Speech Tools Minicourse Week 1: WSL, vim, and bash

Mark Hasegawa-Johnson

Outline

- WSL
- vim
- bash

WSL

- Windows subsystem for linux (WSL) is a linux machine that runs inside your windows machine.
  - WSL1 is a translator – it translates linux commands into windows operating system commands. This allows you to basically use the bash window as a way of moving things around in your windows machine. This is what I use.
  - WSL2 is a virtual machine. Some open-source code that doesn’t work in WSL1 might work in WSL2. The disadvantage is that windows files aren’t available from inside linux; you need an internet connection between your windows machine and your linux machine.

Install

Just follow the directions on https://docs.microsoft.com/en-us/windows/wsl/install-win10

1. Hit the windows key to get a search bar, search for “powershell”. Right-click, and choose “run as administrator.” Then, on the web page above, choose the ”copy” button to copy the command, go to your powershell window, use control-v to paste, hit enter. Wait a little.
2. Back on the web page above: scroll down to “Install your Linux distribution of choice.” Click the link for “Ubuntu 18.04 LTS” and install it.

Get a window

Now get a bash window.

- If you're on Windows: ⊞-s, search "bash", choose "Bash on Ubuntu for Windows".
- If you're on Apple: ⌘-space, search "terminal," hit return.

Five commands you need to know:

- ls – list the contents of the current directory
- pwd – print the working directory
- mkdir foo – make a directory called "foo"
- cd foo – change to the directory foo
- cd .. – change to the next higher directory

Where am I, though?

- On Apple, the terminal first opens (by default) in a logical place: your home directory (/Users/jhasegaw for me).
- On Windows, not so much… the default home directory under Linux might be in any one of two or three different places, depending on version and what year it is. I recommend that you create a “symbolic link” to a directory you know, e.g.,

```
>> ln –s /mnt/c/Users/Mark/Documents docs
>> ls –l docs
```

… then go there, first thing, every time you start linux:

```
>> cd docs
```

Outline

- WSL
- vim
- bash

vim

- vim (Bram Moolenaar, 1993) is the most widely-used extension of vi (Bill Joy, 1976).
- vi was written to be useful even if you're trying to edit a file over a very slow telephone modem.
- The reason you need to know vim is that every unix machine has either vi or vim installed, usually vim. So even if you've logged in to a machine with no friendly programs, you can still edit programs on that machine.
- Here's a good tutorial: https://webhostingprof.com/vim-text-editor/
- Here's the standard documentation: https://vimhelp.org/quickref.txt.html

vim modes

Instead of just having one mode where you enter text, vim has five modes. You really really need to know these two:

- Normal mode. When you hit a key, it issues a command. Hit “i” in order to enter…
- Insert mode. When you hit a key, it actually enters the corresponding text into the document you’re editing. Hit “escape” in order to go back to normal mode.

vim movement commands

When you’re in “normal mode,” you can enter commands. Here are the first ones that vim normally teaches people. I never use these – I just use the arrow keys on my keyboard instead – but these might be useful if you’re ever stuck without arrow keys.

- h – move left
- j – move down
- k – move up
- l – move right

vim useful commands

Here are the commands that I actually use, in normal mode:

- dl – delete one character
- dd – delete the whole line
- i – insert before the current character
- a – insert after the current character
- :wq – write the file, and exit
- :q! – quit without saving (useful if you screw up)
- :undo – undo the last command (also useful if you screw up)

I just learned these two commands today, they look useful:

- y$ - copy everything from the cursor to the end of the line
- p – paste whatever you most recently copied

vim practice

OK, now try it:

```
>> vim hello.txt
```

- Modify the file so it says “hello there!”
- Save it.
- Type “more hello.txt”. You should see the content that you just created.
- more about more: https://en.wikipedia.org/wiki/More_(command)

Outline

- WSL
- vim
- bash

why do you need to know bash?

- Just as vim is vi improved, bash (Brian Fox, 1989) is a bourne-again sh (Stephen Bourne, 1979).
- Working with bash is exactly like working in the interpreter window of python, except that:
    1. bash is much easier to use for running programs – the syntax for running programs, listing files, changing directories and so on is much easier than python.
    2. bash is harder to use for object-oriented programming, defining subroutines, and so on. It’s possible, but the syntax is bad.

How to learn bash scripting

- The only way to learn bash scripting is by examples.
  - Here's a good initial tutorial: https://linuxhint.com/30_bash_script_examples/
  - This is the best one I've seen, well organized: https://devhints.io/bash
  - …and, of course, https://github.com/kaldi-asr/kaldi
- You will also occasionally want to look at the manual pages, in order to check what the creators say about a particular functionality
  - bash: https://www.gnu.org/software/bash/manual/bash.html
  - This one is confusing. The character "[" in a bash script is actually an alias for the command "test": http://man7.org/linux/man-pages/man1/test.1.html
  - Other commands that are especially useful for scripting are seq, bc, cut, head, tail. These each have their own separate man pages at http://man7.org/linux/man-pages/man1/

bash practice

In your terminal window, type

```
>> echo “hello”
```

You should see “hello”. Now type

```
>> cd docs
>> vim sayhello.sh
```

Use your vim knowledge to make this file contain the following line:

```
echo “hello”
```

Then close it, and type

```
>> ./hello.sh
```

or try

```
>> bash hello.sh
```

bash scripts

- A bash script is a program, containing bash commands.
- You can edit it in any text editor. Open your favorite editor (Word, even), and open the file “hello.sh” in your Documents folder. Edit it so it says

```
echo “hello, this is a new and improved version of hello.sh!”
```

Then save it (be sure to save it as plain text!!!), go back to your linux window, and type

```
>> ./hello.sh
```