Caderno de InfraSoft

Marconi Gomes

August 15, 2019

1 Introdução

→ **O que acontece quando um programa executa?**
Ele se transforma num processo e executa instruções (**uma por vez, compartilhando tempo**). Basicamente na seguinte sequência (Modelo de Von Neumann): FECTH → DECODE → EXECUTE → STORE.

2 Sistema Operacional

É o conjunto de softwares responsável por:
- Tornar fácil rodar programas (ao mesmo tempo)
- Fazer os programas compartilharem memória
- Fazer a interação com os dispositivos de I/O

O SO pode virtualizar recursos de hardware para diversos fins, além disso ele necessita de drivers para se comunicar com o hardware (específico para cada hardware).
Ainda mais, o SO oferece interfaces de comunicação com o sistema (APIs) para executar operações de forma rápida e fácil pelos programas.

Os SOs de **primeira geração** eram constituídos por **válvulas e painéis de programação.** Já os de **segunda geração** começaram a fazer o uso de **transistores** e eram capazes de fazer a leitura de sistemas em lote.
A **terceira geração** começou a usar os **circuitos integrados**, introduzindo o conceito também de multiprogramação, os CIs foram melhoras cada vez mais em desempenho e são usados até hoje.
A **quarta geração** até o presente, os computadores avançaram o suficiente para serem usados como computadores pessoais.

2.1 Gerenciador de Recursos

O gerenciador de recursos do SO, como o nome sugere, gerencia os recursos de hardware disponíveis para serem usados pelos processos na fila.
De forma básica, um processo pode esstar em 3 estados básicos, sendo eles: **Pronto** (quando o programa foi carregado na memória e está pronto para a execução), **Execução** (quando o processo está realmente usando da CPU) e **Bloqueado** (quando o programa está ou esperando uma instrução de outro dispositivo, ou seu tempo de uso da CPU se esgotou.)
Resumidamente, quando um programa é executado a sequência de estados que seu processo sofre é a seguinte: **Pronto** (pois o programa acabou de ser carregado na memória e está pronto

para execução) → **Execução** (quando o programa usa a CPU para executar suas instruções com um tempo predefinido pelo escalonador de processos) → **Bloqueado** (pois o programa excedeu seu tempo limite de uso da CPU). O processo então fica transitando nesse ciclo, até que eventualmente seja finalizado.

Pergunta: Se rodarmos dois processos ao mesmo tempo, quem será executado primeiro?
Resposta: Não sabemos! Isso vai depender do algoritmo de escalonamento do sistema operacional que os processos serão sendo executados.

2.2 Virtualizando a memória

Os processos fazem uso da memória, e cada processo pode necessitar de uma quantidade específica diferente de memória, então surge a necessidade de alocação de memória.
Para gerenciar a necessidade de uso dessa memória, o SO faz um mapeamento da memória física para uma forma virtua, no caso usando **endereços de memória**, criando assim uma noção abstrata de memória.

2.3 Memória virtual

Quando temos uma série de instruções (páginas) para serem executados e todos precisam estar na memória, precisamos usar o espaço disponível na memória para carregar as instruções nela. Entretanto, pode existir um contratempo onde não há memória física (RAM) suficiente para carregar todas as instruções. Se isso acontecer, o SO mapeia todas as instruções necessárias com o que couber na memória RAM e o que não couber, será armazenado em outra memória um pouco mais lenta (no Disco Rígido).
Então, o conceito de executar a operação de todas as instruções necessárias (independente do tamanho da memória) porém sua localização (ou seja, onde esse conjunto de instruções esteja em armazenado em memórias diferentes) ser descentralizada em mais de uma localização se chama **Memória virtual**.

References