System Calls

CSE 4001 Operating Systems Concepts

E. Ribeiro

January 24, 2022

Outline

What is a process?

- A process is an abstraction of a program in execution.

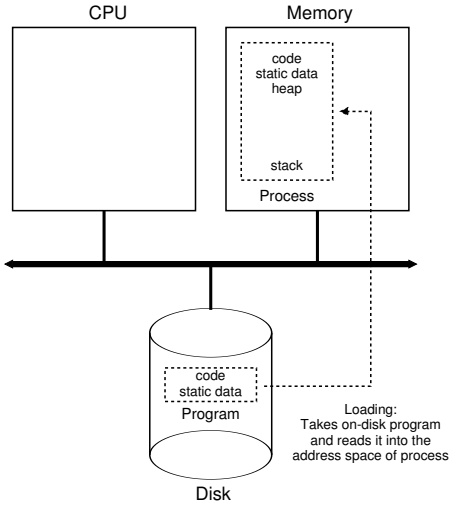

Figure from: OS in three easy pieces

Main question:

How can the OS **regain control** of the CPU from a process so that is can switch to another process?

Life cycle of a process

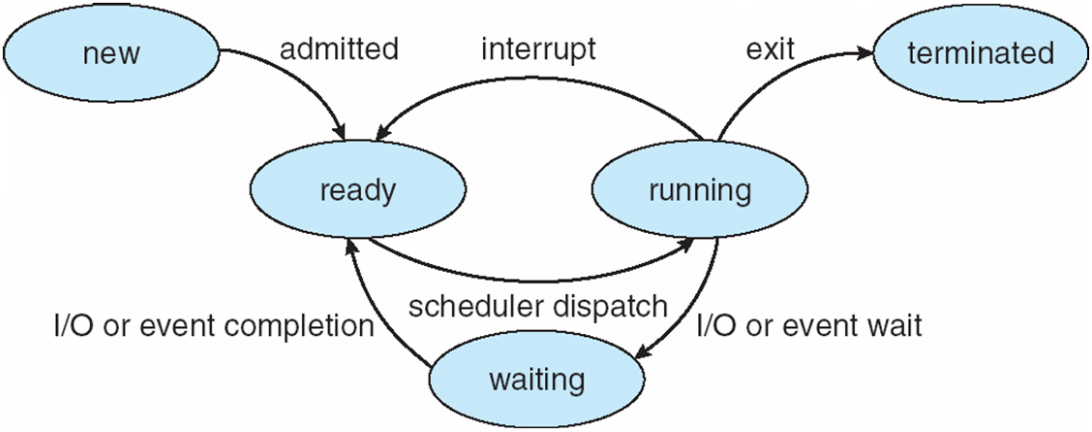

Figure adapted from Silberschatz, Galvin, and Gagne, 2009.

Limited Direct Execution

Main question:

How can the OS **regain control** of the CPU from a process so that is can switch to another process?

Two Aproaches:

- Cooperative processes
- Non-cooperative processes

Approach 1: Cooperative processes

- OS trusts processes will cooperate and give up control of CPU. For example, process can periodically calls system call `yield()`.
- Process gives up control when it causes a trap.

Slide text adapted from original slide by Larry Zhang.

Approach 1: Cooperative processes

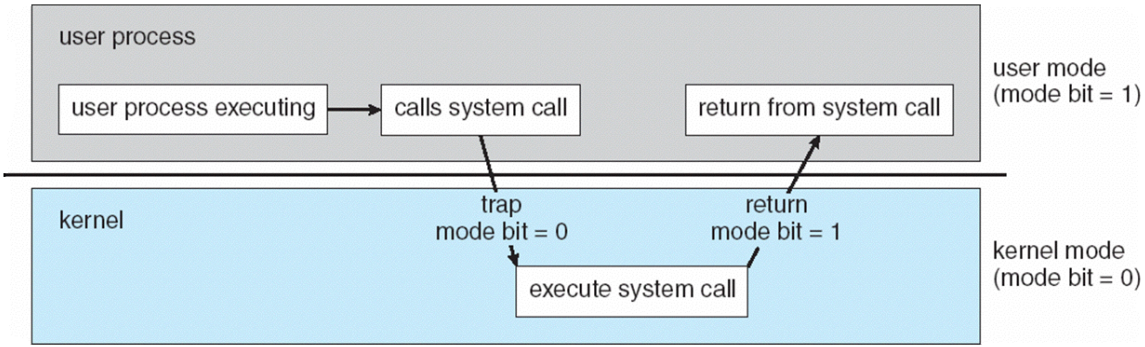

Figure adapted from Silberschatz, Galvin, and Gagne, 2009.

Approach 2: Non-cooperative processes

- OS takes control periodically (e.g., timer interrupt).
- Timer can be programmed to raise an interrupt periodically.
- When interrupt is raised, OS *Interrupt Handler* runs, and OS regains control.

Now, OS has control. How to switch to another process?

- OS decides the process to which to switch (i.e., scheduler decides).
- OS executes a piece of assembly code (i.e., context switch).

Context switch

Context-switch steps:

1. Save register values of current process to kernel stack.
2. Restore register values of the next process from its kernel stack.

In the next slides, let's see two examples of context switch in OS/161, one caused by the timer and the other caused by a trap (or exception).

OS/161 Examples: Context switch triggered by timer.

function `hardclock` in `/kern/thread/clock.c`

```
/*
 * This is called HZ times a second (on each processor) by the timer
 * code.
 */
void
hardclock(void)
{
        /*
         * Collect statistics here as desired.
         */

        curcpu->c_hardclocks++;
        if ((curcpu->c_hardclocks % MIGRATE_HARDCLOCKS) == 0) {
                thread_consider_migration();
        }
        if ((curcpu->c_hardclocks % SCHEDULE_HARDCLOCKS) == 0) {
                schedule();
        }
        thread_yield();
}
```

OS/161 Examples: Context switch triggered by timer.

function `thread_yield` in `/kern/thread/thread.c`

```
/*
 * Yield the cpu to another process, but stay runnable.
 */
void
thread_yield(void)
{
        thread_switch(S_READY, NULL, NULL);
}
```

OS/161 Examples: Context switch triggered by timer.

function `thread_switch` in `/kern/thread/thread.c` calls low-level context switcher in assembler in `/kern/arch/mips/thread/switch.S`

```
659             */
660            curcpu->c_curthread = next;
661            curthread = next;
662
663            /* do the switch (in assembler in switch.S) */
664            switchframe_switch(&cur->t_context, &next->t_context);
665
666            /*
```

OS/161 Examples: Context switch triggered by timer.

https://github.com/eribeiroClassroom/os161-Kernel-Src-Add-System-Call-Assignment/blob/master/kern/arch/mips/thread/switch.S

OS/161 Examples: Context switch triggered by an exception or trap.

General exception occurs which causes the hardware to call:

`https://github.com/eribeiroClassroom/os161-Kernel-Src-Add-System-Call-Assignment/blob/master/kern/arch/mips/locore/exception-mips1.S`

`exception-mips1.S` creates and fills in the `trapframe` and then calls `mips_trap()`. This C-language function is a general trap (exception) handling function.

OS/161 Examples: Context switch triggered by an exception or trap.