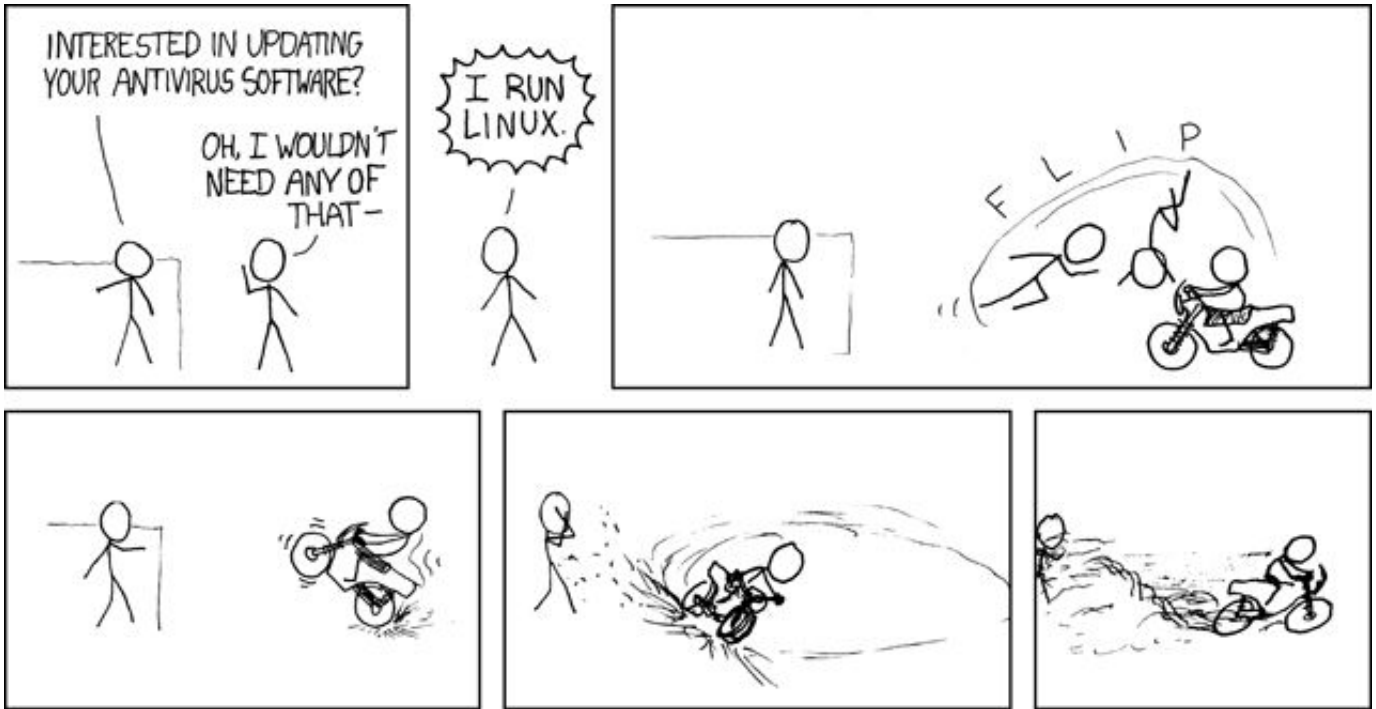
4 - Operating systems and Linux!



Recap & Agenda

Last we talked about:

- Haskell types and functions
  - We rewrote functions in the standard library using pattern matching
- How logic gates are used to make the “brain” of your computer
  - Arithmetic and logic unit (ALU)

Today, we’re going to talk about:

- What happens when you turn a computer on?
- Operating systems
  - Booting Linux in your current computer! Woohoo!
- More haskell functions if we have time.

What happens when you turn on a computer?

We've talked about physical components (hardware) of a computer, but what happens when you turn a computer on? Let's find out!

To start:

Your group is tasked with identifying what happens when you turn on your computer. To begin answering this question, your group will:

1. reset one member's laptop,
2. record your observations on a timeline.

*__Listen__ well and __watch__ the computer carefully! Be sure to *note the time* when certain changes took place on your timeline.

What happens when you turn on a computer? P. 2

With your group reassembled, combine your knowledge. Those taken to the other room, get caught up with your groupmates findings and be sure to teach your groupmates what you've learned!

On your timeline, identify the following steps:

1) Process loads BIOS
2) BIOS loads bootloader from hard drive
3) The bootloader runs the Kernel
4) Kernel launches the operating system
5) Operating system is ready for user input (end)

OS? What even is an OS?

This video will do a way better job explaining it than we can.

Linux

An operating system is just an example of a program (with special privileges). Today we're going to try a totally new operating system, Linux!

Released in the early 90s, Linux is completely free and open source! It is the product of decades of work from a whole bunch of different volunteers at different times.

Linux has different *distributions*, which are different software suites that come with the Linux kernel. The distribution we will use is *Arch Linux*!

We will be using Linux to write your website's code and host it on the B21 server!

Let’s start!

Like the crash course video showed, we use Linux using a *terminal*, where we enter commands!

Let’s learn some basic syntax.

Open VirtualBox and start Arch! There we have a prepared a scavenger hunt for you to help you learn the basic syntax.

First step is to log in

Username: student

Password: password

Scavenger hunt

First, you need to `cd` into the scavenger hunt directory. `cd` stands for “change directory”, and a directory is just a fancy word for folder.

Now, enter the command `ls`. `ls` stands for “list” which will list the contents of a directory on your terminal.

This is all you need to know to start your hunt! Good luck!

Reviewing the commands as a class

What do the following commands do?

- cd
- ls
- cat
- less
- mkdir
- cp
- mv

Enough linux, let’s coding!

First you’re going to need the compiler for Haskell! To do this, you’ll need to download GHCi:

```
curl -sSL https://get.haskellstack.org/ | sh
```

When this completes, run `stack setup`.

Ok this might take a while, so while this downloads let’s review some functions in the standard library of Haskell.

Remember long division? Me neither.

Last week we looked at the `div` function, which divides numbers (obviously). But how does it work? Let's look to the board.

Let's look at other functions in the standard library. Search for "useful haskell functions", clicking on the first search result.

Higher order functions

One of the most powerful things about Haskell is you can have **functions** as **input** or as the **output**.

Please direct yourself to the higher order function section of the site from earlier.

The first function we're going to look at it is `map`.

Let's look at its type signature by typing `:t map` in GHCi. What does it read?

Now that we know the signature, let's try and code it ourselves to understand how it works!

Let’s jump next to the filter function

filter is another important higher order function. Let’s see its type signature using :t filter.

This function **filters** a list according to the function a -> Bool. This function decides whether an element is “good” or “bad”.

For example, we can filter a list of numbers to extract only the even ones or the primes ones. That’s a function Int -> Bool. Or filter a list of strings to select only those that are properly capitalized. That would be a function String -> Bool.

Now let’s code it up in pairs!

Recap and future vision

Today we learned:

- About operating systems, and tried Linux!
- Higher order functions, like map

Next week, we'll:

- Take higher order functions to the next level with functors
- Write our first code in HTML to make our sites! It's happening!