Unidade 08: Introdução aos Sistemas Operacionais

Prof. Daniel Caetano

Objetivo: Apresentar a lógica básica de funcionamento de um sistema operacional, de maneira a facilitar a compreensão do funcionamento da CPU.

Bibliografia:

- MACHADO, F. B; MAIA, L. P. Arquitetura de Sistemas Operacionais. 4ª. Ed. São Paulo: LTC, 2007.
- TANENBAUM, A. S. Sistemas Operacionais Modernos. 2ª.Ed. São Paulo: Prentice Hall, 2003.
- STALLINGS, W. **Arquitetura e organização de computadores**. 5ed. São Paulo: Ed. Pearson Prentice Hall, 2003.
- MURDOCCA, M. J; HEURING, V.P. **Introdução à arquitetura de computadores**. S.I.: Ed. Campus, 2000.

INTRODUÇÃO

* Problema: como tornar um computador útil?
* Quais as tarefas básicas que devem ser oferecidas?

Embora o uso de Sistemas Operacionais seja, hoje, parte do dia a dia da maioria das pessoas, muitas vezes sua função não é clara e é muito freqüente que algumas pessoas confundam as funções de alguns softwares aplicativos com a função do sistema operacional em si. Neste aula será apresentado os conceitos fundamentais que definem "o que é" e "para que serve" um sistema operacional, além de apresentar seus mecanismos mais básicos de funcionamento.

1. O CONCEITO DE "SISTEMA OPERACIONAL"

* Função: executar ou auxiliar a execução de tarefas básicas
    - Ex: Carregar um programa, gerenciar impressão de documento
* Sistema operacional faz tudo?
* O que é?
    - Conjunto de rotinas, em geral de baixo nível
    - Carregador de Programas x Infinidade de Funções
    - Padronização de Acesso a Recursos x Compartilhamento de Recursos

Sempre que se deseja usar um equipamento, algumas tarefas precisam ser executadas: carregar um programa na memória, enviar alguns dados para a impressora ou simplesmente escrever algo na tela. Essas estão entre algumas das funções para as quais um sistema operacional é projetado, seja para executá-las em sua completude, seja para facilitar a execução de algumas destas tarefas por parte do usuário ou de outros programas.

O nível de complexidade de um sistema operacional pode variar enormemente, assim como o nível de complexidade de utilização do mesmo. Isto significa que um sistema operacional pode fazer "mais" ou "menos" pelos usuários e pelos programas que nele irão ser executados. Independentemente de sua complexidade, um sistema operacional não passa, entretanto, de um conjunto de rotinas executadas pelo processador, como qualquer outro programa que um usuário possa desenvolver. A diferença fundamental é que, no sistema operacional, estas rotinas são, em geral, rotinas de baixo nível, rotinas que conversam diretamente com o hardware.

Um sistema operacional pode ser tão simples quanto um mero carregador de programas (praticamente o que o DOS era) ou possuir uma infinidade de funções (a maioria dos sistemas operacionais atuais.

1.1. Facilidade e Padronização do Acesso aos Recursos do Sistema

* Como facilitar o acesso a dispositivos?
    - Ex.: gravar um arquivo no HD
    - Como lidar com dispositivos de fabricantes diferentes?
* Virtualização de Dispositivos
    - Atuação como Intermediário
    - Ex.: Read / Write

Considerando que praticamente todo sistema computacional possui um grande número de dispositivos (drive de CD/DVD, monitores, impressoras, scanners etc) e que cada dispositivo tem uma operação bastante complexa, é interessante que exista uma maneira de poder utilizar tais dispositivos sem ter de se preocupar com o grande número de detalhes envolvidos no processo.

Por exemplo: o simples ato de escrever um arquivo em um *harddisk* exige um grande número de operações, envolvendo divisão do arquivo em blocos menores que caibam nos espaços livres disponíveis no disco e atualização das tabelas de diretório... para não falar em todos os detalhes envolvidos com a escrita em si, como posicionalmento da cabeça de gravação no disco, escrita propriamente dita, verificação de erros etc.

Ora, a maioria destas atividades é um processo repetitivo e metódico que não requer qualquer intervenção do usuário. As únicas informações que realmente o usuário precisa fornecer são: o nome do arquivo, quais são os dados a gravar e em que disco ele deseja que

estes dados sejam gravados. Todo o resto pode ser automatizado... e é exatamente o que o sistema operacional faz, neste caso.

Assim, o sistema operacional pode ser visto como uma interface entre o usuário/programa e o hardware, de forma a facilitar o acesso ao recursos do mesmo, como pode ser visto na figura 1.

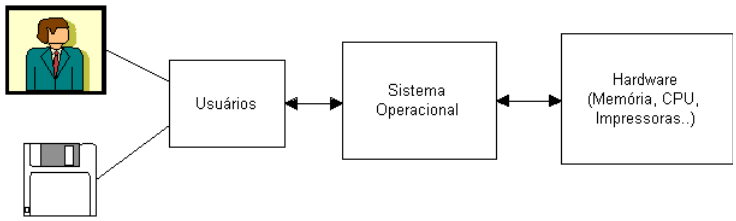

Figura 1 - Sistema Operacional como Interface entre usuários e recursos

Este tipo de "interface" que simplifica o acesso aos dispositivos (e os padroniza, em alguns casos) é também chamado de "**virtualização de dispositivos**". O caso mais comum deste tipo de virtualização são as operações "**read**" e "**write**" que, em geral, servem para ler e escrever em qualquer dispositivo, independente de seu funcionamento interno. Esta função de "virtualização" faz com que muitos usuários enxerguem o sistema operacional como uma "extensão da máquina" ou como uma *máquina extendida*.

É importante ressaltar, entretanto, que compiladores e bibliotecas fazem parte do Sistema Operacional. Embora sejam ferramentas muito importantes para o desenvolvimento, este tipo de software **não** faz parte do mesmo.

1.2. Compartilhamento de Recursos do Equipamento de Forma Organizada

* Compartilhar dispositivos?
* Vários programas tentando imprimir?
    a) Fazer o programa esperar
    b) Receber os dados e aguardar que a impressora esteja livre (spool)
* O que mais compartilhar?
    - Tela, teclado, mouse: múltiplas janelas
    - Disco, Rede?
* Múltiplos usuários
* Sistema Operacional: gerenciador de recursos!

Atualmente, o usuário médio está habituado a poder executar diversos programas ao mesmo tempo, em um mesmo equipamento. Ora, vários destes softwares podem precisar utilizar um mesmo dispositivo simultaneamente e, em geral, cada dispositivo só pode executar uma tarefa de cada vez. Como resolver este problema?

Mais uma vez surge uma das responsabilidades do sistema operacional, que é a de controlar o acesso a dispositivos de forma organizada, de maneira que os usuários deste sistema (no caso, softwares) possam compartilhar recursos. Assim, se um software está usando a impressora e um segundo deseja utilizá-la, o sistema operacional deve agir de uma de duas formas: **a) fazer com que o programa aguarde** ou **b) receber os dados a serem impressos e liberá-los para a impressora assim que ela esteja livre**. No caso de impressoras, é muito comum que a segunda alternativa seja implementada, e o nome deste recurso costuma ser *spool*.

Entretanto, não existem apenas equipamentos em que um usuário possui muitos programas. Em alguns sistemas, múltiplos usuários podem, ainda, estar usando o mesmo equipamento ao mesmo tempo. Neste caso, existe ainda mais a necessidade de um gerenciamento de recursos adequado, já que cada usuário terá seu conjunto de aplicativos, todos executando ao mesmo tempo, com os mesmos dispositivos, na mesma CPU e com a mesma memória.

Estas funções de gerenciamento fazem com que muitos usuários enxerguem o sistema operacional como um *gerenciador de recursos*.

2. GERENCIAMENTO DE PROCESSOS, DISPOSITIVOS E MEMÓRIA

Como os computadores modernos todos - inclusive os mais simples - executam com múltiplos processos, dentre as funções mais importantes do Sistema Operacional estão o gerenciamento de CPU, memória e processos.

2.1 Gerenciamento de CPU / PROCESSOS

Como há muitos processos e poucas CPUs (por simplicidade consideraremos apenas uma), se a CPU executar os programas de forma simplesmente sequencial pode haver um problema sério. Imagine que você abra o seu MSN e o seu programa de tocar MP3. Execução sequencial significaria que você teria que fechar o MSN para poder ouvir a música... ou desligar a música para usar o MSN.

Para que isso não ocorra, a CPU faz um **compartilhamento de CPU**. Isso significa que, como não existe uma CPU para cada processo/programa, cada programa poderá usar a CPU por um intervalo de tempo (por exemplo, 32ms) e, depois, terá de esperar pela execução de outros programas.

Esse compartilhmento de CPU é chamado de **Gerenciamento de Processos**. Mas o que seria um processo?

Simplificadamente, um *processo* pode ser definido como um **programa em execução**. Os processos ocupam memória, utilizam entrada e saída, ocupam o processador... e, inclusive, podem possuir *subprocessos* a serem executados simultaneamente.

O sistema operacional precisa permitir que tudo isso funcione em harmonia e, para tal, tem as seguintes funções diante relativas ao gerenciamento de processos:

a) Criação e remoção de processos (de sistema ou do usuário).
b) Suspensão e reativação de processos.
c) Sincronização de processos.
d) Comunicação entre processos.
e) Tratamento de impasses entre processos.

2.2 Gerenciamento de Memória

A memória principal é um dos principais recursos de um computador, sendo o local onde processos e seus dados ficam armazenados. É comum que um computador tenha menos memória principal que o necessário para carregar todos os programas e que, se esta memória não for utilizada de uma maneira ordenada, o equipamento "trave" ou não consiga executar algumas tarefas. Por exemplo: se seu computador tem 2GB e já estão ocupados com programas em torno de 1.75GB, se você quiser abrir uma grande imagem, de 0.5GB, em um trabalho... o computador poderia simplesmente lhe informar: "Falta Memória".

Para que isso não ocorra, o sistema operacional precisa fazer o gerenciamento de memória: como não há memória para todos os programas, aqueles programas que não estão sendo usados com tanta frequência serão transferidos para uma **memória virtual**, isto é, no HD, para que os programas realmente ativos tenham mais espaço. Quando o programa que for para o disco tiver de ser executado novamente, aí ele será trazido de volta à memória principal pelo sistema operacional.

Por outro lado, os processos não devem "saber" disso: eles devem ser executados de maneira que, para cada processo, é como se o equipamento (e a memória) fosse somente sua. Isto significa que todos os processos podem pensar que estão sendo executados no mesmo local da memória (mesmo endreço). Isto certamente não será verdade, mas os processadores modernos possuem recursos para permitir que os processos sejam executados com essa ilusão. Este é um recurso chamado de ***espaço de endereçamento virtual***.

Além dos processos sendo executados pela CPU, também os dispositivos acessam a memória através de mecanismos de DMA, já vistos anteriormente. Assim, o sistema operacional também precisa fornecer algumas funções relativas ao gerenciamento de memória. Algumas delas são:

a) Manter a informação de quais partes da memória estão em uso (e por quem).
b) Decidir quais processos devem ser carregados quando a memória ficar disponível.
c) Alocar e remover processos e dados da memória quando necessário.
d) Controlar o sistema de enedereçamento virtual dos processos.

2.3 Gerenciamento de Dispositivos

Os dispositivos de um computador são, normalmente, dispositivos complexos e, em muitos casos, não funcionariam bem se mais de um processo tentasse usá-los simultaneamente. O que aconteceria, por exemplo, se um programa estivesse imprimindo e outro tentasse imprimir ao mesmo tempo?

Certamente teríamos uma impressão toda embaralhada, com partes de um documento e partes de outro. Como não é isso que ocorre, alguma coisa deve ser feita pelo computador ou pelo sistema operacional para evitar esse problema. De fato, esse é o chamado **gerenciamento de dispositivos**.

O gerenciamento de dispositivos, dentre outras coisas, controla quantos processos podem acessá-lo e em qual instante. Dependendo da situação o dispositivo pode aparecer como indisponível, solicitar que o programa aguarde por sua vez ou, simplesmente, receber os dados e processá-los da forma adequada.

Mas o gerenciamento de dispositivos inclui uma outra parte muito importante.

Na aula de dispositivos vimos que cada dispositivo costuma receber um número para o computador, indicando a **porta** em que aquele dispositivo foi ligado. Se um programa quer escrever a instrução 0x37 em um dispositivo que está na porta 0x88A0, ele usa uma instrução do tipo OUT, que teria uma sintaxe parecida com esta:

```
OUT    (0x88A0),0x37
```

Em linguagem C, existe a instrução **outp**, que, para a mesma finalidade, teria a seguinte sintaxe:

```
outp (0x88A0,0x37)
```

Ora, quem é que escolheu a porta 0x88A0 para o dispositivo? Existem várias possibilidades mas, em geral, que escolhe este número é o **fabricante**. E o que significa a instrução 0x37? Certamente só o **fabricante** sabe. Por exemplo: em uma placa de vídeo nVidia isso pode significar "**desenhe uma reta**" e em uma placa da ATI isso pode singificar "**desenhe um ponto**". Como é que o sistema operacional vai saber todas essas informações?

A resposta é...

NÃO VAI!

O sistema operacional espera que o fabricante forneça uma **biblioteca** chamada **driver de dispositivo** que sabe todas essas informações. O código do driver da nVidia que desenha a reta seria algo como:

```
void linha(int x0, int y0, int x1, int y1, int cor) {
        OUT (0x88A0, 0x37);             /* Comando */
        OUT (0x88A0, x0);               /* Dados Esperados */
        OUT (0x88A0, y0);
        OUT (0x88A0, x1);
        OUT (0x88A0, y1);
        OUT (0x88A0, cor);
}
```

Já no **driver** da ATI a mesma função poderia ser algo como:

```
void linha(int x0, int y0, int x1, int y1, int cor) {
        OUT (0x0633, x0);               /* Dados */
        OUT (0x0634, y0);
        OUT (0x0635, x1);
        OUT (0x0636, y1);
        OUT (0x0637, cor);
        OUT (0x0638, 0x42);             /* Comando */
}
```

Observe que não apenas os números, mas a forma de trabalhar dos dispositivos pode ser diferente: no primeiro caso, apenas **uma porta** foi usada, e todos os dados foram mandados para lá. Neste caso, a ORDEM é importante! No segundo caso, foram usadas **6 portas** diferentes e a única "ordem" importante é que o comando deve ser dado por último.

O sistema operacional, por sua vez, quando quer desenhar uma linha na tela, simplesmente usa o comando:

```
linha(x0, y0, x1, y1, cor);
```

E quem deve fazer o serviço é o driver.

Quem faz o driver? A fabricante do dispositivo... ou seja: **engenheiros eletrônicos** e **engenheiros de computação**.

3. CONCLUSÕES

Os Sistemas Operacionais têm uma função fundamental nos equipamentos de processamento modernos.

Nesta aula vimos em linhas gerais o que um Sistema Operacional faz e que a função de criar os drivers de dispositivos é dos frabricantes de dispositivos.

Na próxima aula veremos com o que o engenheiro eletrônico precisa se preocupar para permitir o gerenciamento de memória e de processos por parte da CPU.

4. BIBLIOGRAFIA

DAVIS, W.S. **Sistemas Operacionais**: uma visão sistêmica. São Paulo: Ed. Campus, 1991.

MACHADO, F. B; MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 4ª. Ed. São Paulo: LTC, 2007.

SILBERSCHATZ, A; GALVIN, P. B. **Sistemas operacionais**: conceitos. São Paulo: Prentice Hall, 2000.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 2ª.Ed. São Paulo: Prentice Hall, 2003.

MURDOCCA, M. J; HEURING, V.P. **Introdução à arquitetura de computadores**. S.I.: Ed. Campus, 2000.

STALLINGS, W. **Arquitetura e organização de computadores**. 5ed. São Paulo: Ed. Pearson Prentice Hall, 2003.