09/20/2022:

In today's lecture, we discussed the concept of threads, which are the flows of execution across a process. There are various components that are involved in this process, such as the program counter, system registers, and stack. The program counter keeps track of the next instruction to be executed, while the system registers and stack hold the current working variables. In the context of operating systems, we discussed the importance of context switching. When a process is removed from the processor, it must be saved so that it can pick up where it left off when it's later scheduled to run. This ensures that the process can continue its activity once it's run again.

In order for a process to continue where it left off when it is later scheduled to run on the processor, information about the process must be stored when it is withdrawn from the processor. This makes sure that when the process is performed again, it will start in the same place. Context describes this knowledge about the working situation. This is the best way to see what I mean when I say that a context is to a method what a bookmark is to a book. Because that is what the processor wants to do, the context ought to be useful when it returns to the process.

Following virtualization in the OS, we also talked about threading in operating systems on the client and server sides. A virtual machine is an emulated version of a computer system that can run on top of another system. Virtual machines can access a wide range of resources, such as computing power (via hardware-assisted but constrained access to the host machine's CPU and memory), one or more physical or virtual disk devices for storage, a virtual or actual network interface, as well as any devices like video cards, USB devices, or other shared hardware. When discussing a virtual computer that is stored on a virtual drive, people frequently refer to a disk image. The data required for a virtual computer to boot could be stored on a disk image, or it might store whatever else the virtual machine might need for storage. We covered process VM, hosted VM, and native VM as forms of virtual machine implementations.

09/22/2022:

The lecture started with the discussion on the modification of threads topic, and we then discussed threads in operating systems with the example of "The X Windows system." The X Window System, also referred to as X11, is a graphical user interface (GUI) that offers open source, cross-platform compatibility, client-server architecture, and distributed network capabilities.

Although it runs on a variety of Unix variants most frequently, X is also offered in versions for a number of other operating systems. The X window system includes a wide range of functions, some of which include customizable visual capabilities, network transparency, and network connectivity. As a part of project Athena, a collaboration between Stanford University and the Massachusetts Institute of Technology, the X window system was initially developed in 1984. The X.Org Foundation, an open organization that manages the system's development and standards, is in charge of the X window system. The X Window System is frequently abbreviated as "X," "X11," or "X Windows."

In contrast to a conventional client/server paradigm, the X system's client is the component that runs on the local computer and communicates service requests to the distant server. This is how the X system's client/server model functions. When using the X operating system, the local computer serves as the server and provides the client programs with their display and server-side functionality. Although the client programs might actually be running remotely over several networks or locally, they provide the impression that they are.

A distributed system, commonly referred to as distributed computing, is a set of interconnected components running on many computers that work together to communicate and coordinate tasks so that the end user sees just one cohesive system. The two main operating modes for distributed systems are described here.

1. Each machine makes a contribution toward the accomplishment of a single goal, and the end user perceives the outputs as a single entity.
2. Although sharing resources or communication services is made simpler by the distributed system, each computer still caters to its own end-user.

Finally, we discussed Planet Lab, a distributed system example, to wrap out the lesson.