FUNDAMENTOS DA COMPUTAÇÃO

Prof. Dr. Ruy Ferreira (ruy@ufmt.br / prof_ruy@hotmail.com / @profruy)

Texto produzido com base no livro Beck, L. System Software An Introduction do Systems Programming, Addison Wesley, 1997, para apoiar o ensino da disciplina-título.

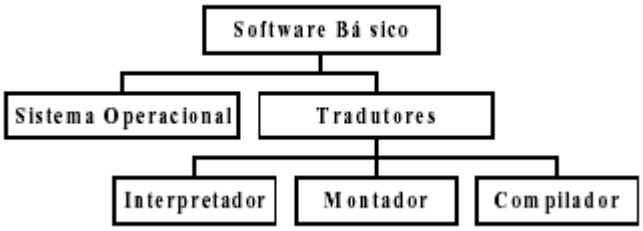
FAMILIARIZAÇÃO COM O USO DE SISTEMAS E AMBIENTES OPERACIONAIS

SOFTWARE

O software é toda parte lógica do computador. Fazem parte do software: os programas, o sistema operacional, os dados, o compilador, o assembler, o interpretador, etc. O software é utilizado para gerir o funcionamento do computador e ampliar sua potencialidade, para que possamos ter a solução de um problema. Podemos dividir o software em três grupos: software básico, software utilitário e software aplicativo.

Software Básico

É um conjunto de programas que define o padrão de comportamento do equipamento, tornando-o utilizável, ou seja, são os programas usados para permitir o funcionamento do hardware. O software básico é orientado para a máquina e torna possível a operação e a própria programação do computador. Seus programas se destinam a realizar tarefas básicas do computador, como: acionar periféricos, gerenciar buffers, mapear memória, manter o relógio e a data, etc.



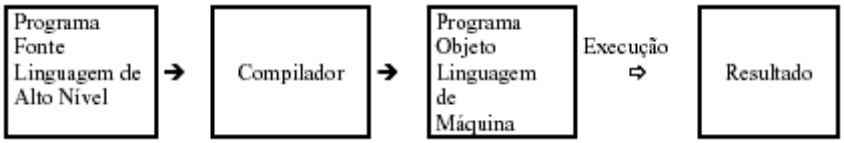
Tradutores

Mas, se os computadores trabalham internamente com a linguagem de máquina, como é que podemos fazer programas usando linguagem de baixo ou de alto nível?

É que existem tradutores que lêem uma linguagem de programação e a transformam para a linguagem de máquina. São programas responsáveis pela tradução da linguagem conhecida pelo homem para a linguagem conhecida pelo computador (código binário ou linguagem de máquina). Esses programas transformam programas escritos em linguagem de alto nível (programa fonte) em linguagem de máquina (programa objeto).

Existem três tipos de tradutores:

Compilador: Programa que traduz as instruções escritas em uma linguagem de programação legível como o Pascal ou Basic e transforma em um programa executável que o computador consegue entender e processar diretamente. O compilador transforma um programa fonte em programa objeto e somente depois de gerado integralmente é que será executado.



Interpretador: É o tradutor de uma linguagem de programação de alto nível, que converte as instruções para a linguagem de máquina, mas não cria uma versão executável do programa. Os interpretadores traduzem e executam os programas ao mesmo tempo. Em geral são mais lentos que os compiladores, já que aqueles fazem a tradução de uma única vez e, a partir daí, executam diretamente o programa traduzido.

Montador: Responsável pela montagem do programa fonte, gerando assim o programa objeto, ou seja, monta um programa em linguagem de baixo nível de forma que se obtém um programa em linguagem de máquina. Podemos citar como exemplo o montador Assembler que é um programa que traduz programas escritos na linguagem Assembly. Em síntese:

- **Montador**: lê uma linguagem de baixo nível e transforma para linguagem de máquina.
- **Interpretador**: lê uma linguagem de alto nível e transforma para linguagem de máquina.
- **Compilador**: lê uma linguagem de alto nível e transforma para linguagem de máquina.

Mas qual é mesmo a diferença entre interpretador e compilador?

Compilador	Interpretador
Lê e analisa todo o programa fonte (escrito em linguagem de alto nível) e traduz para linguagem de máquina.	Interpreta cada comando e executa. Faz linha a linha. Não traduz todo o programa para depois executar.
Cria um programa objeto que corresponde às instruções em linguagem de máquina.	Não gera programa objeto.
Executa-se direto o programa objeto.	Executa-se o programa fonte e sempre é necessário interpretar antes.
Traduz tudo de uma vez. Se encontrar erro, é preciso voltar ao programa fonte, corrigir, recompilar e executar novamente o programa objeto.	Se encontrar erro avisa na hora. Então, se edita o programa fonte, corrige-se o erro e interpreta-se novamente.

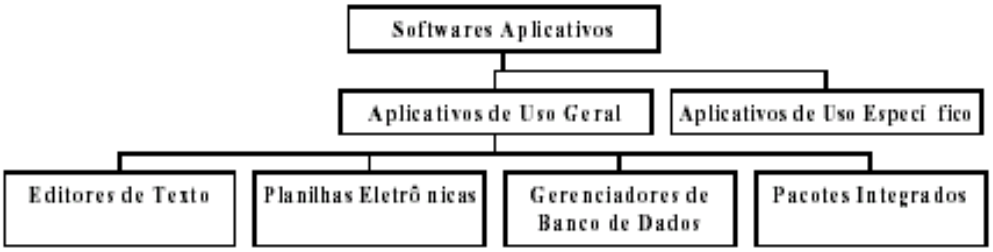
Utilitários

São programas que ampliam os recursos do sistema facilitando o uso e auxiliando a manutenção de programas. Administram o ambiente oferecendo ferramentas ao usuário para organizar os discos, verificar memória, corrigir falhar, etc. Por exemplo:

- Formatadores; Programas de backup; Compactadores de disco;
- Save Smart: grava automaticamente a tela de trabalho do usuário quando a máquina é desligada e quando ela é ligada, retorna ao mesmo ponto onde estava.
- Desfragmentadores: regravam de forma mais eficiente os arquivos que foram fragmentados pelo sistema operacional. Ou seja, faz com que um arquivo que foi armazenado em "pedaços" seja armazenado de forma contígua;
- Antivírus: detectam a presença de algum vírus e tenta eliminá-lo.

Estes programas recebem o nome de utilitários por serem úteis ao sistema computacional.

Software Aplicativo

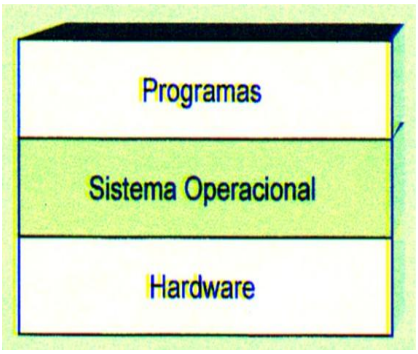


São os programas voltados para a solução de problemas do usuário. Podem ser de:

- **Uso geral**: são programas que podem ser utilizados em vários tipos de aplicações. Exemplos: editores de texto, gráficos, planilhas, gerenciadores de banco de dados, etc.
- **Uso específico**: se destinam exclusivamente a um único tipo de aplicação. Exemplos: folha de pagamento, crediário, imposto de renda, cadastro, contas a pagar e receber, etc.

Sistema Operacional

São os programas que gerenciam todos os sistemas internos da máquina, supervisionando o funcionamento de todo o sistema e administrando os recursos e facilidades do computador. São as funções básicas que o computador realiza tais como conhecer os seus periféricos e realizar tarefas inerentes a ele, como copiar, apagar, mover, renomear arquivos, etc.



O sistema operacional é a camada intermediária entre os programas dos usuários e o hardware, a máquina em si.

Sistemas operacionais monolíticos

O SO é organizado como uma coleção de processos seqüenciais cooperantes, que recebem as solicitações dos usuários (chamadas de sistema), as executam e devolvem um resultado.

Longe de qualquer organização convencional, um sistema operacional monolítico pode ser dito como uma grande desordem. O sistema operacional é escrito como uma coleção de procedimentos (*procedure)* e cada um destes procedimentos podem chamar qualquer outro. Quando esta técnica é utilizada, cada *procedure* no sistema tem a sua interface bem definida em termos de parâmetros e resultados.

Durante a construção de um sistema monolítico, todos os procedimentos são compilados e linkeditados em um único arquivo objeto. Não há nenhum escalonamento e todas as *procedures* são sempre visíveis, mas mesmo assim, existe uma pequena estrutura: os serviços (*system calls*) fornecidos pelo sistema operacional são requisitados pela colocação de parâmetros em lugares bem conhecidos (registradores ou *stack*) e então executa uma instrução especial de *trap* conhecida como ***kernel call*** ou ***supervisor call*** e transfere o controle para o sistema operacional.

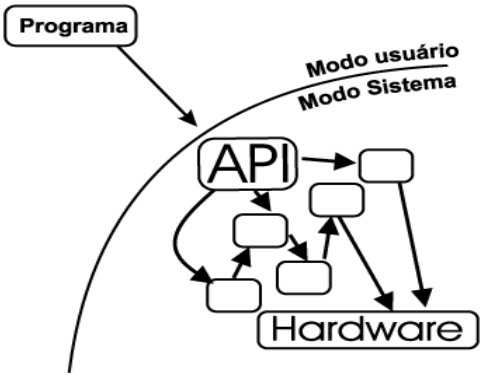
A maioria das CPU tem dois modos: *kernel mode*, onde todas as instruções são permitidas e o *user mode* para os programas do usuário e onde nem todas as instruções são permitidas. O sistema operacional examina os parâmetros da chamada para determinar qual *procedure* deve ser executada. Em seguida o sistema indexa uma tabela que contém um ponteiro para a *procedure* que em seguida é executada. Ao término da execução o controle é retornado ao programa do usuário.

Esta organização pode ser traduzida na seguinte estrutura básica:

1. Um programa principal (*main*) que chama os serviços;
2. Um conjunto de serviços (*procedures*) que executam a *system call*;
3. Um conjunto de procedures utilitárias que ajudam as procedures de serviço.

Os Sistemas Monolíticos permitem livre ação entre suas rotinas; oferecem facilidade de implementação e dificuldade de manutenção.

Esquematicamente são:



A API (Application Programming Interface ou Interface de Programação de Aplicações), é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

De modo geral, a API é composta por uma série de funções acessíveis somente por programação, e que permitem utilizar características do software menos evidentes ao utilizador tradicional.

Por exemplo, um sistema operacional possui uma grande quantidade de funções na API, que permitem ao programador criar janelas, acessar arquivos, criptografar dados, etc.. Mas a API dos sistemas operacionais costumam ser dissociada de tarefas mais essenciais, como manipulação de blocos de memória e acesso a dispositivos. Estas tarefas são atributos do núcleo de sistema, e raramente são programáveis. Outro exemplo de uso das API: programas de desenho geométrico que possuem uma API específica para criar automaticamente entidades de acordo com padrões definidos pelo utilizador.

Mais recentemente o uso de API tem se generalizado nos plugins, acessórios que complementam a funcionalidade de um programa. Os autores do programa principal fornecem uma API específica para que outros autores criem plugins, estendendo as funcionalidades do programa.

Sistemas operacionais em camadas

O primeiro sistema operacional que utilizou uma hierarquia de camadas foi o **THE** (Technische Hogeschool Eindhoven) desenvolvido em 1968 na Holanda por E. W. Dijkstra e seus alunos. Era um sistema operacional para processamento em lotes (*batch*) criado para um computador holandês, o Electrologica X8.

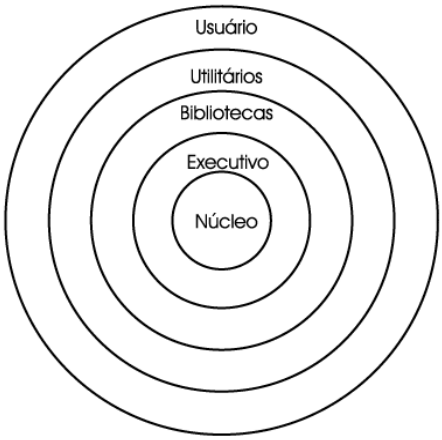
O sistema tinha seis camadas:

- Camada zero: que implemanta a multiprogramação básica da CPU, gerenciando a alocação do processador, interrupções e limite de tempo de cada processo. Essa camada era a responsável pela execução de múltiplos processos em uma única CPU;
- Camada um: cuidava do gerenciamento de memória utilizando recursos de paginação;
- Camada dois: cuida da comunicação entre o processo e o operador de console;

- Camada três: gerencia os dispositivos de E/S e "bufferizava" as *streams* de informação;
- Camada quatro: onde se executam os programas do usuário sem a preocupação de gerenciar um ambiente de multiprogramação, gerenciamento de memória, E/S e comunicação;
- Camada cinco: operador.

A Multiprogramação permite que uma aplicação seja dividida em dois ou mais processos e suporte vários usuários.

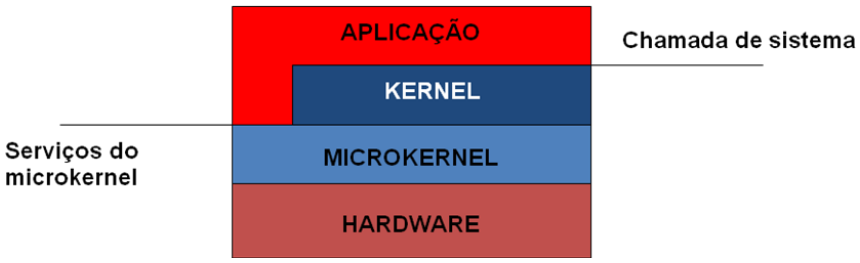
O sistema **Multics** também foi construído utilizando os conceitos de camadas e implementava um sistema em anéis onde os mais internos tinham maior prioridade que os mais externos.



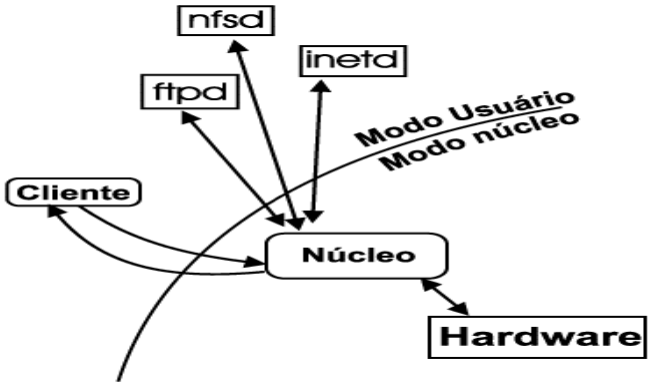
Nos sistemas em camadas/anéis é possível isolar funções do sistema das funções do usuário; É fácil de manter e atualizar; Uma camada só acessa sua inferior.

Modelo Cliente/Servidor

Os sistemas operacionais mais modernos têm adotado a filosofia de mover a maior parte possível do código para as camadas mais altas, tornando o ***Kernel*** (núcleo) mínimo. Isso é possível ao implementar a maior parte do sistema operacional em processos de usuários. Desta forma, para executar a leitura ou gravação de um bloco de arquivo, o processo do usuário (processo cliente) envia um pedido ao processo servidor que executa o trabalho e retorna a resposta. Tudo que o *Kernel* faz é manipular a comunicação.



Outra grande vantagem do modelo cliente-servidor é que se adapta bem em sistemas distribuídos. Em ambientes distribuídos, o *Kernel* também gerencia a origem e o destino na comunicação.



O Sistema Cliente/Servidor é o que permite mais fácil manutenção. Faz a utilização de micronúcleo (microkernel). As Funções do sistema são servidores (daemons) rodando em modo usuário. Oferece maior segurança.

Máquinas virtuais

As primeiras versões do sistema operacional OS/360 da IBM foram projetadas exclusivamente para processamento em lotes (*batch*). Como uma quantidade significativa de usuários desejava um sistema *timesharing* para este equipamento, grupos dentro e fora da IBM decidiram escrever sistemas operacionais com essa tecnologia.

A versão oficial de um sistema *timesharing* da IBM, o TSS/360, chegou tarde e era um sistema muito grande e pesado no sentido que consumia muitos recursos do computador. Tanto, que poucos centros de computação aderiram a ele. Mas um grupo no IBM's Cientifics Center em Cambridge produziu um sistema operacional radicalmente diferente. Esse sistema originalmente chamado de CP/CMS e posteriormente de VM/370 foi baseado numa inteligente observação: um sistema *timesharing* fornece (1) multiprogramação e (2) máquina estendida, com uma interface

mais conveniente que o hardware fornecia. A essência do VM/370 era completar separadamente estas duas funções.

O coração do sistema era o ***Virtual Machine Monitor*** que proporcionava multiprogramação e provia várias máquinas virtuais. Diferente dos outros sistemas operacionais, cada máquina virtual eram cópias exatas do hardware verdadeiro, incluindo *kernel mode* e *user mode*, interrupções e tudo mais que uma máquina real teria.

Uma vez que uma máquina virtual era idêntica ao hardware, ela podia executar qualquer sistema operacional que rodasse naquele equipamento. O VM/370 ganhou muito em simplicidade movendo a grande parte do código do sistema operacional para as camadas mais altas, o CMS.

Funções do sistema operacional

Entre outras um sistema operacional desempenha as seguintes funções:

Solicitação de serviços:

- Através de endereços fixos
- Através de interrupções por software
- Interrupção pode ser causada por instruções como SVC (chamada ao sistema)
- Interrupção comuta máquina do modo usuário para o modo supervisor
- Em modo supervisor todas as instruções da máquina podem ser executadas
- Instruções que só podem ser executadas em modo supervisor são instruções privilegiadas

Processamento de interrupções:

- Uma interrupção é um sinal que altera o fluxo normal de execução de um programa
- A interrupção automaticamente transfere o controle para uma rotina de processamento da interrupção
- As rotinas de processamento normalmente são parte do sistema operacional
- Após execução da rotina, o controle é transferido para o programa
- As interrupções podem ser assíncronas em relação ao programa

- As interrupções podem ser decorrentes da instrução SVC, por alguma condição identificada durante a execução de um programa ou por um dispositivo
- Quando uma interrupção ocorre o estado da CPU é salvo e o controle é transferido para a rotina de tratamento
- Os registradores de estado contêm uma variedade de informações importantes ao tratamento das interrupções
- A cada interrupção é atribuído um nível de prioridade
- O nível de prioridade pode ser determinado pela CPU ou por um dispositivo externo

Escalonamento de processos:

- Um processo é um programa em execução
- Em um sistema multiprogramado, múltiplos processos competem pela CPU
- O escalonador é responsável por definir o processo a ser executado no momento
- Durante a sua existência, um processo pode se encontrar em diferentes estados
- Em um computador com uma única CPU, não pode existir mais de um processo em execução
- Quando um processo sai de execução, informações são armazenadas no bloco de estado (PSB - Path State Block)
- Existem diversos algoritmos de escalonamento com *time-slice* (cada processo tem uma fatia de tempo da CPU), *round robin* (um dos mais antigos e simples algoritmos de escalonamento que atribui frações de tempo para cada processo na CPU), prioridades.
- Escalonamento pode ser: preemptivo ou não preemptivo.

Supervisão de entrada e saída de dados:

- A entrada e saída de dados podem ser feita byte a byte sob controle da CPU ou podem existir processadores dedicados a esta atividade
- As operações a serem executadas em um canal de entrada e saída de dados são definidas em um programa para o canal

Gerência de memória real:

- A memória real é compartilhada entre os diversos processos
- A memória alocada a um processo precisa ser protegida dos outros processos
- A proteção da memória depende do sistema operacional e do suporte provido pelo hardware
- São necessárias técnicas para combater a fragmentação
- O combate à fragmentação requer a realocação
- A memória pode ser dividida em partições com tamanho fixo ou variável

Gerência de memória virtual:

- Programas podem usar um espaço de endereçamento maior que o espaço físico
- A memória virtual pode ser maior que a memória efetivamente existente
- Uma abordagem comum na implementação de memória virtual é o uso de paginação por demanda
- A memória é dividida em páginas de tamanho fixo
- A conversão entre endereços virtuais e reais é feita através de um dispositivo de tradução, para isso é usada uma tabela de tradução
- Algoritmos são implementados com o objetivo de identificar a página a ser removida como o *least recently used* (menos recentemente usada)
- As tabelas de tradução são freqüentemente implementadas em memória associativa nas unidades de gerência de memória
- O sistema operacional precisa evitar a situação de *thrashing* mantendo em memória as páginas que fazem parte dos conjuntos de trabalho
- Além da paginação, é possível a gerência de memória usando segmentação ou segmentação com paginação

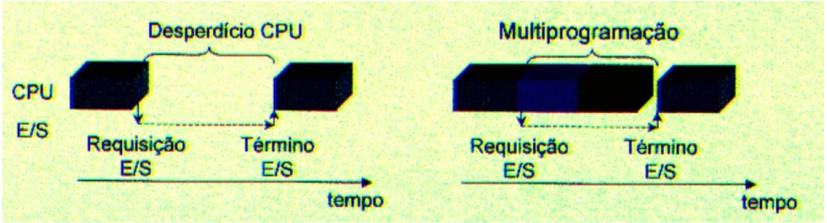
Processamento de arquivos:

- SO faz a intermediação do acesso ao sistema de arquivos
- Solicitações do programa são traduzidas em acesso às controladoras de disco
- Mantém informações sobre a estrutura e as localizações dos arquivos

- Áreas de *buffer* (retentor) podem ser usadas para agilizar os acessos aos arquivos (é uma região de memória temporária utilizada para escrita e leitura de dados)
- O *double buffering* pode ser usado de modo a possibilitar o *overlap* (sobrepor - realizar várias operações simultaneamente) do processamento de um bloco com a leitura do próximo bloco. A vantagem de se usar dois *buffers* ao invés de um só é muito simples: enquanto o computador armazena dados em um dos *buffers*, ele pode ler no outro. É como se fosse um revezamento, o PC grava dados no *Buffer* A e lê no *Buffer* B; depois, lê no *Buffer* A e grava no B. Dessa forma ele nunca para e simplesmente corta pela metade o tempo que gastaria.

Nível de multiprogramação:

- Nível de multiprogramação pode ser limitado através de um sistema de escalonamento em dois níveis
- Desempenho cai após um determinado nível de multiprogramação
- Nível de multiprogramação varia de acordo com as características dos processos em execução
- Sistema de escalonamento em três níveis facilita a redução do nível de multiprogramação.
- Sistemas de escalonamento procuram maximizar o *throughput* (taxa de transferência) e minimizar o *turnaround time* (tempo de duração da ação) e o *response time* (tempo de resposta)



Alocação de recursos:

- Facilidade pode ser usada para controlar a alocação de recursos definidos pelos usuários

- Sistema operacional responde à solicitação de controle de um recurso verificando se o recurso encontra-se atribuído a algum outro processo
- Se o recurso estiver alocado, o processo é bloqueado
- Se o recurso estiver disponível, é alocado e o controle retornado ao processo
- Mecanismos podem ser implementados com o objetivo de evitar situações de *deadlock* (bloqueio mútuo, impasse).

Proteção:

- Usuários devem ser protegidos de ações não autorizadas de outros usuários
- Sistemas operacionais programam mecanismos de controle de acesso
- Mecanismos de controle de acesso normalmente se baseiam em uma matriz de acesso
- Informações são normalmente armazenadas como uma lista de autorizações ou uma lista de capacidades
- Mecanismos de controle de acesso dependem de mecanismos de autenticação

Interface com o usuário:

- Linguagem de comando
- Representação gráfica baseada em menus e janelas

Ambiente de execução:

- Rotinas disponíveis em tempo de execução
- Facilidades para alocação dos recursos aos programas
- Muitas das rotinas voltadas para entrada e saída de dados
- Rotinas definem uma máquina estendida
- A máquina estendida é uma máquina virtual
- Cada programa tem a ilusão de ter a sua própria máquina

Tipos de sistemas operacionais:

- *Single-job* (O sistema operacional era um programa monitor, usado para enfileirar tarefas ou *Jobs*).
- Multiprogramado

    - *Time-sharing*
    - *Batch*
- Multiprocessado
- Rede
- Distribuído
- Orientado a objeto

Sistemas operacionais multiprogramados:

- Manter mais de um programa em "execução" simultaneamente
- Duas inovações de *hardware* possibilitam o surgimento da multiprogramação
    - Interrupção
    - Discos magnéticos
- Rotinas de I/O fornecidas pelo sistema.
- Gerenciamento de memória
- O sistema deve alocar e liberar a memória para vários Jobs/programas.
- Escalonamento da CPU
- O sistema deve escolher entre os vários Jobs/programas qual está pronto para ser executado.
- Alocação dos dispositivos.

Sistemas orientados *Time-sharing*:

- Tipo de multiprogramação
- Usuários possuem um terminal
    - Interação com o programa de execução
- Ilusão de possuir a máquina dedicada à execução de seu programa
    - Divisão do tempo de processamento entre usuários
    - Tempo de resposta é importante

Sistemas orientados *Batch*:

- Introdução de operadores profissionais
    - Usuário era mais o operador da máquina
- A batelada de *Job*

    - Programa a ser compilador e executado, acompanhado dos dados de execução
    - *Jobs* é organizado em lote (*batch*)
    - Necessidades semelhantes (ex: mesmo compilador)
- Passagem entre diferente *Job* continua sendo manual

Sistemas operacionais multiprocessados:

- Existem múltiplos processadores no computador
- A arquitetura do computador pode ser fortemente ou fracamente acoplada
- Os sistemas operacionais distribuídos podem ser supervisores separados, mestre-escravo ou simétricos

Sistemas operacionais em REDE:

Um sistema operacional de rede pode ser visto como sendo formada por um conjunto de máquinas interligadas por uma *rede de comunicação*, cada uma rodando o seu próprio sistema operacional e compartilhando recursos. Um sistema operacional de rede pode ser construído a partir de um sistema operacional tradicional, com a incorporação de um conjunto de funções que permitem a comunicação entre os diferentes processadores e o acesso aos arquivos. Um exemplo significativo de sistema operacional de rede é Sun Solaris, que possui um sistema de arquivos distribuídos, o NFS (Network File System), que permite o acesso aos arquivos independentemente de sua localização física.

Sistemas operacionais distribuídos:

- Sistemas operacionais de rede não tornam a rede transparente aos usuários
- Sistemas operacionais distribuídos tornam a presença da rede transparente

Sistemas orientados a objetos:

- Sistemas operacionais são programados como coleções de objetos
- O modelo de objetos é implementado usando processos chamados servidores ou gerentes de objetos
- Os servidores são responsáveis por criar novos objetos e por destruir objetos que não sejam mais necessários
- O núcleo do sistema operacional programa atividades simples como a comunicação entre processos e escalonamento

- A maioria das atividades é realizada por objetos e não pelo *kernel*

CONCLUSÃO

Para concluir é necessário comentar sobre dois enfoques de sistemas operacionais em especial:

Sistemas paralelos

- Máquinas multiprocessadores possuem mais de um processador
- Sistemas fortemente acoplados
    - Processadores compartilham memória e relógios comuns
    - Comunicação é realizada através da memória
- Vantagens
    - Aumento de *troughput* (número de processos executados)
    - Aspectos econômicos
    - Aumento de confiabilidade

Sistema de tempo real

- Empregado para o controle de procedimentos que devem responder dentro de certo intervalo de tempo
    - Ex: experimentos científicos, tratamento de imagens médicas, controle de processos, etc.
- Noção de tempo real é dependente da aplicação
    - Milissegundos, minutos, horas, etc.
- Dois tipos
    - *Hard real time* - Tarefas críticas são completadas dentro de um intervalo de tempo
    - *Soft real time* - Tarefa crítica tem maior prioridade que as demais

Concluindo podemos afirmar que um Sistema Operacional é um Programa ou um conjunto de Programas cuja função é gerenciar os recursos do Sistema (definir qual programa recebe atenção do Processador, gerenciar Memória, criar um Sistema de Arquivos, etc.), além de fornecer uma Interface entre o Computador e o usuário. É o primeiro Programa que a Máquina executa no momento em que é ligada, num processo chamado de *Bootstrapping* e a partir de então, não deixa de funcionar até que o Computador seja desligado. O Sistema Operacional reveza sua execução com a de

outros Programas, como se estivesse vigiando, controlando e orquestrando todo o Processo Computacional.

No Apêndice um temos uma linha do tempo sobre os sistemas operacionais.

REFERÊNCIAS

BECK, Leland L. System Software - An Introduction to Systems Programming. 3ª Ed. Addison Wesley, 1997.

APÊNDICE 1

Sistemas Operacionais em Ordem Cronológica até 1999:

- 1951
  - LEO Lyons Electronic Office foi o desenvolvimento comercial do EDSAC plataforma de computação, apoiada pela empresa britânica J. Lyons and Co.
- 1954
  - MIT é o sistema operacional feito para UNIVAC 1103
- 1955
  - General Motors do sistema operacional feito para IBM 701
- 1956
  - GM-ANA I / O para IBM 704 com base na General Motors do Sistema Operacional
- 1957
  - Atlas Supervisor Universidade de Manchester *Atlas de início do projeto do computador*
  - BESYS Bell Labs para o IBM 7090 IBM 7094
- 1958
  - Universidade de Michigan Executive System UMES, para o IBM 704, 709 e 7090
- 1959
  - SHARE Sistema Operacional SOS, com base na GM-ANA E / S
- 1960
  - IBSYS IBM para o seu 7090 e 7094
- 1961
  - CTSS MIT Time-Sharing System compatível para o IBM 7094
  - MCP (Burroughs Programa de Controle Mestre)
- 1962
  - Atlas Supervisor Universidade de Manchester *Atlas computador comissionado*
  - GCOS GE Geral Integral sistema operacional, originalmente GECOS, General Electric Comprehensive Supervisor Operacional
- 1964
  - EXEC 8 UNIVAC
  - OS/360 primário SO's IBM para sua série S/360 *anunciado*
  - TOPS-10 dezembro o nome TOPS-10 não foi adotada até 1970
  - Berkeley Timesharing System para sistemas de dados científicos SDS 940
  - Dartmouth Time Sharing System Dartmouth College DTSS para computadores GE
- 1965
  - Multics GE MIT, Bell Labs para a GE-645 *anunciado*
  - BOS/360 Básica do Sistema Operacional da IBM

    - TOS/360 a fita do sistema operacional da IBM
- 1966
    - OS/360 primário SO's IBM para sua série S/360 PCP e MFT *fornecido*
    - DOS/360 do disco do sistema operacional da IBM
    - MS / 8 Richard F. Lary 's DEC PDP-8 do sistema
- 1967
    - CP / CMS IBM, também conhecida como CP-67
    - Michigan Terminal System MTS (sistema de partilha de tempo para a IBM S/360-67 e sucessores
    - ITS Incompatible Timesharing do Sistema do MIT para o DEC PDP-6 e PDP-10
    - ORVYL Sistema de partilha da Universidade de Stanford tempo para o IBM S/360
    - TSS/360 compartilhamento do Sistema de Tempo-IBM para o S/360-67, nunca oficialmente lançado, cancelado em 1969 e novamente em 1971
    - OS/360 MVT
    - ESPERA SAIL Stanford Laboratório de Inteligência Artificial, de partilha de tempo para o sistema DEC PDP-6 e PDP-10, depois TOPS-10
- 1968
    - Programa de Controle de avião ACP IBM
    - TSS-8 DEC para o PDP-8
    - O sistema de multiprogramação Technische Hogeschool Eindhoven
- 1969
    - TENEX Bolt Beranek and Newman para sistemas de dezembro, depois TOPS-20
    - Unics mais tarde Unix AT & T inicialmente em computadores DEC
    - RC Multiprogramação System 4000 RC
    - Multics MIT, GE, Bell Labs para a GE-645 e depois a Honeywell 6180 *aberto para clientes pagantes em Outubro*

- 1970
    - DOS-11 PDP-11
- 1971
    - RSTs-11 2A-19 *primeira versão lançada, PDP-11*
    - OS / 8
- 1972
    - RDOS
    - SVS
    - VM / CMS
- 1973
    - Эльбрус-1 Elbrus-1 computador soviético uЭль criado usando linguagem de alto nível-76 AL-76/ALGOL 68
    - VME linguagem de implementação S3 ALGOL 68
    - RSX-11D
    - RT-11
    - Alto OS
- 1974
    - DOS-11 V09-20C *versão estável passado, junho 1974*
    - SINTRAN III
    - MONECS

- 1975
  - CP / M
  - BS2000 V2.0 *Primeira versão lançada*
  - Sexta Unix Edition
- 1976
  - Cambridge computador PAC Todos os procedimentos do sistema operacional escrito em ALGOL 68C com alguns procedimentos estreitamente associado protegidos em BCPL
  - Cray Sistema Operacional
  - FLEX
  - TOPS-20
- 1977
  - 1BSD
  - KERNAL
  - OASIS sistema operacional
  - TRS-DOS
  - Memória Virtual VMS V1.0 *versão comercial inicial, 25 de outubro*
- 1978
  - 2BSD
  - Apple DOS
  - HDOS 1,0
  - Tripos
  - UCSD p-System *Primeira versão lançada*
  - Lisp Machine CADR
- 1979
  - Atari DOS
  - POS
  - NLTSS
  - UNIX/32V
  - Unix Versão 7

- 1980
  - CTOS
  - OS-9
  - 86-DOS
  - SOS
  - Piloto *Xerox Star*
  - Xenix
- 1981
  - PC-DOS
  - MS-DOS
  - UTS
  - Acorn MOS
  - Aegis SR1 *Primeira Apollo domínio de sistemas / enviado em 27 de março*
- 1982
  - Commodore DOS
  - LDOs Por Logical Systems, Inc. Para o Radio Shack TRS-80 Modelos I, II e III
  - Sun UNIX mais tarde SunOS 0,7

    - QNX
    - Ultrix
- 1983
    - Lisa Office System 07/07
    - Coerente
    - GNU *início do projeto*
    - Novell NetWare S-Net
    - ProDOS
    - SunOS 1,0
- 1984
    - Mac OS *sistema 1.0*
    - MSX-DOS
    - Sinclair QDOS
    - QNX
    - UNICOS
    - Venix 2,0
- 1985
    - AmigaOS
    - Atari TOS
    - DG / UX
    - MIPS OS
    - Oberon escrito em Oberon-2
    - SunOS 2,0
    - 8 Unix Versão
    - Windows 1,0
    - Xenix 2,0
- 1986
    - AIX 1,0
    - GS-OS
    - Gêneros 7,0
    - HP-UX
    - SunOS 3,0
    - GEOS
    - 9 Unix Versão
- 1987
    - Arthur
    - IRIX *3.0 é a versão SGI primeiro*
    - MINIX 1,0
    - BS2000 V9.0
    - OS / 2 1.0
    - PC-MOS/386
    - Windows 2,0
- 1988
    - A / UX Apple Computer
    - RISC iX
    - LynxOS
    - Mac OS *Sistema 6*
    - MVS / ESA
    - OS/400
    - SpartaDOS X

    - SunOS 4,0
    - TOPS-10 7.04 *última versão estável, Julho de 1988*
    - Helios 1,0
- 1989
    - EPOC
    - NEXTSTEP 1,0
    - RISC *Primeiro lançamento era para ser chamado Arthur 2, mas foi renomeado para RISC OS 2, e foi vendido pela primeira vez como RISC 2,00 em abril de 1989*
    - SCO UNIX *versão 3*
    - TSX-32
    - 10 Unix Versão
    - Xenix 2.3.4 *última versão estável*

- 1990
    - AmigaOS 2,0
    - BeOS v1
    - Gêneros 8,0
    - OSF / 1
    - AIX 3,0
    - Windows 3,0
- 1991
    - Linux
    - Mac OS *System 7*
    - MINIX 1,5
    - Penpoint OS
    - RISC OS
- 1992
    - 386BSD 0,1
    - AmigaOS 3,0
    - Amiga Unix 2.01 *Última versão estável*
    - RSTs / E 10,1 *última versão estável, Setembro de 1992*
    - Solaris 2.0 *Sucessor do SunOS 4.x, com base em SVR4 ao invés da BSD*
    - OpenVMS V1.0 *Primeira OpenVMS AXP (Alpha) versão específica, Novembro de 1992*
    - Plan 9 Primeira Edição *Primeira versão pública foi disponibilizada para as universidades*
    - Windows 3,1
- 1993
    - FreeBSD
    - NetBSD
    - Newton OS
    - Windows NT 3.1 *Primeira versão do kernel do Windows NT pública*
    - Open Genera 1,0
    - Sistema Operacional IBM 4690
    - Novell NetWare 4
    - Slackware 1,0
    - Primavera
- 1994
    - AIX 4.0, 4.1

    - RISC OS 3,5
    - NetBSD 1.0 *Primeira versão multi-plataforma, Outubro de 1994*
- 1995
    - Digital UNIX *aka* Tru64 UNIX
    - OpenBSD
    - OS/390
    - Plan 9 Second Edition *(segunda versão comercial foi disponibilizado ao público em geral)*
    - Ultrix 4.5 *última versão principal*
    - Windows 95
- 1996
    - Mac OS 7.6 *oficialmente pela primeira vez chamado Mac OS*
    - Windows NT 4.0
    - RISC OS 3,6
    - AIX 4,2
    - Palm OS
- 1997
    - Inferno
    - Mac OS 8
    - SkyOS
    - MINIX 2,0
    - RISC OS 3,7
    - AIX 4,3
- 1998
    - Solaris 7 *First 64-bit versão do Solaris. Nomes de essa queda "2.", caso contrário já teria sido Solaris 2.7*
    - Windows 98
    - RT-11 5.7 *versão estável passado, Outubro de 1998*
    - Novell NetWare
    - JUNOS
- 1999
    - AROS *Boot pela primeira vez em versão Stand Alone*
    - RISC OS
    - Mac OS 9
    - Windows 98 2 ª edição
    - Inferno Second Edition (*Inferno Lucent Business Unit)*
      Fonte: *http://pt.wikipedia.org*

O DOS sofreu uma grande queda quando a Microsoft lançou o Windows95 em 1995, mas muitos dos que se acostumaram às Linhas de

Comando não o largaram mais. O MS-DOS 3.3 lançado em 1987, é considerada a melhor versão do Sistema

Os Ambientes Operacionais Windows 3.x da família Microsoft Windows foram lançados entre 1990 e 1994. A versão 3.0 foi o primeiro sucesso amplo do Windows, permitindo que a Microsoft pudesse competir com a Apple, com o Macintosh e com o Commodore Amiga, que utilizavam Interfaces Gráficas. Não eram propriamente Sistemas Operacionais, pois tinham como pré-requisito a existência de uma versão do DOS – Disk Operating System – instalada no Computador

O Control Program for Microcomputers (CP/M), criado por Gary Kildall, estava presente no começo da Revolução da Computação Pessoal. Dois dos primeiros Softwares mais populares, o WordStar e o dBase, foram feitos para o CP/M. O Software também introduziu as opções de Linhas de Comando popularizadas pelo DOS. Fonte desta matéria: http://www.bertozi.com.

A Be Inc decidiu criar seu próprio Sistema Operacional, o BeOS para rodar na Plataforma Mac e mais tarde resolveu criar uma Máquina para rodar o Sistema, o BeBox. Quando o Processador usado no BeBox foi paralisado pela AT&T, a empresa ficou sem Hardware e sem acordo com a Apple. Em 2001, a Be foi comprada pela Palm, que infelizmente acabou com o BeOS

Muita coisa mudou no momento em que pudemos clicar no Botão “Salvar”, usando um Sistema Operacional Gráfico e muitos ainda lembram com carinho do Sistema Operacional da Microsoft: o Windows 95 foi um grande avanço para a Informática mundial

Em 1989, Steve Jobs fundou a NeXT Computer. O Sistema Operacional das Máquinas NeXT era baseado em Unix e oferecia uma camada orientada a objetos para Desenvolvedores. Em parceria com a Sun, o NeXTStep foi rebatizado de OpenStep e tornou-se a chave de um acordo feito em 1996 que trouxe Jobs de volta à Apple.

Hoje a função Multitarefa é comum em um Sistema Operacional, mas há 20 anos, era o Santo Graal das Plataformas de Computação Pessoal. Na época, as Máquinas Amiga eram usadas para dar suporte a seriados de TV, incluindo efeitos em tempo real nas transmissões. Hoje, o Sistema que pertence a Gateway ainda conta com fiéis seguidores.

Nascido em 1987, o OS/2 destacou-se até 1995, quando seu “rival”, o Windows 95 chegou para atrair todas as atenções. O OS/2 não vingou nos PCs, mas sobreviveu em ATMs como um bravo soldado até que a IBM o “aposentou” em 2001, mantendo o suporte até 2006.

O clássico Mac OS rodava em Chips Motorola, não era baseado em BSD Linux e era conhecido como System. Os clones ajudaram a Apple a dar ao System o novo nome pelo qual ele é conhecido até hoje. A partir da versão 7.6, ao ligar a Máquina com Chips PowerPC, o usuário via: “Mac OS” para saber que não se tratava de um clone.

(Extraído do site RetroPlayerBrazil: http://retroplayerbrazil.wordpress.com/2011/03/31/sistemas-operacionais-atraves-dos-tempos/)