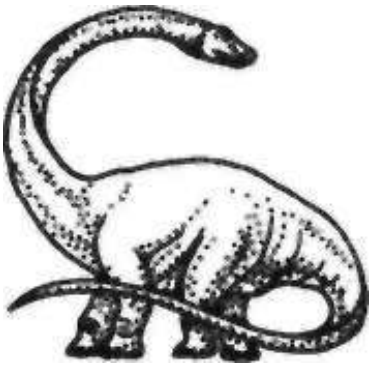
Capítulo 1



INTRODUÇÃO

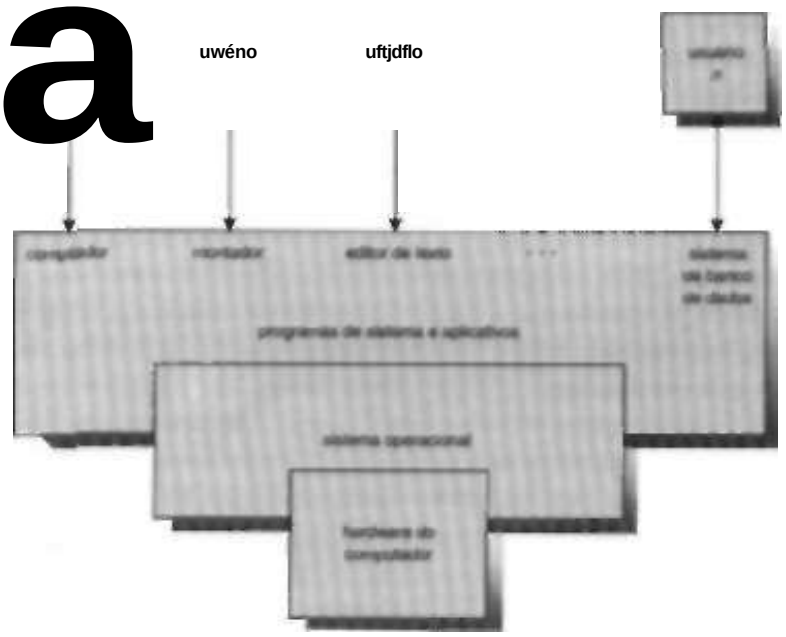
Um sistema operacional é um programa que atua como intermediário entre o usuário e o hardware de um computador. O propósito de um sistema operacional c fornecer um ambiente no qual o usuário possa executar programas. O principal objetivo de um sistema operacional é portanto tornar o uso do sistema de computação conveniente. Uma meta secundária é usar o hardware do computador de forma eficiente.

Para entender o que são sistemas operacionais, primeiro precisamos compreender como eles se desenvolveram. Neste capítulo, fazemos um apanhado do desenvolvimento dos sistemas operacionais desde os primeiros sistemas aos atuais sistemas de muItiprogramação e de tempo compartilhado. Ao avançarmos pelos diferentes estágios, vemos como os componentes dos sistemas operacionais evoluíram como soluções naturais para os problemas dos primeiros sistemas de computação. Compreender as razões por trás do desenvolvimento dos sistemas operacionais permite observar as tarefas que eles executam e como o fazem.

1.1 • O que é um sistema operacional?

Um sistema operacional é um componente importante de praticamente todo sistema de computação. Um sistema de computação pode ser dividido em basicamente quatro componentes: o hardware, o sistema operacional, os programas aplicativos e os usuários (Figura 1.1).

Figura 1.1 Visão abstrata dos componentes de um sistema de computação.



O hardware - a unidade central de processamento (CPU, *central processing Uttit),* a memória e os dispositivos de entrada/saída (I/O, Input/Output) - fornece os recursos básicos de computação. Os programas aplicativos - processadores de texto, planilhas eletrõnicas, compiladores c navegadores Web - definem as maneiras em que esses recursos são usados para resolver os problemas de computação dos usuários. Pode haver muitos usuários diferentes (pessoas, máquinas, outros computadores) tentando resolver problemas diferentes. Da mesma forma, pode haver muitos programas aplicativos diferentes. O sistema operacional controla e coordena o uso do hardware entre os vários programas aplicativos para os vários usuários.

Um sistema operacional í semelhante a *um governo.* Os componentes de um sistema de computação são seu hardware, software e dados. O sistema operacional fornece o meio para o uso adequado desses recursos na operação do sistema de computador. Como um governo, o sistema operacional não executa nenhuma função útil por si mesma. Simplesmente fornece um *ambiente* no qual outros programas podem realizar tarefas úteis.

Podemos considerar um sistema operacional como um alocador de recursos. Um sistema de computação possuí muitos recursos (hardware e software) que podem ser necessários para resolver um problema: tempo de CPU, espaço na memória, espaço de armazenamento de arquivos, dispositivos de entrada/saída (l/O), entre outros. O sistema operacional atua como gerente desses recursos e os aloca a programas e usuários específicos, conforme necessário, para a execução das tarefas. Como pode haver muitos pedidos de recursos, possivelmente conflitantes entre si, o sistema operacional deve decidir em que pedidos serão alceados recursos para que ele possa operar o sistema de computação de forma eficiente c justa.

Uma visão ligeiramente diferente de um sistema operacional enfatiza a necessidade de controlar os vários dispositivos de 1/0 e programas de usuário. Um sistema operacional é um programa de controle. Um programa de controle controla a execução dos programas de usuário para evitar erros e o uso indevido do computador. Preocupa-se especialmente com a operação e o controle de dispositivos de l/O.

Em geral, no entanto, não existe uma definição completamente adequada de um sistema operacional. Os sistemas operacionais existem porque são uma forma razoável de resolver o problema de criar um sistema de computação que possa ser usado. O objetivo primordial dos sistemas de computação é executar programas de usuário e tornar fácil a resolução dos problemas de usuário. Para atingir essa meta, 0 hardware é construído. Como o hardware por si só não *é* particularmente fácil de usar, programas aplicativos são desenvolvidos. Esses vários programas exigem certas operações comuns, como aquelas que controlam os dispositivos de I/O. As funções comuns de controle e alocação de recursos são então reunidas em um único software: o sistema operacional.

Não existe uma definição universalmente aceita do que faz e do que não faz parte do sistema operacional. Um ponto de vista simples *é* que tudo o que o fornecedor entrega quando você solicita"o sistema operacional" deve ser considerado. Os requisitos de memória c os recursos incluídos, no entanto, variam muito de sistema para sistema. Alguns usam menos de 1 megabyte de espaço e não têm nem um editor de tela inteira, enquanto outros exigem centenas de megabytes de espaço e baseiam-se inteiramente em sistemas gráficos de janelas. Uma definição mais comum *é* que o sistema operacional é um programa que está sempre executando no computador (geralmente chamado núcleo ou *kernel),* todo o resto consistindo em programas aplicativos. Normalmente, adotamos essa última definição. A questão em torno do que constitui um sistema operacional está se tornando importante. Em 1998, o Departamento de justiça norte-americano entrou com um processo contra a Microsoft basicamente alegando que a empresa incluía um número excessivo de funcionalidades em seus sistemas operacionais, impedindo, assim, qualquer concorrência por parte de outros fabricantes.

E mais fácil definir um sistema operacional pelo que ele *faz* do que pelo q u e d e i . O principal objetivo de um sistema operacional *è* a *conveniência do usuário.* Os sistemas operacionais existem porque têm como missão tornar a tarefa computacional mais fácil. Essa visão fica particularmente nítida quando analisamos sistemas operacionais para computadores pessoais de pequeno porte (PCs).

Uma meta secundária é a operação *eficiente* do sistema de computação. Essa meta é particularmente importante para sistemas multiusuário compartilhados e de grande porte. Esses sistemas são geralmente caros, por isso devem ser o mais eficientes possível. Essas duas metas - conveniência e eficiência - às vezes são contraditórias. No passado, a eficiência era frequentemente mais importante que a conveniência. Assim, boa parte da teoria dos sistemas operacionais concentra-se no uso otimizado dos recursos computacionais.

Para entender o que são sistemas operacionais e o que eles fazem, vamos considerar o seu desenvolvimento nos últimos 35 anos. Ao acompanhar essa evolução, poderemos identificar quais são os elementos comuns dos sistemas operacionais e verificar como e por que esses sistemas se desenvolveram dessa maneira.

Os sistemas operacionais e a arquiterura dos computadores tiveram grande influência mútua. Para facilitar o uso do hardware, os pesquisadores desenvolveram sistemas operacionais. A medida que os sistemas operacionais foram criados e utilizados, ficou claro que mudanças no projeto do hardware poderiam simplificá-los. Nessa breve revisão histórica, observe como a identificação dos problemas de sistemas operacionais levou ã introdução de novos recursos de hardware.

1.2 • Sistemas em lote (batch)

Os primeiros computadores eram máquinas exageradamente grandes (em termos físicos), operadas a partir de um console. Os dispositivos de entrada comuns eram leitoras de cartões e unidades de fita. Os dispositivos de saída comuns eram impressoras de linhas, unidades de fita e perfuradoras de cartões. O usuário não interagia dirctamente com os sistemas de computação. Km vez disso, ele preparava um job (tarefa), que consistia no programa, dados e algumas informações de controle sobre a natureza da tarefa (cartões de controle), c o submetia ao operador do computador. A tarefa geralmente tinha a forma de cartões perfurados. Algum tempo depois (minutos, horas ou dias), a saída aparecia. A saída consistia no resultado do programa, um dump de memória e registradores no caso de erro de programa.

O sistema operacional nesses primeiros computadores era bem simples. Sua principal tarefa era transferir controle automaticamente de um job para o próximo. O sistema operacional estava sempre residente na memória (Figura 1.2).

Para acelerar o processamento, os operadores reuniam os jobs em lotes com necessidades semelhantes e os executavam no computador como um grupo. Assim, os programadores deixavam seus programas com o operador. O operador classificava os programas em lotes com requisitos semelhantes e, à medida que o computador ficava disponível, executava cada lote ou batch. A saída de cada job seria enviada de volta ao programador apropriado.

Neste ambiente de execução, a CPU muitas vezes fica ociosa, porque as velocidades dos dispositivos mecânicos de I/O são intrinsecamente mais lentas do que as dos dispositivos eletrônicos. Mesmo uma CPU lenta funciona na faixa de microssegundos, executando milhares de instruções por segundo. Uma leitora de cartões rápida, por outro lado, pode ler 1.200 cartões por minuto (20 cartões por segundo). Assim, a diferença em velocidade entre a CPU e seus dispositivos de I/O pode ser de três ordens de grandeza ou mais. Com o tempo, é claro, melhorias na tecnologia e a introdução de discos resultaram em dispositivos de I/O mais rápidos. No entanto, as velocidades de CPU aumentaram ainda mais, por isso o problema não só não foi resolvido mas também exacerbado.

Figura 1.2. Layout da memória para um sistema em batch simples.

A introdução da tecnologia de disco permitiu que o sistema operacional mantivesse todos os jobs cm um disco, em vez de cm uma leitora de cartões serial. Com acesso direto a vários jobs, o escalonamento de jobs poderia ser executado para usar recursos e realizar tarefas de forma eficiente. O Capítulo 6 aborda em detalhes jobs e escalonamento de CPU; alguns aspectos importantes são discutidos aqui.

O aspecto mais importante do escalonamento de jobs é a capacidade de multiprogramação. Um único usuário não pode, em geral, manter a CPU ou os dispositivos de *VQ* ocupados em todos os momentos. A multiprogramação aumenta a utilização de CPU organizando jobs de forma que a CPU sempre tenha um job para executar.

A ideia é explicada a seguir. O sistema operacional mantém vários jobs na memória ao mesmo tempo (Figura 1,3). Esse conjunto de jobs é um subconjunto dos jobs mantidos no pool de jobs (já que o número de jobs que pode ser mantido simultaneamente na memória geralmente é muito menor do que o número de jobs que pode estar no pool de jobs). 0 sistema operacional escolhe e começa a executar um dos jobs na memória. Em alguns momentos, o job terá de esperar a conclusão de alguma tarefa, como uma operação de I/O. Em um sistema não-multiprogramado, a CPU ficaria ociosa. Em um sistema de multiprogramação, o sistema operacional simplesmente passa para outro job e o executa. Quando *esse job* precisa esperar, a CPU passa para outro job e assim por diante. Por fim, o primeiro job termina a espera e tem a CPU de volta. Desde que haja pelo menos um job para executar, a CPU nunca fica ociosa.

Essa ideia é comum cm outras situações da vida. Um advogado não trabalha apenas para um cliente de cada vez. Em vez disso, vários clientes podem estar sendo atendidos ao mesmo tempo. Enquanto um caso está aguardando julgamento ou preparação de documentos, o advogado pode trabalhar em outros casos. Se o advogado tiver um número suficiente de clientes, ele nunca ficará ocioso por falta de trabalho. (Advogados ociosos tendem a se tornar políticos, por isso existe um certo valor social cm manter os advogados ocupados.)

A multiprogramação é a primeira instância em que o sistema operacional precisa tomar decisões pelos usuários. Os sistemas operacionais multiprogramados são, portanto, bastante sofisticados. Todos os jobs que entram no sistema são mantidos no pool de jobs. Esse pool consiste cm todos os processos residentes no disco aguardando alocação da memória principal. Se vários jobs estiverem prontos para serem carregados na memória e, se não houver espaço suficiente para todos, o sistema deverá fazer a escolha. Essa tomada de decisão é chamada *escalonamento de jobs* c será discutida no Capítulo 6. Quando o sistema operacional seleciona um job do pool de jobs, ele o carrega na memória para execução. Ter vários programas na memória ao mesmo tempo requer alguma forma de gerência de memória, tema que será abordado nos Capítulos *9* e 10. Além disso, se vários jobs estiverem prontos para executar ao mesmo tempo, o sistema deverá escolher um deles. Essa tomada de decisão *é* chamada escalonamento de CPU e será discutida no Capítulo 6. Finalmente, múltiplos jobs sendo executados ao mesmo tempo vão exigir que haja pouca interferência mútua em todas as fases do sistema operacional, incluindo escalonamento de processos, armazenamento em disco e gerência de memória. Essas considerações são discutidas em todo o texto.

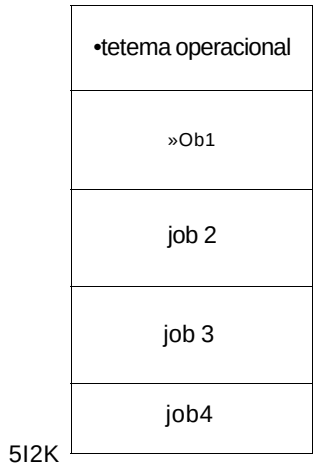


Figura 1.3 Layout de memória para um sistema de multiprogramação.

1.3 • Sistemas de tempo compartilhado

Os sistemas em batch multiprogramados forneceram um ambiente no qual os vários recursos do sistema (por exemplo, CPU, memória, dispositivos periféricos) eram utilizados de forma eficaz, mas não permitiam a interação do usuário com o sistema de computação. Tempo compartilhado, ou multitarefa, *c* uma extensão lógica da muItiprogramação. A CPU executa vários jobs alternando entre eles, mas as trocas ocorrem com tanta frequência que os usuários podem interagir com cada programa durante sua execução.

Um sistema de computação interativo permite a comunicação direta entre o usuário e o sistema. O usuário passa instruções ao sistema operacional ou a um programa diretamente, usando um teclado ou um mouse, e espera por resultados imediatos. Da mesma forma, o tempo de resposta deve ser curto - geralmente em torno de 1 segundo.

Um sistema operacional de tempo compartilhado permite aos muitos usuários compartilharem o computador ao mesmo tempo. Como cada ação ou comando em um sistema de tempo compartilhado tende a ser curto, apenas um pequeno tempo de CPU é necessário para cada usuário. Como o sistema alterna rapidamente de um usuário para outro, cada usuário tem a impressão de que todo o sistema de computação está dedicado ao seu uso, enquanto, na verdade, um computador está sendo compartilhado por muitos usuários.

Um sistema operacional de tempo compartilhado utiliza o escalonamento de CPU e a mu Iti programação para fornecer a cada usuário uma pequena parte de um computador de tempo compartilhado. Cada usuário tem pelo menos um programa separado na memória. Um programa carregado na memória e em execução é normalmente chamado de processo. Quando um processo executa, geralmente executa durante um curto espaço de tempo antes de terminar ou de precisar realizar uma operação de I/O- A operação de entrada/saída pode ser interativa, ou seja, a saída para o usuário é feita em um monitor c a entrada é a partir de um teclado, mouse ou outro dispositivo. Como a I/O interativa geralmente tem velocidade humana, pode levar muito tempo para terminar. A entrada, por exemplo, pode ser limitada pela velocidade de digitação do usuário; sete caracteres por segundo é rápido para pessoas, mas é incrivelmente lento para computadores. Hm vez de deixar a CPU inativa durante essa operação de entrada interativa, o sistema operacional rapidamente passará a CPU para o programa de algum outro usuário.

Os sistemas operacionais de tempo compartilhado são ainda mais complexos do que os sistemas operacionais multiprogramados. Em ambos, vários jobs devem ser mantidos na memória ao mesmo tempo, por isso o sistema deve contar com recursos de gerência de memória e proteção (Capítulo 9). Para alcançar um tempo de resposta razoável, os jobs talvez precisem ser passados rapidamente da memória principal para o disco que agora serve como extensão da memória principal. Um método comum de alcançar essa meta é a *memória virtual,* uma técnica que permite a execução de um job que talvez não esteja completamente na memória (Capítulo 10). A principal vantagem óbvia desse esquema é que os programas podem ser maiores do que a memória física. Além disso, ele abstrai a memória principal em um vetor grande e uniforme de armazenamento, separando a memória lógica conforme vista pelo usuário da memória física. Esse arranjo libera os programadores da preocupação relativa aos limites da memória. Os sistemas de tempo compartilhado também fornecem um sistema de arquivos (Capítulo 11). O sistema de arquivos reside cm uma coleção de discos; portanto, o gerenciamnto de disco deve ser realizado (Capítulo 13). Além disso, os sistemas de tempo compartilhado fornecem um mecanismo para execução concorrente, que requer esquemas sofisticados de escalonamento de CPU (Capítulo 6). Para garantir a execução correra, o sistema deve fornecer mecanismos para a comunicação e sincronização de jobs, e pode garantir que os jobs não fiquem presos em deadlocks, eternamente esperando uns pelos outros (Capítulo 7).

A ideia do tempo compartilhado foi demonstrada já em 1960, mas como os sistemas de tempo compartilhado são difíceis e caros de construir, só se tornaram comuns no início dos anos 70. Embora algum processamento em batch ainda ocorra, a maioria dos sistemas hoje é de tempo compartilhado. Por essa razão, a multiprogramação e o tempo compartilhado são os temas centrais dos sistemas operacionais modernos e também deste livro.