From Course Readiness survey:

Working from the Command Line

Understanding how to use the command line, particularly the Bash (Unix) shell, is a great asset for this course. If you have a firm grasp of how to navigate and use the command line, then the building and debugging of C programs -- especially when no IDE is prescribed -- can be made much easier. Also, this is particularly useful when sharing projects with source code management (SCM) tools such as Git or Subversion.
Concepts to Know
Minimally, being able to do the following tasks from the command line will be helpful for this course:

- •Read man pages (man)
- •Navigate directories (cd, ls, pwd)
- •Move and copy files (mv, cp)
- •Adjust permissions or groups (chown, chmod)
- •Run executables (gcc, C executables, or other tools)

Command Line Example

We have provided a short example of building and running the quintessential "Hello, World" program from the command line. If you can understand and follow the steps being taken here, then you should be prepared for most command line interactions in this course.

```
# navigate to home dir
$ cd ~

# create main.c file
$ touch main.c

# edit main.c (write program)
$ nano main.c

# compile (and link) program
$ gcc main.c -o helloWorld

# run program
$ ./helloWorld
```

Using Makefiles

When building large, maintainable C programs (with multiple entry points and shared code), the use of Makefiles is paramount. In this course, students must submit their projects using Makefiles to automate the build process of their C program(s). This will allow our graders to more quickly grade student submissions and provide timely feedback.

Concepts to Know

A basic understanding of Makefiles is all that is necessary for your own projects; however, the following concepts are also very useful:

Targets and dependencies
Comments (always useful!)
Variables (compiler, flags, etc.)
Calling make from the command line

Makefile Example

If you have never used Makefiles before, then here is a great tutorial explaining how they work (tutorial uses g++ instead of gcc).

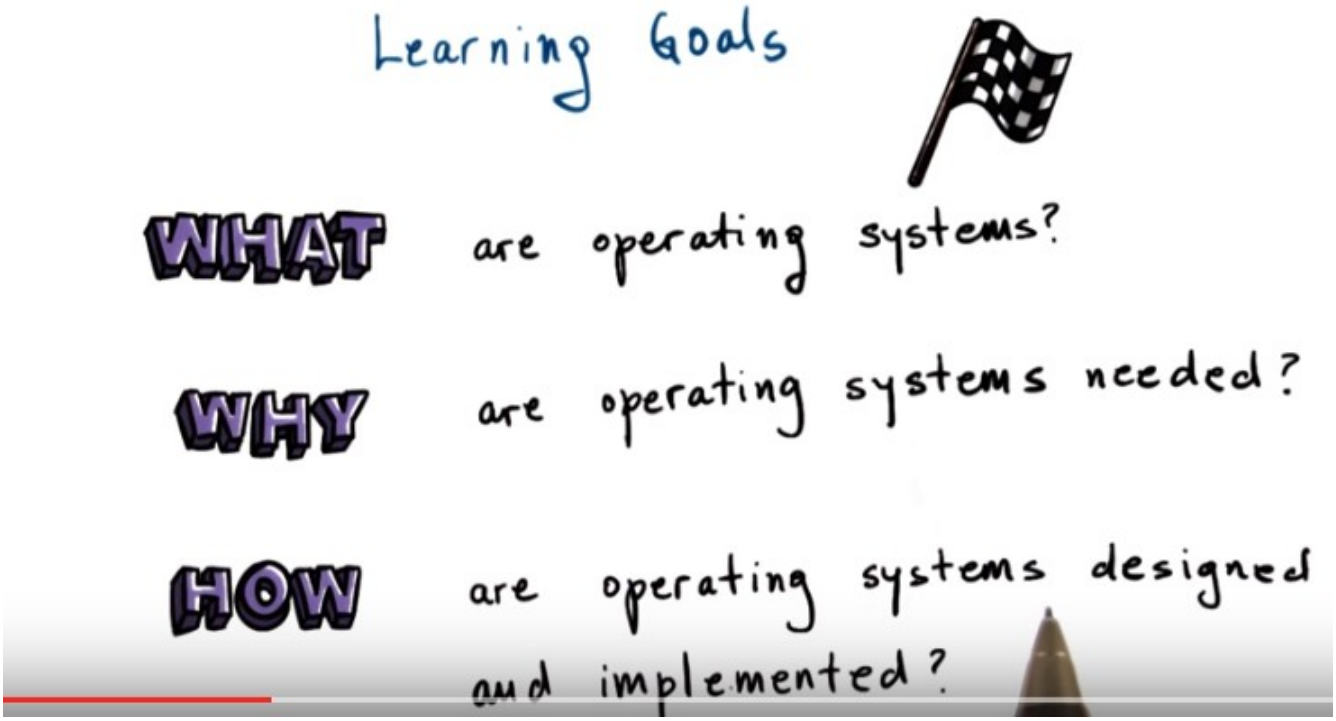
While more complex syntaxes are possible, we have limited the following Makefile example to something that is both simple and readable. If you can understand this example, then you should be capable of following the Makefile policy for the course's projects.

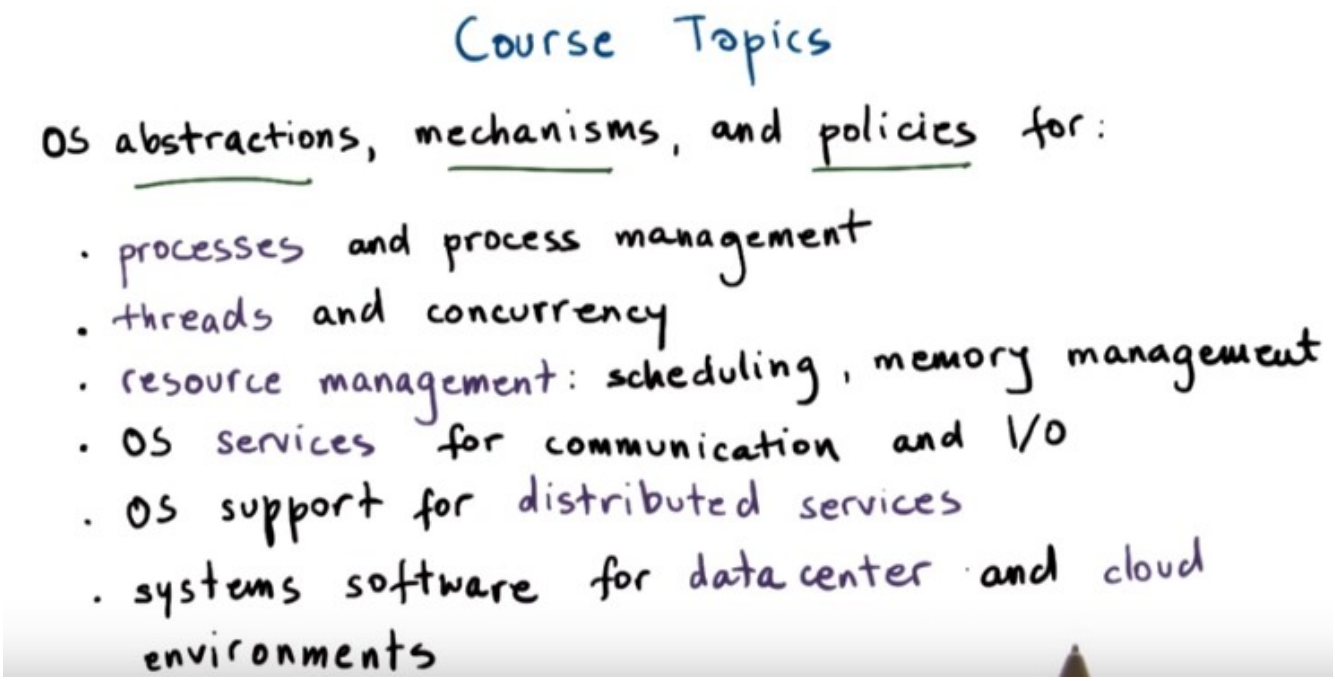
```
# specify the compiler
CC=gcc

# specify options for the compiler
CFLAGS=-c -Wall

all: hello

hello: main.o hello.o
   $(CC) main.o hello.o -o hello

main.o: main.cpp
   $(CC) $(CFLAGS) main.cpp

hello.o: hello.cpp
   $(CC) $(CFLAGS) hello.cpp

clean:
   rm -rf *o hello
```

Normal notes:

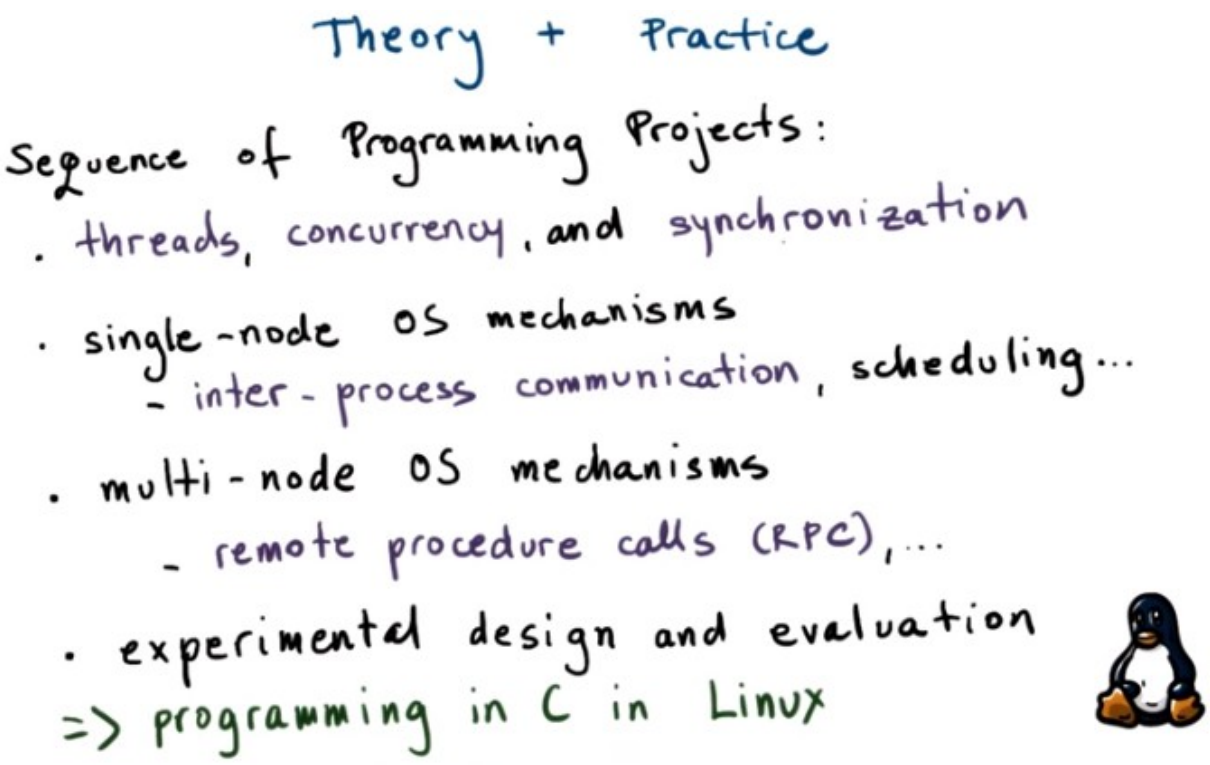
1. Welcome to Graduate Introduction to Operating Systems. My name is Ada Gavrilovska, and I will be your instructor for this course. I've been a research faculty at Georgia Tech since 2004 when I finished my PhD here. And in these years, I've been involved in and led a number of research projects in the area of systems that specialize on topics related to operating systems, virtualization, or other aspects of system software. In these years, I've also taught a number of times at Georgia Tech. This particular class, I've taught it over a dozen times, and over the years, we've evolved it to update it to reflect any changes in technology or any changes in the student interest or the student population in general. I've also been teaching the advanced operating systems class and the special topics class in high performance communication. For additional information about me, you can look at my website, and the link is provided off of the instructor notes.

2. Some of you may be asking yourselves why should you take this class. So, let me give you a little bit of context how this class came to be. Most of the masters and PhD students at Georgia Tech want to or need to take a, the Advanced Operating Systems class 60210, and likely, many of you will consider taking it in the future. But as the title of that class suggests, Advanced Operating Systems, that course deals with a number of advanced topics related to operating systems. However, over the years, the population of the graduate students at Georgia Tech has diversified a lot. And a lot of the students that come to us either don't come with formal operation systems training, so they may not have seen some of the more introductory topics related to operating systems, or maybe it's simply been a number of years since they've looked at that content. To address this gap, we offer this course, CS 8803 Graduate Introduction to Operating Systems. The intent with this course is to teach for the first time, or review, operating systems basics. In terms of metrics, once you finish this course, you should be able to answer every single one of the questions on the diagnostics test that's associated with the Advanced Operating Systems course.



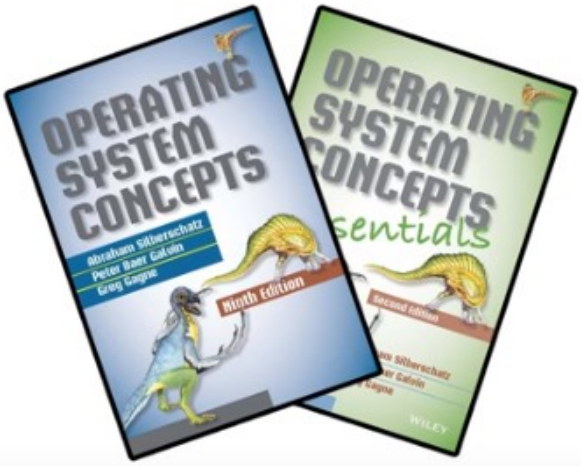


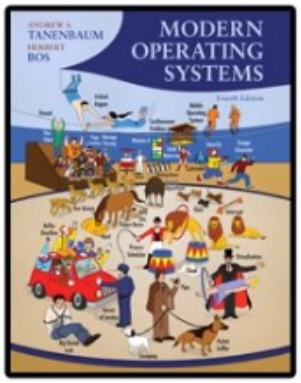
3. To begin talking about this course, let's look at the learning goals. In this course, you will learn what operating systems are, why are they needed, and how are they designed and implemented. This will include discussion of key mechanisms and abstractions that operating systems must support to provide the required functionality. In this course, we will cover a number of topics in depth. In particular, we will describe the design and implementation of operating systems' abstractions, mechanisms, and policies related to several of the key types of functionality provided by operating systems. This includes processes and process management, threads and concurrency, and in general, challenges related to multithreading. OS functionality for managing hardware resources like CPU and memory. OS services for communication and I/O, which includes file I/O. OS support for distributed services, including remote procedure calls. We call those RPCMs. Distributed file systems, distributed shared memory. And we will briefly review systems software that's common in data center and cloud environments.



4. To fully understand and appreciate OS internals, we will supplement the theoretical part of the course material with a practical component, and that will be a sequence of programming projects. The goal of these projects is to give you a hands-on experience with operating systems. For instance, you will be put face-to-face with challenges related to multithreading, concurrency, and synchronization. You will have to use operating system services for interaction among multiple processes on a single node, like a server machine. You will also have to look beyond the single node and prototype distributed, or multi-node services, using some of the mechanisms that are supported at the operating system level, like remote procedure calls. Finally, one key issue with operating systems design and implementation is the performance that they can deliver. You will gain experience with designing and performing experimental evaluations. These experiments will ideally provide you with insight into the behavior of the operating system so you can better understand its performance capabilities, overheads, limitations, or other issues. Each of the projects will require you to program in the C programming language in a Linux environment, and you will have to use some of the standard Linux libraries, such as Pthreads, for threads.

Recommended Textbooks

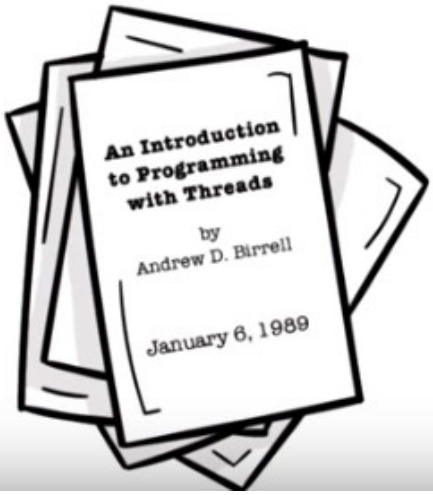


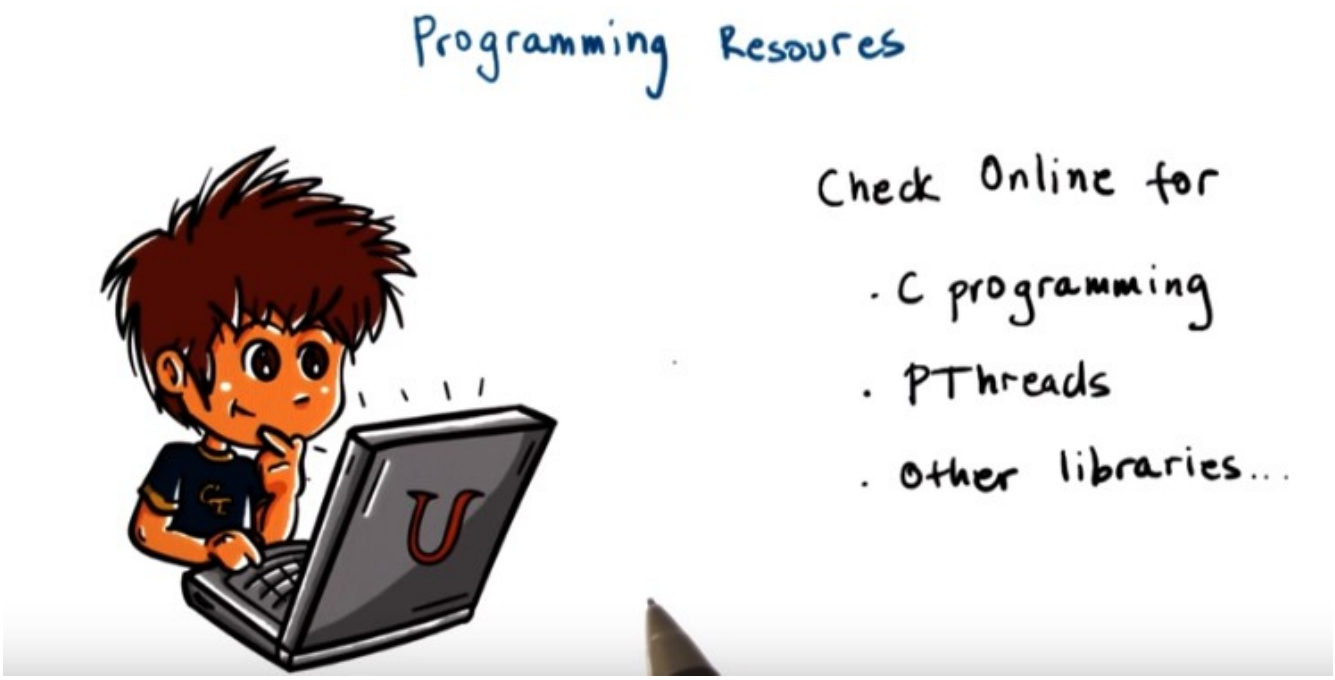


Papers and Other Resources

INSTRUCTOR NOTES or COURSE WIKI

- research papers
- tutorials
- technology surveys

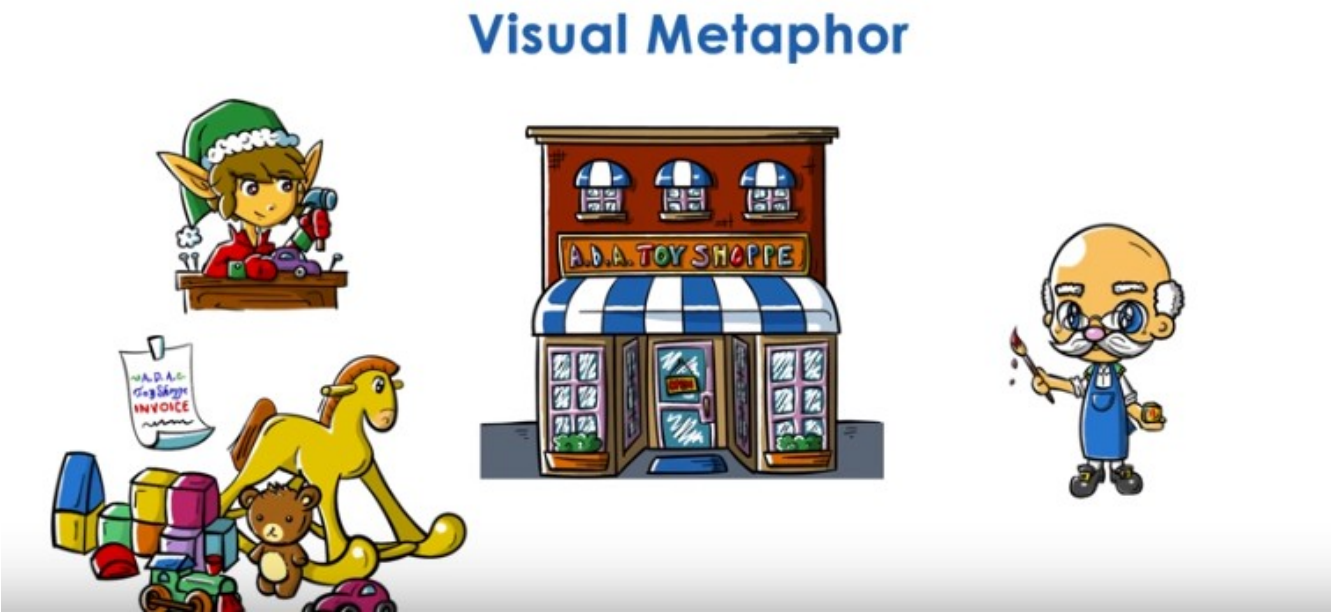




5. For the theoretical component of this course there's no required textbook. It is recommended, however, that you have access to some solid operating system textbook as an additional reference. There are a number of options out there from a variety of authors. And there are also many books that are available online, including with completely free content. My personal favorites are the Silberschatz operating systems books, often called the dinosaur books because of the dinosaur on the cover. There is one that includes more detail, Operating System Concepts, and then an abbreviated one, Operating System Concepts Essentials. I also like the Tanenbaum textbook called Modern Operating Systems. And to this one, I have personally contributed to the Linux case study chapter. If you look online, each of these books has a number of available online resources. There's some slides for the chapters. There's some free available text online. And there are also affordable Kindle editions that are for rent even. And there is also a lot of material that's freely available online. There's one recent title prepared by Andrea and Remzi Arpaci-Disseai. It's called Operating Systems Three Easy Pieces. This is a good book and its content matched fairly well with the content that we'll cover in this course. For certain lectures, I will also provide you with specific references from which the lecture content is drawn. Mostly these will be research papers, and some of them can be quite old. But they're still quite seminal works in computer science. Less often there will be some tutorials, manuals on specific OS technologies, or even technology surveys. In any case, all of these resources will be linked to in the Instructor Notes or in the Course Wiki online. Finally, regarding the programming aspect of this course, I said that you will be required to program in C in a Linux environment and that your programming assignments will require multi threaded programming. To help with these topics there are a number of resources online for C programming, for threading libraries such as PThreads. And we'll make sure to provide links to some such content with your programming assignment

6. Next, I want to make sure you're aware of the online wiki for this course. It has information such as the class syllabus, lecture schedule, project schedule, various resource links that are useful for the class. It also has grading criteria and other policies that are valid for the course. Let me now explain the grading policy in this class. Your grade will roughly be split between the theoretical and the practical part. The theoretical part, you will have two exams, a mid-term and a final. The final will not be cumulative and these will be worth 25 and 30% of your grade. Your projects will be worth 40% of

your grade, so these are your programming assignments. and you will have 5% of your grade come from class participation. Given that the class is online, for class participation, we expect you to participate in Piazza, the online forum that we'll use in this class. How well you answer to questions posted by other students, how often you raise an interesting and useful topics for discussion in the forum, and also how well you do on the quizzes on the Audacity platform that will be incorporated in the class lessons. You can always monitor your grades in T-Square. For the exams, we'll use an online service called ProcterU. As the name suggests, this is a service that proctors you during the examination and insures academic honesty. The tests are to be taken individually, and they will be closed notes, closed book exams. For the projects, you can work in teams. Across teams, you can have high level discussions about your design approach or implementation approach. But, you're not allowed to share code, results, text for the final report. You should have these discussions thru Piazza, so both the core staff, as well other students can help you.



7. Before we get started, you may have noticed these cute, wooden blocks on my desk. They belong to my daughter, but I'm borrowing them for this course. And during each lesson, I will illustrate some more complex concepts related to operating systems using a metaphor that's based on toys and a toy shop example. To give you a high-level idea of how this will work, we will think of the hardware in computer system as the toy shop with its tools and work spaces and storage areas. Then we will think of all of the processes and applications that execute on hardware as all of the toys that are being produced in the toy shop. And then central to this course will be that we will drawing be analogies between operating systems and the toy shop manager that oversees everything that's happening in the toy shop. This will become much clearer in the following classes. For now, this concludes the introduction. I sincerely hope you will enjoy this class. Good luck and have fun.

8. Here's your first quiz. We would like to know what your expectations are for this course. So please answer the following. What do you expect to learn in this course? Also please feel free to mention any of your prior experience, goals, or even fears about the class.