MAC0422 - Sistemas Operacionais

Daniel Macêdo Batista

IME - USP, 3 de Setembro de 2020

Roteiro

Introdução

Introdução

Computador moderno

- 1+ processadores
- memória
- discos
- Diversos dispositivos de E/S (mouse, monitor, impressora, placas de rede, etc...)
  - Aplicações farão uso desses recursos
  - Aplicações precisam "saber" como usar esses recursos
  - Imagine ter que reescrever a rotina para ler algo da placa de rede toda hora!

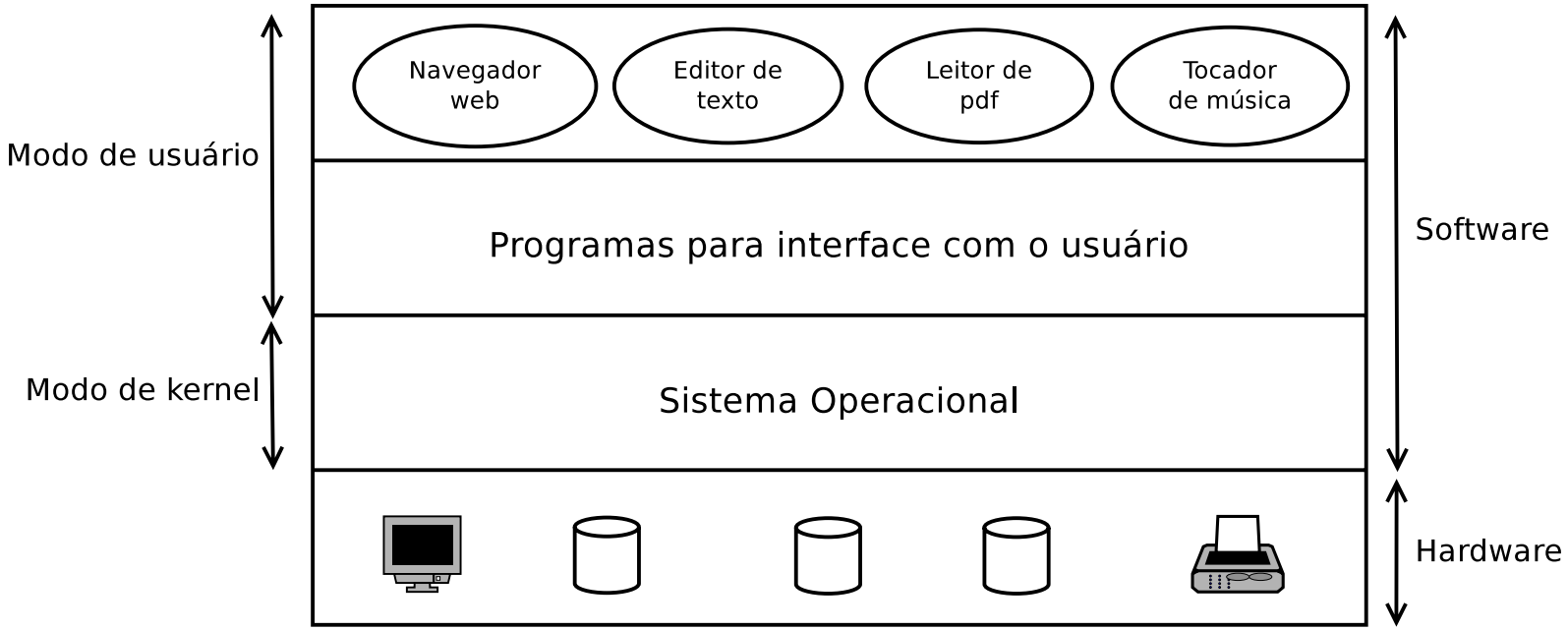
Por que ter um sistema operacional (SO)?

- Para agir como uma camada de software que facilita a "vida" das aplicações (e dos programadores também)
- O SO fornece um modelo (melhor) de computador que facilita o gerenciamento dos recursos:
  - simples
  - abstrato

Contato com SO

- Todos temos, mas como usuário na maioria das vezes
- A principal interação com o SO é pedir para ele executar outras aplicações (não "pegamos" no SO de fato)
  - Shell
  - GUI

Como tudo isso se conecta?



- O SO roda em modo kernel ou modo supervisor
- Instruções "perigosas" não costumam estar presentes no modo usuário (ex de exceção: `passwd`). Difícil separar o que é do SO e o que não é...
- As aplicações podem ser modificadas, o SO não com tanta facilidade

Como tudo isso se conecta?

- SOs em relação a aplicações de usuário
  - grandes
  - complexos
  - tempo de vida longo
  - evolui sem ter muitas partes "jogadas fora" (se o hardware não muda, para que mudar o SO?)
  - grande parte do código relacionada com drivers (parte do SO que cuida de gerenciar e se comunicar com hardwares específicos)

O que é então um SO?

- Um software que roda em modo kernel (mas, segundo alguns autores, com algumas partes que rodam em modo usuário)
- SOs possuem 2 tarefas principais:
  - abstrair os recursos de baixo nível para as aplicações e para os programadores
  - gerenciar os recursos de baixo nível

SO como uma ferramenta para abstração

- A arquitetura no nível de linguagem de máquina é primitiva e confusa para programar (principalmente com E/S – Entrada e Saída)
- Exemplo (obsoleto) de um controlador de drive de disco flexível
  - 16 comandos: ler e escrever dados, mover o braço magnético, formatar trilhas, etc...
  - ler e escrever exige 13 parâmetros (!): endereço, número de setores, modo de gravação, espaçamento entre setores, etc...
  - rodar os comandos em ordem incorreta, pode destruir o disco!
  - Em resumo: o programador teria que fazer um curso de cada tipo de hardware antes de utilizar aquele hardware!

SO como uma ferramenta para abstração

- O usuário e o programador querem só ler e escrever dados
- O programador quer uma abstração simples e de alto nível
  - No caso de discos: nomes de arquivos, formatar o disco, abrir, fechar, ler e escrever arquivos.
- O SO faz isso pelo programador. Quem programa o SO precisará entender o hardware mas como as tarefas serão feitas repetidamente por outras aplicações, não tem pra que reescrever isso (mesma ideia de ter uma biblioteca que faça o trabalho secundário permitindo que o programador foque no principal problema a ser resolvido)
- Dominar a interface de usuário é importante mas não é o foco da disciplina. Vamos estudar mais a interação com o SO do ponto de vista de programador, não de usuário

SO como um gerenciador de recursos

- Mas e quando há várias aplicações rodando em cima do SO?
- O programador da aplicação quer usar o recurso (está abstraído) mas e se mais de uma aplicação quiser usar?
- Exemplo de uso da impressora:
  - Duas aplicações enviam trabalhos para a impressora
  - Não pode simplesmente enviar os dois arquivos a qualquer momento para imprimir. Pode sair tudo corrompido
  - Copiar os trabalhos para algum buffer, controlar o acesso a esse buffer, enviar os trabalhos em ordem, esperar a impressora aquecer e esperar a confirmação de que um trabalho foi impresso corretamente antes de enviar outro
- Em resumo: o programador teria que escrever rotinas complexas para controlar a concorrência pelo hardware

SO como um gerenciador de recursos

- O usuário e o programador querem só usar o recurso. Não quer arbitrar a utilização
- O SO faz isso. Ele permite alocação controlada e organizada de processadores, memória, dispositivos de E/S e demais recursos de hardware
- De um modo geral o SO precisa gerenciar e proteger os dispositivos e os dados para que aplicações não façam acessos indevidos (tanto sem querer quanto de forma maliciosa)

SO como um gerenciador de recursos

- O compartilhamento dos recursos pode ser feito de duas formas:
  - Tempo: os usuários e programas tem rodadas para usar aquele recursos. Primeiro um usa, depois outro usa, e assim por diante. O processador é um exemplo. Cada programa usa ele por um tempo (supondo que exista apenas um processador na máquina). Como definir quanto tempo usar e quem vai ser o próximo é assunto da disciplina
  - Espaço: os usuários e programas ficam com parte do recurso. Um usa 90%, outro usa 5% e outro usa os outros 5%. A memória é um exemplo. Se há espaço na memória para todas as aplicações, cada uma ocupa um espaço. Como proteger essas áreas de memória e como reutilizá-las caso não seja suficiente para todas as aplicações é assunto da disciplina