2 Conceitos básicos de sistemas operacionais

2.1 Introdução e definição

UM sistema de computação é composto por um ou mais processadores, uma certa quantidade de memória, terminais, discos magnéticos, *interfaces* de rede e dispositivos de I/O etc., ou seja, é um sistema extremamente complexo, e desenvolver *software* que gerencie e integre esses componentes, fazendo-os trabalhar corretamente e de forma otimizada, não é tarefa fácil.

O sistema operacional é uma camada de *software* colocada entre os recursos de *hardware* e os programas que executam tarefas para os usuários, sendo que uma de suas principais responsabilidades é permitir e promover o acesso aos periféricos, sempre que um programa solicitar uma operação de I/O. Assim, por meio dessa intervenção do sistema operacional, o programador não precisa conhecer detalhes do *hardware*, pois informações de como, por exemplo, "enviar um caracter para a impressora", são internas ao sistema operacional. Além disso, como todos os acessos aos periféricos são feitos através do sistema operacional, fica mais fácil controlar a disponibilização dos recursos, buscando uma distribuição justa, eficiente e conveniente, não só dos recursos de I/O, mas de todo o sistema de computação.

Uma utilização mais eficiente do sistema de computação é obtida a partir da distribuição de seus recursos entre os programas, por exemplo, a distribuição do espaço em disco, da memória principal, do tempo do processador, do acesso a disco etc. Já a utilização mais conveniente, é obtida a partir da disponibilização dos recursos do sistema de computação, sem que os usuários conheçam os detalhes dos recursos. Por exemplo, supondo que um usuário especializado, um programador, que, ao desenvolver um programa, precisa colocar um caracter na tela. Para tanto, em geral, é necessária toda uma seqüência de acessos à *interface* do vídeo, diversos registradores devem ser lidos ou escritos e, além disso, podem existir diferentes tipos de *interfaces* que exigirão diferentes seqüências de acesso. Porém, por meio do sistema operacio-

nal, o programador apenas informa, no programa, qual caracter deve ser colocado na tela e todo o trabalho de acesso ao periférico é feito pelo sistema operacional.

Ao esconder detalhes dos periféricos, muitas vezes são criados recursos de mais alto níveis, ou seja, as abstrações, como, por exemplo, os arquivos. Os programas utilizam o espaço em disco através do conceito de arquivo. Arquivos não existem no *hardware*, mas são um recurso criado a partir do que o *hardware* oferece. Para o programador é muito mais confortável trabalhar com arquivos do que receber uma área de espaço em disco que ele próprio teria que organizar. Outro exemplo de abstração, pode ser uma instrução de I/O, tal como, *read* ou *write* em um IBM PC, que deve ser acompanhada de 13 parâmetros, especificando o endereço do bloco a ser lido, o número de setores por trilha, o modo de gravação no meio físico, o espaçamento entre setores etc., sem contar com as tarefas de ordem física/mecânica, ou seja, verificar se o motor do *drive* já está acionado etc.

A grande maioria dos programadores não se envolve com tais detalhes, pois lida com uma abstração de alto nível e, portanto mais simples. No exemplo em questão, a abstração feita nos discos é visualizá-los como uma coleção de arquivos identificados por nomes, onde os eventos, a manipulação dos arquivos, não consideram maiores detalhes e restringem-se a simplesmente abrir, ler/escrever e fechar.

2.2 Arquitetura de sistemas operacionais

A arquitetura de um sistema operacional é a estrutura básica sobre a qual é projetado o sistema operacional, de como as abstrações são realmente implementadas, como o sistema operacional deve ser solicitado e atender aos aplicativos, como interagem as partes do sistema operacional entre si e como o sistema operacional responde às solicitações dos aplicativos.

2.2.1 Arquitetura monolítica

É a arquitetura mais antiga e mais comum. Cada componente do sistema operacional é contido no núcleo (*kernel*) e pode comunicar-se com qualquer outro componente diretamente. Essa intercomunicação direta permite rapidez na resposta de sistemas operacionais monolíticos, entretanto, como núcleos monolíticos agrupam os componentes todos juntos, torna-se difícil identificar a origem de um determinado problema ou erro. Além disso, todo o código do sistema operacional é executado

com acesso irrestrito ao sistema, o que pode facilitar a ocorrência de danos provocados intencionalmente, ou não, por outros aplicativos.

2.2.2 Arquitetura em camadas

À medida que os sistemas operacionais tornaram-se mais complexos e maiores, projetos puramente monolíticos tornaram-se inviáveis e, então a arquitetura em camada, ou modular, tornou-se uma boa opção, agrupando "camadas" de componentes, ou seja, conjunto de procedimentos, que realizam tarefas similares. Cada camada comunica-se somente com as suas camadas imediatamente inferior e superior. Uma camada inferior sempre presta um serviço à sua camada superior, sendo que a camada superior não sabe como o serviço é feito, apenas o solicita. A implementação de uma camada pode ser modificada sem exigir modificação em outra camada, pois possuem componentes autocontidos[1]. Em uma abordagem em camadas, a solicitação de um serviço pode precisar passar por muitas camadas antes de ser atendida, assim o desempenho se degrada em comparação ao de núcleos monolíticos.

2.2.3 Arquitetura de micronúcleo

A arquitetura de micronúcleo (*microkernel*), representada na Figura 8, também, é uma forma de arquitetura modular ou em camadas. Na tentativa de reduzir os procedimentos mais fundamentais, somente um pequeno número de serviços, tais como, parte do gerenciamento de memória, a sincronização entre processos e a comunicação entre processos, terá acesso direto ao *hardware*. Por sua vez, o serviço de comunicação entre processo, que está dentro do micronúcleo, é o responsável por habilitar os serviços de, por exemplo, redes, sistemas de arquivos e gerenciamento de dispositivos, que, normalmente, podem ser implementados no núcleo do sistema operacional, não no micronúcleo, ou até como procedimentos (aplicativos) externos ao núcleo. Alguns sistemas operacionais permitem que aplicações acessem diretamente os serviços oferecidos pelo micronúcleo. Por ser similar a arquitetura modular, a arquitetura em micronúcleo possui, em geral, as mesmas vantagens e desvantagens da arquitetura em camadas.

[1] Cada componente oculta o modo de como realiza sua tarefa e apresenta uma *interface* padrão para que outros componentes possam requisitar seus serviços.

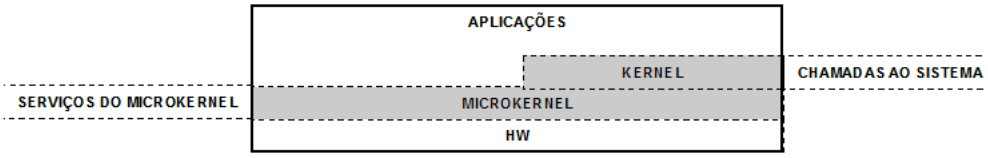

Figura 8: Representação da organização de um sistema operacional com arquitetura de *microkernel*.

2.3 Chamadas ao sistema

As chamadas ao sistema[2] (*system calls*) são interrupções (*traps*) originadas por *software*, que fornecem a *interface* entre processos e o sistema operacional. Os programas solicitam serviços ao sistema operacional por meio de chamadas ao sistema que provocam interrupções da execução de determinada tarefa para que uma outra atividade seja executada. Uma atividade protegida por uma interrupção é específica do sistema operacional e, portanto, tem privilégio sobre a anterior, que, posteriormente, será restaurada.

As chamadas aos sistema são semelhantes às chamadas de sub-rotinas, entretanto, enquanto que as chamadas de sub-rotinas são transferências para procedimentos do mesmo programa que fez a chamada, as chamadas ao sistema transferem a execução para o sistema operacional, e, por meio de parâmetros, o programa solicitante, informa exatamente o que necessita. Desse modo, todos os serviços que o sistema operacional disponibiliza para os aplicativos são atendidos pelas chamadas ao sistema. Por exemplo, se um programa lista o conteúdo de um arquivo texto na tela, com certeza fará uma chamada ao sistema para verificar se o arquivo existe e um dos parâmetros dessa chamada será nome do arquivo a ser listado. Além disso, tanto o procedimento de leitura do arquivo, quanto o de escrita no vídeo, também correspondem a chamadas ao sistema.

A implementação das chamadas ao sistema, ou seja, a implementação dos procedimentos que correspondem a essas, é feita no núcleo do sistema operacional, pois cada chamada ao sistema corresponde a um procedimento de uma biblioteca de procedimentos do sistema operacional, que têm sua execução de forma privilegiada. Em geral, os aplicativos somente conseguem acessar esses procedimentos através das chamadas ao sistema.

Ao ser executado, o procedimento coloca os parâmetros das chamadas ao sistema nos locais específicos – por exemplo, nos registradores – e é executado a partir de uma interrupção, denominada *trap*, que faz a execução protegida do procedi-

[2] Alguns autores dizem "chamadas de sistema".

mento, isto é, em modo *privilegiado*[3]. A interrupção, no caso de chamadas ao sistema, é gerada por um pedido específico de um aplicativo para que um serviço do sistema operacional seja executado.

O procedimento de atendimento a uma solicitação de serviço ocorre da seguinte forma:

1. o aplicativo solicita o serviço ao sistema operacional, que será atendido por meio da execução de uma ou mais chamadas ao sistema;
2. cada chamada ao sistema corresponde a execução de um procedimento no núcleo do sistema operacional, que inicialmente coloca os parâmetros da chamada - por exemplo, um nome de arquivo - nos devidos lugares;
3. em seguida é emitida uma *trap*, que provocará a intervenção direta do sistema operacional, por meio da execução do procedimento específico, correspondente à chamada ao sistema. A solicitação será atendida se os parâmetros forem válidos e existirem recursos para tal;
4. após a execução da solicitação, o sistema operacional coloca em um registrador um código de estado (*status*), indicando se operação foi bem sucedida ou se falhou;
5. o sistema operacional devolve o controle ao processo que provocou a chamada ao sistema, que terá acesso ao código de estado gerado.

As chamadas ao sistema variam entre sistemas operacionais, pois os sistemas operacionais mais específicos para uma determinada atividade podem possuir chamadas específicas para essa atividade. Entretanto, algumas chamadas ao sistema são comuns entre os sistemas operacionais, por exemplo:

- chamadas ao sistema para gerenciamento de processos: é um grupo de chamadas extremamente importante sendo responsável por criar e finalizar processos;
- chamadas ao sistema para tratamento de sinais: seus serviços são utilizados para sincronizar processos, suspender a execução de processos etc.;
- chamadas de gerenciamento de arquivos: possibilita a criação de arquivos, a leitura, a gravação, a alteração de nomes de arquivos etc.;

[3] A grande maioria dos sistemas operacionais podem trabalhar em dois modos: o modo *privilegiado*, também chamado de modo *supervisor*, modo *sistema* ou modo *mestre*, e o modo *não-privilegiado*, também chamado de modo *usuário*, modo *programa* ou modo *escravo* citeTOC03.

- chamadas de gerenciamento de diretório e de sistema de arquivos: possibilita a criação de um novo diretório, a remoção de um diretório existente, criação de uma nova entrada no diretório (ou seja, a inclusão de um novo arquivo no diretório), "montar e desmontar"[4] um sistema de arquivos etc.;
- chamadas ao sistema de proteção: permite que se descubra e que sejam alterados os donos de processos e arquivos;
- chamadas de serviços de gerenciamento de tempo: trata as datas e as horas em um sistema de computação, assim como permite a contabilidade do tempo de utilização por parte de um usuário ou processo;

2.4 Interpretador de comandos

UM dos programas que mais executam chamadas ao sistema é o interpretador de comandos e, como exemplo, será apresentado o *shell*, que é um interpretador de comandos do Unix.

Apesar do interpretador de comandos, em geral, não fazer parte do núcleo do sistema operacional, possui a importante tarefa de prover a *interface* primária entre o usuário e o sistema operacional, sem a qual, muitas vezes, não seria possível para o usuário acessar o sistema de computação. Pode-se visualizar o interpretador de comandos como uma cápsula circundando o sistema operacional, ou seja, criando uma camada, uma *interface*, entre o usuário e o sistema operacional, permitindo que o usuário execute comandos para acessar todos os recursos que estiverem disponíves no sistema de computação. Os usuários comunicam-se com o interpretador de comandos através de entrada de comandos, provocando eventos por meio do *mouse* ou por linha de comando, que são interpretados e geram as solicitações de serviços.

Um interpretador de comandos simples é basicamente um programa composto de um laço que aguarda uma entrada (um comando), cria um novo processo, com uma chamada ao sistema (no Unix ou no Linux seria a chamada *fork*), atribui a esse novo processo o programa especificado na entrada e executa o processo (no Unix ou no Linux, tanto a atribuição do programa quanto a execução do processo são feitos pela chamada *exec*). Em seguida, aguarda a finalização do processo e, então, espera por uma nova entrada. Veja o Algoritmo 1

[4] Montar um disco significa designar uma identificação (nome, rótulo ou volume) para um disco ou parte desse. Desmontar, significa retirar essa designinação

```
enquanto (TRUE) faça
    ler comando (comando, parâmetros);
    fork ();
    se (criação de novo processo == ok) então
        exec (comando, parâmetro);
    senão
        "tratamento da falha";
    fim
fim
```

Algoritmo 1: Um interpretador de comandos simplificado.