Chamadas de Sistema

Além de **escalonar** o uso do(s) processador(es) para a execução das instruções dos processos, o SO tem a função de **prover serviços** a esses processos. Serviços incluem o **uso** dos dispositivos de E/S, normalmente para **gravar** e **recuperar** dados de arquivos, para **ler** e **escrever** nos terminais de usuário, e para **transmitir** e **receber** dados através da rede de comunicação. Esses serviços do SO são providos para **simplificar** os códigos dos processos, que não precisam preocupar-se com os detalhes da operação dos dispositivos, e também para promover o **compartilhamento ordenado** do uso dos dispositivos. Há também serviços para o gerenciamento de **processos** e de ***threads*** e para a **comunicação** e **sincronização** de suas execuções. Outras tantas funcionalidades adicionais podem ser providas por um SO.

As **rotinas de serviço** oferecidas por um sistema operacional aos programas são denominadas **chamadas de sistema**, ou ***system calls***.

Por questões de desempenho e facilidade de acesso, o **código** executado pelas chamadas de sistema está permanentemente **residente na memória**, pronto para ser usado por qualquer processo que as solicite. Para acessá-lo, normalmente é utilizado um mecanismo baseado em instruções de interrupção (INT), ou instruções específicas do processador para esse fim, como SYSCALL ou SYSENTER. Além de prover uma forma de localizar o código do SO na memória, sem saber previamente o endereço onde cada rotina foi posicionada, as instruções de acesso ao SO também elevam o privilégio de execução do processador.

Em sistemas Linux, por exemplo, o acesso às chamadas de sistema é comumente feito usando uma chamada à instrução de interrupção para o número 80h (128).

Para escrever uma mensagem na tela, usando a chamada de sistema ***write***, o acesso em baixo nível pode ser assim:

```
...
msg    db     "Eu adoro SO!", 0x0a   ; mensagem + caracter de nova linha (newline)
...
                    ; SYSCALL: write (1, msg, 13)
mov eax, 4          ; coloca-se 4 no registrador eax: 4 = número da syscall write
mov ebx, 1          ; coloca-se 1 no registrador ebx: 1 = número do arquivo de saída padrão (stdout)
mov ecx, msg        ; coloca-se o endereço da mensagem no registrador ecx
mov edx, 13         ; coloca-se 13 no registrador edx, indicando o número de bytes a escrever
int 0x80            ; faz-se a chamada de sistema usando instrução de interrupção número 80h
```

Para favorecer a portabilidade dos programas e facilitar as chamadas às rotinas, contudo, cada SO define um conjunto de funções em alguma **linguagem de alto nível**, como C. Cabe a essas funções de alto nível, então, fazer os ajustes necessários em registradores específicos do hardware e usar uma instrução de interrupção (INT), SYSCALL ,ou SYSENTER, para que o acesso efetivo ao código do SO seja realizado. Assim, não é preciso programar em baixo nível para fazer acesso às chamadas de sistema.

E sistemas compatíveis com Unix, as chamadas de sistema são padronizadas. Há diversos padrões associados à *Single Unix Specification* [1], que definem também as chamadas de sistema que devem estar disponíveis. O mesmo tipo de chamadas de sistema está disponível para a plataforma dos SOs MacOS, que atendem às especificações POSIX, além de suas chamadas específicas.

Nos sistemas operacionais Microsoft, uma API chamada WinAPI (Windows API), anteriormente denominada Win32 API, define as chamadas de sistema disponíveis. Usando essa interface, é possível criar aplicações usando C/C++, .NET, or JavaScript. Embora essa API tenha milhares de funções, nem todas as chamadas disponíveis nessa API geram chamadas de sistema. Muitas delas são tratadas dentro do escopo do programa de usuário, como chamadas para gerenciamento de janelas e aspectos da interface gráfica.

Aparentemente, a nova versão do Windows (Windows 8) traz uma grande mudança na implementação das chamadas de sistema. Chamada de WinRT (*Windows Run Time*), a nova API substitui a API Win32. Programas poderão escolher entre uma das duas versões de API para usar chamadas de sistema.

Nem sempre que queremos realizar operações de entrada e saída de dados em nossos programas precisamos recorrer diretamente às chamadas de sistema. Usando bibliotecas (ou classes) de linguagens de alto nível, é comum usarmos funções mais simples e comumente até com mais recursos. Essas funções, ou métodos, muitas vezes fazem uso de mais de uma chamadas de sistema.

[1] The Single Unix Specification.
<http://www.unix.org/what_is_unix/single_unix_specification.html>

[2] Creating Win32-Based Applications (C++). 12/09/2011
<http://msdn.microsoft.com/en-us/library/bb384843.aspx>

[3] Microsoft Win32 to Microsoft .NET Framework API Map. 12/02/2011
<http://msdn.microsoft.com/en-us/library/aa302340.aspx>

Leitura complementar

Para complementar a aprendizagem sobre processos é interessante a leitura da unidade 1.6 do livro “Sistemas Operacionais Modernos” (Tanenbaum).

Tanembaum, A. S. *Sistemas Operacionais Modernos*. Pentice Hall Brasil, 2003.