

1- INTRODUÇÃO

Quem já utilizou um computador teve naturalmente oportunidade de interactuar com o sistema operativo para realizar operações indispensáveis, por vezes mesmo elementares, como guardar um ficheiro numa directoria ou executar uma aplicação. Contudo, apesar deste conhecimento operacional, o funcionamento interno do sistema operativo é relativamente desconhecido e, nalguns casos até, envolto em mistério, porque é frequentemente considerado responsável por situações inexplicáveis de mau funcionamento ou bloqueio dos computadores.

Neste capítulo, o objectivo fundamental é procurar esclarecer qual a missão e funções do sistema operativo, e que critérios devem ser usados para validar a adequação de um sistema operativo à sua missão.

Para que possamos entender melhor a realidade de entidades complexas, como a do sistema operativo, podemos olhar para a sua história que ilustra muitos dos conceitos, presentes nos sistemas actuais, ainda que eventualmente sob formas mais sofisticadas. A cronologia das principais etapas de desenvolvimento dos sistemas operativos permite descrever a evolução, e simultaneamente introduzir a maioria dos conceitos que serão objecto de estudo dos capítulos seguintes.

No final do capítulo, é introduzida uma classificação dos sistemas operativos que se baseia nas designações habituais do mercado informático, e que permite identificar diferentes requisitos e características dos sistemas.

1.1 FUNÇÃO DOS SISTEMAS OPERATIVOS

A principal função de um sistema informático é executar uma ou várias aplicações para os seus utilizadores. Nas aplicações e nos respectivos dados reside, em última instância, o valor que pode ser atribuído ao sistema informático e que justifica o investimento que as organizações, ou mesmo os indivíduos, fazem na sua aquisição, no seu desenvolvimento e manutenção.

Uma aplicação é um conjunto de programas e de informação persistente, normalmente guardada por exemplo em ficheiros ou bases de dados. Para que os programas se executem e mantenham a informação persistente, são necessários recursos quer de nível físico como processador, memória, discos, periféricos que vulgarmente designamos por *hard-*

2 - ORGANIZAÇÃO DO SISTEMA OPERATIVO

Neste capítulo definimos os principais elementos da arquitectura do sistema operativo, descrevendo os blocos constituintes mais relevantes e as funções que realizam.

Dado o papel de charneira do sistema operativo entre as aplicações e o *hardware*, na descrição da organização interna do sistema é fundamental perceber a sua ligação à arquitectura física do computador, sobretudo aos mecanismos básicos que suportam a implementação dos conceitos do sistema operativo. O modo de funcionamento núcleo/utilizador, as interrupções e a unidade de gestão de memória são os elementos chave do *hardware* que permitem a instanciação das abstrações do sistema operativo. Iremos analisar estes elementos numa perspectiva da sua utilização no contexto do sistema operativo, e não de uma análise mais detalhada, que deverá ser efectuada no estudo da arquitectura dos computadores.

A interface programática do sistema operativo corresponde às chamadas sistema, estando o seu funcionamento directamente relacionado com estes mecanismos de base, nomeadamente a barreira de protecção núcleo/utilizador e as interrupções. O funcionamento geral das chamadas sistema é aqui explicado porque todas elas obedecem a uma estrutura de base, independentemente da funcionalidade específica a que cada uma está associada.

Um aspecto importante na compreensão do sistema operativo reside nas alternativas da sua organização interna e na forma como esta se traduz nas tentativas para tornarem a sua evolução mais fiável e mais facilmente adaptável a novos requisitos. Veremos as principais linhas de evolução e, apesar de muitas delas não terem uma materialização nos sistemas mais divulgados, tiveram forte influência na respectiva evolução.

O capítulo encerra com a descrição da organização do sistema operativo dos sistemas de referência, o Unix/Linux e o Windows, que será sucessivamente aprofundada nos capítulos seguintes.

2.1 ORGANIZAÇÃO DO SISTEMA OPERATIVO

O sistema operativo é normalmente composto por três entidades que podemos ver representadas na Figura 2.1:

- **Núcleo** do sistema operativo – O núcleo ou *kernel* corresponde aos programas que implementam a funcionalidade básica do sistema operativo;

3 - PROCESSOS: MODELO COMPUTACIONAL

A execução, em paralelo, de múltiplos programas, na mesma máquina, corresponde ao conceito de multiprogramação e constitui a base dos sistemas operativos. O modo como se consegue a execução paralela de vários fluxos de actividade é um aspecto fundamental na compreensão dos sistemas operativos.

Neste capítulo introduz-se o conceito de processo que é a entidade básica responsável pela execução dos programas. Como os restantes objectos do sistema operativo, os processos têm associado um conjunto de variáveis que os caracterizam e um conjunto de funções que definem o seu modelo computacional e que basicamente gerem o seu ciclo de vida e o seu estado de execução.

O modelo incorpora diversas extensões para tratar mecanismos de excepção ou acontecimentos assíncronos. Mais recentemente, o modelo multitarefa permite a programação de actividades concorrentes no interior de um processo, dotando-os de uma maior capacidade de gestão de actividades paralelas.

No final do capítulo analisamos como o modelo computacional dos processos se materializa nos sistemas Unix e Windows.

3.1 MULTIPROGRAMAÇÃO

A MULTIPROGRAMAÇÃO CONSISTE NA EXECUÇÃO EM PARALELO OU CONCORRENTE DE MÚLTIPLOS PROGRAMAS NO MESMO COMPUTADOR.

Como é óbvio, a execução concorrente de diversas actividades, numa máquina monoprocessador (um único CPU) não pode ser considerada no sentido estrito do termo; num determinado instante, o processador apenas pode executar uma instrução para uma dada actividade. Quando se considera um grão de tempo muito fino, o paralelismo não é real, daí a designação de **pseudoparalelismo** ou pseudoconcorrência para designar os sistemas operativos multiprogramados que se executam num computador com um único processador.

Para ilustrar o conceito de pseudoparalelismo nos sistemas operativos, é vulgar efectuar uma analogia com a organização do trabalho em algumas profissões, como seja o secreta-

4 - GESTOR DE PROCESSOS

Neste capítulo vamos analisar a forma como o modelo computacional dos processos até agora apresentado é implementado no núcleo do sistema operativo. Os mecanismos que permitem criar e gerir os processos fazem parte do gestor de processos que constitui a camada mais interna do núcleo e cuja funcionalidade é necessária para todas as outras.

O capítulo focará dois aspectos essenciais na implementação do gestor de processos. O primeiro é a forma como é implementada a pseudoconcorrência: como são os processos representados internamente ao núcleo, de que forma é feita a comutação entre processos, e que operações efectuam as chamadas sistema relacionadas com a gestão de processos. O outro aspecto essencial no gestor de processos está relacionado com os algoritmos que determinam a escolha dos processos que se executam em cada instante, denominado o algoritmo de escalonamento. Como veremos, a política de escalonamento de um sistema operativo é crítica para o desempenho global do sistema, devendo portanto ser cuidadosamente estudada.

4.1 ARQUITECTURA DO GESTOR DE PROCESSOS

Como já dissemos, a existência de vários processos, numa máquina monoprocesador, só é possível através da multiplexagem no tempo da sua actividade, atribuindo a cada um, durante períodos de tempo limitados, o recurso único que constitui o processador. O gestor de processos encarrega-se de efectuar transparentemente todas as operações, permitindo ao programador considerar que o processo se executa continuamente.

Na arquitectura genérica que introduzimos no Capítulo 2, o gestor de processos é a componente do núcleo encarregue de todos os aspectos do controlo do ciclo de vida dos processos e que inclui todas as chamadas sistemas com eles relacionadas.

É habitual subdividir o gestor de processos em três unidades funcionais:

- Comutação de processos;
- Escalonamento da execução dos processos;
- Chamadas sistema de gestão dos processos que determinam o ciclo de vida e o estado de execução dos processos.

5 - SINCRONIZAÇÃO: SECÇÕES CRÍTICAS

Neste e no próximo capítulo abordamos a sincronização dos processos e tarefas, que, como veremos, é imprescindível na programação concorrente. Apesar do tema da sincronização obrigar a uma abordagem integrada, quando se programa existem dois aspectos que são claramente distintos: a competição por recursos e a cooperação entre actividades. No texto optámos por separá-los em dois capítulos para facilitar a leitura, procurando que no final se perceba a forte integração dos dois.

Neste capítulo abordamos o problema dos acessos concorrentes a estruturas de dados por mais do que uma tarefa e da necessidade de garantir que esses acessos se efectuem dentro de secções críticas.

O capítulo começa por motivar a necessidade de sincronizar tarefas quando partilham recursos e, em seguida, apresenta os requisitos de uma solução que permita suportar de forma correcta as secções críticas em programação concorrente. Descreve-se um conjunto de soluções, a partir da mais simples programada ao nível aplicacional, até à incorporação no núcleo de mecanismos de sistema operativo.

O capítulo termina com uma descrição dos mecanismos de sincronização dos sistemas operativos Linux e Windows disponíveis para os programadores definirem secções críticas, e ainda com uma descrição de como é que no núcleo são resolvidos os problemas básicos de sincronização.

5.1 NECESSIDADE DAS SECÇÕES CRÍTICAS

As tarefas que se executam em paralelo não são normalmente independentes, tendo de se sincronizar para poderem executar um algoritmo global comum. Existem diversos motivos que justificam a necessidade de sincronizar explicitamente essas tarefas. Para simplificar a sua análise agrupamo-los em duas categorias: competição de várias tarefas por um determinado recurso e cooperação entre tarefas. A primeira constitui o tema principal deste capítulo; a segunda categoria é analisada no capítulo seguinte.

Considere-se o exemplo seguinte: uma conta bancária a que várias tarefas pretendem aceder ou actualizar. A Figura 5.1 ilustra uma função em linguagem C, que poderia fazer parte de um programa que permite efectuar várias operações de índole bancária. Esta

6- PROGRAMAÇÃO CONCORRENTE

No capítulo anterior abordámos a sincronização, tratando a exclusão mútua, um mecanismo nuclear em todas as situações em que actividades paralelas partilham um recurso. Contudo, a exclusão mútua é normalmente apenas um dos aspectos da sincronização numa aplicação concorrente.

Neste capítulo, alargamos o âmbito da utilização da sincronização, procurando identificar os principais mecanismos que permitem programar algoritmos de cooperação entre processos e tarefas. Para ilustrar a programação concorrente, descrevemos alguns dos problemas “clássicos” de sincronização e a forma como podem ser resolvidos. Estes problemas correspondem a arquétipos de algumas das situações mais vulgares encontradas na programação concorrente, razão pela qual se justifica o seu estudo mais detalhado.

Dada a importância da sincronização, os sistemas operativos oferecem diversos conceitos aos programadores; neste capítulo descrevemos os diversos objectos de sincronização que habitualmente os sistemas disponibilizam. Apesar de semelhantes, existem diferenças na semântica de utilização que é fundamental compreender. Em seguida ilustramos alguns destes tipos de objectos de sincronização com exemplos de objectos do Unix e Windows.

Finalmente abordamos o conceito de monitor que corresponde a um conceito unificador da sincronização, procurando integrar o suporte à programação de estruturas de dados partilhadas, garantindo as secções críticas e a sincronização explícita em função do estado do monitor. Os monitores nasceram nas linguagens de programação concorrentes e a sua adaptação ao modelo computacional do sistema operativo implica sempre considerar um determinado tipo de ambiente de programação. A razão de os tratarmos neste texto reside no facto de serem o mecanismo de sincronização escolhido pelos ambientes J2EE e .NET, pelo que ilustram a transição entre os modelos exclusivamente baseados nas chamadas sistema e os mais sofisticados oferecidos pelas máquinas virtuais.

6.1 COOPERAÇÃO ENTRE PROCESSOS

A cooperação entre processos, na sua forma mais simples, pode ser entendida como a cooperação entre pessoas que realizam uma actividade conjunta. Numa situação típica de trabalho cooperativo, um participante tem uma tarefa para realizar, de cuja conclusão outros dependem para continuar. Os exemplos são óbvios: num projecto de um edifício,

7 - MECANISMOS DE GESTÃO DE MEMÓRIA

O módulo de gestão de memória do sistema operativo efectua a gestão e optimização da memória física, suporta a memória virtual dos processos e implementa um conjunto de algoritmos associados à manipulação do espaço de endereçamento dos processos, nomeadamente a transparência da informação entre a memória primária e secundária.

Podemos considerar o módulo de gestão memória segundo duas facetas: mecanismos de gestão de memória e algoritmos de gestão de memória. Os algoritmos serão abordados no capítulo seguinte. Eles determinam que decisões devem ser tomadas e quando devem ser tomadas, usando os mecanismos de baixo nível para as levar a cabo.

Os **mecanismos de gestão de memória**, descritos neste capítulo, determinam a organização da memória do computador, ou seja, se o seu endereçamento é real ou virtual, se a memória é segmentada ou paginada, e qual o tamanho das páginas ou segmentos. Estes mecanismos são, em geral, executados pelo *hardware* de gestão de memória do processador, devidamente programado pelo sistema operativo.

Neste capítulo é apresentado o modelo computacional da gestão de memória salientando as operações que permitem alterar o espaço de endereçamento dos processos como resultado da alocação e libertação de memória. Com efeito, como já foi antes referido, o espaço de endereçamento modifica-se durante a execução dos programas. A pilha varia dinamicamente e a zona de dados pode ser alterada através da criação dinâmica de estruturas de dados (por exemplo, `malloc` nos programas em C). O sistema operativo controla estas variações de modo a garantir sempre o isolamento dos espaços de endereçamento.

7.1 ESPAÇO DE ENDEREÇAMENTO DE UM PROCESSO

Em capítulos anteriores apresentámos as noções de programa e processo. Foi na altura dito que um programa executa as suas instruções num determinado espaço de memória associado ao processo.

DESIGNA-SE POR ESPAÇO DE ENDEREÇAMENTO DE UM PROCESSO O CONJUNTO DE POSIÇÕES DE MEMÓRIA QUE UM PROGRAMA EXECUTADO POR ESSE PROCESSO PODE REFERENCIAR.

O espaço de endereçamento está associado ao processo, sendo toda a informação relevante para a sua definição guardada ou acessível através do contexto do processo. O espaço

8 - ALGORITMOS DE GESTÃO DE MEMÓRIA

Os algoritmos de gestão de memória, que são da exclusiva responsabilidade do sistema operativo, determinam que decisões devem ser tomadas e quando devem ser tomadas, usando os mecanismos básicos de memória virtual para as executar. Em particular, definem como, quando e quais os blocos de memória que devem ser alocados em memória primária, transferidos entre a memória principal e a memória secundária e qual o espaço de endereçamento que um processo deve ter disponível na memória primária.

No final do capítulo são apresentados os algoritmos de gestão de memória usados nos sistemas operativos Linux e Windows.

8.1 INTRODUÇÃO

Os algoritmos de gestão de memória são utilizados para decidir:

- Onde se deve colocar um bloco (segmento ou página) de programa dada a memória primária livre;
- Quando transferir um bloco de memória secundária para memória primária e vice-versa;
- Que bloco retirar de memória quando não existe mais memória primária livre ou quando a que existe disponível é inferior a um determinado valor considerado mínimo para o bom funcionamento do sistema.

Ao contrário dos mecanismos de endereçamento vistos no capítulo anterior, que dependem fortemente da arquitectura do processador, os algoritmos de gestão de memória dependem fundamentalmente do sistema operativo. Podemos ter, por exemplo, políticas semelhantes de gestão de memória realizadas sobre *hardware* diferente (a mesma versão de Linux existe sobre diversas máquinas) ou podemos ter sobre o mesmo *hardware* políticas diferentes de gestão de memória (num processador Intel tanto pode correr Windows como Linux, por exemplo).

Aos três tipos de decisões que o sistema operativo tem de tomar em relação à memória vamos designá-los por:

- **Alocação** – Onde colocar um bloco na memória primária;

9 - SISTEMA DE FICHEIROS

Sendo os sistemas de ficheiros elementos fundamentais de um sistema operativo, frequentemente, a introdução aos computadores é efectuada pela compreensão da estrutura de ficheiros e directórios criada pelo sistema de ficheiros.

O principal objectivo dos sistemas de ficheiros é simplificar a organização, o acesso, a partilha e a gestão de grandes volumes de informação de forma eficiente. Frequentemente, o sistema de ficheiros é confundido com os dispositivos de memória de massa (vulgarmente chamados discos). De facto, os sistemas de ficheiros são utilizados fundamentalmente para gerir e organizar a informação guardada em dispositivos deste tipo. No entanto, é possível utilizar sistemas de ficheiros para gerir informação guardada em memória, tal como é possível imaginar um sistema operativo que não use um sistema de ficheiros para gerir a informação contida nos dispositivos de memória de massa.

Como para outras funções do sistema operativo, era possível que as aplicações gerissem a informação a manter em memória secundária directamente, mas tal complicaria significativamente a partilha de informação e multiplicaria o esforço de programação de cada uma das aplicações.

Os sistemas de ficheiros são usualmente constituídos por: um conjunto de estruturas armazenadas em memória secundária; um conjunto de métodos – as chamadas sistemas – que sabem aceder e gerir a informação; e um conjunto de estruturas em memória primária que permitem otimizar o acesso à informação em memória secundária.

O modelo computacional do sistema de ficheiros e as estruturas que o suportam não são, por vezes, os mais indicados para gerir a informação em memória secundária, em inúmeras situações é necessário utilizar estruturas e modelos computacionais mais complexos, como aqueles presentes nos sistemas de gestão de bases de dados (SGBD), mas a simplicidade do modelo computacional dos sistemas de ficheiros e a sua popularidade entre os utilizadores faz com que muitas aplicações o utilizem preferencialmente.

9.1 ORGANIZAÇÃO DO SISTEMA DE FICHEIROS

Um sistema de ficheiros é composto por um conjunto de entidades fundamentais, um sistema de organização de nomes para identificação dos ficheiros, uma interface programática para comunicação entre os processos e o sistema de ficheiros. Esta secção tem por objectivo descrever cada uma destas componentes.

10 - COMUNICAÇÃO ENTRE PROCESSOS

No Capítulo 6 foram apresentados algoritmos que, fazendo uso da sincronização, permitem orquestrar, de forma estruturada e clara, o modo como diferentes tarefas e processos podem cooperar entre si para a realização de uma determinada actividade.

As soluções apresentadas, por si só, são suficientes quando a cooperação entre tarefas e processos envolve apenas manipulação de reduzidas quantidades de informação, normalmente específicas à sincronização, e quando partilham um mesmo espaço de endereçamento.

Contudo, na maioria das situações, a cooperação entre tarefas e processos envolve também a transferência de dados, de carácter mais geral e de volume variável, que são gerados numa tarefa ou processo para serem processados noutro(s). Designa-se normalmente como produtor, o processo ou tarefa que gera, produz e envia a informação, e como consumidor, aquele que recebe, consome, e processa a informação. Tendo por base este modelo predefinido, os sistemas operativos oferecem mecanismos específicos para a comunicação entre processos que simplificam a programação.

Um caso particularmente relevante desta situação é aquele em que as várias tarefas e processos, produtores e consumidores, residem e se executam em computadores diferentes. Actualmente, são inúmeros os exemplos de aplicações e sistemas que cooperam entre si transferindo grandes volumes de dados, entre computadores, quer no âmbito de uma rede local, quer envolvendo grandes distâncias geográficas. Apenas para mencionar as mais divulgadas, referem-se aquelas que fazem uso de protocolos para a *World Wide Web* (HTTP, HTTPS), de transferência de ficheiros (FTP), de estabelecimento de sessões interactivas de terminal remotas (Telnet, SSH), transmissão de mensagens electrónicas (SMTP), aplicações de partilha de ficheiros (*Peer-to-Peer* – P2P) em larga escala.

Os dados transmitidos entre as aplicações são geralmente opacos para o sistema operativo, na medida em que este apenas se encarrega de os recolher do **produtor** (emissor) e entregá-los ao **consumidor** (receptor), ignorando a sua organização e estrutura internas (que podem ir desde simples sequências de caracteres ou valores numéricos até documentos estruturados, ou grafos de objectos serializados, quer em XML quer em codificação binária). Cabe aos processos acordar o formato que os dados devem respeitar, bem como o protocolo a que obedece a troca dos mesmos. Estas operações são efectuadas recorrendo aos mecanismos de comunicação entre processos que são disponibilizados pelo siste-

1 1- ENTRADAS/SAÍDAS

A comunicação entre o computador e os seus periféricos é fundamental para a construção de todos os sistemas informáticos. Sem esta comunicação o computador seria ainda hoje uma simples máquina de cálculo com fraca ligação ao mundo que o rodeia. As necessidades de comunicação entre computador e periféricos têm vindo a aumentar significativamente nas últimas décadas, pelo que o subsistema de Entradas/Saídas (E/S) tem vindo a sofrer alterações significativas.

Nos primórdios da utilização de computadores, estes recebiam os dados de entrada em bandas magnéticas e entregavam os resultados dos cálculos efectuados também em bandas magnéticas. Actualmente, os computadores têm que interagir com um elevado número de tipos de periféricos de E/S, desde a simples impressora, até à máquina fotográfica, passando pelo teclado, monitor, rato, disco, DVD, microfone, altifalantes, etc. As redes de comunicação tornaram-se também omnipresentes, sendo frequente um computador ter ligações a diversos tipos de redes. Ao contrário dos computadores iniciais, actualmente é necessário que os computadores se adaptem dinamicamente aos periféricos que estão disponíveis em cada momento, pois o número e a frequência com que os periféricos ficam disponíveis ou indisponíveis para um computador não são compatíveis com a sua instalação e remoção manual.

1 1.1 OBJECTIVOS DO SUBSISTEMA DE E/S

Para facilitar a comunicação entre aplicações e toda a multiplicidade de periféricos, os sistemas operativos fornecem um modelo de E/S que permite o estabelecimento de canais virtuais com os periféricos, através dos quais os processos recebem e enviam informação que pode ser gerada, armazenada ou mostrada pelo periférico físico.

Esta descrição funcional esconde a complexidade real que a implementação das E/S envolve. A maior dificuldade advém da grande diversidade de periféricos existentes e das suas características e modos diferenciados de funcionamento. As operações executadas pelos periféricos são extremamente variadas e, conseqüentemente, as sequências de controlo e códigos de erro que lhes estão associadas também são muito diferentes. As próprias características físicas da interacção também variam de periférico para periférico, por exemplo:

- A unidade de transferência de informação pode ser *byte*, cadeia de caracteres, ou blocos de tamanho fixo;

12 - SEGURANÇA

A segurança é um requisito fundamental para que os sistemas operativos possam fornecer os seus serviços e gerir a informação que lhes é confiada. Na visão actual, um sistema operativo deve garantir a **confidencialidade, integridade e disponibilidade** da informação que gere.

A segurança dos sistemas operativos é assegurada por **serviços específicos de segurança**, tais como os serviços de autenticação, de controlo de acessos, de confinamento e de auditoria. Estes serviços actuam em conjunto de modo a garantir a confidencialidade, integridade e disponibilidade da informação manipulada pelo sistema operativo.

Os primeiros sistemas operativos não possuíam qualquer serviço de segurança intrínseco. A segurança da informação e dos serviços disponibilizados nos sistemas dessa época era essencialmente física. Nos sistemas típicos de tratamento em lotes, o computador era colocado numa sala segura só acessível a entidades autorizadas, os programadores e os utilizadores depositavam os seus trabalhos, em cartões ou em banda magnética, e a equipa de gestão do sistema é que se encarregava de toda a interacção com o computador.

Com o aparecimento de sistemas operativos interactivos, multi-utilizador, a segurança revelou-se um requisito fundamental. Nestes sistemas vários utilizadores partilham em simultâneo um mesmo recurso físico – o computador. O sistema operativo na sua qualidade de gestor do computador tem de ser capaz de garantir a confidencialidade e integridade da informação pertencente a cada utilizador bem como a não interferência entre os serviços e aplicações executadas em benefício de cada um.

O sistema operativo Multics [Organick, 1972] foi um dos primeiros a possuir um modelo de segurança completo que assegurava a confidencialidade, integridade e disponibilidade da informação e dos serviços num ambiente multi-utilizador. O modelo de segurança dos sistemas operativos actuais é ainda basicamente o mesmo do Multics, embora com desenvolvimentos que os tornam mais flexíveis e mais facilmente geríveis.

12.1 OBJECTIVOS DE SEGURANÇA

O objectivo da segurança informática é assegurar a confidencialidade, integridade e disponibilidade da informação e dos serviços fornecidos pelo sistema operativo. A confidencialidade e a disponibilidade são propriedades de simples definição e compreensão: