Team Apol

INTROOS S23 - Nieva, Perez, Pol

1. How do you execute a program in your kernel?

At first hand, the CPU is booting in kernel mode. In this mode, there is a complete unrestricted access to the system hardware. The CPU can access any memory addresses and executes any instructions. This mode is reserved for the most trusted functions and operations of the OS to prevent crashing the entire PC. After that, the boot loader loads and runs the kernel or the operating system available.

During program executions, the operating system prepares the memory for the program and loads the instruction from the disk. But before running the first instruction, the CPU will switch into a less privileged level by marking the mode flag from kernel mode to user mode. In this mode, the executing code cannot reference memory or access computer hardwares directly. Switching to user mode is necessary to run programs simultaneously. In this way, programs would not be able to access each other’s data since programs are have limited access to memory and other peripheral devices. This system promotes destruction isolation since programs are running independent to one another. Instead of crashing the entire computer system, only a particular program or application crashes.

For example, when a user-mode application needs to access computer hardware such as reading from a disk or getting an input from the keyboard, it needs to switch to kernel mode to be able to execute such operations. The mechanism on which program from user-mode can switch to kernel mode without having control over the instructions that will be executed in the kernel mode is called a ‘system call’. This system call is implemented in the CPU as the ‘trap instruction’. The processes involved in executing such instructions are as follows:

* To indicate the specific service a user-mode program requires from the OS, it needs to create a stack frame with arguments or place values in the registers.
* The program then executes the trap instruction.
* The CPU switches from user-mode to kernel mode and jumps to instructions at a fixed location in the memory. These instructions have memory protection so that they cannot be modified and read by user-mode programs.
* These instructions, also known as trap or system call handler, read the details of the requested service and arguments, and then perform it in kernel mode.
* After execution of the system call, the OS resets the mode to user-mode and returns from the system call.

2. Differentiate programs and processes

A program is a passive entity while processes are active. A program is a binary which can be executed while a process is a running instance of a program. Programs only became processes when executable files are loaded into memory. A program can be several processes when multiple users are executing the same program. There are different states of a process when it executes, it can be new, running, waiting, ready or in a terminated state.

3. What is the kernel’s responsibility to app developer’s?

Kernel is responsible for resource management that serves requests from applications. The kernel should be able to request resources such as memory whenever a program or process requires it. The kernel allocates the memory to the process itself. The kernel helps application developers to worry about the code in a way the operating system understands and the kernel makes it work on the hardware that is why the applications running in its own user-space is separated from the kernel.

References:

<https://www.blikbook.com/>

<http://blog.codinghorror.com/understanding-user-and-kernel-mode/>

http://minnie.tuhs.org/CompArch/Lectures/week05.html

<http://www.systhread.net/texts/200606kern_def.php>