Faculdade Pitagoras – Unidade Divinopolis
Prof. Eng. Alexandre Dezem Bertozzi, Esp.
Disciplina: Sistemas Operacionais
Proposta de Temas de trabalhos.

TIPOS DE SISTEMAS OPERACIONAIS

Sistemas operacionais para computadores pequenos e grandes dividem-se em quatro categorias que se diferenciam pelo tempo de resposta e pela forma de entrada de dados no sistema: em lotes (*batch*), interativa, em tempo real e sistemas híbridos.

Os **sistemas em lotes (*batch*)** existem desde o tempo dos primeiros computadores, os quais trabalhavam com cartões perfurados ou fita magnética para a entrada de dados. Para a execução de uma tarefa, os cartões eram agrupados em um lote e processados através de um leitor de cartões. Os sistemas em lotes de hoje não trabalham mais com cartões perfurados ou com fitas magnéticas, mas as tarefas ainda são processadas em série, sem interação com o usuário.

A eficiência do sistema era medida em termos de throughput — o número de tarefas completadas em determinado período de tempo (por exemplo, 30 tarefas por hora) — e o tempo de resposta era medido em horas, ou até mesmo dias. Hoje, é muito difícil encontrar um sistema que se limite a programas em lotes.

Os **sistemas interativos** (também conhecidos como sistemas de tempo compartilhado) produzem um tempo de resposta mais rápido do que os sistemas em lotes, mas são mais lentos do que os sistemas em tempo real, dos quais falaremos a seguir. Foram criados para atender às necessidades dos usuários que necessitavam de um tempo de resposta mais rápido na depuração de seus programas. O sistema operacional exigiu a criação de programas de tempo compartilhado que permitiam ao usuário interagir diretamente com o sistema de computação através de comandos digitados em um terminal semelhante a uma máquina de datilografar.

O sistema operacional proporciona um retorno imediato para o usuário e o tempo de resposta pode ser medido em minutos ou segundos, dependendo do número de usuários ativos.

Os **sistemas em tempo real**, os mais rápidos de todos, são utilizados em ambientes onde o tempo é um fator crítico, ou seja, onde os dados devem ser processados muito rapidamente, pois influenciam as decisões mais imediatas: vôos espaciais, controle de tráfego aéreo, aeronaves de alta velocidade, processos industriais, equipamentos médicos mais sofisticados, distribuição de eletricidade e comutação telefônica. Um verdadeiro sistema em tempo real deve produzir respostas 100% imediatas em 100% das vezes. O tempo de resposta é medido em frações de segundo, mas esse ideal muitas vezes não é alcançado.

Apesar de, teoricamente, ser possível converter um sistema operacional de múltiplos fins em um sistema em tempo real, na prática, a sobrecarga dos sistemas gerais é tão grande que o desempenho em tempo real não pode ser alcançado. Assim sendo, grande parte das tarefas em tempo real requer sistemas operacionais especialmente concebidos para esse fim.

Os **sistemas híbridos** são uma combinação entre os sistemas em lotes e os sistemas interativos. Eles parecem interativos porque os usuários individuais podem acessar o sistema através de terminais e obter respostas rápidas. Na verdade, um sistema híbrido aceita programas em lotes e executa-os em segundo plano, quando a carga interativa não é muito grande. Um sistema híbrido beneficia-se do tempo livre entre demandas de processamento para executar programas que não precisam da interferência de um operador. Muitos dos grandes sistemas de computação são híbridos.

UMA BREVE HISTÓRIA DO DESENVOLVIMENTO DOS SISTEMAS OPERACIONAIS

A evolução dos sistemas operacionais é paralela à evolução dos computadores para os quais eles foram criados. Eis aqui uma rápida visão desse processo.

Anos 40

A **primeira geração** de computadores (1940–1955) surgiu na época da tecnologia dos tubos a vácuo e dos computadores do tamanho de uma sala de aula. Cada computador era único em termos de estrutura e função. Não havia muita necessidade para um software operacional padronizado, já que o uso de cada computador restringia-se a poucos profissionais trabalhando em aplicações matemáticas, científicas ou militares, e todos eles acabavam por se acostumar com as idiossincrasias de seu equipamento.

Um programa típico incluiria todas as instruções necessárias para que o computador executasse as tarefas desejadas. Daria instruções explícitas para o leitor de cartões (quando começar, como interpretar os dados nos cartões, quando finalizar), para a CPU (como e onde armazenar as instruções na memória, o que calcular, onde encontrar os dados, para onde enviar os resultados) e para o dispositivo de saída (quando começar, como imprimir o produto final, como formatar a página e quando finalizar).

As máquinas eram operadas por programadores a partir do console principal — o processo era manual. Com efeito, para depurar um programa, o programador tinha de parar o processador, ler o conteúdo de cada registrador e fazer as correções nas posições de memória para, em seguida, retomar a operação.

Os programadores tinham de reservar a máquina durante todo o período de tempo estimado para a execução do programa. Como resultado, o aproveitamento das máquinas era muito baixo. A CPU processava dados e fazia cálculos em apenas uma fração do tempo disponível e, na verdade, o sistema inteiro permanecia ocioso entre os períodos reservados.

Com o tempo, padronizaram-se os equipamentos e os programas de computação e a execução de um programa passou a requerer menos passos e menos conhecimento sobre o funcionamento interno do computador. Compiladores e montadores foram criados para transformar em código binário os comandos em palavras das linguagens de alto nível da época.

Sistemas operacionais rudimentares começaram a tomar forma com a criação de macros, programas de biblioteca, sub-rotinas padronizadas e programas utilitários. Esses sistemas também incluíam sub-rotinas para *drivers* de dispositivo — programas pré-escritos que padronizavam a forma pela qual os dispositivos de entrada e saída eram utilizados.

Os programas mais antigos possuíam uma desvantagem significativa. Concebidos para utilizar seus recursos de forma conservadora, em prejuízo da legibilidade e da clareza, muitos programas utilizavam uma lógica confusa, que apenas o programador original era capaz de compreender. Assim, era quase impossível para qualquer outra pessoa depurar ou alterar o programa posteriormente.

Anos 50

Os computadores de **segunda geração** (1955–1965) foram desenvolvidos para atender às necessidades de um novo mercado: as empresas. O ambiente empresarial valorizava muito mais o equilíbrio entre os custos e os benefícios de um sistema. Os computadores ainda eram muito caros, especialmente quando comparados a outros equipamentos de escritório. (O IBM 7094 custava US$ 200.000.) Portanto, a velocidade de processamento tinha de ser maximizada para justificar o investimento em termos empresariais. Isso significava que o grau de aproveitamento do sistema deveria ser dramaticamente melhorado.

Duas melhorias foram amplamente adotadas: operadores de computador foram contratados para facilitar a operação das máquinas e a programação de execução de tarefas (*scheduling*) foi implementada. A programação de execução de tarefas é um esquema de aumento de produtividade que agrupa programas com requerimentos semelhantes. Por exemplo, vários programas em FORTRAN eram executados juntos enquanto o compilador FORTRAN estivesse em memória; ou todas as tarefas que precisavam do leitor de cartões para a entrada de dados eram executadas juntas, e aquelas que precisavam de fitas magnéticas eram executadas mais tarde. Alguns operadores perceberam que a combinação entre os requerimentos de entrada e saída seria mais eficiente. Isto é, com a combinação entre programas de entrada via fita magnética e programas de entrada via cartão perfurado, as fitas poderiam ser montadas ou rebobinadas enquanto o leitor de cartões estivesse ocupado.

A programação de execução de tarefas resultou na necessidade de cartões de controle, os quais definiam a natureza exata de cada programa e de seus requerimentos. Essa foi uma das primeiras aplicações da linguagem de controle de tarefas — **job control language** (JCL) —, e sua função era auxiliar o sistema operacional na coordenação e no gerenciamento dos recursos de sistema através da identificação dos usuários e de suas tarefas e da especificação dos recursos necessários para a execução de cada tarefa.

A seguir, exemplo de uma simples configuração de programa para o DEC-10:

```
$JOB  (insere número do usuário)          < identifica tarefa e usuário
$PASSWORD (insere senha do usuário)       < valida a identificação do usuário
$LANGUAGE (indica o compilador necessário < especifica o recurso
 [programa fonte]
$DATA                                     < identifica o recurso
[dados]
$EOJ                                      < identifica fim da tarefa e libera
                                            recursos
```

Mesmo com as técnicas de formação de lotes, os computadores de segunda geração, mais rápidos, ainda geravam lapsos de tempo muito longos e caros entre a CPU e os dispositivos de entrada e saída (E/S). Por exemplo, uma tarefa com 1.600 cartões podia levar 79 segundos para ser lida pelo leitor de cartões e apenas 5 segundos de tempo de CPU para ser montada (ou compilada). Isso significa que a CPU estava ociosa durante 94% do tempo, processando de fato durante apenas 6% do tempo dedicado àquela tarefa.

Com o passar do tempo, vários fatores contribuíram para a melhoria do desempenho da CPU. Em primeiro lugar, os dispositivos de E/S — unidades de fita, discos e tambores — tornaram-se gradualmente mais rápidos. Em segundo lugar, visando otimizar a área de armazenagem disponível nesses dispositivos, os registros eram divididos em blocos antes de ser recuperados ou armazenados. (**Divisão em blocos** significa que vários registros lógicos são agrupados em um único registro físico.) Naturalmente, quando os registros eram recuperados, eles deviam ser reagregados para que o programa pudesse utilizá-los. Para auxiliar os programadores nas funções de divisão em blocos e reagregação, métodos de acesso foram criados e acrescentados ao código-objeto através do editor de ligações.

Em terceiro lugar, para reduzir a discrepância em termos de velocidade entre os dispositivos de entrada/saída e a CPU, uma interface denominada unidade de controle foi colocada entre eles para desempenhar a função de armazenamento em buffer. Um buffer é uma área de armazenagem temporária que funciona da seguinte forma: enquanto o lento dispositivo de entrada lê um registro, a unidade de controle coloca cada caractere do registro no buffer. Quando o buffer está cheio, o registro inteiro é rapidamente enviado para a CPU. O processo é exatamente o inverso para os dispositivos de saída: o registro inteiro é armazenado no buffer pela CPU e em seguida é enviado para a unidade de controle na velocidade menor requerida pelo dispositivo de saída.

Se uma unidade de controle possui mais de um buffer, o processo de entrada e saída pode ser ainda mais rápido. Por exemplo, se a unidade de controle possui dois buffers, o segundo pode ser carregado enquanto o primeiro está transmitindo seu conteúdo para a CPU. Em termos ideais, depois de terminada a transmissão do primeiro buffer, o segundo já está pronto para transmitir, e assim por diante. Dessa forma, o tempo de entrada é reduzido pela metade.

Em quarto lugar, além do mecanismo de armazenamento em buffer, uma forma primitiva de *spooling*[1] foi desenvolvida, permitindo que as operações de perfuração e leitura de cartões e impressão fossem executadas off-line. Apesar de ser usada, em inglês, como substantivo ou como verbo, a palavra SPOOL é o acrônimo de Simultaneous Peripheral Operation On-Line (Operação Periférica Simultânea On-Line). Por exemplo, as tarefas recebidas eram transferidas, off-line, de lotes de cartões para fita magnética. Em seguida, eram enviadas à CPU através da fita magnética a uma velocidade muito maior do que a velocidade do leitor de cartões. Um *spooler* trabalha da mesma forma que um buffer, mas, nesse exemplo, é um dispositivo off-line separado, ao passo que um buffer é parte integrante do hardware de um sistema de computação.

Ainda durante a segunda geração, técnicas foram desenvolvidas para gerenciar programas de biblioteca, criar e manter arquivos de dados e índices, criar endereços de acesso direto não-seqüencial e criar e verificar identificadores de arquivo. Várias técnicas padronizadas de organização de arquivos foram desenvolvidas: seqüencial, seqüencial indexada e de acesso direto. Com o aparecimento dessas técnicas, miniprogramas padronizados, chamados macros, dispensaram os programadores da necessidade de escrever rotinas específicas de abertura e fechamento para cada programa.

Comandos de interrupção por lapso de tempo foram criados para proteger a CPU de laços infinitos em programas que erroneamente foram instruídos para executar uma única série de comandos infinitamente e para permitir o compartilhamento de tarefas. Uma quantidade fixa de tempo de execução era alocada a cada programa assim que ele entrava no sistema e monitorada pelo sistema operacional. Se, ao final desse tempo, um programa qualquer ainda estivesse em execução, ele era finalizado e uma mensagem de erro era enviada ao usuário.

Durante a segunda geração, os programas ainda eram executados no modo serial em lotes, ou seja, um de cada vez. O passo seguinte para a otimização dos recursos de sistema foi a introdução do processamento compartilhado.

---

1. N.T.: *Spool*: processo de armazenamento dos dados que compõem um documento em uma fila, até que a impressora esteja disponível para imprimi-los.

Anos 60

A **terceira geração** de computadores tem início em meados da década de 60. Seus computadores tinham CPUs mais rápidas, mas o aumento de velocidade causava problemas quando as CPUs interagiam com os dispositivos de entrada e saída. A solução encontrada foi a multiprogramação, ou seja, a ativação simultânea de vários programas que compartilhavam a capacidade de processamento de uma única CPU.

Os primeiros sistemas de multiprogramação permitiam que os programas fossem executados seqüencialmente, um depois do outro. O mecanismo mais comum para a implementação da multiprogramação foi a introdução do conceito de interrupção, que acontece quando a CPU é avisada sobre a existência de eventos que precisam dos serviços do sistema operacional. Por exemplo, quando um programa emite um comando de E/S, ele gera uma mensagem de interrupção requisitando os serviços do processador de E/S e a CPU é liberada para iniciar a execução da tarefa seguinte. Isso era chamado de **multiprogramação passiva**, uma vez que o sistema operacional não controlava os comandos de interrupção e esperava até que cada tarefa terminasse uma seqüência de execução. Isso não era o ideal porque, no caso de uma tarefa limitada pela CPU (isto é, uma tarefa que executa grande volume de processamento contínuo antes de emitir um comando de interrupção), esta ficava ocupada por longo período, enquanto outras tarefas tinham de esperar.

Para contrabalançar esse efeito, a **multiprogramação ativa** foi desenvolvida, conferindo ao sistema operacional um papel mais dinâmico. No novo esquema, cada programa podia utilizar apenas uma fatia preestabelecida de tempo de CPU. Ao expirar o tempo, a tarefa era interrompida e outra podia entrar em execução. A tarefa interrompida tinha de esperar a vez para recomeçar mais tarde. O conceito de divisão do tempo em fatias logo se tornou comum em muitos sistemas de tempo compartilhado.

A programação de execução, introduzida nos sistemas de segunda geração, foi mantida nesse período, mas teve de lidar com uma nova complicação: o fato de que a memória principal passou a ocupar-se com muitas tarefas. A solução encontrada foi agrupar as tarefas de acordo com algum princípio e em seguida executar os programas em uma seqüência rotativa preestabelecida. Os grupos normalmente eram definidos por prioridade ou por requerimentos de memória — o que fosse considerado a forma mais eficiente de utilização de recursos.

Além da programação de execução de tarefas, da emissão de comandos de interrupção e da alocação de memória, os sistemas operacionais tiveram de resolver conflitos sempre que duas tarefas requisitavam simultaneamente o mesmo dispositivo.

Nesse período, poucos avanços importantes foram introduzidos no gerenciamento de dados. As funções de biblioteca e os métodos de acesso permaneceram os mesmos desde os últimos anos da segunda geração. O sistema operacional para máquinas de terceira geração consistia em vários módulos entre os quais o usuário podia escolher; assim, o sistema operacional total era personalizado para atender às necessidades do usuário. Os módulos empregados com maior freqüência permaneciam na memória principal e os menos utilizados eram alojados em memória secundária, sendo ativados apenas quando necessário.

Anos 70

Depois da terceira geração, no final da década de 70, os computadores receberam CPUs mais rápidas, criando assim uma disparidade ainda maior entre a rapidez de processamento e a lentidão de acesso dos dispositivos de entrada e saída. Os esquemas de multiprogramação criados para otimizar a utilização da CPU eram limitados pela capacidade física da memória principal, à época, um recurso restrito e muito caro.

A solução para essa limitação física foi o desenvolvimento da **memória virtual**, que se beneficiou do fato de que a CPU era capaz de processar apenas uma instrução por vez. Com a memória virtual, o programa como um todo não precisava mais permanecer em memória para que a execução pudesse iniciar-se. Um sistema com memória virtual dividia o programa em segmentos e os mantinha em armazenagem secundária, colocando-os em memória apenas quando necessário. (Os programadores dos computadores de segunda geração valeram-se do conceito para a criação do método de programação *roll in/roll out*, hoje conhecido como sobreposição de camadas — *overlay*, possibilitando a execução de programas que excediam a memória física desses computadores.)

No período, houve crescente preocupação com a necessidade de conservação de dados. Os softwares de gerenciamento de bancos de dados tornaram-se uma ferramenta popular, pois organizavam dados de maneira integrada, minimizavam redundâncias e simplificavam os processos de acesso e atualização. Vários sistemas de consulta foram introduzidos, permitindo que até mesmo usuários menos experientes acessassem áreas específicas de um banco de dados. As consultas normalmente eram feitas através de um terminal, o que, por sua vez, estimulou o avanço dos softwares de suporte a terminais e comunicação de dados.

Gradativamente, os programadores se distanciaram das complexidades internas do computador e começaram a surgir programas aplicativos que utilizavam palavras da linguagem humana, estruturas modulares e operações padronizadas. A tendência à padronização resultou em avanço no gerenciamento de programas, uma vez que a manutenção dos mesmos tornou-se mais rápida e mais simples.

Anos 80

Os desenvolvimentos da década de 80 produziram grande melhoria na relação custo/desempenho dos componentes de computação. O hardware tornou-se mais flexível, com funções lógicas inseridas em placas facilmente substituíveis. Tornou-se também mais barato, fazendo com que mais funções de sistema operacional fossem incorporadas ao hardware propriamente dito, trazendo à tona um novo conceito — ***firmware*** (o termo indica que um programa é colocado permanentemente em ROM — memória de apenas leitura, do inglês *read-only memory* — e não em armazenagem secundária). O trabalho do programador mudou radicalmente em relação à prática dos anos anteriores, já que muitas das funções de programação passaram a ser executadas pelo software do sistema, tornando a tarefa do programador mais simples e menos dependente do hardware.

Finalmente, a indústria passou a trabalhar com o **multiprocessamento** (com mais de um processador), e linguagens mais complexas foram concebidas para coordenar as atividades de múltiplos processadores executando uma mesma tarefa. O resultado foi a possibilidade de executar programas em paralelo; em pouco tempo, tornou-se comum esperar que os sistemas operacionais de computadores de qualquer porte fossem capazes de operar com o multiprocessamento.

A evolução dos microcomputadores e das comunicações de alta velocidade promoveu uma mudança em direção ao processamento distribuído e aos sistemas em rede, permitindo o compartilhamento remoto de recursos de hardware e software. Esses sistemas requeriam um novo tipo de sistema operacional, que fosse capaz de lidar com múltiplos gerenciadores de subsistema e com máquinas localizadas no outro lado do mundo.

Com os sistemas operacionais em rede, os usuários tomaram conhecimento da existência de vários recursos de rede, passaram a acessar localidades remotas e a manipular arquivos em computadores conectados em redes muito amplas. Os sistemas operacionais em rede assemelhavam-se aos sistemas operacionais de processador único, à medida que cada máquina operava o

próprio sistema operacional local e tinha usuários próprios. A diferença estava no acréscimo de um controlador de interface de rede com software de baixo nível para comandar o sistema operacional local, bem como em programas que permitem conexão remota e acesso remoto a arquivos. Mesmo com esses acréscimos, todavia, a estrutura básica do sistema operacional em rede ainda era muito semelhante à estrutura de um sistema autônomo.

Por outro lado, com os sistemas operacionais distribuídos, os usuários podiam ter a impressão de estar trabalhando com um sistema comum de processador único, quando na verdade estavam conectados a um agrupamento de vários processadores trabalhando harmoniosamente. Com esses sistemas, os usuários não precisavam saber qual processador executava suas aplicações ou quais dispositivos armazenavam seus arquivos. Esses detalhes eram todos administrados de maneira invisível ao usuário pelo sistema operacional — algo que requeria mais do que acrescentar algumas linhas de código a um sistema operacional de processador único. A desvantagem de um sistema operacional tão complexo assim era a necessidade de algoritmos de programação de execução mais complexos também. Além disso, os atrasos de comunicação em rede às vezes faziam com que os algoritmos de programação tivessem de operar com informações incompletas ou obsoletas.

Anos 90

No decorrer da década de 90, a demanda avassaladora por recursos de internet resultou na proliferação das configurações em rede. Hoje, acesso a web e correio eletrônico são recursos comuns a quase todos os sistemas operacionais. Entretanto, a proliferação das redes implicou demanda maior por esquemas mais rígidos de segurança para a proteção de hardware e software.

A década assistiu também a uma proliferação das aplicações multimídia, e os sistemas operacionais tiveram de incorporar recursos, tornar-se mais flexíveis e proporcionar uma compatibilidade maior entre os dispositivos. Um computador multimídia típico está habilitado a criar e editar arquivos gráficos, de áudio e de vídeo. Isso pode requerer uma série de dispositivos especializados: microfone, piano digital, interface MIDI (*Musical Instrument Digital Interface*, ou Interface Digital para Instrumentos Musicais), câmera digital, unidade de DVD, unidades de CD, alto-falantes, monitores adicionais, dispositivos de projeção, impressoras em cores, conexões de Internet de alta velocidade, etc. Software e hardware (controladores, placas, barramentos, etc.) especializados também são necessários para que esses dispositivos possam trabalhar adequadamente juntos.

As aplicações multimídia demandam grandes volumes de memória, que devem ser manipulados sem problemas pelo sistema operacional. Cada segundo de um vídeo de tela cheia de 30 quadros por segundo, por exemplo, precisa de 27 megabytes de memória. Isso significa que aproximadamente 24 segundos de vídeo podem preencher um compact disc comum (Stair, 1999). Portanto, um programa de televisão de 20 minutos precisaria de 50 discos para ser armazenado, a não ser que os dados fossem compactados de alguma forma. Para atender à demanda pela compactação de vídeo, chips específicos e placas de vídeo foram criados pelos fabricantes de hardware. Um padrão, denominado DVI (*Digital Video Interactive*, ou Vídeo Interativo Digital), compacta arquivos de vídeo a uma taxa de 150:1, permitindo que uma hora de vídeo seja armazenada em menos de um gigabyte.

Qual é o efeito desses avanços tecnológicos sobre o sistema operacional? Cada avanço implica a melhoria paralela nas capacidades de gerenciamento do software. Os sistemas operacionais que não conseguem manter-se em dia com as novas demandas rapidamente se tornam obsoletos.