Computer architexture layer

- Abstraction
- Computer Architecture Overview
- Kevin: Advice for the world of IT

Abstraction

When we interact with our computers, we use our mouse, keyboard, or even a touch screen. We don't tell it the actual zeros and ones, it needs to understand something. But wait, we actually do. We just don't ever have to worry about it. We use the concept of abstraction to take a relatively complex system and simplify it for our use. Use abstraction everyday in the real-world and you may not even know it. If you've ever driven a car, you don't use to know how to operate the transmission or the engine directly. There's a steering wheel, pedals, maybe a gear stick. If you buy a car from a different manufacturer, you operate it in pretty much the same way, even though the stuff under the hood might be completely different. This is the essence of abstraction. Abstraction hides complexity by providing a common interface. The steering wheel, pedals, gear stick engages in our car example. The same thing happens in our computer. We don't need to know how it works underneath the hood. We have a mouse and a keyboard we can use to interact with it. Thanks to abstraction, the average computer user doesn't have to worry about the technical details. We'll use this under the hood metaphor throughout the program to describe the area that contains the underlying implementation of a technology. In computing, we use abstraction to make a very complex problem, like how to make computers work easier to think about. We do that by breaking it apart into simpler ideas that describe single concepts or individual jobs that need to be done, and then stack them in layers. This concept of abstraction will be used throughout this entire course. It's a fundamental concept in the computing world. Another simple example of abstraction in an IT role that you might see a lot is an error message. We don't have to dig through someone else's code and find a bug. This has been abstracted out for us already in the form of an error message. Symbol error message like file not found, actually tells us a lot of information and saves us time to figure out a solution. Can you imagine if instead of abstracting an error message our computer did nothing and we had no clue where to start looking for answers. Abstraction helps us in many ways that we don't even realize.

Computer Architecture Overview

In the last video, I mentioned that people don't need to understand how a computer works for them to use it, because abstraction makes things simpler for us. That's technically true, but since you're stepping to the world of IT, you do need to understand all the layers of a computer and how they work. It's essential that you understand how the different pieces interact so you can resolve any issues that may arise. Computer can be cut into four main layers, hardware, operating system, software, and users. The hardware layer is made up of the physical components of a computer. These are objects you can physically hold in your hand. Laptops, phones, monitors, keyboards. You get the idea. In the next lesson, you'll learn all of the components of a computer and how they work. You'll even be able to build your own computer by the end of this module. The operating system allows hardware to communicate with the system. Hardware is created by many different manufacturers. The operating system allows them to be used with our system regardless of where it came from. In the next few lessons, you'll learn about the major operating systems that we use today and you'll be able to understand all of the underlying components that make up an operating system. By the end of these lessons, you'll have a strong grasp on the major components of any operating system like Android or Windows, and use that knowledge to navigate any operating system. The software layer is how we as humans interact with our computers. When you use a computer, you're given a vast amount of software that you interact with. Whether it's a mobile app, a web browser, a word processor with an operating system itself. Later in this course, we'll learn how software is installed on our systems and how we interact with different types of software. The last layer may not seem like it's part of the system, but it's an essential layer of the computer architecture. The user, the user interacts with the computer. The user layer is one of the most important layers we'll learn about. When you step into the field of IT, you may have your hands full with the technical aspects. The most important part of IT is the human element. While we work with computers every day, it is the user interaction that makes up most of our job from responding to user emails to fixing their computers. By the end of the course, you'll also learn how to apply your knowledge of how a computer works to fix real-world issues that can sometimes seem random and obscure. We'll do this by learning how to utilize problem-solving tactics to identify issues and solutions. There's a lot ahead. The next instructor you're going to meet as a friend of mine, Devin [inaudible] and I know there's no better person to teach you about hardware or even show you how to build a computer from its component parts. Pretty cool

Kevin: Advice for the world of IT

We have a lot of people that have nontraditional backgrounds that have made it here at Google. I've worked with people who have history degrees. I work with people who have economic degrees, and they're writing scripts, automating us on how we can process these credits for this client. I think people have a misconception that you have to have a traditional path in order to succeed in IT. A lot of people do follow the traditional path. A lot of people do succeed following that traditional path. But I think the benefit of IT is that in the end, people just want to know whether or not you can fix the problem. Make sure you have strong fundamentals. They do end up coming back. A lot of times people think that like, I'm not going to need to worry about how to do. Like, I don't need to know, understand the TCP IP model or the OSI model. That's like low-level stuff. I can focus specifically on this one particular application or program that I'm going to be dealing with. There are instances where you will run into problems were in having that foundational knowledge will be like very integral to solving the problem. As long as you're able to get to a point where you feel comfortable working with users, fixing their problems and supporting them in the best way for you and them, you're going to always be viable in the world of IT.