IT 1100 : Introduction to Operating Systems

Finding Files

Follow Along

Log into scratch and follow along with this slide.

Locate Programs and Files by Name

The `locate` command allows us to find programs, directories and files by name. The locate program performs a rapid database search of pathnames and then outputs every name that matches your search string. If I want to find the location of the `/bin/zip` file, I should type the following.

```
locate /bin/zip
```

You can even locate your own files:

```
locate fruit/apple/sauce.txt
```

You can also combine it with grep to filter your results or expand your results.

```
locate zip | grep bin
```

```
locate fruit | grep sauce
```

Notice how it searches the entire computer every time and brings up every instance of fruit and sauce that it can find. To limit this to only your own files add another `grep` with your username.

```
locate fruit | grep sauce | grep jdoe
```

You can also refine your search to only show sauce.txt:

```
locate fruit | grep sauce.txt | grep jdoe
```

Locate is a very fast search engine. However, it has a drawback. Just after you create a file, locate cannot find it. The system administrator or someone with superuser powers must run `updatedb` to update the database so that `locate` can find it.

On most machines `updatedb` will automatically run on a regular basis such as every night at midnight.

Finding files with the find command

To find files at any time no matter when they were created, we use a slower yet more powerful search tool called `find`. `find` searches every file in our search location – at the time we use the command. It will be quick on small servers such as `scratch`. But on larger computers with more files this can take several minutes. It's worth the effort though because of the power that `find` has.

We will learn the basics of `find` now and return to the power of `find` later in the semester.

The find command is one of the few commands with a unique format. To find files by name we use the format:

```
command argument -expression expression_argument
```

It must be in this order or it will not work and it must have all four to produce the expected results.

```
find search_location -name 'filename'
```

To find the `/bin/zip` file from above, we have to specify an exact search location and an exact name. The quotation marks are not required and either single or double quotations are acceptable. However, I highly recommend getting used to using quotation marks because they help you avoid errors.

```
find /usr/bin -name "zip"
```

Well that's no fun - where's the power? What if I don't know the exact search location and the exact name?

And if I knew the exact location and the exact name I wouldn't have to "find" it.

Well then, if you don't know where it is - you have the option of searching the entire computer from the root directory `/`.

```
find / -name zip
```

Took a while didn't it? That's because it had to search every file on the computer.

And what about all those Permission denied's? This doesn't seem so helpful yet, does it. When we search from the root `/` of the computer it literally searches every single directory and file on the computer. As a Standard User we don't have permission to see everywhere so we get a lot of permission denied's during our search.

To get rid of the Permission denied's you need to `redirect your errors to /dev/null`. This is a very handy way to not see the errors that we have no interest in.

```
find / -name zip 2> /dev/null
```

It still takes a while, but we only see the relevant results.

What if we wanted `gunzip` but couldn't remember the exact name of it. In that case we can use wildcards `*` to search for anything with 'zip' anywhere in the name.

```
find / -name '*zip*' 2> /dev/null
```

Now we've got too many results again. We are seeing files from everywhere. However, we know that it is in a directory called `bin`, we just don't know which `bin` directory. So we add a `grep`.

```
find / -name '*zip*' 2> /dev/null | grep bin
```

We can also refine our search to search directly in the `/bin` directory instead.

```
find /bin -name '*zip*' 2> /dev/null
```

Notice the different results between the two above commands.

This is also useful for finding files we just created, to make sure of our absolute path, or to check our work.

```
find / -name '*sauce*' 2>/dev/null
```

But searching the entire computer turns up so many results and we know that we created the file in our **$HOME** directory. By changing the search location to `~` we can search only our own files. In our **$HOME** directory there should not be any errors to redirect.

```
find ~ -name '*sauce*'
```

We can even refine our search to an exact filename.

```
find ~ -name 'sauce.txt'
```

Linux is case sensitive - so if our file has a capital letter in the name, a search for `sauce.txt` won't find `Sauce.txt`. To fix this we use the `-iname` option to denote case insensitive.

```
find ~ -iname '*sauce.txt'
```

This is just the beginning for the `find` command. It has such great power. We will learn more about `find` later in the semester.

Textbook Time

There is no textbook reading for this section
