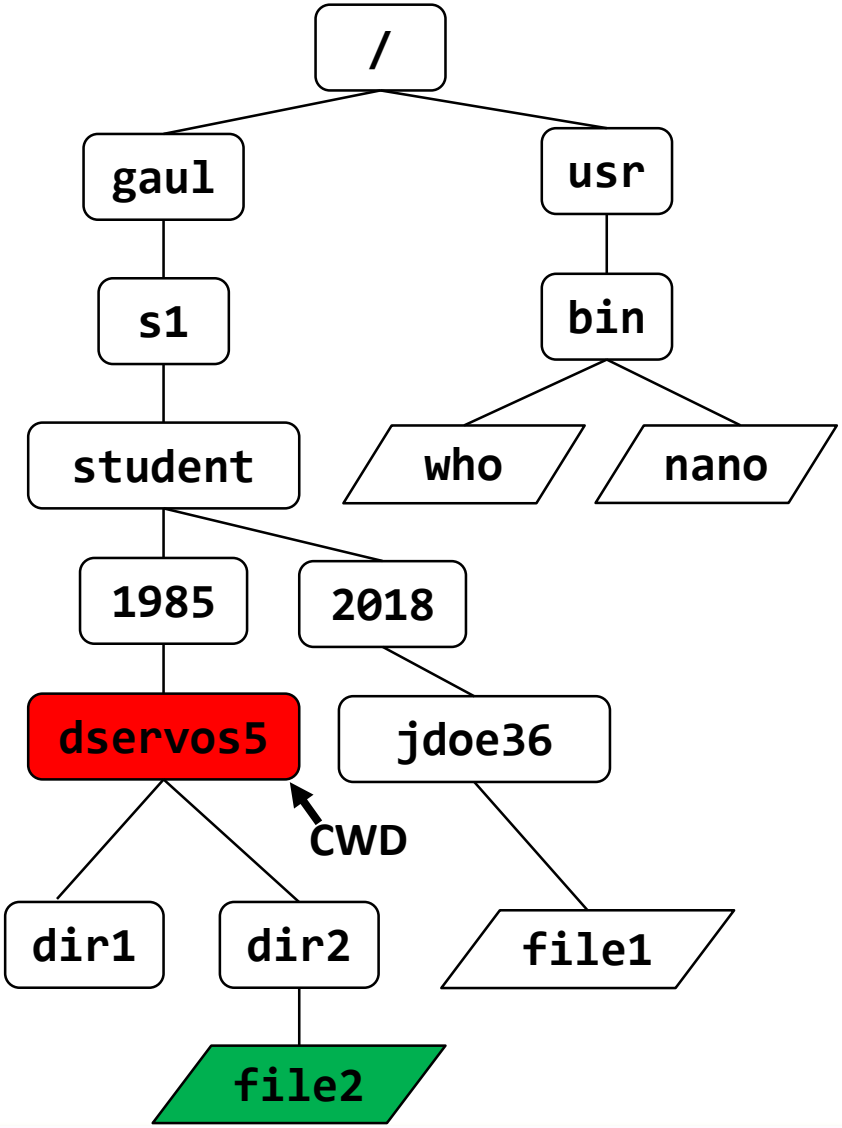
`dir2`

Relative Paths



- Relative paths are easier to work with then absolute paths when navigating via the command line.
- Relative paths do not start with / or ~.
- **Example 2:** What is the relative path to file2 if you are in ~dserovs5?

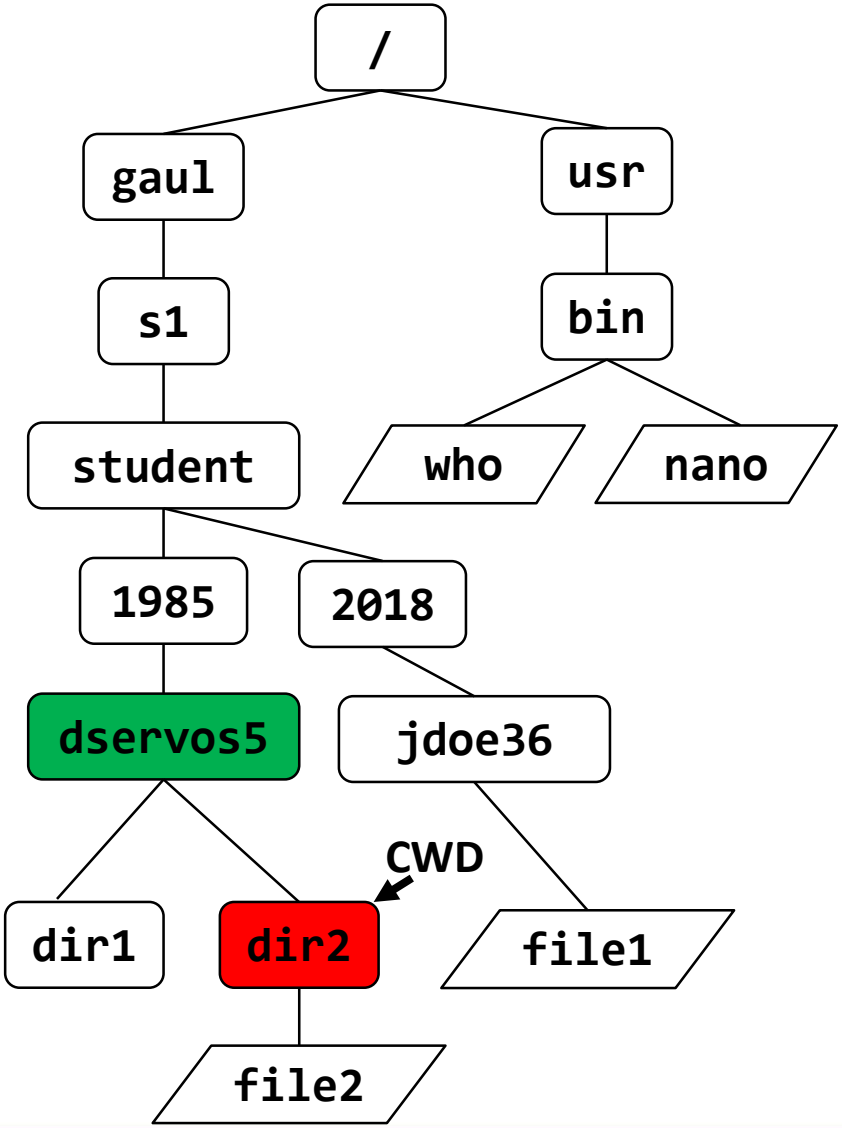
Relative Paths



- Relative paths are easier to work with then absolute paths when navigating via the command line.
- Relative paths do not start with / or ~.
- **Example 2:** What is the relative path to file2 if you are in ~dserovs5?

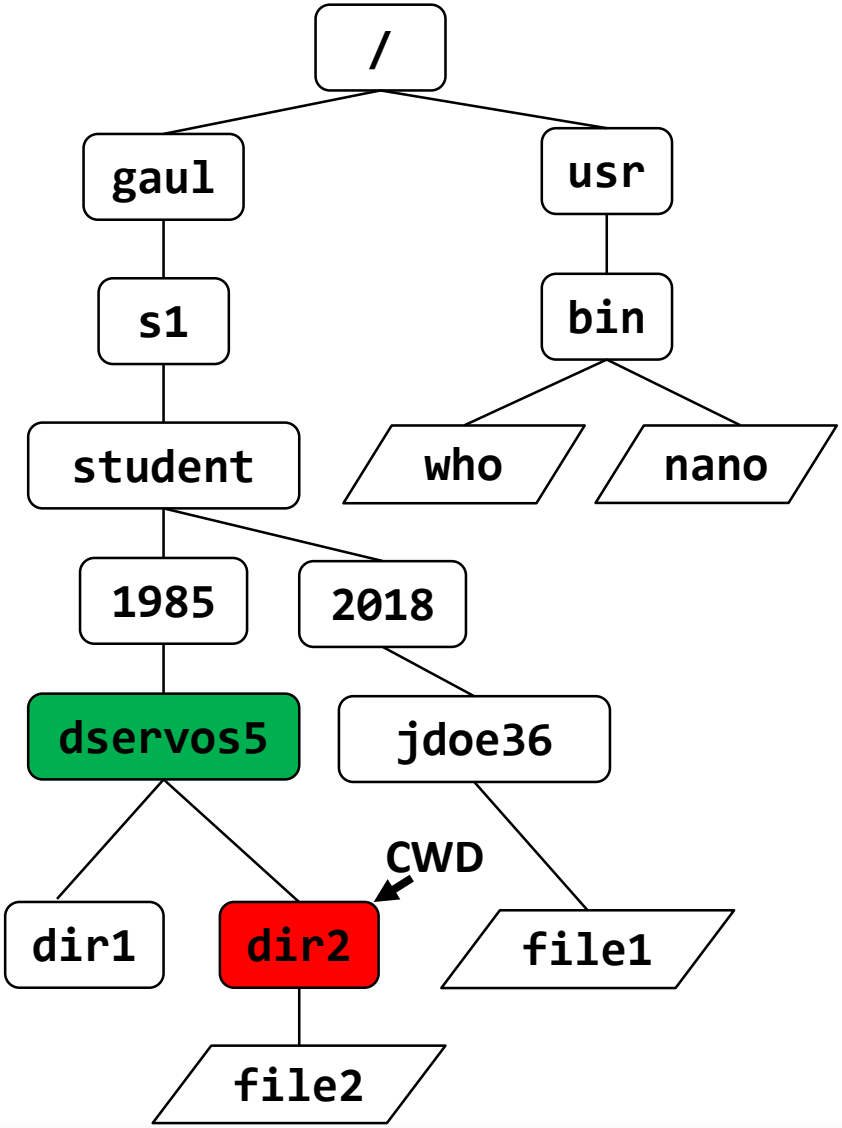
```
dir2/file2
```

Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 1:** If dir2 is the current working directory, what is the relative path of ~dservos5?

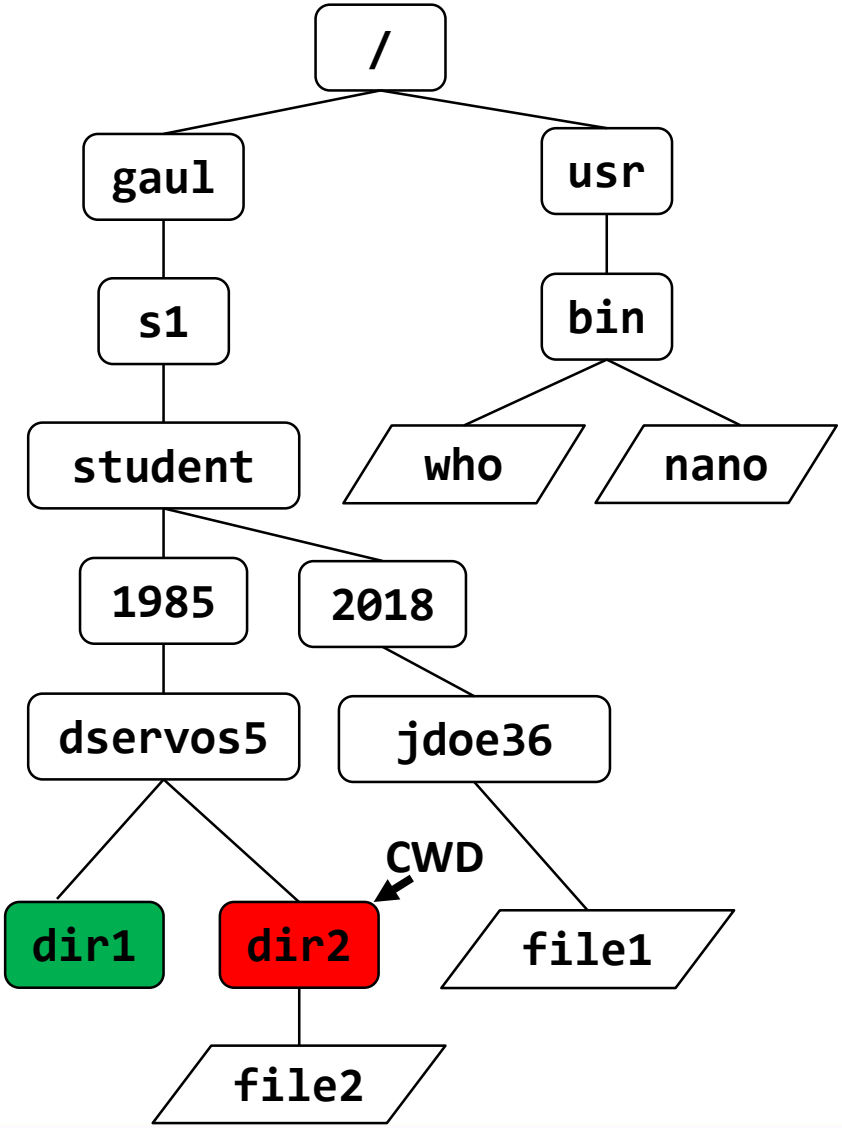
Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 1:** If dir2 is the current working directory, what is the relative path of ~dservos5?

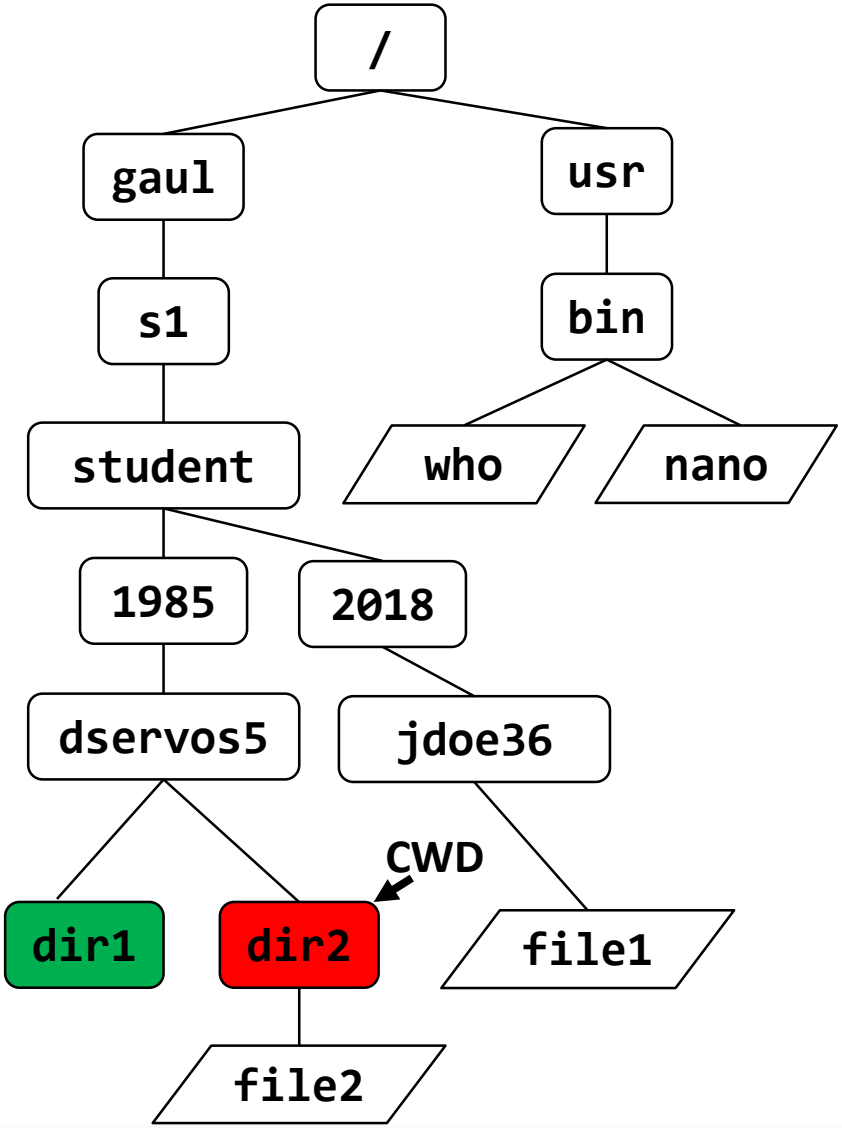
. .

Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 2:** If dir2 is the current working directory, what is the relative path of dir1?

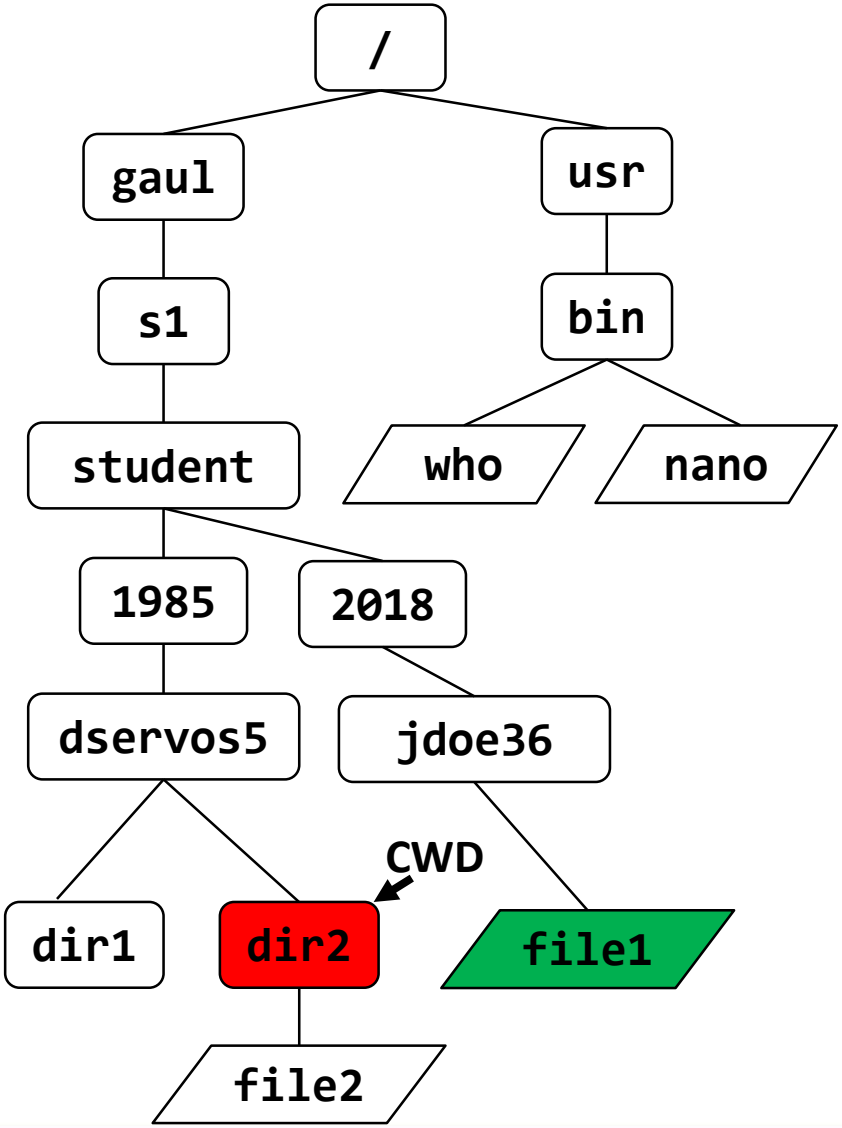
Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 2:** If dir2 is the current working directory, what is the relative path of dir1?

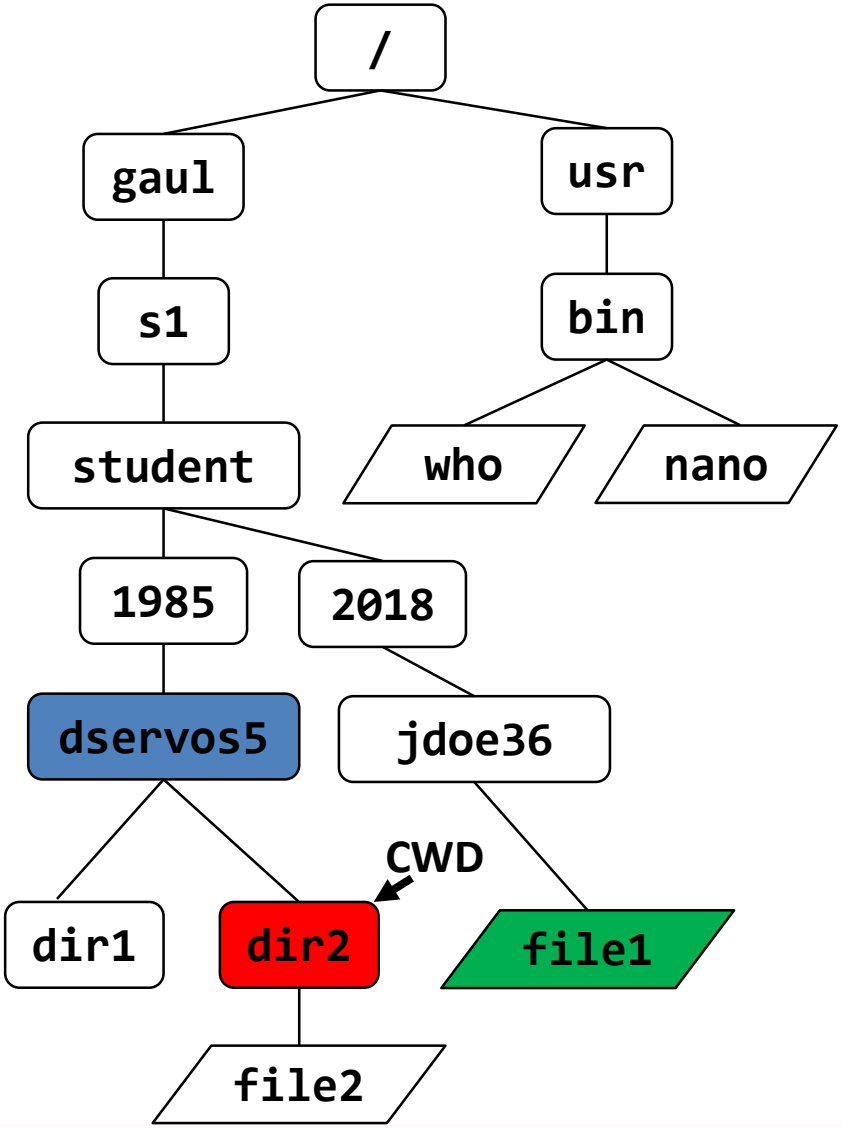
```
../dir1
```

Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 3:** If dir2 is the current working directory, what is the relative path of file1?

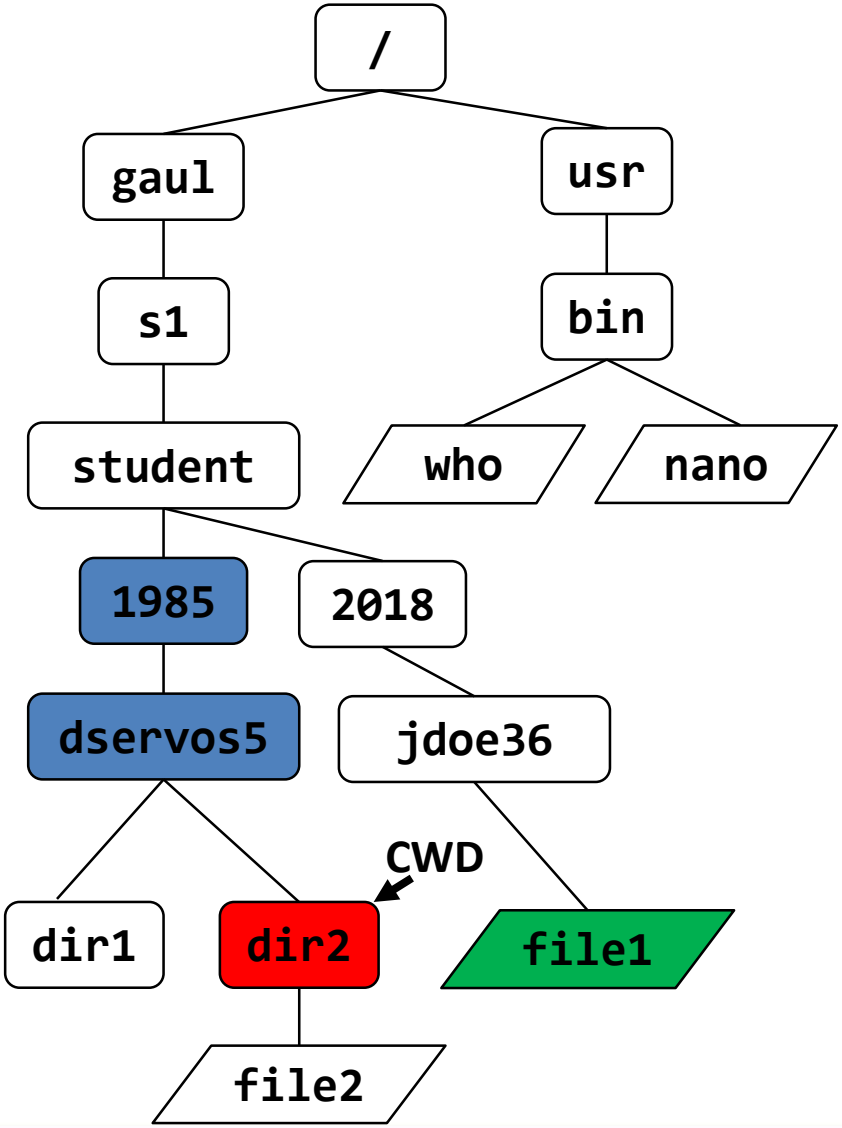
Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 3:** If dir2 is the current working directory, what is the relative path of file1?

. .

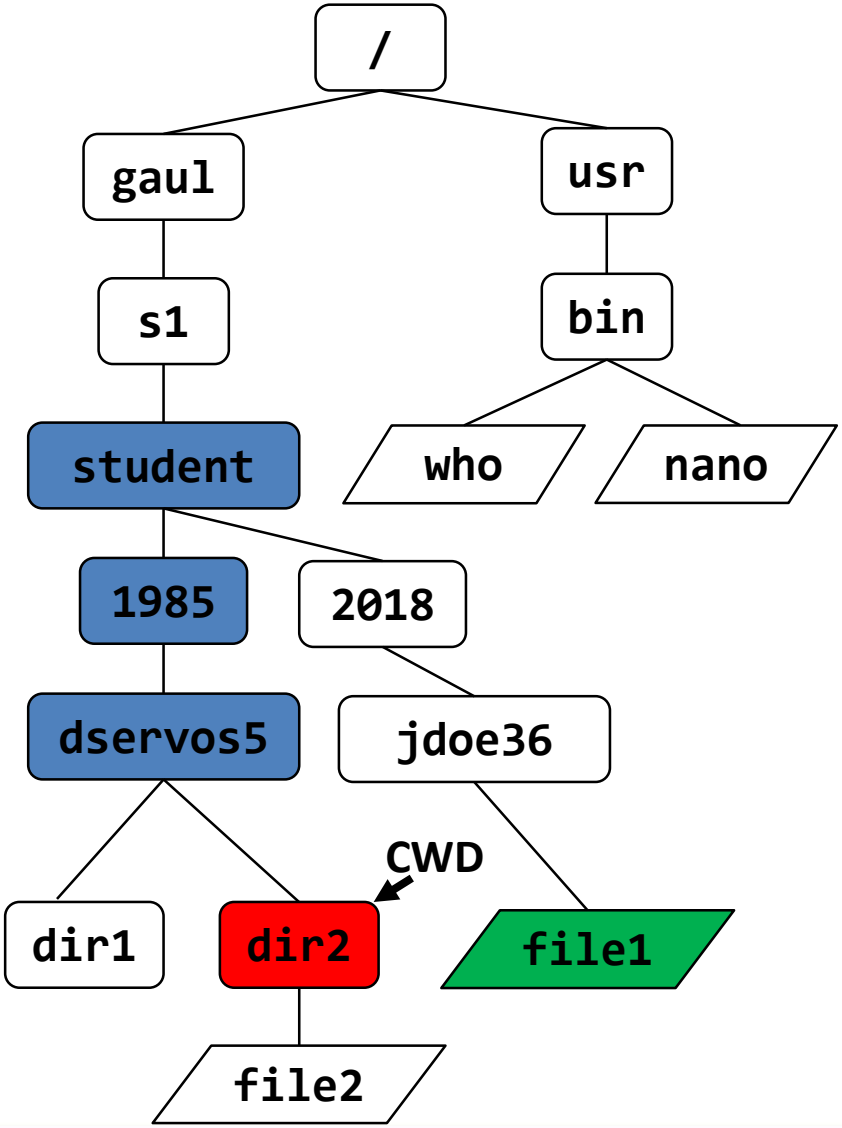
Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 3:** If dir2 is the current working directory, what is the relative path of file1?

../..

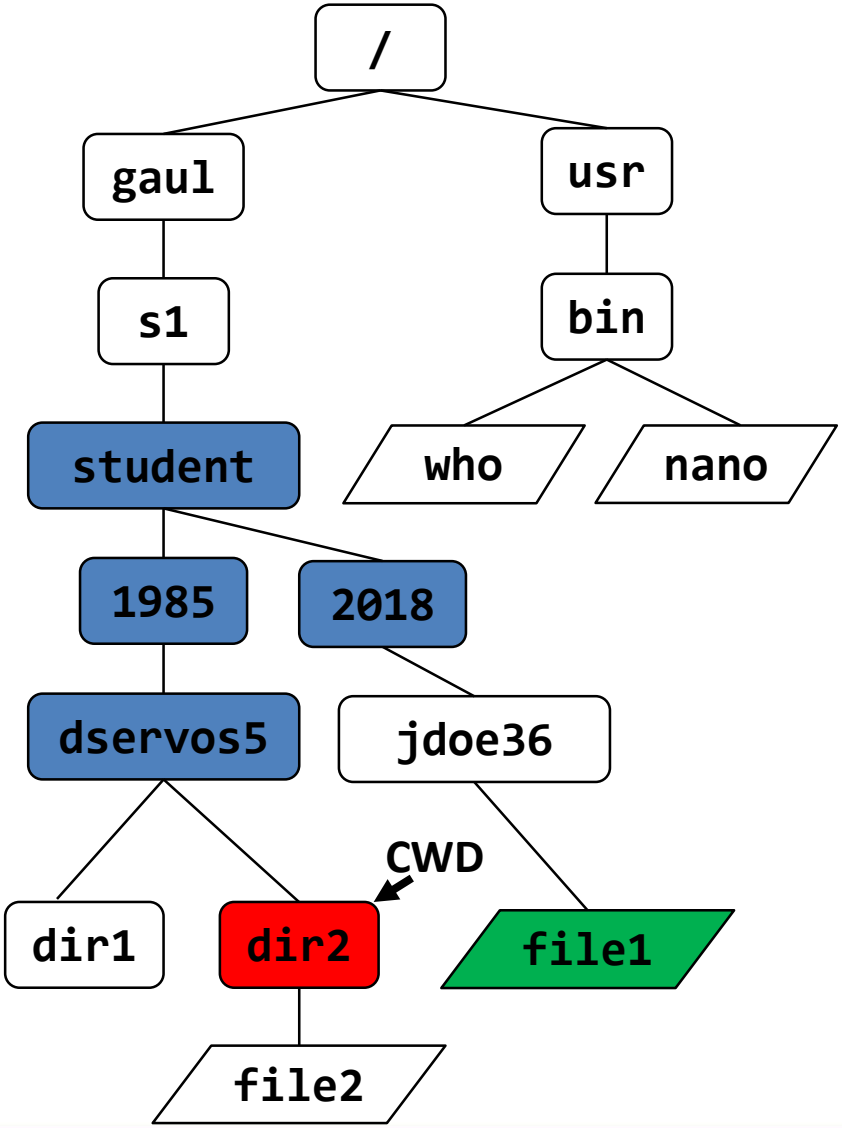
Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 3:** If dir2 is the current working directory, what is the relative path of file1?

../../..

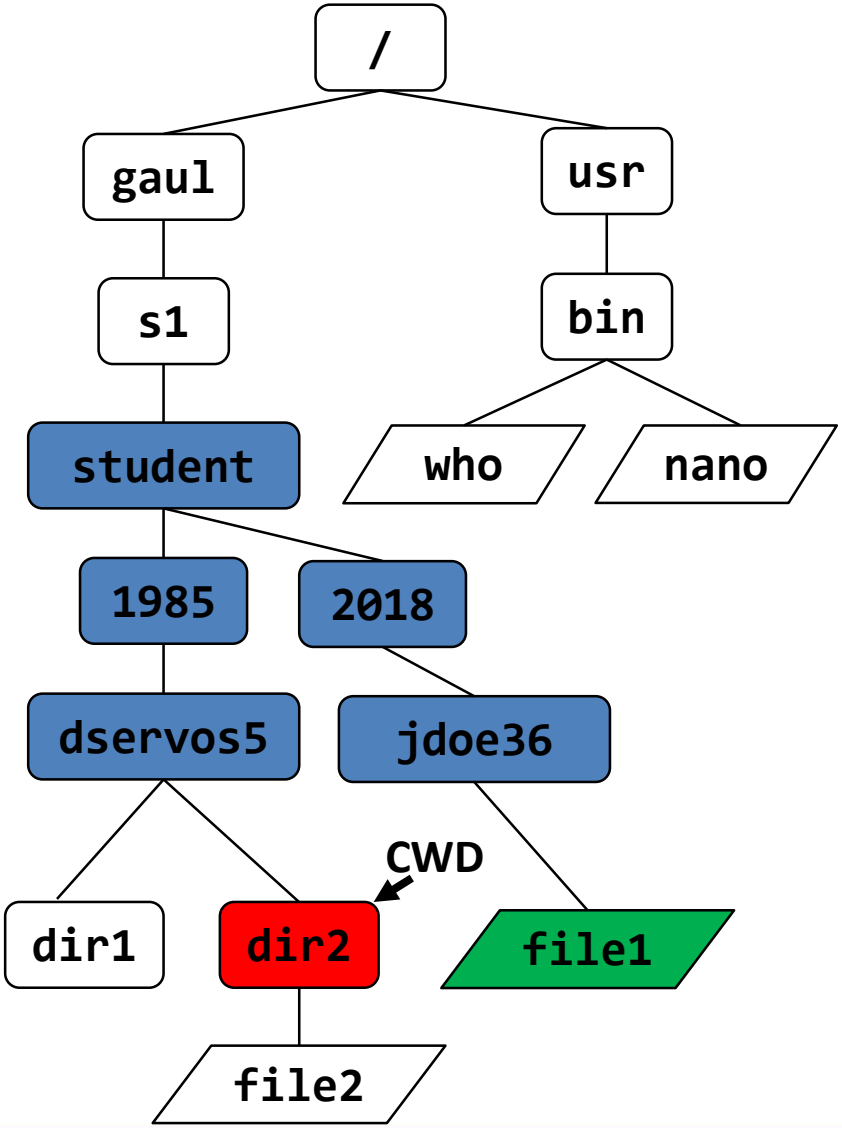
Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 3:** If dir2 is the current working directory, what is the relative path of file1?

```
../../../2018/
```

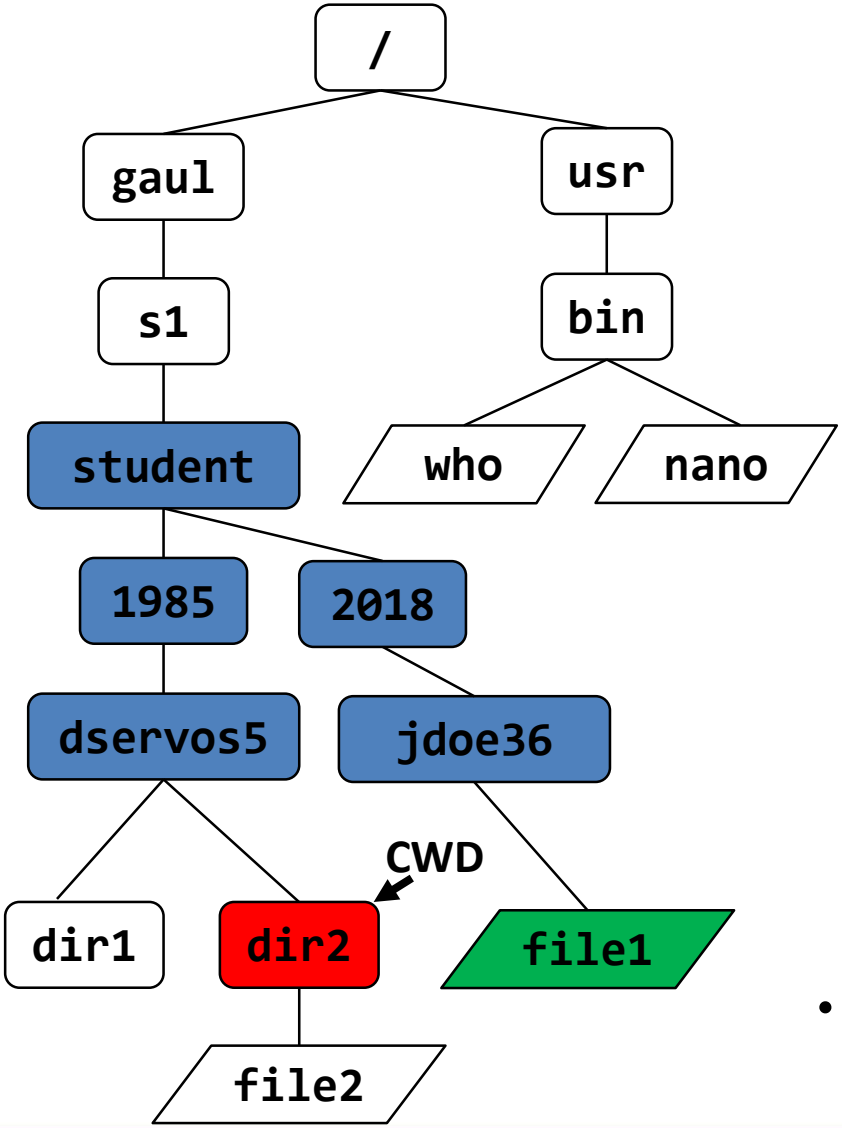
Relative Paths



- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 3:** If dir2 is the current working directory, what is the relative path of file1?

```
../../../2018/jdoe36
```

Relative Paths

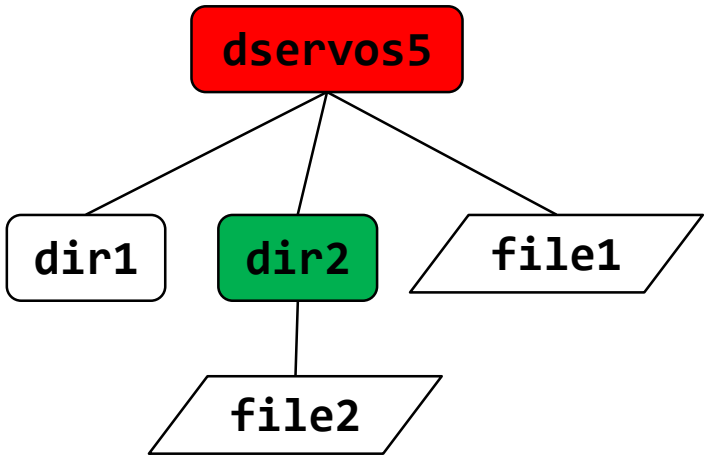


- Two special symbols exist for relative paths:
  - . (a single dot)
  - .. (two dots)
- . represents the current directory and .. represents the parent directory of the current directory.
- **Example 3:** If dir2 is the current working directory, what is the relative path of file1?

```
../../../2018/jdoe36/file1
```

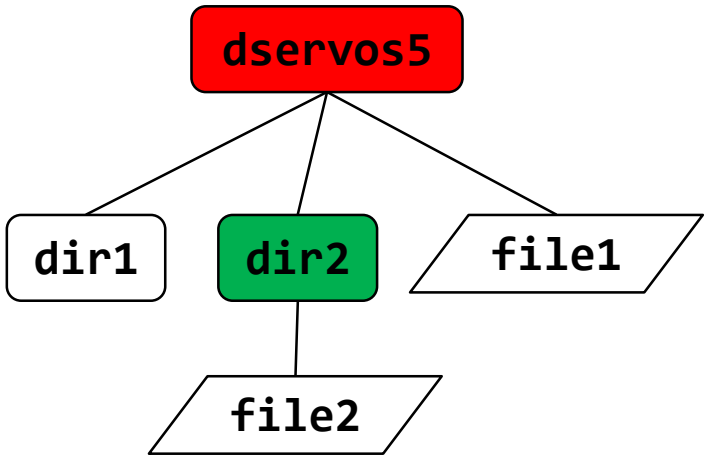
Navigating the File System

- We can use the **cd** command to change our current working directory.
- We can use absolute or relative paths.
- **Example 1:** *dservos5* is the current working directory. How do we change to *dir2*?



Navigating the File System

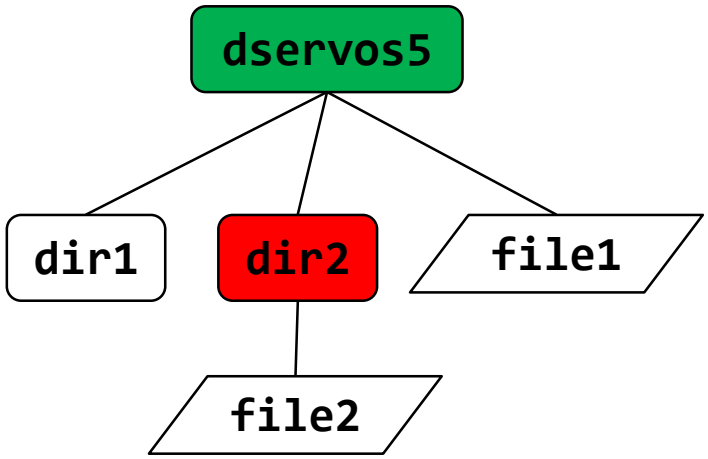
- We can use the **cd** command to change our current working directory.
- We can use absolute or relative paths.
- **Example 1:** *dservos5* is the current working directory. How do we change to *dir2*?




```
cd dir2        OR        cd ~dservos5/dir2
```

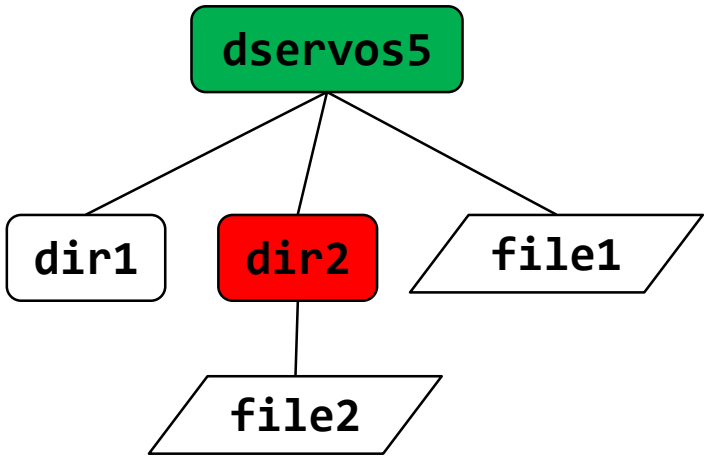
Navigating the File System

- We can use the **cd** command to change our current working directory.
- We can use absolute or relative paths.
- **Example 2:** *dir2* is the current working directory. How do we change to our home directory (we are dservos5)?



Navigating the File System

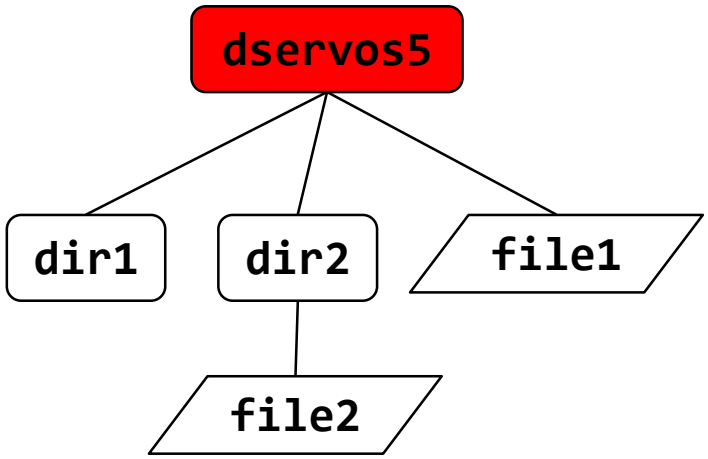
- We can use the **cd** command to change our current working directory.
- We can use absolute or relative paths.
- **Example 2:** *dir2* is the current working directory. How do we change to our home directory (we are dservos5)?



`cd ..` **OR** `cd ~` **OR** `cd`

Navigating the File System

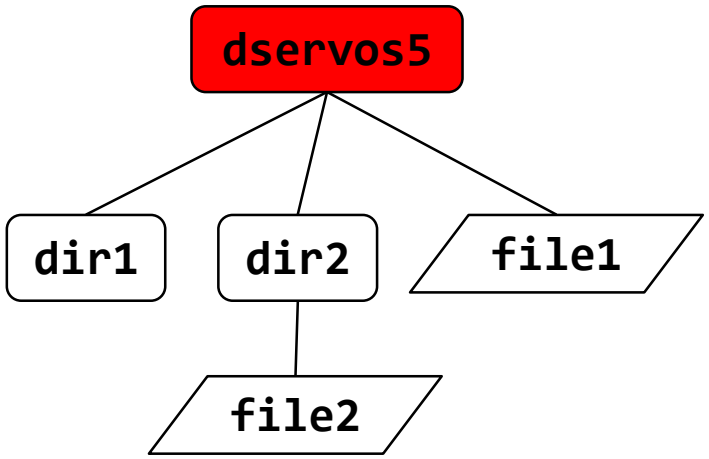
- We can list the files in a directory using the **`ls`** command.
- With no arguments, **`ls`** lists the files in the current directory.
- If **`ls`** is given a directory as an argument, it lists the files inside.
- If **`ls`** is given a file as an argument, it lists details on that file.



Navigating the File System

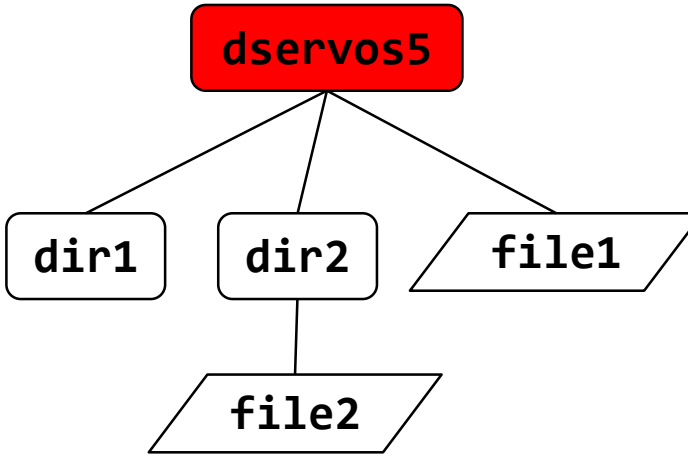
- **Examples:**

```
[dservos5@cs2211b ~]$ ls
dir1  dir2  file1

[dservos5@cs2211b ~]$ ls dir2
file2

[dservos5@cs2211b ~]$ ls file1
file1

[dservos5@cs2211b ~]$ ls file1 dir2
file1

dir2:
file2
```



Navigating the File System

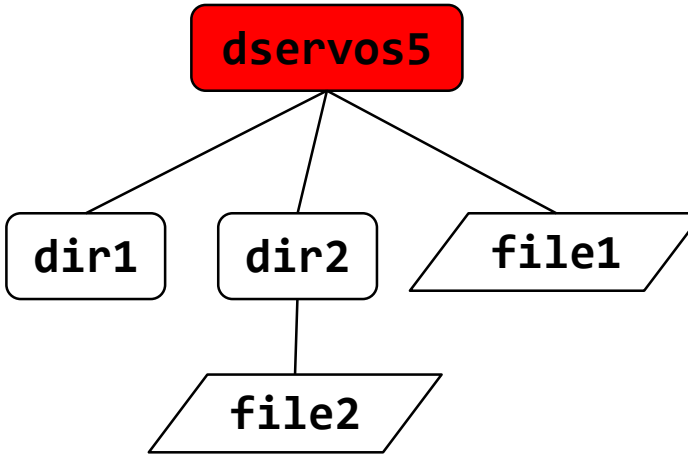
- The `-l` option lists more details about the files.
- **Examples:**




```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1

[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**



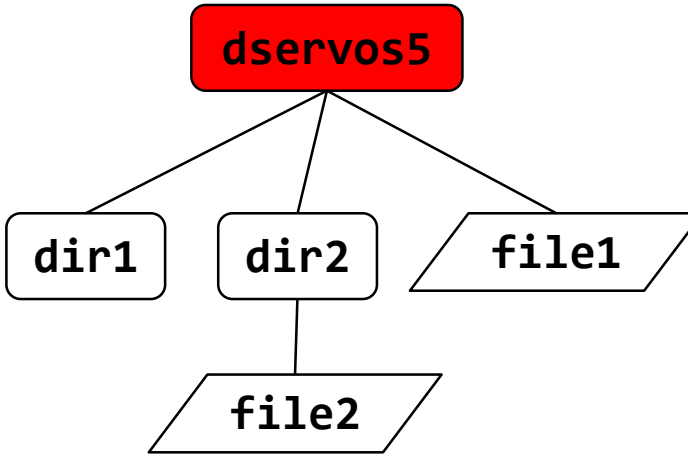

```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1
```

Permissions

```
[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**



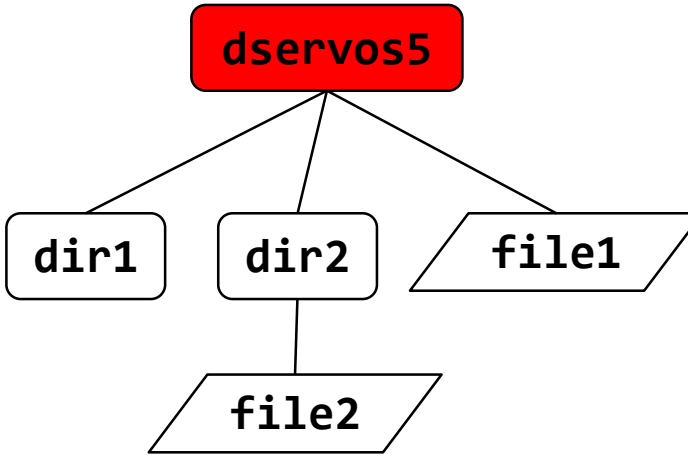

```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1
```

Link Count

```
[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**



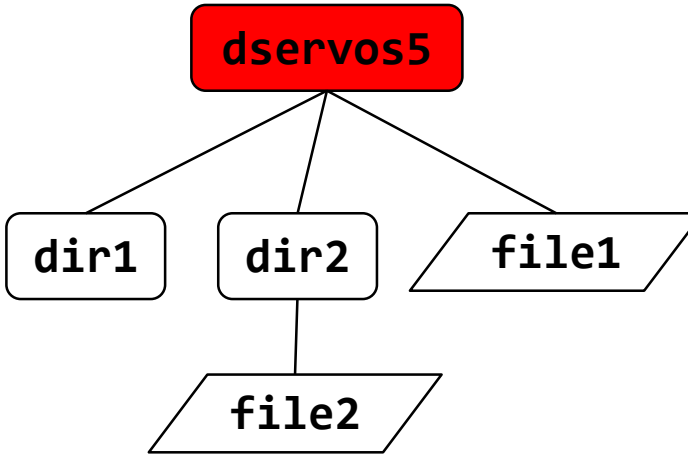

```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1
```

Owner

```
[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**



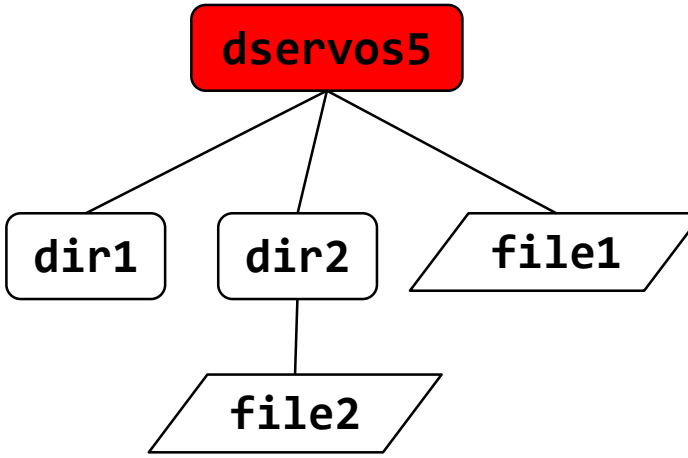

```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1
```

Group

```
[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**



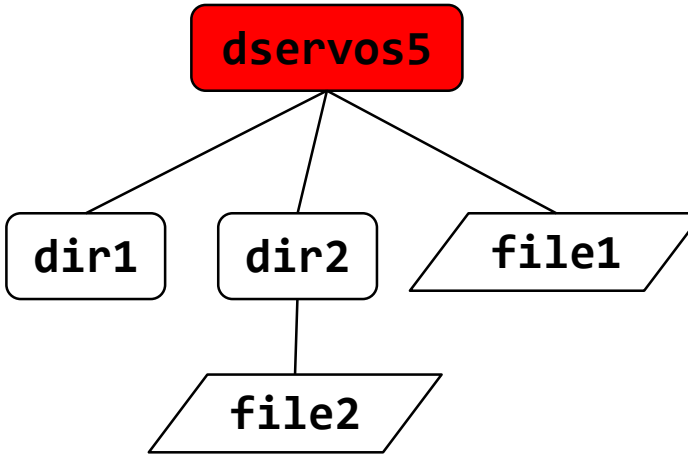

```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1
```

File Size (in bytes)

```
[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**



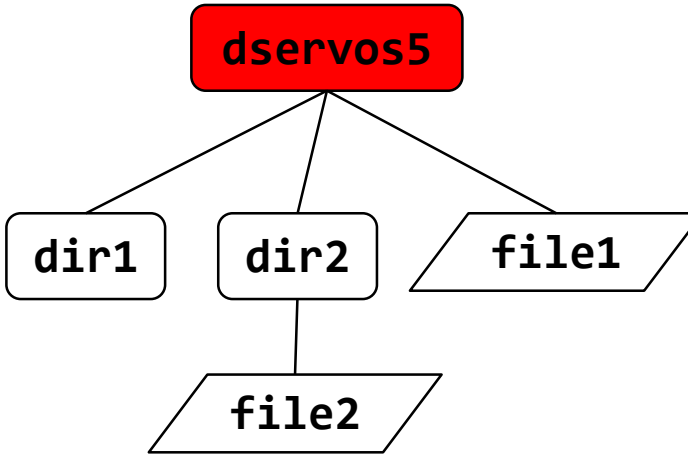

```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1
```

Last Modification Date/Time

```
[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**



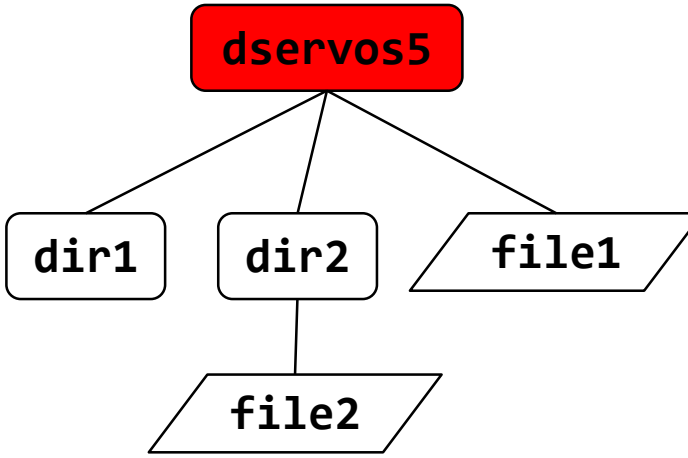

```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1
```

File Name

```
[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**

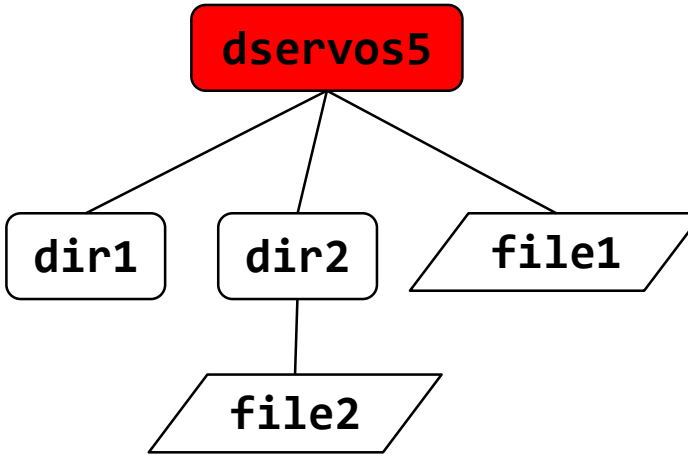



```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1

[dservos5@cs2211b ~]$ ls –l dir2
total 0
-rw-------. 1 dservos5 grad 0 Jan 14 06:17 file2
```

How can we get details just for dir2 and not the files in it?

Navigating the File System

- The `-l` option lists more details about the files.
- **Examples:**

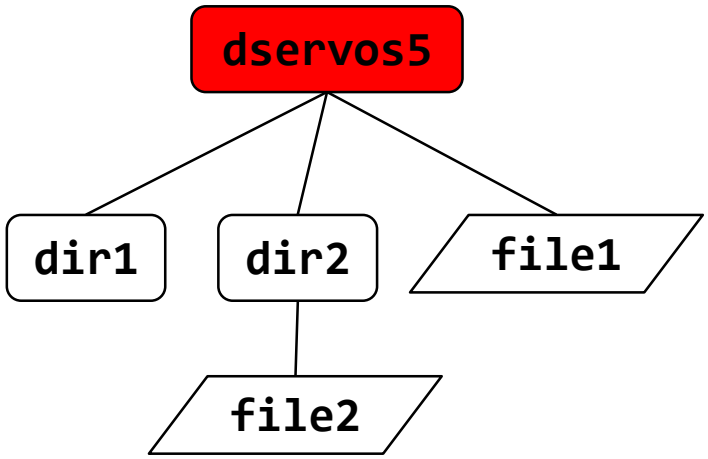



```
[dservos5@cs2211b ~]$ ls -l
total 4
drwx------. 2 dservos5 grad 19 Jan 14 05:55 dir1
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
-rw-------. 1 dservos5 grad  4 Jan 14 05:54 file1

[dservos5@cs2211b ~]$ ls –ld dir2
drwx------. 2 dservos5 grad 19 Jan 14 06:17 dir2
```

Use -d option.

Navigating the File System

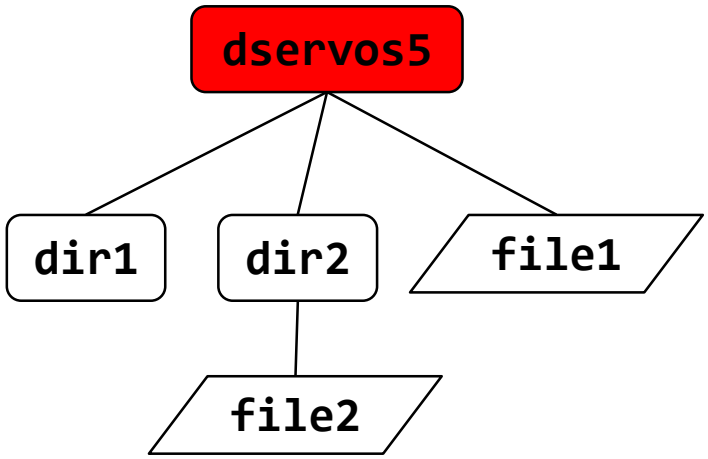
- Normally, files that start with a . (single dot) are hidden from the output of **`ls`**.
- **`-a`** option allows us to see them.
- **Example:**




```
[dservos5@cs2211b ~]$ ls –a
.   ..  .bash_history  .bash_logout   .bash_profile  .bashrc
.emacs  dir1    dir2    file1
```

Making and Removing Directories

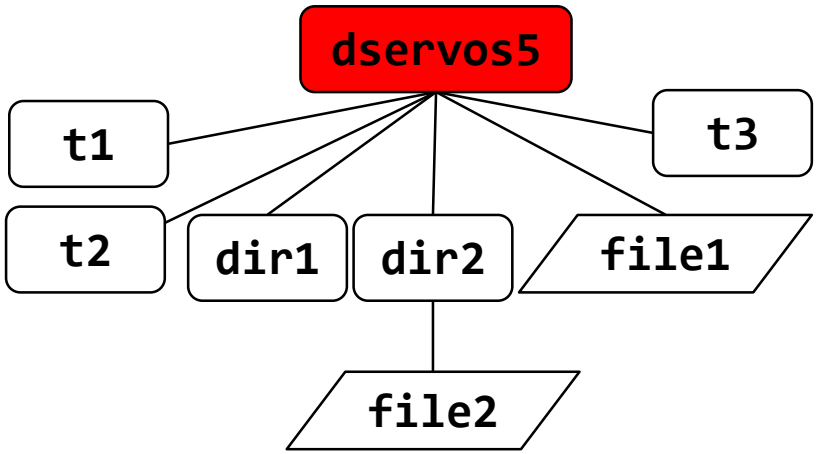
- The mkdir command can make one or more directories at a time.
- Each argument is a directory name to create.
- **Example:**




```
[dservos5@cs2211b ~]$ mkdir t1 t2 t3
```

Making and Removing Directories

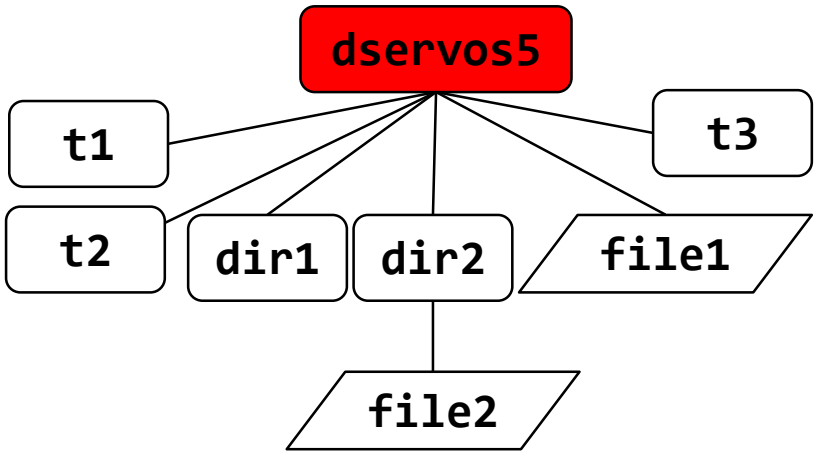
- The **`mkdir`** command can make one or more directories at a time.
- Each argument is a directory name to create.
- **Example:**




```
[dservos5@cs2211b ~]$ mkdir t1 t2 t3
[dservos5@cs2211b ~]$ ls
dir1  dir2  file1  t1  t2  t3
```

Making and Removing Directories

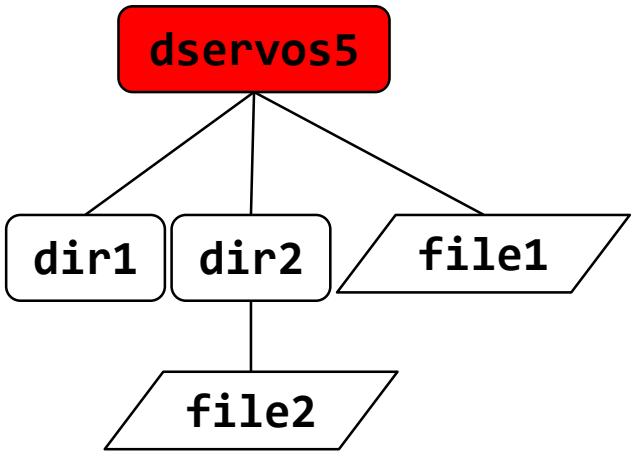
- The **`rmdir`** command can delete directories but only if they are empty.
- Each argument is a directory to be deleted.
- **Example:**




```
[dservos5@cs2211b ~]$ rmdir t1 t2 t3
```

Making and Removing Directories

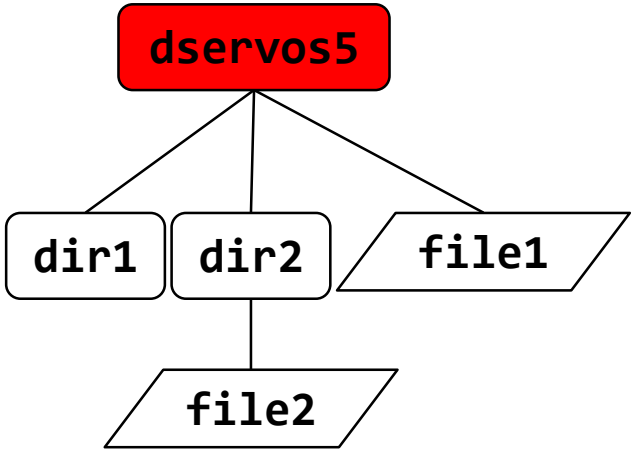
- The **`rmdir`** command can delete directories but only if they are empty.
- Each argument is a directory to be deleted.
- **Example:**




```
[dservos5@cs2211b ~]$ rmdir t1 t2 t3
[dservos5@cs2211b ~]$ ls
dir1  dir2  file1
```

Making and Removing Directories

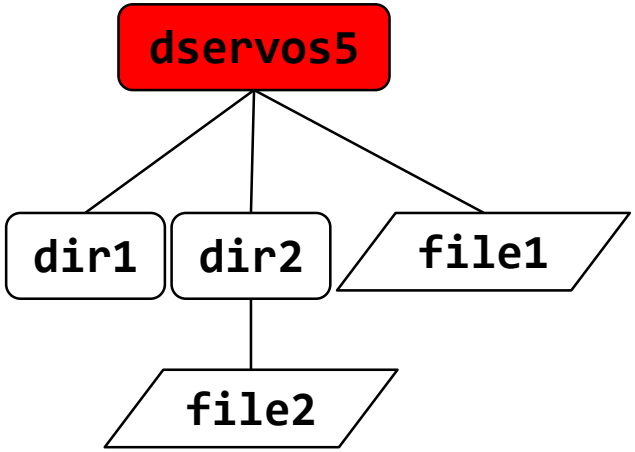
- You will get an error if they contain a file or subdirectory.
- **Example:**




```
[dservos5@cs2211b ~]$ rmdir dir2
rmdir: failed to remove ‘dir2’: Directory not empty
```

Making and Removing Directories

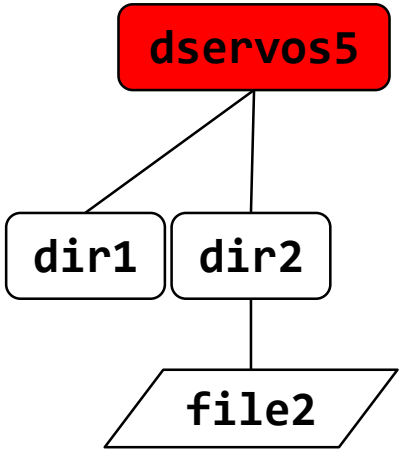
- **`rm`** can delete files, but it can also delete directories if used with the -r (recursive) option.
- **Example 1:** Delete file1




```
[dservos5@cs2211b ~]$ rm file1
```

Making and Removing Directories

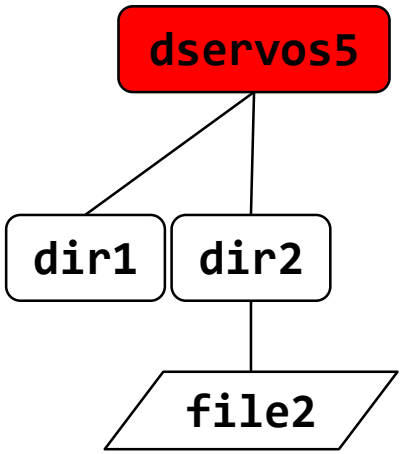
- **`rm`** can delete files, but it can also delete directories if used with the -r (recursive) option.
- **Example 1:** Delete file1




```
[dservos5@cs2211b ~]$ rm file1
[dservos5@cs2211b ~]$ ls
dir1 dir2
```

Making and Removing Directories

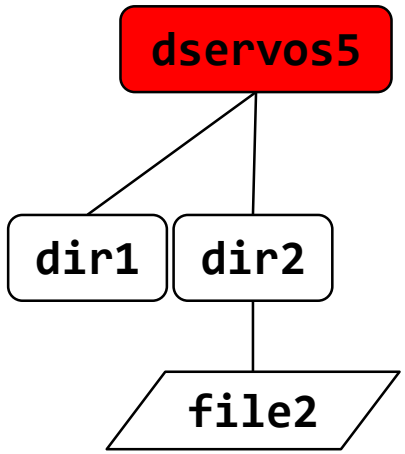
- **`rm`** can delete files, but it can also delete directories if used with the -r (recursive) option.
- **Example 2:** Delete dir2




```
[dservos5@cs2211b ~]$ rm dir2
```

Making and Removing Directories

- **`rm`** can delete files, but it can also delete directories if used with the -r (recursive) option.
- **Example 2:** Delete dir2

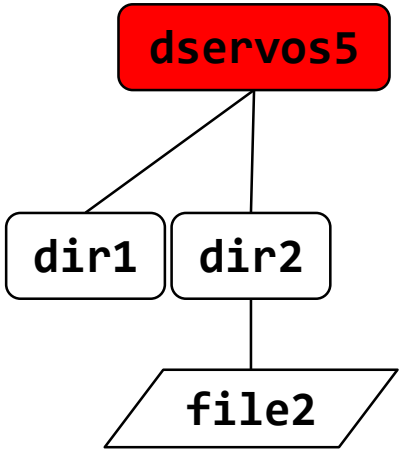



```
[dservos5@cs2211b ~]$ rm dir2
[dservos5@cs2211b ~]$ rm: cannot remove ‘dir2’: Is a directory
```

Need to use -r option!

Making and Removing Directories

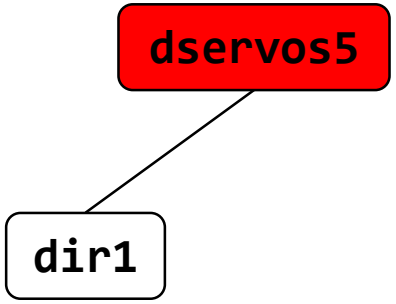
- **`rm`** can delete files, but it can also delete directories if used with the -r (recursive) option.
- **Example 2:** Delete dir2




```
[dservos5@cs2211b ~]$ rm –r dir2
```

Making and Removing Directories

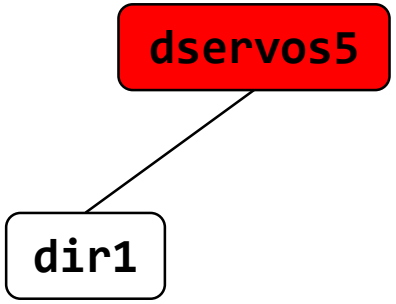
- **`rm`** can delete files, but it can also delete directories if used with the -r (recursive) option.
- **Example 2:** Delete dir2




```
[dservos5@cs2211b ~]$ rm –r dir2
[dservos5@cs2211b ~]$ ls
dir1
```

Making, Copying and Moving Files

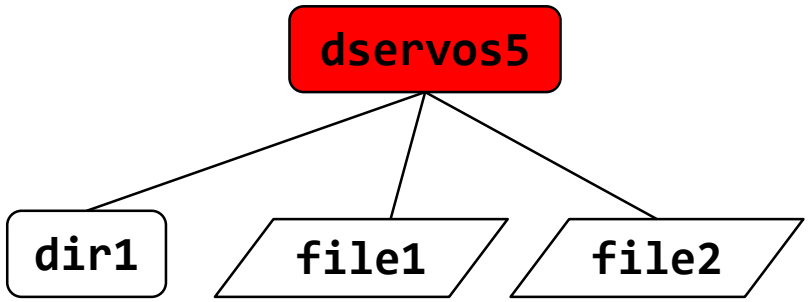
- The **`touch`** command can create files.
- Each argument is a file that will be created.
- **Example:**




```
[dservos5@cs2211b ~]$ touch file1 file2
```

Making, Copying and Moving Files

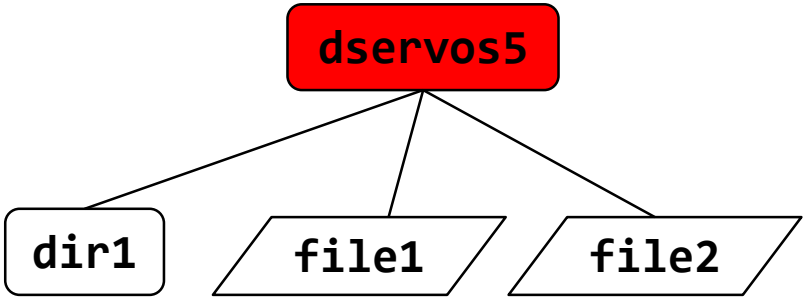
- The **`touch`** command can create files.
- Each argument is a file that will be created.
- **Example:**




```
[dservos5@cs2211b ~]$ touch file1 file2
[dservos5@cs2211b ~]$ ls
dir1   file1  file2
```

Making, Copying and Moving Files

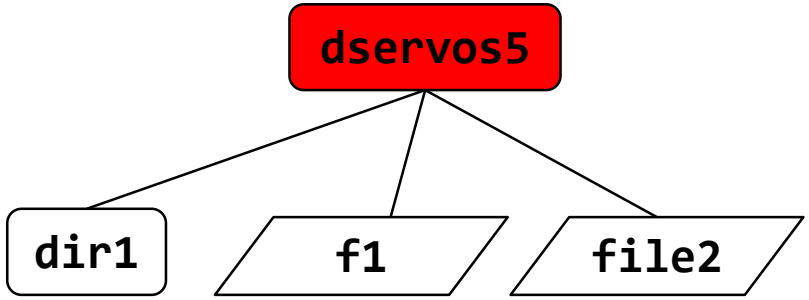
- The **mv** command can move or rename files.
- **Example 1:** Rename file1




```
[dservos5@cs2211b ~]$ mv file1 f1
```

Making, Copying and Moving Files

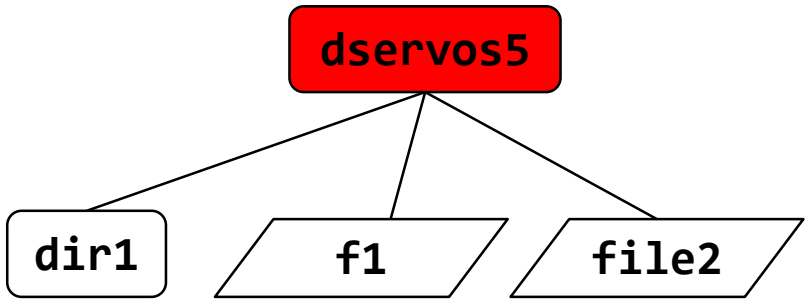
- The **mv** command can move or rename files.
- **Example 1:** Rename file1




```
[dservos5@cs2211b ~]$ mv file1 f1
[dservos5@cs2211b ~]$ ls
dir1   f1  file2
```

Making, Copying and Moving Files

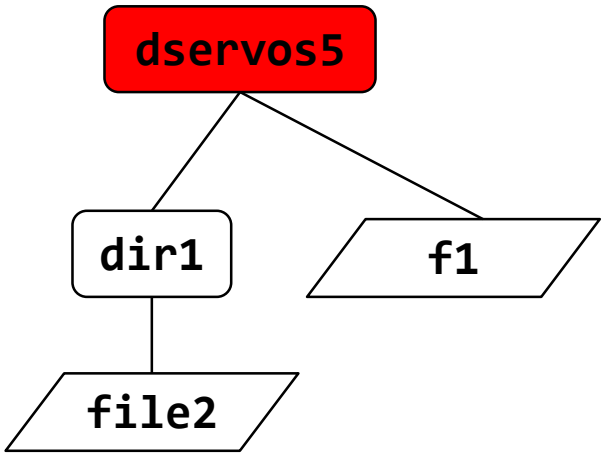
- The **mv** command can move or rename files.
- **Example 2:** Move file2 into dir1




```
[dservos5@cs2211b ~]$ mv file2 dir1
```

Making, Copying and Moving Files

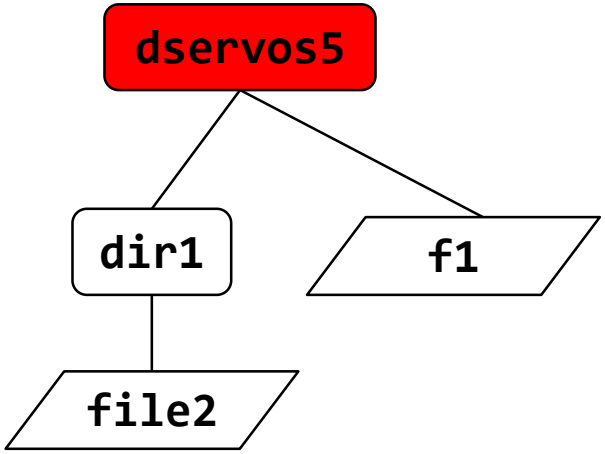
- The **mv** command can move or rename files.
- **Example 2:** Move file2 into dir1




```
[dservos5@cs2211b ~]$ mv file2 dir1
[dservos5@cs2211b ~]$ ls
dir1   f1
[dservos5@cs2211b ~]$ ls dir1
file2
```

Making, Copying and Moving Files

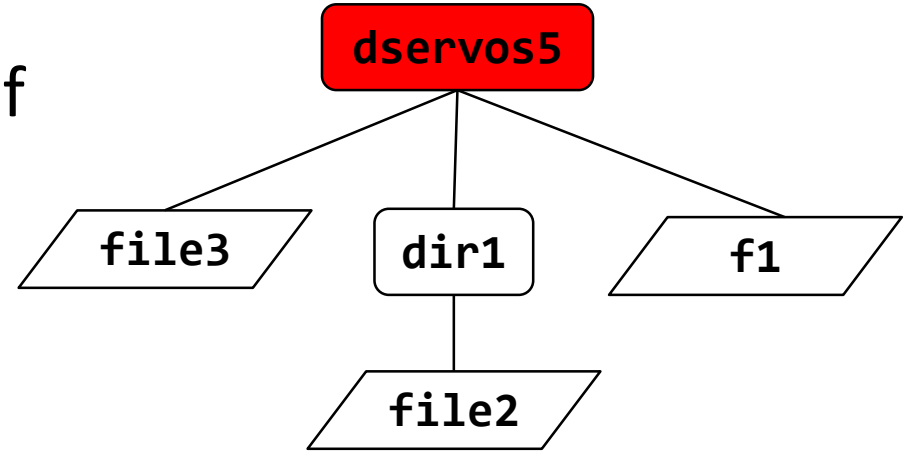
- The **cp** command is used to copy files.
- **Example 1:** Make a copy of f1 in the same directory




```
[dservos5@cs2211b ~]$ cp f1 file3
```

Making, Copying and Moving Files

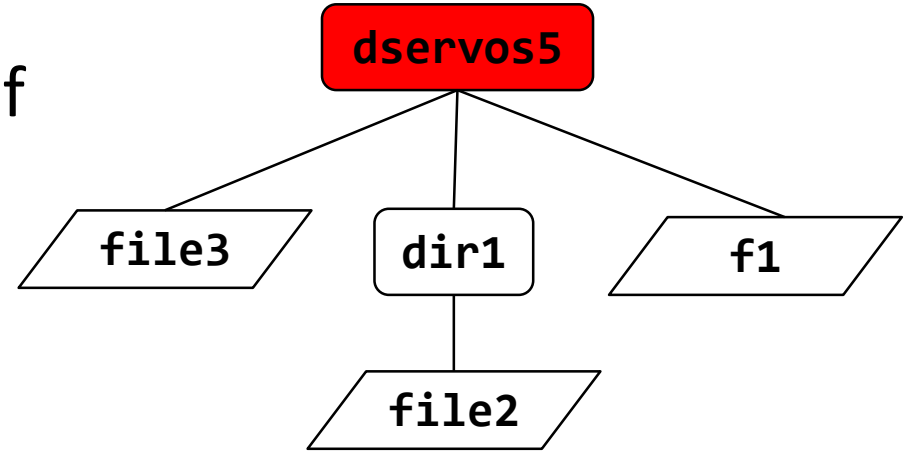
- The **cp** command is used to copy files.
- **Example 1:** Make a copy of f1 in the same directory




```
[dservos5@cs2211b ~]$ cp f1 file3
[dservos5@cs2211b ~]$ ls
dir1   f1  file3
```

Making, Copying and Moving Files

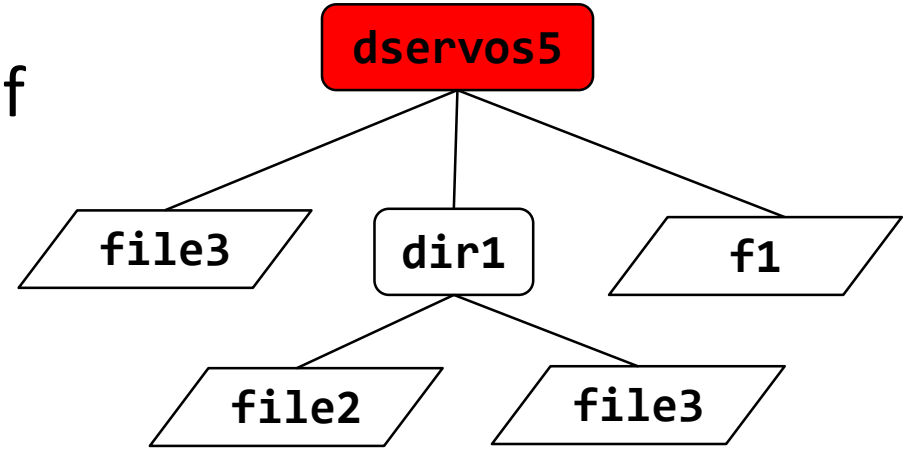
- The **cp** command is used to copy files.
- **Example 2:** Make a copy of file3 in dir1




```
[dservos5@cs2211b ~]$ cp file3 dir1
```

Making, Copying and Moving Files

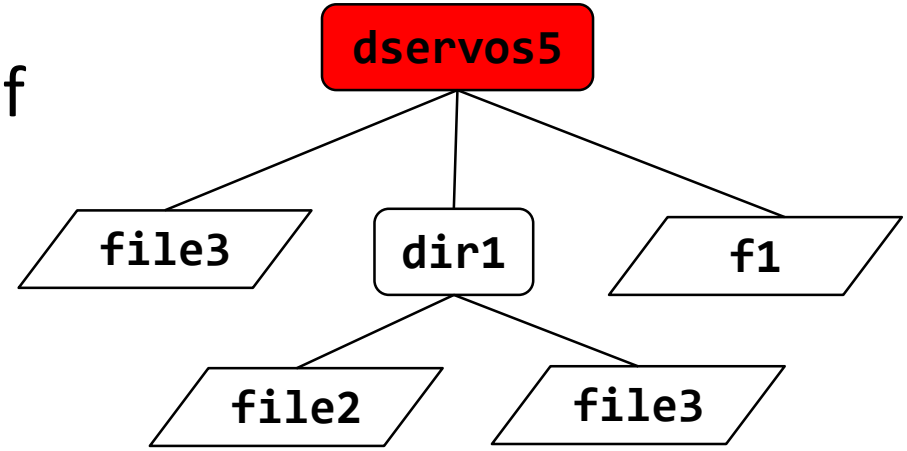
- The **cp** command is used to copy files.
- **Example 2:** Make a copy of file3 in dir1




```
[dservos5@cs2211b ~]$ cp file3 dir1
[dservos5@cs2211b ~]$ ls dir1
file2  file3
```

Making, Copying and Moving Files

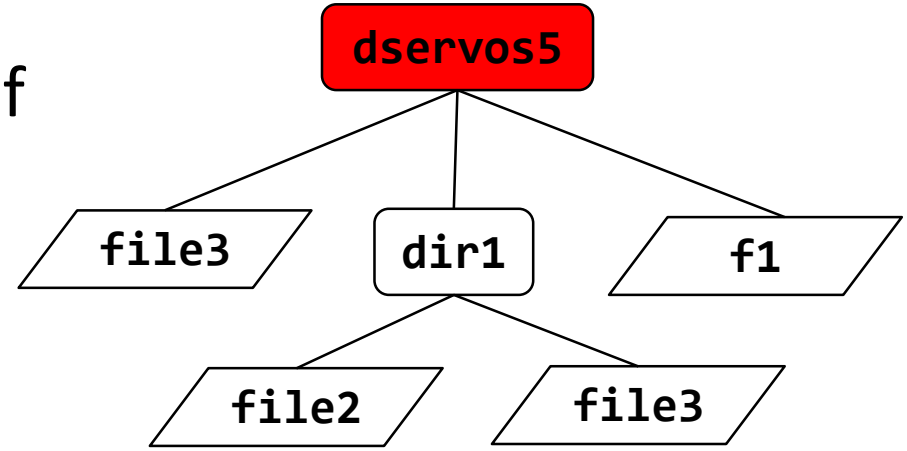
- The **cp** command has an –r (recursive) option.
- **Example 3:** Make a copy of dir1 and its contents




```
[dservos5@cs2211b ~]$ cp dir1 dir2
```

Making, Copying and Moving Files

- The **cp** command has an –r (recursive) option.
- **Example 3:** Make a copy of dir1 and its contents

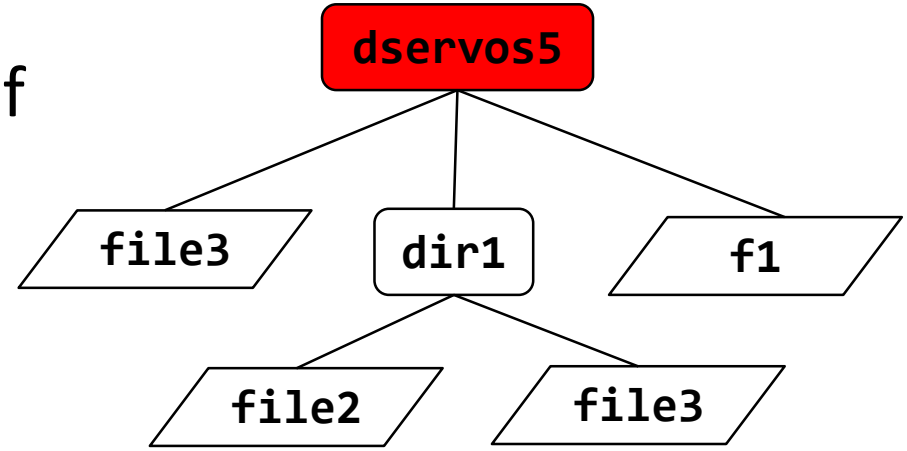



```
[dservos5@cs2211b ~]$ cp dir1 dir2
cp: omitting directory ‘dir1’
```

Need to use -r option!

Making, Copying and Moving Files

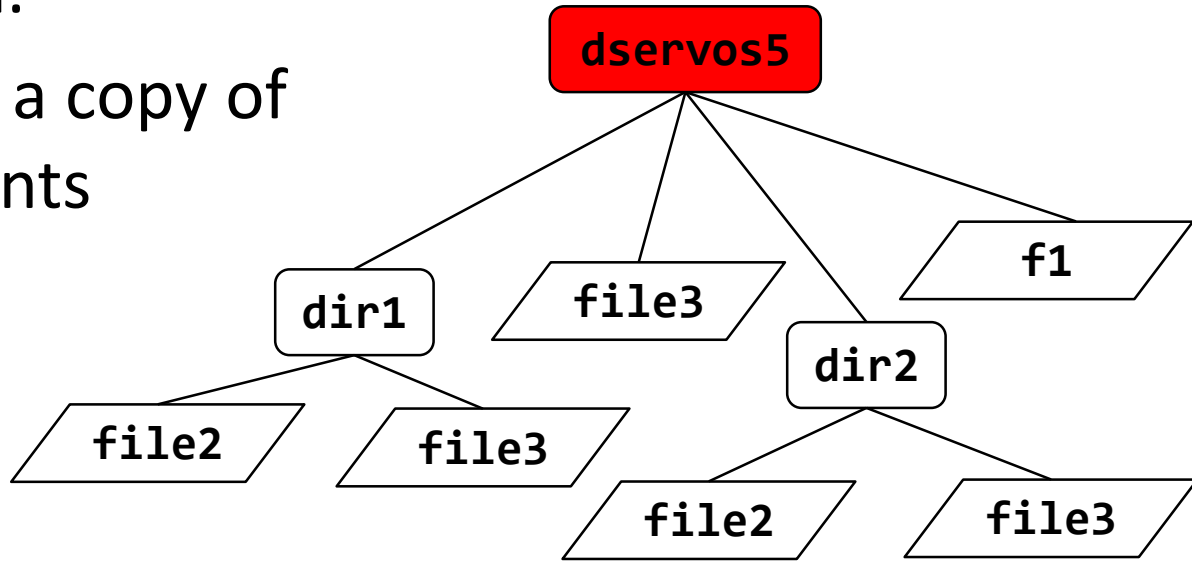
- The **cp** command has an –r (recursive) option.
- **Example 3:** Make a copy of dir1 and its contents




```
[dservos5@cs2211b ~]$ cp –r dir1 dir2
```

Making, Copying and Moving Files

- The **cp** command has an –r (recursive) option.
- **Example 3:** Make a copy of dir1 and its contents




```
[dservos5@cs2211b ~]$ cp –r dir1 dir2
[dservos5@cs2211b ~]$ ls
dir1 dir2 f1 file3
[dservos5@cs2211b ~]$ ls dir2
file2  file3
```

Making, Copying and Moving Files

- More details on **`rm`**, **`rmdir`**, **`mkdir`**, **`cp`**, and **`mv`** in your UNIX textbook (Chapter 3).

Displaying Files

- The **`cat`** command will output a file to the terminal:

```
[dservos5@cs2211b ~]$ cat test.txt
1
2
3
4
5
6
7
8
9
10
11
```

- In this case, *test.txt* contains the numbers 1 to 11 on new lines.

Displaying Files

- We can display the last lines of a file using the **`tail`** command:

```
[dservos5@cs2211b ~]$ tail -4 test.txt
8
9
10
11
```

- The **`tail`** command takes the option `-n` where n is a whole number, the number of lines from the end of the file to print.

Displaying Files

- Similarly, we can display the first lines of a file using the **head** command:

```
[dservos5@cs2211b ~]$ head -4 test.txt
1
2
3
4
```

- The **head** command also takes the option `-n` where n is a whole number, the number of lines from the end of the file to print.

Wildcards

Wildcarding is the use of “special” characters to represent or match a sequence of other characters.

- A short sequence of characters can match:
  - A long file name
  - Many file names

Wildcards

Wildcard characters include:

- * matches a sequence of zero or more characters
  - Example: a*.c* matches abc.c, abra.cpp,
- ? matches any single character
  - Example: a?.c matches ab.c, ax.c, but not abc.c
- [...] matches any one character between the braces
  - Example: b[aei]t matches bat, bet, or bit, not baet

Within [...], a pair of characters separated by "-" matches any character lexically between the two.

Example: [0-9]* matches any file starting with a number.

Wildcards

Wildcard sequences can be combined:

```
mv a*.[ch] cfiles/
```

mv all files beginning with a and ending with .c or .h into the directory cfiles.

```
ls [abc]*.?
```

List files whose name begins with a, b, or c and ends with . (dot) followed by a single character.

Wildcards do not cross "/" boundaries.

Example: `dservos5*1` does not match `dservos5/dir1`

Wildcards

Wildcards are expanded by the shell, and not by the program.

Example:

```
echo *
```

Outputs a list of files in the current directory as the * is replaced with the list of files and they become arguments to the echo command.

Wildcards

Example:

```
echo *
```

If the current directory contains the files: f1, f2 and f3 this command is equivalent to:

```
echo f1 f2 f3
```

And outputs:

```
f1 f2 f3
```

Wildcards

Examples

Delete any file with a .c or .sh file extension.

```
rm *.c *.sh
```

List any files starting with a P.

```
ls P*
```

Move any files with 3 letter names into dir3.

```
mv ??? dir3
```

Wildcards

Examples

List any files that have a vowel as the third letter.

```
ls ??[aeiou]*
```

Delete any files that end in a 9.

```
rm *9
```

List any files with the word “cat” in them.

```
ls *cat*
```