Um pouco da evolução dos sistemas operacionais

Aleardo Manacero

1 Introdução

Este texto é apenas um breve resumo sobre a evolução dos sistemas operacionais. Infelizmente não apresenta as devidas referências, fazendo apenas histórico sobre como os SO surgiram e sobre como eles são projetados hoje a partir de textos lidos pelo professor. Grande parte do que está aqui aparece, sob diversas formas, nos bons livros de SO. Assim, se desejarem mais detalhes sugiro que recorram aos mesmos.

2 Antes dos SO

Os computadores (com arquitetura semelhante a que conhecemos hoje) surgiram no final da década de 1930, inicialmente a partir de trabalhos de Konrad Zuse na Alemanha e depois com trabalhos separados na Inglaterra (para criptografia) e EUA (para cálculos balísticos) durante a segunda guerra mundial. A característica comum a todos eles era a ausência de um software de controle da máquina, ou seja, todo o controle era exercido pelo próprio programa em execução. Isso, obviamente, tornava a tarefa de programação extremamente complexa, pois cabia ao programador não apenas resolver o seu problema, como também gerenciar todos os componentes do hardware.

Nesse período se percebeu que boa parte das rotinas de controle eram, na prática, repetidas e quase idênticas em qualquer programa. Assim, pouco antes de 1950 apareceram os primeiros programas monitores, que nada mais eram do que coleções de rotinas comuns de controle e entrada/saída. Com isso os programadores podiam se concentrar na resolução de problemas específicos, sem perder tempo com rotinas de controle. A desvantagem nesse processo é que os monitores, por serem de caráter geral, não garantiam o máximo desempenho possível, que poderia ser atingido por um programador experiente que fizesse todo o controle do sistema.

Para concluir, os monitores, vistos como um ambiente, podem ser considerados como o embrião dos sistemas operacionais.

3 E então surgiram os primeiros SO

A evolução do hardware disponível, associado com o aumento nas tarefas controladas pelos monitores levou ao surgimento dos primeiros sistemas operacionais. Eles vieram inicialmente na forma de "pooling", em que se tornou possível realizar as operações de leitura do programa, execução do mesmo e impressão de seus resultados por máquinas distintas. Assim, enquanto uma máquina lia um programa (tipicamente em cartões perfurados), a cpu executava um outro programa, previamente lido, e a impressora fazia a saída dos resultados de um terceiro programa.

Com o tempo esses ambientes foram integrados de forma a evitar que dados fossem transferidos entre eles através de dispositivos externos (fitas magnéticas). Isso permitia agilizar o processamento, levando ao conceito de processamento em "batch", em que a execução de programas semelhantes era agrupada de modo a evitar a troca constante de compiladores e outros aplicativos. Deve ser lembrado que máquinas daquela época não eram capazes de executar mais do que um programa por vez, levando à necessidade de se trocar o compilador cada vez que um programa demandasse por uma linguagem diferente da anterior.

Essa estrutura beneficiava aos proprietários dos computadores da época, uma vez que maximizava o tempo útil de processamento. Para os desenvolvedores, entretanto, era bastante complicado obter resultados de forma rápida. Eles tinham que esperar por sua vez dentro dos batches de processamento.

4 Os primeiros sistemas compartilhados

Com novos avanços do hardware se tornou possível, por volta de 1960, executar mais programas de uma vez. Isso permitiu o aparecimento de sistemas operacionais de tempo-compartilhado ("time-sharing"), em que vários programas são executados simultaneamente, um pedacinho por vez. O conceito de compartilhamento no tempo foi introduzido pela IBM através do IBM Stretch (primeiro "supercomputador", lançado comercialmente em 1961).

É nessa época que surgiu um dos mais importantes projetos na área de SO até hoje. Trata-se do **Multics**, que pretendia ser um SO de ampla cobertura, capaz de executar desde sistemas em batch até sistemas paralelos, cobrindo todos os tipos de ambientes entre eles. Apesar de nunca ter atingido a funcionalidade completa, tivemos sistemas rodando Multics até cerca de 20 anos atrás. Mas sua importância vai além do pouco sucesso comercial. É que dele surgiram muitos dos conceitos usados atualmente, inclusive é dele que surgiu o UNIX, por volta de 1970.

Com a introdução do time-sharing se tornou possível aos desenvolvedores ter acesso direto aos sistemas computacionais. Isso era tipicamente feito através dos chamados terminais remotos (ou terminais burros, como eram também conhecidos). O tempo de cpu era então compartilhado entre os vários terminais que o computador dispusesse, dando alguns milissegundos de execução para cada um deles, que se revezavam no seu uso.

Nos últimos 50 anos foi o modelo de time-sharing que dominou a área dos sistemas operacionais. A única exceção veio com o surgimento dos computadores pessoais, em que se adotou sistemas monoprocessados (logo sem necessidade de compartilhar tempo), como CP/M e DOS. Mesmo essas máquinas passaram a executar em time-sharing com a evolução do hardware utilizado. Esse domínio pode ser melhor entendido percebendo-se como o mesmo evoluiu para acompanhar a evolução do hardware dos computadores. As variações do modelo de tempo compartilhado são vistos na próxima seção.

5 Modelos atuais de SO

O modelo de compartilhamento no tempo foi sendo modificado ao longo dos anos. IBM, Digital e Bell Labs, entre outros, foram os grandes responsáveis por alguns dos avanços mais relevantes. Eles são apresentados a seguir.

5.1 A máquina virtual

No final dos anos 60 a IBM introduziu o conceito de máquina virtual (não confundir com as máquinas virtuais produzidas através de virtualização), que consistia em separar as diversas funcionalidades do SO em camadas, que são sucessivamente acessadas pelas camadas superiores, como visto na figura 1. O chamado núcleo do sistema operacional é a primeira camada acima da máquina física.

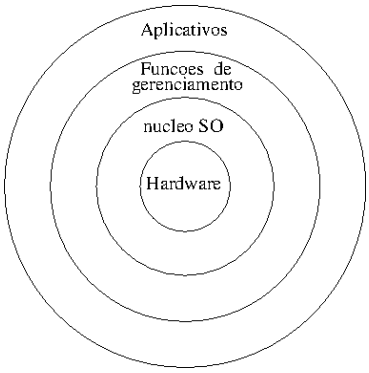

Figura 1: Modelo de camadas em um SO de máquina virtual

Funções do núcleo são as únicas capazes de executar instruções de hardware que impliquem em operações protegidas. Os programas de usuários e de outras camadas do SO apenas conseguem o acesso

a essas instruções através do núcleo. Isso permite um certo tratamento de segurança dentro do sistema operacional, embora bastante precário.

O uso de máquinas virtuais como modelo de implementação de sistemas operacionais permanece ainda bastante atual. Na prática as máquinas virtuais criadas para o processo de virtualização são uma extensão daquele modelo. A diferença é que agora é permitido executar um sistema operacional sobre um sistema operacional nativo, que executa de fato sobre o hardware.

5.2 O microkernel

O conceito de camadas das máquinas virtuais simplifica bastante o projeto do SO, permitindo separar componentes próximos do hardware de componentes mais gerenciais. O custo disso é que as aplicações apenas conseguem acesso ao núcleo do sistema através de chamadas para primitivas de nível mais alto, acarretando em um *overhead* de processamento. Uma abordagem para diminuir essa sobrecarga veio com o conceito de microkernel. Nele se busca eliminar parte dos acessos intermediários criando mecanismos para minimizar as funcionalidades do núcleo, habilitando o acesso das mesmas também para os aplicativos, como visto na figura 2.

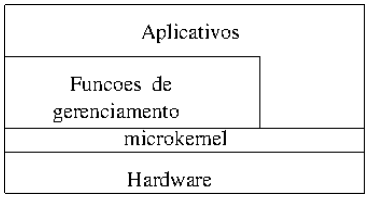

Figura 2: Modelo de um SO baseado em microkernel

Com essa estrutura as aplicações podem executar algumas chamadas de sistema mais rapidamente, sem a interferência de mecanismos de gerenciamento. Claro que tais chamadas não podem incluir ações que devem obrigatoriamente ser controladas pelo SO.

5.3 Ambientes *multithreaded*

Um dos problemas com sistemas operacionais é que o seu funcionamento deve influenciar o mínimo possível o tempo de execução dos demais programas executando. Isso ocorre pois cada vez que o SO atua é preciso que ele ocupe a CPU, retirando o programa que estava executando dela. Isso, é claro, implica em atrasos na execução dos programas de cada usuário, o que deve ser minimizado. O uso de em threads procura fazer isso ao reduzir o tempo consumido na troca de contexto entre um programa e outro, incluindo o SO.

Isso é possível pois *threads* podem ser gerenciados a partir de um volume menor de informações, pois cada um deles compartilha informações do processo pai. Essa abordagem se tornou ainda mais atraente com a introdução de processadores de vários núcleos. Neles o ganho deixa de ser apenas na troca de contexto, passando também para a paralelização efetiva da execução de cada *thread*.

6 Tipos de SO

Ao longo da evolução dos SO definiu-se alguns esquemas para a classificação dos mesmos. Um desses esquemas parte do campo de aplicação do SO. Outro parte do número de programas e usuários que podem fazer uso do SO. Primeiramente, segundo a quantidade de programas/usuários temos os seguintes tipos:

1. Monousuário/Monotarefa, com apenas um usuário por vez, capaz de executar apenas um programa por vez;
2. Multiusuário/Monotarefa, com vários usuários em tempo compartilhado, mas capazes de executar apenas um programa cada um por vez;
3. Monousuário/Multitarefa, com apenas um usuário por vez, capaz de executar vários programas por vez;

4. Multiusuário/Multitarefa, com vários usuários em tempo compartilhado, capazes de executar vários programas cada um por vez (nosso foco será nesse tipo de sistema);

Já quanto ao campo de aplicação temos:

1. Sistemas dedicados, em que os programas (poucos ou até mesmo apenas um) tem objetivo comum bastante específico;
2. Sistemas de uso geral, em que os programas tem objetivos variados (nosso foco será nesse tipo de sistema);
3. Sistemas de tempo-real, em que os programas necessitam produzir resultados dentro de intervalos de tempo pré-definidos;
4. Sistemas distribuídos, em que se faz uso de computadores distribuídos em rede para a produção de resultados;
5. Sistemas paralelos, em que se faz uso de computadores ou processadores paralelos para a produção de resultados.