Using Commands

Introduction to Unix

May 10, 2009
Cairo, Egypt

Patrick Okui

GUIs and CLIs

What's are some example GUIs?

- Windows
- Mac OS X (Darwin, X and Aqua)
- Gnome, KDE (on XWindow)

What about example CLIs?

- DOS
- CP/M
- Unix/Linux shells (bash, csh, sh, tsh, etc.)

Q. What's faster...?

A **G**raphical **U**ser **I**nterface: **GUI**?

A **C**ommand **L**ine **I**nterpreter: **CLI**?

A. For many (most) operations a CLI.

Q. What's hard, or slower to do in a CLI?

A. Copying multiple, unique files.

A. Other?

The format of a command

`command [options] parameters`

"Traditionally, UNIX command-line options consist of a dash, followed by one or more lowercase letters. The GNU utilities added a double-dash, followed by a complete word or compound word."

Two very typical examples are:

```
-h
--help
```

and

```
-v
--version
```

Command parameters

The ***parameter*** is what the command ***acts on***.

- Often there are multiple parameters.
- In Unix uppercase and lowercase for both options and parameters matter.
- **Spaces** are critical.

  "**`-- help`**" is wrong.

  "**`--help`**" is right.

Some command examples

Let's start simple:

- Display a list of files:
  - `ls`
- Display a list of files in a long listing format:
  - `ls -l`
- Display a list of all files in a long listing format with human-readable file sizes:
  - `ls -alh`

Some command examples cont.

Some equivalent ways to do "`ls -alh`":

```
ls -lah
ls -l -a -h
ls -l --all –human-readable
```

Note that there is no double-dash option for "`-l`".

You can figure this out by typing:

```
man ls
```

Or by typing:

```
ls --help
```

Where's the parameter?

We typed the "`ls`" command with several options, but no parameter. Do you think "`ls`" uses a parameter?

What is the parameter for "`ls -l`"?

It is "." -- our current directory.

"`ls -l`" and "`ls -l .`" are the same.

We'll discuss files and directories in our next section.

A disconcerting Unix feature

If a command executes successfully and there is no output returned from the command execution *this is normal*.

That is, if you type:

```
cp file1 file2
```

The result is that you get your command prompt back. *Nothing means success*.

Let's give this a try...

A disconcerting Unix feature cont.

Try doing the following on your machine:

```
# cd                    [cd = change dir]
# touch file1           [touch = create/update]
# cp file1 file2        [cp = copy]
```

The “#” indicates the command prompt. A “#” usually means you are the *root* user. A “$” for the command prompt indicates a normal user.

Using pipes

In Unix it is very easy to use the result of one command as the input for another.

To do this we use the pipe symbol “|”. For example:

```
ls -l /sbin | sort
ls -l /sbin | sort | more
```

What will these commands do?

Take advantage of the command line

The command line in Unix is *much more powerful* than what you may be used to in Windows.

- You can easily edit long commands
- You can find and recover past commands
- You can quickly copy and paste commands.

Your mission

Should you choose to accept it...

- Pay close attention to options and parameters.
- Use "`man command`" or "`command --help`" to figure out how each command works.
- A command, generally, acts upon it's parameter or parameters based on the options you give to the command...