OS and Command Line

Using the Command Line

Why Command Line

1. It works everywhere
2. You don't have to re-train for every tool
3. Continuous Integration
4. It is scalable (has loops, scripting etc)
5. It works on tiny systems (IoT)
6. It is much faster for most development processes

Linux / Max / Unix

Downlaod a file - it shows up in you ./Downloads directory.

Bring up "Terminal" or install and use "iTerm2" (I recommend iTerm2) you start in your "Home" directory.

On Linux/Unix/Ubuntu/Red-Hat etc - you Start a terminal. In each of these cases you end up at a "prompt" to type something in:

```
$
```

Commands

All of these commands work on Mac / Linux and in Windows PowerShell.

List the files in the current directory:

```
$ ls
```

Mac/Window - list with attributes and size:

```
$ ls -l
```

Make a directory to work in:

```
$ mkdir hw1
```

Make the directory your current working directory:

```
$ cd hw1
```

See where you are in the directory tree:

```
$ pwd
```

Go up 1 directory

```
$ cd ..
$ pwd
```

Go back into the hw1 directory

```
$ cd hw1
$ pwd
```

Rename or move a file (like the ones that you downloaded): `~` refers to you home directory. `.` refers to the current directory. (Note that `..` refers to the parent of the current home directory)

```
$ mv ~/Downloads/Asm .
```

Mac/Linux requires that a file be “executable” to be run. When you download the file it is not executable. To fix this:

```
$ ls -l ./Asm
$ chmod +x ./Asm
$ ls -l ./Asm
```

Now we “may” be able to run the program if the OS is tweaked into it. On Linux - no tweaking is needed. On Mac we need to run the program and then tell it that it is Ok to run programs from non-trusted sources.

```
$ ./Asm --in InputFile.mas --out Output.hex
```

You can setup a directory where the "shell" (bash/zsh) will look for your commands. Google for "setting your PATH".

To see what is in the file:

```
$ cat Output.hex
```

PowerShell / Windows

The prompt is different.

```
H:\>
```

Lot's of commands don't have any useful options - Example.

Very few commands to use.

You have to install and configure everything.

Virus software may need to be turned off to get these programs to run - I am working on that.

On 4th floor of Engineering (4072) there is a lab where you can use PCs and do this stuff.
That is where I tested.

In the PowerShell the following commands work sort of the same as in Linux(Posix)

```
ls
mv
pwd
cd
```

Instead of "cat" use "type".

Windows is missing the concept of an "executable" program so there is nothing like "chmod".

What most persons see

1. User Interface (input)

    - web
    - gui
    - command line
    - binary input to a computer
    - electricity / on / off
2. User Interface (output)
    - reports
    - printouts
    - control of systems
    - screen

A Single Programming Language

1. The “categories” - the “model” that the program uses. Von Neumann Fetch Execute
2. Imperative is the most common
3. Object oriented
4. Functional
5. Concurrent (real concurrency) - Microcode instructions
6. DFA - Deterministic Finite Automata
7. Push-Down Acceptor: LALR(1)
8. Stack Architecture Machine (forth, HP-41 calculator, Ethereum and Bitcoin)

Inputs change “state” in a program - that then results in the possibility of new “state”.

Multiple Languages Interacting

HTML on a web page, Cascading Style Sheets (CSS), with JavaScript - talking to a remote system in some language like Python and SQL (database).

Simple C Program

K&R C or Early ANSI C:

```
#include <stdio.h>
int main() {
        putc('a',stdout);
}
```

More recent ANSI C

```
#include <stdio.h>
int main(int argc, char *argv[]) {
        putc('a',stdout);
}
```

1. System software like a “compiler” that turns our “language” into what the machine can run.
2. An Operating system to run the program.
3. Connection for the program to input/output.
4. Memory.
5. A loader and a file system.
6. Load - and Run The Code.
7. Display the output.

Assembly Language Code

```
/ Program to output 'a' as a number
        Load X
        Output
        Halt
X,      DEC 97
```

And in Hex we get 0061 - is that an ‘a’? Why?

ASCII

Representation

What is “HEX”.

There are 10 kinds of people. Those that understand binary and those that don’t. Either you already understand binary or you are going to before the end of this class.

Binary Representation of Program

In hex:

```
1003
6000
```

```
7000
0061
```

Electrical Representation of Program

Memory

Layers of Storage

Gates

Microcode Architecture

Boolean logic

Copyright