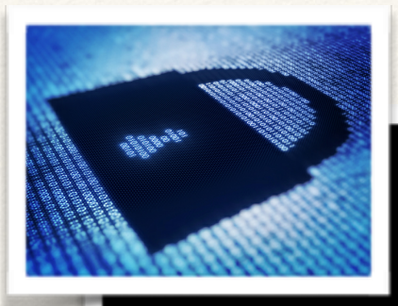
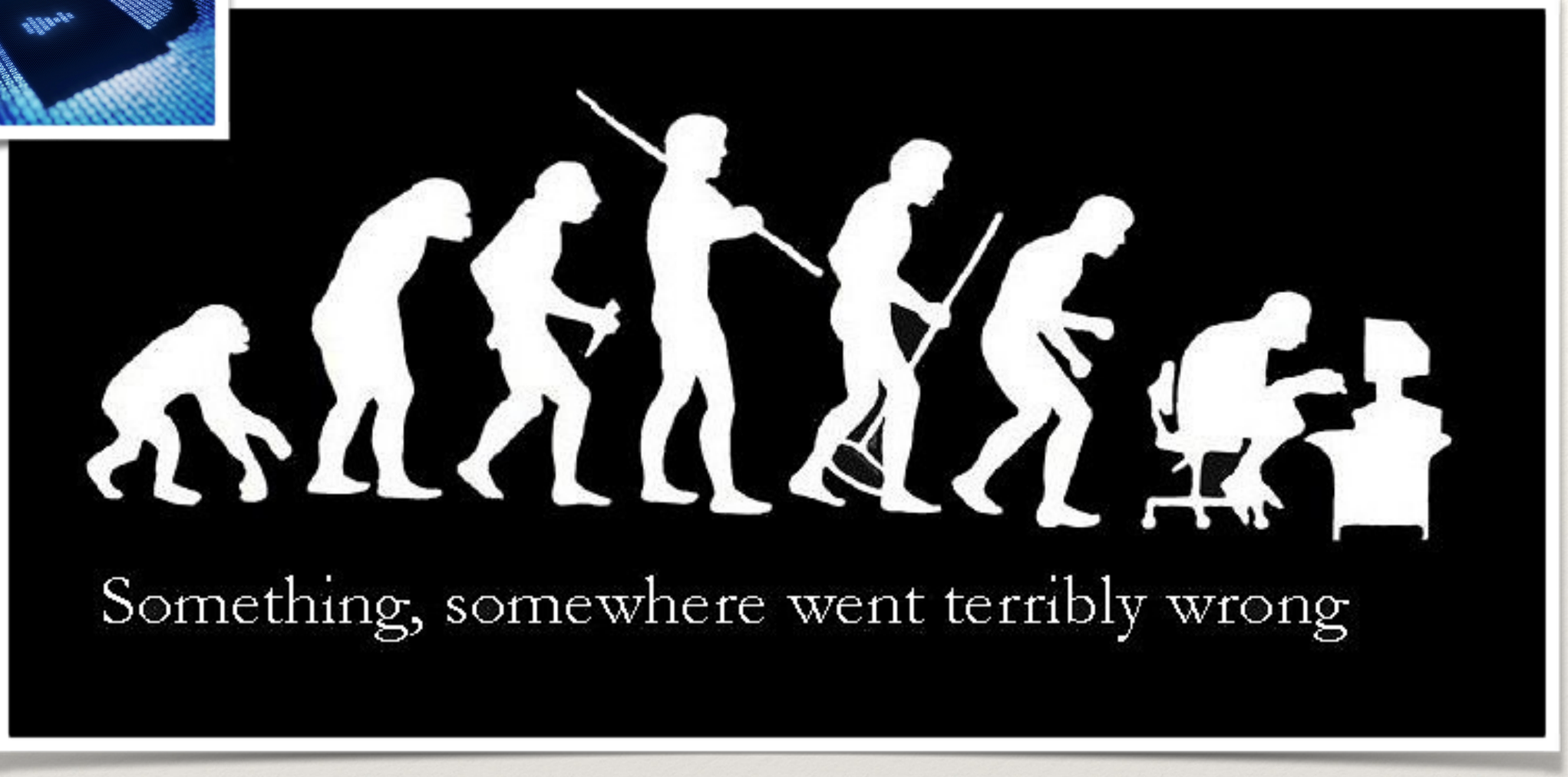
ULBRA
CAMPUS CACHOEIRA DO SUL





Estrutura de Sistemas Operacionais
e
Cliente Servidor

Professor
Wagner Gadêa Lorenz
wagnerglorenz@gmail.com

Disciplina: Sistemas Operacionais
Curso de Sistemas de Informação

Cachoeira do Sul, 05 de Março de 2015.

Estrutura de Sistemas Operacionais

Agora que já tivemos uma noção inicial de sistemas operacionais e sua evolução no decorrer das décadas, vamos conhecer as diferentes estruturas internas dos sistemas operacionais.

Os cinco grupos abordados serão os seguintes: **sistemas monolíticos, sistemas em camadas, máquinas virtuais, exonúcleos e sistemas cliente-servidor**.

Sistemas Monolíticos

A organização **monolítica** é de longe a mais comum, mas poderia muito bem ter o subtítulo de “A grande bagunça”.

Simplesmente n**ão há estruturação**. O sistema operacional é escrito como uma **coleção** de **procedimentos**, sendo que cada um pode **chamar um dos demais sempre que necessário**.

Nessa abordagem cada procedimento do sistema tem uma interface bem definida quanto a parâmetros e resultados e cada um deles é livre para chamar qualquer outro, se este oferecer alguma computação útil de que o primeiro necessite.

Sistemas Monolíticos

Para construir o programa-objeto real do sistema operacional usando essa abordagem, **primeiro compilam-se todos os procedimentos individualmente, ou os arquivos que contêm os procedimentos**.

Então, **juntam-se todos** em um **único arquivo-objeto** usando o ligador (*linker*) do sistema.

Não existe essencialmente **ocultação** de **informação**, **todos** os **procedimentos são visíveis uns aos outros** (o oposto de uma estrutura de módulos ou packages, na qual muito da informação é ocultada dentro de módulos e somente os pontos de entrada designados podem ser chamados do lado de fora do módulo).

Sistemas Monolíticos

Contudo, mesmo em sistemas operacionais **monolíticos** é possível ter um **mínimo** de **estrutura**.

Os **serviços** (**chamadas** ao **sistema**) providos pelo sistema operacional são requisitados colocando-se os parâmetros em um local bem definido (na pilha, por exemplo) e então executando uma instrução de desvio de controle (*trap*).

Essa instrução altera a máquina do modo usuário para o modo núcleo e transfere o controle para o sistema operacional.

O sistema operacional busca então os parâmetros e determina qual chamada ao sistema será executada.

Depois disso, ele indexa uma tabela que contém na linha *k* um ponteiro para o procedimento que executa a chamada ao sistema *k*.

Sistemas Monolíticos

Essa organização sugere uma **estrutura básica** para o **sistema operacional**:

- Um programa principal que invoca o procedimento do serviço requisitado;
- Um conjunto de procedimentos de serviços que executam as chamadas ao sistema;
- Um conjunto de procedimentos utilitários que auxiliam os procedimentos de serviços.

Sistemas Monolíticos

Segundo esse modelo, **para cada chamada ao sistema há um procedimento de serviços que se encarrega dela**.

Os **procedimentos utilitários** realizam **tarefas necessárias** para os vários **procedimentos** de **serviço**, como buscar dados dos programas dos usuários.

Sistemas Monolíticos

Divisão de procedimentos em três camadas.

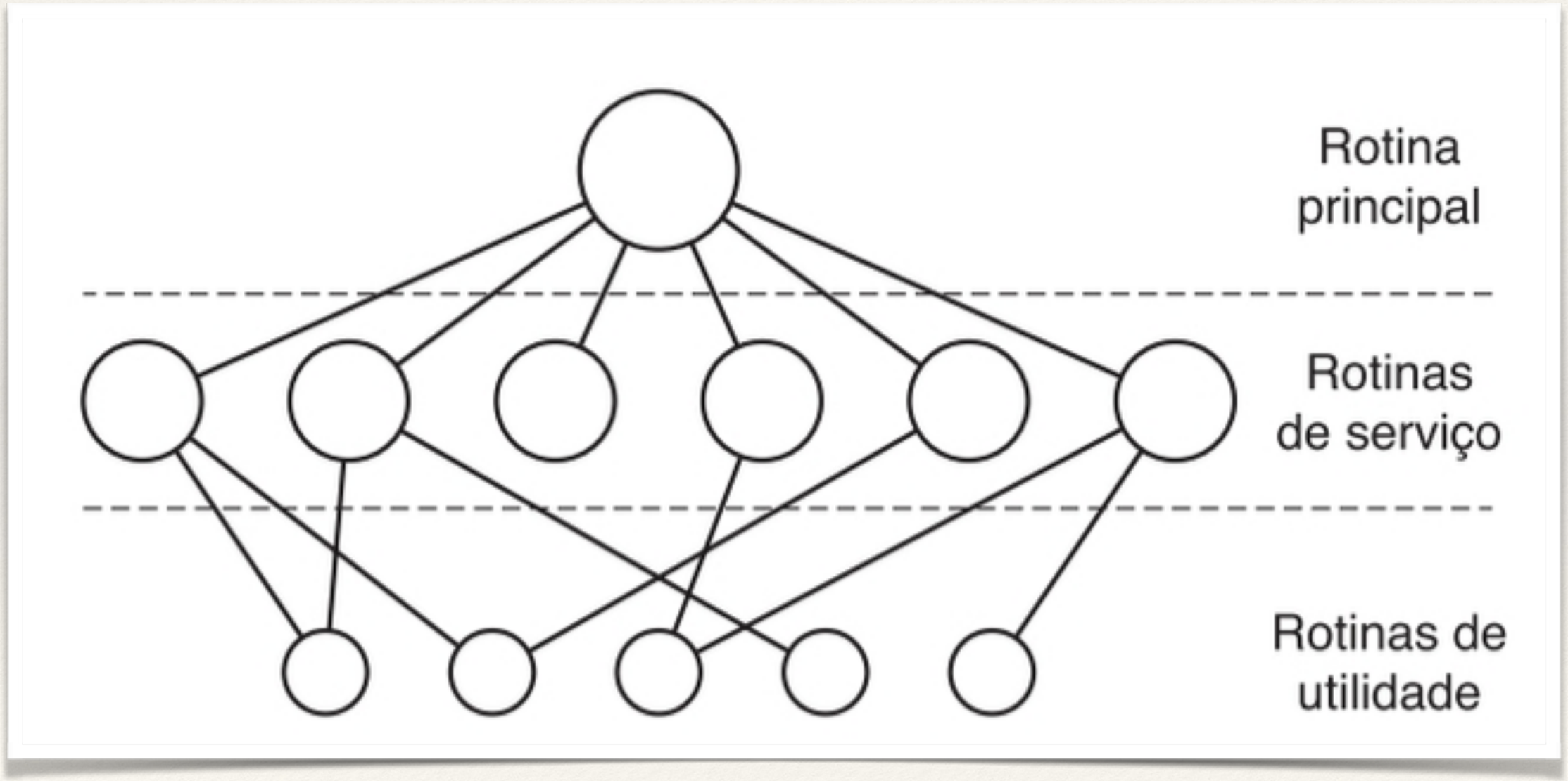


Figura. Um modelo simples de estruturação de um sistema monolítico.

Sistemas de Camadas

Uma generalização da abordagem da Figura anterior é a organização do sistema operacional como uma **hierarquia de camadas**, **cada uma delas construída sobre a camada imediatamente inferior.**

O primeiro sistema construído dessa maneira foi o **THE**, cuja sigla deriva do **Technische Hogeschool Eindhoven**, na Holanda, onde foi implementado por E.W. Dijkstra (1968) e seus alunos.

O sistema THE era um sistema em **lote** (***batch***) simples para um computador holandês, o Electrologica X8, que tinha 32 K de palavras de 27 bits (os bits eram caros naquela época).

O sistema possuía **seis camadas**, conforme pode-se observar na Figura a seguir.

Sistemas de Camadas

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Figura. Estrutura do sistema operacional THE.

Sistemas de Camadas

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Figura. Estrutura do sistema operacional THE.

Sistemas de Camadas

A **camada 0 tratava** da **alocação** do **processador**, do **chaveamento** de **processos**, quando ocorriam as **interrupções** ou quando os **temporizadores** expiravam.

Acima da camada 0, o sistema era formado por processos sequências; cada um deles podia ser programado sem a preocupação com o fato de múltiplos processos estarem executando em um único processador.

Em outras palavras, a camada 0 fornecia a multiprogramação básica da CPU.

Sistemas de Camadas

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Figura. Estrutura do sistema operacional THE.

Sistemas de Camadas

A **camada 1 encarregava-se do gerenciamento de memória.**

Ela **alocava espaço** para **processos** na **memória principal** e em um **tambor magnético** de 512K palavras, que armazenava as **partes** de **processos** (**páginas**) para os quais não havia espaço na memória principal.

Acima da camada 1, os processos não precisavam prestar atenção se estavam na memória principal ou no tambor magnético; a camada 1 cuidava disso, assegurando que as páginas eram trazidas para a memória principal quando necessárias.

Sistemas de Camadas

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Figura. Estrutura do sistema operacional THE.

Sistemas de Camadas

A **camada 2 encarregava-se da comunicação entre cada processo e o console de operação.**

Acima dessa camada, cada processos tinha efetivamente seu próprio console de operação.

Sistemas de Camadas

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Figura. Estrutura do sistema operacional THE.

Sistemas de Camadas

A **camada 3 encarregava-se do gerenciamento dos dispositivos de E/S e armazenava temporariamente os fluxos de informação que iam para esses dispositivos ou que vinham deles.**

Acima da camada 3, cada processo podia lidar com dispositivos abstratos de E/S mais amigáveis, em vez de dispositivos reais cheios de peculiaridades.

Sistemas de Camadas

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Figura. Estrutura do sistema operacional THE.

Sistemas de Camadas

Na **camada 4 encontravam-se os programas de usuários.**

Eles não tinham de se preocupar com o gerenciamento de processo, de memória, console ou E/S.

Sistemas de Camadas

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

Figura. Estrutura do sistema operacional THE.

Sistemas de Camadas

O processo operador do sistema estava localizado na camada 5.

Sistemas de Camadas

Outra generalização do conceito de camadas estava presente o sistema **MULTICS**.

Em vez de camadas, o **MULTICS era descrito como uma série de anéis concêntricos, sendo que cada anel interno era mais privilegiado do que os externos.**

Quando um procedimento em um anel externo queria chamar um procedimento no anel interno, ele deveria fazer o equivalente a uma chamada ao sistema, isto é, uma instrução de desvio, TRAP, e a validade dos parâmetros era cuidadosamente verificada antes de permitir que a chamada continuasse.

Embora no MULTICS todo o sistema operacional fosse parte do espaço de endereçamento de cada processo do usuário, o hardware possibilitava ao sistema designar procedimentos individuais (na verdade, segmentos de memória) como protegidos contra leitura, escrita ou execução.

Sistemas de Camadas

O esquema de camadas do sistema **THE** era somente um suporte, pois todas as partes do sistema eram, ao final, agrupadas em um **único programa-objeto**.

Já no **MULTICS**, o mecanismo de anéis estava muito mais presente em tempo de execução e reforçado pelo hardware. Esse mecanismo de anéis era vantajoso porque podia facilmente ser estendido para estruturar subsistemas de usuário.

Máquinas Virtuais

As **versões iniciais** do **sistema operacional OS/360** eram estritamente em **lote** (*batch*). Mas muitos usuários da IBM 360 desejavam **compartilhar tempo**.

Então, vários grupos, de dentro e de fora da IBM, decidiram escrever **sistemas** de **tempo compartilhado** para o IBM 360.

O sistema de tempo compartilhado oficial da IBM, o TSS/360, foi lançado muito tarde e , quando finalmente se tornou mais popular, estava tão grande e lento que poucos clientes converteram suas aplicações.

Ele foi finalmente abandonado depois de já ter consumido 50 milhões de dólares em seu desenvolvimento.

Máquinas Virtuais

Um grupo do Centro Científico da IBM em Cambridge, Massachusetts, produziu um outro sistema, radicalmente diferente, que a IBM finalmente adotou e que é até hoje amplamente usado em seus computadores de grande porte remanescentes.

Esse sistema, originalmente denominado **CP/CMS** e depois renomeado para **VM/370**, foi baseado em uma observação perspicaz: um sistema de tempo compartilhado fornece **(1) multiprogramação**, e **(2) uma máquina estendida com uma interface mais conveniente do que a que o hardware exposto oferece**.

A essência do VM/370 é a **separação completa** dessas **funções**.

Máquinas Virtuais

O coração do sistema, conhecido como **monitor de máquina virtual**, é executado diretamente sobre o hardware e implemente multiprogramação, provendo assim não uma, mais várias máquinas virtuais para a próxima camada situada acima.

Contudo, ao contrário dos demais sistemas operacionais, essas **máquinas virtuais não são máquinas estendidas, com arquivos e outras características convenientes**.

Na verdade, são cópias exatas do hardware, inclusive com modos núcleo/usuário, E/S, interrupções e tudo o que uma máquina real tem.

Máquinas Virtuais

Como cada máquina virtual é uma cópia exata do hardware, cada uma delas pode executar qualquer sistema operacional capaz de ser executado diretamente sobre o hardware.

Diferentes máquinas virtuais podem - e isso ocorre com frequência - executar diferentes sistemas operacionais.

Em algumas dessas máquinas virtuais são executados um dos sistemas operacionais descendentes do OS/360 para processamento em lote (batch) ou de transações, enquanto se executa em outras um sistema operacional monousuário interativo, denominado **CMS** (*Conversational Monitor System* - sistema monitor conversacional) dedicado a usuários interativos em tempo compartilhado.

Máquinas Virtuais

Quando um programa CMS executa uma chamada ao sistema, ela é desviada para o sistema operacional que executa em sua própria máquina virtual, e não para o VM/370, como se estivesse executando sobre uma máquina real e não sobre uma máquina virtual.

O sistema operacional CMS então emite as instruções normais de hardware para E/S a fim de, por exemplo, ler seu disco virtual ou executar outro serviço qualquer pedido pela chamada.

Essas instruções de E/S são por sua vez desviadas pelo VM/370, que então as executa como parte de sua simulação do hardware real.

A partir da separação completa das funções de multiprogramação e da provisão de uma máquina estendida, pode-se ter partes muito mais simples, flexíveis e fáceis de serem mantidas.

Máquinas Virtuais

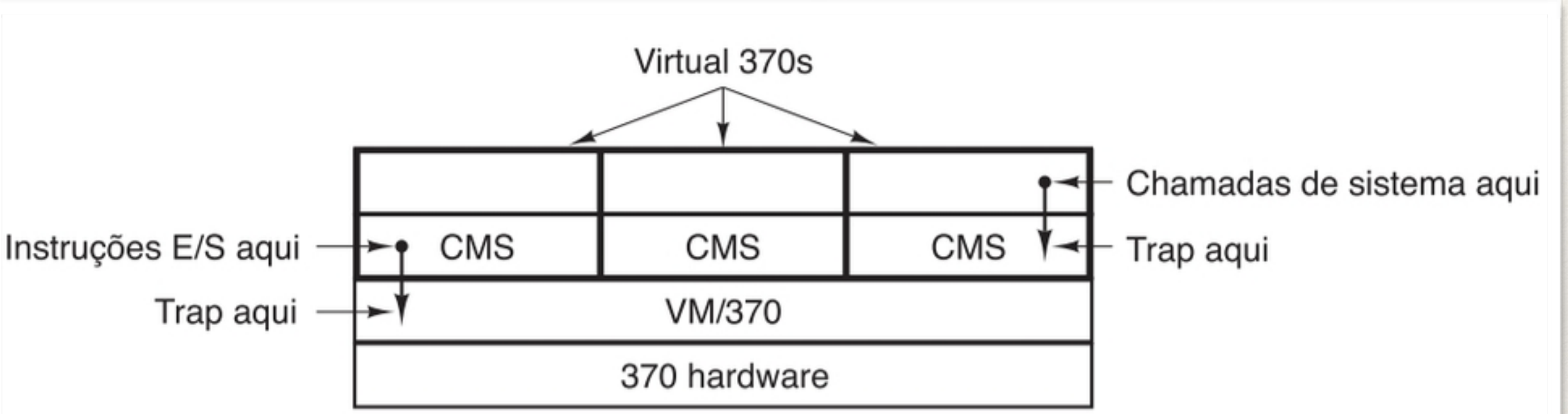


Figura. A estrutura do VM/370 com o CMS.

Máquinas Virtuais

A ideia de uma máquina virtual é bastante usada hoje em dia em um contexto diferente: a execução de velhos programas do MS-DOS.

Máquinas Virtuais

Outra área na qual máquinas virtuais são usadas, mas de maneira um pouco diferente, é a execução de **programas Java**.

Quando a Sun Microsystems inventou a linguagem de programação **Java**, inventou também uma máquina virtual (isto é, uma arquitetura de computador) denominada **JVM** (*Java Virtual Machine* - máquina virtual Java).

O compilador Java produz código JVM, que então é normalmente executado por um programa interpretador da JVM. A vantagem desse sistema é que o código JVM pode ser enviado pela internet a qualquer computador que tenha um interpretador JVM e ser executado lá.

Exonúcleos

Com o VM/370, cada processo de usuário obtém uma cópia exata do computador real.

No modo virtual 8086 do Pentium, cada processo de usuário obtém uma cópia exata de um computador diferente.

Os pesquisadores do MIT construíram um sistema que entrega a cada usuário um clone do computador real, mas com um subconjunto dos recursos.

Assim, uma máquina virtual pode obter os blocos 0 a 1023 do disco, um outra os blocos 1024 a 2047 e assim por diante.

Exonúcleos

Na **camada mais inferior**, executando em **modo núcleo**, há um **programa** denominado **exonúcleo**.

Sua tarefa é alocar recursos às máquinas virtuais e então verificar as tentativas de usá-las para assegurar-se de que nenhuma máquina esteja tentando usar recursos de outra.

Cada máquina virtual, em nível de usuário, pode executar seu próprio sistema operacional, como no VM/370 e na máquina virtual 8086 do Pentium, exceto que cada uma está restrita a usar somente os recursos que pediu e que foram alocados.

O modelo Cliente-Servidor

Uma ligeira variação da ideia de micronúcleo é distinguir entre duas classes de processos, os **servidores**, que prestam algum serviço, e os **clientes**, que usam esses serviços. Esse modelo é conhecido como **modelo cliente-servidor**.

Frequentemente a camada inferior é o micronúcleo, mas ele não é obrigatório. A essência é a presença de processos cliente e servidores.

O modelo Cliente-Servidor

A comunicação entre clientes e servidores é muitas vezes realizada pela troca de mensagens.

Para obter um serviço, um processo cliente constrói uma mensagem dizendo o que deseja e envia ao serviço apropriado.

Este faz o trabalho e envia a resposta de volta.

Se o cliente e o servidor forem executados na mesma máquina, certas otimizações são possíveis, mas, conceitualmente, estamos falando da troca de mensagens neste caso.

O modelo Cliente-Servidor

Uma generalização dessa ideia é executar clientes e servidores em computadores diferentes, conectados por uma rede local ou de grande área.

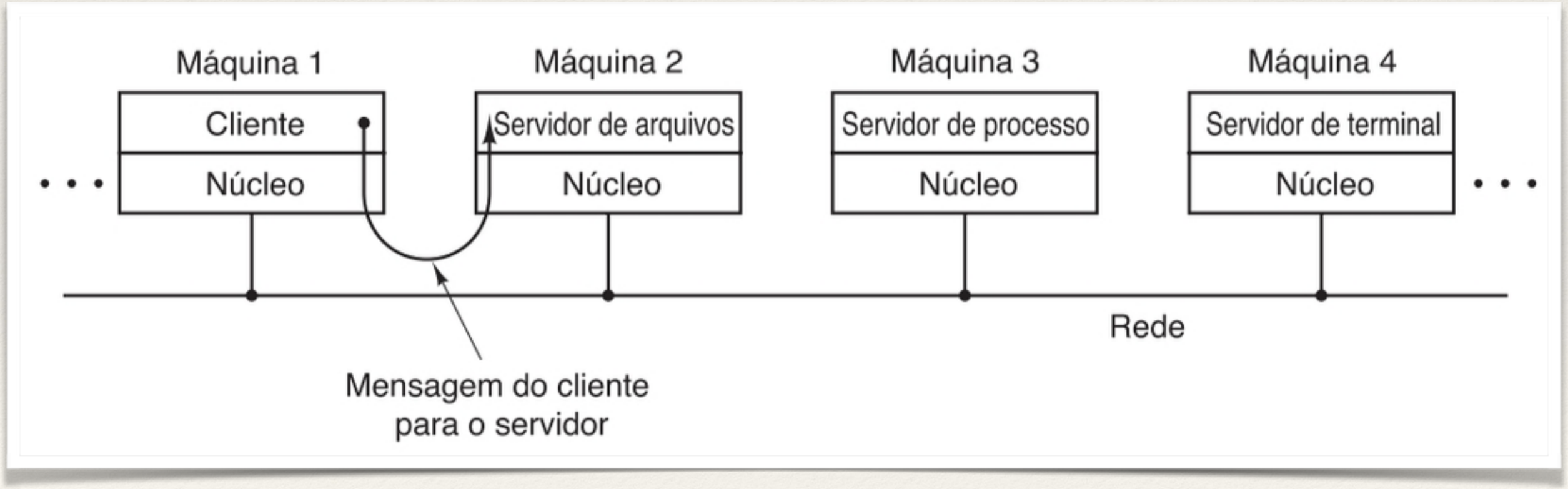


Figura. O modelo cliente-servidor em uma rede.

O modelo Cliente-Servidor

Uma vez que os clientes se comunicam com os servidores enviando mensagens, eles não precisam saber se as mensagens são entregues localmente em suas próprias máquinas ou se são enviadas através de uma rede a servidores em uma máquina remota.

No que se refere aos clientes, o mesmo ocorre em ambos os casos: solicitações são enviadas e respostas, devolvidas.

Dessa forma, o **modelo cliente-servidor é uma abstração que pode ser usada para uma única máquina ou para uma rede de máquinas**.

Próxima Aula

- Processos.

Dúvidas

- Conteúdo
  - Moodle
  - (http://wagnerglorenz.com.br/moodle/)
- Dúvidas
  - wagnerglorenz@gmail.com

Referências Bibliográficas

- TANENBAUM, A. Sistemas Operacionais Modernos. São Paulo: Prentice Hall, 2003.
- Sistemas Operacionais Modernos - 3ª edição Tanenbaum, Andrew S. http://ulbra.bv3.digitalpages.com.br/users/publications/9788576052371