Chapter 3: Processes

Andrés Ponce

October 14, 2020

A **process** is a program that is currently running. One of the main functions of an operating system is to run user programs, so the way an OS manages process execution, scheduling and memory management can become quite important.

The status of the current executing program can be known with the value of the **program counter** [1] and the values currently in the registers.

[1] The PC stores the address of the currently executing instruction.

Usually, a program executable file has four distinct sections. The **text section** contains the executable code, while the **data section** contains the global variable declarations. the **heap** is where the dynammic memory is allocated, and the **stack** is where temporary variables are allocated. These variables are the ones used for function executions and are then deleted.

The data and text sections of a program are fixed, since they do not change during execution. However, the stack and heap shrink as needed, and they grow **toward** each other but never **overlap** each other. [2]

[2] Then how do we account for really big programs?

The program state can fall into certain categories:

- **New**: Process is being created.
- **Running**: There are instructions being executed.
- **Waiting**: Process is waiting for some event.
- **Ready**: Process is ready to be assigned to CPU.
- **Terminated**: Process finished execution.

Process Control Block

The proces is represented in teh OS by using a block of memory containing program info called the **Process Control Block**. It stores a couple things:

- **Process State**: One of previously mentioned process state.
- **Program Counter**: Address of the next instruction to be executed.
- **CPU registers**: There are different types of registers that store different information. This has to be saved on interrupt to be able to resume when the interrupt is resolved.
- **CPU scheduling**: Scheduling parameters.

- **Mem. Mgmt. Information**: The value of the base and limit registers, page tables, segment tables, depending on memory scheme in computer.
- **Accounting Information**: Amount of CPU and real time used, time limits, process/job numbers, etc...
- **I/O Status Information**: List of open files, I/O devices allowed to use, etc...

The set of PCB's is stored as a linked-list, with the info stored in `include/linux/sched.h`. When we want to schedule a process, we have to change the state. Then we execute until we are finished or waiting for a process, and move to the next program in the **wait queue**.

In terms of scheduling, each process gets a CPU core when it needs to and gets removed when not in need of execution, such as waiting for I/O. For CPU bound instructions though, they also don't get to execute constantly throughout its lifetime, the CPU scheduler might forcible remove it after a specified time limit.