Operating Systems Lecture Notes
Prof. Ghassan Shobaki

May 9, 2025

Contents

Operating Systems Lecture 5: OS Introduction (Part 5): The OS is Interrupt Driven

An example of exceptions

When you try to run a program that is compiled for an ARM machine on an intel processor. So you are trying to execute an invalid binary for a particular machine, so that will get caught as an invalid operation. Now when these exceptions occur, what happens? The OS gets control and it terminates the process that is trying to do something wrong and that provides protection. Also, in particular, if a process is trying to do an "out of bound memory access" then it may crash the whole system.

Killing a process

It is legal for an operating system to kill a process when an exception gets generated. Becuase killing the process will not allow it to crash the system.

Time quantum/quanta

That period of time that the OS gives to a process on the CPU is called a "time quantum". A process will get enough time quanta to complete but how long is this time quantum is determined by the OS. In timed interrupt, we switch between the processes.

Are timed interrupts common?

Yes, they happen very frequently because they will allow the OS to switch between processes. Otherwise one process may keep the CPU forever. In a typical system that we use everyday, there are multiple processes running at the same time. And that is time sharing and without time interrupt the OS will not be able to implement that.

What is the purpose of switching between the processes?

1. User responsivenss
2. Fairness to the users. Because these different processes may belong to the same user or to the different user. Think of a server. An operating system on a server and multiple users are using that server. Like if you are using an athena machine and 10 other students are using athena at the same time so in that case, if it is not heavily overloaded, you will not feel any lateness. You will feel like the machine is all yours. You will feel like you are the only user on the machine, but you are not the only user on the machine. There are other users and the OS is just switching between the users so frequently that each user gets the illusion that the machine is all theirs. So this is the point in doing this kind of switching.

Are timed Interrupt good for terminating infinite loops??

So if we have a program that does this:

```
while (1) {---}
```

It doesn't have a break in it and it will loop forever. Now what does a timed interrupt do in this case, how does a timed interrupt interact with this. Will the timed interrupt terminate the loop?

Suppose that a time quantum that is given to each process is 10 miliseconds. So the system will give this above process 10 miliseconds. And we can assume that during the first 10 miliseconds given to our above process the system is going to execute part of the loop. Then the OS is going to switch to another process. And it may switch to 10 or 12 other processes before it gives the CPU back to this process. And now this process is going to process the other portion of the above while loop. So if in this example, we assume that the body of the loop take 20 miliseconds, the process is going to take 2 time quantum. But then in the next time around, it will get another time quantum and it will execute the first portion of the loop again. SO the point is that the OS are not even aware of INFINITE LOOPS. OSs don't have a mechanism of detecting infinite loops and the operating system doesn't care if a program is in an infinite loop because as long as the program is behaving well, the OS doesn't care. THere are programs that run for days, or even weeks or months. YOu can start your web browser today and you leave it running for weeks. It is typical. For OS point of view, the program is still running and the operating system is not going to terminate it as long as it is behaving well. So the OS is not like a human being that may get sick or tired of a program and say that this program has been running for a long time and i am going to kill it. As long as the process is behaving like a well behaved citizen and it is not doing anything wrong, the OS will keep giving it time quantum forever. But now, the point here is that when the OS keeps switching between processes and our process will be kept stuck in an infinite loop, whether for a good reason or a bad reaosn, it doesn't matter. Other processes are making progress as well. And in fact whne we get to scheduling, we will see how an OS will try to give more CPU time to processes that are more active or interactive. A process that is doing more activity or interacting with the user is going to get more attention from the OS and it is going to get more CPU time and we will see that when we get to CPU scheduling.

Summary

The OS doesn't detect an infinite. It doesn't prevent an infinite loop. But what it does is prevent the slowing of whole system or killing the systme. Other processes are making progress as well.

System interrupt

It is a way for an appication program to ask the OS for a certain service that only the OS provides. All the services that only the OS are allowed to implement because they invoke access to certain device that only the OS is allowed to access or perform certain previlliged instructions. Anything of that sort, a user program is not allowed to do directly. It must go through the OS. And it does that via system call. System call is like making a function call. But it is a special kind of function call where the implementation of that function call is in the Kernel.So it is not implemented a user level library, it is implemented by the OS. And usually system calls are implemented by interrupts. Some machiens have a system called "instruction" but it does the same thing basically (whether it is an interrupt or it is a system call instruction that the user program executes, either way, control will

be given to the OS. It is basically the application program saying, "Now I need the OS to do this for me, so now the operating system should get control.