Intro

O Sistema Operacional é uma camada de software colocada entre o hardware e os programas que executam tarefas para os usuários. O sistema operacional é o responsável pelo acesso aos periféricos, sempre que um programa necessita de algum tipo de operação de entrada e saída, ele solicita ao Sistema Operacional. Desta forma, o programador não precisa conhecer os detalhes do hardware.

Ao mesmo tempo, como todos os acessos aos periféricos são feitos através do sistema operacional, ele pode controlar qual programa está acessando qual recurso. É possível, então, obter uma distribuição justa e eficiente dos recursos. (*SO controla os softwares e o hardware*).

Objetivos do SO

A utilização do SO eficientemente busca um maior retorno no investimento feito em hardware pelas pessoas e empresas. Maior eficiência significa mais trabalho obtido pelo mesmo hardware. Uma utilização mais conveniente vai diminuir o tempo necessário para a construção dos programas, o que também implica uma redução no custo de software e também de mão-de-obra, uma vez que menos tempo é gasto em cada tarefa. (*ou seja, a otimização do tempo*).

Uma utilização mais conveniente do computador é obtida quando se esconde do programador detalhes do hardware, em especial dos periféricos. Por exemplo, para colocar um caractere na tela do terminal, em geral é necessário toda uma sequência de acessos à interface do terminal. Diversos registradores de controle e de status devem ser lidos ou escritos. Além disso, pode haver mais de um tipo de interface, com diferentes sequências de acesso. Ao usar o Sistema Operacional, o programador apenas informa qual caractere deve ser colocado na tela. Todo o trabalho de acesso ao periférico é feito pelo Sistema Operacional.

Arquivos não existem no hardware. Eles formam um recurso criado a partir do que o hardware oferece. Para o programador, é muito mais confortável trabalhar com arquivos do que receber uma área de espaço em disco que ele próprio teria que organizar. (*É o SO quem cria a idéia de arquivos e pastas*).

Tipos de serviço

Todo SO oferece meios para que um programa seja carregado na memória principal e executado. O SO recebe o nome do arquivo, aloca memória para o programa, copia o conteúdo do arquivo para a memória principal e inicia sua execução. Também é possível abortar a execução de um programa.

Talvez o mais importante serviço oferecido pelo SO seja o que permite a utilização de arquivos, serviço este implementado através do sistema de arquivos. Através dele, é possível criar, escrever, ler e destruir arquivos. Através da leitura e escrita, é possível copiar, imprimir, consultar e atualizar arquivos. Em geral, também existem operações do tipo renomear, obter tamanho, data e outras informações do arquivo. (Serviços relacionados aos arquivos).

Todo acesso aos periféricos é feito através do SO, porém é necessário um serviço para alocação, leitura, escrita e liberação. Quando diversos usuários compartilham o cmputador, passa a ser interessante também saber quanto de quais recursos cada usuário necessita.

Diversas informações sobre o estado do sistema são mantidas pelo SO. Em geral, essas informações são necessárias para o próprio funcionamento do sistema. Entretanto elas também podem ser fornecidas aos programas e usuários.

Na busca de um melhor aproveitamento do hardware, diversos usuários podem compartilhar um computador. Entretanto, isso somente é viável se houver algum tipo de proteção entre os usuários. Não é aceitável, por exemplo, que um usuário envie dados para a impressora no meio da listagem de outro usuário. Um exemplo ainda pior seria a destruição de arquivos ou o cancelamento da execução do programa de outra pessoa. Uma coisa importante é que se não houver

segurança com respeito à execução dos programas e à manutenção dos dados, os usuários simplesmente não utilizarão o computador. O SO tem que garantir que cada usuário possa trabalhar sem sofrer influência danosa dos demais (*SO cria mecanismos de proteção para o usuário*).

Sistema Operacional na visão do usuário – Chamadas de Sistema

Os programas solicitam serviços ao SO através das chamadas de sistema. O retorno da chamada de sistema, assim como o retorno de uma sub-rotina, faz com que a execução do programa seja retomada a partir da instrução que segue a chamada.

A parte do SO responsável por implementar as chamadas de sistema é normalmente chamada **núcleo**, ou ***kernel***. Os principais componentes do kernel de qualquer SO são a *gerência de processador, gerência de memória, sistema de arquivos e gerência de entrada e saída*.

Como o kernel é muito complexo, muitos SOs são implementados em camadas. Primeiro, um pequeno componente de software chamado **micronúcleo** ou ***microkernel*** implementa os serviços mais básicos associados com Sistemas Operacionais. Em cima do microkernel, usando seus serviços, o kernel propriamente dito implementa os demais serviços.

O microkernel oferece serviços básicos tais como gerência do processador, alocação e liberação de memória física e instalação de novos tratadores de dispositivos. O kernel do sistema oferece serviços tais como sistema de arquivos, memória virtual e protocolos de comunicação. Qualquer programa pode acessar as chamadas de sistema do kernel, porém apenas o kernel possui direitos de acessar o microkernel.

Programas de Sistema

Os programas de sistema (também chamados utilitários) são programas normais executados fora do kernel. Eles utilizam as mesmas chamadas de sistema disponíveis aos demais programas, e muitas vezes são confundidos com o próprio SO. (*exemplos: Windows Explorer, Systrey, data e hora...*).

O mais importante programa do sistema é o interpretador de comandos. Esse programa é ativado pelo SO sempre que um usuário inicia sua sessão de trabalho. Sua tarefa é receber comandos do usuário e executá-los.

Alguns comandos (como a listagem de um arquivo) podem ser feitos através de um interpretador de comandos (*tipo DOS*) ou de um utilitário. Em qualquer situação, as mesmas chamadas de sistema estarão envolvidas.

Tudo o que foi dito sobre interpretador de comandos é também válido para a situação em que o SO oferece uma **interface gráfica de usuário (GUI – graphical user interface)**. A única diferença está na comodidade para o usuário, que passa a usar ícones, menus e mouse no lugar de digitar comandos textuais. Na maioria das vezes, a GUI é o "Sistema Operacional" para o usuário, e será "julgada" pela sua facilidade de uso em um primeiro momento, para depois ser julgada pela sua flexibilidade e robustez. Em todo caso, na maioria das vezes, o usuário padrão utiliza apenas softwares, e o SO propriamente dito fica escondido do usuário comum.

Sistema Operacional na visão de Projeto

O SO não resolve os problemas do usuário final, porém é ele quem determina a eficiência da máquina, e essa eficiência é obtida através de compartilhamento de recursos e de uma interface mais agradável e confortável.

Dois tipos de eventos ativam o SO: Uma chamada de sistema ou uma interrupção de periférico.

Uma *chamada de sistema* corresponde a uma *solicitação de serviço* por parte de um programa em execução. O SO deve, então, verificar a legalidade da solicitação. Por exemplo, um pedido para apagar dados de outro usuário deve ser

negado. Já para o caso de uma solicitação legal, ele deve ser realizada e a resposta devolvida ao programa.

Em função das chamadas de sistema, o SO envia comandos para os controladores de periféricos. O controlador deve informar ao SO quando a operação estiver concluída. Isso é feito através de uma **interrupção**. Quando uma interrupção acontece, o processador pára o que tá fazendo e passa a executar uma rotina específica do SO.

O *conceito de interrupção* é possivelmente o mecanismo de hardware mais importante para a construção de um Sistema Operacional Moderno, juntamente com o conceito de *Unidade de Gerência de Memória (MMU – Memory Management Unit)*.

Histórico de Sistemas Operacionais

Na década de 40 não existia SO. Nesse ambiente, o programador é também o operador do computador. Existe uma planilha para alocação de horário na máquina. Durante o seu horário, o programador controla todo o equipamento. Um programa, quando executado, tem controle total da máquina. O programa acessa diretamente aos periféricos. No máximo, existe uma biblioteca com rotinas de entrada e saída já programadas.

A primeira modificação introduzida nesse esquema foi a utilização de operadores profissionais. O programador não mais opera o computador durante a execução de seu programa. Ele entrega ao operador o seu **job**. O job é formado pelo programa a ser compilado e executado, acompanhado dos dados para a execução. Geralmente, os programas e dados são preparados na forma de cartões perfurados. Após a execução, o programador recebe uma listagem com a saída gerada pelo programa.

O emprego de operadores profissionais diminui o tempo que o computador fica ocioso. Não existem as perdas inerentes a uma planilha de alocação de horário. Enquanto o programador pensa o que saiu errado, outro programa está sendo executado. Ainda assim, o tempo de preparação de um job continua sendo grande. É necessário primeiro retirar as fitas magnéticas, cartões e listagens do job que terminou, preparar as fitas e cartões pro próximo job. Para diminuir o tempo entre os jobs, eles passaram a ser agrupados em lotes. Em um mesmo **lote**, ou **batch**, são colocados jobs com necessidades semelhantes. Essa é a origem do termo sistema em batch.

Ainda assim, a passagem de jobs continuava sendo feita de forma manual. O operador precisa ficar atento ao console do computador. Quando o programa em execução terminar, ele deve manualmente comandar a carga e execução do próximo programa.

Na década de 50, surgem os primeiros monitores residentes, que automatizam a transição do computador entre programas. O **monitor residente** é um programa que fica o tempo todo na memória. Quando um programa em execução termina, ele avisa o monitor, que, por sua vez, automaticamente carrega o próximo programa e inicia a execução dele. Assim, o tempo em que o computador fica parado diminui. A transição entre programas é mais rápida, pois dispensa a operação manual.

Para que o monitor residente consiga substituir o operador na transição dos jobs, ele precisa saber o que fazer. Ele precisa saber qual programa deve ser carregado e executado a seguir. São os **cartões de controle** que fornecem essa informação. Além de informar ao monitor residente o que fazer a seguir, eles podem ser utilizados para outras informações, como por exemplo tempo máximo de execução do programa, identificação do usuário, etc. Na verdade, os cartões de controle são a origem das linguagens de comandos utilizadas em sistemas atuais.

O monitor residente também é o local indicado para as rotinas de acesso aos periféricos. São rotinas utilizadas por todos os programas. Com elas no monitor residente, as aplicações não precisam acessar diretamente os periféricos. Apenas

chamar a rotina apropriada dentro do monitor. Esse é o início da idéia de chamada de sistema.

Na década de 60, a partir do monitor residente, surgiu o conceito de **multiprogramação**. No monitor residente, apenas um programa é executado de cada vez. Quando ele precisa fazer alguma entrada e saída, o processador fica parado. Em geral, periféricos são dispositivos eletromecânicos e trabalham na faixa dos milisegundos. Ao mesmo tempo, o processador, que é um dispositivo eletrônico, que trabalha na faixa dos microsegundos. Por exemplo, enquanto é feito um único acesso À leitora de cartões, poderiam ser executadas 10.000 instruções de máquina ou mais. Por isso, a utilização do processador é muito baixa, e na maior parte do tempo, o programa está parado, esperando o término de uma operação de entrada ou saída.

A solução imagina foi, então, manter diversos programas na memória principal ao mesmo tempo. Quando um dos programas está esperando a conclusão de entrada e saída, outro programa inicia sua execução. Nesse caso, o tempo do processador é dividido entre diversoso programas. O objetivo disso é otimizar o hardware.

Duas inovações de hardware possibilitaram o desenvolvimento da multiprogramação. Em primeiro lugar o uso de **interrupções**. Quando um comando é enviado para um periférico, imediatamente o sistema operacional inicia a execução de um outro programa. É necessário que o periférico avise ao sistema Operacional quando o acesso tiver sido concluído. Isso se dá através da interrupção.

A outra inovação foram os discos magnéticos. Os jobs eram submetidos em cartões perfurados. Inicialmente, eles eram lidos pelo computador diretamente dos cartões. Mais tarde, passaram a ser antes copiados para fita magnética. Como a fita é mais rápida, esse mecanismo diminuiu o tempo de entrada e saída e, por consequência, o tempo de ociosidade do processador. Entretanto, leitoras de cartões e unidades de fita magnética são dispositivos essencialmente seqüênciais. Somente se pode ler o segundo job quando o primeiro é concluído.

O disco magnético permite a implementação da multiprogramação. Vários jobs são lidos de cartão perfurado ou fita magnética para o disco. Como o disco permite um *acesso direto a qualquer posição*, é possível ler parte do primeiro job e do segundo job. Á medida que, em função de sua execução, um job solicita leitura de mais dados, eles são buscados no disco. A execução de vários jobs pode agora ser sobreposta sem problemas.

Com a utilização de disco magnético, não havia mais a necessidade de reunir jobs semelhantes em lotes. O termo batch passou então a designar um sistema no qual nõa existe interação entre o usuário e a execução do programa. Hoje em dia, o termo batch foi substituído por "**execução em backgroud**".

Na década de 70 ocorre a disseminação de sistemas **timesharing**. Em um ambiente de multiprogramação diversos programas dividem o tempo do processador. Em um sistema timesharing, além da multiprogramação, cada usuário possui um terminal, e através desse terminal o usuário pode interagir com o programa em execução.

Cada usuário, em seu terminal, tem a sensação de possuir o computador apenas para seus programas. Esse compartilhamento é possível pois usuários em terminais consomem pouco tempo de processador. Enquanto o usuário pensa, bate papo ou toma café, ele não está usando o processador. Mas não somente quando o Usuário pára que o processador está ocioso. Considere um processador de texto. Entre o teclar e outro de um usuário, existe muito tempo de processador desperdiçado que pode ser utilizado em timesharing.

Historicamente, sistemas operacionais preocuparam-se principalmente com a eficiência no uso do computador. Atualmente existe também a preocupação com a usabilidade, comodidade e conforto do usuário.

A interface tradicional do interpretador de comandos é bastante dura, baseada em comandos mnemônicos e em seus parâmetros, que, obviamente

devem ser memorizados pelo usuário, dificultando seu uso. Atualmente, existe grande ênfase em interfaces baseadas em menus de comando, mouse, janelas e ícones. Tudo isso para simplificar a utilização do computador.

Histórico da interface gráfica

Tudo começou nos anos 70, quando um centro de pesquisas da Xerox Corporation, localizado em Palo Alto, Califórnia, perto da Universidade de Stanford, começou a investigar as interfaces de usuário do futuro. O PARC, como era chamado (Palo Alto Research Center), tinha entre seus pesquisadores verdadeiros gênios e pioneiros da computação, como Alan Kay e Douglas Engelberg. Engelberg liderou o desenvolvimento de uma nova estação de trabalho gráfica, chamada Xerox Star, cujo sistema operacional e interface de usuário eram inteiramente gráficas, baseadas em uma metáfora visual (a organização e a forma de trabalho de uma pessoa em uma escrivaninha). Assim, os pesquisadores do PARC "só" inventaram: o mouse, o sistema de janelas, os menus do tipo "drop-down", e os ícones !

Xerox 8010 Star Information Computer System (1981). Tinha até 6 terminais com 1.5 MB de memória e 40 MB de disco rígido em cada estação. Funcionava em rede. O servidor tinha 600 MB de memória.

Comercialmente, a Xerox Star foi um fracasso. Era revolucionária demais, e foi muito mal divulgada pela Xerox (que, também, não era um fabricante muito conhecido e bem posicionado na área de computadores). O projeto foi arquivado e o PARC quase foi fechado. Mas, o brilho e a genialidade das invenções do grupo de "Doug"Engelberg viraram um mito na comunidade de Informática, e, assim, quando a empresa Apple Computer, fundada por Steve Wozniak e Steven Jobs, resolveu desenvolver o PC do futuro, ela foi buscar inspiração no PARC. A primeira tentativa da Apple, um microcomputador chamado LISA (Local Integrated System Architecture), de 16 bits, incorporava o primeiro sistema operacional baseado em janelas do mercado, e foi lançado mais ou menos na mesma época que o primeiro IBM-PC (1981). Também foi um fracasso comercial. Jobs, então, liderou um processo de reengenharia radical do produto, que levou ao famoso Macintosh, considerado por muitos uma das maiores revoluções da história da Informática, pois integrava de forma absolutamente brilhante a nova interface gráfica, o pequeno tamanho, a alta potência de memória, gráficos de alta resolução e memória, e o marketing competentíssimo. Muito antes do IBM-PC (que usava um sistema operacional muito antiquado, não gráfico, e sem mouse, o não menos famoso MS-DOS), já usava o disquete de 3", por exemplo.

O Macintosh penetrou rapidamente o mercado acadêmico dos EUA, e, em menor extensão, o de automação de escritórios. Mas, foi imediatamente prejudicado pela política aberta da IBM em relação ao seu produto, que permitiu o surgimento dos clones. Em poucos anos, a participação do Macintosh no mercado mundial começou a declinar, indo de 30 %, em seu auge, a menos de 8 %, hoje. Outro erro da Apple foi ignorar a importância do software no futuro. A prova é que o hardware (o computador) fica cada vez menos importante, e surgem gigantes como a Microsoft, dedicadas a um conceito, ao invés de um produto eletrônico.

Windows 1.0 foi o primeiro da família Windows. Inicialmente, o Windows não era um sistema operacional próprio, mas sim uma interface entre o DOS e o usuário. Suas funcionalidades eram relativamente limitadas.

O Windows 2.0 foi um sistema operacional da Microsoft distribuído com o software para scanners da Hewlett-Packard. Era usado em plataformas Intel, com um sistema de reconhecimento de caracter. Foi lançado em 2 de abril de 1987 e substituído, posteriormente, pelo Windows 3.0 em 1990.

O Windows 2.0 foi lançado em 1987 e praticamente tem a mesma interface do Windows 1.0, com a diferença de ter mais recursos e ferramentas. Existem mais duas versões especiais do Windows 2.0:

-Windows 2.0/386 - Versão do Windows 2.0 lançada em 1988, otimizada para aproveitar todos os recursos dos microprocessadores 386;

-Windows 2.0/286 - Versão do Windows 2.0 lancada em 1989, em 7 disquetes de 3,5 polegadas de 720 KB cada um, otimizada para aproveitar todo o potencial dos microprocessadores 286.

Presente

Uma das áreas de pesquisa mais importantes atualmente são os sistemas operacionais distribuídos. Em um sistema desse tipo, vários computadores estão interconectados através de uma rede de comunicação de algum tipo, sendo possível, a partir de um dos computadores, acessar recursos de outros.

Exercício – Citar vantagens e desvantagens da operação de Sistemas Operacionais em rede comparada a operação clássica stand-alone.