Manual de SOD

Fundamentos de Sistemas Operativos e Sistemas Distribuídos

Manual de SOD

Índice do Manual

O objectivo deste manual é apresentar as principais características que são comuns nos Sistemas Operativos existentes, referindo as principais estratégias e abordagens que têm sido adoptadas ao longo da evolução dos sistemas informáticos, desde a origem - em que não eram usados sistemas operativos – até à actualidade, em que a aposta se centra em Sistemas Operativos Distribuídos que suportem a partilha robusta e eficiente de um conjunto de recursos que estão disponíveis em redes de computadores, e/ou a resolução de problemas computacionais complexos ou muito pesados através da distribuição da sua execução por diversos processadores.

No final serão referidas as características de alguns dos Sistemas Operativos que, na actualidade, constituem referência tecnológica e de mercado.

Capítulo 1
Introdução Histórica

No presente capítulo será feito um breve descrição do enquadramento histórico e tecnológico que conduziu à ao surgimento e à evolução de Sistemas Operativos e, em particular, de Sistemas Operativos Distribuídos.

Em finais dos anos 40, teve início a Era da Informática. No entanto, nos primeiros tempos, os computadores eram grandes e caros, pelo que o seu uso se restringia às empresas ou instituições que podiam pagar o seu alto preço.

As primeiras máquinas eram programadas directamente e não dispunham de sistema operativo. Os programadores tinham de ter um conhecimento e contacto profundo com o hardware e, caso a execução do programa falhasse, tinham de examinar o estado de um conjunto de registos e painéis de luzes indicadores do funcionamento do computador para determinar a causa da falha e corrigir o programa. Para executar uma nova versão do programa tinham de ser repetidos todos os procedimentos de reserva de tempo do sistema e de colocação em funcionamento dos compiladores, ligações, etc. característicos do processamento série (*serial processing*).

A Figura 1.1 apresenta uma fotografia do ENIAC (Electronic Numerical Integrator And Computer), que foi o primeiro computador a ser construído, em 1946.

A importância dos sistemas operativos manifestou-se nos anos 50, quando se tornou evidente que a operação dos computadores por meio de cartões perfurados, na primeira geração, e por meio do trabalho em lote, na segunda geração, podia ser melhorado significativamente, uma vez que havia um conjunto de operações sequenciais que tinham sempre de ser repetidas. Sendo esta uma das características contempladas na definição do que é um programa, as tarefas do operador foram programadas num algoritmo, o qual se passou a chamar "Sistema Operativo" e que foi evoluindo em funcionalidades e complexidade. Os primeiros sistemas

operativos a serem desenvolvidos foram o Fortran Monitor System (FMS) e o IBSYS.

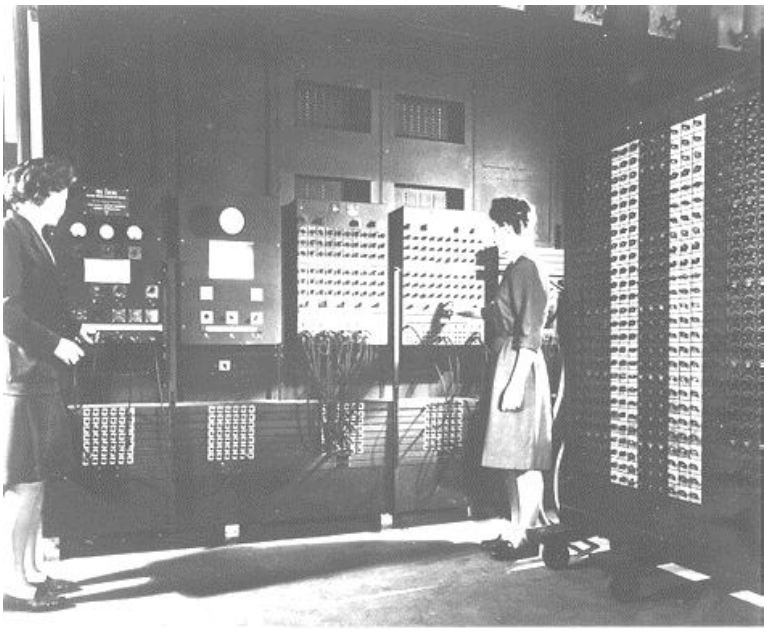


Figura 1.1 – O ENIAC, o primeiro computador

A primeira aproximação ao que, actualmente, são os sistemas operativos foi o desenvolvimento de um programa, designado monitor de controlo, que apenas se destinava a facilitar a utilização do sistema, oferecendo as funções estritamente necessárias para tornar possível a execução dos programas, sem que existissem quaisquer preocupações com a optimização do desempenho do sistema (Figura 1.2).

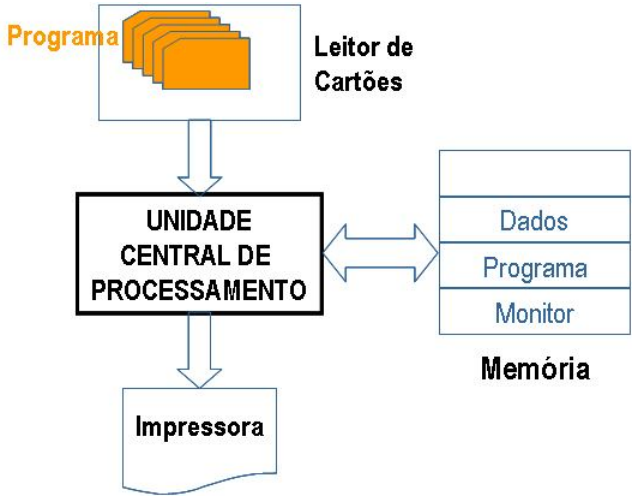


Figura 1.2 – Monitor de controlo residente em memória

O tratamento por lotes (*Batch*) tentou resolver a principal ineficiência dos sistemas, que resultava do escasso tempo de execução do processador, devido à demora imposta pelas operações de E/S (E/S) sobre periféricos mecânicos (Figura 1.3). As lentas operações de E/S passaram a ser realizadas por equipamentos auxiliares, o que permitiu que o processador central passasse a executar os trabalhos em fluxo contínuo. Esta evolução forçou o desenvolvimento de novas funcionalidades dos Sistemas Operativos. Por exemplo, uma necessidade que surgiu foi a do desenvolvimento de funções de gestão de estruturas de memória secundária que permitissem a manutenção dos programas e dos seus dados.

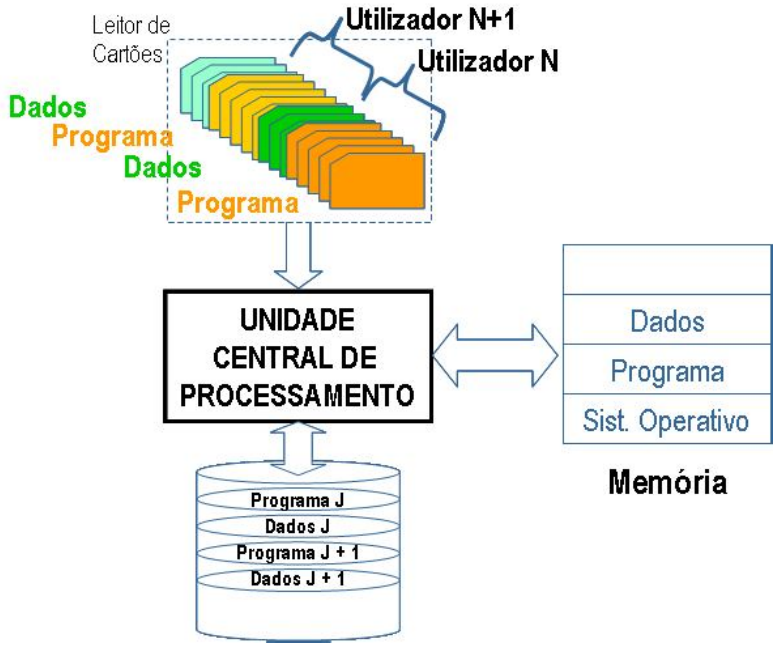


Figura 1.3 – O sistema operativo para suportar o tratamento por lotes

A evolução da arquitectura dos processadores e, em particular, o desenvolvimento das interrupções e dos canais de acesso directo à memória permitiram que as operações de E/S pudessem ser realizadas em simultâneo com a execução, na unidade central de processamento (CPU), dos programas dos utilizadores. A extrapolação deste mecanismo conduziu à possibilidade de multiplexar a execução de vários programas de forma a aproveitar todos os momentos de inactividade do processador. A multiprogramação tornou-se o mecanismo fundamental de gestão de sistema operativo, permitindo o desenvolvimento de sistemas de tratamento por lotes de grande eficácia.

A atribuição cíclica de tempo do processador a vários utilizadores permitiu que cada um tivesse a ilusão de dispor de uma máquina própria. Os sistemas de tempo partilhado tornaram-se predominantes quando a tecnologia dos terminais evoluiu de modo a tornar-se economicamente viável.

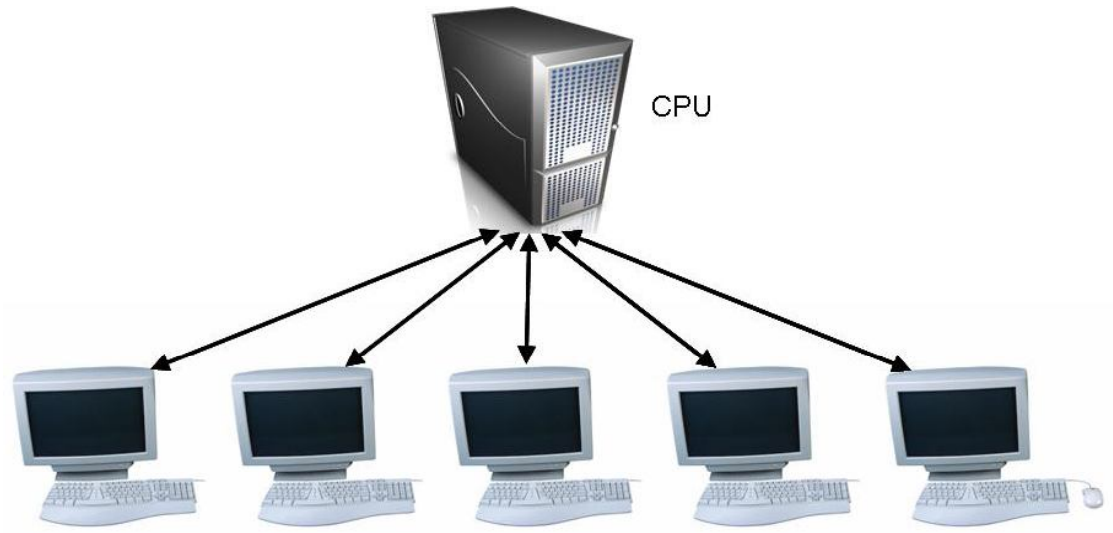


Figura 1.4 – Sistema Centralizado

Assim, a terceira geração de computadores corresponde ao surgimento do conceito da multiprogramação. Devido aos elevados custos de aquisição e sustentação dos computadores foi introduzido um esquema de trabalho que rentabilizava a CPU, maximizando o seu tempo de ocupação, e que envolvia, igualmente, a gestão da leitura dos dados e do código de diversos programas, a sua atribuição a blocos de memória livres, bem como as operações de escrita de resultados. Apesar desta evolução, durante a terceira geração os sistemas continuaram no essencial a ser sistemas de lote.

Na terceira geração de computadores surgem os primeiros sistemas operativos desenhados para administrar uma família de computadores, de que é exemplo o OS/360 da IBM. Este projecto foi tão inovador e ambicioso que enfrentou pela primeira vez uma série de problemas conflituantes. Até então os computadores tinham sido concebidos para dois tipos de aplicação genéricas: comerciais e o científicas. Durante o processo de criação de um sistema operativo único para computadores que podiam ser usados num tipo de aplicação, no outro ou em ambos, começaram a evidenciar-se os problemas e a complexidade das questões a enfrentar pelas equipas de análise, desenho e implantação de grandes sistemas.

Na quarta geração a electrónica progrediu no sentido da integração em grande escala, disponibilizando circuitos com milhares de transístores num centímetro quadrado de silício, o que permitiu evoluir para o emprego de computadores pessoais e de estações de trabalho. Nestas máquinas tornaram-se populares os sistemas operativos MS-DOS e UNIX. Generalizaram-se os clones dos computadores pessoais e surgiu uma grande quantidade de pequenas empresas que os montavam por todo o mundo. O conceito de interfaces amigáveis afirmou-se como uma forma de atrair o público em geral para o uso dos computadores como ferramentas para utilização nas funções do dia-a-dia.

Os grandes sistemas centralizados foram lentamente cedendo lugar a sistemas muito mais descentralizados, formados por vários computadores ou por sistemas multiprocessador. Esta evolução criou novas necessidades de interligação dos equipamentos, e conduziu ao desenvolvimento das redes de área local (LAN), como a Ethernet ou o Token Ring (Figura 1.5). Assim, em meados dos anos 80, com a generalização do uso das redes de computadores, revelou-se a necessidade de sistemas operativos em rede e de sistemas operativos distribuídos.

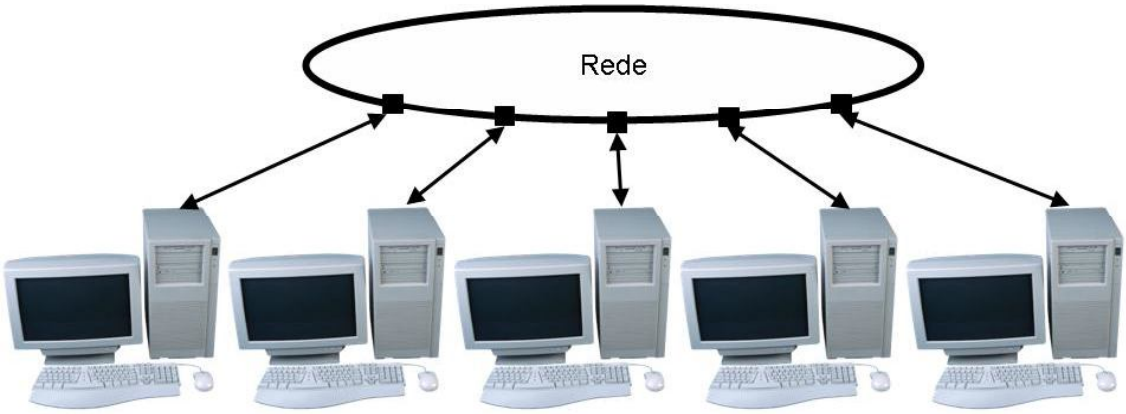


Figura 1.5 – Arquitectura em Rede

Nos anos 90 afirmou-se o paradigma da programação orientada a objectos, e com ele a gestão de objectos a partir dos sistemas operativos. A Internet torna-se acessível a todo o tipo de instituições e começam a surgir soluções para interligar recursos residentes em computadores com sistemas operativos diferentes. As aplicações são desenvolvidas de modo a serem executadas numa plataforma específica e os resultados poderem ser apresentados nos terminais de outra diferente (por exemplo, executar uma simulação numa

máquina com UNIX e ver os resultados noutra com DOS). Os níveis de interacção vão-se tornando cada vez mais profundos.

Actualmente as redes locais de computadores proporcionam altos débitos com elevada fiabilidade e a evolução da micro electrónica, reduzindo os custos do hardware, permite novos estilos de utilização dos computadores. A distribuição do sistema operativo introduz um conjunto de problemas que começam por ser resolvidos nos sistemas mais avançados.

Apesar dos sistemas de rede actuais solucionarem uma parte significativa das necessidades de comunicação entre computadores, ainda existem algumas limitações importantes, que não permitem dar resposta a uma grande quantidade de problemas. Por isso ainda persiste a necessidade de se desenvolverem sistemas distribuídos que substituam os sistemas de rede ou sistemas multiprocessador existentes.

Na entrada do século XXI, a Internet é a rede de maior tamanho e a mais usada, e mantém um impressionante ritmo de crescimento. Além disso, a Internet é a base de muitos novos projectos de sistemas distribuídos.

Verifique os seus conhecimentos...

1. O que é que levou à necessidade da introdução de sistemas operativos nos computadores?

2. Indique quais foram as evoluções que permitiram que as operações de entrada/saída pudessem ser realizadas em simultâneo com a execução, na unidade central de processamento (CPU), dos programas dos utilizadores?

3. Explique como é que a introdução de sistemas de tempo partilhado contribuiu para que os computadores passassem a poder ser usados por mais de um utilizador.

4. O que é que diferencia um sistema centralizado de um sistema distribuído?

5. Indique se as seguintes frases são verdadeiras ou falsas

	V	F
a. Para funcionar um computador tem obrigatoriamente de ter um sistema operativo.		
b. Todos os sistemas operativos são interoperáveis.		
c. Todos os sistemas operativos diferentes são incompatíveis, não podendo os computadores que os usam ser ligados.		
d. O desenvolvimento dos sistemas operativos distribuídos visa ultrapassar as limitações dos sistemas operativos convencionais.		

Capítulo 2
Tipos de Sistemas Operativos

No presente capítulo serão descritas as principais características dos sistemas operativos, considerando diversos pontos de vista pelos quais se podem abordar os sistemas.

Ao longo deste capítulo serão descritas as principais características dos sistemas operativos. Para o efeito serão utilizadas três tipos de classificações:

- considerando a estrutura do SO (visão interna);
- considerando os serviços que os SO disponibilizam; e
- considerando a forma como os SO disponibilizam os seus serviços (visão externa).

2.1 Classificação dos SO considerando a estrutura

Os sistemas operativos são programas residentes entre o hardware e as aplicações dos utilizadores.

Figura 2.1 – Posicionamento do sistema operativo no sistema

Existem duas razões principais para a utilização dos sistemas operativos:

- A criação de uma máquina virtual, de fácil utilização, que oculte a complexidade do controlo do hardware e dos periféricos; e
- a melhoria do desempenho do sistema, de forma a rentabilizar o investimento no equipamento.

Para dar resposta a estes objectivos, no desenvolvimento dos sistemas operativos, devem observar-se os aspectos relativos a dois tipos de requisitos:

Requisitos de usabilidade facilidade de utilização e aprendizagem, segurança, rapidez e adequação ao seu campo de aplicação; e

Requisitos do software manutenção, forma de operação, restrições de uso, eficiência, tolerância a erros e flexibilidade.

De seguida descrevem-se os principais tipos de estruturas que os sistemas operativos apresentam.

2.1.1 Estruturas monolítica e de micronúcleo

A estrutura monolítica é a estrutura dos primeiros sistemas operativos. É constituída, fundamentalmente, por um único programa composto de um conjunto de rotinas entrelaçadas de tal forma que cada uma possa chamar qualquer outra (Figura 2.2). As características fundamentais deste tipo de estrutura são:

- Construção do programa final efectuada com base em módulos compilados separadamente que se unem através de um programa ligador;
- Boa definição de parâmetros de enlace entre as diferentes rotinas existentes, que pode provocar muito acoplamento;
- Carecem de protecções e privilégios para aceder a rotinas que fazem a gestão de diferentes aspectos dos recursos do computador (p. ex., memória ou discos).

Em geral estes SO são desenvolvidos à medida, pelo que são eficientes e rápidos na sua execução e gestão. Em compensação

falta-lhes a flexibilidade para poderem suportar diferentes ambientes de trabalho ou tipos de aplicações.

A arquitectura de micronúcleo baseia-se numa programação altamente modular, que tem um tamanho muito menor que o núcleo monolítico. O refinamento e o controlo de erros é mais rápido e simples, e a actualização dos serviços é mais simples e ágil, já que neste caso apenas se torna necessária a recompilação do serviço e não da totalidade do núcleo. No entanto, o rendimento é afectado negativamente.

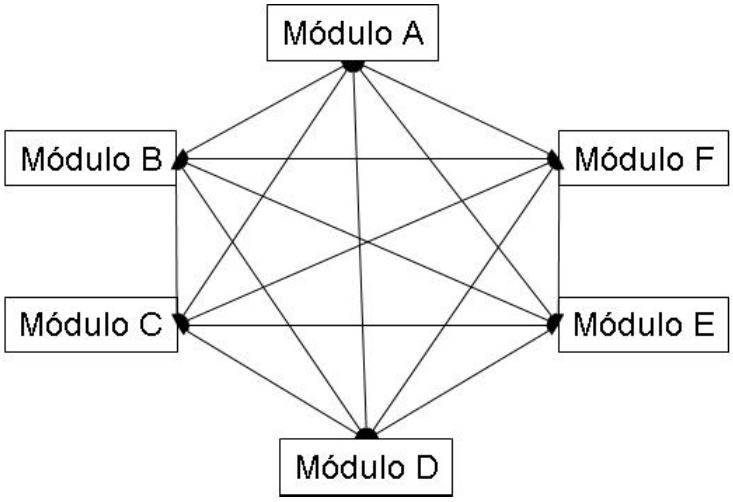


Figura 2.2 – Estrutura micronúcleo

A maioria dos sistemas operativos distribuídos em desenvolvimento na actualidade tendem a adoptar um desenho de micronúcleo. Estes núcleos tendem a conter menos erros e a ser mais fáceis de implementar e de corrigir. O sistema perde ligeiramente em rendimento, mas esta desvantagem é compensada pelo grande aumento de flexibilidade.

2.1.2 Estrutura hierárquica

À medida que foram crescendo as necessidades dos utilizadores e se aperfeiçoaram os sistemas, tornou-se necessária uma maior organização do software do sistema operativo, pelo que as componentes do sistema passaram a contemplar sub-componentes organizadas em níveis.

Deste modo o sistema operativo está dividido em pequenas partes. Cada uma destas partes encontra-se perfeitamente definida e tem um claro interface com os restantes elementos.

Do ponto de vista conceptual, em geral, pode considerar-se a arquitectura dos sistemas operativos como uma estrutura hierárquica "multicamada", onde cada camadas implementa uma parte das funções essenciais requeridas a um sistema operativo. Uma das estruturas hierárquicas comummente aceite é a apresentada na Figura 2.3.

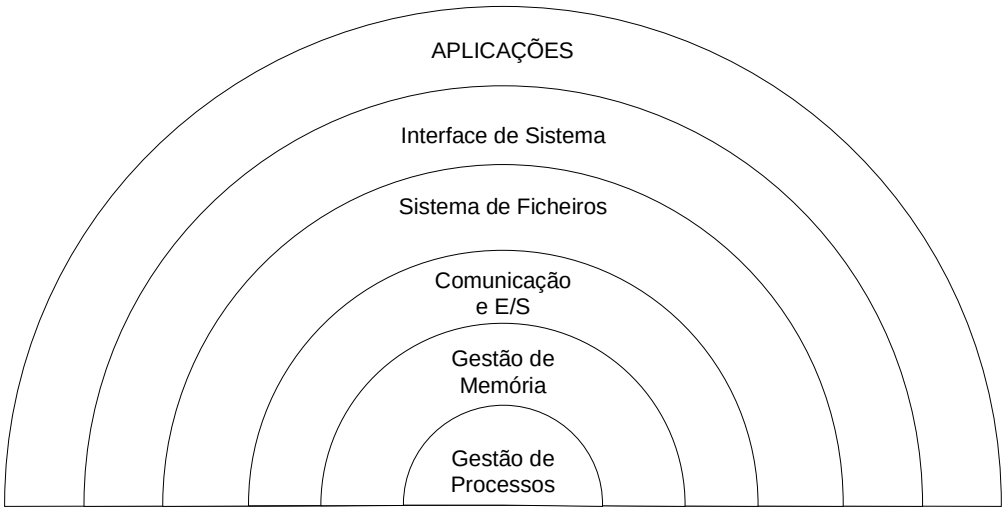


Figura 2.3 – Visão multicamada dos Sistemas Operativos

Considerando a abordagem hierárquica ilustrada, os sistemas operativos compreendem as seguintes camadas:

- **Gestão de Processos**: camada responsável por multiplexar o tempo de execução do processador entre os processos activos que, do ponto de vista do utilizador, se comportam como máquinas virtuais executando um programa. Esta camada encarrega-se, igualmente, do tratamento de interrupções, do despacho dos processos e da respectiva sincronização;

- **Gestão de Memória**: camada responsável pela gestão da memória física e do espaço de endereçamento virtual onde decorre a execução dos processos;

- **Comunicação e Entradas/Saídas**: camada responsável pelos mecanismos necessários à troca de informação entre processos. A interacção das E/S com os periféricos é um tipo particular de comunicação que costuma ser tratado nesta camada;

- **Sistemas de Ficheiros**: camada responsável pela gestão de memória secundária e pela organização lógica que suporta o

conjunto de informação persistente que é mantida no sistema; e

- **Interface de Sistema**: camada que lida com dois tipos de interface. O primeiro inclui o conjunto das funções sistema, que permitem interagir com as restantes camadas do SO. O segundo constituído por um interpretador de comandos que materializa o interface usado pelo utilizador para controlar o sistema.

Cada camada tem uma "abertura", conhecida como porta (trap), por onde são efectuadas as chamadas às camadas inferiores. Desta forma, as zonas mais internas do sistema operativo ou núcleo do sistema estão mais protegidas de acessos indesejados provenientes das camadas mais externas. As camadas mais internas são, portanto, mais privilegiadas que as externas.

A maioria dos sistemas operativos actuais baseiam-se nesta estrutura.

2.1.3 Máquina Virtual

Trata-se de um tipo de sistema operativo que partilha o hardware entre diversos utilizadores, mas que apresenta uma interface para cada processo. Nesta abordagem cada programa vê uma máquina privada que parece idêntica à máquina real subjacente. Estes sistemas operativos separam dois conceitos que são tratados conjuntamente no resto de sistemas: a multiprogramação e a máquina estendida. O objectivo dos sistemas operativos de máquina virtual é o de integrar diferentes sistemas operativos dando a sensação aos utilizadores de estarem a utilizar várias máquinas diferentes.

O núcleo destes sistemas operativos denomina-se monitor virtual e tem como missão levar a cabo a multiprogramação, apresentando para os níveis superiores tantas máquinas virtuais quantas as solicitadas. Estas máquinas virtuais não são máquinas estendidas, antes uma réplica da máquina real. Deste modo cada uma delas pode executar um sistema operativo diferente, que será o que a máquina estendida oferece ao utilizador (Figura 2.4).

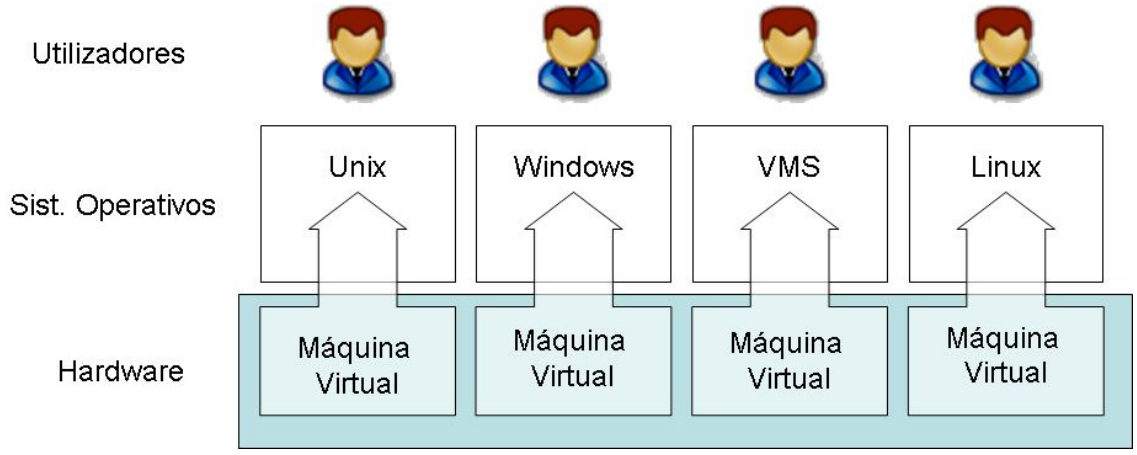


Figura 2.4 – Máquina Virtual

2.1.4 Cliente-Servidor

Um tipo mais recente de sistemas operativos é o denominado Cliente-Servidor, que pode ser executado na maioria dos computadores, sejam grandes ou pequenos.

Este sistema pode ser utilizado em todas as classes de aplicações. Assim sendo, é um SO de aplicação geral e pode realizar as mesmas actividades que os sistemas operativos convencionais.

O núcleo tem como missão estabelecer a comunicação entre os clientes e os servidores. Os processos podem ser tanto servidores como clientes. Por exemplo, um programa de aplicação normal é um cliente que chama o servidor correspondente para aceder a um ficheiro ou realizar uma operação de E/S num dispositivo concreto. Por sua vez, um processo cliente pode actuar como servidor para outro. Este paradigma garante uma grande flexibilidade relativamente aos serviços que podem ser oferecidos pelo sistema, uma vez que o núcleo se limita a disponibilizar funções muito básicas de memória, entrada/saída, ficheiros e processos, cabendo aos servidores dar resposta à maioria das necessidades que os utilizadores finais ou os programadores possam ter. Estes servidores devem ter mecanismos de segurança e protecção que, por sua vez, são filtrados pelo núcleo que controla o hardware.

2.2 Classificação dos SO considerando os Serviços

Esta classificação é a mais comummente usada e conhecida, e assume o ponto de vista do utilizador final. Compreende-se facilmente se se considerar o esquema apresentado na Figura 2.5.

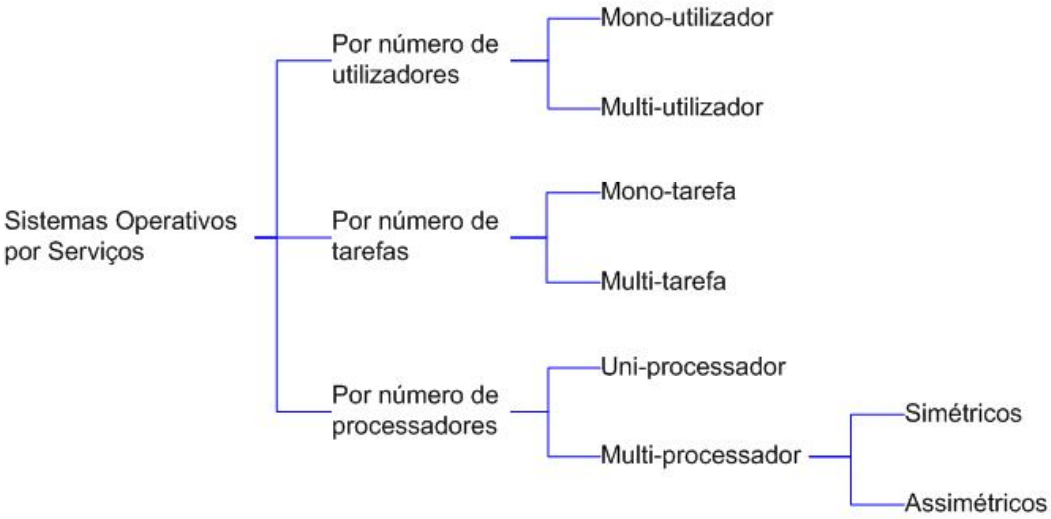


Figura 2.5 – Classificação dos Sistemas Operativos de acordo com os serviços

Usando esta abordagem os SO podem ser classificados de diferentes modos. Uma baseia-se no número de utilizadores que podem operar um sistema, outra no número de tarefas que um sistema pode realizar e, uma última, no número de processadores envolvidos na execução dos programas.

2.2.1 Número de utilizadores

Considerando o número de utilizadores que podem operar um sistema, os SO podem ser mono-utilizador ou multi-utilizador (Figura 2.6). Nesta classificação considera-se o seguinte:

Sistema Operativo Mono-utilizador: são aqueles que suportam um único utilizador de cada vez, sem importar o número de processadores que tenha o computador ou o número de processos ou tarefas que o utilizador possa executar num mesmo instante de tempo. Tipicamente os computadores pessoais enquadram-se nesta categoria.

Sistema Operativo Multi-utilizador: são capazes de prestar serviço a mais de um utilizador de cada vez, seja por meio de vários terminais ligados ao computador ou por meio de sessões remotas numa rede de comunicações. Nesta classificação não importa o número de processadores na máquina nem o número de processos que cada utilizador pode executar simultaneamente.

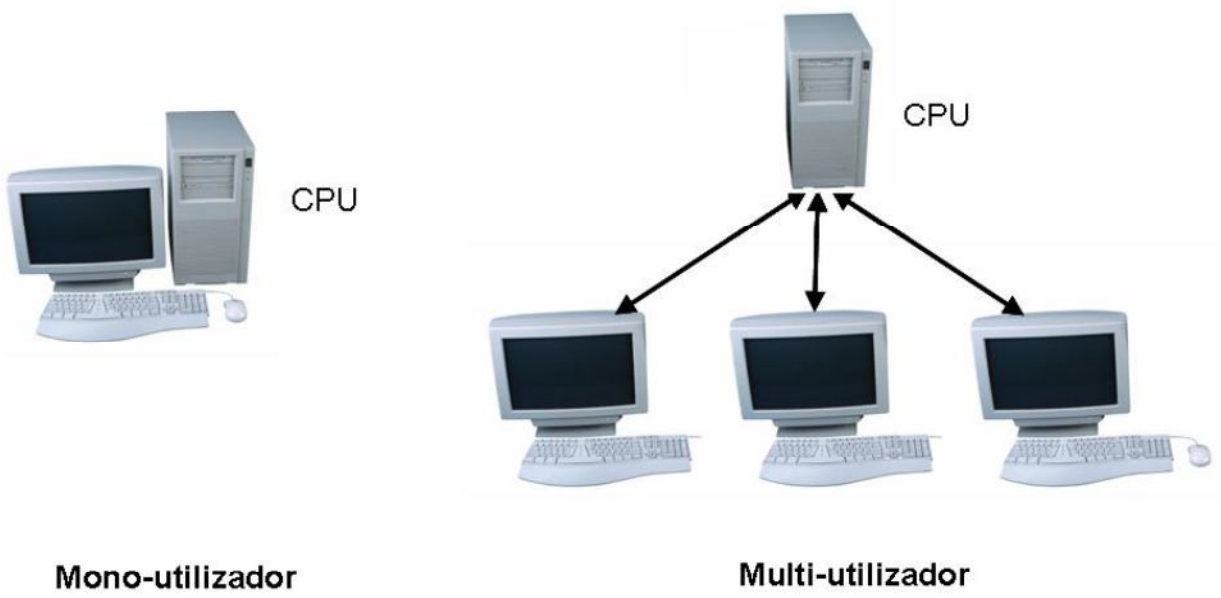


Figura 2.6 - Número de utilizadores

2.2.2 Número de tarefas

Considerando o número de tarefas que podem ser realizadas por um sistema, os SO podem ser mono-tarefa ou multi-tarefa (Figura 2.7). Nesta classificação considera-se o seguinte:

Sistema Operativo Mono-tarefa: são aqueles que permitem somente uma tarefa de cada vez por utilizador. Podem existir sistemas multi-utilizador e mono-tarefa, que servem vários utilizadores ao mesmo tempo, mas cada um destes executando uma única tarefa de cada vez.

Sistema Operativo Multi-tarefa: é aquele que oferece ao utilizador a realização de várias actividades ao mesmo tempo. Por exemplo, pode editar o código fonte de um programa e ao mesmo tempo estar a compilar outro programa e estar a receber correio electrónico em background. Estes SO normalmente têm interfaces gráficas que permitem o uso de menus e de rato que permitem que o utilizador comute rapidamente de tarefas, melhorando a sua produtividade.

2.2.3 Número de processadores

Considerando o número de processadores existentes num sistema, os SO podem ser uni-processador ou multi-processador (Figura 2.8). Nesta classificação considera-se o seguinte:

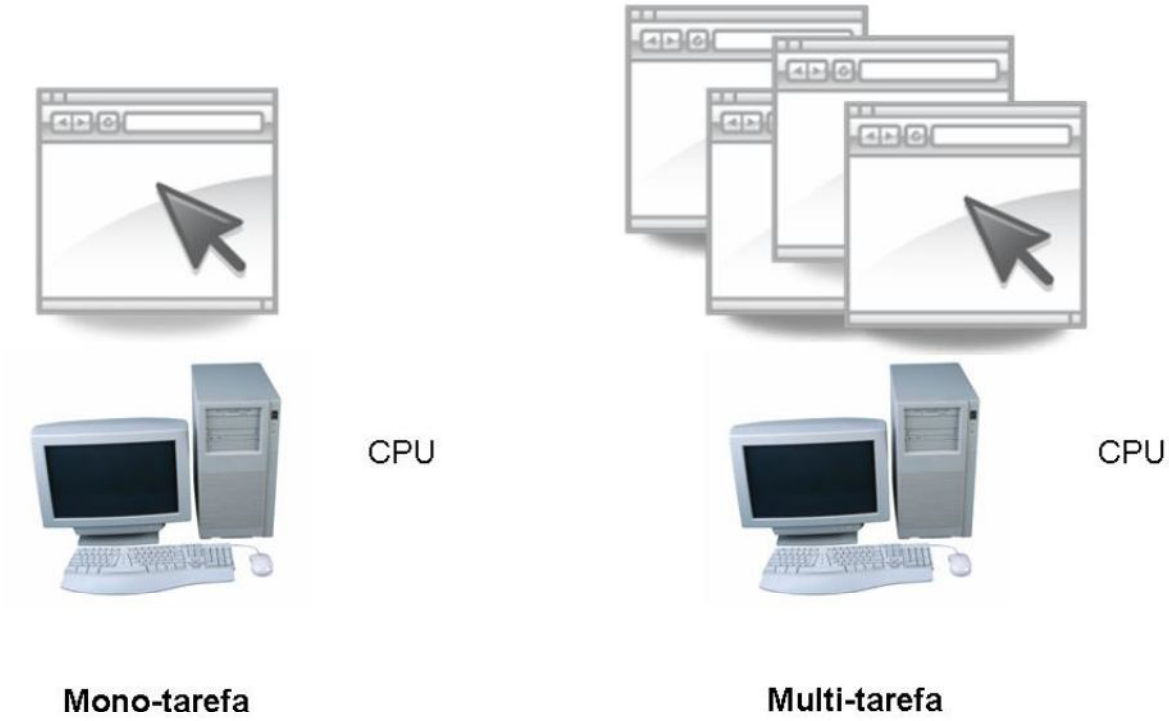


Figura 2.7 - Número de tarefas

Sistema Operativo Uni-processador: é aquele que é capaz de gerir somente um processador do computador, sendo inútil a existência mais de uma CPU no computador. O exemplo mais típico deste tipo de SO é o DOS e MacOS.

Sistema Operativo Multi-processador: é aquele que é capaz de gerir mais do que um processador do sistema, usando-os a todos para distribuir os pedidos de trabalho. Geralmente estes sistemas trabalham de duas formas: simétrica ou assimetricamente. Quando trabalha de forma assimétrica, o sistema operativo selecciona um dos processadores que desempenhará o papel de processador Mestre e servirá como pivot para distribuir as entradas pelos restantes processadores, que recebem o nome de Escravos. Quando se trabalha de forma simétrica, os processos ou partes destes (*threads*) são enviados indistintamente a qualquer um dos processadores disponíveis. Teoricamente, com este esquema obtém-se um melhor distribuição e equilíbrio na atribuição de trabalho ao sistema.

Um **thread** é a parte activa em memória e em execução de um processo, o qual pode consistir em: uma área de memória, um contexto composto por um conjunto de

registos com valores específicos e a pilha. Um aspecto importante a considerar nestes sistemas é a forma de criar aplicações para aproveitar os vários processadores. Existem aplicações que foram feitas para executar em sistemas monoprocesso e que não beneficiam desta vantagem, a menos que o próprio sistema operativo ou o compilador detecte secções de código que sejam paralelizáveis, as quais serão executadas ao mesmo tempo em processadores diferentes. Por outro lado, o programador pode modificar os seus algoritmos e definir a forma como o programa irá aproveitar os múltiplos processadores. No entanto, esta última opção geralmente é onerosa em horas homem e muito complicada, obrigando o programador a despender tanto ou mais tempo na paralelização como para elaborar o algoritmo inicial.

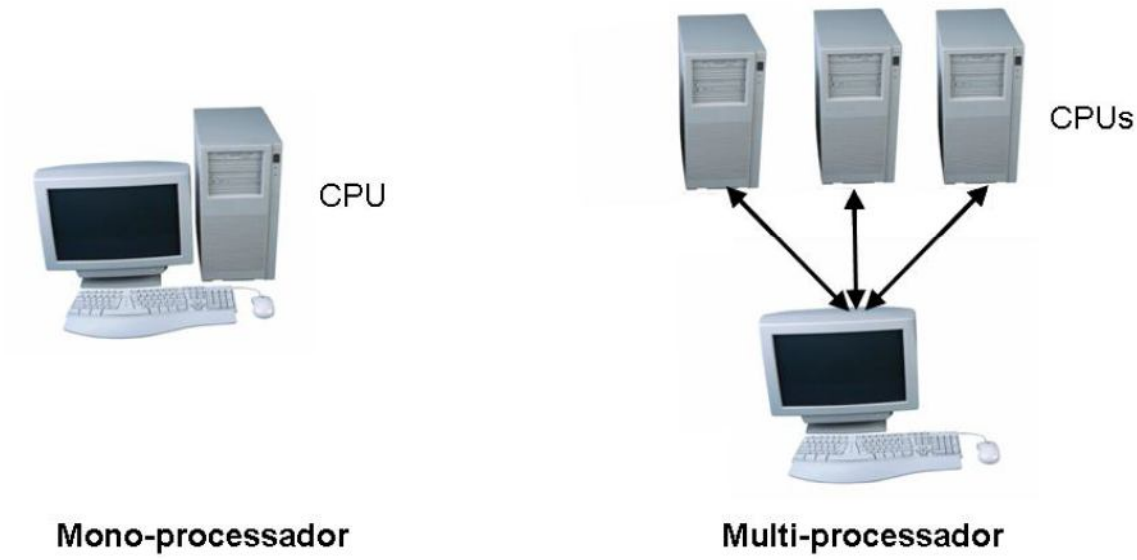


Figura 2.8 - Número de processadores

2.3. Classificação de acordo com a forma de disponibilizar os Serviços

Esta classificação também se refere a uma visão externa, e neste caso diz respeito à forma como o utilizador acede aos serviços. Nesta classificação consideram-se dois tipos principais de SO: sistemas operativos de rede e sistemas operativos distribuídos.

2.3.1 Sistemas Operativos de Rede

Os sistemas operativos de rede são aqueles que têm a capacidade de interagir com os sistemas operativos existentes noutros computadores recorrendo a um meio de transmissão por forma a transferir informação e ficheiros, executar comandos remotos e realizar um grande número de outras actividades. O aspecto crucial destes sistemas é que o utilizador tem de saber a sintaxe de um conjunto de comandos ou chamadas de sistema para executar estas operações, para além de ter de conhecer a localização dos recursos que pretende aceder. Considere-se o exemplo de um utilizador que, usando o sistema operativo UNIX, se situa no computador Master e necessita de copiar o ficheiro "exemplo.pas" que se localiza no directório "/software/código" do computador "lisboa". Este utilizador pode copiá-lo através da rede, utilizando os comandos seguintes: master%master%rcp lisboa:/software/codigo/exemplo.pas. master% Neste caso, o comando rcp que executa a "*remote copy*" traz o ficheiro indicado do computador "Lisboa" e coloca-o no directório onde o comando foi executado. Apesar de poder haver alguma complexidade no procedimento, o que é importante salientar é que o utilizador pode aceder e partilhar muitos recursos, desde que conheça a sua localização.

2.3.2 Sistemas Operativos Distribuídos

Os sistemas operativos distribuídos integram os serviços disponibilizados pelo SO de rede, mas conseguem gerir os recursos (impressoras, unidades de entrada/saída, memória, processos, unidades centrais de processo) como uma única máquina virtual a que o utilizador acede de forma transparente. Agora o utilizador já não necessita de conhecer a localização dos recursos, usando-os simplesmente como se todos eles fossem locais ao seu lugar de trabalho habitual.

Este é o objectivo que, de um ponto de vista teórico se desejaria alcançar com um sistema operativo distribuído. Na realidade ainda não se conseguiu criar um SO global. A dificuldade reside na complexidade que é inerente a distribuir os processos pelas várias unidades de processamento disponíveis, reintegrar sub-resultados, resolver problemas de concorrência e paralelismo, recuperar de falhas de alguns recursos distribuídos e consolidar a protecção e a segurança entre os diferentes componentes do sistema e os utilizadores.

Os avanços tecnológicos nas redes de área local e a criação de microprocessadores de 32 e 64 bits permitiram que computadores

relativamente baratos dispusessem, de uma forma autónoma, de suficientes capacidades para desafiar em certo grau as *mainframes*, e simultaneamente criou-se a possibilidade de os interligar, surgindo a oportunidade de partir processos de cálculo muito pesados em blocos mais pequenos e de os distribuir por vários microprocessadores para depois reunir os resultados parciais, criando assim uma máquina virtual na rede com capacidade de cálculo semelhante a uma *mainframe*.

O sistema integrador dos microprocessadores que permite ver as várias memórias, processadores, e todos os demais recursos como uma única entidade de forma transparente chama-se sistema operativo distribuído.

As razões para criar ou adoptar sistemas distribuídos são principalmente de duas naturezas distintas:

- por necessidade (porque os problemas a resolver são inerentemente distribuídos); ou
- para aumentar a fiabilidade e a disponibilidade em recursos.

No primeiro caso temos, por exemplo, o controlo dos terminais Multibanco existentes no país. Neste caso não é possível nem eficiente manter um controlo centralizado. Além disso não existe capacidade computacional e de E/S para prestar o serviço correspondente a milhões de operações por minuto.

No segundo caso, pode considerar-se um exemplo em que uma grande empresa tem vários grupos de trabalho, cada um com a necessidade de armazenar grandes quantidades de informação em disco rígido com uma alta fiabilidade e disponibilidade. A solução pode ser atribuir a cada grupo de trabalho uma partição de disco rígido em servidores diferentes, de forma a que, em caso de falha de um dos servidores, não deixa de se prestar o serviço a todos, antes somente a alguns. Além disso, caso a rede disponha de discos espelho (*mirror*), se o servidor cair então um servidor espelho entra em funcionamento e o utilizador nem se apercebe da ocorrência da falhas, isto é, continua a aceder aos recursos de forma transparente, como se ilustra na Figura 2.9.

2.3.2.1 Vantagens dos Sistemas Distribuídos

Em geral, os sistemas distribuídos (e não somente os sistemas operativos) exibem algumas vantagens sobre os sistemas centralizados que descrevem-se a seguir.

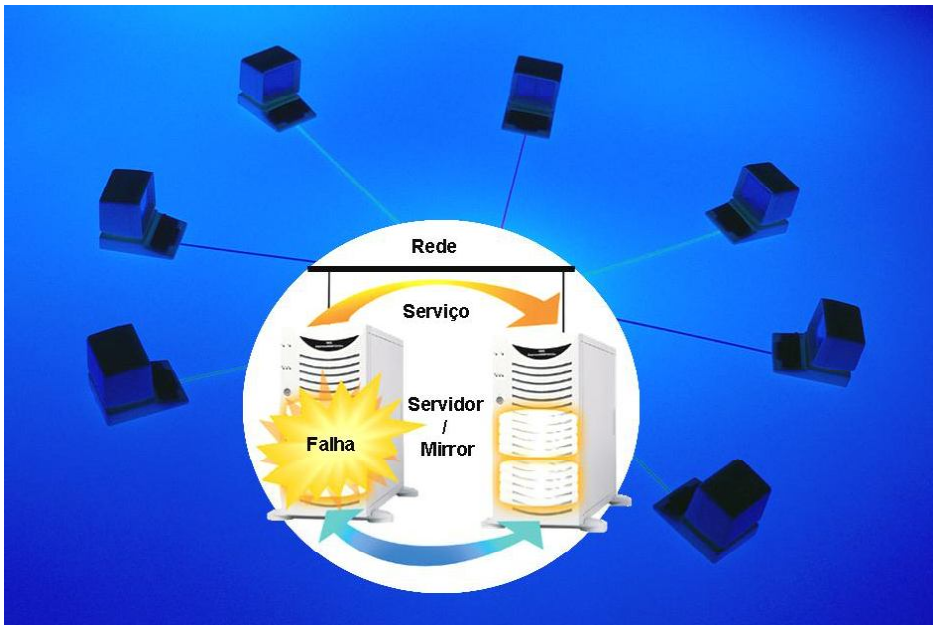


Figura 2.9 – Adopção de sistemas distribuídos para aumentar a fiabilidade do sistema computacional

- **Economia**: A relação preço/desempenho da soma da capacidade dos processadores separados contra a capacidade de um único processador centralizado é favorável à solução distribuída;
- **Velocidade**: Relacionado com o ponto anterior, a velocidade somada é muito superior;
- **Fiabilidade**: Se uma única máquina falha, o sistema total continua a funcionar;
- **Crescimento**: A capacidade total do sistema pode ser incrementada adicionando pequenos sistemas, o que é muito mais difícil num sistema centralizado;
- **Distribuição**: Algumas aplicações, dadas as suas características, requerem a existência de distribuição física.

Por outro lado, os sistemas distribuídos também exibem algumas vantagens sobre os sistemas isolados. Estas vantagens são:

- **Partilha de dados**: Um sistema distribuído permite partilhar dados mais facilmente que os sistemas isolados, que para conseguirem o mesmo efeito teriam de replicar os dados em cada nó;

- **Partilha de dispositivos**: Um sistema distribuído permite aceder a dispositivos a partir de qualquer nó de forma transparente, o que é impossível com os sistemas isolados. O sistema distribuído consegue um efeito de sinergia.

- **Comunicações**: A comunicação entre utilizadores é exequível nos sistemas distribuídos, nos sistemas isolados não.

- **Flexibilidade**: A distribuição dos pedidos de trabalho é exequível no sistema distribuídos, contribuindo para incrementar a capacidade computacional disponível para os utilizadores.

2.3.2.2 Desvantagens dos Sistemas Distribuídos

Se apresentam grandes vantagens, os sistemas distribuídos não são isentos de desvantagens. Algumas delas são tão severas que têm travado a produção comercial destes sistemas operativos. O problema mais importante na criação de sistemas distribuídos é o software: os problemas de partilha de dados e de recursos são tão complexos que os mecanismos para o resolverem geram tanta sobrecarga que tornam o sistema ineficiente. A verificação de quem tem ou não acesso a um recurso, a aplicação de mecanismos de protecção ou o registo de permissões são exemplos de outras actividades que consomem demasiados recursos.

Outros problemas dos sistemas operativos distribuídos dizem respeito à concorrência e ao paralelismo. Tradicionalmente, as aplicações são criadas para computadores que executam sequencialmente. A identificação das secções de código que são "paralelizáveis" é uma tarefa difícil mas necessária para dividir um processo grande em sub-processos e enviá-los a diferentes unidades de processamento e conseguir a sua distribuição. Com a concorrência devem implantar-se mecanismos para evitar as condições de pedidos simultâneos, de bloqueios indefinidos (resultantes, por exemplo, de uma situação em que, enquanto se ocupa um recurso, se entrar em espera por um outro recurso que está ocupado por um processo que se encontra à espera do recurso em utilização pelo primeiro processo (espera circular)) e de interblocagem. Estes problemas já existem nos sistemas operativos multi-utilizador e multi-tarefa, mas o seu tratamento nos sistemas distribuídos é ainda mais complexo, pelo que necessita de algoritmos mais complexos, sendo a consequência esperada um aumento da carga imposta pelo SO ao sistema.

Verifique os seus conhecimentos...

1. Quais as diferenças e as vantagens/inconvenientes dos sistemas operativos com uma estrutura monolítica relativamente aos que apresentam uma estrutura de micronúcleo?

2. Do ponto de vista conceptual, a implementação do sistema operativo pode ser vista como uma estrutura hierárquica constituída por camadas. Quais são essas camadas?

3. O que entende por de sistema operativo de tipo máquina virtual?

4. Explique o que entende por sistemas multi-utilizador, multi-tarefa e multi-processador? O que é que podem ter em comum?

5. O que é um thread?

6. O que entende por sistema operativo distribuído?

7. Em que medida é que a adopção de um sistema operativo distribuído pode contribuir para melhorar a fiabilidade de um sistema? Dê exemplo de soluções que permitem assegurar uma elevada fiabilidade de um sistema distribuído?

8. Quais as vantagens e as desvantagens que os sistemas operativos distribuídos oferecem relativamente aos sistemas operativos de rede?

Capítulo 3
Gestão de Processos

Uma das funções mais importantes de um sistema operativo é a administração dos processos e tarefas do sistema computacional. Neste capítulo serão apresentados dois temas centrais da gestão de processos: o planeamento do processador e os problemas de concorrência.

3.1 Processos

Os sistemas operativos destinados a suportar a "execução simultânea" de diversos programas baseiam-se no conceito de processo. Um processo é um bloco de código com controlo independente que executa um programa ou uma parte de um programa.

A expressão execução simultânea foi colocada entre aspas, porque o paralelismo não é real. A maioria dos computadores apenas dispõe de um processador, sendo os mecanismos de gestão do sistema operativo que se encarregam de ocultar ao utilizador os detalhes da multiplexagem do processador. Do ponto de vista dos utilizadores os processos são executados concorrentemente.

A Figura 3.1 ilustra a multiplexagem do tempo de execução da CPU pelos diversos processos que se encontram em estado executável no sistema.

Conceptualmente, cada processo pode ser visto como uma máquina virtual capaz de executar um determinado repertório de instruções e que tem a possibilidade de endereçar uma zona de memória limitada. As instruções executáveis pelo processo correspondem a um subconjunto das instruções do processador, acrescido de um conjunto de primitivas ou funções do sistema operativo que permitem ao processo controlar a sua execução e interagir com os restantes processos. A limitação do espaço de endereçamento a que um processo pode aceder tem em vista proteger os restantes processos e o sistema de interacções não autorizadas por parte desse processo.

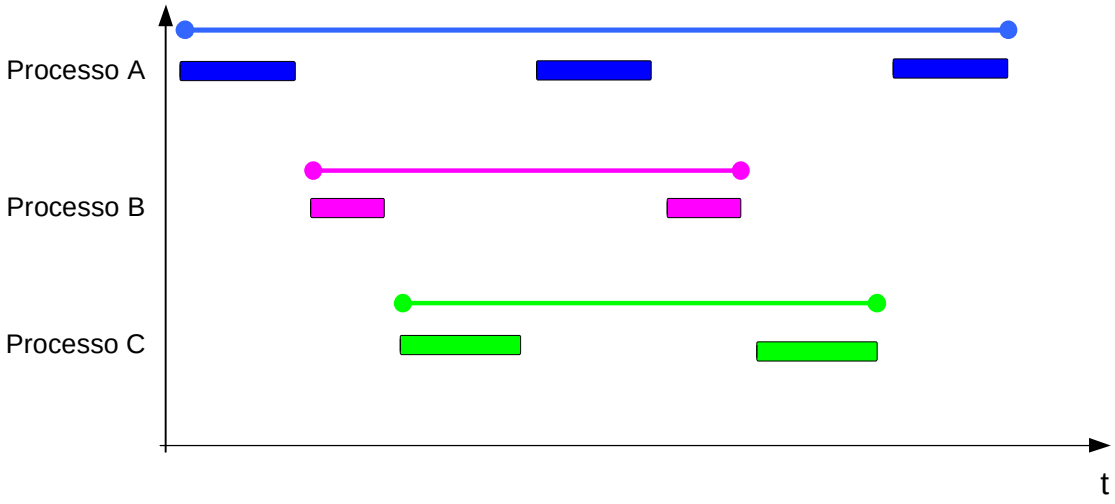


Figura 3.1 – Multiplexagem do tempo de execução da CPU pelos diversos processos executáveis

O estado de execução de cada processo encontra-se caracterizado no contexto desse processo. O contexto de um processo é materializado por um conjunto de dados estruturados, interno ao sistema operativo, onde se incluem todas as informações sobre o estado e ambiente de execução, e sobre os recursos utilizados e a respectiva contabilização.

A criação dos processos é efectuada de forma explícita recorrendo a uma primitiva que, normalmente, permite especificar o ambiente de execução. A criação de processos pode levar à definição de uma relação hierárquica entre os processos, com processos pai e processos filho. Em geral um processo pai pode interferir com a execução dos seus filhos.

Quando um conjunto de processos colabora entre si para a execução de um algoritmo comum, é indispensável que haja mecanismos de sincronização que permitam controlar essa colaboração.

Os três principais problemas a resolver com a sincronização são:

- **Cooperação entre processos**: necessária quando um processo espera que outro lhe assinale a conclusão da execução de uma determinada operação;
- **Competição por recursos**: como os sistemas dispõem de recursos que têm de ser utilizados em exclusividade ou cujo número é limitado, a sincronização deve garantir que a gestão dos recursos é correcta; e

- **Exclusão mútua**: deriva à multiplexagem do tempo de execução no processador, o acesso de um processo a uma estrutura de dados partilhada com outros processos pode ser intercalada no tempo por acessos de outros processos. Se não houver mecanismos de prevenção, o resultado da leitura de estruturas de dados partilhados torna-se imprevisível.

3.2 Planeamento do processador

O planeamento do processador refere-se à forma ou técnicas que se usam para decidir quanto tempo de execução e quando se atribui a cada processo do sistema. Obviamente, se o sistema é mono-utilizador e mono-tarefa não é necessária esta decisão. No entanto, nos restantes sistemas esta gestão é crítica para o bom funcionamento do sistema.

3.2.1 Níveis de planeamento

Quando se consideram os sistemas de planeamento do sistema operativo, geralmente identificam-se três níveis: o alto, o médio e o baixo.

O **alto nível** decide que trabalhos (conjuntos de processos) são candidatos a converter-se em processos competindo pelos recursos do sistema.

O **nível médio** decide que processos são suspensos ou reactivados para conseguir atingir certas metas de rendimento.

O processo de planeamento de **baixo nível** (ou **despacho**) decide qual dos processos (que já foram listados pelos outros dois processos de planeamento) é que deve ter oportunidade de executar na unidade central de processamento.

Nesta secção serão analisados, em particular, as estratégias utilizadas no planeamento de baixo nível, por serem estas que conduzem à selecção do processo a executar.

3.2.2 Objectivos do planeamento

As estratégias de planeamento têm por objectivo assegurar que os processos tenham adequada oportunidade de executar e que se atinja um bom rendimento do sistema, minimizando, em simultâneo, a sobrecarga (overhead) resultante da execução do próprio processo de planeamento. Em geral, são definidos cinco objectivos principais:

- **Justiça ou Imparcialidade:** Todos os processos são tratados de igual modo, e até terminarem, conseguem obter oportunidade de execução ou intervalos de tempo de execução.
- **Maximizar a Produção:** O sistema deve finalizar o maior número de processos por unidade de tempo que seja possível.
- **Maximizar o Tempo de Resposta:** O sistema deve responder consistentemente aos pedidos de cada utilizador ou processo.
- **Evitar situações de espera indefinida:** Os processos devem terminar num prazo finito de tempo.
- **Previsibilidade do sistema:** quando o número de pedidos de trabalho é reduzido o sistema deve responder rapidamente, e com o aumento dos pedidos de trabalho a degradação da velocidade de resposta deve ser gradual. Por outro lado, se um processo é executado em condições de operação do sistema semelhantes, então a resposta deve ser semelhante.

3.2.3 Características dos processos a considerar

Nem todos os equipamentos computacionais processam o mesmo tipo de trabalhos. Um algoritmo de planeamento que num sistema tem um desempenho excelente, pode ter um rendimento péssimo noutro, cujos processos tenham características diferentes. Estas características podem ser:

- **Quantidade de Entradas/Saídas:** Existem processos que realizam uma grande quantidade de operações de E/S (as aplicações de bases de dados, por exemplo).
- **Quantidade de utilização de CPU:** Existem processos que não realizam muitas operações de E/S mas que, em vez disso, usam intensivamente a unidade central de processamento. Por exemplo, operações de cálculo com matrizes.
- **Processos de Lote ou Interactivos:** Quando se compara um processo de lote (que processa dados previamente guardados) com um programa interactivo (em que os dados são introduzidos pelo operador "na hora") verifica-se que o primeiro é mais eficiente considerando a leitura de dados, já que geralmente o faz a partir de ficheiros, enquanto que no segundo se espera muito tempo pelas respostas dos utilizadores.

- **Processos em Tempo Real:** Se os processos devem dar resposta em tempo real é necessário que sejam tidas em conta as prioridades para a execução dos processos.

- **Longevidade dos Processos:** Existem processos que tipicamente requerem várias horas para concluir a sua actividade, enquanto que outros apenas necessitam de alguns segundos.

3.2.4 Planeamento apropriativo ou não apropriativo (preemptive or not preemptive)

O planeamento apropriativo, também designado de planeamento com preempção, é aquele em que os processo que adquirem vez para entrar em execução não podem ser suspensos, ou seja, não lhes pode ser retirado o acesso à CPU. Este esquema pode ser perigoso, já que se um processo, acidental ou deliberadamente, contém ciclos infinitos, o resto dos processos podem ficar indefinidamente em espera.

O planeamento apropriativo apresenta alguns inconvenientes, já que um trabalho muito grande atrasa muito a execução dos pequenos. Um outro exemplo é o da entrada de um pedido de execução de um processo de alta prioridade, este processo vai ter de esperar que o processo actualmente em execução termine.

O planeamento não apropriativo é aquele em que existe um relógio que gera interrupções periódicas nas quais o despacho, que é o processo de planeamento, toma controlo sistema e decide se o actual processo deve continuar em execução ou se a vez deve passar para outro processo. Este mesmo relógio pode servir para despoletar processos gerados pelo relógio do sistema, por exemplo os que são programados com base na hora, minuto, dia do mês, dia da semana ou dia do ano.

3.2.5 Métodos de selecção de processos para entrada em execução

Os principais critérios utilizados pelos despachos para realizarem a selecção dos processos que devem entrar em execução são os seguintes:

- **Por prioridade:** Os processos de maior prioridade são executados em primeiro lugar. Se existem vários processos de maior prioridade que outros, mas de igual prioridade entre si, estes podem ser executados por ordem de chegada ou por "round robin". A vantagem deste algoritmo é que é flexível

relativamente a permitir que certos processos sejam executados em primeiro lugar e, inclusivamente, por mais tempo. A principal desvantagem é que pode provocar a colocação em espera indefinida dos processos de menor prioridade. Considere-se. por exemplo, que o sistemas admite processos de 1ª prioridade e processos de 2ª prioridade. Se durante todo o tempo de execução do sistema entrarem novos processos de 1ª prioridade ao mesmo ritmo que terminam os processos deste grau de prioridade, o efeito é que os de 2ª prioridade ficam à espera para sempre. Deste modo, o comportamento do sistema torna-se imprevisível para os processos de baixa prioridade.

- **O trabalho mais curto primeiro:** É difícil de levar a cabo porque requer uma estimativa de quanto tempo cada processo necessita para terminar. Caso tal informação esteja disponível, executam-se primeiro os trabalhos que necessitam de menos tempo e desta forma obtém-se o melhor tempo médio de resposta para todos os processos. Por exemplo, se chegam 5 processos A, B, C, D e E cujos tempos de CPU são 26, 18, 24, 12 e 4 unidades de tempo, a ordem de execução será E, D, B, C e A (4, 12, 18, 24 e 26 unidades de tempo respectivamente).
  Na tabela seguinte mostra-se em que unidade de tempo começa a execução de cada processo. Neste exemplo todos os processos começaram a esperar na unidade de tempo zero. Nestas condições o tempo médio de espera é de 39 unidades.

Processo	Entrada em Espera	Entrada em Execução	Tempo de Espera
A	0	4	4
B	0	16	16
C	0	34	34
D	0	58	58
E	0	84	84

Tempo médio = (4 + 16 + 34 + 58 + 84)/5 = 39 unidades

- **FIFO:** É um critério muito simples de implementar, uma vez que a ordem de execução dos processos coincide com a sua ordem de chegada. A vantagem deste algoritmo é que é justo e não provoca colocação em espera indefinida. A desvantagem é que não aproveita nenhuma característica dos processos e pode não servir para processamento em tempo real. Na

realidade, o tempo médio de resposta pode ser muito mau comparado com o resultante da aplicação do critério anterior. Retomando o mesmo exemplo, obtém-se um tempo médio de resposta (26+44+68+80+84)/5 = 60 unidades, o qual é muito superior às 39 unidades (que é o melhor tempo possível).

- ***Round Robin*:** Também designada por "à vez", consiste em dar a cada processo um intervalo de tempo de execução (chamado fracção de tempo). Cada vez que termina esse intervalo de tempo o contexto do processo é copiado para um lugar seguro e é dada a vez a outro processo. Os processos estão ordenados numa fila circular. Por exemplo, se existirem três processos A, B e C, após duas passagens do processo de planeamento atribuiriam a vez aos processos na ordem A, B, C, A, B, C. A vantagem deste algoritmo é a sua simplicidade, é justo e não provoca a colocação de processos em espera indefinida.

- **O tempo restante mais curto:** É parecido com o do trabalho mais curto, mas aqui calcula-se continuamente quanto tempo resta a cada processo para terminar, incluindo os novos, e aquele que apresentar menos tempo para finalizar é o escolhido para entrar em execução. A vantagem é que é muito útil para sistemas de tempo partilhado porque se aproxima muito do melhor tempo de resposta, para além de responder dinamicamente à longevidade dos processos. A sua desvantagem é provocar uma maior sobrecarga para o sistema, uma vez que o algoritmo de selecção é mais complexo.

- **A taxa de resposta mais alta:** Este algoritmo concede a vez de execução ao processo que apresentar o valor mais elevado quando é aplicada a seguinte fórmula:

$$\text{valor} = \frac{\text{tempo que esperou} + \text{tempo total para terminar}}{\text{tempo total para terminar}}$$

Isto é, o valor é modificado dinamicamente o que permite melhorar um pouco as deficiências do algoritmo "trabalho mais curto".

- **Por política:** Uma forma de atribuir a vez de execução é por política, na qual se estabelece algum regulamento específico que o processo de planeamento deve observar. Por exemplo,

uma política pode ser que todos os processos recebam o mesmo tempo de utilização da CPU em qualquer momento. Isto significa, por exemplo, que, se existirem 2 processos que já utilizaram 20 unidades de tempo cada um (tempo acumulado em fracções de tempos de 5 unidades) e nesse momento entrar um terceiro processo, o processo de planeamento lhe irá dar imediatamente a vez de execução durante 20 unidades de tempo. Uma vez que todos os processos estejam equilibrados em termos de tempo de utilização da CPU, passa a aplicar-se o método "round robin".

3.3 Problemas de Concorrência

Os sistemas de tempo partilhado (aqueles com vários utilizadores, processos, tarefas, trabalhos que repartem entre si a utilização da CPU) apresentam muitos problemas uma vez que os processos entram em concorrência pelos recursos do sistema. Imagine-se, por exemplo, que um processo está a escrever na unidade de banda magnética e termina a sua fracção de tempo de execução e que o processo seleccionado em seguida também começa a escrever sobre a mesma banda magnética. O resultado é uma banda magnética cujo conteúdo é um conjunto inútil de dados misturados. De modo idêntico às bandas magnéticas, existe uma grande quantidade de recursos cujo acesso deve ser controlado para evitar os problemas da concorrência.

O sistema operativo deve disponibilizar mecanismos para sincronizar a execução de processos assegurando a exclusão mútua no acesso aos recursos, por exemplo através de semáforos, envio de mensagens, "pipes", etc.

Os semáforos são rotinas de software que, no seu nível mais interno, recorrem ao hardware para conseguir atingir exclusão mútua no uso de recursos. Para entender este e outros mecanismos é importante entender os problemas gerais de concorrência, os quais se passam a descrever.

- **Condições de Corrida ou de concorrência**: A condição de corrida (*race condition*) ocorre quando dois ou mais processos acedem a um recurso partilhado sem que haja controlo. O resultado combinado destes acessos depende da ordem de chegada. Considere-se, por exemplo, que dois clientes de uma mesma conta bancária realizam, simultaneamente, cada um uma operação em terminais Multibanco diferentes. O utilizador A quer fazer um depósito. O B um levantamento. O utilizador A começa a transacção e vê que o saldo é 1000. Nesse

momento perde a sua vez de execução (e o seu saldo permanece como 1000). Entretanto o utilizador B inicia o levantamento. Como o saldo é 1000, retira 200, armazena o novo saldo (que é 800) e termina. A vez de execução regressa ao utilizador A que faz um depósito de 100, ficando *saldo* = *saldo* + 100 = 1000 + 100 = 1000. Como se vê, o levantamento perdeu-se. Apesar de isto poder agradar muito aos utilizadores A e B, ao banco não convém esta transacção. Mas o erro podia ser em sentido contrário, ficando o saldo final em 800.

- **Colocação em espera indefinida**: Corresponde à situação em que os processos nunca dispõem de suficiente tempo de execução para terminarem as suas tarefas. Pode ocorrer, por exemplo, quando um processo obtém um recurso, o marca como "ocupado" e termina a sua execução sem o marcar como "desocupado". Se algum outro processo solicitar esse recurso, vai encontrá-lo "ocupado" e esperará indefinidamente que fique "desocupado".

- **Condição de Espera Circular**: Isto ocorre quando dois ou mais processos formam uma cadeia de espera que os envolve a todos. Considere-se, por exemplo, que o processo A tem atribuído o recurso "banda magnética" e o processo B tem atribuído o recurso "disco". Entretanto o processo A pede o recurso "disco" e o processo B pede o recurso "banda magnética". Fica-se perante uma situação de espera circular entre estes dois processos. A situação pode ser evitada forçando a retirada de um recurso a qualquer um dos dois processos.

- **Condição de Preempção**: Esta condição não resulta precisamente da concorrência, mas joga um papel importante neste ambiente. Ocorre quando um processo tem atribuído um recurso que não lhe pode ser retirado por nenhum motivo, e estará indisponível até que o processo o "liberte" por sua própria vontade.

- **Condição de Espera Activa**: Ocorre quando um processo solicita um recurso que já está atribuído a outro processo e a condição de não apropriação está em vigor. O processo gastará o resto da sua fracção de tempo verificando se o recurso já foi libertado. Isto é, desperdiça o seu tempo de execução à espera de um evento que não irá acontecer. A solução mais comum para este problema é, quando se detecta esta situação, colocar o processo num estado de bloqueado, concedendo imediatamente a vez de execução a outro processo.

- **Condição de Exclusão Mútua**: Quando um processo usa um recurso do sistema realiza uma série de operações sobre o recurso e posteriormente deixa de o usar. A secção de código em que ocorre a utilização do recurso chama-se "região crítica". A condição de exclusão mútua estabelece que só é admissível haver um (e um só) processo dentro de uma dada região crítica. Isto é, em qualquer momento somente um processo pode usar um determinado recurso. O conceito de "região crítica" foi desenvolvido para se conseguir atingir a exclusão mútua, e recorre a algumas técnicas destinadas a controlar a entrada na região crítica, como sejam semáforos, monitores ou o algoritmo de Dekker e Peterson.

- **Condição de Ocupar e Esperar por um Recurso**: Ocorre quando um processo solicita e obtém um recurso mas antes de o libertar, solicita outro recurso que já está atribuído a outro processo.

Os problemas descritos são todos importantes para o sistema operativo, já que deve ser capaz de os evitar ou de os corrigir. Talvez o problema mais importante que pode ocorrer num ambiente de concorrência é a interblocagem (em inglês deadlock). A interblocagem é uma condição que todos os sistemas ou conjunto de processos tentam evitar e que ocorre quando, ao mesmo tempo, se conjugam quatro condições: a condição de preempção, a condição de espera circular, a condição de exclusão mútua e a condição de ocupar e esperar um recurso.

Se o deadlock envolver todos os processos do sistema, o sistema não poderá fazer nada de produtivo. Se o deadlock envolve apenas alguns processos, estes ficarão congelados para sempre.

A tentativa de resolução do problema do deadlock originou uma série de estudos e técnicas muito úteis, já que este problema pode surgir tanto numa única máquina como ser uma consequência da partilha de recursos numa rede. As técnicas para prevenir o deadlock consistem em disponibilizar mecanismos para evitar que ocorram uma ou várias das quatro condições necessárias do deadlock. Algumas delas são:

- **Atribuir recursos em ordem linear**: Isto significa que todos os recursos estão etiquetados com um valor diferente e os processos só podem fazer pedidos de recursos "para a frente". Isto é, que se um processo tem o recurso com etiqueta "5" não pode pedir recursos cuja etiqueta seja menor que "5". Com este método evita-se a condição de ocupar e esperar um recurso.

- **Atribuir tudo ou nada**: Este mecanismo obriga a que um processo peça todos os recursos de que vai necessitar de uma vez. O sistema somente os pode atribuir se estiverem todos disponíveis. Caso contrário o sistema operativo não lhe atribui nenhum recurso e bloqueia o processo.

- **Algoritmo do banqueiro**: Este algoritmo recorre a uma tabela de recursos para registar quantos recursos existem de cada tipo. Por outro lado todos os processos têm de informar sobre o máximo de recursos que pretendem usar de cada tipo. Quando um processo solicita um recurso, o algoritmo verifica se atribuindo esse recurso ainda continua a dispor de outros recursos do mesmo tipo para poder satisfazer outros processos no sistema que ainda não tenham atingido o seu máximo. Se a resposta é afirmativa, diz-se que o sistema está em "estado seguro" e o recurso pode ser atribuído. Se a resposta é negativa, diz-se que o sistema está em estado inseguro e o processo entra fica bloqueado.

Para detectar um deadlock, pode usar-se o algoritmo do banqueiro que, apesar de não confirmar a existência de um deadlock, permite verificar a existência de estados inseguros, os quais são o estado prévio à ocorrência de um deadlock. Não obstante, para detectar o deadlock podem usar-se os "gráficos de recursos". Nestes gráficos podem usar-se círculos para representar os processos, quadrados para os recursos e setas para indicar se um recurso já está atribuído a um processo ou se um processo está à espera de um recurso. O deadlock é detectado quando, partindo de um processo ou recurso, e percorrendo as setas se pode regressar ao dito processo ou recurso. Por exemplo, suponha os seguintes eventos:

- t1: O processo A solicita o recurso 1, que lhe é atribuído.
- t2: O processo A termina a sua fracção de tempo.
- t3: O processo B solicita o recurso 2, que lhe é atribuído.
- t4: O processo B termina a sua fracção de tempo.
- t5: O processo C solicita o recurso 3, que lhe é atribuído.
- t6: O processo C solicita o recurso 2 e, como está ocupado pelo processo B, entra em espera.
- t7: O processo B solicita o recurso 1 e entra em espera, porque aquele está ocupado pelo processo A.
- t8: O processo A solicita o recurso 3 e entra em espera porque está ocupado pelo processo C.

Na Figura 3.2 pode observar-se como o gráfico de recursos evolui com a progressão dos eventos até ocorrer o deadlock que, como se referiu, acontece quando se podem seguir as setas desde um processo ou recurso até regressar ao ponto de partida. Neste deadlock estão envolvidos os processos A, B e C.

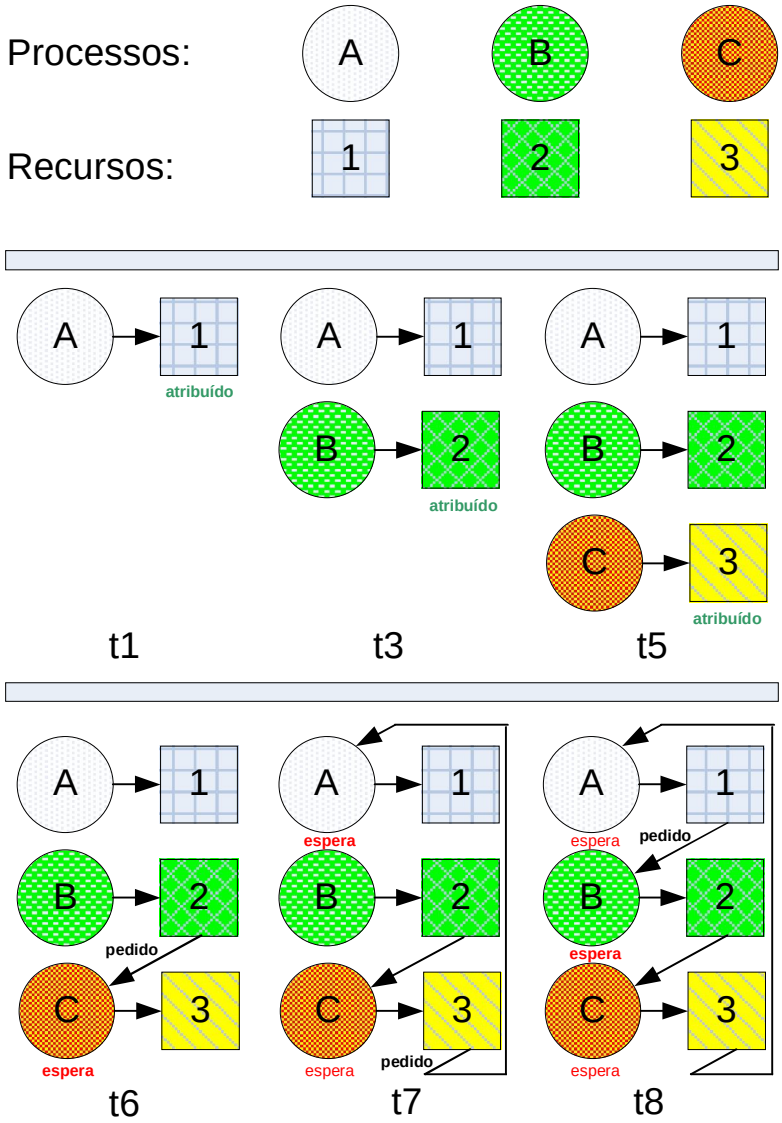


Figura 3.2 – Detecção de deadlocks

Se for detectado um deadlock, é obvio que o sistema está com problemas e só existem duas hipóteses de resolução:

- ter um mecanismo de suspensão que permita copiar todo o contexto de um processo incluindo valores de memória e

aspecto dos periféricos que esteja a usar, e reagendá-lo para outra ocasião; ou

- simplesmente eliminar um processo retirando-lhe o recurso, mas levando esse processo a perder os dados e o tempo de execução.

3.4 Semáforos

Existem diversas soluções possíveis para implementar a exclusão mútua. No entanto muitas soluções são demasiado penalizantes para o desempenho do sistema, por exemplo todas as que envolvem a espera activa pela libertação do recurso.

Os semáforos são uma solução eficiente para assegurar a exclusão no acessos a recursos. Um semáforo é constituído por uma variável de controlo inteira e por uma lista de processos. O conteúdo da variável reflecte a quantidade de recursos disponíveis no sistema. A primitiva **Esperar** bloqueia um processo quando a variável do semáforo tem, à partida, o valor zero. O contexto do processo é colocado na fila de processos do semáforo e o seu estado passa a bloqueado. A primitiva **Assinalar** incrementa a variável de controlo, se não existirem processos bloqueados, ou torna executável um processo.

A Figura 3.3 ilustra os estados possíveis que um processo pode apresentar no sistema. Quando um processo é criado passa ao estado Executável. Periodicamente esse processo será seleccionado pelo Despacho e entrará em execução na CPU. Quando termina a respectiva fracção de tempo o Despacho retira-o de execução regressando ao estado Executável. Se durante a execução o processo pedir um recurso que o sistema de semáforos verifica não estar disponível, então é invocada a primitiva Esperar e o processo passa ao estado Bloqueado. Quando o recurso pedido pelo processo for libertado o mecanismo de controlo faz com que o processo regresse ao estado Executável através da invocação da primitiva Assinalar.

Como foi anteriormente mencionado, a variável do semáforo indica o número de recursos de uma dado tipo que estão disponíveis no momento. Cada vez que um recurso desse tipo é pedido o valor da variável de controlo é decrementado. Se o número resultante for inferior a 0, então quer dizer que não existem recursos disponíveis e o processo tem de entrar em espera. Quando um recurso é libertado o valor da variável de controlo é incrementado. Se o resultado for do incremento for negativo, quer dizer que existem pelo menos um

processo em espera que deve ser desbloqueado. Quando a variável é inicializada com uma unidade, o recurso é único e será utilizado em exclusividade. Se for inicializada com um valor superior, quer dizer que existem múltiplos recursos disponíveis do tipo considerado.

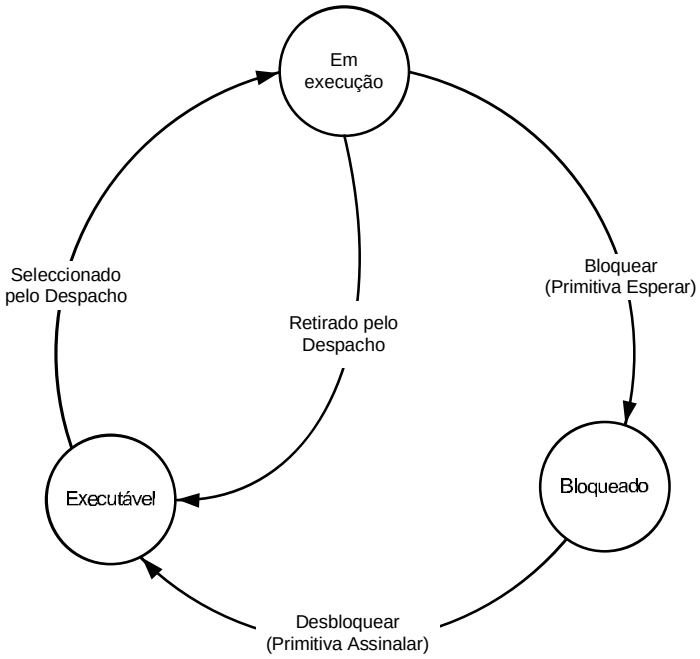


Figura 3.3 – Diagrama de estados dos processos utilizando semáforos

A acção de Assinalar pode ser feita de forma directa ou indirecta. No primeiro caso um processo actua directamente sobre o estado de outro, no segundo é utilizada uma estrutura de dados intermédia que implementa a sincronização. A sincronização indirecta ainda se pode subdividir consoante o acontecimento fica, ou não, memorizado quando o processo a que se dirige está ou não está à espera. Os mecanismos indirectos com memória são os mais frequentes e são facilmente implementados utilizando semáforos.

Verifique os seus conhecimentos...

1. O que entende por processo?

2. O que entende por multiplexagem do tempo de execução da CPU? Em que medida é que esta característica dos sistemas operativos contribui para o multiprocessamento?

3. Indique quais são os três problemas essenciais que se pretendem resolver com os métodos de sincronização entre processos e como se caracterizam.

4. Qual a função do planeamento do processador realizado pelo sistema operativo? Quais os principais objectivos a alcançar com este planeamento?

5. Quais os principais critérios utilizados pelos despachos para realizarem a selecção dos processos que devem entrar em execução?

6. Os semáforos são um mecanismo de sincronização sem espera activa. Explique o que entende por espera activa e indique como é que os semáforos resolvem este problema.

7. O que entende por exclusão mútua?

8. Qual a consequência da ocorrência de um deadlock em todos os processos em execução num computador?

Capítulo 4
Núcleo do Sistema Operativo

Como se viu anteriormente pode considerar-se que o sistema operativo é composto por um conjunto de camadas que implementam as funções necessárias ao suporte do modelo computacional. As operações associadas a cada um destes níveis não têm todas o mesmo grau de complexidade e, sobretudo, não implicam o mesmo tipo de protecções. É vulgar agrupar as operações que necessitam de se executar no nível mais privilegiado da máquina e considerá-las como o núcleo (kernel) do sistema operativo.

As operações incluídas neste núcleo variam de sistema para sistema. No entanto existem duas grandes categorias de núcleos do sistema operativo: núcleos monolíticos e micronúcleos (microkernels). O primeiro tipo de núcleos é o tradicionalmente mais usado, enquanto que os micronúcleos constituem a tendência no desenho dos sistemas operativos modernos.

No presente capítulo serão apresentados os conceitos que permitem compreender as diferenças entre ambas as categorias.

4.1 Revisões sobre a arquitectura e funcionamento dos processadores

A arquitectura mínima de um processador típico inclui as seguintes unidades internas (Figura 4.1):

- unidade central de processamento (CPU);
- memória, que pode incluir:
  - memória central; e
  - memória cache.
- barramentos (buses), incluindo:
  - Bus de dados;
  - Bus de endereços;
  - Bus de controlo.

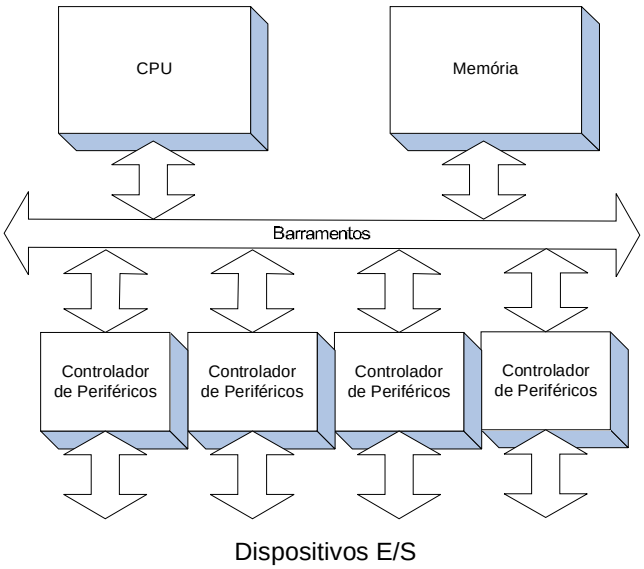


Figura 4.1 – Arquitectura típica dos Processadores

- controladores de periféricos, por exemplo:
  - Controlador de disco;
  - Controlador de linhas assíncronas;
  - Controlador de protocolos síncronos;
  - Controlador de portos paralelos;
  - Controlador de redes locais de computadores.

Internamente, uma CPU é composta, no mínimo, pelos seguintes componentes:

- **Unidade Aritmética e Lógica –** é a unidade responsável pela execução das operações aritméticas e lógicas. Nas CPU actuais é comum existir coprocessamento para a realização destas operações, o qual é realizado por coprocessadores que podem estar integrados na chip da CPU;
- **Unidade de Controlo** – é a unidade que descodifica as instruções, fazendo a sua conversão para os microcomandos que são executados internamente na CPU, por exemplo na UAL.
- **Unidade de Gestão de Endereços** – é a unidade que possibilita:

  - Recolocar dinamicamente o código e os dados na memória sem que seja necessário alterar o programa original;
  - Detectar pedidos de acesso, originados nos processos, a endereços inválidos;
  - Proteger os blocos de memória de acordo com o modo de execução do processador.

- **Tratamento de Interrupções –** é o componente que permite:
  - Controlar a aceitação das interrupções;
  - Salvaguardar o contexto;
  - Seleccionar a rotina de tratamento da interrupção.

No que diz respeito às funcionalidades do núcleo do sistema operativo, os elementos da arquitectura dos processadores com maior relevância são as interrupções, os mecanismos de protecção e a gestão de endereços.

4.2 Interrupções e Mecanismos de protecção

As **interrupções** são o mecanismo de base na implementação dos sistemas operativos. As interrupções são o único meio existente que permite realizar a transferência assíncrona da execução de um programa para outro. A importância de se poder controlar o processamento de forma assíncrona está associado à possibilidade de se poder suspender o actual processo que está em execução e activar a resposta a eventos importantes imediatamente quando eles ocorrem, em vez de se esperar que a fracção de tempo do actual processo se esgote, e só depois se vá verificar se, entretanto, ocorreu algum evento que deva ser atendido com algum grau de prioridade fora do comum.

As interrupções podem ser provocadas:

- pelo **hardware** (originadas, por exemplo, nos controladores de periféricos);
- por **situações de excepção** (isto é, por erros ou pela ocorrência pedidos ilegais, como sejam uma operação de divisão por zero); ou
- por **instruções especiais**.

Uma fonte de interrupção de particular importância é o relógio do sistema que cria uma base de tempo que permite coordenar as acções executadas pelo sistema.

Considerando agora a **protecção**, normalmente, os mecanismos de protecção actuam a dois níveis, um que restringe aos processo de utilizador o emprego de funções de sistema e o outro relacionado com a gestão de memória.

O primeiro mecanismo de protecção está relacionado com a possibilidade de impedir que os processos do utilizador executem instruções que possam afectar o funcionamento global da máquina (ex.: inibir interrupções, entradas/saídas). Para este efeito a unidade central de processamento possui, normalmente, dois modos de funcionamento:

- o **modo sistema**, onde é permitida a execução de todas as instruções; e
- o **modo utilizador**, que restringe as instruções privilegiadas.

O segundo nível de protecção relaciona-se com as características de funcionamento da unidade de gestão de memória existente na CPU. Para garantir o correcto funcionamento do sistema é fundamental que o espaço de endereçamento dos processos do utilizador e do sistema sejam perfeitamente delimitados, impedindo que os respectivos dados possam ser corrompidos.

Como a gestão dos processos do sistema operativo necessita de interagir intimamente com a máquina esta é uma das funções executadas pelo núcleo. Conceptualmente, as funções de gestão dos processos activos realizadas no núcleo podem subdividir-se em (Figura 4.2):

- Gestão das interrupções;
- Despacho/scheduling; e
- Sincronização.

Figura 4.2 – Funções realizadas pelo núcleo relacionadas com a gestão de processos

A **gestão das interrupções** controla as interrupções provenientes do hardware que são desencadeadas por excepções ou provocados por instruções especiais. Como foi referido anteriormente, as interrupções são o único mecanismo que permite efectuar a mudança não explícita de uma sequência de instruções para outra, sendo, portanto, a base do funcionamento da gestão dos processos. As funções sistema[1] são normalmente desencadeadas por interrupções que permitem efectuar a modificação do domínio de protecção do utilizador para o do núcleo.

Os processos são caracterizados no sistema através de um contexto onde são memorizadas todas as informações que definem o respectivo ambiente de execução, tanto na perspectiva do estado do hardware como do software. O conjunto de dados do contexto inclui:

- Contexto de Hardware:
  - Acumulador
  - Registos de uso geral
  - Contador de Programa

[1] As funções sistema são constituídas por duas componentes, uma rotina de interface que é ligada ao código do utilizador e uma rotina interna do núcleo. A passagem de uma componente para a outra é efectuada através de uma interrupção de software que permite efectuar a mudança de nível de protecção e agulhar para o código do núcleo. O mecanismo de chamada das funções sistema assegura também a ligação dinâmica entre o código do utilizador e o do núcleo, evitando que os programas tenham de ser recompilados quando se efectuam alterações internas ao sistema operativo.

  - Apontador de Pilha
  - Registo de Estado do Processador
- Contexto de Software:
  - Identificação do processo e do utilizador
  - Prioridade
  - Estado do processo
  - Periféricos utilizados
  - Ficheiros abertos
  - Programa em execução
  - Directoria actual e por omissão
  - Cotas de utilização dos recursos
  - Contabilização de utilização dos recursos

O **despacho** encarrega-se de escolher o próximo processo a executar e efectuar a comutação de contextos. Dada a sua função nuclear a rotina de despacho é muito simples baseando-se, normalmente, na ordenação da lista dos processos executáveis.

Para além da função de multiplexagem do processador, é necessário que exista no sistema uma outra entidade que se encarregue da gestão global da máquina e que procure atribuir os recursos (tempo de processador, memória) aos processos que melhor se enquadram nos objectivos globais do sistema. A gestão do processador é implementa por uma entidade designada "***scheduler***" que procura optimizar a gestão dos recursos, privilegiando os processos que utilizam os recursos da forma mais adequada. Por exemplo, nos sistemas interactivos procura-se dar mais tempo de processador aos processos que efectuam frequentes E/S para os terminais em detrimento dos que usam intensivamente o processador.

Embora existam muitas outras políticas de gestão, as mais representativas são:

- **Gestão de Tempo Partilhado** – que corresponde à atribuição de um período de tempo pré-determinado a cada processo;
- **Preempção** - que retira de execução um processo sempre que outro mais prioritário se torna executável;
- **Multilista –** que selecciona os processos a executar com base em listas separadas a que correspondem diferentes prioridades de execução. A lista onde os processos são colocados depende do modo como utilizam o processador;

- **Prioridades Dinâmicas –** que modifica a prioridade dos processos em função do modo como utilizam os recursos;
- **Quantum Variável –** que modifica o valor do quantum (tempo de execução atribuído a cada processo) para permitir reduzir o número de comutações do processador quando são executados processos de processamento intenso.

Os **mecanismos de sincronização** têm de garantir que, quando necessário, uma posição de memória é lida e actualizada atomicamente. Para alcançar este objectivo a implementação da sincronização pode recorrer a:

- **Secções Críticas**, criadas através de:
  - Inibição das Interrupções;
  - Implementação dos Trincos; e
  - Exclusão Mútua em Arquitecturas Multiprocessador.
- **Semáforos**

Por exemplo, para efeitos da criação de secções críticas, a inibição das interrupções é utilizada para impedir a comutação do processador, o que impede que um segundo processo possa interferir com as acções realizadas pelo processo que está a executar as instruções que constituem a secção crítica. No entanto, o recurso à inibição das interrupções deve restringir-se a pequenas secções, dado que afecta o funcionamento global do sistema.

Considerando o recurso a trincos lógicos, a solução mais eficiente passa pela utilização da instrução "*test and set*" que permite testar e modificar uma posição de memória numa única instrução. Esta instrução é a única que garante atomicidade em operações de leitura seguida de escrita na memória.

Nas arquitecturas multiprocessador a implementação de um trinco lógico implica, igualmente, que o acesso à memória se efectue num só ciclo do bus, para garantir que não exista a possibilidade de violação da sincronização por dois processos que estão a ser executados em processadores independentes.

Finalmente, os semáforos são facilmente implementados com base no mecanismo de exclusão mútua e numa lista onde são colocados os contextos dos processos bloqueados.

4.3 Trabalhos, Processos e Threads

Os conceitos de "Trabalho", "Processo" e "Thread" servem para definir o grau de granularidade com que o sistema operativo trata o conjunto de operações que a CPU tem que realizar. Onde:

- **Trabalho**: é um conjunto de um ou mais processos. Por exemplo, se o trabalho for fazer um inventário, talvez se subdivida esse trabalho em vários processos, como sejam obter a lista de artigos, as quantidades existentes, artigos vendidos, artigos extraviados, etc.
- **Processo**: pode ser definido como sendo a imagem de um programa que se encontra em execução, isto é, em memória e usando a CPU. A este nível de granularidade, um processo tem um espaço de endereçamento de memória, uma pilha, registos e um "program counter".
- **Thread**: é uma porção ou secção de um processo que tem os seus próprios registos, pilha e "program counter" e que pode partilhar a memória com todos os outros threads que fazem parte do mesmo processo.

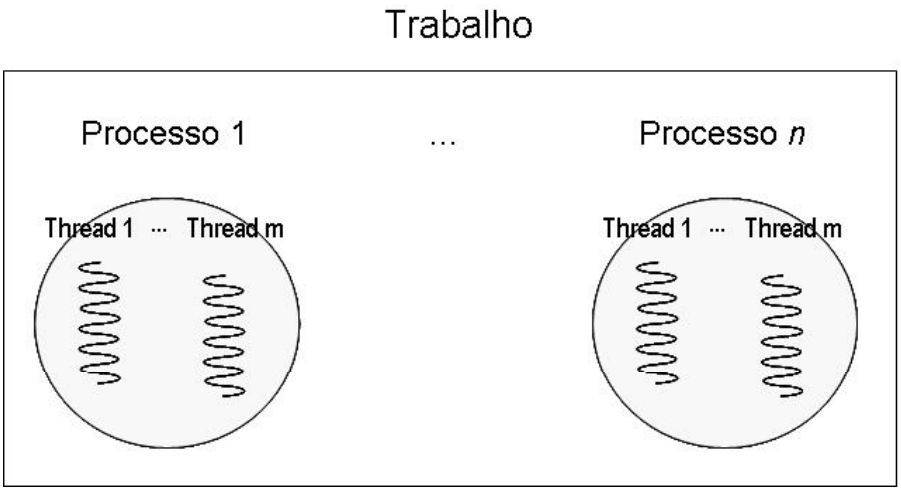


Figura 4.3 - Granularidade das operações da CPU

4.4 Objectos

Um objecto é uma entidade que contém duas partes principais: uma colecção de atributos e um conjunto de métodos (também chamados serviços). Geralmente os atributos do objecto não podem ser alterados directamente pelo utilizador, somente através dos métodos. Como os métodos são os únicos elementos do objecto visíveis ao utilizador, os métodos constituem o que se pode designar

como a "interface" do objecto. Por exemplo, para o objecto "ficheiro" os métodos disponíveis serão abrir, fechar, escrever, eliminar , etc. O modo como de facto se abre, se fecha, se apaga, o ficheiro está escondida do utilizador, isto é, os atributos e o código estão "encapsulados". A única forma de activar um método é através do envio de mensagens entre os objectos, ou até dentro de um objecto.

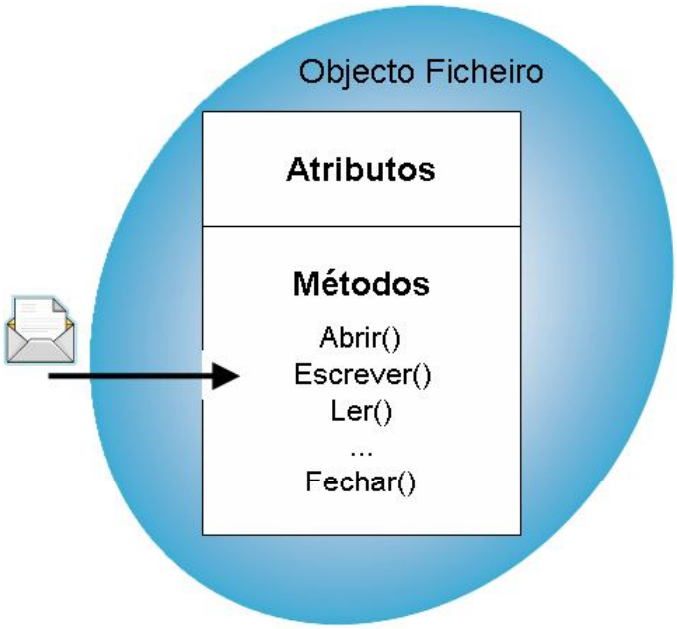


Figura 4.4 – Exemplo de um Objecto

4.5 Cliente/Servidor

O conceito de Cliente/Servidor reflecte a relação existente entre dois tipos de processos, como se indica:

- **Cliente**: é um processo que necessita de algum valor ou de alguma operação externa para poder trabalhar;
- **Servidor**: é o processo que fornece esse valor ou que realiza essa operação.

Por exemplo, um servidor de ficheiros deve executar no núcleo (ou num processo "guardião" do servidor de ficheiros) os pedidos de abertura, leitura, escrita, etc. dos ficheiros. Um cliente é um processo guardião que monitoriza os acessos nas máquinas clientes e que está em comunicação com o processo servidor através da rede, dando a ilusão ao utilizador que os ficheiros existem de forma local na máquina cliente.

4.6 Núcleo Monolítico

Os núcleos monolíticos geralmente estão divididos em duas partes estruturadas (Figura 4.5):

- o núcleo dependente do hardware; e
- o núcleo independente do hardware.

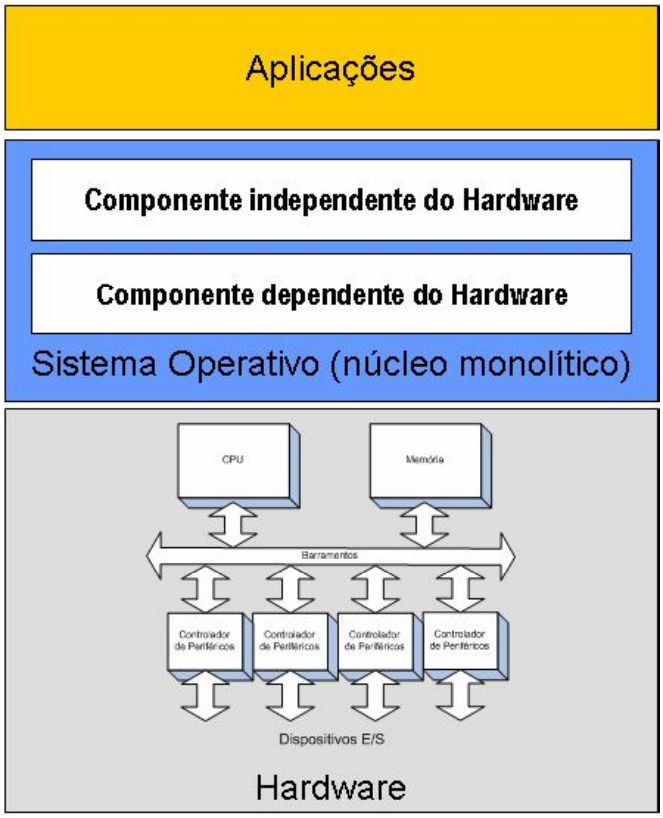


Figura 4.5 – Visão de um SO de núcleo monolítico

O núcleo dependente do hardware encarrega-se, principalmente, de gerir as interrupções do hardware, fazer a gestão de baixo nível da memória e dos discos e trabalhar com os gestores de dispositivos de baixo nível.

O núcleo independente do hardware encarrega-se de disponibilizar as chamadas ao sistema, gerir os sistemas de ficheiros e o planeamento dos processos.

Em geral esta divisão passa desapercebida ao utilizador. Para um mesmo sistema operativo correndo em diferentes plataformas, o núcleo independente é exactamente o mesmo, enquanto que o dependente deve ser modificado.

4.7 Micronúcleo ou Microkernel

Núcleos com "arquitectura micronúcleo" são aqueles que contêm unicamente a gestão de processos e threads, a gestão de baixo nível da memória, o suporte às comunicações e a gestão das interrupções e operações de baixo nível das Entradas/Saídas.

Nos sistemas operativos que contam com este tipo de núcleo são usados processos "servidores" que se encarregam de disponibilizar o resto de serviços (por exemplo, o do sistema de ficheiros) e que utilizam o núcleo recorrendo a mecanismos de comunicação (Figura 4.6).

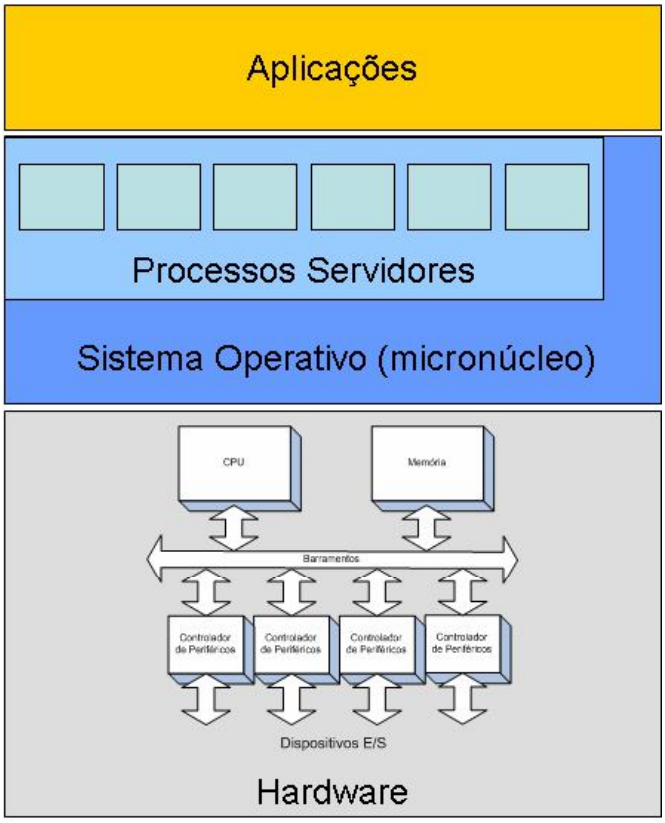


Figura 4.6 - Visão de um SO de micronúcleo

Este desenho permite que os servidores não permaneçam amarrados a um fabricante em especial. Na realidade, o utilizador pode seleccionar ou programar os seus próprios servidores. A maioria dos sistemas operativos que usam este esquema gerem os recursos do computador como se fossem objectos: os servidores disponibilizam uma série de "chamadas" ou "métodos" utilizáveis com um comportamento coerente e estruturado.

Outra das características importantes dos micronúcleos é a gestão de threads. Quando um processo é formado por um único thread, então é um processo normal, igual ao de qualquer outro sistema operativo.

Os usos mais comuns dos micronúcleos são nos sistemas operativos que tentam ser distribuídos, e naqueles que servem como base para a instalação de outros sistemas operativos. Por exemplo, o sistema operativo AMOEBA tenta ser distribuído e o sistema operativo MACH serve de base para a instalação do DOS, UNIX, etc.

Verifique os seus conhecimentos...

1. Qual a diferença entre multiprocessamento e multiprocessador?

2. Quando se fala de Gestão de Tempo Partilhado, Preempção, Multilistas, Prioridades Dinâmicas e Quantums Variáveis está a falar-se de várias políticas de gestão referentes à gestão dos processos. Caracterize sucintamente 2 destas políticas.

3. O mecanismo que está na base da implementação dos sistemas operativos e que permite comutar assincronamente a execução de um programa para outro é:
   a. a preempção
   b. a espera activa
   c. a interrupção
   d. nenhum dos anteriores

4. Um conjunto de um ou mais processos designa-se por:
   a. multi-processamento
   b. multi-threading
   c. objecto
   d. nenhum dos anteriores

5. As interrupções podem ser provocados por:
   a. hardware
   b. por situações de excepção
   c. instruções especiais
   d. todas as anteriores

6. Indique se as seguintes frases são verdadeiras ou falsas

	V	F
a. A inibição das interrupções impede a comutação do processador, mas a sua utilização deve apenas restringir-se a pequenas secções, dado que afecta o funcionamento global do sistema.		
b. O despacho encarrega-se de escolher o próximo processo a executar e efectuar a comutação de contextos.		

	V	F
c. Um processo que necessita de um valor ou de uma operação externa para poder continuar a trabalhar designa-se de servidor		
d. Os processos servidores de um sistema operativo com arquitectura micronúcleo gerem os recursos do computador como se fossem objectos		

Capítulo 5
Gestão de Memória

Dada a necessidade de se acomodar, numa memória física de dimensões fixas, múltiplos programas com tamanhos e requisitos de memória muito diferentes, a gestão de memória nos sistemas multiprogramados coloca diversos problemas. Além disso, existe também a necessidade de ultrapassar as restrições ao tamanho dos programas e facilitar a tarefa de gestão da memória física, o que foi conseguido graças ao conceito de um espaço de endereçamento virtual.

Neste capítulo serão descritas as técnicas mais usuais na gestão de memória, e revistos os principais conceitos. Serão abordados os esquemas de gestão de memória real, a multiprogramação em memória real e as suas variantes, o conceito de "overlay", a multiprogramação com transferência e os esquemas de gestão de memória virtual.

5.1 Mecanismos de Gestão de Memória

Os mecanismos de gestão de memória estão intimamente relacionados com a organização da memória do computador. Como tal, sendo o sistema operativo, uma camada de software que interage directamente com o hardware, tem de programar e gerir a Unidade de Gestão de Memória (UGM) e a memória, tentando assegurar a eficiência na utilização dos recursos da máquina.

Cada processo dispõe de um espaço de endereçamento privativo, que é definido pelo conjunto de posições de memória física a que pode aceder.

Como é sabido a memória de um sistema, onde se guarda o código e os dados das aplicações, compreende duas componentes:

- a memória principal ou física, que normalmente é volátil, e que é acedida directamente pela CPU; e
- a memória secundária ou memória de massa, onde é guardada de forma persistente a informação do sistema e que pode constituir também uma memória de trabalho.

Os programas (isto é, pelos processos) só se executam em memória física.

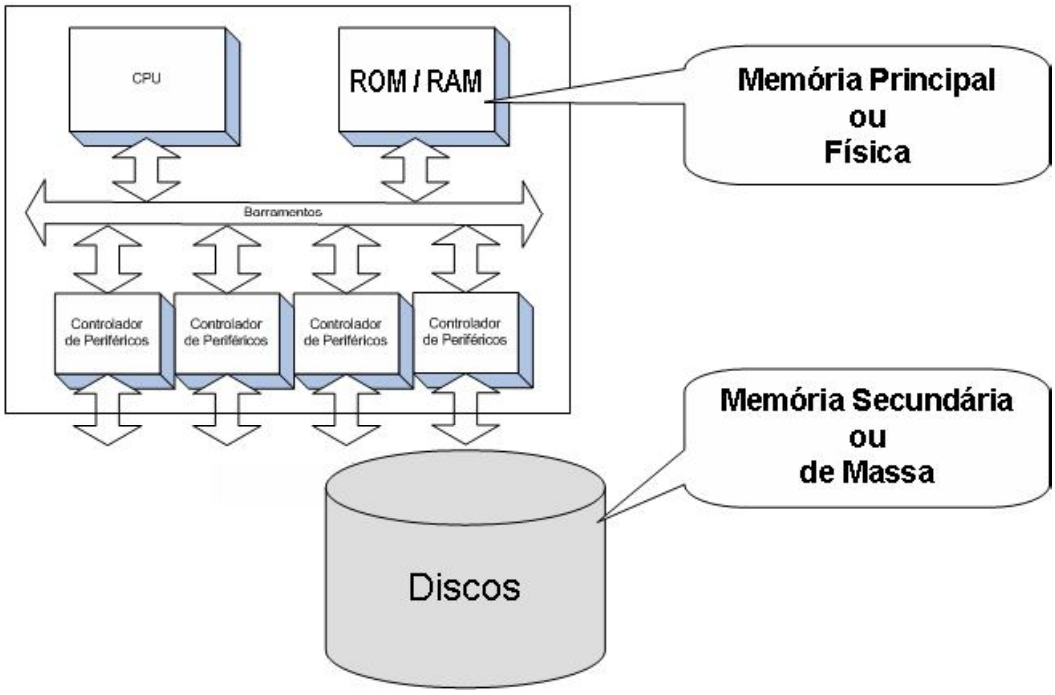


Figura 5.1 – Tipos de Memória

Os endereços gerados pelo programa são analisados pela UGM do processador que lhes aplica uma função de transformação e os envia para os circuitos de memória. Se a função utilizada for a função identidade, os endereços não são modificados e, neste caso, está-se perante um endereçamento real. Quando existe alguma transformação dos endereços está-se perante o endereçamento virtual. A Figura 5.2 apresenta uma síntese das abordagens mais comuns empregues na gestão de memória.

<table>
<tr><th colspan="3">Real</th><th colspan="2">Virtual</th></tr>
<tr><td>Mono utilizador</td><td colspan="4">Multi-programação</td></tr>
<tr><td rowspan="3"></td><td colspan="2" rowspan="2">Partições</td><td>Paginação Pura</td><td>Segmentação Pura</td></tr>
<tr><td colspan="2">Solução Mista</td></tr>
<tr><td>Fixas</td><td>Variáveis</td><td colspan="2">Protecção</td></tr>
<tr><td colspan="2">Relocalização</td><td colspan="2"></td></tr>
</table>

Figura 5.2 – Síntese de abordagens relativas à gestão de memória

5.2. Gestão de memória em sistemas mono-utilizador sem transferência de dados (*swap*)

O endereçamento real é o mais simples, e corresponde ao que foi utilizado nos primeiros computadores e nos sistemas mono-utilizador e mono-tarefa, de que são exemplo as máquinas com o sistema operativo DOS. No entanto, esta forma de endereçamento limita o tamanho dos programas e dos respectivos dados à dimensão da memória física, para além de obrigar que o programa seja executado numa zona de memória pré-definida.

Neste esquema, a memória real é usada para armazenar o programa que está em execução num dado momento, com a óbvia desvantagem de que existe limite rígido na quantidade de RAM disponível.

A organização física da memória é geralmente muito simples: o sistema operativo instala-se nas posições superiores ou inferiores da memória, seguido pelos dispositivos de gestão ("drivers"). Isto deixa um espaço contínuo de memória disponível que é utilizado pelos programas do utilizador. Geralmente a localização da pilha ("stack") corresponde às últimas posições de memória do programa, com o objectivo de que a pilha possa crescer o máximo possível. Estas diferentes opções são ilustradas na Figura 5.3.

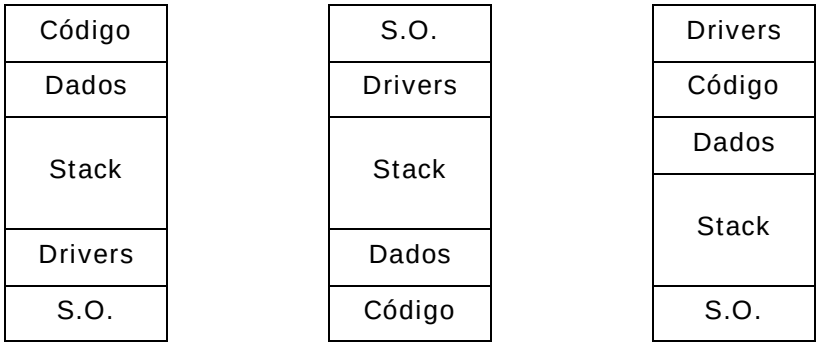


Figura 5.3 – Organização simples de memória

Como é obvio, estes esquemas não requerem algoritmos sofisticados para atribuição da memória aos diferentes processos, já que estes são executados sequencialmente à medida que os anteriores vão terminando.

5.3. Multiprogramação em memória real

Nos anos 60, as empresas e instituições que tinham investido grandes somas na compra de equipamento computacional, rapidamente se deram conta que os sistemas de processamento em lote ocupavam uma grande quantidade de tempo em operações de entrada e saída, onde a intervenção da unidade central de processamento era praticamente nula, e começaram a investigar uma solução que assegurasse que a CPU se mantinha mais tempo ocupada. Foi deste modo que nasceu o conceito de multiprogramação, o qual consiste em colocar na memória física mais de um processo em simultâneo, de forma que, se o que está em execução num dado momento entrar num período de entrada/saída, se pode seleccionar um outro processo para usar a CPU. Nesta abordagem, a memória física divide-se em secções de tamanho suficiente para conter vários programas.

Desta forma, se um sistema gastasse, por exemplo, 60% do tempo médio por processo em operações de entrada/saída, a CPU mantinha-se desocupada durante toda aquela percentagem de tempo. Com a nova técnica, pode aproveitar-se mais a CPU, pelo que o tempo médio desocupado diminui. O tempo médio que a CPU está ocupada designa-se "grau de multiprogramação". Se o sistema tivesse um único processo sempre, e se este gastasse 60% em operações de entrada/saída, o grau de multiprogramação seria 1 - 60% = 40% = 0.4. Com dois processos, para que a CPU estivesse desocupado seria necessário que ambos os processos necessitassem em simultâneo de executar entradas/saídas. Supondo que são independentes, a probabilidade de que ambos realizarem E/S é o produto das suas probabilidades, isto é, 0.6 x 0.6 = 0.36. Neste caso, o grau de multiprogramação é 1 - (probabilidade de ambos os processos realizarem E/S) = 1 - 0.36 = 0.64.

Como se vê, o sistema melhora a sua utilização da CPU em 24% ao passar de um para dois processos. Para três processos o grau de multiprogramação é $1 - (0.6)^3 = 0.784$, isto é, o sistema estará ocupado 78.4% do tempo. A fórmula do grau de multiprogramação, apesar de não ser exacta, pode servir de referência para avaliar o crescimento potencial do rendimento da CPU resultante do aumento da memória real. De notar que, a partir de determinado ponto o aumento do número de processos em RAM não resulta num incremento no uso da CPU.

No entanto, a introdução do esquema de multiprogramação em memória real fez surgir novos problemas, por exemplo o do reposicionamento de programas e da protecção e partição da memória, que serão abordados de seguida.

5.3.1. O problema do reposicionamento

Apesar de ter sido detectado pela primeira vez com a multiprogramação em memória real, o problema do reposicionamento não é exclusivo deste esquema de gestão da memória, uma vez que também ocorre nos esquemas de memória virtual.

O problema resulta do facto de que os programas que necessitam de entrar na memória real já estão compilados e ligados. Isto quer dizer que internamente os programas contêm uma série de referências a endereços de instruções, rotinas e procedimentos que podem não ser válidas no espaço de endereçamento da memória real da secção onde o programa foi colocado. Isto é, quando o programa é compilado os endereços de memória são definidos ou resolvidos de acordo com a secção de memória que o programa ocupa nesse momento, mas se o programa for colocado numa secção diferente, os endereços reais já não coincidem. Neste caso, o gestor de memória pode solucionar o problema de duas formas: de forma "estática" ou de forma "dinâmica".

A solução **estática** consiste em que todos os endereços do programa sejam recalculados no momento em que o programa é colocado em memória, isto é, praticamente volta-se a compilar o programa.

A solução **dinâmica** consiste em ter um registo que guarda o endereço base da secção que contém o programa. Cada vez que o programa faz uma referência a um endereço de memória, é-lhe somado o valor do registo base da secção para encontrar o endereço real. Por exemplo, suponha que o programa é colocado numa secção que começa na endereço 100. O programa faz referência aos endereços 50, 52 e 54. Como é fácil de entender não é o conteúdo daqueles endereços que é o desejado, antes sim o dos endereços 150, 152 e 154, já que é aí que reside o programa. Para obter os novos endereços de memória as somas 100+50, 100+52 e 100+54 são realizadas em tempo de execução. A primeira solução é preferível à segunda se o programa contém ciclos e é grande, uma vez que consumirá menos tempo na resolução inicial dos endereços que a segunda solução, que envolve a resolução dos endereços durante a execução.

5.3.2. O problema da protecção

Este problema resulta do facto de que depois de um programa ter sido carregado em memória num segmento em particular, não há nada que impeça o programador de tentar aceder (por erro ou

deliberadamente) a posições de memória anteriores ao limite inferior do seu programa ou posteriores ao endereço maior, isto é, a referenciar posições fora do seu espaço de endereços. Obviamente, este é um problema de protecção, já que não é legítima a leitura ou escrita nas áreas dos outros programas.

A solução para este problema envolve o utilização de um registo base e de um registo limite. O registo base contém o endereço de início da secção que contém o programa, enquanto que o registo limite contém o endereço onde esta termina. Cada vez que o programa faz uma referência a um endereço de memória é verificado se este se encontra dentro dos limites definidos pelos registos. Caso não seja o caso é enviada uma mensagem de erro e a execução da instrução do programa é abortada.

5.3.3. Partições fixas e partições variáveis

No esquema da multiprogramação em memória real existem duas alternativas para definir a partição de memória a atribuir a cada programa:

- partições de tamanho fixo; e
- partições de tamanho variável.

A alternativa mais simples são as partições fixas. Estas partições são criadas quando o equipamento é ligado e permanecerão com os tamanhos iniciais até que o equipamento seja desligado. É uma alternativa muito velha. Nos primeiros sistemas quem fazia a divisão de partições era o operador analisando os tamanhos estimados dos trabalhos a executar nesse dia. Por exemplo, se o sistema tinha 512 kBytes de RAM, podia atribuir 64 k para o sistema operativo, uma partição mais de 64 k, outra de 128 k e uma maior de 256 k. Este esquema era muito simples, mas inflexível, já que se surgissem trabalhos urgentes, por exemplo, de 400k, tinham que esperar por outro dia ou tinha de se reparticionar a memória, o que implicava reinicializar o equipamento.

A outra alternativa, que surgiu posteriormente em resposta à necessidade de melhorar o esquema anterior, corresponde a criar partições contíguas de tamanho variável. Para este efeito, o sistema tem de manter actualizada uma estrutura de dados, que indica onde existem espaços de RAM disponíveis, e onde se localizam as partições ocupadas por programas em execução. Assim, quando um programa faz o pedido para ser colocado na RAM, o sistema analisa os espaços para verificar se existe algum com tamanho suficiente para o programa e, se for o caso, atribui o espaço ao programa. Se

não houver nenhum, tenta reposicionar os programas existentes com o propósito de juntar todo o espaço ocupado (o mesmo é dizer todo o espaço livre) e assim obter um espaço de tamanho suficiente. Se ainda assim o programa não couber, então fica bloqueado e é seleccionado outro. O processo utilizado para juntar os espaços de memória ocupados chama-se "compactação".

Como se pode perceber facilmente, a aplicação do esquema das partições fixas suscita a questão de qual o critério a utilizar para escolher o melhor tamanho da partição a atribuir a um programa. Por exemplo, se o sistema tem dois espaços disponíveis em memória, um de 18 k e outro de 24 k, e surgir um processo com 12 k, qual deve ser atribuído?

Existem vários estratégias possíveis para dar resposta à pergunta anterior, os quais se enumeram e descrevem a seguir e se ilustram na Figura 5.4.

- **Primeiro Ajuste (*first-fit*):** corresponde a atribuir o primeiro espaço de memória que é maior que o tamanho desejado.
- **Melhor Ajuste (*best-fit*)**: corresponde a atribuir o espaço cujo tamanho exceda na menor quantidade o tamanho desejado. Requer a análise exaustiva dos espaços de memória disponíveis.
- **Pior Ajuste (*worst-fit*):** corresponde a atribuir o espaço cujo tamanho exceda na maior quantidade o tamanho desejado. Requer também a análise exaustiva.
- **O Ajuste Seguinte (*next-fit*):** é igual ao "primeiro ajuste" com a diferença que se deixa um ponteiro no lugar onde se atribuiu o último espaço para realizar a busca seguinte a partir desse ponto.
- **Ajuste Rápido:** mantêm-se listas ligadas separadas de acordo com os tamanhos dos espaços, para assim se atribuir mais rapidamente aos processos um espaço na vez correspondente;
- ***Buddy* binário**: devolve sempre blocos de dimensão igual a uma potência de 2, tal que, sendo R a dimensão do pedido, $2^{k-1} < R \leq 2^k$

Outro problema que resulta destas atribuições é a "fragmentação interna", que corresponde à quantidade de memória desperdiçada dentro da partição. Considere-se, por exemplo, a utilização da estratégia do "pior ajuste" para atribuir memória a um processo que requeira 12 kBytes e que o espaço atribuído fosse de 64 kBytes. Uma vez atribuído o espaço nesta partição estão a desperdiçar-se 52

kBytes, o que é uma grande quantidade de memória, se não poder ser utilizada por mais nenhum processo.

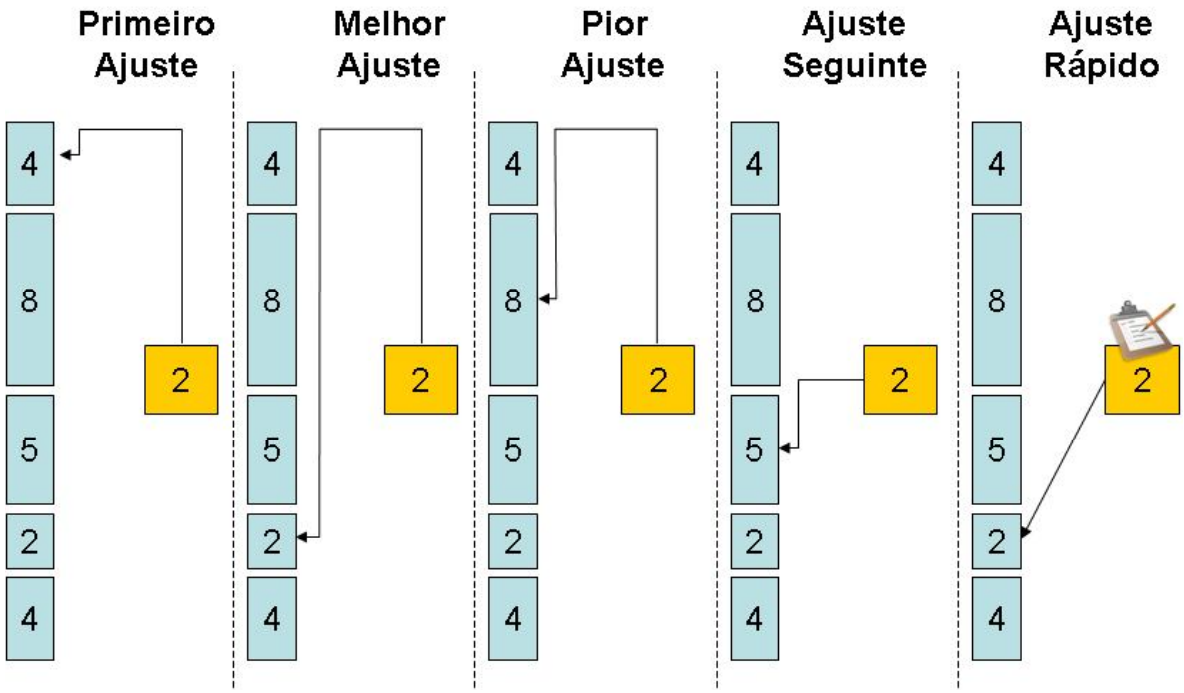


Figura 5.4 – Estratégias para afectação de memória

Por outro lado, pode ocorrer "fragmentação externa", que corresponde à existência de espaços de memória livres não contíguos. À medida que o sistema vai funcionando ao longo do dia, terminando processos e começando outros, a configuração da memória vai-se modificando, alterando a sequência de espaços livres e de lugares atribuídos. Imagine-se, a título de exemplo, que num dado instante existem três espaços livres de 12 k, 28 k e 30 k, que somados dão 70 k. No entanto, se nesse momento chegar um processo com esse tamanho, não pode entrar em memória uma vez que as posições de memória livres não são contíguas (a menos que se realize a compactação).

De qualquer forma, a multiprogramação foi um progresso significativo para melhorar o aproveitamento da unidade central de processamento e da própria memória.

5.3.3. Mecanismo de Sobreposição (Overlay)

Face às limitações do tamanho de memória, foram investigadas formas de executar grandes quantidades de código em áreas de

memória muito pequenas, auxiliados por algumas chamadas ao sistema operativo. É assim que surgiu o mecanismo de sobreposição, também conhecido por "overlays".

Esta técnica consiste na divisão lógica, realizada pelo programador, de um programa maiores que o espaço disponível na memória física em secções que se podem armazenar nas partições da RAM (Figura 5.5).

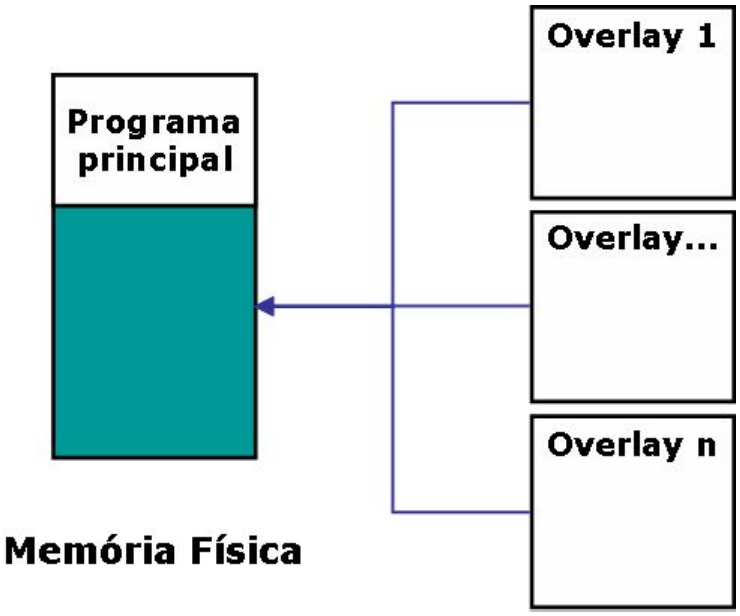


Figura 5.5 – Utilização de overlays

No final de cada secção do programa (ou onde necessário) o programador insere chamadas ao sistema operativo ordenando a substituição da secção actualmente existente na RAM por outra, que nesse momento reside em disco rígido ou num outro meio de armazenamento secundário. Apesar desta técnica ser eficaz (porque resolve o problema) não é eficiente (já que não o resolvia da melhor forma). Esta solução obriga que o programador tenha um conhecimento muito profundo do equipamento computacional e das chamadas ao sistema operativo. Outra desvantagem é a portabilidade entre sistemas, uma vez que as chamadas podem mudar e os tamanhos das partições também.

Em resumo, a técnica de sobreposição (overlay), apesar de permitir executar programas maiores do que a memória, é de difícil utilização e não resolve o problema satisfatoriamente, sendo a divisão do código definida pelo programador e o controlo dos dados em memória realizado pelo sistema operativo. Este esquema de gestão de memória caiu em desuso.

5.4. Multiprogramação em memória virtual

A necessidade cada vez mais imperiosa de executar programas grandes e o crescimento de capacidade das unidades centrais de processamento levaram os desenhadores dos sistemas operativos a implementar mecanismos para executar automaticamente programas maiores que a memória real disponível, isto é, de disponibilizar "memória virtual".

A memória virtual chama-se assim porque o programador vê uma quantidade de memória muito maior que a real. Na realidade trata-se da soma da memória de armazenamento primário com uma quantidade de disco atribuída para armazenamento secundário (Figura 5.6). O sistema operativo, no seu módulo de gestão de memória, encarrega-se de transferir programas inteiros, segmentos ou páginas entre a memória real e o meio de armazenamento secundário. Se o que se transfere são processos inteiros, fala-se então de multiprogramação em memória real, mas se o que se transferem são segmentos ou páginas, então fala-se de multiprogramação com memória virtual.

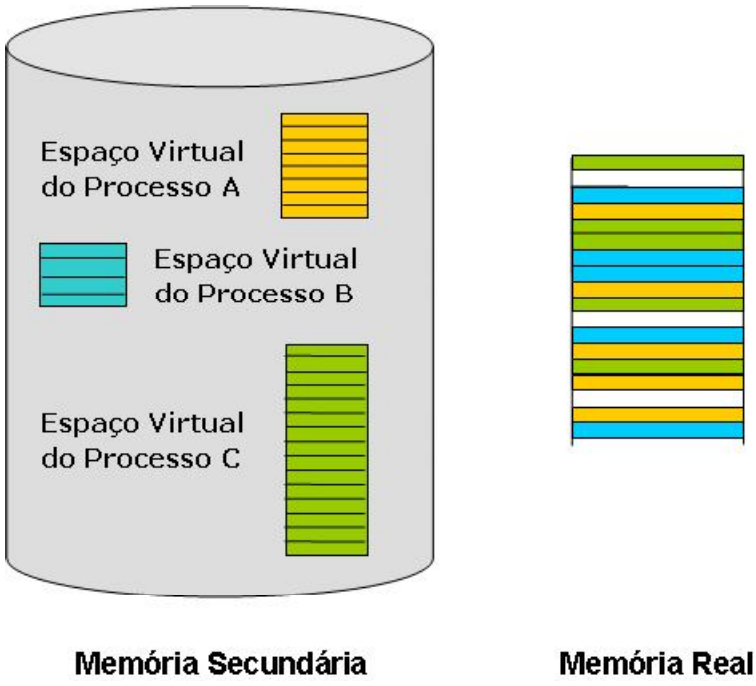


Figura 5.6 – Memória Virtual

O endereçamento virtual permite ultrapassar algumas das limitações referidas nos esquemas anteriores, uma vez que o espaço de endereçamento colocado à disposição do programador é muito maior que a dimensão da memória física. os endereços virtuais gerados

pelos programas são traduzidos em endereços reais pelo hardware, durante a sua execução.

Os endereços virtuais recorrem a um formato bloco:deslocamento. No sistema operativo existe uma tabela que contém endereço físico correspondente a cada bloco. A conversão dos endereços virtuais em endereços reais recorre a esta tabela, para obter o endereço físico do bloco e o somar ao deslocamento que é indicado no endereço virtual.

De forma idêntica à partição em memória real, a partição em memória virtual pode ser de:

- **dimensão variável**: onde a organização da memória em blocos de tamanho variável recebe o nome de **segmentação**; ou
- **dimensão fixa**: onde a organização em blocos de tamanho fixo é designada de **paginação**.

Numa memória paginada um programa é dividido em secções lógicas (módulos, rotinas ou código, dados, pilha), e cada secção é mapeada num segmento virtual. O dispositivo de conversão de endereços da UGM soma o valor do segmento com o endereço base da tabela de segmentos, verifica se o segmento está presente em memória e se os direitos de acesso associados são adequados. Posteriormente compara o deslocamento com a dimensão máxima do segmento e, se o endereço for válido, soma o endereço base do segmento (indicado pela tabela) com o deslocamento dentro do segmento. Certas entradas da tabela de segmentos podem ser mantidas em cache para acelerar o processo de conversão de endereços. Em certas arquitecturas, a cache é composta por registos para os segmentos lógicos mais importantes (código, dados, pilha e um extra de uso genérico).

Quando ocorre um erro na conversão de endereços, que significa uma situação de falha de código ou dados em memória, é gerada uma excepção. As excepções geradas pela UGM distinguem-se das interrupções normais porque a instrução é interrompida a meio, sendo continuada (ou, por vezes, repetida) quando terminar o tratamento da excepção.

A memória paginada oferece ao programador um espaço de endereçamento virtual linear muito maior que a memória física do computador. Os mecanismo de conversão de endereços na paginação e na segmentação são idênticos. Mas, no caso da memória paginada não é necessário guardar a dimensão máxima da página na tabela, porque o tamanho das páginas é constante. Por

outro lado, a soma do deslocamento dentro da página pode ser feita por concatenação. No entanto o funcionamento da UGM é mais complicado porque, como os programas podem ficar divididos por mais do que uma página, pode acontecer que uma instrução fique seccionada entre duas páginas, o que obriga a que todas as instruções tenham de ser recomeçáveis.

Tanto a memória segmentada como a memória paginada recorrem aos mesmos mecanismos de protecção. Cada processo tem uma tabela de segmentos (ou de páginas) que é diferente da dos restantes processos, pelo que não pode aceder à memória de outros processos, e para cada página ou segmento são especificados os direitos de acesso permitidos (leitura, escrita ou execução). A partilha de memória entre processos é conseguida colocando em diferentes tabelas de páginas (ou segmentos) o mesmo endereço base.

A memória segmentada/paginada é uma solução de partição que recorre a segmentos que são paginados internamente. O seu objectivo é conjugar as vantagens dos dois métodos básicos. Neste caso o endereço é composto por segmento:página:deslocamento. A entrada na tabela de segmentos, que é idêntica a da segmentação pura, contém o endereço físico da tabela de páginas. Seguidamente a conversão do par página:deslocamento é feita como na paginação.

A utilização de memória virtual baseia-se, igualmente, no emprego de várias técnicas interessantes. Uma das teorias mais fortes é a do "conjunto de trabalho", que assume que um programa ou processo não acede à totalidade do seu espaço de endereçamento em cada momento. Em vez disso considera que existe um conjunto de posições activas que forma o "conjunto de trabalho". Este pressuposto relaciona-se com outro, que reflecte o facto de que os programas exibem um fenómeno chamado "localidade", o que significa que, normalmente, os programas tendem a usar as instruções que estão próximas da posição da instrução que está actualmente em execução. Se se conseguir que as páginas ou segmentos que contêm o conjunto de trabalho permaneçam sempre na RAM, então o programa terá um desempenho muito melhor.

A Figura 5.7 ilustra as diferenças entre as diversas abordagens adoptadas pelos sistemas operativos para realizar a partição de memória física dos computadores.

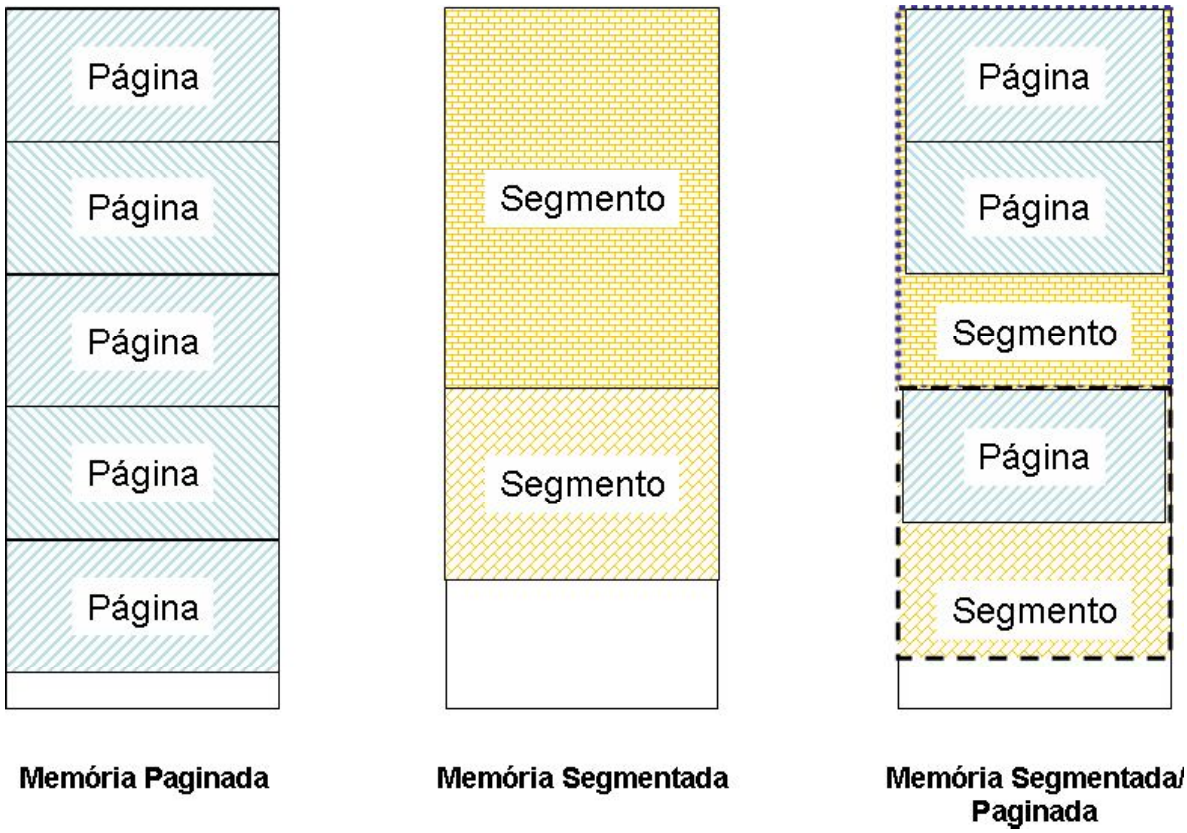


Figura 5.7 - Abordagens usadas para partição de memória física

5.4.1 Algoritmos de Gestão de Memória

Os algoritmos de gestão de memória determinam as decisões tomadas pelo sistema operativo respeitantes à afectação, transferência e substituição de blocos de memória.

O algoritmo de **afectação de páginas** é trivial, pois qualquer página livre pode ser utilizada. As páginas livres são mantidas numa fila gerida em FIFO.

A **afectação de segmentos** é mais complicada e baseia-se numa fila de segmentos livres ordenada por endereços e dimensão do segmento. A lista é percorrida à procura de um segmento livre de dimensão igual ou superior à pretendida. Se for maior é partido em dois, uma parte é devolvida e a outra é novamente inserida na lista de blocos livres. Quando um bloco é libertado, é associado aos blocos livres que lhe sejam contíguos. Os vários algoritmos diferem na escolha do bloco livre, o que influencia o seu desempenho e o grau de fragmentação que originam. Na afectação de segmentos são usadas as mesmas estratégias que anteriormente foram apresentadas para a multiprogramação em memória real, designadamente, o *best-fit* (que devolve o bloco livre cuja dimensão seja mais próxima por excesso do pretendido), o *worst-fit* (que

escolhe o maior bloco livre), o *first-fit* (que escolhe o primeiro bloco que encontrar igual ou maior ao pretendido), o *next-fit* (que é idêntico ao *first-fit* mas começa a procura onde a anterior terminou) e o *buddy* binário (que devolve sempre blocos de dimensão igual a uma potência de 2, tal que, sendo R a dimensão do pedido, $2^{k-1} < R < 2^k$)

A **transferência de blocos** entre memória física e secundária pode ser feita:

- **a pedido**;
- **por necessidade**; ou
- **por antecipação**.

A **transferência de segmentos** é geralmente feita a pedido. Existe uma área de transferência em disco para onde são copiados os segmentos que não cabem de momento em memória central. Quando um processo está em disco e passa ao estado de executável, o sistema poderá decidir carregá-lo em memória se achar provável que este irá ser eleito em breve para execução. Certos processadores permitem a ocorrência de faltas de segmento, podendo a transferência ser feita por necessidade. O carregamento é efectuado quando o programa acede a um segmento que não está em memória. Um segmento é transferido de memória para disco quando é necessário libertar espaço para satisfazer um pedido de afectação.

A **transferência de páginas** é habitualmente feita por necessidade, quando ocorre uma falta nessa página. Devido à forma aleatória como o programa é subdividido em páginas, é impossível utilizar mecanismos de carregamento a pedido, como no caso da segmentada. No entanto, é possível estabelecer políticas de carregamento por antecipação que consistem em ler em avanço algumas páginas, para diminuir o número de faltas de página. Tal como na segmentação, uma página é transferida para disco sempre que há necessidade de libertar espaço para satisfazer um pedido de afectação de memória. Geralmente, o sistema operativo agrupa as páginas numa fila e desencadeia a operação de escrita quando esta atinge um determinado valor, para optimizar os acessos a disco.

A **substituição de blocos** de memória é realizada quando não existe memória livre para satisfazer um pedido de afectação. Neste caso o sistema operativo tem de determinar qual o bloco a retirar.

Em memória paginada, mantêm-se duas listas, a lista de páginas livres e a lista de páginas livres mais modificadas. As primeiras podem ser reutilizadas em qualquer momento, as segundas têm

obrigatoriamente de ser escritas em disco antes de serem re-afectadas. O algoritmo óptimo de substituição escolhe a página que seja acedida mais tarde no tempo. Como tal é impossível de prever, os algoritmos analisam a utilização das páginas num passado recente e, devido ao princípio da localidade de referência, assumem que a utilização se manterá idêntica para fazer uma previsão do futuro próximo.

A substituição de segmentos é mais complicada que a substituição de páginas, pois o segmento ideal para ser retirado pode não ter a dimensão desejada. Habitualmente transfere-se um processo completamente entre memória e disco. A decisão é tomada com base na dimensão do processo, prioridade, estado e tempo de permanência em memória central.

5.4.2 Paginação pura

A gestão de memória por paginação pura é uma abordagem em que o sistema operativo divide dinamicamente os programas em blocos de tamanho fixo (geralmente múltiplos de 1 kByte) os quais são transferidos do disco para a RAM e vice-versa. O processo de transferência de páginas, segmentos ou programas completos entre a RAM e o disco é designado por "transferência de dados" ou "swapping". Na paginação, o principal factor a considerar é o tamanho das páginas, já que se estas forem muito pequenas aumenta o controlo requerido ao sistema operativo para registar quais são os blocos que estão na RAM e quais os que estão no disco, os seus endereços reais, etc., o que provoca muita sobrecarga no sistema. Por outro lado, com páginas grandes, diminui-se a sobrecarga mas aumenta a possibilidade de haver desperdício de memória com processos pequenos. Como tal, deve ser encontrado um ponto de equilíbrio.

Um dos aspectos mais importantes da paginação, assim como de qualquer esquema de memória virtual, é a forma de converter um endereço virtual num endereço real. A Figura 5.8 ilustra este processo de conversão.

Como se pode observar, o endereço virtual v = (b, d) é formado pelo número de página virtual "b" e pelo deslocamento "d". Por exemplo, se o sistema disponibiliza um espaço de endereçamento virtual de 64 kBytes, com páginas de 4 kBytes e a RAM for somente de 32 kBytes, então temos 16 páginas virtuais mas só 8 reais. A tabela de endereços virtuais contém 16 entradas, uma por cada página virtual. Em cada entrada, o registo da tabela de endereços virtuais armazena vários dados: se a página está no disco ou em memória, quem é o dono da página, se a página foi modificada, se é

só de leitura, etc. Mas o dado que nos interessa agora é o número de página real que corresponde à página virtual. Obviamente, das 16 páginas virtuais, somente 8 terão um valor de controlo que diz que a página está colocada na RAM, bem como o endereço real da página, que na Figura 5.8 é designado como b′. Por exemplo, considere-se o exemplo em que, relativamente à página virtual número 14, a tabela diz que esta está colocada na página real 2. Uma vez definida a página real, o deslocamento é-lhe somado, resultando no endereço que se deseja aceder dentro da página pretendida (b′ + d).

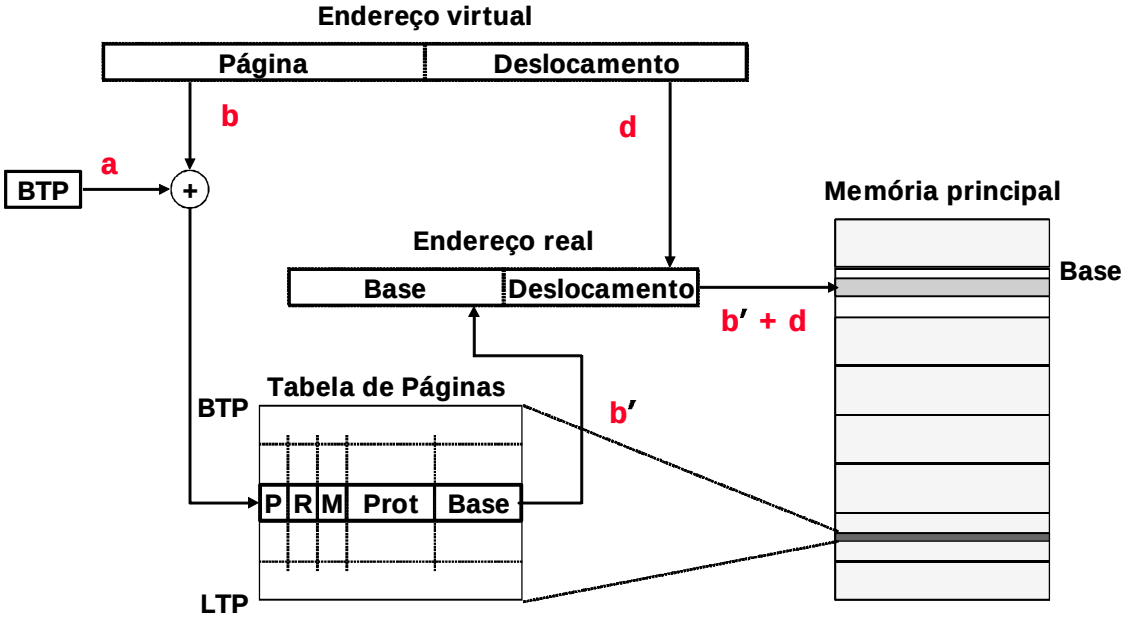


Figura 5.8 – Paginação: Conversão de endereços de memória

Por vezes a tabela de endereços virtuais está situada na própria memória RAM, pelo que também necessita de saber em que endereço começa. Neste caso, existe um registo com o endereço base que, na Figura 5.8, é designado pela letra "a".

Quando é necessária uma página que não está colocada em RAM, ocorre uma "falta de página" (*page fault*), que é enviada para o gestor de memória, que é responsável por realizar a série de passos requeridos para resolver o endereço desejado e fornecer o seu conteúdo a quem o solicitou. Primeiro, detecta-se que a página não está presente e então procura-se na tabela o endereço desta página no disco. Uma vez localizada no disco tenta inserir-se numa página livre da RAM. Se não existirem páginas livres tem que se seleccionar uma para enviar para o disco. Uma vez seleccionada e enviada para o disco, regista-se o seu valor de controlo na tabela de endereços virtuais, indicando que já não está na RAM, enquanto que a página

desejada é colocada na RAM e o seu valor é registado na tabela, indicando que está na RAM. Todo este procedimento é demorado, já que se sabe que os acessos ao disco rígido são da ordem de dezenas de vezes mais lentos que à RAM. No exemplo anterior mencionou-se que, quando se necessita de retirar uma página de RAM para o disco se deve seleccionar a página a transferir. Para realizar esta escolha existem os seguintes algoritmos:

- **FIFO:** este algoritmo escolhe a página que esteja há mais tempo em memória. A realização habitual deste algoritmo mantém uma lista FIFO de páginas em memória. Quando o número de páginas livres desce abaixo de um valor pré-determinado, o processo paginador retira um certo número de páginas e coloca-as na lista de páginas livres ou na lista de páginas modificadas. Quando ocorre uma falta de página, o sistema procura em primeiro lugar se a página em falta está numa destas listas. Se estiver, a página é recolocada no fim das páginas em memória, passando a ser uma página velha. Este algoritmo não é eficiente porque não aproveita nenhuma característica do sistema, no entanto é justo e imparcial.

- **Não usada recentemente:** Selecciona a página que não tenha sido usada (referenciada) no ciclo anterior. Pretende aproveitar o facto da posição no conjunto de trabalho.

- **A menos usada recentemente:** É parecida com a anterior, mas selecciona a página que foi usada há mais tempo, considerando que como já passou bastante tempo sem ser usada é muito provável que continue sem ser usada nos próximos ciclos. Necessita de uma busca exaustiva.

- **A menos frequentemente usada:** Este algoritmo escolhe a página que não foi acedida há mais tempo. Nas implementações práticas deste algoritmo, dividem-se as páginas em grupos etários e escolhe-se uma página do grupo mais antigo. O processo paginador percorre as tabelas de páginas, analisa os bits de referência e modificação, incrementa a idade das páginas que não tenham sido acedidas, e recoloca a zero o contador para as páginas que tenham sido lidas ou escritas. Quando o contador atinge um valor máximo, a página é colocada na lista de páginas livres ou na lista de páginas modificadas. É parecida com a anterior, mas aqui procura-se de forma exaustiva a página que foi menos utilizada que todas as outras.

- **Não frequentemente usada:** Este algoritmo é uma simplificação do algoritmo anterior. Usando os bits de referência e modificação, automaticamente posicionados pela

maioria das UGM, as páginas ficam divididas em quatro grupos, consoante tenham sido acedidas ou não e modificadas ou não nos últimos instantes. A página escolhida é uma qualquer do grupo que não tenha sido lida nem escrita no último intervalo de tempo.

- **Aleatória:** Escolhe uma página qualquer. É justa e imparcial, mas ineficiente.

Análises estatísticas realizadas ao funcionamento de programas revelaram que o espaço de trabalho de um processo, definido como o número de páginas acedidas por um processo durante um certo intervalo de tempo, se mantém aproximadamente constante. Com base neste princípio, o número de páginas que um processo deve ter em memória em cada momento deve ser igual ao seu espaço de trabalho. Se tiver menos páginas, gera um número muito grande de faltas de páginas, o que pode provocar o colapso do sistema. Se tiver mais páginas, elas não serão acedidas. Os sistemas operativos que utilizam este algoritmo só mantêm o processo em memória se este dispuser de um espaço de trabalho mínimo. O espaço de trabalho cresce até um valor máximo, nesse ponto o processo começa a paginar contra si mesmo.

Outro dado interessante da paginação é que já não se requer que os programas permaneçam situados em zonas de memória adjacente, já que as páginas podem estar situadas em qualquer lugar da memória RAM.

5.4.3 Segmentação pura

A segmentação aproveita o facto de que os programas se dividem em partes lógicas, designadamente dados, código e pilha (*stack*). A segmentação atribui partições de memória a cada segmento de um programa tendo por objectivos tornar fácil a partilha de segmentos (por exemplo bibliotecas partilhadas) e a transferência entre memória e os meios de armazenamento secundário.

Por exemplo, na versão Sun OS 3.5 do UNIX, não existiam bibliotecas partilhadas para as ferramentas orientadas para gerir o rato e os menus. Cada vez que um utilizador invocava um programa que utilizava estas ferramentas, tinha que se reservar 1 MByte de memória. A versão 4 daquele sistema operativo passou a fazer a gestão de memória em segmentos, que por sua vez são divididos em páginas. As ferramentas que são usadas frequentemente e que são comuns a todos os utilizadores, são geridas de forma que a primeira vez que um utilizador a invocar, é reservado 1 MByte de memória como antes, mas para o segundo, terceiro e restantes

utilizadores, apenas são necessários 20 kBytes de memória. A poupança é impressionante. Observe-se que na segmentação pura as partições de memória são de tamanho variável, em contraste com as páginas de tamanho fixo na paginação pura. Também significa que a segmentação pura tem uma granularidade menor que a paginação, considerando o tamanho dos segmentos versus o tamanho das páginas.

Uma vez mais, para se compreender melhor a segmentação, pode observar-se a forma como os endereços virtuais são convertidos em endereços reais. Observando a Figura 5.9, verifica-se que a conversão é praticamente igual à utilizada na paginação pura. No entanto agora tem de se ter em consideração que o tamanho dos blocos a controlar pela tabela de conversão são variáveis. Assim, cada entrada na tabela tem de conter as coordenadas de cada segmento a controlar. Do mesmo modo que anteriormente existe um registo base que contém o endereço de início da tabela de segmentos. o endereço virtual é composto por um número de segmento (s) e um deslocamento (d) para localizar um Byte (ou palavra) dentro do dito segmento. É importante que o deslocamento não seja maior que o tamanho do segmento. O controlo é efectuado de forma simples verificando se o valor de tal endereço é maior que o endereço do início do segmento e menor que o endereço do início somado do tamanho do segmento.

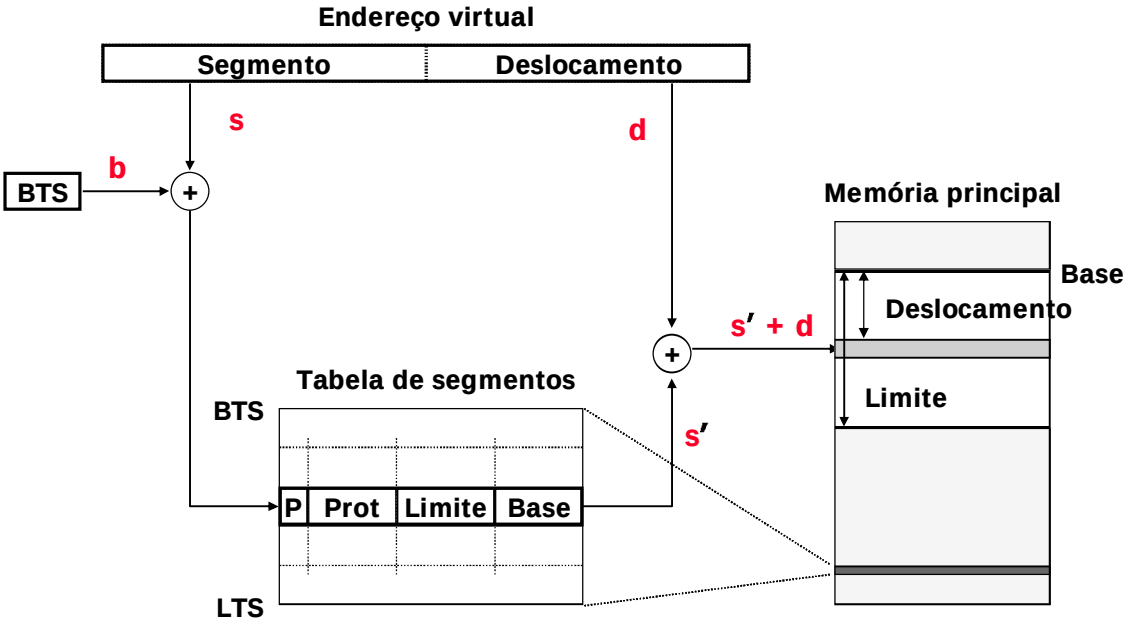


Figura 5.9 – Segmentação: Conversão de endereços de memória

Uma vez dado um endereço virtual v = (s, d), é realizada a operação b + s para identificar o registo (ou entrada da tabela de segmentos) que contém o endereço de início do segmento na memória real, denotado por s'. Conhecido o endereço de início em memória real s', basta encontrar o Byte ou palavra desejada, o qual se obtém somando-se a s' o valor do deslocamento, resultando no endereço real r = s' + d.

Cada entrada na tabela de segmentos tem um formato semelhante ao mostrado na Figura 5.10, onde existem campos que indicam a coordenada, as autorizações, a indicação de presença em RAM e o endereço de início do segmento em memória real.

Bit em RAM	Endereço em disco	Coordenadas	Autorizações	Endereço Real
0 está 1 não está			Leitura Escrita Execução Adição	

Figura 5.10 – Conteúdo das Tabelas de Segmentos

Diversos testes realizados sugeriram que um tamanho de páginas de 1024 Bytes conduz, geralmente, a um desempenho muito aceitável. Intuitivamente poderia parecer que ter páginas de maior tamanho possível faria que o desempenho fosse óptimo mas não é assim, a menos que a página fosse do tamanho do processo total. Com tamanhos grandes de página menores que o processo, o desempenho não é óptimo uma vez que quando se traz para a memória principal uma página devido à falta de um único Byte ou palavra, se estão a trazer muitíssimos mais Bytes do que os desejados.

Um facto importante a considerar nos sistemas que gerem paginação é que quando o processo entra em execução ocorre um grande número de faltas de página, uma vez que nesta ocasião são referenciados muitos endereços novos pela primeira vez, posteriormente o sistema estabiliza e o número de blocos em memória aproxima-se do tamanho do conjunto de trabalho. A Figura 5.11 apresenta a relação entre o tempo médio entre faltas de página e o número de páginas atribuídas a um processo.

Figura 5.11 – Relação Tempo entre Faltas e Número de Páginas por Processo

O tempo entre faltas aumenta com o aumento do número de páginas por processo. A curva do gráfico inflecte num ponto que corresponde à situação em que o processo tem um número de páginas atribuído igual ao que necessita para armazenar o seu conjunto de trabalho. Passado esse ponto, não convém atribuir a um processo mais páginas que as necessárias ao seu conjunto de trabalho, porque o tempo médio entre faltas não melhora. Um fenómeno curioso ocorre quando aumenta o número de páginas por processo e o algoritmo de selecção de páginas é a FIFO, e é designado por "anomalia FIFO" ou "anomalia de Belady". Belady encontrou exemplos em que um sistema com um número de páginas igual a três tinha menos faltas de páginas que um sistema com quatro páginas. A situação descrita é injusta, no entanto, a razão é compreensível. Se se comparar um sistema com 10 páginas com outro com 5, pode constatar-se que, logo à partida, o primeiro sistema terá 5 faltas de página mais que o de 5, porque são necessárias dez faltas para colocar o programa em memória. Durante a execução este diferencial de número de páginas continuará a gerar mais faltas de página.

5.4.4 Sistemas combinados

A paginação e a segmentação puras são métodos de gestão de memória bastante efectivos. Apesar disso, a maioria dos sistemas operativos modernos implementam esquemas combinados, isto é, combinam a paginação e a segmentação. A ideia de combinar estes esquemas permite que se aproveitem os princípios da divisão lógica dos programas (segmentos) com a granularidade das páginas. Desta forma, um processo estará repartido na memória real em pequenas unidades (páginas) cuja ligação são os segmentos. Assim também é possível a partilha de segmentos à medida que as processos com necessidade de memória vão gerando faltas de página.

Do mesmo modo que anteriormente, os endereços virtuais têm de ser convertidos em posições de memória real. A Figuras 5.12 apresenta o modo como é efectuada a conversão de endereços quando a paginação e a segmentação são combinadas.

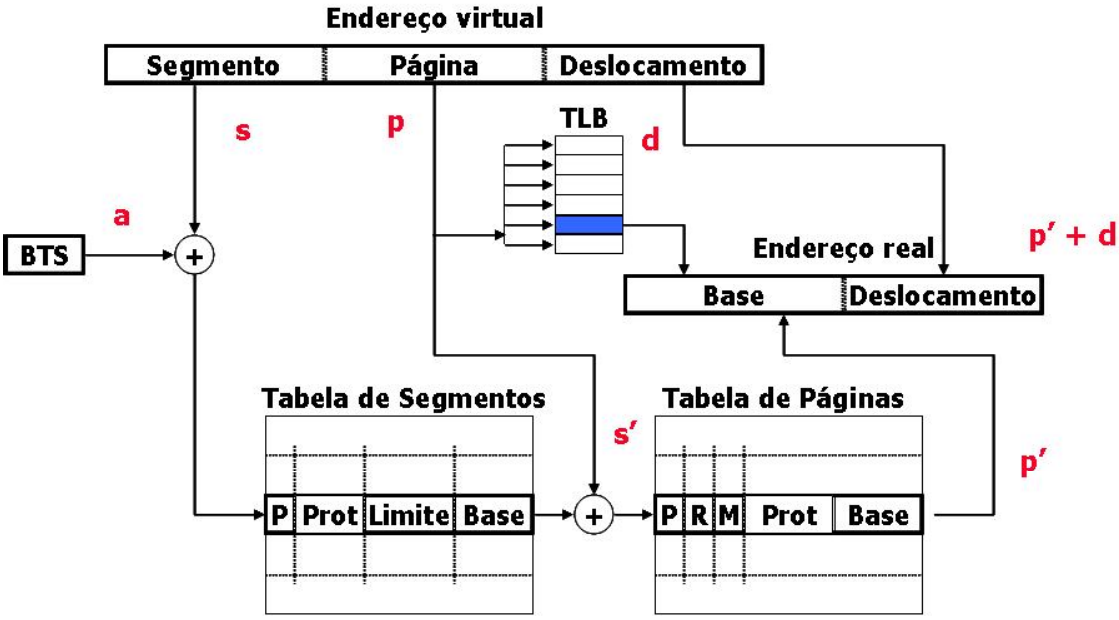


Figura 5.12 – Segmentação-Paginação: Conversão de endereços de memória

O sistema deve dispor de uma tabela de processos (TP) onde, para cada categoria, existe um número de processo e um endereço de uma tabela de segmentos. Ou seja, cada processo passa a ter uma tabela de segmentos. Quando um processo faz uma referência à memória, a tabela de processos TP é consultada para encontrar a tabela de segmentos desse processo. Em cada tabela de segmentos de processo (TSP) existem os números dos segmentos que compõem esse processo. Por cada segmento tem-se um endereço de uma tabela de páginas. Cada tabela de páginas tem os endereços das páginas que pertencem a um único segmento. Por exemplo, o segmento "A" pode ser composto pelas páginas reais "a", "b", "c", "p" e "x". O segmento "B" pode ser composto pelas páginas "f", "g", "j", "w" e "z".

Para traduzir um endereço virtual v = (s, p, d) donde "s" é o segmento, "p" é a página e "d" o deslocamento na página procede-se do seguinte modo: Primeiro refere-se em qual segmento o processo foi colocado e localiza-se a tabela de segmentos desse processo na TP. Com "s" como índice localiza-se uma categoria (registo) na tabela de segmentos desse processo e nessa categoria

está o endereço da tabela de páginas que compõem o segmento. Uma vez na tabela de páginas utiliza-se o valor "p" como índice para encontrar o endereço da página em memória real. É neste endereço de memória real que se encontra o Byte (ou palavra) requerido por meio do valor de "d".

Com este esquema podem ocorrer dois tipos de faltas: falta de página e falta de segmento. Quando é feita referência a um endereço e o segmento que a contém não está na RAM (mesmo que seja parcialmente), é gerada uma falta por falta de segmento, que origina a transferência do meio de armazenamento secundário e a sua introdução na tabela de páginas. Uma vez carregado o segmento, é necessário localizar a página correspondente, mas esta não existe na RAM, pelo que é gerada uma falta de página. A página é transferida do disco e, finalmente, já se pode aceder ao endereço desejado, utilizando o deslocamento do endereço virtual.

A eficiência da conversão de endereços na paginação pura, na segmentação pura e nos esquemas combinados melhora quando se usam memórias associativas para as tabelas de páginas e segmentos, bem como memórias cache para guardar os mapeamentos mais solicitados.

Outro aspecto importante é a estratégia adoptada para inserção das páginas (ou segmentos) na memória RAM. Existem duas estratégias mais comuns: colocação de páginas a pedido e colocação de páginas antecipada. Na estratégia de colocação a pedido as páginas somente são transferidas para a RAM se foram solicitadas, isto é, se ocorrer uma referência a um endereço que cai dentro dessas páginas. A colocação antecipada consiste em tratar de prever quais as páginas que serão solicitadas no futuro imediato e carregá-las de antemão, para que quando sejam acedidas já não ocorram faltas de página. A previsão baseia-se no princípio de que as posições a aceder, e as páginas a colocar por antecipação, devem ser aquelas que têm endereços contíguos ao endereço que acaba de ser referenciado.

Existem algumas variantes a estas estratégias. Por exemplo, o sistema operativo VMS usa um esquema combinado para inserir páginas: quando é feita a referência a um endereço cuja página não está na RAM, é gerada uma falta de página e é colocada essa página junto com algumas páginas adjacentes. Neste caso a página solicitada é colocada a pedido e as adjacentes são transferidas por antecipação.

5.4.5 Comparação dos métodos de gestão de memória virtual

Quando se tentam comparar as diferentes organizações de memória, esta análise tem de ser feita considerando múltiplos factores que incluem o custo do hardware, a complexidade e o desempenho do sistema operativo e as funcionalidades oferecidas ao programador.

As principais vantagens da segmentação são este método adaptar-se bem à estrutura lógica dos programas, ser possível de realizar sobre hardware simples e ser eficiente em operações que actuam sobre um segmento inteiro. As suas principais desvantagens são nunca se conseguir esconder completamente do programador a organização da memória, o tempo de transferência dos segmentos pode ser incomportável, os algoritmos de gestão tornam-se muito complexos para sistemas mais sofisticados, e a dimensão dos segmentos ser limitada.

Considerando a paginação as suas principais desvantagens são permitir que o programador nunca se preocupe com a gestão de memória, ser muito eficiente e a dimensão dos programas deixar de ser um problema. As suas principais desvantagens são obrigar a uma certa complexidade do processador para tratar faltas de página, as operações sobre segmentos lógicos são realizadas de forma pouco elegante, e introduzir uma sobrecarga adicional com o tratamento das faltas de página.

Verifique os seus conhecimentos...

1. Qual a diferença/semelhança entre os termos memória física, memória de massa, memória principal e memória secundária? O que é que as caracteriza?

2. Em que consiste o problema do reposicionamento?

3. O que entende por memória virtual?

4. Diga o que entende por espaço de endereçamento.

5. Indique quais os tipos de partição de memória que conhece e por que nomes são designados.

6. O problema da protecção corresponde a um:
   a. referenciar posições fora do seu espaço de endereços
   b. ler posições de memória de outro processo
   c. escrever em posições de memória de outro processo
   d. todas as anteriores

7. A Unidade de Gestão de Memória é:
   a. um dispositivo de gestão de periféricos que controla a memória física
   b. um dispositivo que detecta a necessidade de transferir dados entre a memória e o disco
   c. um dispositivo de gestão de periféricos que controla a memória de secundária
   d. nenhuma das anteriores

8. Indique se as seguintes frases são verdadeiras ou falsas

	V	F
a. A substituição de páginas é mais complicada que a substituição de segmentos, pois a página ideal para ser retirada pode não ter a dimensão desejada.		
b. O espaço de endereçamento de um processo divide-se em dois níveis: memória física e memória secundária		

	V	F
c. As estratégias de atribuição de espaço de memória dão sempre origem a fragmentação interna		
d. As estratégias de atribuição de espaço de memória dão sempre origem a fragmentação externa		
e. A compactação pretende resolver o problema da fragmentação interna		
f. A utilização de overlays é uma técnica sofisticada muito utilizada nos SO actuais		

Capítulo 6
Comunicação e Gestão de E/S

No presente capítulo será feito um breve enquadramento histórico e tecnológico que conduziu à necessidade do surgimento e evolução de Sistemas Operativos e, em particular, de Sistemas Operativos Distribuídos.

A comunicação entre processos pode ser analisada com base em modelos idênticos a qualquer outro sistema de comunicação ligando interlocutores através de um canal.

O código destinado a gerir as entradas e saídas dos diferentes periféricos num sistema operativo tem uma extensão considerável e é extremamente complexo. Resolve as necessidades de sincronizar, detectar interrupções e disponibilizar chamadas ao sistema para os programadores.

Neste capítulo serão revistos os princípios mais importantes a considerar nesta camada do sistema operativo.

6.1 Comunicação entre Processos

A comunicação entre processos não é diferente de outros sistemas de comunicação. A informação originada num produtor ou emissor é transferida para um consumidor ou receptor usando um canal através do qual são enviadas as mensagens (Figura 6.1).

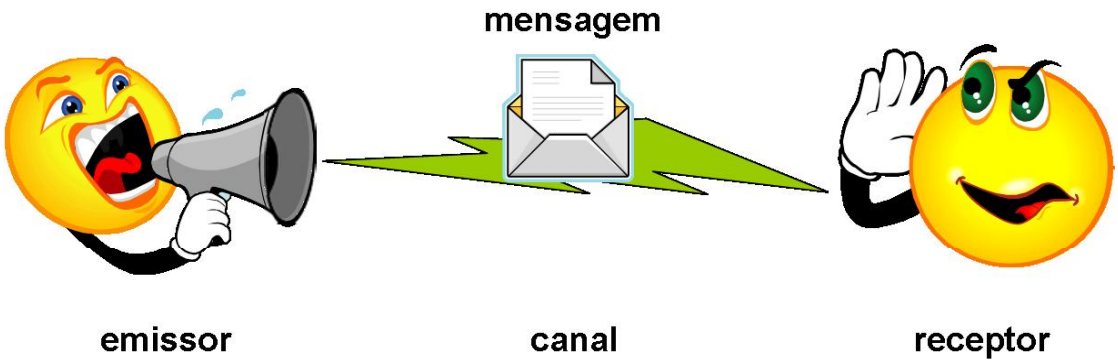


Figura 6.1 – Comunicação entre processos

Na comunicação entre processos o canal pode apresentar diversas variantes:

- identificação directa;
- identificação indirecta;
- capacidade de memorização das mensagens;
- mecanismos de sincronização associados.

A **sincronização** pode recorrer a diversas semânticas para o envio de informação (Figura 6.2):

- **semântica assíncrona**: é a mais vulgar e corresponde ao produtor da mensagem continuar a executar, depois do envio da informação;
- **semântica síncrona**: o produtor fica bloqueado até que o consumidor leia a mensagem;
- **semântica Cliente/Servidor**: o produtor espera até que uma mensagem de resposta lhe seja enviada.

O modelo geral adapta-se aos tipos de interacção mais frequentes, permitindo optimizar os mecanismos de comunicação para estes casos. Os quatro tipos de interacção mais representativos são:

- **Modelo Mestre/Escravo**: a actividade do processo escravo é totalmente controlada pelo mestre (Figura 6.3). Em termos de sincronização o escravo normalmente está à espera que o mestre lhe envie informação. Por seu lado, o mestre tem de se assegurar que o escravo concluiu o tratamento da mensagem anterior, antes de lhe enviar uma nova mensagem;

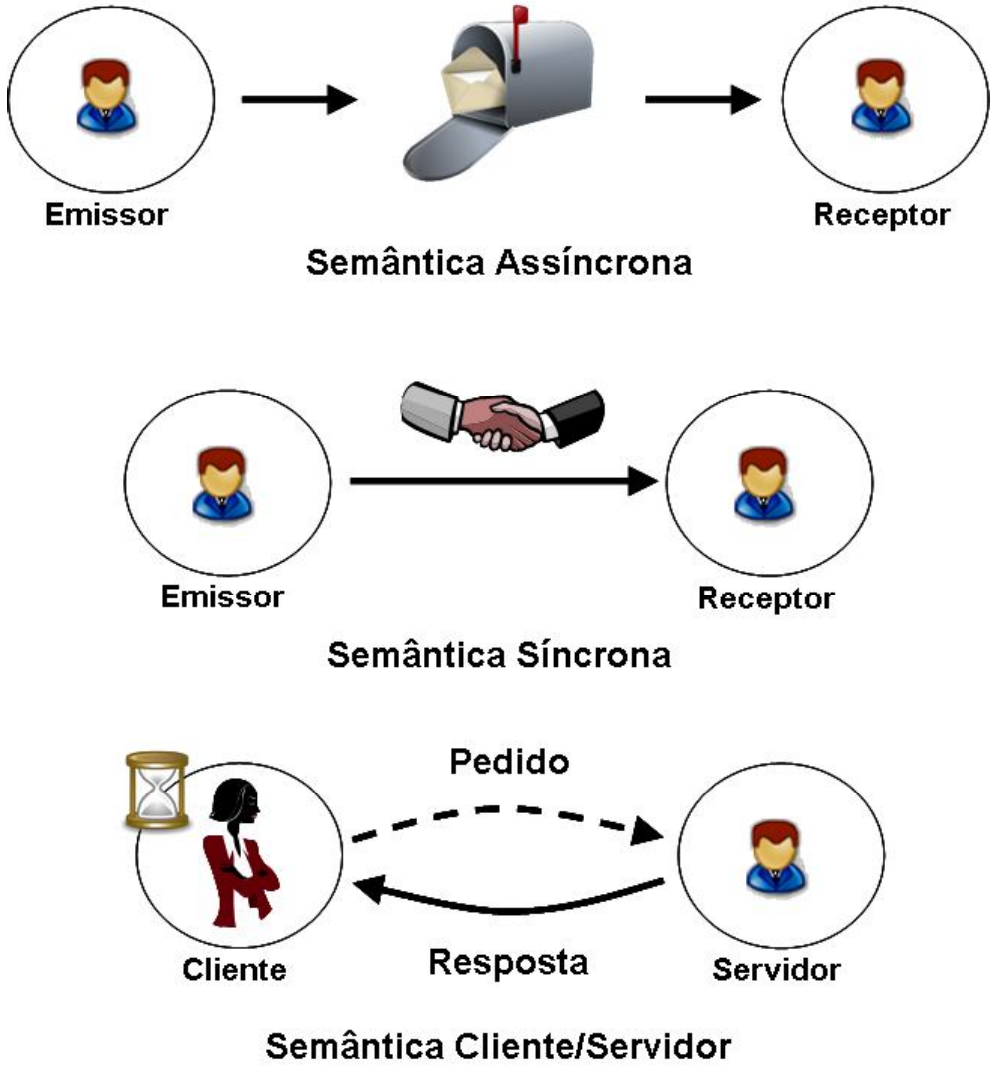


Figura 6.2 – Semânticas para sincronização da transferência de informação

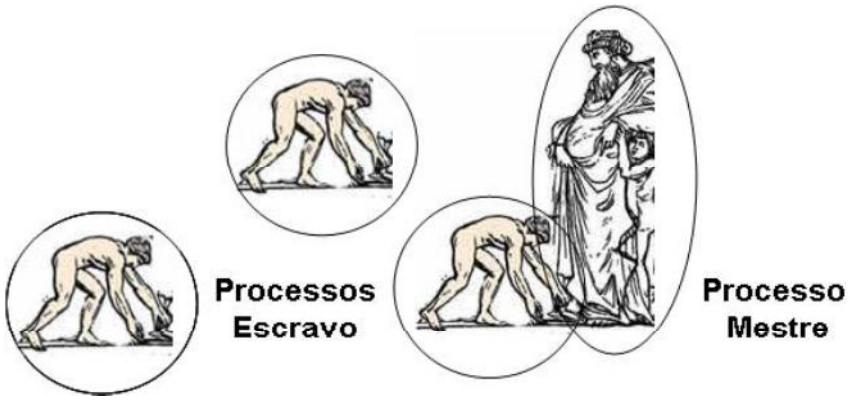


Figura 6.3 – Modelo Mestre/Escravo

- **Modelo de correio**: os processos não têm uma ligação fixa. Quando necessitam de enviar informação associam-se a um canal de comunicação e enviam a mensagem. O canal de comunicação tem de possuir capacidade de memorização de forma a permitir que múltiplos produtores possam enviar informação sem que tal dependa da velocidade de leitura do consumidor (Figura 6.4);

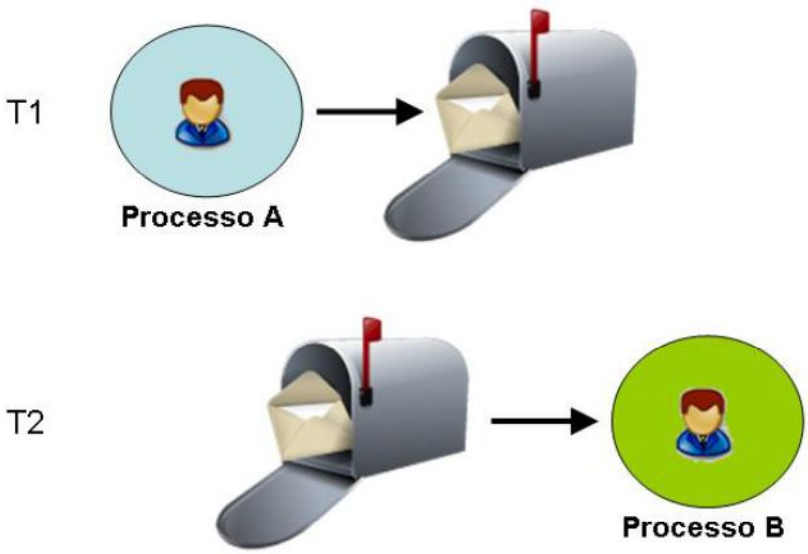


Figura 6.4 – Modelo de Correio

- **Modelo de diálogo**: um processo actua como um servidor recebendo pedidos dos restantes processos para o estabelecimento de um canal dedicado de comunicação (Figura 6.5). O canal é criado temporariamente estabelecendo uma ligação virtual entre os dois processos comunicantes;

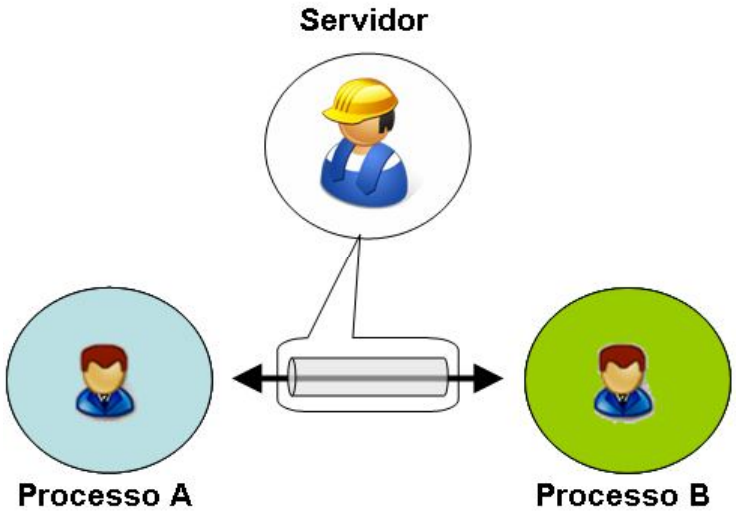


Figura 6.5 – Modelo de Diálogo

- **Difusão**: consiste no envio simultâneo de informação a vários processos receptores (Figura 6.6). A difusão pode ser utilizada para enviar a informação a vários potenciais interessados ou para seleccionar um interlocutor num conjunto de possíveis candidatos.

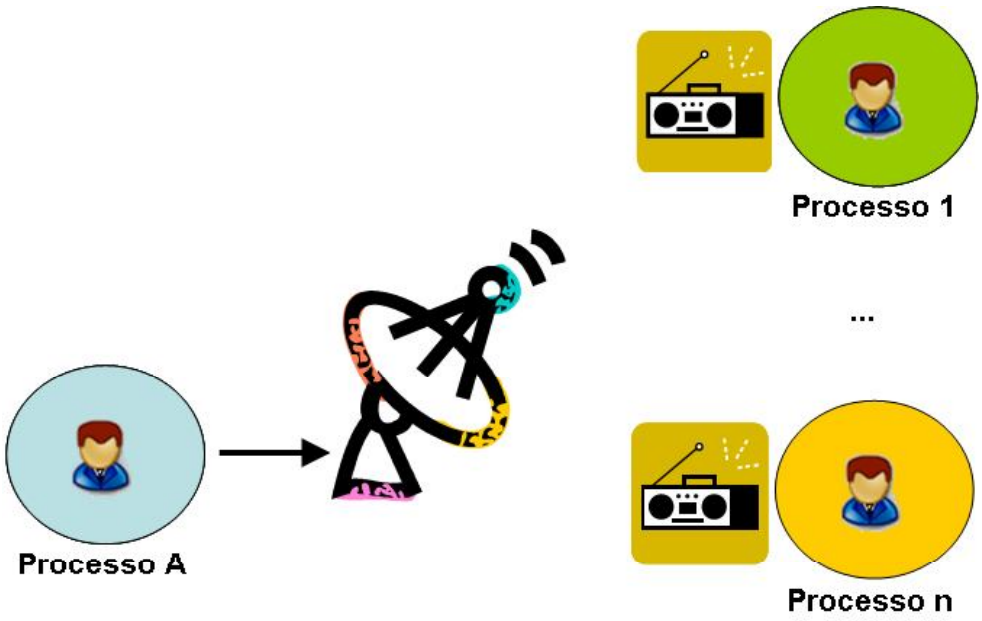


Figura 6.6 – Modelo de Difusão

Estes modelos são suportados no modelo computacional de diversas formas. Apesar de a maioria dos sistemas não implementar a totalidade dos mecanismos, alguns podem ser emulados com recurso a mecanismos mais elementares. A estrutura mais simples consiste na utilização de uma zona de memória partilhada. A zona de memória partilhada faz parte, simultaneamente, do espaço de endereçamento de dois ou mais processos que acedem a esta zona como se das suas variáveis internas se tratasse. A memória partilhada permite a difusão de informação e, quando complementada com os mecanismos de sincronização adequados, a implementação do modelo Mestre/Escravo.

O modelo de correio pode ser implementado sobre objectos do tipo caixa de correio que oferecem a capacidade de memorização temporária da informação no espaço de endereçamento do núcleo.

O mecanismo de diálogo é implementado com base nas primitivas que permitem criar ligações virtuais entre os processos.

6.2 Entradas/Saídas

As funcionalidades do sistema operativo destinadas a controlar as E/S permitem isolar a interacção com periféricos dos pormenores relativos à interface física com os dispositivos.

As E/S são normalmente consideradas como directamente relacionadas com o hardware, mas é possível serem estruturadas de forma a que as partes dependentes dos periféricos físicos fiquem encapsuladas num processo, com o qual os restantes podem interagir através de um dos modelos de comunicação referidos anteriormente. Assim, as E/S podem ser inseridas no modelo de comunicação, isto é, é possível redireccionar as E/S através de mecanismos de comunicação.

Para este efeito são consideradas três entidades:

- **Periféricos virtuais**: são entidades abstractas identificadas de forma lógica, às quais os processos se podem associar;
- **Rotinas de E/S**: são rotinas de interface que desencadeiam as operações de E/S. As rotinas são genéricas e não dependem das características particulares dos dispositivos físicos;
- **Gestores de Periféricos**: são responsáveis por traduzir as operações de E/S genéricas em comandos específicos para os controladores dos dispositivos.

A estrutura interna dos gestores de periféricos pode assumir diversas formas, podendo ser implementados, por exemplo, como:

- **processos independentes**: que são processos sistema com privilégios suficientes para interagir com as interrupções e executar operações de E/S; ou
- **funções do núcleo**: que são rotinas ligadas directamente ao código do núcleo do sistema operativo.

Os processos independentes garantem mais modularidade, o que possibilita a substituição do gestor sem que tal acção interfira com o funcionamento do sistema. No entanto, os gestores directamente no núcleo apresentam um melhor desempenho, uma vez que reduzem o número de comutações entre processos.

Os periféricos podem ser partilhados internamente ou serem controlados por processos servidores. Um exemplo muito comum de utilização de processos servidores para periféricos é o utilizado no controlo de impressoras, onde uma fila interna de pedidos é gerida

pelo processo servidor de acordo com um algoritmo específico (*spooling*).

6.3 Dispositivos de Entradas/Saídas

Os dispositivos de E/S são, geralmente, de dois tipos:

- dispositivos orientados a blocos; e
- dispositivos orientados a caracteres.

Os dispositivos orientados a blocos têm a característica de poderem ser redireccionados, isto é, o programador pode escrever ou ler em qualquer bloco do dispositivo, bastando realizar primeiro uma operação de posicionamento dirigida ao dispositivo. Os dispositivos mais comuns orientados a blocos são os discos rígidos, a memória e os discos compactos.

Os dispositivos orientados a caracteres são aqueles que trabalham com sequências de bytes sem terem em conta as suas coordenadas nem nenhum agrupamento em especial. Não são dispositivos direccionáveis. Exemplos destes dispositivos são o teclado e o ecrã.

A classificação anterior não é perfeita, porque existem vários dispositivos que geram entradas ou saídas que não se podem englobar em nenhuma destas categorias. Por exemplo, um relógio que gera impulsos. Não obstante, apesar de existirem alguns periféricos que não se podem categorizar, todos são administrados pelo sistema operativo através de uma componente electrónica/mecânica e de uma parte de software.

6.4 Controladores de Dispositivos

Os controladores de dispositivos (também designados de adaptadores de dispositivos) são a parte electrónica dos periféricos, cujos circuitos podem estar ou não integrados na placa do computador (*motherboard*). Por exemplo, existem controladores de discos que são vendidos separadamente e que são montados em portos de expansão do computador, enquanto outros são integrados pelos fabricantes de computadores na mesma placa de circuito impresso onde é instalada a CPU.

Geralmente, os controladores de dispositivos comunicam com o processador através de interrupções. Estas interrupções viajam pelo

bus do computador e são recebidas pela CPU, a qual por sua vez colocará em execução um programa dedicado a servir o pedido associado ao sinal. Este programa chama-se "gestor de dispositivo" (*device driver*). Por vezes o próprio controlador contém um pequeno programa em memória ROM ou em memória de RAM não volátil que interage com o gestor de dispositivo correspondente no computador. Na Figura 6.7 é apresentado um esquema simples de dispositivos orientados a blocos e outro a caracteres.

Figura 6.7 – Dispositivos de E/S

Por exemplo, o monitor tem um circuito integrado que se encarrega de enviar, através de um cabo série, cadeias de bits que são recebidas num controlador de porto série no computador. Este circuito integrado também se encarrega de ler sequências de bits que agrupa para a sua visualização no ecrã ou para executar algumas funções de controlo. O importante em todos estes dispositivos é que deve existir um mecanismo para sincronizar o envio e a chegada dos dados de forma concorrente.

Para transferir dados ou sinais entre o computador e os controladores, muitas vezes são usados registos ou secções predefinidas da memória do computador. A este esquema chama-se "gestão de E/S mapeada em memória" (memory mapped I/O). Na Tabela 6.1 apresenta-se um exemplo de vectores de interrupção e dos endereços para E/S de um PC.

Tabela 6.1 Endereços de Mapeamento de Dispositivos de E/S

Controlador	Direcção(Hex)	Vector de Interrupção
Relógio	040 - 043	8
Teclado	060 - 063	9
Disco Duro	320 - 32F	13
Impressora	378 - 37F	15
Monitor	380 - 3BF	-
Disco Flexível	3D0 - 3DF	14

6.5 Acesso Directo à Memória (DMA)

O Acesso Directo à Memória foi criado com o objectivo de libertar a CPU de ter de atender os pedidos de alguns controladores de dispositivos. Para se compreender o seu funcionamento considere-se primeiro como trabalha um controlador sem DMA. Quando um processo requer alguns blocos de um dispositivo, envia um sinal ao controlador com o endereço do bloco desejado. O controlador recebe o pedido através do bus. O processo pode ficar à espera da resposta (trabalho síncrono) ou pode realizar outra tarefa (trabalho assíncrono). O controlador recebe o sinal e o endereço do bus, e envia um ou vários sinais ao dispositivo mecânico (caso exista) e espera pelos dados. Quando recebe os dados coloca-os num *buffer* local e envia um sinal à CPU indicando que os dados estão prontos. A CPU recebe esta interrupção e começa a ler byte a byte, ou palavra a palavra, os dados do *buffer* do controlador (através do *device driver*) até terminar a operação.

Como se vê, a CPU gasta vários ciclos a ler os dados desejados. O DMA soluciona esse problema da seguinte forma: quando um processo requer um ou vários blocos de dados, a CPU envia ao controlador o pedido junto com o número de bytes desejados e o endereço onde quer que estes sejam armazenados. O DMA actuará como uma CPU secundária que tem a capacidade de assumir o controlo do bus e indicar à CPU que deve esperar. Quando o controlador tem listados os dados, o DMA verifica se o bus está livre aproveitando esses ciclos para ir lendo os dados do *buffer* do controlador e os escrever na área de memória que a CPU lhe indicou. Quando a escrita dos dados está concluída envia uma interrupção à CPU para que este use os dados. O ganho obtido com o DMA é que a CPU já não é interrompida (apesar da sua execução

poder ser atrasada pelo DMA) poupando assim a "alteração de contexto", uma vez que o DMA aproveita os ciclos em que o bus não está a ser usado pela CPU.

O facto de os controladores disporem de *buffers* internos permite funcionamentos assíncronos, ou seja, os dados provenientes dos dispositivos que controlam são armazenados temporariamente, até que a CPU esteja pronta para os ler.

6.6 Algoritmos de Gestão de Entradas/Saídas

Os algoritmos usados na gestão de E/S seguem quatro princípios básicos:

- o software deve disponibilizar gestão de interrupções;
- o software deve disponibilizar gestão de dispositivos;
- o software deve ser independente dos dispositivos; e
- o software de E/S deve ter um interface com o software de utilizador.

6.6.1 Gestão de interrupções

O primeiro objectivo, referente à gestão de interrupções, é que o programador ou o utilizador não se dêem conta da gestão que ocorre internamente quando um dispositivo está ocupado e há que suspender um processo ou que sincronizar algumas tarefas. Do ponto de vista do processo ou do utilizador, o sistema simplesmente demorou mais ou menos tempo a responder ao seu pedido.

6.6.2 Gestão de dispositivos

O sistema deve disponibilizar os meios de gestão de dispositivos necessários para os periféricos, e ocultar as peculiaridades da gestão interna de cada um deles, tais como o formato da informação, os meios mecânicos, os níveis de voltagem e outros. Por exemplo, se o sistema tem vários tipos diferentes de discos rígidos, para o programador ou para o utilizador não lhe devem importar as diferenças técnicas entre eles, e a gestão devem assegurar o mesmo conjunto de rotinas para leitura e escrita dos dados.

6.6.3 Software independente do dispositivo

Este é um nível de independência superior ao que é oferecido pela gestão de dispositivos. Aqui o sistema operativo deve ser capaz, na medida do possível, de disponibilizar um conjunto de ferramentas para aceder ou programar os periféricos de uma forma consistente. Por exemplo, todos os dispositivos orientados a blocos devem dispor de uma chamada para decidir se se deseja usar *buffers* ou não, ou para posicionar os blocos nos *buffers*.

6.6.4 Software para utilizadores

A maioria das rotinas de E/S trabalham em modo privilegiado, ou são chamadas ao sistema que se ligam aos programas do utilizador formando parte das suas aplicações e que não deixam nenhuma flexibilidade ao utilizador relativamente ao formato dos dados.

No entanto, existem algumas bibliotecas que oferecem ao utilizador algum poder de decisão (por exemplo, a chamada "printf" na linguagem "C").

Outra facilidade oferecida são as áreas de trabalhos nos bastidores (*Simultaneous Peripheral Operation On-Line* [*spool*] areas), como as que são utilizadas para impressão e para correio electrónico.

6.7 Relógios

Os relógios são essenciais para o bom funcionamento de qualquer sistema porque jogam um papel decisivo na sincronização dos processos, na calendarização de trabalhos por lote e na afectação de turnos de execução entre tarefas relevantes.

Geralmente os sistemas contam com dois relógios: um que regista a hora e a data do sistema e que tem um frequência de 50 ou 60 Hz e outro que tem uma frequência de centenas de MHz e que se encarrega de enviar interrupções à CPU de forma periódica. O relógio de maior frequência serve para controlar o tempo de execução dos processos, para despertar os processos que estão a "dormir" e para gerar ou iniciar processos que foram calendarizados.

Para manter a hora e a data do sistema recorre-se, geralmente, a um registo alimentado por uma pilha de grande duração que armazena estes dados. Assim, mesmo que se desligue a energia da máquina, a data permanece. Para gerar processos (verificação do tempo desocupado de um dispositivo, terminação da fracção de

tempo de um processo, etc.), é armazenada um valor num registo (valor QUANTUM) o qual é decrementado a cada ciclo do relógio. Quando o valor deste registo chega a zero é iniciado um processo do SO que executará as operações necessárias (seleccionar um novo processo para entrar em execução, verificar o funcionamento do motor do disco flexível, fazer eco de um caracter do teclado, etc.).

Verifique os seus conhecimentos...

1. Refira quais os elementos envolvidos na comunicação entre processos e quais os modelos de comunicação mais representativos. Descreva dois destes modelos.

2. Considerando os aspectos relativos à sincronização entre processos, quais as semânticas que podem ser utilizadas para o envio de informação?

3. Quais as entidades envolvidas nas Entradas/Saídas e qual o seu papel?

4. Dê exemplos de periféricos e refira como é que os respectivos controladores comunicam com o processador.

5. O que entende por gestão de E/S mapeada em memória?

6. Indique se as seguintes frases são verdadeiras ou falsas

	V	F
a. No modelo Mestre/Escravo de comunicação entre processos, um processo actua como um servidor recebendo pedidos dos restantes processos para o estabelecimento de um canal dedicado de comunicação.		
b. O DMA permite à CPU dedicar mais tempo a atender os pedidos dos controladores de dispositivos		
c. O DMA actua como uma CPU secundária que tem a capacidade de assumir o controlo do bus e indicar à CPU que deve esperar		
d. Os *buffers* internos existentes nos controladores destinam-se a assegurar um funcionamento síncrono na transferência de dados entre os dispositivos e a CPU		

Capítulo 7
Sistemas de Ficheiros

No presente capítulo será feito um breve enquadramento histórico e tecnológico que conduziu à necessidade do surgimento e evolução de Sistemas Operativos e, em particular, de Sistemas Operativos Distribuídos.

O sistema de ficheiros é uma camada fundamental do sistema operativo, pois a informação guardada neste sistema é vital para o funcionamento dos sistemas. Um sistema de ficheiros é uma estrutura de directórios com algum tipo de organização que permite armazenar, criar e eliminar ficheiros em diferentes formatos. Neste capítulo serão revistos conceitos importantes relacionados com os sistemas de ficheiros.

7.1 Ficheiros

Um ficheiro é uma colecção de dados persistentes, e compreende três componentes:

- **nome**: identifica o ficheiro perante o utilizador;
- **descritor de ficheiro**: que o descreve no sistema operativo; e
- **informação** propriamente dita.

Os nomes dos ficheiros são catalogados em directórios. Um sistema de ficheiros é um conjunto de ficheiros, directórios e estruturas de dados auxiliares, autónomos do ponto de vista da administração e do suporte físico. O sistema de gestão de ficheiros do sistema operativo é responsável pela organização e pelo acesso ao sistema de ficheiros.

O acesso a um ficheiro passa por três fases:

- **abertura**;

- **leitura/escrita**;
- **fecho**.

Na abertura fornece-se o nome do ficheiro e obtém-se o identificador de ficheiro aberto, que permite encontrar rapidamente na tabela de ficheiros abertos a cópia do descritor do ficheiro. Leituras e escritas são feitas utilizando este identificador. O fecho do ficheiro serve para remover a referência ao ficheiro da tabela de ficheiros abertos.

O nome de um ficheiro pode ainda conter alguma informação sobre o seu tipo. Habitualmente, as extensões são apenas convenções mantidas pelos utilizadores e pelos utilitários que empregam os ficheiros.

A protecção no sistema de ficheiros é muito importante, servindo de base a quase todos os mecanismos de protecção do sistema. O sistema de gestão de ficheiros guarda em disco, para cada ficheiro, o número de utilizador do dono do ficheiro e os acessos permitidos aos vários utilizadores. Em cada acesso é verificado se o utilizador tem o direito de acesso correspondente. Os direitos de acesso podem ser guardados em listas de acesso, que especificam, para cada utilizador, quais os acessos permitidos. Uma simplificação consiste em agrupar os utilizadores por grupos e especificar apenas os direitos para o dono, para o grupo e para outros, que não sejam o dono nem pertençam ao seu grupo.

7.2 Armazenamento Físico de Dados

Num sistema computacional é evidente que existe a necessidade de os utilizadores e as aplicações armazenarem os dados de alguma forma, umas vezes por períodos grandes, outras vezes por breves instantes. Cada aplicação e cada utilizador deve ter certos direitos relativamente aos seus dados, como seja a capacidade de os criar e de os eliminar, ou de os mudar de lugar. Por outro lado deve também ser assegurada a sua privacidade relativamente a outros utilizadores ou aplicações.

O sistema de ficheiros do sistema operativo deve encarregar-se destas questões, para além de estabelecer o formato físico a usar no armazenamento dos dados em unidades de memória secundária ou memória de massa. Os dispositivos de memória secundária mais importantes são os discos magnéticos, disquetes, discos ópticos e bandas magnéticas. O sistema de ficheiros reside normalmente sobre os discos magnéticos, os outros tipos de dispositivos são

utilizados, sobretudo, para operações auxiliares, tais como salvaguarda (backup).

Os discos rígidos, flexíveis e unidades de disco óptico têm todos em comum serem animados de um movimento rotativo e que a informação é lida e escrita em pistas concêntricas por cabeças de leitura e escrita que se movem segundo os raios do(s) prato(s) do disco. Cada pista é dividida em sectores. O conjunto de pistas com o mesmo raio chama-se cilindro. Qualquer pista do mesmo cilindro pode ser lida ou escrita sem movimentar as cabeças. A Figura 7.1 ilustra a organização da superfície de um disco em sectores e pistas, e mostra a imagem de um disco e da respectiva cabeça de leitura e escrita.

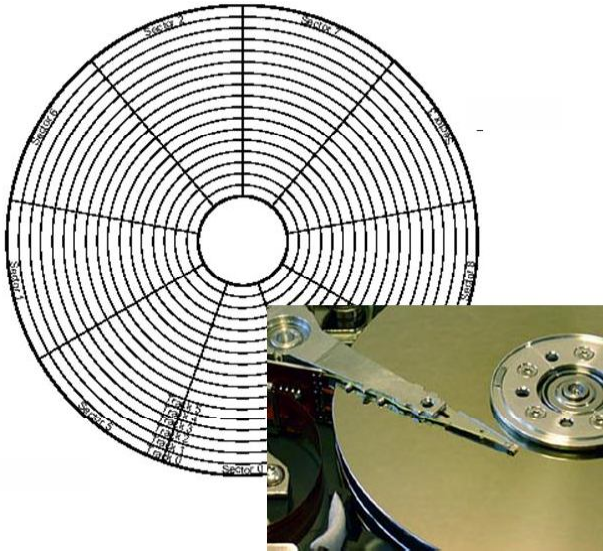


Figura 7.1 - Os discos são animados de um movimento rotativo e a informação é organizada em pistas concêntricas e é lida e escrita por cabeças que se movem segundo os raios dos pratos

Nas unidades que permitem operações de escrita, os dados são gravados nos sectores das pistas modificando o estado das superfícies por acção das cabeças.

O tempo que uma cabeça demora a ir de uma pista a outra chama-se tempo de busca e depende da distância entre a posição actual e a pista pretendida. O tempo que demora uma cabeça a ir do sector actual até ao sector desejado chama-se tempo de latência e depende da distância entre sectores e da velocidade de rotação do disco. O tempo médio de acesso é dado pelos valores médios das grandezas anteriores, sendo habitualmente utilizado para caracterizar a velocidade de um disco. O impacto que as operações

de leitura e escrita têm sobre o sistema é determinado pela tecnologia usada nos pratos e nas cabeças e pela forma de resolver os acessos de leitura e escrita, isto é, nos algoritmos de planeamento.

A optimização mais importante que se pode fazer num sistema de ficheiros é diminuir o número de acessos a disco. As leituras e escritas são feitas através de uma *cache* que mantém os últimos blocos lidos. Periodicamente, os blocos modificados na *cache* são escritos em disco.

A segunda prioridade na optimização dos acessos a disco é a diminuição do tempo de posicionamento. Utilizam-se diversos algoritmos que reordenam os pedidos de leitura e escrita consoante o cilindro a que se destinam.

Outra optimização possível é a diminuição do tempo de latência. Os controladores mais sofisticados lêem pistas inteiras para uma *cache* local. Outro método é ordenar os sectores físicos segundo um factor de entrelaçamento, de forma a permitir ler ou escrever vários sectores logicamente contíguos na mesma revolução do disco.

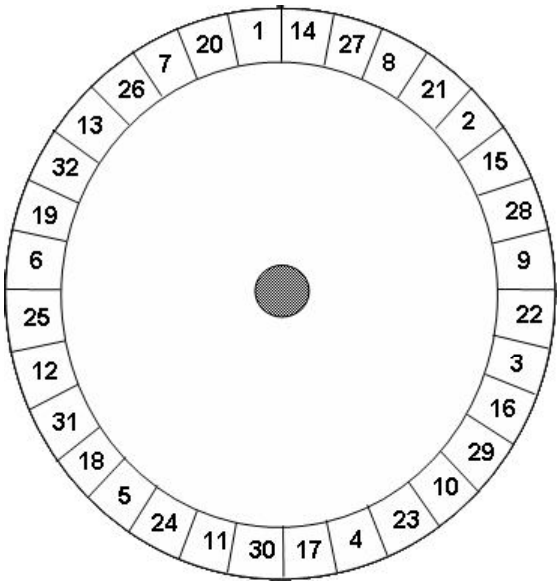


Figura 7.2 - Sectores ordenados com um factor de entrelaçamento

Os algoritmos de optimização dos acessos a disco encarregam-se de registar os pedidos de acesso e de lhes dar resposta num tempo razoável. Os algoritmos mais comuns para esta tarefa são:

- **ordenação por ordem de chegada (FIFO)**: Os acessos são encadeados de acordo com a ordem de chegada e é nessa

mesma ordem que os dados são lidos ou escritos. A vantagem deste algoritmo é a sua simplicidade e não causar sobrecarga. A principal desvantagem é que não aproveita de forma alguma as característica dos acessos, de forma que o mais provável é que o braço do disco se mova de forma muito ineficiente, já que os acessos podem ter endereços no disco umas mais afastadas do que outras. Por exemplo, se se estão a servir pedidos às pistas 6, 10, 8, 21 e 5, os acessos vão ocorrer nessa mesma ordem.

- **por menor deslocamento em relação à posição actual**: Neste algoritmo os acessos são ordenados de acordo com a posição actual da cabeça, servindo primeiro os acessos mais próximos e reduzindo, assim, o movimento do braço, o que constitui a principal vantagem deste algoritmo. A desvantagem é que podem haver pedidos que ficam indefinidamente à espera, caso estejam sempre a entrar pedidos de acesso para endereços próximos uns dos outros mas muito afastados dos endereços dos ditos pedidos. A sequência de pedidos de acesso 6, 10, 8, 21 e 5 será satisfeita na ordem 6, 5, 8, 10 e 21.

- **algoritmo do elevador**: Neste algoritmo o braço movimenta-se sempre do perímetro do disco para o centro e vice-versa, resolvendo os acessos que existam em endereços que estejam no seu caminho. Neste caso os acessos 6, 10, 8, 21 e 5 podem ocorrer na ordem 6, 10, 21, 8 e 5, se se considerar que a posição actual é 6 e que a cabeça se está a deslocar para as pistas de maior endereço (para a periferia, por exemplo). Deste modo no caminho passa para a pista 10, depois para a 21. Invertendo o sentido de movimentação, o braço resolverá os restantes acessos encontrando primeiro a pista 8 e depois a 5. A vantagem deste algoritmo é que o braço se moverá muito menos que na FIFO e evita esperas indefinidas. A sua desvantagem é que não é justo, uma vez que não serve os pedidos na ordem em que chegaram, para além de que os acessos nos extremos interior e exterior terão um tempo de resposta um pouco maior.

- **algoritmo do elevador e elevador circular**: É uma variante do algoritmo anterior, com a única diferença que ao chegar à parte central, o braço regressa ao exterior sem resolver nenhum pedido, o que permite tempos de resposta individuais mais próximos do tempo médio de todos os acessos, independentemente de os endereços estarem perto do centro ou do exterior.

A Figura 7.3 ilustra o funcionamento dos algoritmos de optimização dos acessos a disco.

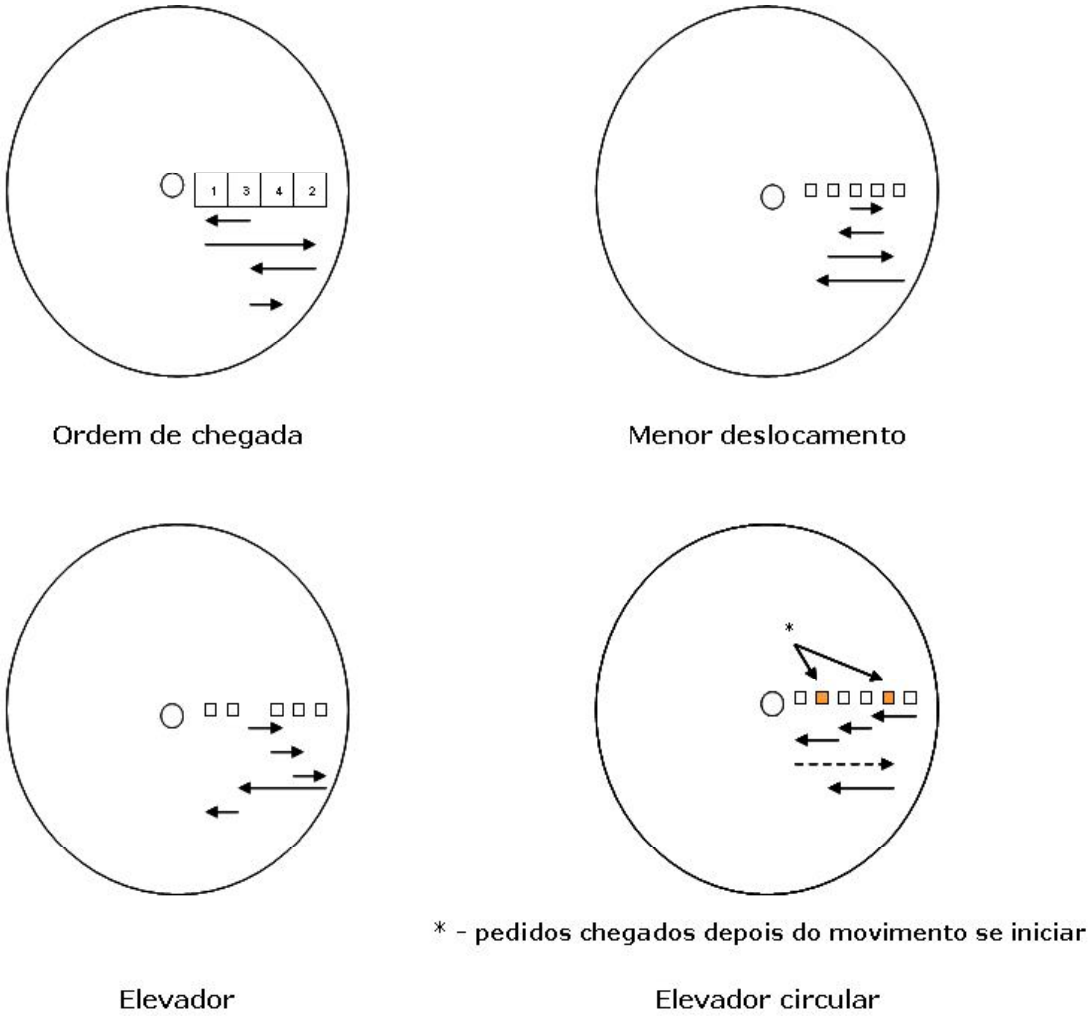


Figura 7.3 - Algoritmos de acesso a disco

7.2.2. Afectação do espaço de armazenamento

Os ficheiros são organizados hierarquicamente, na forma de uma árvore invertida (Figura 7.4). Os nós intermédios designam-se directórios e contêm os seus descendentes, que podem ser outros directórios ou ficheiros. O nome de um ficheiro é um caminho de acesso na árvore.

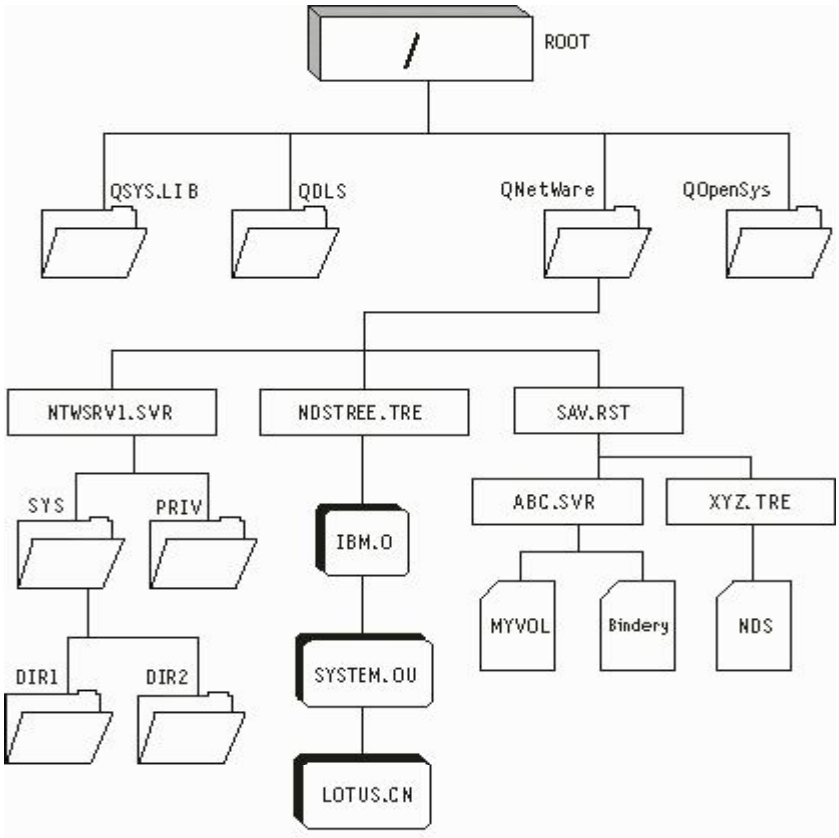


Figura 7.4 - Organização hierárquica dos ficheiros

Um directório é uma tabela que associa nomes a descritores de ficheiros.

Um ficheiro é descrito internamente pelo seu descritor, que armazena o tipo do ficheiro, as datas de criação e de acesso, a dimensão e o conjunto de blocos em disco que compõem o ficheiro.

O sistema de ficheiros tem de se encarregar de localizar espaço livre nos meios de armazenamento para guardar novos ficheiros e para, depois de eliminar os ficheiros existentes, identificar ou ampliar esses espaços. Os directórios contêm a lista de ficheiros criados e, para cada ficheiro, uma série de endereços que contêm os dados dos mesmos. Para atribuir espaço aos ficheiros existem três métodos gerais que se descrevem a seguir.

- **Afectação contígua:** Nesta abordagem os ficheiros são guardados em endereços consecutivos do disco. Os directórios contêm os nomes dos ficheiros, o endereço do bloco inicial de cada ficheiro, bem como o tamanho total do mesmo. Por exemplo, se um ficheiro começa no sector 17 e mede 10 blocos, quando o ficheiro for acedido, o braço mover-se-á

inicialmente para o bloco 17 e lerá até ao bloco 27. Se o ficheiro é eliminado e depois criado outro mais pequeno, ficarão espaços desperdiçados entre ficheiros úteis, fenómeno que se designa por fragmentação externa.

- **Afectação encadeada:** Nesta abordagem os ficheiros podem ficar partidos em blocos. Neste caso os directórios contêm os nomes dos ficheiros e, para cada um deles, o endereço do bloco inicial que compõe o ficheiro. Quando um ficheiro é lido, o braço lê o endereço inicial onde se encontram os dados iniciais bem como o endereço do bloco seguinte e assim sucessivamente. Usando esta abordagem não é necessário que os blocos permaneçam contíguos e não existe a fragmentação externa, mas em cada fracção do ficheiro é desperdiçado espaço com os endereços do mesmo. Por outras palavras, o que se cria no disco é uma lista ligada.

- **Afectação indexada:** Neste esquema o directório guarda um bloco de índices para cada ficheiro, com ponteiros para todos os blocos constituintes, de maneira que o acesso directo é muito agilizado. A diferença aqui é que se têm de sacrificar várias posições de memória para armazenar os ditos ponteiros. Quando se quer ler um ficheiro ou qualquer das suas partes, efectuam-se dois acessos: um ao bloco de índices e outro ao endereço desejado. Este é um esquema excelente para ficheiros grandes mas mau para ficheiros pequenos, porque o tamanho dos blocos destinados aos índices comparado com os que são atribuídos aos dados é comparável.

A gestão de blocos de disco é feita em múltiplos da dimensão dos sectores físicos de disco, chamados blocos. Em memória paginada, um bloco tem a dimensão das páginas de memória física. A unidade de afectação de blocos é o segmento, de dimensão múltipla dos blocos, para aumentar a contiguidade dos ficheiros. Como este método pode gerar uma grande fragmentação, segmentos parcialmente ocupados podem ser utilizados para outros ficheiros. O descritor de cada ficheiro contém os endereços em disco dos diversos segmentos que constituem o ficheiro.

7.2.3. Métodos de acesso ao ficheiros

Os métodos de acesso referem-se às capacidades que o subsistema de ficheiros disponibiliza para se aceder aos dados dentro dos directórios e nos meios de armazenamento em geral. Existem três esquemas gerais de acesso:

- **Acesso sequencial:** É o método mais lento e consiste em ler as posições de um ficheiro uma-a-uma, desde o princípio, até chegar ao registo desejado. É necessário que a ordem lógico dos registos seja igual à ordem física no meio de armazenamento. Este tipo de acesso é usado normalmente em bandas magnéticas e cartuchos.

- **Acesso directo:** Permite aceder a qualquer sector ou registo imediatamente, por meio de chamadas ao sistema como a de *seek*. Este tipo de acesso é rápido e é usado normalmente nos discos rígidos ou nos ficheiros existentes em memória de acesso aleatório.

- **Acesso directo indexado ou por chave:** Este tipo de acesso é útil para grandes volumes de informação. Neste tipo de acesso cada arquivo tem uma tabela de ponteiros. Em cada ponteiro está o endereço de um bloco de índices, o que permite que o ficheiro se expanda através de um espaço enorme. Apesar de as tabelas de índices consumirem uma quantidade importante de recursos, é muito rápido.

7.2.4. Operações suportadas pelo subsistema de ficheiros

Independentemente dos algoritmos de afectação de espaço, dos métodos de acesso e da forma de resolver os acessos de leitura e escrita, o subsistema de ficheiros deve disponibilizar um conjunto de chamadas ao sistema para lidar com os dados e disponibilizar mecanismos de protecção e segurança. As operações básicas que a maioria dos sistemas de ficheiros suportam são:

- **Criar (*create*) :** Permite criar um ficheiro sem dados, com o propósito de reservar o nome do ficheiro e de criar as estruturas básicas para o suportar.

- **Eliminar (*delete*):** Elimina o ficheiro e liberta os blocos para utilização posterior.

- **Abrir (*open*):** Antes de se usar um ficheiro ele tem de ser aberto para que o sistema conheça os seus atributos, tais como o proprietários, a data da última alteração, etc.

- **Fechar (*close*):** Posteriormente a realizar as operações desejadas, o ficheiro deve ser fechado para assegurar a sua integridade e para libertar da memória os recursos para o seu controlo.

- **Ler/Escrever (*read/write*):** Adicionar informação ao ficheiro ou ler um caracter ou uma cadeia de caracteres a partir da posição actual.

- **Concatenar (*append*):** É uma forma restrita da chamada "*write*", que apenas permite adicionar informação no final do ficheiro.

- **Localizar (*seek*):** nos ficheiros de acesso directo permite posicionar o ponteiro de leitura ou escrita num registo aleatório, relativamente ao início ou ao final do ficheiro.

- **Ler atributos:** Permite obter uma estrutura com todos os atributos do ficheiro especificado, tais como as permissões para escrita, eliminação, execução, etc.

- **Alterar atributos:** Permite transferir os atributos de um ficheiro, por exemplo em UNIX, onde todos os dispositivos são geridos como se fossem ficheiros.

- **Alterar nome (*rename*):** Permite alterar o nome e inclusivamente às vezes a posição na organização de directórios do ficheiro especificado.

Os subsistemas de ficheiros também disponibilizam um conjunto de chamadas para actuar sobre os directórios, as mais comuns são criar, eliminar, abrir, fechar, renomear e ler. As suas funcionalidades são óbvias, mas existem também outras operações que não são tão comuns, como sejam "criar uma ligação" ou "destruir a ligação". A operação de criar uma ligação serve para que a partir de diferentes pontos da organização de directórios se possa aceder a um mesmo directório sem necessidade de copiá-lo ou de duplicá-lo. A chamada "destruir a ligação" elimina estas referências, isto é, elimina as ligações e não o directório real.

7.2.5. Algumas funcionalidades dos sistemas de ficheiros

Alguns sistemas de ficheiros disponibilizam ao administrador do sistema ferramentas para lhe facilitar a vida. As mais notáveis são a funcionalidade de partilha de ficheiros e os sistemas de "quotas".

A funcionalidade de partilha de ficheiros refere-se à possibilidade de que as permissões dos ficheiros ou directórios deixem que um grupo de utilizadores possam aceder a diferentes operações de ler, escrever, eliminar, criar, etc. O verdadeiro dono é quem decide quais as permissões que serão conferidas ao grupo e, inclusivamente, a outros utilizadores que não façam parte desse grupo. A funcionalidade de "quotas" refere-se ao controlo que o

sistema de ficheiros faz relativamente ao tamanho do espaço que cada utilizador ocupa do disco duro, considerando um limite máximo. Quando o utilizador excede esse limite, o sistema envia-lhe uma mensagem e nega-lhe a autorização para continuar a escrever, obrigando-o a eliminar alguns ficheiros se quiser armazenar novos ficheiros ou que alguns aumentem de tamanho.

7.3 Sistemas de Ficheiros Isolados

Os sistemas de ficheiros isolados são aqueles que residem num único computador e em que não existe a possibilidade de outros sistemas poderem usar os seus directórios e ficheiros, ainda que estejam numa rede (Figura 7.5). Por exemplo, os ficheiros nos discos rígidos no sistema MS-DOS clássico podem considerar-se um exemplo desta categoria.

Figura 7.5 – Sistema de Ficheiros Isolado

7.4 Sistemas de Ficheiros Partilhados ou de Rede

Nestes sistemas de ficheiros é possível aceder e usar os ficheiros a partir de outros nós numa rede. Geralmente existe um "servidor", que é o computador onde fisicamente reside o sistema de ficheiros, e "clientes" (Figura 7.6), que recorrem ao servidor para aceder aos ficheiros e directórios, como se fossem locais ao cliente. Alguns autores chamam a estes sistemas de ficheiros "sistemas de ficheiros distribuídos".

Os sistemas de ficheiros partilhados em rede mais populares são os disponibilizados pela Netware, o Remote File Sharing (RFS do UNIX),

o Network File System (NFS da Sun Microsystems) e o Andrew File System (AFS). Em geral, o que os servidores disponibilizam é um meio para que os clientes, localmente, realizem pedidos de operações sobre os ficheiros os quais são detectadas por um "driver" ou um "módulo" no núcleo do sistema operativo, o qual comunica com o servidor através da rede executando a operação no servidor. Existem servidores de tipo "stateless" e "non-stateless". Um servidor "stateless" não regista o estado das operações sobre os ficheiros, pelo que é o cliente que se tem de encarregar de todo esse trabalho. A vantagem deste esquema é que se o servidor falha, o cliente não perderá informação já que ela se encontra guardada localmente em memória, de forma que quando o servidor restabelece o seu serviço o cliente prossegue como se nada tivesse acontecido. Com um servidor "non-stateless" isto não é possível.

Figura 7.6 – Sistema de Ficheiros Partilhado, com Servidor de Ficheiros

A protecção sobre as operações é realizada tanto nos clientes como no servidor: se o utilizador quer executar uma operação indevida sobre um ficheiro, receberá um mensagem de erro e possivelmente é enviado um registo ao subsistema de "segurança" para informar o administrador do sistema da dita intenção de violação.

Na prática, o conjunto de permissões que cada utilizador tem sobre os ficheiros é armazenada em estruturas chamadas "listas de acesso" (*access lists*).

7.5 Tendências actuais

Com o aumento das redes de comunicações e o incremento da largura de banda, a proliferação de pacotes que disponibilizam a partilha de ficheiros é comum. Os esquemas mais solicitados na indústria requerem a capacidade de aceder a grandes volumes de informação que residem em grandes servidores, tanto a partir de computadores pessoais como de outros servidores. Uma das solução mais frequentes nas empresas pequenas é usar soluções de tipo Novell Netware a correr num servidor, muitas vezes com características semelhantes às das máquinas que acedem aos ficheiros.

Por vezes as soluções são mais complexas e envolvem ambientes heterogéneos: diferentes sistemas operativos e diferentes arquitecturas. Um dos sistemas de ficheiros mais divulgados em estações de trabalho é o NFS, e praticamente todas as versões de UNIX trazem instalado um cliente e até um servidor deste serviço. É possível assim que uma grande quantidade de computadores pessoais (de 10 a 80) acedam a grandes volumes de informação (da ordem dos GigaBytes) residentes numa única estação de trabalho, e inclusivamente ter a flexibilidade de usar ao mesmo tempo servidores de Novell e NFS. Apesar da referência a estes sistemas, estão disponíveis soluções semelhantes em outros pacotes comerciais. O importante aqui é observar que o mundo está a evoluir para soluções distribuídas, e que está a ocorrer alguma normalização nos sistemas.

Verifique os seus conhecimentos...

1. Indique os tipos de acesso a ficheiros previstos nos sistemas operativos.

2. Indique quais as componentes que compõem o tempo de acesso a discos. Refira as principais formas de optimização dos acessos.

3. O acesso sequencial a ficheiros:
   a. É usado normalmente em bandas magnéticas e cartuchos.
   b. É o método mais lento.
   c. Consiste em ler as posições de um ficheiro uma-a-uma, desde o princípio, até chegar ao registo desejado.
   d. Todas as anteriores

4. O acesso a ficheiros existentes em suportes físicos:
   a. É sempre feita em dispositivos animados de um movimento rotativo, onde a informação é lida e escrita em pistas concêntricas por cabeças de leitura e escrita.
   b. É sempre feita em dispositivos onde a informação é lida e escrita por cabeças de leitura e escrita.
   c. Na maior parte dos casos é feita em dispositivos animados de um movimento rotativo, onde a informação é lida e escrita em pistas concêntricas por cabeças de leitura e escrita.
   d. Nenhuma das anteriores

5. Relativamente ao sistema de ficheiros:
   a. Os nomes dos ficheiros são organizados hierarquicamente, na forma de uma árvore invertida.
   b. Os ficheiros intermédios designam-se por directórios e contêm os seus descendentes, que podem ser outros directórios ou ficheiros.
   c. O nome de um ficheiro é um caminho de acesso dentro do directório e os nomes absolutos especificam o caminho de acesso a partir da raiz do directório
   d. todas as anteriores

6. Os sistemas de ficheiros que residem num computador e em que os directórios e ficheiros não podem ser usados por outros computadores são designados de:
   a. sistemas de ficheiros isolados.
   b. sistemas de ficheiros partilhados.
   c. sistemas de ficheiros em rede.
   d. Nenhum dos anteriores.

7. Indique se as seguintes frases são verdadeiras ou falsas

	V	F
a. Ordenar os sectores físicos segundo um factor de entrelaçamento permite diminuir o tempo de latência		
b. Utilizar o algoritmo do elevador permite diminuir o tempo de latência		
c. O tempo de posicionamento é composto pelo tempo de acesso ao disco, pelo tempo de latência e pelo tempo de transferência.		
d. O subsistema de ficheiros deve disponibilizar um conjunto de chamadas ao sistema para lidar com os dados e disponibilizar mecanismos de protecção e segurança.		

Capítulo 8
Princípios dos sistemas distribuídos

Os sistemas distribuídos baseiam-se em princípios básicos de transparência, eficiência, flexibilidade, escalabilidade e fiabilidade.

Apesar de alguns destes princípios serem algo contraditórios entre si, os sistemas distribuídos e os respectivos sistemas operativos têm de assegurar que todos estes princípios são cumpridos de uma forma aceitável.

No presente capítulo serão apresentados alguns dos conceitos subjacentes aos princípios que se visam assegurar com os sistemas distribuídos.

Os sistemas distribuídos baseiam-se em princípios básicos de transparência, eficiência, flexibilidade, escalabilidade e fiabilidade. Assim sendo um dos objectivos a atingir no desenvolvimento de sistemas operativos distribuídos é assegurar que todos estes princípios são cumpridos de uma forma aceitável.

8.1 Transparência

Um sistema transparente deve funcionar de forma semelhante em todos os pontos da rede, independentemente da localização do utilizador. Um sistema operativo distribuído deve dispor de mecanismos que ocultem a natureza distribuída do sistema. Os utilizadores devem poder trabalhar no sistema como se estivessem a utilizar uma única máquina.

Figura 8.1 – Transparência no acesso aos recursos

Em termos gerais, um sistema operativo distribuído tem de garantir o controlo da unicidade dos recursos e a resolução dos problemas de concorrência. Considerando, por exemplo, o acesso a ficheiros, se o sistema dispuser de várias cópias de um ficheiro, estas devem aparecer ao utilizador como um único ficheiro. Para o efeito, o sistema operativo tem de dispor dos mecanismos que permitam o controlo as cópias e a sua actualização em caso de alteração.

8.2 Eficiência

Um sistema distribuído eficiente deve apresentar uma rapidez muito superior à dos sistemas actuais. Uma vez mais a questão está relacionada com o paralelismo.

A eficiência do sistema é conseguida distribuindo as tarefas a executar pelos processadores mais rápidos que estiverem disponíveis em cada momento.

Figura 8.2 – Um sistema distribuído eficiente deve distribuir as tarefas pelos recursos disponíveis

A rapidez com que um processador vai realizar uma tarefa é algo difícil de avaliar, dependendo de muitas questões concretas, como

sejam a velocidade do processador ou a posição do processador, dos dados, dos dispositivos, etc.. (tem de se evitar, por exemplo, que um trabalho de impressão seja enviado para um computador que não disponha de uma impressora local).

8.3 Flexibilidade

O desenho de um sistema operativo distribuído deve prever alterações e actualizações que melhorem o funcionamento do sistema. Como se viu anteriormente existem duas alternativas na arquitectura do núcleo do sistema operativo: o núcleo monolítico e o micronúcleo.

A arquitectura de micronúcleo baseia-se numa programação altamente modular, que tem um tamanho muito menor que o núcleo monolítico. O refinamento e o controlo de erros é mais rápido e simples, e a actualização dos serviços é mais simples e ágil, já que neste caso apenas se torna necessária a recompilação do serviço e não da totalidade do núcleo. No entanto, o rendimento é afectado negativamente.

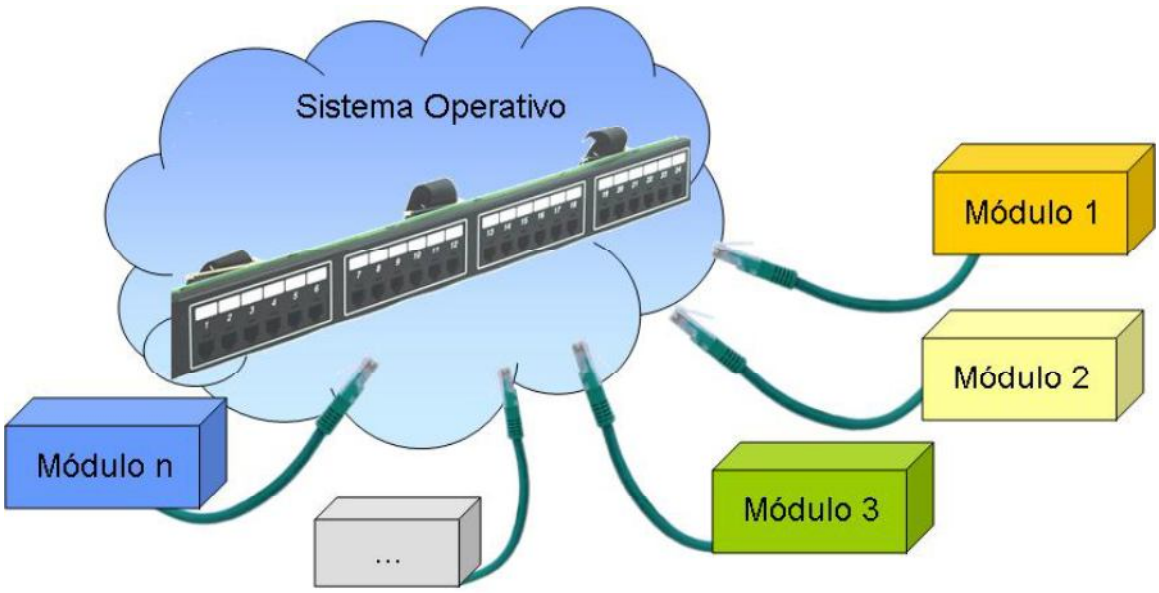


Figura 8.3 – Um sistema distribuído flexível baseia-se numa programação altamente modular

A maioria dos sistemas operativos distribuídos em desenvolvimento na actualidade tendem a adoptar um desenho de micronúcleo. Estes núcleos tendem a conter menos erros e a ser mais fáceis de

implementar e de corrigir. O sistema perde ligeiramente em rendimento, mas esta desvantagem é compensada pelo grande aumento de flexibilidade.

8.4 Escalabilidade

Um sistema operativo distribuído deve funcionar independentemente do número de computadores, do tipo de rede utilizada (LAN ou WAN), das distâncias entre os equipamentos, etc.

Na realidade as soluções válidas para um pequeno número de computadores podem não ser aplicáveis quando o número de computadores é elevado. Do mesmo modo, o tipo de rede condiciona determinantemente o rendimento do sistema, e pode acontecer que a abordagem para um tipo de rede não funcione noutro diferente.

A escalabilidade propõe que qualquer computador individual deverá poder trabalhar tanto isoladamente como ligado a um sistema distribuído, onde existam muitas máquinas.

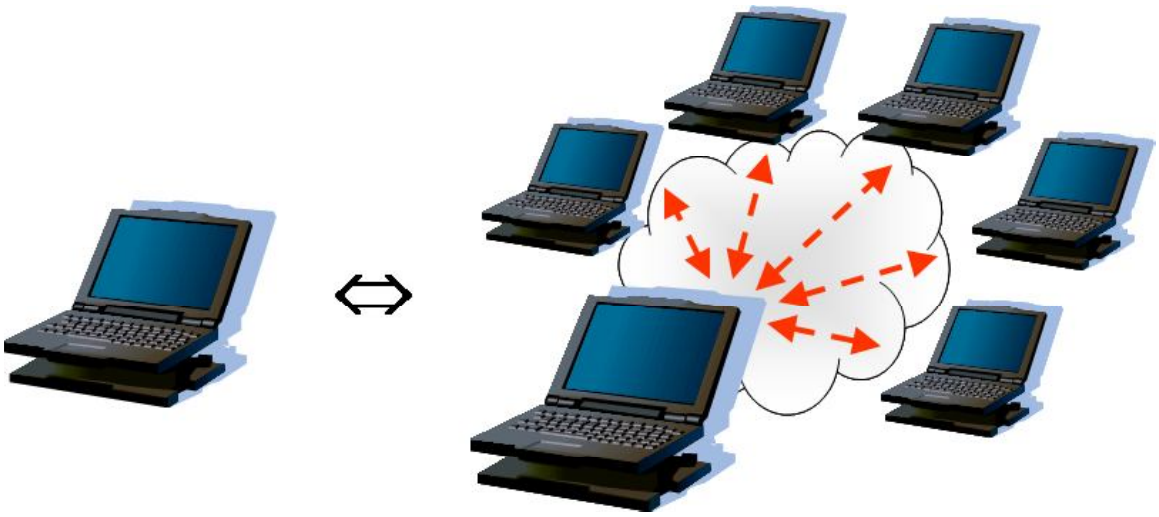


Figura 8.4 – Um sistema operativo distribuído escalável deve ser independente do número de computadores

8.5 Fiabilidade

O funcionamento de um sistema distribuído não depende de nenhuma das máquinas da rede, uma vez que qualquer equipamento pode ser substituído em caso de avaria ou falha.

Figura 8.5 - Um sistema distribuído fiável não depende de nenhuma das máquinas da rede

A fiabilidade é conseguida à custa de redundância. A informação não deve estar armazenada num único servidor de ficheiros, antes em, pelo menos, dois. A redundância dos ficheiros evita que uma falha num servidor bloqueie todo o sistema, já que existe uma cópia idêntica noutro equipamento.

Um tipo de redundância mais complexo refere-se aos processos. As tarefas críticas podem ser realizadas por vários processadores independentes, havendo um processador que realiza a tarefa normalmente, no entanto a responsabilidade da execução passa para outro processador no caso do primeiro falhar.

8.6 Comunicação

A comunicação entre processos em sistemas com um único processador é realizada mediante o uso de memória partilhada entre os processos. Nos sistemas distribuídos, pelo facto de não haver ligação física entre as diferentes memórias dos equipamentos, a comunicação é realizada mediante a transferência de mensagens.

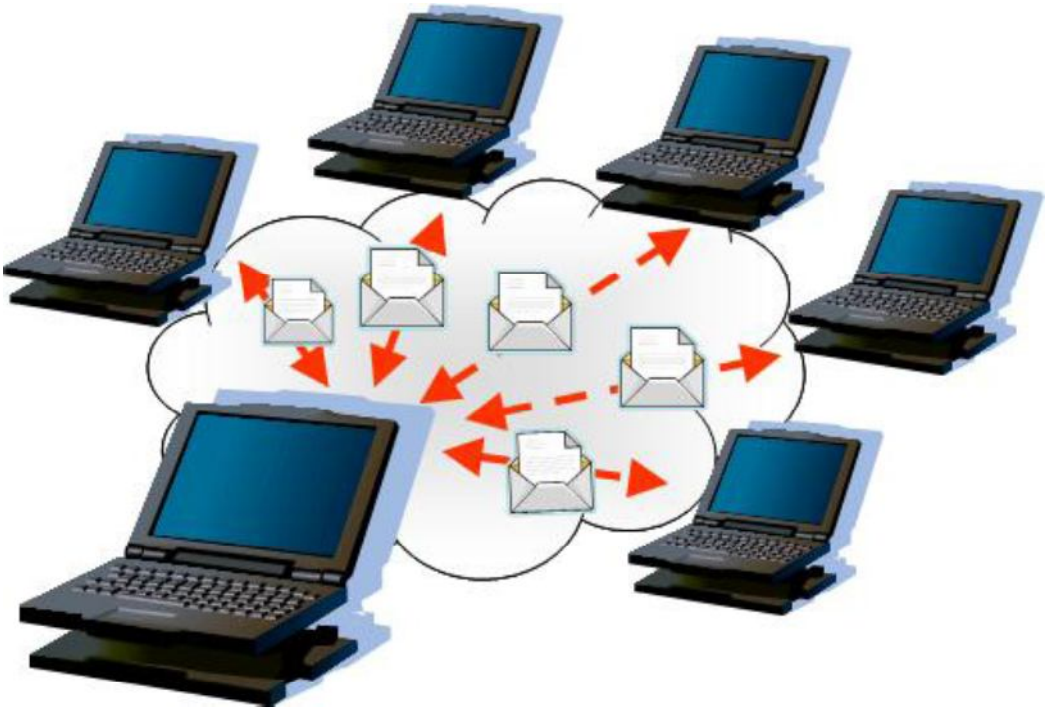


Figura 8.6 – Nos sistemas distribuídos a comunicação é realizada por transferência de mensagens

Verifique os seus conhecimentos...

1. Qual o princípio que se pretende atingir com os mecanismos dos sistemas operativos distribuídos que ocultam a natureza distribuída do sistema?

2. Qual o princípio que se pretende atingir distribuindo as tarefas a pelos processadores mais rápidos que estiverem disponíveis em cada momento?

3. Desenvolver um sistema operativo que preveja alterações e actualizações que melhorem o funcionamento do sistema assegura que princípio básico a atingir pelos sistemas operativos distribuídos?

4. Um sistema modular:
   a. é flexível
   b. tende a conter menos erros
   c. tem menor rendimento
   d. todas os anteriores

5. A fiabilidade é:
   a. com hardware muito caro
   b. usando vários processadores independentes
   c. usando memória partilhada
   d. nenhuma das anteriores

6. A comunicação em sistemas distribuídos:
   a. com hardware muito caro
   b. usando vários processadores independentes
   c. usando memória partilhada
   d. nenhuma das anteriores

7. A redundância dos ficheiros:
   a. evita que uma falha num servidor bloqueie todo o sistema
   b. assegura que existe uma cópia idêntica noutro equipamento
   c. requer mecanismos de transferência do controlo de um controlo para o seu mirror
   d. todas as anteriores

8. Um sistema operativo distribuído:
    a. tem de garantir o controlo da unicidade dos recursos
    b. tem de garantir a resolução dos problemas de concorrência
    c. tem de dispor dos mecanismos que permitam o controlo as cópias
    d. todas as anteriores

Capítulo 9
Caso de Estudo: UNIX

No presente capítulo serão apresentadas sucintamente as principais características do sistema operativo UNIX.

O UNIX é um dos sistemas operativos mais amplamente usados em computadores desde as aplicações pessoais até às grandes aplicações em rede. Existem versões para máquinas uni-processador e multiprocessador. A história deste SO começa em 1969 nos Laboratórios Bell da AT&T associado a um simulador de viagens espaciais no sistema solar. Nos anos 70 e 80 foi muito influenciado pelos círculos académicos que o expandiram, criaram novas versões (das quais se destaca a BSD da Universidade de Berkeley) e o adoptaram em grande escala.

A história do UNIX é bastante atribulada, tendo dado origem a várias linhas de desenvolvimento, de acordo com as guerras comerciais e jurídicas entre as empresas que se envolveram no desenvolvimento e distribuição do produto. Porventura a versão UNIX mais amplamente distribuída seja o Solaris OS da Sun Microsystems, que actualmente está na sua versão 10. Existem cerca de uma dezena de outros sistemas operativos que se baseiam nos mesmos princípios do UNIX. Naturalmente um dos mais populares na actualidade é o LINUX.

Figura 9.1 – Logotipo do Solaris OS, da Sun MicroSystems

9.1 Normalização de UNIX

Devido às múltiplas versões de UNIX existentes no mercado, foram publicadas normas para assegurar a compatibilidade entre todas as

versões. A primeira delas foi lançada pela AT&T e chamava-se SVID (System V Interface Definition) que, entre outras questões, definia como deveriam ser as chamadas ao sistema, o formato dos ficheiros. No entanto a sua versão mais importante, a de Berkeley (o BSD - Berkeley Software Distribution) pura e simplesmente ignorou estas normas. Posteriormente a IEEE usou um algoritmo consistente para rever as chamadas ao sistema de ambas versões (System V e BSD) e definiu aquelas que eram iguais como normas, surgindo assim a definição "Portable Operating System for UNIX" ou POSIX, que teve algum êxito e que vários fabricantes adoptaram rapidamente. A norma POSIX chama-se 1003.1. Posteriormente os institutos ANSI e ISO interessaram-se em normalizar a linguagem "C" e conjuntamente publicaram as definições de normas para outras áreas do sistema operativo como a interoperabilidade, o interpretador de comandos e outras. Na Tabela 9.1 listam-se as normas POSIX.

Tabela 9.1 As normas POSIX

Norma	Descrição
1003.0	Introdução
1003.1	Chamadas ao sistema
1003.2	Interpretador de comandos
1003.3	Métodos de prova
1003.4	Extensões para tempo real
1003.5	Linguagem Ada
1003.6	Extensões para segurança
1003.7	Gestão do Sistema
1003.8	Acesso transparente a ficheiros
1003.9	Linguagem Fortran
1003.10	Supercomputação

Apesar desta iniciativa, e quando as normas POSIX estavam no seu auge, foi formado um grupo de fabricantes de computadores (IBM, DEC e Hewlett-Packard) que criaram a sua própria versão de UNIX chamada OSF/1 (Open Software Foundation). Esta versão tinha como objectivo cumprir com todas as normas do IEEE, para além de apresentar um sistema de janelas (o X11), uma interface amigável para os utilizadores (MOTIF) e as definições para computação distribuída (DCE) e gestão distribuída (DME). A ideia de disponibilizar uma interface amigável em UNIX não foi original do OSF, uma vez que já a versão 3.5 do SunOS, da Sun Microsystems, oferecia uma interface amigável e um conjunto de bibliotecas para criar aplicações com uma interface gráfica tecnicamente eficiente e poderosa chamada SunWindows ou SunVIEW. Esta interface juntamente com

as suas bibliotecas estavam a evoluir apresentando versões para máquinas isoladas e para rede, em que as aplicações podiam estar a ser executadas num nó da rede e os resultados gráficos serem apresentados num outro nó da rede. No entanto a Sun demorou tanto a lançá-lo que deu tempo ao MIT de criar o X11, que veio a ganhar-lhe em popularidade.

A AT&T formou, juntamente com a Sun Microsystems e outras companhias a UNIX International e criou mais uma versão de UNIX, levando a que existam duas correntes principais do UNIX.

Actualmente a The Open Group, que é uma organização que agrupa grandes empresas do sector, como sejam a Capgemini, a EDS, a IBM, a HP, a NEC ou a BAE, têm vindo a trabalhar com a IEEE no sentido de criar standards abertos que assegurem a interoperabilidade entre sistemas. No âmbito do SO UNIX desenvolveram uma Especificação denominada Single UNIX, que actualmente vai na sua versão 3.

Figura 9.2 – Logotipo usado pela The Open Group para a sua Versão 3 da Especificação Single UNIX

9.2 Filosofia de UNIX

As ideias principais de UNIX foram derivadas do projecto MULTICS (Multiplexed Information and Computing Service) do MIT e da General Electrics. Estas ideias são:

- **Tudo é gerido como uma cadeia de Bytes:** Os dispositivos periféricos, os ficheiros e os comandos podem ser vistos como sequências de Bytes ou como entidades que as produzem. Por exemplo, a utilização de um terminal em UNIX é conseguida através de um ficheiro (geralmente no directório /dev com o nome ttyX).

- **Gestão de três descritores normalizados:** Todos os comandos possuem três descritores que, por defeito são chamados "stdin", "stdout" e "stderr". Estes descritores têm associado uma identificação número que é interpretada pelo SO por forma a identificar qual o canal de comunicação que o sistema operativo deve usar. Os valores são: *stdin = 0; stdout = 1 e stderr = 2*. O "stdin" é o teclado e, por defeito, o "stdout" e o "stderr" são o ecrã. Porém é possível redireccionar os canais de saída *stdou* e *stderr* para entidades distintas e diferentes do ecrã. As entidades podem ser ficheiros (registo de *log* para avaliação posterior) ou processos (localizados local ou remotamente).

Capacidades de canalizar e redireccionar: O "stdin", o "stdout" e o "stderr" podem ser usados para transferir o lugar a partir de onde os dados são lidos, para onde se enviam os resultados e para onde se enviam os erros, respectivamente. A nível dos comandos disponiveis através da janela de consola, os simbolos "maior que" (>) e "menor que" (<) servem para redireccionar os descritores associados aos canais de comunicação de saída (stdout e stderr) e de entrada (stdin) de dados, respectivamente. Por exemplo, em UNIX o comando "ls" lista os ficheiros do directório actual (é o mesmo que o comando "dir" no DOS). Se em vez de ver os nomes dos ficheiros no ecrã se quiser guardar no ficheiro "listafich", o redireccionamento é executado através do comando "ls > listafich". Uma situação que ocorre com frequência quando é usada a linha de comandos, consiste em identificar o estado de funcionamento de um programa através de mensagens que são enviadas para o ecrã, mesmo que este corra em modo gráfico. É possível redireccionar para entidades diferentes (ficheiros p.ex.) a informação de erro e a informação estado de funcionamento. Os comandos a usar são:

- ***ls -l > ls-l.txt*** - apenas o stdout é redireccionado para o ficheir;
- ***grep da * 2> grep-errors.txt*** - apenas o stderr é redireccionado para o ficheiro;
- ***grep da * 1>&2*** - o stdout é redireccionado para o stder;
- ***grep * 2>&1*** - o stderr é redireccionado para o stdou;
- ***rm -f $(find / -name core) &> log.txt*** - o stdout e stderr são ambos redireccionados para log.tx;
- programa ***< comandos.txt*** - o stdin é redireccionado para o ficheiro comandos.txt.

Se o que se deseja é imprimir os nomes, a canalização dos dados seria efectuada através do comando "ls | lpr", onde o símbolo "|" (pipe) traduz o comando de canalização e "lpr" é o comando para imprimir em UNIX BSD.

- **Criar sistemas grandes a partir de módulos:** Cada instrução em UNIX está desenhada para poder ser usada com "pipes" ou com "redireccionamento". Deste modo podem criar-se sistemas complexos utilizando comandos simples e elegantes. Como um exemplo simples, considere que existem quatro comandos separados A, B, C e D cujas funcionalidades são:

  A: lê matrizes verificando os tipos de dados e o formato.

  B: recebe matrizes, inverte-as e apresenta o resultado numa forma matricial.

  C: recebe uma matriz e coloca-lhe cabeçalhos "bonitos"

  D: manda uma matriz para a impressora cuidando dos saltos de página, etc.

  Como se vê, cada módulo tem uma actividade específica. Se o que se pretende é um pequeno sistema que leia um sistema de equações e que apresente o resultado como uma lista "bonita", basta recorrer a A para ler a matriz, canalizar o resultado para B que faz o cálculo da solução, passar essa solução para C para que lhe ponha os cabeçalhos "bonitos" e, finalmente, que os dados sejam passados para D que os manda imprimir. Assim, o comando completo seria "A | B | C | D".

9.3 Sistema de Ficheiros em UNIX

O sistema de ficheiros do UNIX, do ponto de vista do utilizador, tem uma organização hierárquica ou de árvore invertida que parte de uma raiz conhecida como "/" (diagonal). É uma diagonal ao contrário da que é usada no DOS.

O sistema de ficheiros de UNIX disponibiliza um poderoso conjunto de comandos e chamadas ao sistema. Na Tabela 9.2 são apresentados os comandos mais úteis para a gestão de ficheiros em UNIX.

A protecção de ficheiros em UNIX é gerida por meio de uma cadeia de permissões de nove caracteres. Os nove caracteres dividem-se em três grupos de três caracteres cada um.

RWX	RWX	RWX
1	2	3

Tabela 9.2 Gestão de Ficheiros em UNIX

Comando do UNIX	Função
rm	apaga ficheiros
cp	copia ficheiros
mv	altera o nome de ficheiros
ls	lista um directório
mkdir	cria um directório
cd	Mudar de directório
rmdir	apaga um directório
ln	cria uma "ligação simbólica" *(semelhante ao shortcut do windows)*
chmod	gere as autorizações
chown	muda de dono
man	acesso ao manual on-line do sistema

O primeiro grupo (1) especifica as permissões do dono do ficheiro. O segundo grupo especifica as permissões para os utilizadores que pertencem ao mesmo grupo de trabalho que o dono. Finalmente, o terceiro grupo indica as permissões para o resto do mundo. Em cada grupo de três caracteres podem aparecer as letras RWX, por esta ordem, indicando autorização de ler (READ), escrever (WRITE) e executar (EXECUTE). Por exemplo, a cadeia completa RWXR-XR-- indica que o dono tem os três permissões (READ, WRITE, EXECUTE), os membros do seu grupo de trabalho têm permissões de ler e executar (READ, EXECUTE) e o resto do mundo somente tem autorização de ler (READ).

As chamadas ao sistema mais úteis em UNIX são "open", "close" e "ioctl". Servem para abrir e fechar ficheiros e para definir as características de trabalho. Uma vez que no UNIX os terminais são acedidos através de ficheiros especiais, o "ioctl" (input/output control) serve para estabelecer a velocidade, paridade, etc. do terminal.

9.4 O núcleo do UNIX

O núcleo (kernel) do UNIX é classificado como sendo de tipo monolítico, mas este núcleo compreende duas partes principais: o

núcleo dependente da máquina e o núcleo independente. O núcleo dependente encarrega-se das interrupções, da gestão de dispositivos de baixo nível (lower half) e da parte da gestão da memória. O núcleo independente é igual em todas as plataformas e inclui a gestão das chamadas de sistema, o planeamento dos processos, a canalização, a paginação e transferência de dados, a gestão de discos e do sistema de ficheiros.

9.5 Gestão de processos no UNIX

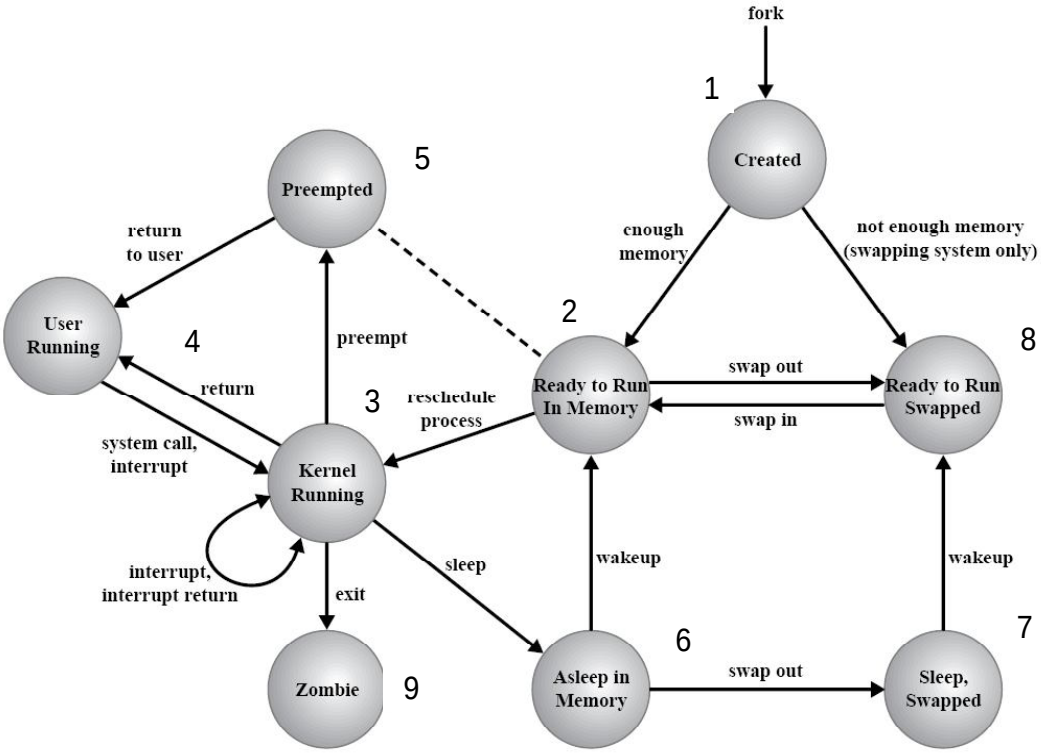
A gestão de processos no UNIX é por prioridade e round robin. Em algumas versões também é usada a gestão por ajuste dinâmico da prioridade, de acordo com o tempo que os processos tenham esperado e com o tempo que já tenham usado a CPU. O sistema disponibiliza facilidades para criar "pipes" entre processos, contabilizar o uso da CPU por processo e uma pilha comum para todos os processos quando necessitam de executar em modo privilegiado (quando fizeram uma chamada ao sistema). O UNIX permite que um processo faça uma cópia de si mesmo por meio da chamada "fork", a qual é muito útil quando se realizam trabalhos paralelos ou concorrentes. Também são disponibilizadas facilidades para o envio de mensagens entre processos. Recentemente a Sun Microsystems, a AT&T, a IBM, a Hewlett Packard e outros fabricantes de computadores chegaram a um acordo para usar um pacote chamado ToolTalk para criar aplicações que usem um mesmo método de transferência de mensagens.

A Figura 9.3 é um diagrama de estados que ilustra os possíveis estados que os processos podem apresentar no sistema operativo UNIX.

9.6 A gestão de memória em UNIX

Os primeiros sistemas com UNIX nasceram em máquinas cujo espaço de endereçamento era muito pequeno (por exemplo 64 kBytes) e tinham uma gestão de memória real relativamente complexa. Actualmente todos os sistemas UNIX utilizam a gestão de memória virtual, sendo o esquema mais usado a paginação a pedido e a combinação segmentação-paginação, em ambos casos com páginas de tamanho fixo. Em todos os sistemas UNIX é usada uma partição de disco rígido para a área de transferência de dados. Essa área é reservada durante a instalação do sistema operativo. Uma regra muito difundida entre administradores de sistemas é atribuir uma partição de disco rígido que seja pelo menos o dobro da quantidade de memória real do computador. Com esta regra é

possível trocar, de uma forma flexível, todos os processos que permaneçam em memória RAM num dado momento por outros que permaneçam no disco. Os processos que fazem parte do kernel não podem ser transferidos para o disco. Alguns sistemas operativos (como o Sun OS) permitem incrementar o espaço de transferência inclusivamente enquanto o sistema está em uso (no caso do Sun OS através do comando "swapon").



Legenda:

1. criado
2. executável
3. em execução em modo núcleo
3. em execução em modo utilizador
5. em preempção
6. bloqueado
7. bloqueado memória secundária
8. executável memória secundária
9. zombie

Figura 9.3- Diagrama de estados dos processos em UNIX

9.7 A gestão de E/S no UNIX

Como consequência da filosofia de gerir tudo como um fluxo de Bytes, os dispositivos são considerados como ficheiros que se acedem mediante descritoras de ficheiros cujos nomes se encontram geralmente no directório "/dev". Cada processo em UNIX mantém uma tabela de ficheiros abertos (onde um ficheiro pode ser qualquer

dispositivo de entrada/saída). Essa tabela tem entradas que correspondem aos descritores, os quais são números inteiros obtidos por meio da chamada de sistema "open". Conforme mencionado no ponto 9.2, os descritores stdin, stdout e stderr têm associados por omissão os valores numéricos 0, 1 e 2 respectivamente. Estes descritores podem ser usados directamente nas chamadas ao sistema read (ler dados do teclado) ou write (escrever dados no ecrã ou consola de erros). Na Tabela 9.3 apresentam-se as chamadas mais usuais para realizar entradas/saídas.

Tabela 9.3 - Chamadas ao sistema de entrada/saída

Chamada	**Função**
open	Obter um descritor inteiro
close	Terminar as operações sobre o ficheiro
lseek	Posicionar a entrada/saída
read, write	Ler ou escrever o ficheiro (dispositivo)
ioctl	Estabelecer o modo de trabalho do dispositivo

No UNIX é possível executar chamadas ao sistema de E/S de duas formas: síncrona e assíncrona. O modo síncrono é o modo normal de trabalho e consiste em fazer pedidos de leitura ou escrita que colocam o originador à espera até que o sistema lhe responda, isto é, que lhe forneça os dados desejados. Às vezes é necessário que o mesmo processo seja capaz de supervisionar o estado de vários dispositivos e de tomar certas decisões dependendo se existem dados ou não. Neste caso é necessária uma forma de trabalho assíncrona. Para este tipo de situações existem as chamadas às rotinas "select" e "poll" que permitem saber o estado de um conjunto de descritores.

Capítulo 10
Caso de Estudo: VMS

No presente capítulo serão apresentadas sucintamente as principais características do sistema operativo VMS.

O sistema operativo VMS (Virtual Memory System) é um dos mais robustos no mercado, sendo um sistema proprietário da companhia Digital Equipment Corporation. Actualmente com a sua versão OpenVMS 5.x está disponível para os processadores das máquinas VAX (CISC) e com o Alpha-chip (RISC). Disponibiliza um amplo conjunto de comandos através do seu interpretador Digital Command Language (DCL), utilidades de rede (DECnet), formação de "clusters" de computadores para partilha de recursos, correio electrónico e outras facilidades. É um sistema operativo multi-utilizador/multi-tarefa monolítico.

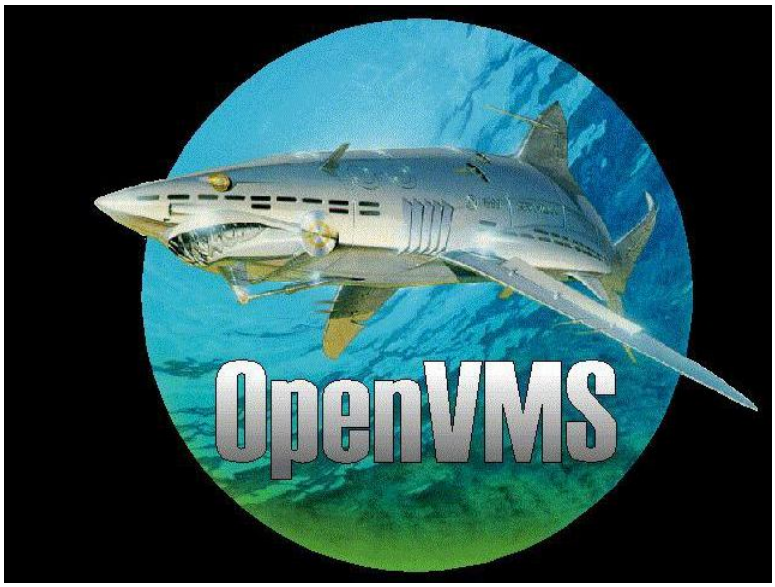


Figura 10.1 – Logótipo da versão OpenVMS

10.1 A gestão de ficheiros em VMS

O sistema de ficheiros do VMS é hierárquico apesar de a descrição das localizações dos ficheiros ter uma sintaxe própria.

Os ficheiros em VMS são referenciados com a sintaxe "nome.tipo;versão", onde "nome" é uma cadeia de caracteres alfanuméricos, "tipo" é a extensão do ficheiro utilizada geralmente para indicar a que aplicação pertence ("pas"=pascal, "for"=fortran, etc.) e "versão" é um número inteiro que o sistema se encarrega de atribuir de acordo com o número de vezes que o ficheiro foi modificado. Por exemplo, se se editou três vezes o ficheiro "teste.pas", poderão encontrar-se as versões "teste.pas;1", "teste.pas;2" e "teste.pas;3". Desta forma o utilizador obtém automaticamente a história dos seus ficheiros.

A protecção dos ficheiros é realizada mediante listas de controlo de acesso (*Acess Controlo Lists*). Podem estabelecer-se protecções para o dono do ficheiro (*owner*), para utilizadores privilegiados (system), para utilizadores que pertencem ao mesmo grupo de trabalho que o dono e para o resto do mundo. Para cada um dos anteriores utilizadores são geridas quatro autorizações: leitura (r), escrita (w), execução (e) e eliminação (d). Por exemplo, o seguinte comando:

```
$ set protection=(S:rwed,O:rwed,G:d:W:e) teste.pas
```

estabelece que o ficheiro "teste.pas" dará todas as permissões ao sistema (S:rwed) e ao dono (O:rwed), enquanto que aos membros do grupo de trabalho apenas dá autorização para eliminar (G:d) e ao resto do mundo autorização para execução (W:e).

Na Tabela 10.1 apresenta uma lista dos comandos para manipulação de ficheiros que são mais úteis no VMS. Estes comandos são bastante mnemónicos quando comparados com os comandos do UNIX, que são bastante crípticos.

O "Record Management System" (RMS) disponibiliza as facilidades para a manipulação de ficheiros tanto locais como em rede, como sejam: múltiplos modos de acesso a ficheiros para conseguir que tal acesso ocorra de forma concorrente assegurando a sua consistência e integridade, estabelecimento automático de canais no momento de abertura para evitar actualizações erróneas e para optimização interna das operações de E/S de acesso aos ficheiros. No caso dos ficheiros serem remotos, isto é não serem locais, é utilizado internamente o protocolo chamado "Data Access Protocol" (DAP).

Tabela 10.1 - Gestão de Ficheiros em VMS

Comando	Função
delete	apaga ficheiros
copy	copia ficheiros
rename	altera o nome de ficheiros
dir	lista um directório
create/directory	cria um directório
delete	apaga um directório
-	cria uma "ligação simbólica"
set protection	gere as autorizações
set uic	muda de dono

10.2 Gestão de processos em VMS

A gestão de processos em VMS suporta muitos ambientes de utilizador diferentes, como sejam, tempo crítico, desenvolvimento de programas interactivos e batch, tanto de forma concorrente como independente, ou uma combinação destas.

O scheduler VAX/VMS realiza calendarização de processos normais e de tempo real, baseando-se na prioridade dos processos executáveis no Balance Set. Um processo normal é referido como um processo de tempo partilhado ou processo background, enquanto que os processos em tempo real são referidos como sendo de tempo crítico.

No VMS os processos são geridos por prioridades e de forma apropriativa. Os processos são classificados da prioridade 1 à 31, sendo as primeiras quinze prioridades para processos normais e trabalhos em lote, e a 16 à 31 para processos privilegiados e do sistema. As prioridades não permanecem fixas todo o tempo, variando de acordo com alguns eventos do sistema. As prioridades dos processos normais podem sofrer variações até 6 pontos, por exemplo, quando um processo está à espera de um dispositivo e este é libertado. Um processo não liberta a unidade central de processamento enquanto não surgir um processo com maior prioridade.

O processo residente de maior prioridade a ser executado é sempre seleccionado para execução. Os processos em tempo crítico são iniciados pelo utilizador e não podem ser alterados pelo sistema. A prioridade dos processos normais pode ser alterada pelo sistema para optimizar a sobreposição da computação e de outras actividades de I/O.

Um aspecto importante do processo de planeamento de processos em VMS é a existência de um processo "monitor" ou "supervisor", o

qual é executado periodicamente para actualizar algumas variáveis indicadoras de desempenho e para re-calendarizar os processos em execução.

Existem versões de VMS que correm em vários processadores, e que disponibilizam bibliotecas para criar programas com múltiplos "threads". Disponibiliza, em particular, as interfaces "cma", "pthread" e "pthread-exception-returning". Todas estas bibliotecas são conhecidas como DECthreads e incluem bibliotecas com semáforos e transacções atómicas para a comunicação e a sincronização entre threads. O uso de threads serve para enviar parcelas de um programa para serem executadas em diferentes processadores aproveitando assim as capacidades de multiprocessamento.

Alguns dos serviços VMS do Sistema o controlo dos processos são:

- **Criar um processo:** O serviço de criação do sistema permite a um processo criar outro. O processo criado pode ser um subprocesso ou um processo completamente independente (são necessários privilégios para criar um processo).
- **Suspender um processo:** Permite que um processo se suspenda a si próprio ou suspenda outro (também necessita ter privilégios).
- **Reactivar um processo:** Permite a um processo reactivar outro, caso tenha privilégios para o fazer.
- **Eliminar um processo:** Permite que um processo se elimine a si próprio ou a outro que seja um seu subprocesso, não tendo que ter privilégios de eliminação.
- **Ceder Prioridade:** Permite que um processo ceda a prioridade a outros, considerando o scheduler.
- **Modo de espera:** Permite que o processo escolha um de dois modos: o modo por defeito corresponde ao processo entrar em espera quando requer um recurso e este está ocupado, ficando em espera até que este esteja desocupado; o outro modo corresponde a que, em vez de esperar quando os recursos estão ocupados, o processo não espera e notifica o utilizador que o recurso nesse momento não se encontra disponível.
- **Hibernar:** ocorre quando um processo está inactivo mas continua presente no sistema. Para que o processo entre em execução necessita de um evento que o desperte.

- **Acordar:** serviço que activa os processos que estão a hibernar.
- **Sair:** Aborta um processo.
- **Dar nome ao processo:** Serviço que permite dar ou mudar o nome de um processo.

10.3 Gestão de memória no VMS

O sistema operativo VMS utiliza um esquema de gestão de memória virtual combinado de segmentação paginada, que corresponde exactamente à descrição efectuada no capítulo sobre a gestão de memória. A inovação no VMS é que usa um esquema de dupla paginação quando as páginas são transferidas da memória RAM para o disco duro. Em primeiro lugar, quando uma página necessita de entrar em RAM, ela é colocada juntamente com outras páginas que lhe estão adjacentes, o que se justifica por meio da teoria do conjunto de trabalho, que especifica que é muito provável que no futuro imediato as referências à memória incidam precisamente nessas páginas. Deste modo, tem-se um algoritmo duplo: à entrada das páginas que são necessárias chama-se "paginação a pedido" e à transferência em antecipação das outras páginas do conjunto de trabalho chama-se "paginação antecipada".

10.4 A gestão das E/S no VMS

No VMS, usam-se nomes "lógicos" para descrever os dispositivos existentes no sistema, por exemplo, uma unidade de banda magnética pode receber o nome "BMAG0".

Um conceito importante tanto para os ficheiros como para os dispositivos é o "User Identification Code" (UIC) que permite estabelecer protecções adicionais. Por exemplo, são geridos cinco tipos de autorizações relativamente aos dispositivos: ler, escrever, executar, eliminar e controlar. No entanto nem todas as permissões se aplicam a todos os dispositivos, e algumas são geridas pelo sistema enquanto outras são geridas pelos utilizadores. Por exemplo, as permissões dos discos, das unidades de banda magnética e de outros dispositivos são estabelecidas pelo administrador do sistema.

Capítulo 11
Caso de Estudo: Windows NT

No presente capítulo serão apresentadas sucintamente as principais características do sistema operativo Windows NT e das suas subsequentes versões.

O Windows NT é um sistema operativo da Microsoft, cuja primeira versão foi lançada em Julho de 1993. O início do desenvolvimento, em 1998, resultou de uma parceria entre a Microsoft e a IBM, tendo dado origem à versão 3.0 do sistema operativo OS/2. Quando a parceria foi rompida a IBM continuou a trabalhar no OS/2, enquanto a Microsoft lançou a sua versão a que deu o nome de Windows NT.

O Windows NT foi desenhado para tirar partido de todas as capacidades disponibilizadas pelos processadores mais avançados da Intel, bem como de alguns dos processadores RISC. Alguns dos processadores em que corre são os seguintes Intel IA-32, MIPS, Alpha, PowerPC, SPARC, Intel i860, e Intel i960. O Windows NT foi a resposta da Microsoft ao UNIX.

O NT disponibiliza os mesmos serviços que o UNIX, pelo que é interoperável com as redes UNIX mas substitui os comandos crípticos do UNIX e a sua estrutura de ficheiros ARCANE. Também não foram adoptados os diferentes de tipos de interface de utilizador (GUI) do UNIX. Em vez disso foi adoptada uma interface simples e normalizada que é o Windows. Esta solução foi adoptada no versão 4 do NT, que adoptou uma interface semelhante à do Windows 95, que tinha acabado de sair.

Como ficou patente, o NT herdou as características que originalmente foram definidas para o sistema operativo OS/2 (que, entretanto, acabou por deixar de ser produzido, no final de 2006). Essas características são: um avançado sistema operativo de 32 bits, compatibilidade com o GUI do Windows, e o suporte às aplicações desenvolvidas para o DOS (apesar de as libertar das limitações daquele SO).

Microsoft Windows NT

Actualmente o kernel do Windows NT é usado em diversas versões dos Sistemas Operativos da Microsoft, como sejam, os Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista e Windows Server "Longhorn".

As características de desenho que fizeram do Windows NT um sistema operativo avançado são:

- **Extensibilidade**: O código poderá ser alterado (crescer ou ser alterado) de forma simples de modo a evoluir com as necessidades do mercado.
- **Portabilidade**: O código poderá utilizar qualquer processador sem que isto afecte o seu desempenho de forma negativa.
- **Fiabilidade e robustez**: O sistema deverá auto-proteger-se tanto dos maus funcionamentos internos como externos. Devendo comportar-se de forma previsível em qualquer momento e as aplicações não deverão afectar o seu funcionamento de forma negativa.
- **Compatibilidade:** O sistema evoluirá com a tecnologia, mas ao mesmo tempo os seus API e UI manter-se-ão compatíveis com os sistemas já existentes da Microsoft.
- **Multiprocessamento e escalabilidade:** As aplicações poderão beneficiar das vantagens oferecidas por qualquer computador e os utilizadores poderão executar as aplicações tanto em computadores monoprocessador como multiprocessador.
- **Computação distribuída:** o NT é capaz de repartir as suas tarefas computacionais por outros computadores da rede para oferecer aos utilizadores mais poder que o disponibilizado por qualquer computador individual da rede. Poderá usar computadores tanto local como remotamente de forma transparente para o utilizador (efeito de sinergia da rede).
- **Desempenho:** O sistema deve responder e ser o mais rápido possível em cada plataforma de HARDWARE.
- **Compatibilidade com POSIX:** o POSIX (Portable Operating System based on UNIX) é uma norma especificada pelo governo dos EUA, que deverá ser cumprida por todos os contratos na área computacional que sejam estabelecidos com

aquele governo. NT pode proporcionar um ambiente opcional para a execução de aplicações POSIX.

11.1 Características de Windows NT

Um sistema operativo é um programa complexo que necessita de um modelo unificado para assegurar que o sistema pode incluir características próprias sem que tal implique a alteração do desenho dos sistemas. O desenho do Windows NT foi orientado por uma combinação de diversos modelos que foram agregados no Windows NT. As principais características do NT são:

- Estrutura de 32-bits;
- Suporte de memória virtual;
- Preemptive multitasking;
- Suporte para multiprocessador;
- Arquitectura cliente/servidor;
- Segurança e integridade do sistema;
- Compatibilidade com outros Sistemas Operativos;
- Independência das plataformas; e
- Networking (Interoperabilidade).

11.2 O núcleo de Window NT

O núcleo é a base do sistema operativo, onde reside o executivo do Windows NT, que realiza as seguintes operações:

- Entrada e saída de tarefas no sistema.
- Processamento de interrupções e excepções.
- Sincronização de multiprocessadores.
- Recuperação do sistema em caso de falha.

11.2.1 Entrada e saída de tarefas no sistema

Cada objecto de tipo tarefa é criado em resposta a um pedido da aplicação feito através de uma chamada ao kernel, que seja proveniente de uma mini-tarefa consistente, e que peça o início da execução de uma tarefa maior.

Cada tarefas pode encontrar-se num dos seguintes estados: em execução, na lista para execução à espera de vez, bloqueada à espera de recursos, ou finalizada. O kernel conta com um módulo chamado "despacho" que se encarrega de autorizar a entrada em execução dos processos e de os dar por terminados. O despacho examina, igualmente, a prioridade dos processos para determinar qual a ordem em que vão ser executados, e que se encarrega de suspender e activar os processos.

11.2.2 Processamento de interrupções e excepções

Os serviços do Windows NT são activados através de mensagens e de interrupções.

O NT gere as interrupções como qualquer outro sistema operativo. A chegada de sinais pelo bus, devidos a falhas dos programas ou por pedidos de E/S dos periféricos, são detectadas pelo núcleo. Na Figura 11.1 podem observar-se as partes do núcleo de Windows NT.

11.2.3 Sincronização de multiprocessadores

Esta característica assegura que somente uma tarefa pode aceder a um recurso de cada vez. Num sistema baseado em multiprocessadores com memória partilhada, dois ou mais processadores podem estar executando tarefas que necessitam de aceder à mesma página de memória ou realizar operações sobre um mesmo objecto. O núcleo e o executivo do NT disponibilizam mecanismos para assegurar a integridade do sistema através de sincronização. No caso do kernel a sincronização é gerida através de trincos colocados em pontos críticos das instruções do nível despacho, desta forma nenhum outro processador pode executar código ou aceder a dados protegidos por um dos trincos de tipo spin, até que este seja libertado. O executivo do NT realiza a sincronização através da família dos objectos de sincronização.

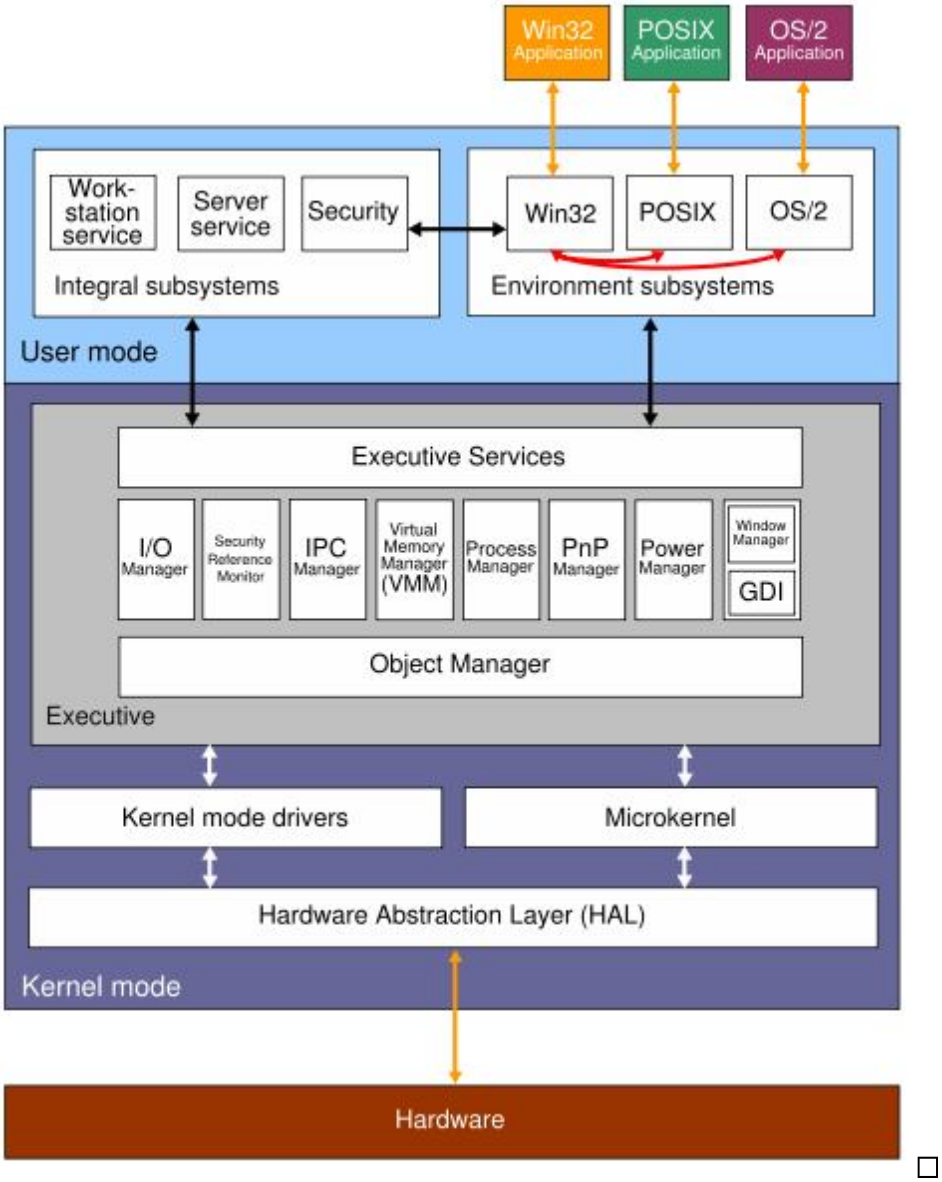


Figura 11.1 Núcleo de Windows NT

11.2.4 Recuperação do sistema

A última função do kernel consiste na recuperação do sistema em caso de uma caída. Quando existe uma falha de alimentação num sistema NT é gerada uma interrupção de alta prioridade, a qual dispara, tão rápido quanto seja possível, uma série de tarefas desenhadas para preservar a integridade do sistema operativo e dos dados.

O micro-núcleo de Windows NT contém uma camada de abstracção do hardware que é o limite entre o executivo do NT e o hardware específico do computador. O NT foi desenhado, tomando como

exemplo os sistemas UNIX, de modo a que no acoplamento a diferentes plataformas de hardware as alterações de código sejam mínimas.

11.2.5 Arquitectura cliente/servidor

Como se pode observar na Figura 11.1 o Windows NT pode operar de dois modos diferentes: modo de utilizador e modo privilegiado (kernel).

Os programas dos utilizadores (como sejam as bases de dados, folhas de cálculo, ou um sistema de reservas de um hotel) são executados em modo utilizador.

O código do Windows NT (*Executive*), que é o coração do sistema operativo, é executado em modo kernel, que é um modo de alta segurança livre de interferências dos processos dos utilizadores. O executivo do NT realiza tarefas como a gestão de entradas e saídas, da memória virtual, e de todos os processos, para além de controlar as ligações entre o NT e o hardware do computador.

O modo de utilizador, inclui subsistemas protegidos (referidos na Figura como *Environment subsystems*). Um exemplo é o Win32 API. Usando esta API os programadores podem controlar o hardware sem preocupações, uma vez que este conjunto de funcionalidades protege o sistema de operações ilegais, como sejam modificações no espaço de endereçamento de memória, e assegura a recuperação em caso de falha do sistema. Adicionalmente o API tem regras de segurança que protegem os outros subsistemas de interferências.

No ambiente NT as aplicações dos utilizadores (Win32, POSIX ou OS/2) são os clientes e os subsistemas protegidos são os servidores. As aplicações (clientes) mandam mensagens aos subsistemas protegidos, que interagem com o executivo do NT, o qual disponibiliza um conjunto de serviços partilhados para todos os servidores. Os servidores também respondem aos clientes usando mensagens.

No NT, os servidores que estão em execução num processador local podem mandar mensagens dos seus clientes para outros servidores que estejam a ser executados em processadores remotos, sem que o cliente tenha de saber detalhes sobre esses servidores remotos.

O modelo cliente/servidor tornou os sistemas operativos mais eficientes eliminando duplicações de recursos e aumentou o suporte que o sistema operativo oferece para multiprocessamento e redes. Esta arquitectura permite que outros API's sejam adicionados sem

ter que acrescentar um novo executivo de NT para a sua gestão. Por outro lado cada subsistema é um processo separado com a sua própria memória protegida. Assim, se um dos subsistemas falha não ocorre uma falha geral do sistema.

O executivo NT (Figura 11.1) funciona como um sistema operativo completo que não disponibiliza um interface de utilizador e que é composto por quatro camadas, que são as seguintes:

- **Serviços do sistema:** são as chamadas ao sistema que servem como meio de comunicação entre os modos dos processos e os componentes do executivo. A forma como o modo de utilizador e o modo kernel interagem entre si é através de chamadas ao sistema. Visto de outra maneira, os serviços do sistema são o API para o modo de utilizador.

- **Componentes do executivo:** o executivo do NT é constituído por um conjunto de componentes primários, que se encarregam das seguintes operações críticas do sistema: gestão de objectos, monitorização da segurança do sistema, gestão de chamadas de procedimentos locais, gestão da memória virtual, gestão de processos, gestão das entradas e saídas, gestão de Plug-and-play e gestão de energia.

11.2.6 Gestor de Objectos

Este módulo é o responsável por criar, gerir e eliminar os objectos do executivo do NT. Os objectos são os processos e os dados, bem como os objectos próprios dos níveis do sistema.

Existem dois tipos principais de objectos:

- os objectos executivos, que são criados dentro do executivo e que são acessíveis para o executivo e para os subsistemas protegidos; e

- a restante classe de objectos, que estão somente acessíveis ao executivo e que são designados " objectos do kernel" e que somente podem ser modificados dentro do kernel.

O gestor de objectos tem as seguintes funções:

- Atribuir memória;

- Atribuir descritores de segurança do objecto, que permitem inibir o acesso ao dito objecto;

- Colocar o nome do objecto dentro da posição adequada no directório de objectos; e
- Criar e gerir um ponteiro para o objecto, que elimina a necessidade de chamar o objecto com base na sua localização.

11.2.7 Monitor de segurança do sistema

O monitor de segurança do sistema trabalha em conjunção com o gestor de objectos para disponibilizar um mecanismo de controlo de acesso aos objectos.

A informação de controlo de acesso está associada a cada objecto. Dentro desta informação cada objecto gere uma lista de controlo de acessos (ACL). Esta lista regista as permissões de acesso estabelecidas para o objecto. No entanto, o dono do objecto pode transferir as autorizações.

11.3 Gestão de ficheiros em Windows NT

No respeitante ao sistema de ficheiros, o NT tem compatibilidade com os seguintes sistemas de ficheiros:

- FAT (DOS);
- HPFS (OS/2)

A migração de ficheiros do DOS ou do Windows 16-bits para o sistema de ficheiros do Windows NT (NTFS) pode originar alguma confusão nas permissões de segurança daqueles ficheiros, situação facilmente ultrapassável com a intervenção do administrador.

A facilidade de suportar diferentes tipos de ficheiros ajuda a conseguir atingir uma característica chamada "personalidade do sistema operativo". Esta característica consiste na facilidade de um sistema operativo suportar a execução de aplicações criadas para sistemas operativos diferentes. Como já se referiu, e se pode observar na Figura 11.1, o NT suporta aplicações Win32, de POSIX e de OS/2 (as suas diferentes personalidades).

11.4 Gestão de processos no Windows NT

Na arquitectura do NT, os processos são segmentados em componentes mais pequenos chamados "threads". O Windows NT suporta várias tarefas ao mesmo tempo. Existem dois tipos de multitarefas, o apropriativo (*preemptive*) e o não apropriativo (*no preemptive*).

Com a multitarefa apropriativa a execução de um thread pode ser suspendida após um determinado tempo (a parcela de tempo atribuída ao processo) pelo sistema operativo, para permitir que outro thread seja executado. Enquanto que com a multitarefa não apropriativa, é o thread que determina quando devolve o controlo ao sistema operativo para permitir que outro thread seja executado. O NT, de modo idêntico ao que acontece com o OS/2 e o UNIX, usam *preemptive multitasking* para suportar a execução "simultânea" de vários processos.

11.4.1 Gestor de Processos

O gestor de processos é um componente ambiental que cria e elimina processos e tarefas. Como o gestor de objectos, o gestor de processos vê os processos como se fossem objectos. De facto, o gestor de processos pode ser considerado como uma instância específica do gestor de objectos porque este gestor cria, gere e destrói um único tipo de objectos.

Existe apenas uma funcionalidade adicional no gestor de objectos relativamente ao gestor de processos, que consiste na gestão do estado de cada um dos processos (executar, suspender, reiniciar, terminar uma tarefa).

As chamadas a procedimentos locais (*Local Procedure Calls* -LPC, ver Figura 11.1) são usadas para passar mensagens entre dois processos diferentes que estejam em execução dentro de um mesmo sistema NT. Estes sistemas foram modelados utilizando como modelo as chamadas a procedimentos remotos (*Remote Procedure Calls* - RPC). As RPC consistem numa forma normalizada de passar mensagens entre um cliente e um servidor através de uma rede. De modo semelhante os LPCs passam mensagens de um procedimento cliente a um procedimento servidor num mesmo sistema NT.

Cada processo cliente num sistema NT que tem capacidade de comunicação por meio de LPCs deve ter, pelo menos, um objecto de tipo porto atribuído a ele. Este objecto tipo porto é o equivalente a um porto de TCP/IP de um sistema UNIX.

11.4.2 Suporte para multiprocessador

Existem dois tipos de sistema multiprocessador: o assimétrico e o simétrico.

Na abordagem assimétrica existe um processador (Mestre) que executa o sistema operativo enquanto os restantes (Escravos) executam as tarefas das aplicações. A vantagem desta abordagem é que a gestão do sistema é fácil e as alterações resultantes da variação do número de processadores são mínimas, o que elimina muitos problemas de integridade dos dados. A grande desvantagem é que existindo somente uma cópia do sistema operativo num único processador (Mestre), se este processador falhar todo o sistema falha porque todos os recursos que são geridos pelo sistema operativo deixam de poder ser acedidos.

Na abordagem simétrica o sistema operativo - ou uma grande parte dele - é executado em qualquer dos processadores disponíveis, e todos estes têm acesso aos recursos, a menos que cada recurso seja atribuído a um processador específico. Apesar de ser mais difícil de implementar tem muitas vantagens relativamente à abordagem anterior.

Primeiro, este tipo de sistemas tende a ser mais eficiente porque as tarefas tanto do sistema operativo como dos utilizadores podem ser distribuídas de forma balanceada por todos os processadores. Como as solicitações do sistema operativo podem ser repartidas por todos os processadores, o tempo de inactividade de um processador enquanto outro está sobrecarregado é mínimo.

Segundo, se um processador falha, é possível que as suas tarefas sejam repartidas entre os demais e não originando que todo o sistema fique parado ou que falhe. E finalmente, a portabilidade do sistema é maior, uma vez que não segue a arquitectura de master/slave.

O NT implementa o modelo multiprocessador simétrico.

11.5 Segurança e integridade do sistema

No essencial, a segurança no Windows NT refere-se a dois aspectos:

- O controlo total no acesso ao sistema e aos ficheiros ou subdirectórios existentes no sistema. (Controlo de acesso e segurança do sistema)

- A protecção individual dos processos e do sistema operativo, para que um bug ou um programa destrutivo não provoque a falha do sistema nem afecte ou outros programas ou aplicações. (Integridade do sistema)

No primeiro ponto, o controlo sobre o acesso ao sistema refere-se à gestão de usernames e passwords para poder aceder ao sistema operativo. Desta forma é possível manter os utilizadores que não têm autorização fora do sistema. O nível de segurança seguinte associado a este ponto, são os privilégios atribuídos a cada utilizador, a todos os utilizadores ou a grupos de utilizadores no acesso aos directórios e aos ficheiros do sistema. Por exemplo, o acesso aos ficheiros do sistema do NT está estritamente limitado ao administrador do sistema, enquanto que as aplicações comuns como sejam os processadores de palavras ou as folhas de cálculo podem ser acedidos por todos os utilizadores.

O segundo aspecto trata da integridade do sistema. A perda de informação em sistemas operativos para um único utilizador não é tão grave comparada com a dos sistemas operativos para redes, nos quais a perda de informação pode levar muito tempo a ser recuperada. O NT disponibiliza amplas funcionalidades para assegurar a integridade do sistema quando o NT se encontra a executar em condições difíceis, bem como para recuperar o sistema de forma rápida e simples.

11.5.1 Controlo de Acesso e Segurança do sistema

O Windows NT conta com um extenso sistema de controlos de segurança para o acesso a ficheiros. O objectivo da segurança no Windows NT é disponibilizar o acesso somente aos utilizadores que estejam autorizados, controlar o acesso concorrente aos ficheiros, aos directórios e aos recursos do sistema.

A segurança no sistema Windows NT deve ser conFigurada pelo administrador do sistema, sendo necessário que todos os sistemas tenham um administrador (incluindo os sistemas mono-utilizador). O administrador estabelece os nomes de utilizador, cria grupos de utilizadores, atribui os utilizadores aos grupos, controla as passwords e define os níveis de acesso às funcionalidades do sistema. Em poucas palavras o administrador controla todos os pontos de acesso do sistema.

O administrador pode controlar o acesso específico a certas funções do sistema, especialmente aquelas que afectam o seu funcionamento.

Este sistema de controlo é designado de política de direitos do utilizador. Assim, o administrador através desta política pode controlar as actividades que um utilizador efectua, tanto local como remotamente.

11.5.2 Integridade do sistema

Por integridade do sistema entende-se a capacidade que o sistema tem de permanecer activo quando uma das suas aplicações falha. Windows NT está desenhado para prevenir a falha catastrófica do sistema no caso de algumas das suas aplicações falharem e para esse efeito estabelece os seguintes quatro mecanismos de protecção de memória:

- **Espaço de endereçamento separado:** cada processo gere os seus próprios endereços virtuais e o sistema proibe o acesso a espaços de memória de outros processos;
- **Modos de Kernel e de utilizador separados:** todas as aplicações correm em modo de utilizador, estando proibido o acesso ou a modificação do código ou dos dados do sistema que residam no kernel;
- **Flags de páginas:** cada página da memória virtual tem uma flag que determina como pode ser acedida em modo utilizador e em modo kernel;
- **Segurança dos Objectos:** o gestor virtual da memória cria um tipo especial de objecto chamado objecto-secção o qual funciona como uma janela para a memória virtual. Portanto, cada vez que um processo acede um objecto-secção, o sistema determina se o processo tem as permissões de leitura e/ou escrita na memória.

Ainda relacionado com a integridade do sistema, o Windows NT estabelece políticas e procedimentos de protecção ou de acesso a recursos. Desta forma protege os processos de entrar em deadlock em situações de competição por recursos.

11.6 Gestão de memória em Window NT

Como foi referido no início do capítulo, o Windows NT é um sistema operativo de 32 bits com a facilidade da gestão de memória virtual. Em seguida serão referidas as características do S.O. nesta vertente.

11.6.1 Endereçamento de 32 bits

Este tipo de endereçamento tem várias vantagens. Primeiro, elimina a memória segmentada, pelo que o desenvolvimento do software é mais fácil e rápido. Os programadores não necessitam de estar familiarizados com os pedidos de memória das suas aplicações. Além disso, o endereçamento de 32-bits melhora o desempenho do sistema eliminando parte do "overhead" do software com a gestão da memória. Ao deixar a gestão de memória elimina também as incompatibilidades com o hardware e o software, o que significa que a instalação e a conFiguração do NT se tornou tão simples e fácil como a do DOS ou a do Windows de 16-bit.

Outra vantagem do endereçamento de 32-bits é o considerável incremento no tamanho disponível para os programas e os dados. O NT suporta um máximo de 4 GigaBytes de programas e sistema, o que é muitas vezes mais do que o DOS e as versões Windows de 16-bit podem suportar. Esta é uma grande vantagem quando se tem de gerir aplicações complexas que processam ficheiros muito grandes (como os de processamento de imagens) ou para as aplicações orientadas a transacções críticas, as quais seriam impossíveis de implementar em DOS.

11.6.2 Suporte de memória virtual

O endereçamento de 32-bits oferece às aplicações acesso a 4 GigaBytes de memória, dos quais 2 GB estão reservados para uso do sistema operativo, e que são mais que suficientes para a quase todas as aplicação imagináveis.

Quando o NT é instalado pela primeira vez, o NT setup program verifica quanto espaço está disponível na RAM e em DD. Com base nisto o NT cria um swap file, o qual deve ser pelo menos do mesmo tamanho que a RAM.

O gestor de memória virtual do NT realiza duas tarefas básicas:

- Primeiro, gere os dados guardados no disco e mapea os endereços dos dados que estão no disco com o espaço de endereçamento em 32-bits lineares. As aplicações podem fazer operações com os dados sem importar a localização física destes (disco ou RAM).

- Segundo, o gestor de memória virtual transfere algumas parcelas da RAM para o swap file quando os processos tentam usar mais RAM do que a que está disponível. Neste caso, as partes inactivas da RAM são colocadas temporariamente no

swap file até que sejam necessárias na RAM, o tamanho de página que é transferida da RAM para o disco é de 4 K. Neste caso recorre-se a paginação a pedido.

11.6.3 Gestor de memória virtual

O gestor de memória virtual (MMV) do NT realiza três funções essenciais:

- a gestão do espaço virtual de cada um dos processos;
- a gestão do espaço de memória partilhada entre os processos; e
- a protecção da memória virtual de cada processo.

Dentro da gestão da memória virtual de cada processo são realizadas as seguintes tarefas:

- Reservar e libertar a memória virtual;
- Fazer a leitura e a escrita de páginas da memória virtual;
- O estabelecimento de trincos nas páginas seleccionadas da memória virtual, o que força que essas páginas se mantenham na memória real sem serem transferidas para o disco (*swap*);
- O encadeamento da informação dentro das páginas de memória virtual protegida; e
- A limpeza das páginas virtuais do disco.

O gestor de memória virtual permite que um ou vários processos partilhem as mesmas páginas de memória virtual, de tal forma que dois ou mais processos podem ter de gerir a mesma área de memória virtual. O MMV tem uma característica singular que consiste na capacidade de endereçar uma pequena área do espaço de memória virtual de outro processo, esta janela do espaço total de memória virtual de processos é chamada uma vista e permite que um processo trabalhe com muitas parcelas pequenas de grandes espaços de memória virtual para criar o seu próprio espaço de memória virtual.

11.6.4 Memória protegida

O gestor de memória do Windows NT permite proteger certas regiões de memória de acessos inadvertidos ou deliberados provenientes de outros processos. O MMV é responsável pelo

mapeamento entre os endereços de memória virtual e os endereços de hardware específicos, assegurando desta forma que dois processos não podem aceder a uma mesma página de memória. O MMV utiliza técnicas de gestão de memória em hardware que estão disponíveis no computador anfitrião (host) e desta forma estabelece a protecção a cada uma das páginas. As protecções das páginas não estão previstas no hardware pelo que Windows NT tem que fazê-lo através do software definindo páginas individuais de memória como sendo de leitura e escrita, somente de leitura, somente de escrita, de execução ou sem acesso.

Para aplicações que utilizam grandes espaços de memória, o Windows NT introduziu um conceito chamado " bookend " o qual consiste num página que marca o final do código ou dos dados. Quando o processo chega a uma destas páginas, chamadas páginas guarda, sabe que se encontra num estado fora da memória e solicita memória adicional ao MMV protegendo desta forma a falha da aplicação.

Em situações onde dois ou mais processos necessitam de aceder à mesma região de memória, o MMV cria uma cópia da página para o segundo processo utilizar, estabelecendo desta forma o mecanismo de protecção de páginas e a memória partilhada.

Quando um processo quer modificar certos dados na memória partilhada deve primeiro modificá-los na sua cópia das páginas de memória e, posteriormente, notificar o MMV de que necessita de actualizar as alterações nas páginas dos demais processos, prevenindo desta forma que o processo modifique directamente as páginas de memória que não lhe pertencem.

11.7 Gestão de E/S no Windows NT

O gestor das entradas e saídas do Windows NT pode ser considerado como um despacho das entradas e saídas do sistema, uma vez que este módulo estabelece a comunicação, por um lado, entre os subsistemas protegidos e os controladores de dispositivos, por outro lado.

Quando uma aplicação solicita um serviço de entradas/saídas, o gestor de E/S converte o pedido num IRP (I/O request packet) e identifica o gestor de dispositivos adequado para servir o pedido feito pelo processo. Um gestor de dispositivos recebe o pacote de dados e processa-o, mandando o resultado para o gestor de entradas e saídas ou para outro gestor de dispositivos que continue

a processar esse resultado. O destino final do pacote de dados é o gestor de entradas e saídas.

Quando um pedido é enviado a um gestor de dispositivos este é responsável pelo seu controlo através de sistemas de filas.

11.8 Compatiblidade com outros Sistemas Operativos

Uma dos maiores qualidades do Windows NT é a sua capacidade de interoperar com múltiplos sistemas operativos. Um sistema NT pode executar a maioria dos programas do DOS, do Windows 16-bits, e a maioria das aplicações orientadas a caracteres do OS/2, bem como as que cumpram com a norma POSIX.

11.8.1 Independência de plataformas

O objectivo do Windows NT é ser um sistema operativo que possa ser executado em diferentes plataformassendo compatível, por exemplo, com os seguintes processadores:

- Família Intel x86
- Motorola 680x0
- MIPS 400
- ALFA da Dec.
- HP-PA da Hewlett Packard
- SPARC RISC da Sun Microsystems.
- RS/6000 de IBM
- Powerpc (Apple, IBM e Motorola)

A independência de plataforma baseia-se no conceito de desenvolver um kernel específico para cada um dos diferentes processadores que sirva de interface entre o hardware específico e as chamadas de sistema do NT.

11.9 Interoperabilidade (Networking)

O Windows NT disponibiliza quatro tipos diferentes de suporte de redes:

- Ponto a ponto: nas ligações ponto a ponto com outros sistemas Windows NT e Windows para grupos.
- Interoperabilidade: com outros sistemas operativos orientados a rede (por exemplo, DEC Pathworks, Novell Network, BanyanVINES através da arquitectura de sistemas abertos de Windows (WOSA), bem como sistemas UNIX baseados em TCP/IP).
- SNA: Ligações a host baseados em redes SNA através de uma versão própria de servidores de comunicações da Microsoft DCA.
- Suporte para redes Microsoft baseadas em sistemas operativo de rede LAN Manager.

Bibliografia / Literatura recomendada

Andrew S. Tanenbaum: "Sistemas Operativos Distribuídos", Prentice Halh (1996).

Andy Johnston, Robin Anderson: Unix Unleashed (4th Edition)

Curso de redes e linux, Faculdade de Engenharia da Universidade do Porto (documento PDF disponível na INTERNET)

Daniel Barrett: Linux Pocket Guide (Pocket Guide: Essential Commands)

DeeAnn LeBlanc: Linux For Dummies® (For Dummies)

Ellen Siever, Aaron Weber, Stephen Figgins: Linux in a Nutshell (In a Nutshell (O'Reilly))

Fernando Pereira: Linux - Curso Completo 5ª Edição ; F C A-Editora Informática

George Coulouris, Jean Dolhimore, Tim Kindberg: "Distributed Systems, Concepts and Design", Addison-Wesley (1994).

http://mega.ist.utl.pt/~ic-so/public05-06/index.html

José Alves Marques, Paulo Guedes: "Fundamentos de Sistemas Operativos", Editorial Presença

Mike McGrath: Linux in Easy Steps (In Easy Steps)

Paulo Trezentos: Fundamental do Linux; 3ª Edição Actualizada; F C A-Editora Informática

R. Love: Linux Systems Programming

Robert Shimonski: Sams Teach Yourself Unix in 10 Minutes (2nd Edition) (Sams Teach Yourself)

Robin Burk, David B. Horvath: UNIX Unleashed: System Administrator's Edition

Steve Frampton: Linux Administration Made Easy

Tom Adelstein, Bill Lubanovic, Falko Timme: Linux System Administration