Hardware e Softwares

Uma das funções principais de um sistema operacional é controlar todos os dispositivos de e/s de um computador, tratar erros, interceptar interrupções, fornecer uma interface entre o dispositivo e o sistema, emitir comandos para os dispositivos.

Os dispositivos de entrada e saída podem ser divididos em um modo genérico como dispositivos de bloco e caractere.

Um dispositivo de bloco armazena as informações em blocos de tamanho fixo, cada qual com seu endereço. Cada bloco pode ser lido ou escrito de maneira independente uns dos outros. Um dispositivo de bloco pode estar com um ponteiro em qualquer lugar e pode ser posicionado para outro cilindro.

Outro dispositivo de e/s é o dispositivo de caractere. O dispositivo de caractere não utiliza estrutura de blocos nem posicionamento. No dispositivo de caractere ele recebe um fluxo de caracteres, além de não ser endereçável.

Os dispositivos de e/s tem uma grande variedade, cada uma trabalha a uma velocidade, assim pressionando o sw a trabalhar com essas diferentes taxas de transferências.

Os relógios não são dispositivos de blocos nem de caracteres. Os relógios só causam interrupções. Dispositivos diferentes dos discos podem ser considerados dispositivos de caracteres. Mas esse modelo de classificação não é perfeito.

Controladores de Dispositivos

As unidades de e/s constituem de um componente eletrônico e um mecânico. O elemento eletrônico é chamado de controlador de dispositivo ou adaptador. Nos computadores pessoais, o controlador de dispositivo aparece em forma de uma placa de circuito impresso.

Nessa placa, tem um conector que pode ser plugado outros dispositivos. (Se for uma interface padrão, entre o dispositivo e o controlador), ou seja, deve ter uma interface baixa entre o controlador e um dispositivo.

Preâmbulo é escrito quando um disco é formatado. Nele, contém o número do cilindro, tamanho do setor, informações dos dados e sincronização.

O trabalho do controlador de dispositivo é converter fluxo de bits em bloco de bytes, além de corrigir erros. O bloco de bytes é formado dentro do controlador. Após converter em blocos de bytes, é somado e checado, se o bloco estiver com a soma correta e sem erros ele é copiado para a memória principal.

Entrada e Saída Mapeada na Memória

Cada controlador de dispositivo tem seus registradores. Esses registradores são usados para comunicar com a CPU. Por meio da escrita nesses registradores do controlador de dispositivo, o S.O pode comandar o dispositivo para aceitar, executar, desligar.

A partir da escrita nesses registradores, o S.O pode saber o estado de um dispositivo, se ele está apto a receber um novo comando, etc. Além dos registradores, os dispositivos têm buffers, no qual o S.O lê e escreve.

Como a cpu se comunica com os registradores do controlador e com os buffers do dispositivo?

Há Duas Possibilidades:

Cada registrador é associado a um número de porta de e/s. Usando uma instrução, a CPU pode ler o registrador do controlador e armazenar o resultado no seu registrador. A mesma pode escrever o conteúdo do registrador da CPU para o registrador de controle.

Visa mapear todos os registradores de controle no espaço de endereçamento. Quando a CPU quer ler uma palavra, ou da memória, ou da e/s, a CPU coloca o endereço que precisa nas linhas do barramento. Um segundo sinal é emitido, ele informa se o espaço requisitado é da memória, ou da e/s.

Se o espaço requisitado é da memória, a memória responderá a requisição, se for da e/s o dispositivo e/s responderá.

Se existe somente um espaço, cada módulo de memória e cada dispositivo de e/s compara as linhas de endereço associado a cada dispositivo de e/s, compara as linhas do endereço com a faixa de endereço associada a cada um. Se os endereços estão dentro da faixa, esse componente responde a requisição.

Os dois esquemas de endereçamento dos controladores apresentam vantagens e desvantagens específicas

As Vantagens da E/S Mapeada Na Memória:

Primeiro: Quando são necessárias instruções especiais de e/s para ler ou escrever nos registradores dos dispositivos, requer código em assembly, pois não tem nenhum modo de executar uma instrução. Assim, com e/s mapeada na memória, um driver do dispositivo pode ser escrito em C.

Em segundo lugar, não é preciso qualquer mecanismo de proteção para impedir que os processos dos usuários executem e/s. Tudo que o S.O faz é deixar de mapear aquela porção do espaço de endereçamento associada aos registradores de controle no espaço de endereçamento virtual do usuário.

Em terceiro lugar, Na memória, cada instrução capaz de referenciar a memória, pode também referenciar os registradores de controle.

Desvantagens:

A maioria dos computadores atuais usa alguma forma de cache para as palavras de memória. O uso de cache para registradores de controle seria desastroso. O HW deve ser equipado com a capacidade de desabilitar a cache.

Em segundo lugar, se existe somente um espaço de endereçamento, todos os módulos de memória e todos os dispositivos devem examinar as referências de memória, para verificar quais delas devem ser respondidas por cada um. Se tiver somente um barramento, cada componente pode olhar para cada endereço diretamente.

Quando se tem um barramento de memória separado em máquinas mapeadas na memória, surge a preocupação que os dispositivos não têm como enxergar os endereços de memória quando estes são lançados no barramento da memória, de modo que estas não tenham como responder.

Uma Solução pode ser enviar todas as referências de memória para a memória, se a memória falhar para responder, então a CPU tenta outros barramentos.

Uma segunda solução poderia colocar um dispositivo de escuta no barramento da memória para repassar aos dispositivos de e/s podem não serem capazes de processar as requisições na velocidade da memória. Já um terceiro método consiste em filtrar os endereços no chip da ponte PCI.

Acesso Direto a Memória (DMA)

Não importa se a CPU tem ou não E/S mapeada na memória, ela precisa endereçar os controladores dos dispositivos para poder trocar dados com eles. A CPU pode requisitar dados de um controlador de E/S, um byte de cada vez, mas desperdiça muito tempo, de modo que um esquema diferente (DMA) seja usado.

Um S.O pode utilizar um DMA somente se o HW tem o controlador de DMA.

O controlador de DMA tem acesso ao barramento do sistema. Eles contém vários registradores que podem ser lidos ou escritos na CPU, os quais possuem registrador de endereço de memória, registrador de controle e registrador de contador de bytes.

O controlador lê um bloco do dispositivo, bit a bit, até que todo bloco esteja no buffer do controlador. Em seguida, ele calcula a soma de verificação, para constatar de que não houve algum erro de leitura. Então, o controlador causa uma interrupção.

Quando o S.O inicia o atendimento, ele pode ler o bloco do disco a partir do buffer do controlador. Um bloco de byte ou uma palavra é lida no registrador do controlador e armazenada na memória principal.

Quando o DMA é usado, o procedimento é diferente. A CPU programa o controlador do DMA, inserindo valores em seus registradores, de modo que ele saiba que tem algo para transferir e para onde transferir.

Ele emite um comando para o controlador de disco, ordenando carregar os dados do disco para seu buffer interno e então verificar a soma de verificação. Quando os dados que estão no buffer do controlador são validos, o DMA pode começar.

O controlador do DMA inicia a transferência emitindo pelo barramento uma requisição de leitura para o controlador de disco. Normalmente, o endereço de memória, para onde escrever, está nas linhas de endereço do barramento, de modo que quando o controlador de disco busca a próxima palavra do seu buffer interno ela sabe onde escrever.

A escrita na memória é outro ciclo de barramento. Quando a escrita está completa, o controlador de disco emite um sinal de confirmação para o controlador, também pelo barramento. O controlador de DMA incrementa o endereço de memória e diminui o contador de bytes.

Se o contador é maior que O, os passos são repetidos até que ele se torne 0. Nesse momento, o controlador de DMA interrompe a CPU para deixa-la ciente de que a transferência está completa. Quando o S.O inicia o atendimento da interrupção, ele não precisa copiar o bloco de disco para a memória, pois ele já está lá.

Muitos barramentos podem operar em dois modos: modo palavra e modo bloco. Alguns controladores de DMA também são capazes de operar em outro modo.

No modo palavra, operação funciona como descrita anteriormente, o controlador de DMA solicita a transferência de uma palavra e consegue.

Se a CPU também quer o barramento, ela tem que esperar. O mecanismo é chamado de roubo de ciclo, pois o controlador de dispositivo rouba da CPU um ciclo do barramento, a cada vez alternando.

No modo bloco, o controlador do DMA diz ao dispositivo para obter o barramento, emite uma serie de transferências e então libera o barramento. Esse modo de operação é chamado de modo surto. Este é mais eficiente que o anterior, pois várias palavras podem ser transferidas com uma aquisição do barramento.

A única desvantagem do modo surto é que ele pode bloquear a CPU e outros dispositivos por um período grande de tempo, caso um longo surto tenha de ser transferido.

Modo Direto, o controlador de DMA diz para o controlador de o dispositivo transferir dados diretamente para a memória principal. Um modo alternativo, que alguns DMAs usam estabelece que o controlador de dispositivo deve enviar uma palavra para o controlador de DMA, que por sua vez requisita o barramento para escrever a palavra para qualquer que seja seu destino.

As maiorias dos controladores de DMA usam endereçamento de memória física para suas transferências. O uso de endereços de memória física requer que S.O converta o endereço virtual do buffer de memória em um endereço físico.

Então o controlador de DMA deve usar a unidade de gerenciamento da memória (MMU) para fazer essa tradução. Somente no caso em que a MMU é parte da memória, e não da CPU, os endereços virtuais são colocados no barramento.

Nem todos os computadores usam DMA, pois se argumenta que a CPU é muito mais veloz que o controlador de DMA e pode fazer o trabalho muito mais rápido.

Interrupções Revistadas

Em hardware, as interrupções trabalham: quando um dispositivo de e/s finaliza seu trabalho, ele gera uma interrupção (se estiverem habilitadas).

Ele envia um sinal pela linha do barramento a qual está associado. O sinal é detectado pelo chip, controlador de interrupção localizado na placa mãe, o qual decide o que fazer.

Se nenhuma outra interrupção está pendente, o controlador de interrupção processa a interrupção imediatamente. Se outra interrupção está em tratamento, ou outro dispositivo fez uma requisição com maior prioridade, o dispositivo é ignorado. Ele continua a gerar interrupção no barramento até ser atendido.

Para tratar a interrupção, o controlador coloca um número nas linhas de endereço, citando qual dispositivo deve observar e passa a interrupção para a CPU.

O sinal de interrupção faz com que a CPU pare aquilo que está fazendo e inicie outras atividades. Os números colocados na linha de endereçamento são usados como índice no vetor de interrupção. Esse vetor aponta para uma rotina de tratamento de interrupção.

Interrupção Precisa

Uma Interrupção que deixa a máquina num estado bem definido. Propriedades: PC é salvo em um lugar conhecido. Todas as instruções anteriores a apontadas pela CPU foram executadas. Nenhuma instrução posterior a apontada pela CPU foi executada. O estado da instrução apontada pelo PC é conhecido uma instrução que não atende a estes requisitos são chamadas de interrupções Imprecisas.

Objetivos do SW de E/S

Independência do dispositivo: Propõe que deveria ser possível escrever programas aptos a acessar qualquer dispositivo. Relacionado a isso, está a nomeação uniforme. O nome de um arquivo ou dispositivo deve ser uma cadeia de caracteres ou números inteiros independentes do dispositivo. Tratamento de erros. Os erros deveriam ser tratados e mais perto possível do HW.

Orientada a Interrupções: Quando a impressora imprime um caractere e está preparado para aceitar o próximo caractere, ela gera uma interrupção. Esta interrupção da impressora é executada. Se não existem mais caracteres para imprimir, o tratador de interrupções executa alguma ação para desbloquear o usuário solicitante.

Ou, ele envia o caractere seguinte, confirma a interrupção e retorna para o processo que parou E/S usando DMA.

DESVANTAGEM: É a ocorrência de interrupções para cada caractere. Desperdiçando tempo de CPU. Uma solução é usar DMA.

Camadas de SW de E/S

O SW de E/s é dividido em 4 camadas. Cada camada tem uma função bem definida para executar e uma interface para as camadas.

Drivers do Dispositivo

Cada controlador tem alguns registradores do dispositivo, utilizado para dar comandos. O número de registradores do dispositivo e a natureza dos comandos variam de dispositivos para dispositivos. EX: um driver de mouse deve aceitar informações do mouse dizendo o quanto se moveu e qual botão foi pressionado.

Em contrapartida, o driver do disco deve saber sobre o setor, trilhas, cilindros e cabeçotes. Obviamente esses drivers serão muito diferentes. Como consequência, cada dispositivo de e/s ligado ao computador precisa de algum código especifico do dispositivo para controlá-lo.

Esse código, chamado de driver do dispositivo, é em geral escrito pelo fabricante do dispositivo, juntamente com o dispositivo.Visto que cada sistema operacional precisa de seus próprios drivers dos dispositivos, os fabricantes fornecem drivers para os sistemas operacionais mais populares.

Cada driver de dispositivo normalmente trata um tipo de dispositivo. Para acessar o HW do dispositivo, o driver deve ser parte do S.O.

Os S.O geralmente classificam os drivers em categoria de dispositivo de blocos, os quais contem vários blocos de dados que podem ser endereçados independentemente - e os dispositivos de caractere, os quais geram ou aceitam um fluxo de caracteres.

Gerenciamento de Entrada/Saída

Responsabilidade

Forma de atuação

Mecanismos de controle de E/S em disco

Problemas com dispositivos de entrada não controlada

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

Modern Operating Systems, Tanenbaum, cap. 5

Operating Systems Concepts, Peterson/Silberschatz, cap. 9

Outros bons livros de sistemas operacionais, nos capítulos sobre gerenciamento de E/S ou gerenciamento de armazenamento em discos ou ainda gerenciamento de periféricos.

Responsabilidade

Na conclusão do último capítulo é indicado que as operações de E/S representam um fator preponderante no desempenho de um sistema operacional. Isso ocorre porque todo e qualquer processo depende de operações de entrada ou saída de dados e elas sempre tomam tempo para serem realizadas. É possível realizar E/S de várias formas, quando a operação é avaliada olhando-se para o responsável pelo seu gerenciamento. Nesse aspecto as formas são:

E/S programada, em que toda a ação é comandada pelo próprio processo que quer fazer E/S;

E/S por interrupção, em que a E/S é comandada por um processo especial, ativado por um sinal de interrupção gerado pelo processo que quer fazer E/S;

E/S por roubo de ciclo, quando a operação de E/S é controlada por um dispositivo especial (o DMA - Direct Memory Access), que rouba ciclos do relógio da CPU para transferir blocos de bytes entre a memória e o dispositivo de E/S;

E/S em cadeia, em que se permite que a fila de espera por operações de E/S seja controlada pelo DMA e não por algum mecanismo que execute na CPU.

As duas primeiras formas são estratégias que consomem muito tempo de CPU e, portanto, valem apenas para aplicações dedicadas. Com elas é possível ao programador desenvolver formas ótimas para se realizar a E/S. As duas últimas são equivalentes e tem amplo uso hoje em dia. Têm também a facilidade de esconder do usuário o que realmente ocorre para se realizar a operação.

Forma de Atuação

O gerenciamento de E/S, independente de quem o realiza, é feito em duas etapas ou níveis: controle de E/S e controle de periféricos. O primeiro, mais alto, se preocupa com aspectos administrativos das solicitações de E/S realizadas, enquanto o segundo se preocupa com a operacionalização física desses pedidos.

Essa divisão faz com que o controle de periféricos trabalhe diretamente com o hardware envolvido, sendo necessário o desenvolvimento de uma interface diferente para cada dispositivo específico (são os drivers que instalamos no sistema).

Ao fazer isso percebe-se que o S.O. não influencia esse controle, sendo o estudo do mesmo deixado para áreas mais ligadas à engenharia.

Já o controle de E/S independe de características elétricas/mecânicas do hardware por atuar apenas no controle lógico dos mesmos. Isso significa, em outras palavras, que o controle de E/S irá gerenciar filas de controle de acesso aos dispositivos de E/S e não executar a atividade de E/S propriamente dita. Isto possibilita, portanto, que um mesmo controle possa ser aplicado para toda uma família de dispositivos.

Esses dois níveis de controle se diferenciam ainda na forma em que tratam dispositivos que exijam acesso privativo (como impressoras, por exemplo) e dispositivos com acesso compartilhado (como discos ou rede, por exemplo).

Em qualquer desses casos o controle de periféricos não faz distinção entre essas categorias de dispositivos por ser apenas o executor da operação de E/S. A diferenciação entre o que pode ter acesso compartilhado ou não é feita no controle de E/S, que é quem trata logicamente todas as solicitações realizadas.

Mecanismos de Controle de E/S em Disco

O controle de E/S em disco é feito de forma a otimizar a movimentação da cabeça de leitura/escrita e do próprio disco. Existem vários algoritmos propostos para a realização dessa otimização, sendo que a maioria deles procura otimizar o chamado tempo de busca (seek time) pela informação no disco. Para entender melhor esses algoritmos é preciso antes entender como um disco é dividido fisicamente, ou seja, entender o que são trilhas e setores de um disco e que os tempos de busca estão associados com o acesso às trilhas e os tempos de latência com os setores.

O tempo de acesso ao disco é, portanto, a soma desses dois tempos. Como o primeiro é bem superior ao segundo, passa a ser a restrição mais importante no momento de escolher um algoritmo para controlar o acesso ao disco.

Os algoritmos de acesso a disco mais usados são o FIFO (que escolhe o próximo pedido a ser atendido pela ordem de requisição), SSTF (shortest seek-time first, que atende o pedido para a trilha mais próxima da atual), o SCAN (que varre o disco atendendo os pedidos linearmente) e suas variações, como C-SCAN, LOOK, C-LOOK, Modified Scan, etc.

Problemas com Dispositivos de Entrada não Controlada

Fazer o gerenciamento de dispositivos de E/S não envolve atividades muito complexas quando quem dispara a operação for o sistema operacional.

Entretanto isso não é verdade para operações de entrada de dados com determinados dispositivos, como mouse, teclado e redes. Nesses casos o sistema deve prover algum mecanismo para que a recepção de dados ocorra sem perdas, pois não se sabe quando eles chegarão, qual processo os deve receber, qual o tamanho do bloco de dados e como sinais de controle devem ser tratados.

O tratamento de quem deve receber os dados e de quantos bytes virão é feito com a identificação através de um cabeçalho (header) no bloco de dados (quando estivermos usando rede) ou pela identificação do dispositivo (quando não for rede). Já o tratamento de quando os dados chegarão pode ser feito com a criação de buffers e sinais de interrupção, com técnicas como as de toggle buffers ou buffers circulares.

Por fim, o tratamento de sinais de controle pode ser feito de forma imediata (com detecção dos mesmos) ou durante o esvaziamento do buffer (com a sua leitura pelo processo destinatário).

Em qualquer das hipóteses existe perda de informação, ficando a cargo do projetista do S.O. identificar qual tipo de informação perdida é menos prejudicial aos sistemas em que ele será aplicado.

Balenceamento de Carga

O balanceamento de carga difunde tarefas entre todos os processadores disponíveis. Isso é importante em qualquer sistema e é fundamental para a capacidade de processamento e a escalabilidade. No IBM® Cognos BI, o balanceamento de carga significa assegurar que as solicitações de processamento sejam distribuídas adequadamente entre todos os servidores IBM Cognos BI disponíveis. O IBM Cognos BI faz isso automaticamente, mas é possível configurar também o balanceamento de carga.

Balanceamento de Carga Automático

Em um ambiente distribuído, o IBM Cognos BI equilibra o carregamento de solicitações automaticamente. Por padrão, conforme servidores são incluídos no sistema, cada dispatcher de servidor processa o mesmo número de solicitações.

Se houver mais de uma instância de um determinado serviço, o dispatcher distribuirá solicitações para todas as instâncias ativadas do serviço registradas no Content Manager.

Configurando Balanceamento de Carga

Embora o balanceamento de carga automático possa ser apropriado quando recursos de hardware são idênticos em toda a topologia do servidor, talvez ele não seja ideal em ambientes contendo uma combinação de recursos de hardware com diferentes características de capacidade.

Em um ambiente de hardware contendo servidores com graus variados de capacidade de processamento, é desejável equilibrar a carga de processamento de acordo com a capacidade do servidor.

No IBM Cognos BI, é possível definir configurações de capacidade de processo usando opções de administração do servidor.

Por exemplo, se tiver dois servidores, sendo que um deles tem duas vezes a capacidade do outro, você pode designar ao servidor mais poderoso um peso igual a dois e ao servidor menos poderoso um peso igual a um. O IBM Cognos BI então envia duas vezes mais solicitações para o servidor mais poderoso.

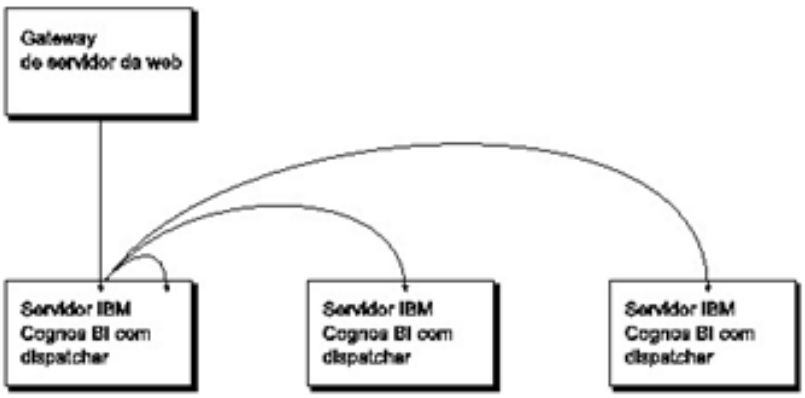
Para obter mais informações sobre as configurações de dispatcher do IBM Cognos BI, consulte o IBM Cognos Business Intelligence Administration and Security Guide.

Dispatchers de Balanceamento de Carga

Sem um mecanismo de balanceamento de carga de software ou hardware, cada gateway do IBM Cognos BI tem conhecimento de apenas um dispatcher e distribui todas as solicitações para ele. O dispatcher então distribui as solicitações entre os servidores IBM Cognos BI.

Como cada solicitação passa inicialmente pelo menos dispatcher em um servidor, a carga nesse servidor aumenta. Uma etapa extra é necessária para equilibrar automaticamente a carga, conforme mostrado no diagrama a seguir.

Figura 1. Dispatchers de Balanceamento de Carga

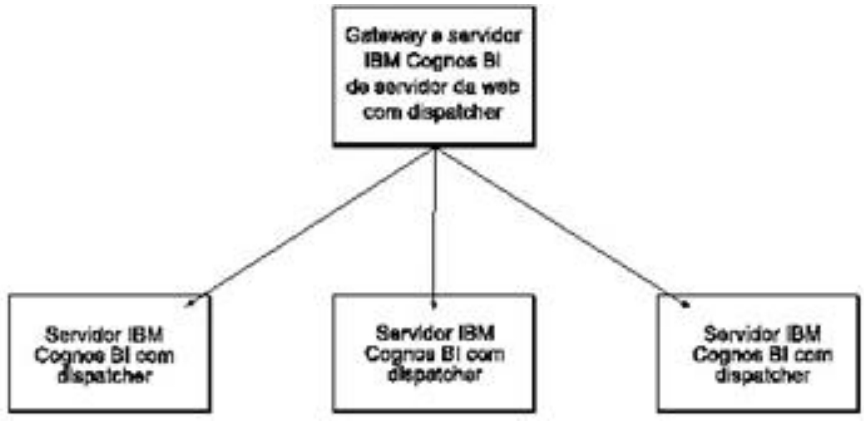


Essa etapa extra pode ser evitada implementando balanceamento de carga sem um mecanismo de balanceamento de carga externo ou usando um roteador ou outro mecanismo de balanceamento de carga.

Balanceamento de Carga sem um Mecanismo Externo

Como servidores gateway geralmente têm menos carga que os servidores IBM Cognos BI, você pode conseguir um desempenho melhor configurando dispatchers junto com os gateways, conforme mostrado no diagrama a seguir.

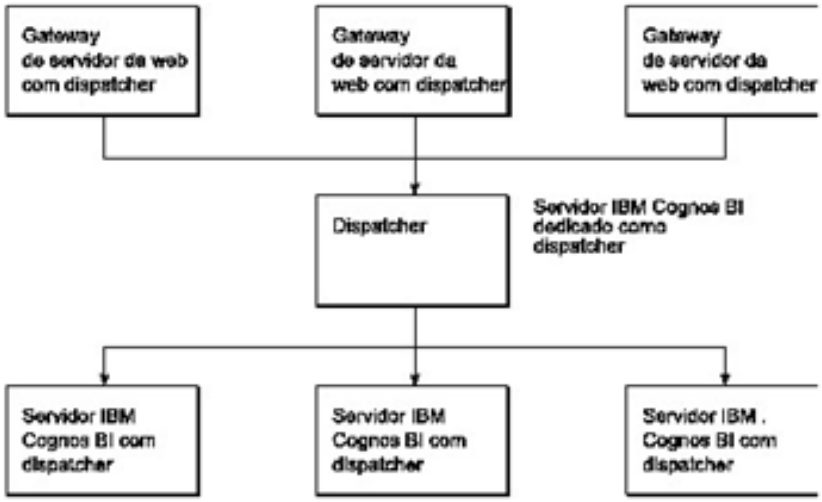
Figura 2. Balanceamento de Carga Configurando Dispatchers com Gateways



Isso assegura que a capacidade de processamento dos servidores IBM Cognos BI seja direcionada para o atendimento de solicitações de relatório, e não de solicitações de balanceamento de carga.

Também é possível conseguir um balanceamento de carga fazendo com que os gateways direcionem todo o tráfego para um computador servidor IBM Cognos BI dedicado a despachos, conforme mostrado no diagrama a seguir.

Figura 3. Balanceamento de Carga Configurando Gateways para Direcionarem o Tráfego para um Servidor IBM Cognos BI Dedicado Usado para Despacho



Essa configuração também remove carregamentos de despachos dos servidores IBM Cognos BI. No entanto, ela não requer computadores de despacho separados.

Usando Mecanismos de Balanceamento de Carga Externos

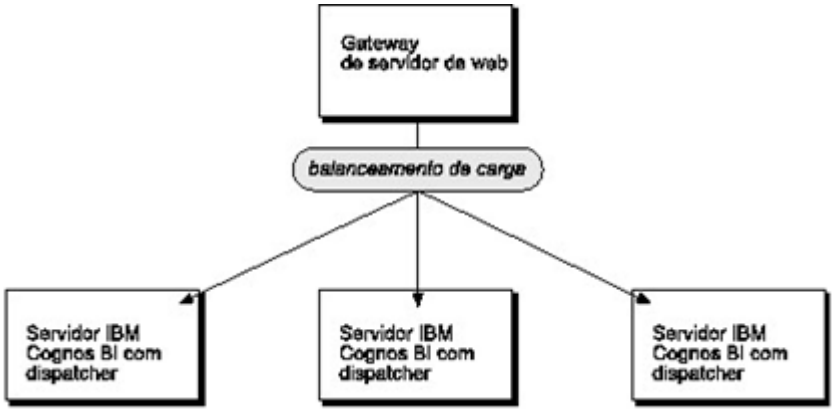
É possível usar mecanismos de balanceamento de carga externos, como roteadores, para distribuir ainda mais as tarefas no IBM Cognos BI. Os roteadores de balanceamento de carga podem ser usados em qualquer um dos ou nos dois locais:

Entre o navegador e a Camada 1: Servidor da Web

Entre a Camada 1: Servidor da Web e a Camada 2: Servidor IBM Cognos BI

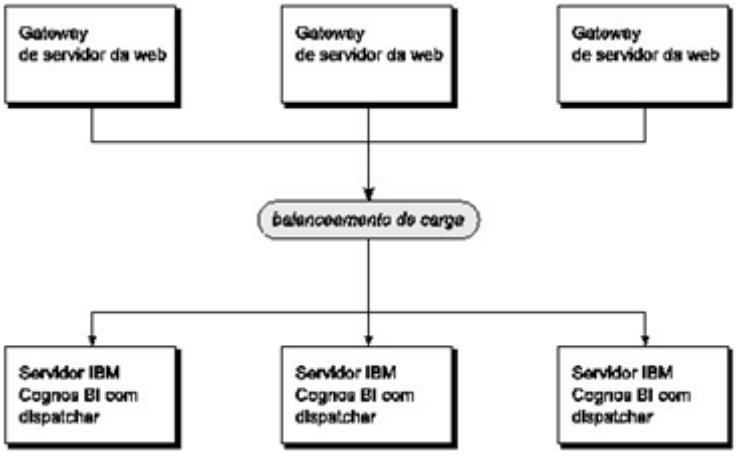
É possível usar um mecanismo de balanceamento de carga externo para distribuir solicitações para dispatchers em todos os servidores disponíveis, conforme mostrado no diagrama a seguir.

Figura 4. Usando Mecanismos de Balanceamento de Carga Externos para Distribuir Solicitações para Dispatchers entre todos os Servidores Disponíveis



Também é possível usar roteadores com diversos gateways, conforme mostrado no diagrama a seguir.

Figura 5. Usando Mecanismos de Balanceamento de Carga Externos para Usar Roteadores com diversos Gateways



Um mecanismo de balanceamento de carga ideal fornece o mesmo reconhecimento de capacidade que um dispatcher do IBM Cognos BI.

Para assegurar que as solicitações não sejam distribuídas por um mecanismo de balanceamento de carga externo e o dispatcher, você deve configurar os dispatchers para não usarem balanceamento de carga integrado para solicitações de baixa afinidade. Isso garante que as solicitações permaneçam no servidor para onde o balanceador de carga de hardware as direcionou.

Linguagens de Programação

Podemos imaginar o computador como uma super calculadora, capaz de fazer cálculos muito mais rápido que nós, mas para isso devemos dizer para o computador o que deve ser calculado e como deve ser calculado. A função das linguagens de programação é exatamente essa, ou seja, servir de um meio de comunicação entre computadores e humanos.

Existem dois tipos de linguagens de programação: as de baixo nível e as de alto nível. Os computadores interpretam tudo como números em base binária, ou seja, só entendem zero e um.

As linguagens de baixo nível são interpretadas diretamente pelo computador, tendo um resultado rápido, porém é muito difícil e incômodo se trabalhar com elas. Exemplos de linguagens de baixo nível são a linguagem binária e a linguagem Assembly.

Exemplo De Código Em Assembly:

```
MOV r0, #0C; load base address of string into r0
LOAD: MOV r1, (r0); load contents into r1
CALL PRINT; call a print routine to print the character in r1
```

INC r0; point to next character
JMP LOAD; load next character

Como pode-se notar, é uma linguagem bastante complicada.

Já as linguagens de alto nível são mais fáceis de se trabalhar e de entender, as ações são representadas por palavras de ordem (exemplo faça, imprima, etc) geralmente em inglês, foram feitos assim para facilitar a memorização e a lógica. Elas não são interpretadas diretamente pelo computador, sendo necessário traduzi-las para linguagem binária utilizando-se de um programa chamado compilador.

Quando programamos em uma linguagem de programação de alto nível primeiramente criamos um arquivo de texto comum contendo a lógica do programa, ou seja, é onde falamos ao computador como deve ser feito o que queremos. Este arquivo de texto é chamado de código-fonte, cada palavra de ordem dentro do código-fonte é chamada de instrução.

Após criarmos o código-fonte devemos traduzir este arquivo para linguagem binária usando o compilador correspondente com a linguagem na qual estamos programando. O compilador irá gerar um segundo arquivo que chamamos de executável ou programa, este arquivo gerado é interpretado diretamente pelo computador.

Existem algumas linguagens de programação que não necessitam de compiladores, como o PHP, uma linguagem dedicada à produção de websites dinâmicos, como o InfoEscola. As instruções em PHP são compiladas e executadas ao mesmo tempo.

Cada linguagem de programação é diferente da outra, contendo palavras-chave próprias. Exemplos de linguagens de alto nível são C++, Java, C#, Delphi (Pascal), PHP, Visual Basic, etc.

Computadores são estúpidos. Isso pode vir como uma surpresa, considerando que eles são capazes de fazer cálculos complexos em poucos segundos, de forma que nenhum humano conseguiria. Mas é verdade! Computadores são extremamente estúpidos e precisam de instruções exatas para tudo o que fazem. O que é óbvio para você, humano, certamente não é óbvio para uma máquina. E se você quer que a máquina faça algo pra você, bem, você precisa, de certa forma, "falar com ela".

Você já deve ter ouvido falar que computadores entendem apenas binário, ou seja, a linguagem de 0 e 1. Todas as instruções que são dadas para eles são traduzidas em sinais elétricos que significam, basicamente, ligado e desligado.

Pense em uma pequena lâmpada. Quando ela está ligada, indica 0. Quando desligada, indica 1. Isso é binário. E há uma infinita quantidade de combinações que você pode fazer com isso.

É claro, nenhum - talvez algum - humano na face da terra iria programar um Skyrim ou mesmo um simples site da Web utilizando apenas 0 e 1. Isso seria extremamente cansativo e demorado. Pra isso que servem as linguagens de programação.

Pense em você e sua máquina. Você fala português, mas a sua máquina fala apenas binário. Entre você e a máquina está um intérprete, o compilador, mas ele só sabe binário e uma outra lingua - Java, por exemplo.

Aprender a "falar" Java é muito mais fácil que aprender a "falar" binário. Isso porque Java possui muitas palavras em comum com o inglês, e você pode escrever instruções que você entende, como "imprima" ou "leia", ao invés de 0010100010101010 10.

Assim, quando você quer se comunicar com a sua máquina e criar uma série de instruções para ela seguir, você escreve em Java para o compilador e ele traduz as instruções para a máquina, em binário. Isso é uma linguagem de programação.

Linguagens de programação podem ser catalogadas em dois tipos: de baixo nível e alto nível. Isso não quer dizer que uma é mais rica que a outra ou algo do tipo. Significa que uma está muito mais próxima da máquina do que a outra.

Chamamos de linguagem de programação (ou informática) uma linguagem destinada a descrever o conjunto das ações consecutivas que um computador deve executar.

Desta forma, uma linguagem informática é uma maneira prática para nós, humanos, darmos instruções a um computador. Por outro lado, o termo 'linguagem natural' representa as possibilidades de expressão compartilhadas por um grupo de indivíduos (por exemplo, o inglês ou o português).

As linguagens que servem para que os computadores se comuniquem entre si não têm nada a ver com linguagens informáticas. Neste caso, falamos de protocolos de comunicação, que são duas noções totalmente diferentes. Em uma linguagem de programação rigorosa, cada instrução gera uma ação do processador.

A linguagem utilizada pelo processador chama-se linguagem das máquinas. Trata-se de dados tal como chegam ao processador, constituídos por uma sequência de 0 e 1 (dados binários). Assim, ela não é compreensível por um ser humano e é por isso que foram criadas linguagens intermediárias, compreensíveis por homens. O código escrito neste tipo de linguagem é transformado em linguagem da máquina para poder ser usada pelo processador.

A linguagem de montagem foi a primeira linguagem de programação a ser usada. Ela é parecida com a linguagem das máquinas, mas é mais compreensível pelos programadores. Ela é tão parecida com a linguagem das máquinas que depende do tipo de processador utilizado, já que cada um pode ter a sua própria linguagem das máquinas.

Assim, um programa desenvolvido para uma máquina não poderá ser aplicado a outro tipo de máquina. O termo mobilidade designa a aptidão que um programa informático tem em ser utilizado em máquinas de tipos diferentes. Às vezes, para poder utilizar um programa informático escrito em linguagem de montagem em outro tipo de máquina, o programa deverá ser reescrito completamente.

Uma linguagem informática tem, então, várias vantagens, ou seja, ela é mais fácil de entender se comparada à linguagem das máquinas e, sobretudo, ela permite uma maior mobilidade, isto é, uma maior facilidade de adaptação nas máquinas de tipos diferentes.

Quais São os Tipos de Linguagem de Programação

As linguagens de programação estão divididas em duas grandes famílias, dependendo da maneira como as instruções são processadas: as linguagens imperativas e as linguagens funcionais.

Como Funciona a Linguagem Imperativa

A linguagem imperativa organiza o programa através de uma série de instruções, reunidas por blocos e compreendendo saltos condicionais que permitem retornar a um bloco de instruções se a condição for realizada.

Historicamente, são as primeiras linguagens, mesmo se várias linguagens modernas ainda utilizem este princípio de funcionamento. Porém, as linguagens imperativas estruturadas sofrem da falta de flexibilidade, dado o caráter sequencial das instruções.

Como É Construída A Linguagem Funcional

A linguagem funcional tem um nível elevado de abstração, já que elimina muitos detalhes da programação, diminuindo assim a eventualidade de erros. Por quê? Como não depende das operações de atribuição, este tipo de linguagem evita estados ou dados mutáveis.

Ela se baseia na aplicação de funções e, desta forma, qualquer função pode ter ou não regras e um simples valor de retorno. As regras são os valores de entrada da função e o valor de retorno é o resultado desta função. A determinação de uma função vai dizer como ela será avaliada em relação às outras, não exigindo definições adicionais. A linguagem funcional é, pelas suas características, mais simples para tratar das provas e análises matemáticas do que a linguagem imperativa.

Como e Feita a Interpretação e a Compilação

As linguagens de programação podem ser classificadas em duas categorias: as linguagens interpretadas e as linguagens compiladas.

O Processo da Linguagem Interpretada

Uma linguagem de programação é, por definição, diferente da linguagem das máquinas. Por isso, é necessário traduzi-la para torná-la compreensível para o processador. Um programa escrito numa linguagem interpretada precisa de um programa auxiliar (o intérprete) para traduzir progressivamente as instruções recebidas.

A Construção da Linguagem Compilada

Um programa escrito numa linguagem compilada vai ser traduzido completamente por um programa anexo, chamado compilador, a fim de gerar um novo arquivo autônomo, que não precisará mais de outro programa para ser executado; dizemos, então, que este arquivo é executável.

Um programa escrito em linguagem compilada não precisa, uma vez compilado, de um programa anexo para ser executado. Além disso, como a tradução é feita completamente, a execução é mais rápida. Contudo, ele é menos flexível do que um programa escrito com uma linguagem interpretada, porque a cada modificação do arquivo-fonte (aquele que vai ser compilado) será preciso recopiá-lo para que as alterações tenham efeito.

Por outro lado, um programa compilado tem a vantagem de garantir a segurança do código-fonte. Na verdade, uma linguagem interpretada e compreensível, permite a qualquer programador conhecer os segredos de construção de um programa e, desta forma, copiar ou alterar o código.

Por causa disso, existe o risco de desrespeito aos direitos autorais. Certos aplicativos protegidos precisam garantir a confidencialidade do código para evitar o hacking.

A Estrutura das Linguagens Intermediárias

Algumas linguagens podem pertencer, de certa forma, às duas categorias (LISP, Java, Python, etc.), já que o programa escrito com estas linguagens pode, em certas condições, sofrer uma fase de compilação intermediária para ser compatível com um arquivo escrito numa linguagem que não é inteligível (logo, diferente do arquivo-fonte) e, por isso, não executável, necessitando de um intérprete.

Por exemplo, os applets Java, pequenos softwares executados na janela de um aplicativo para ampliar as funcionalidades dos navegadores (som, animação, etc.), encontrados nas páginas Web. Eles são arquivos compilados que só podem ser executados a partir de um navegador (arquivos com extensão Class).

As Linguagens na Prática

Veja, abaixo, a lista das linguagens de programação mais conhecidas:

Linguagem	Domínio de aplicação principal	Compilada/interpretada
ADA	O tempo real	Linguagem compilada
BASIC	Programação básica com objetivos educativos	Linguagem interpretada
C	Programação do sistema	Linguagem compilada
C++	Programação do sistema objeto	Linguagem compilada
Cobol	Gestão	Linguagem compilada
Fortran	Cálculo	Linguagem compilada
Java	Programação orientada para a Internet	Linguagem intermediária
MATLAB	Cálculo matemático	Linguagem interpretada
Matemática	Cálculo matemático	Linguagem interpretada
LISP	Inteligência artificial	Linguagem intermediária
Pascal	Ensino	Linguagem compilada
PHP	Desenvolvimento de sites web dinâmicos	Linguagem interpretada
Prolog	Inteligência artificial	Linguagem interpretada
Perl	Processamento de cadeias de caracteres	Linguagem interpretada

Princípios de Sistemas Operacionais

Um sistema operacional é uma camada de software que atua entre o hardware e os diversos aplicativos existentes em um sistema computacional, executando diversas atividades, de modo a garantir a disponibilidade de recursos entre todos os programas em execução.

Tipos de Sistemas Operacionais

Em relação a arquitetura do sistema operacional:

Sistema Monolítico: o kernel consiste em um único processo executando numa memória protegida (espaço do kernel). Ex: Windows, Linux, FreeBsD;

Sistema em Camadas: funções do kernel irão executar em camadas distintas de acordo com o nível de privilégio. Ex. Multics;

Modelo cliente/servidor ou microkernel: o kernel consiste apenas no essencial (comunicação e gerenciamento de processos) e funções como sistemas de arquivos e gereciamento de memória são executadas no espaço do usuário como serviço: as aplicações (programas) são os clientes. Ex: GNU Hurd, Mach;

Monitor de máquinas virtuais: fornece uma abstração do hardware para vários sistemas operacionais. Ex: VM/370, VMware, Xen.

Quanto a capacidade de processamento, pode usar as seguintes classificações:

Monotarefa: pode-se executar apenas um processo por vez.

Ex: DOS;

Multitarefa: além do próprio sistema operacional, vários processos de utilizador (tarefas) estão carregados em memória, sendo que um pode estar ocupando o processador e outros ficam enfileirados, aguardando a sua vez. O compartilhamento de tempo no processador é distribuído de modo que o usuário tenha a impressão de que vários processos estão sendo executados simultaneamente.

Ex: Windows, Linux, FreeBSD

Multiprocessamento ou multiprogramação: o SO pode distribuir as tarefas entre vários processadores. Alguns autores utilizam o termo multiprocessamento como se fosse multitarefa.

Sistemas operacionais para computadores de pequenos e grandes dividem-se em quatro categorias, que se diferenciam pelo tempo de resposta e pela forma de entrada de dados no sistema: em lotes (batch), interativa, em tempo real e sistemas híbridos;

Os sistemas em lotes (batch): existem desde o tempo dos primeiros computadores, os quais trabalhavam com cartões perfurados ou fita magnética para a entrada de dados.

Os sistemas em lotes de hoje não trabalham mais com cartões perfurados ou com fitas magnéticas, mas as tarefas ainda são processadas em série, sem interação com o usuário;

Os sistemas interativos: também conhecidos como sistemas de tempo compartilhado, produzem um tempo de resposta mais rápido do que os sistemas em lotes, mas são mais lentos do que os sistemas em tempo real.

Foram criados para atender a necessidade do usuário, que necessitava de tempo de resposta mais rápido na depuração dos programas. O sistema operacional exigiu a criação de programas de tempo compartilhado, permitindo o usuário interagir diretamente com o sistema de computação;

Os sistemas em tempo real: os mais rápidos de todos, são utilizados em ambientes onde o tempo é um fator crítico: vôos espaciais, controle de tráfego aéreo etc;

Os sistemas híbridos: são uma combinação entre os sistemas em lotes e os sistemas interativos. Executa os programas em lotes em segundo plano.

Um sistema híbrido se beneficia do tempo livre entre demandas de processamento para executar programas que não precisam da interferência do operador.

Base do Sistema Operacional

Base que forma o sistema operacional:

Gerenciador de memória: responsável pela alocação de memória;

Gerenciador de unidade de processamento: responsável pela alocação de recursos da CPU;

Gerenciador de dispositivos: Sua tarefa é escolher a forma mais adequada para alocação de todos os dispositivos;

Gerenciador de arquivos: monitora todos os arquivos no sistema. Faz cumprir as restrições de acesso a cada arquivo.

Módulos do sistema operacional residentes em disco, que são carregados para a memória somente quando são necessários, chamam-se módulos transientes.

Gerenciador de Memória

O gerenciador de memória trabalha com tabelas geradas para controlar os programas em execução. Existem 3 ou 4 tabelas, dependendo do tipo de gerenciamento adotado;

A memória é dividida logicamente em tamanhos iguais (páginas de memória) ou diferentes (segmentos de memória), para acomodar os diversos programas residentes em memória.

Esquemas de gerenciamento de memória:

- Alocação de memória com paginação;

- Alocação de memória com paginação sob demanda;

- Alocação de memória com segmentação;

- Alocação de memória com segmentação / paginação sob demanda.

Alocação de Memória Com Paginação

Divisão da memória em páginas de igual tamanho;

A paginação funciona muito bem quando as páginas, os setores e os quadros de páginas possuem o mesmo tamanho; não faz uso da memória virtual;

Gerenciador de memória prepara o programa da seguinte forma: determina o número de páginas do programa; localiza quadro de páginas suficientes na memória principal; carrega todas as páginas do programa nesses quadros de memória;

As páginas de um programa, quando forem carregadas para a memória, não precisam ser carregadas em blocos contíguos.

Dois programas não podem usar a mesma página de memória ao mesmo tempo, porque causaria a fragmentação interna da memória; esse método requer que o programa inteiro seja alojado na memória na hora da execução.

Alocação de Memória com Paginação Sob Demanda

O gerenciador de memória trabalha com tabelas para monitorar o uso dos quadros de memória. Essencialmente três tabelas executam essa função: a Tabela de Programas (TP), a Tabela de Mapa de Páginas (TMP) e a Tabela de Mapa de Memória (TMM). Todas residem na parte da memória principal;

Cada programa possui sua própria TMP, contendo informações como número da página, endereço de memória do quadro da página correspondente etc;

A TMM possui uma entrada para cada quadro de página, no qual está a localização e o status (livre/ocupada); A partir desse gerenciamento se começou a trabalhar com memória virtual; A memória continua sendo dividida em tamanhos iguais (páginas).

Alocação de Memória Sob Demanda

Conjunto de trabalho: é o conjunto de páginas residentes em memória que podem ser acessadas diretamente sem implicar erro de página. Essa técnica é utilizada para evitar o thrashing (troca excessiva de páginas entre a memória RAM e a virtual).

Conceitos e critérios para substituição de páginas:

- A política que rege a escolha da página que deve sair da memória, ou seja, o critério de substituição das páginas, é crucial para a eficiência do sistema;

- Duas técnicas mais conhecidas são: “primeiro a entrar, primeiro a sair (FIFO, ou first-in first-out)” e “página usada menos recentemente (UMR)”;

- Nas duas técnicas tem que se observar se não houve alteração dos dados antes de retirar da memória principal;

- O algoritmo FIFO consulta apenas os bits de status e de página modificada antes de fazer as substituições, mas o algoritmo UMR consulta os três bits

Alocação de Memória com Segmentação

- O conceito de segmentação baseia-se em uma prática comum entre os programadores: a estruturação dos seus programas em módulos (agrupamento lógico de códigos).

Segundo esse esquema cada programa é dividido em vários segmentos de tamanhos diferentes;

- Este esquema difere fundamentalmente da paginação, onde um programa é dividido em páginas de igual tamanho que muitas vezes contém instruções de mais de um módulo de programa;

- Uma outra diferença importante é que a memória não é mais dividida em quadros de página, pois o tamanho de cada segmento varia, ou seja, a memória é alocada de maneira dinâmica;

- Esse esquema reduz a segmentação da memória (aproveita melhor a memória).

Monitoramento da memória:

Quando um programa é compilado ou montado, os segmentos são definidos de acordo com os módulos estruturais do programa.

Cada segmento é numerado e uma Tabela de Mapa de Segmentos (TMS) é gerada para cada programa; A TMS inclui os números de segmentos, o tamanho, os direitos de acesso, o status e, quando um segmento é carregado em memória, o endereço de memória.

- O gerenciador de memória monitora os segmentos em memória através de 3 tabelas:

Tabela de Programas (TP): exibe todos os programas em processameto (uma para todo o sistema);

Tabela de Mapa de Segmentos (TMS): exibe detalhes sobre cada segmento (uma para cada programa);

Tabela de Mapa de Memória (TMM): monitora a alocação de memória principal (uma para todo o sistema).

Alocação de Memória com Segmentação / Paginação Sob Demanda

É uma combinação entre a segmentação e a paginação sob demanda que alia os benefícios lógicos da segmentação aos benefícios físicos da paginação; Os segmentos são divididos em páginas de igual tamanho, menores do que a maioria dos segmentos e mais facilmente manipuladas.

Assim sendo, muitos dos problemas da segmentação (compactação, fragmentação externa e necessidade de armazenagem secundária) são eliminados, pois as páginas são de tamanho fixo; A maior desvantagem desse esquema são a sobrecarga gerada pelas tabelas adicionais.

Monitoramento da memória:

- Tabela de Programa (TP): exibe todos os programas em processamento (uma para todo o sistema);

- Tabela de Mapa de Segmentos (TMS): exibe detalhes sobre cada segmento (uma para cada programa);

- Tabela de Mapa de Páginas (TMP): exibe detalhes sobre cada página (uma para cada segmento);

- Tabela de Mapa de Memória (TMM): monitora a alocação de quadros de página na memória principal (uma para todo o sistema).

Sistema de Arquivos NTFS

O NTFS (New Technology File System) é um sistema de arquivos que surgiu com o lançamento do Windows NT. Sua confiabilidade e desempenho fizeram com que fosse adotado nos sistemas operacionais posteriores da Microsoft, como Windows XP, Windows Vista, Windows 7 e Windows Server 2008. Antes, o que é um sistema de arquivos?

Não é possível gravar dados em um HD ou em qualquer outro dispositivo de armazenamento de forma a manter as informações acessíveis e organizadas sem um sistema de arquivos (file system) - essencialmente, um tipo de estrutura que indica como os arquivos devem ser gravados e lidos pelo sistema operacional do computador.

É o sistema de arquivos que determina como as informações podem ser guardadas, acessadas, copiadas, alteradas, nomeadas e até apagadas.

Ou seja, resumindo, toda e qualquer manipulação de dados em um dispositivo de armazenamento necessita de um sistema de arquivos para que estas ações sejam possíveis.

Sem um sistema de arquivos, os dados armazenados seriam apenas um conjunto de bits sem utilidade.

Há vários sistemas de arquivos disponíveis, para os mais diversos sistemas operacionais e para as mais variadas finalidades.

O NTFS é um sistema de arquivos amplamente utilizado nos sistemas operacionais da Microsoft. Sua primeira aparição foi no Windows NT, sistema operacional para uso em servidores cuja primeira versão foi lançada em 1993. No entanto, a história do NTFS começa muito antes disso.

Até aquela época, a Microsoft não possuía nenhum sistema operacional capaz de fazer frente ao Unix e suas variações em aplicações de servidores. Seus principais produtos eram o MS-DOS e a linha Windows 3.x, essencialmente, sistemas operacionais para uso doméstico ou em escritório. Era preciso criar algo novo, capaz de disputar mercado com as soluções baseadas em Unix. Foi aí que surgiu o Windows NT.

De nada adianta um sistema operacional novo se o seu sistema de arquivos for limitado. Na época, a Microsoft tinha em mãos o sistema de arquivos FAT. Este funcionava razoavelmente bem em aplicações domésticas, mas não serviria aos propósitos do novo projeto por uma série de restrições, entre elas, baixa tolerância a falhas, inviabilidade de uso de permissões de arquivos e limitações para o trabalho com grande volume de dados.

Para superar esses e outros problemas, a Microsoft decidiu utilizar o NTFS. Porém, ao contrário do que muita gente pensa, a empresa não desenvolveu esse sistema de arquivos sozinha. Ela utilizou como base o HPFS (High Performance File System), sistema de arquivos que tinha a IBM por trás.

No início da década de 1980, ambas as companhias fecharam um acordo para o desenvolvimento do OS/2, um sistema operacional até então moderno, que se destacaria por sua capacidade gráfica (naquela época, era muito comum o uso de sistemas operacionais baseados em linha de comando).

O problema é que, logo, Microsoft e IBM passaram a divergir em relação a diversos pontos. Como consequência, desfizeram a parceria. A IBM continuou tocando o projeto do OS/2, enquanto a Microsoft foi cuidar de seus interesses, mais precisamente, do projeto que resultou no Windows NT.

No entanto, a companhia não abandonou a parceria de mãos vazias: levou vários conceitos do HPFS - o sistema de arquivos do OS/2 - relacionados à segurança, confiabilidade e desempenho para posteriormente implementá-los no NTFS.

Sabe-se também que o NTFS tem alguma relação com o Files-11, sistema de arquivos do sistema operacional VMS, que passou às mãos da Compaq em 1998, empresa que posteriormente foi adquirida pela HP.

Quando os trabalhos no VMS estavam em andamento, parte de sua equipe se transferiu para Microsoft, com destaque para o engenheiro de software Dave Cutler, um dos nomes por trás do NTFS e do próprio Windows NT.

Principais Características do NTFS

Os conceitos aplicados ao NTFS fizeram com que o Windows NT e versões posteriores do sistema fossem bem recebidos pelo mercado. Uma dessas características diz respeito ao quesito "recuperação": em caso de falhas, como o desligamento repentino do computador, o NTFS é capaz de reverter os dados à condição anterior ao incidente.

Isso é possível, em parte, porque, durante o processo de boot, o sistema operacional consulta um arquivo de log que registra todas as operações efetuadas e entra em ação ao identificar nele os pontos problemáticos. Ainda neste aspecto, o NTFS também suporta redundância de dados, isto é, replicação, como o que é feito por sistemas RAID, por exemplo.

Outra característica marcante do NTFS é o seu esquema de permissões de acesso. O Unix sempre foi considerado um sistema operacional seguro por trabalhar com o princípio de que todos os arquivos precisam ter variados níveis de permissões de uso para os usuários. O NTFS também é capaz de permitir que o usuário defina quem pode e como acessar pastas ou arquivos.

O NTFS também é bastante eficiente no trabalho com arquivos grandes e unidades de discos volumosos, especialmente quando comparado ao sistema de arquivos FAT. Você vai entender o porquê no tópico a seguir.

Lidando com Arquivos

Em um disco rígido, a área de armazenamento é dividida em trilhas. Cada trilha é subdividida em setores (saiba mais neste artigo sobre HDs), cada um com 512 bytes, geralmente. FAT e NTFS trabalham com conjuntos de setores, onde cada um é conhecido com cluster (ou unidade de alocação). O FAT16, por exemplo, pode ter, comumente, clusters de 2 KB, 4 KB, 8 KB, 16 KB e 32 KB.

Aqui há um possível problema: cada arquivo gravado utiliza tantos clusters quanto forem necessários para cobrir o seu tamanho. Se, por exemplo, tivermos um arquivo com 50 KB, é possível guardá-lo em dois clusters de 32 KB cada.

Você deve ter percebido então que, neste caso, um cluster ficou com espaço sobrando. Esta área pode ser destinada a outro arquivo, correto? Errado! Acontece que cada cluster só pode ser utilizado por um único arquivo. Se sobrar espaço, este permanecerá vazio. Esse é um dos problemas do sistema FAT.

Há ainda outra limitação: o FAT16 trabalha com discos ou partições com até 2 GB. Essa situação só melhora com o FAT32, que pode trabalhar com até 2 TB (terabytes).

O NTFS, por sua vez, não pode contar com esse tipo de limitação. Por isso, utiliza 64 bits para endereços de dados, contra 16 do FAT16 e 32 do FAT32.

Essa característica, aliada ao tamanho dos clusters, determina o volume máximo de dados com que cada partição NTFS pode trabalhar. Com o uso de clusters de 64 KB, esse limite pode chegar a 256 TB.

Por padrão, o tamanho dos clusters é definido automaticamente com base na capacidade de armazenamento do dispositivo durante o processo de instalação do sistema operacional ou de formatação de uma partição - indo de 512 bytes a 64 KB -, podendo também ser definido pelo usuário com procedimentos específicos.

Tolerância a Falhas

Para a preservação dos dados, o NTFS utiliza um esquema de journaling, isto é, o arquivo de log mencionado anteriormente.

De maneira resumida, seu funcionamento ocorre da seguinte forma: o log registra toda as ações que acontecem no sistema operacional em relação aos arquivos.

Quando um documento é criado, um espaço em disco é alocado para ele, suas permissões são definidas e assim por diante.

A questão é que se, por exemplo, o computador ficar repentinamente sem energia, o espaço definido para o arquivo pode ser alocado, mas não utilizado.

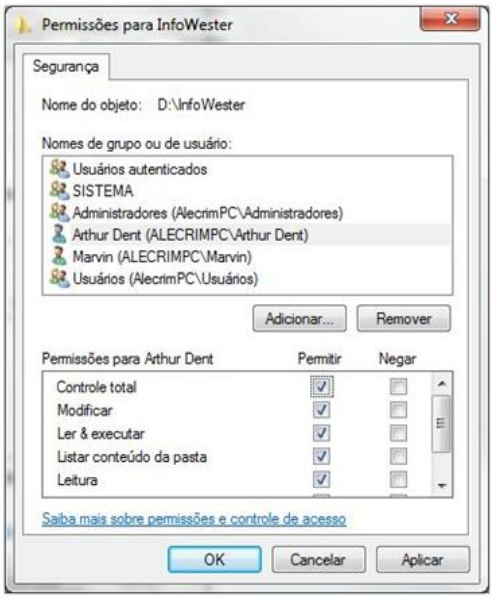
Quando o sistema operacional é reativado, consulta o arquivo de log para saber quais procedimentos não foram executados por completo e executa a ação correspondente para corrigir o problema.

Para manter a integridade do sistema, basicamente, três passos são executados: verificação do log para checar quais clusters devem ser corrigidos; nova execução das transações marcadas como completas no final do log; reversão de procedimentos que não puderam ser concluídos.

Perceba que, com isso, o NTFS pode não conseguir recuperar os últimos dados gravados antes da interrupção, mas garante o pleno funcionamento do sistema operacional eliminando erros que podem comprometer o desempenho ou causar problemas ainda maiores.

Permissões

O NTFS possibilita o uso de permissões no sistema operacional, ou seja, é possível definir como usuários - ou grupos de usuários podem acessar determinados arquivos ou determinadas pastas. Por exemplo, você pode permitir ao usuário Arthur Dent ter controle total da pastaInfoWester, mas só permitir ao usuário Marvin ler e executar o referido conteúdo, sem poder alterá-lo.



Cada conta de usuário criada no sistema (ou grupo) recebe um código único chamado Security Identifier (SID). Assim, se um usuário for eliminado e, posteriormente, outro for criado com o mesmo nome, será necessário reaplicar as permissões, pois o SID deste será diferente, apesar da denominação igual.

Master File Table (MFT)

FAT é a sigla para File Allocation Table e recebe este nome porque trabalha com uma tabela que, basicamente, indica onde estão os dados de cada arquivo. O NTFS, porém, utiliza uma estrutura chamada Master File Table (MFT), que tem praticamente a mesma finalidade do FAT, mas funciona de maneira diferente.

O MFT é uma tabela que registra atributos de cada arquivo armazenado. Esses atributos consistem em uma série de informações, entre elas: nome, data da última modificação, permissões (conceito explicado no tópico anterior) e, principalmente, localização na unidade de armazenamento.

Como necessita guardar várias informações de praticamente todos os arquivos no disco, o NTFS reserva um espaço para o MFT - Zona MFT -, geralmente de 12,5% do tamanho da partição. Cada arquivo pode necessitar de pelo menos 1 KB para o registro de seus atributos no MFT, daí a necessidade de um espaço considerável para este.

Outros Recursos do NTFS

O NTFS é dotado de vários recursos de natureza complementar ou definitiva que enriquecem suas características de segurança, desempenho e confiabilidade. A seguir, algumas delas.

Encrypting File System (EFS)

Este é um recurso que surgiu com o Windows 2000 e, tal como o nome sugere, é um reforço de segurança, pois permite a proteção de dados por criptografia com o uso do esquema de chaves públicas (saiba mais sobre isso clicando no link anterior). A principal vantagem é que o dono dos arquivos protegidos pode determinar quais usuários podem acessá-los. Esse conteúdo é criptografado quando o usuário o fecha, mas passa a estar imediatamente pronto para uso quando aberto.

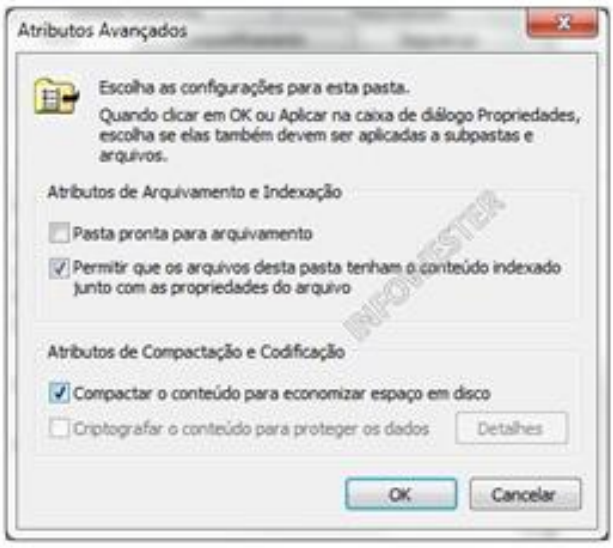
Note que o EFS não está plenamente disponível em todas as versões do Windows compatíveis com NTFS, como o Windows 7 Home Basic, por exemplo. Além disso, vale frisar também que não é possível utilizar criptografia em conteúdo compactado (tópico a seguir).

Compressão dbe Dados Outro

Outro recurso interessante do NTFS é a sua capacidade de lidar com compressão de dados para economizar espaço em disco. Essa compactação consiste, basicamente, em aproveitar estruturas repetidas de arquivos para reduzir seu tamanho.

O FAT também conta com essa capacidade, mas de maneira limitada, uma vez que é necessário compactar a partição como um todo. No NTFS, é possível a compressão somente de pastas ou de determinados arquivos.

Para compactar uma pasta no Windows 7, por exemplo, basta clicar nela com o botão direito do mouse e selecionar Propriedades. Na janela que surgir, basca clicar no botão Avançados da aba Geral e marcar a opção compactar o conteúdo para economizar espaço em disco.



É claro que também é possível comprimir uma partição inteira: clique com o botão direito do mouse na unidade, selecione Propriedades e, na aba Geral, marque a opção compactar este disco para economizar espaço.

Vale frisar que o usuário não precisa se preocupar em descompactar os arquivos quando precisar trabalhar com eles. Ao acessá-los, o próprio sistema operacional se encarrega disso. Além disso, é recomendável utilizar esse recurso com moderação e apenas em caso de necessidade para evitar possíveis problemas de desempenho.

Quotas de Disco

Assim como o EFS, esta é uma funcionalidade mais recente e que não está disponível em todas as versões do Windows compatíveis com NTFS. Sua função, basicamente, é a de permitir ao administrador do sistema definir quanto espaço em disco cada usuário pode utilizar. Dessa forma, pode-se evitar problemas de desempenho ou o esgotamento dos recursos de armazenamento de um servidor, por exemplo.

Versões do NTFS

Você já sabe que O NTFS não é, necessariamente, um sistema de arquivos novo e que, apesar disso, é utilizado com sucesso pela Microsoft até os dias de hoje. Então, nada mais natural ele sofrer atualizações para se adaptar às necessidades das versões mais atuais do Windows.

A versão mais conhecida do Windows NT, o Windows NT 4, fazia uso do NTFS 1.2. Aparentemente, houve versões antes desta: a 1.0, que foi utilizada no Windows NT 3.1, e a 1.1, aplicado ao Windows NT 3.5. O NTFS 1.2 passou a ser chamado de NTFS 4 em alusão à existência do número em questão no nome "Windows NT 4", embora esta versão do NTFS também tenha sido utilizada no Windows NT 3.51.

O Windows NT ficou um bom tempo no mercado, mas depois foi substituído pelo Windows 2000, que trouxe também o NTFS 5, versão dotada de vários novos recursos, entre eles: Reparse Points, onde arquivos e pastas dentro do sistema de arquivos podem ter ações associadas a eles, de forma que operações particulares a estes arquivos possam ser executadas; quotas de discos; encriptação (EFS); suporte a dados esparsos, onde é possível armazenar de forma eficiente arquivos grandes mas que possuem estruturas vazias.

Revisões do NTFS 5 foram lançadas para os sistemas operacionais Windows XP e Serve 2003. O NTFS 6 e suas variações surgiram para as versões Vista, 7 e Server 2008 do Windows.

Novas Versões Podem Surgir Com O Lançamento De Outras Edições Do Windows.

Até quando o NTFS será utilizado? Para o lançamento do Windows Vista, a Microsoft estava trabalhando em um novo sistema de arquivos chamado WinFS, cuja principal característica seria sua capacidade de trabalhar como um banco de dados relacional, permitindo ao usuário localizar facilmente os arquivos que procura.

Uma série de problemas fez com que o WinFS não fosse implementado no Windows Vista, permitindo ao NTFS manter o seu "reinado". Há, inclusive, rumores de que o projeto WinFS tenha sido cancelado.

A Microsoft continua trabalhando no desenvolvimento de novas versões de seus sistemas operacionais, mas pelo menos até a publicação deste artigo, não havia nenhuma informação sobre um novo sistema de arquivos, indicando que o NTFS terá ainda uma longa vida pela frente.

Sistemas de Arquivos FAT16 E FAT32

FAT16 e FAT 32 são nomes de sistemas de arquivos (file systems) utilizados por padrão em versões antigas do sistema operacional Windows (como o Windows 98, por exemplo), da Microsoft. Neste artigo, você conhecerá as principais características de cada um deles, assim como as diferenças entre ambos

Não é possível gravar dados em um HD ou em qualquer outro dispositivo de armazenamento de forma a manter as informações acessíveis e organizadas sem um sistema de arquivos - essencialmente um tipo de estrutura que indica como os arquivos devem ser gravados e lidos pelo sistema operacional do computador.

É o sistema de arquivos que determina como as informações podem ser guardadas, acessadas, copiadas, alteradas, nomeadas e até apagadas.

Ou seja, resumindo, toda e qualquer manipulação de dados em um dispositivo de armazenamento necessita de um sistema de arquivos para que estas ações sejam possíveis. Em resumo, sem um sistema de arquivos, os dados armazenados seriam apenas um conjunto de bits sem utilidade.

Há vários sistemas de arquivos disponíveis, para os mais diversos sistemas operacionais e para as mais variadas finalidades. Por exemplo, sistemas de arquivos utilizados em aplicações críticas - servidores de internet, por exemplo - costumam ter mais recursos para segurança ou para armazenamento de grandes volumes de dados.

Sistema de Arquivos FAT

FAT é a sigla para File Allocation Table (traduzindo: Tabela deAlocação de Arquivos). A primeira versão do FAT surgiu em 1977, para trabalhar com o sistema operacional MS-DOS, mas foi padrão até o Windows 95.

Trata-se de um sistema de arquivos que funciona com base em uma espécie de tabela que indica onde estão os dados de cada arquivo. Esse esquema é necessário porque o espaço destinado ao armazenamento é dividido em blocos, e cada arquivo gravado pode ocupar vários destes, mas não necessariamente de maneira sequencial: os blocos podem estar em várias posições diferentes. Assim, a tabela acaba atuando como um "guia" para localizá-los.

Com o surgimento de dispositivos de armazenamento mais sofisticados e com maior capacidade, o sistema FAT foi ganhando revisões, identificadas pelos nomes FAT12 e FAT16, sendo o primeiro quase um desconhecido e o último padrão dos sistemas operacionais da Microsoft por muito tempo.

As versões surgem com o intuito de eliminar determinadas limitações do sistema de arquivos anterior. O próprio FAT16, por exemplo, passou por isso: esta versão só trabalha com, no máximo, 2 GB, assim, para aplicá-lo em um disco de 5 GB, seria necessário dividi-lo em 3 partições (2 GB + 2 GB + 1 GB, por exemplo) para ser possível o aproveitamento de toda a capacidade da unidade.

Diante deste e de outros problemas, a Microsoft lançou, em 1996, o FAT32, que se tornou o sistema de arquivos do Windows 95 (versão OSR 2) e do Windows 98, sendo também compatível com versões lançadas posteriormente, como Windows 2000 e Windows XP, embora estes tenham um sistema de arquivos mais avançado, o NTFS.

Entendendo os Sistemas de Arquivos FAT

Em um disco rígido, a área de armazenamento é dividida em trilhas. Cada trilha é subdividida em setores (saiba mais neste artigo sobre HDs), cada um com 512 bytes, geralmente. Desse modo, é de se presumir que os sistemas de arquivos FAT trabalhem diretamente com esses setores. Mas não é bem assim.

Na verdade, o FAT trabalha com grupos de setores, onde cada um recebe a denominação cluster (ou unidade de alocação). No caso do FAT16, cada cluster pode ter, comumente, um dos seguintes tamanhos: 2 KB, 4 KB, 8 KB, 16 KB e, por fim, 32 KB.

A definição desse tamanho é uniforme, ou seja, não pode haver tamanhos variados de clusters em uma mesma unidade de armazenamento.

Cada arquivo gravado utiliza tantos clusters quanto forem necessários para cobrir o seu tamanho. Se, por exemplo, tivermos um arquivo com 50 KB, é possível guardá-lo em dois clusters de 32 KB cada. Você deve ter percebido então que, neste caso, um cluster ficou com espaço sobrando.

Esta área pode ser destinada a outro arquivo, correto? Errado! Acontece que cada cluster só pode ser utilizado por um único arquivo. Se sobrar espaço, este permanecerá vazio. Esse é um dos problemas do sistema FAT: desperdício.

Normalmente, o tamanho dos clusters é definido no procedimento de instalação do sistema operacional, na etapa de formatação da unidade de armazenamento.

Diferenças Entre FAT16 E FAT32

O FAT16 utiliza 16 bits para endereçamento dos dados (daí o número 16 na sigla), o que, na prática, significa que o sistema de arquivos pode trabalhar com até 65536 clusters, no máximo. Para chegar a este número, basta fazer 2 elevados a 16 (65536).

Se temos então até 65536 clusters e cada um pode ter até 32 KB de tamanho, significa que o sistema FAT16 é capaz de trabalhar com discos ou partições com até 2 GB: 65536 x 32 = 2.097.152 KB, que corresponde a 2 GB.

O sistema de arquivos FAT32 consegue solucionar esse problema por utilizar 32 bits no endereçamento de dados (novamente, aqui você pode perceber o porquê do número na sigla).

No FAT16, quanto maior o espaço em disco (considerando o limite de até 2 GB, é claro), maior o tamanho do cluster. Com o FAT32, é possível usar clusters menores - geralmente de 4 KB - mesmo com a unidade oferecendo maior capacidade de armazenamento. Desta forma, o desperdício acaba sendo menor.

O limite do FAT32 é de 2 TB (terabytes). Perceba, no entanto, que se você fizer o cálculo anterior considerando 32 em vez de 16 (2 elevados a 32) e, posteriormente, multiplicar o resultado pelo tamanho máximo do cluster (também 32), o valor obtido será de 128 TB. Então, qual o motivo do limite de 2 TB?

Na verdade, cada endereçamento tem tamanho de 32 bits, mas, no FAT32, o número máximo de clusters é calculado considerando apenas 28 bits, fazendo a conta ser 2 elevados a 28, que é igual a 268.435.456, ou seja, pouco mais de 268 milhões de clusters. Multiplicando esse número por 32, teremos então 8 TB.

Ok, novamente a conta não fechou, afinal, 8 TB para 2 TB é uma diferença muito grande. A explicação está no fato de que a Microsoft limitou o FAT32 a ter 2 elevados a 32 como quantidade máxima de setores, não de clusters (se fosse diferente, poderia haver problemas com a inicialização do sistema operacional devido a limitações na área de boot).

Como cada setor, geralmente, possui 512 bytes (ou 0,5 kilobyte), a conta seria 2 elevados a 32 (4.294967296) multiplicado por 0,5, que é igual 2.147.483.648 KB ou 2 TB.

Fragmentação

Para os usuários do Windows 95/98, uma recomendação era frequente: utilizar um aplicativo de desfragmentação de disco regularmente. Isso tem um bom motivo: toda vez que um arquivo é apagado, seus clusters ficam disponíveis para nova utilização. Acontece que o sistema operacional sempre ocupa os primeiros clusters livres e, se houver áreas ocupadas no caminho, continuará utilizando os clusters livres subsequentes.

O resultado é que, com o passar do tempo, há fragmentos de dados por todo o disco. Isso torna o acesso aos arquivos mais lento, uma vez que o sistema precisa procurar "pedaço por pedaço". A desfragmentação consegue amenizar este problema porque reorganiza os arquivos em clusters sequenciais, deixando-os acessíveis mais rapidamente.

VFAT

VFAT é a sigla para Virtual File Allocation Table. Trata-se de uma espécie de extensão para FAT16 introduzido no Windows 95 para que este sistema operacional possa suportar arquivos com mais de 11 caracteres no nome.

É que, por padrão, o FAT16 limita o tamanho dos nomes para 8 caracteres mais 3 destinados à extensão, por exemplo, daniella.txt. Caso nomes maiores sejam utilizados, estes aparecem de forma abreviada no sistema.

Com o uso do VFAT, as características do FAT16 são mantidas e, ao mesmo tempo, nomes maiores podem ser utilizados sem qualquer dificuldade. Isso porque o VFAT faz o sistema enxergar o nome abreviado, mas guarda o nome original em uma área separada.

O FAT32 trabalha com VFAT por padrão.

FAT12

Antes do sistema de arquivos FAT16, que bem conhecemos, existiu o FAT12, um sistema ainda mais primitivo, utilizado em disquetes e também nas primeiras versões do MS-DOS. Nele, são usados endereços de apenas 12 bits para endereçar os clusters, permitindo um total de 4096 clusters de até 4 KB, o que permitia partições de até 16 MB.

Em 1981, quando o IBM PC foi lançado, 16 MB parecia ser uma capacidade satisfatória, já que naquela época os discos rígidos tinham apenas 5 ou 10 MB. Claro que, em se tratando de informática, por maior que seja um limite, ele jamais será suficiente por muito tempo.

Um excelente exemplo é a célebre frase "Por que alguém iria precisar de mais de 640 KB de memória RAM?" dita por Bill Gates em uma entrevista, no início da década de 80. Logo começaram a ser usados discos de 40, 80 ou 120 MB, obrigando a Microsoft a criar a FAT 16, e incluí-la na versão 4.0 do MS-DOS.

Apesar de obsoleto, o FAT12 ainda continua vivo até os dias de hoje, fazendo compania para outro fantasma da informática: os disquetes. Por ser mais simples, o FAT12 é o sistema padrão para a formatação dos disquetes de 1.44, onde são usados clusters de apenas 512 bytes.

EXT2

Ext2 foi projetado e implementado para corrigir as deficiências do Ext e prover um sistema que respeitasse a semântica UNIX. A influência do UNIX pode ser vista, por exemplo, na utilização de grupos de blocos, que são análogos aos grupos de cilindros utilizados pelo FFS.

O bloco, que consiste num conjunto de setores (cada setor tem 512 bytes), é a menor unidade de alocação para o Ext2. O tamanho pode ser de 1024, 2048 ou 4096 bytes e é definido na formatação.

Quando é realizada uma operação de escrita em um arquivo, o Ext2 tenta, sempre que possível, alocar blocos de dados no mesmo grupo que contém o nó-i. Esse comportamento reduz o movimento da (s) cabeça (s) de leitura-gravação da unidade de disco.

Em um sistema de arquivos ocorrem dois tipos de fragmentação:

- A fragmentação interna (ou de espaço) é causada pelo fato do tamanho do arquivo geralmente não ser múltiplo do tamanho do bloco (portanto o último bloco terá um espaço não utilizado) a consequência é a perda de espaço;

- A fragmentação externa (ou de arquivo) decorre da impossibilidade do sistema determinar, a priori, qual o tamanho do arquivo (p.ex., arquivos de texto e de logs são muito modificados, e o seu tamanho pode aumentar ou diminuir) assim um arquivo pode alocar blocos não contíguos, prejudicando o desempenho.

- Para diminuir o impacto do primeiro tipo, existem duas estratégias básicas. A primeira, mais simples, é determinar, na formatação, o menor tamanho de bloco possível. O Ext2 permite tamanhos de blocos de 1024, 2048 e 4096 bytes.

- Um tamanho de bloco pequeno, como 1024 bytes, diminui a fragmentação e perda de espaço, mas em contrapartida gera um impacto negativo no desempenho, pois acarreta o gerenciamento de uma maior quantidade de blocos. O tamanho de bloco padrão para volumes grandes é de 4096 bytes.

- A segunda estratégia é alocar a parte final de um arquivo, menor que o tamanho de um bloco, juntamente com pedaços de outros arquivos. O Reiserfs chama esse método de tail packing; o UFS usa fragmentos, que são submúltiplos do tamanho do bloco. Apesar do Ext2 possuir, no superbloco, a previsão para uso de fragmentos, esse método não foi implementado.

Para diminuir o impacto da fragmentação externa, o Ext2 pré-aloca (reserva) até oito blocos quando um arquivo é aberto para gravação. Esses blocos reservados, quando possível, são adjacentes ao último bloco utilizado pelo arquivo. [CARD, TS'O e TWEEDIE, 1994]

EXT3

O Ext3 (Third Extended file system) é um sistema de arquivos desenvolvido por Stephen C. Tweedie para o Linux, que acrescenta alguns recursos ao Ext2, dos quais o mais visível é o journaling, que consiste em um registro (log ou journal) de transações cuja finalidade é recuperar o sistema em caso de desligamento não programado.

Há três níveis de journaling disponíveis na implementação do Ext3:

Journal: os metadados e os dados (conteúdo) dos arquivos são escritos no journal antes de serem de fato escritos no sistema de arquivos principal. Isso aumenta a confiabilidade do sistema com uma perda de desempenho, devido a necessidade de todos os dados serem escritos no disco duas vezes.

Writeback: os metadados são escritos no journal mas não o conteúdo dos arquivos. Essa opção permite um melhor desempenho em relação ao modo journal, porém introduz o risco de escrita fora de ordem onde, por exemplo, arquivos que são apensados durante um crash podem ter adicionados a eles trechos de lixo na próxima montagem.

Ordered: é como o writeback, mas força que a escrita do conteúdo dos arquivos seja feita após a marcação de seus metadados como escritos no journal. Esse é considerado um meio-termo aceitável entre confiabilidade e performance, sendo, portanto, o nível padrão.

Embora o seu desempenho (velocidade) seja menos atrativo que o de outros sistemas de arquivos (como ReiserFS e XFS), ele tem a importante vantagem de permitir que seja feita a atualização direta a partir de um sistema com ext2, sem a necessidade de realizar um backup e restaurar posteriormente os dados, bem como o menor consumo de processamento.

Enquanto em alguns contextos a falta de funções de sistemas de arquivos “modernos”, como alocação dinâmica de inodes e estruturas de dados em árvore, poderia ser considerada uma desvantagem, em termos de “recuperabilidade” isso dá ao ext3 uma significante vantagem sobre sistemas de arquivos que as possuem.

Os metadados do sistema de arquivos estão todos em locais fixos e bem conhecidos, e há certa redundância inerente à estrutura de dados, que permite que sistemas ext2 e ext3 sejam recuperáveis no caso de uma corrupção de dados significante, em que sistemas de arquivos em árvore não seriam recuperáveis.

Introdução ao Sistema de Arquivos Reiserfs

O sistema de arquivos ReiserFS teve sua primeira aparição no ano de 2001 pelas mãos de Hans Reiser (daí o nome do padrão), que também montou uma equipe de nome NAMESYS para gerenciar os trabalhos do projeto. Desde então, o ReiserFS vem sendo cada vez mais utilizado, principalmente por estar disponível como padrão em muitas das distribuições Linux, fazendo frente ao sistema de arquivos ext3.

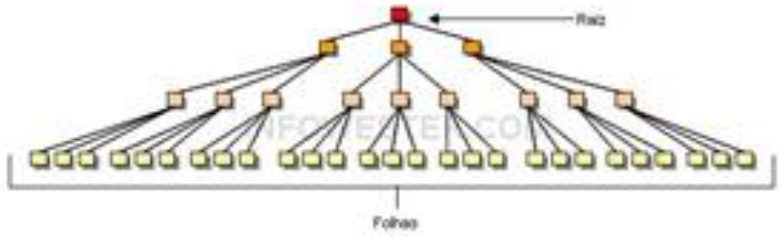
A boa aceitação do ReiserFS é devida ao seu conjunto de características, que o tornam um sistema de arquivos seguro, eficiente, rápido e confiável. Entre seus principais recursos, tem-se:

- Journaling, um recurso que ajuda a manter a integridade dos dados em caso de erros no sistema causados por desligamento incorreto ou determinadas falhas de hardware, por exemplo. O journaling é uma das características mais importantes do ReiserFS, motivo pelo qual é explicado com mais detalhes adiante;

- Suporte a arquivos com mais de 2 GB (limitação existente em alguns filesystems);

- Organização dos objetos do sistema de arquivos em uma estrutura de dados chamada B+Trees(árvores B+). Nesse esquema, os dados são fixados em posições organizadas por divisões denominadas folhas. Por sua vez, as folhas são organizadas por nós ou ponteiros chamados de sub-árvores, que estão ligados a um nó raiz (ver ilustração abaixo para entender melhor).

Esse processo organizacional exige algoritmos mais complexos, porém apresenta performance superior na gravação e no acesso aos dados, se comparado a outros sistemas de arquivos;



Alocação dinâmica de inodes (em poucas palavras, inodes são estruturas que contém informações sobre os arquivos), diminuindo o desperdício de espaço.

Outros sistemas de arquivos têm blocos de tamanho fixo para alocação, assim, se não for necessário usar um bloco inteiro, o espaço restante fica em desuso. No ReiserFS, a alocação é feita com base no tamanho do arquivo.

Organização

O ReiserFS tem como referência o já mencionado esquema B+Trees para organizar e localizar os itens que compõem, em sua essência, todo o sistema de arquivos, isto é, os dados em si e as informações associadas (data de criação, permissões de acesso, proprietário, etc). Basicamente, todos esses itens são classificados em diretórios (directory items), blocos de dados diretos e indiretos (direct items / indirect items) e inodes (stat data items).

Nos inodes é que são classificadas as informações referentes a cada arquivo, isto é, os metadados. Os direct items são os arquivos em si, mas quando armazenados nas "folhas" das sub-árvores e, embora tenham tamanho variável, são compostos por blocos de dados pequenos.

Os direct items ficam próximos aos metadados, já que tanto um como o outro são organizados nas árvores. Já os blocos maiores, isto é, os indirect items, não são incluídos nas árvores (por isso recebem esse nome) e são "localizáveis" por ponteiros que indicam onde estão armazenados.

Todo esse esquema acaba fazendo com que o espaço em disco seja melhor aproveitado no ReiserFS. Por outro lado, também há desvantagens, já que pode causar maior fragmentação de dados, assim como exigir mais recursos de processamento.

Quando o assunto é sistema de arquivos no Linux, é inevitável as comparações entre o ReiserFS e o ext3, além de outros menos usados, como o XFS. Nada mais natural, afinal, cada usuário procura o melhor para o seu computador.

Mas apontar qual realmente é melhor não é uma tarefa fácil. Há vários comparativos na internet que tentam oferecer essa resposta, mas, aqui no InfoWester, somos da opinião de que a melhor coisa a se fazer é testar os sistemas de arquivos que lhe atraíram para definir qual mais lhe agrada.

Introdução Ao LVM - Gerenciamento de Volumes Lógicos

Mudar de sistema operacional é algo que pode intimidar bastante aqueles que estão conhecendo o Linux agora. Embora ele seja bem parecido com outros sistemas operacionais baseados no UNIX, como o BSD, o Solaris e o OS X, ele é muito diferente do Windows.

Hoje em dia a maioria das distribuições é muito fácil de usar e (seja isso bom ou ruim) abstrai o sistema complexo e poderoso que há por baixo do capô. Com isso as distribuições se tornam fáceis de usar, e os usuários muitas vezes sabem pouco sobre o sistema em si. Hoje vamos dar uma olhada no LVM – Gerenciamento de Unidades Lógicas, que oferece aos usuários a capacidade de redimensionar partições enquanto elas estão em uso.

Primeiro, algumas informações sobre HDs e partições. O HD é onde todos os seus dados são armazenados permanentemente. Não confunda com a memória do sistema, usada para armazenar informações temporariamente como ocorre quando você executa aplicativos ou cria arquivos.

Quem vem do Windows deve conhecer as letras de unidades usadas pelo sistema, como C: e D: (unidades C e D).

A maioria dos usuários sabe que seus arquivos estão ali, mas nem todos sabem o que é o C:. Para armazenar informações no disco rígido, o computador precisa saber como ler e escrever dados nele. É preciso dizer ao computador em quais áreas do HD ele pode escrever. Essas áreas são as partições. O disco pode ser todo ocupado por uma única partição ou ser dividido em várias partições menores.

O computador também precisa saber como armazenar os dados em cada partição, e isso é feito com a criação de um sistema de arquivos nela. O Linux tem vários sistemas de arquivos excelentes para você escolher, incluindo (dentre outros) ReiserFS, XFS, JFS, Btrfs, ext2, ext3 e agora o ext4. O Windows geralmente usa os sistemas de arquivo NTFS e FAT32, enquanto o OS X usa o HFS+. A maioria dos dispositivos para o consumidor, como cartões de memória, vem formatada em FAT32.

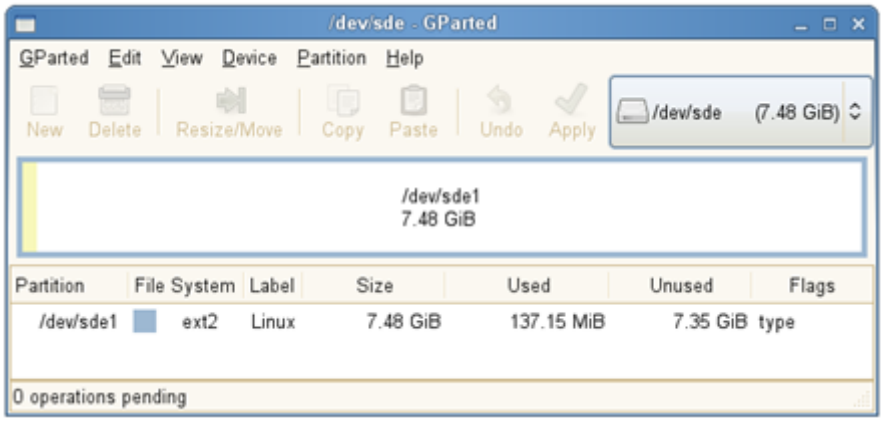
As informações sobre as partições são armazenadas no próprio HD, no que chamamos de tabela de partições. A maioria dos computadores pessoais usa a tabela de partições do MSDOS, embora a Apple use a GPT, que é plenamente suportada pelo Linux. Por padrão, um HD vazio não tem tabela de partições, mas no primeiro particionamento uma tabela é criada. Há muitas ferramentas que você pode usar para particionar suas unidades no Linux, como o fdisk, o parted e sua interface gráfica, o GParted. No Linux, todos os dispositivos respondem por um arquivo no diretório /dev. O primeiro terminal, por exemplo, é o /dev/tty0.

Os nomes de HDs convencionais são relativos à sua posição no subsistema do computador. Por exemplo, /dev/hda refere-se ao disco master na porta IDE primária da placa-mãe. Hoje em dia a maioria dos computadores usa controladoras SATA, logo, a primeira unidade geralmente é /dev/sda. Cada partição da unidade recebe um número. A primeira partição primária é a número 1, a segundo é a número 2, e por aí vai.

A tabela de partições do MSDOS suporta apenas um máximo de quatro partições (que chamamos de primárias), mas suporta um tipo de partição que chamamos de “estendida”. As partições estendidas permitem a criação de um número ilimitado de partições extras (as partições lógicas), que no Linux começam sempre na quinta partição. Com base no número da partição, é possível dizer onde se encontram as partições em relação às outras partições do disco.

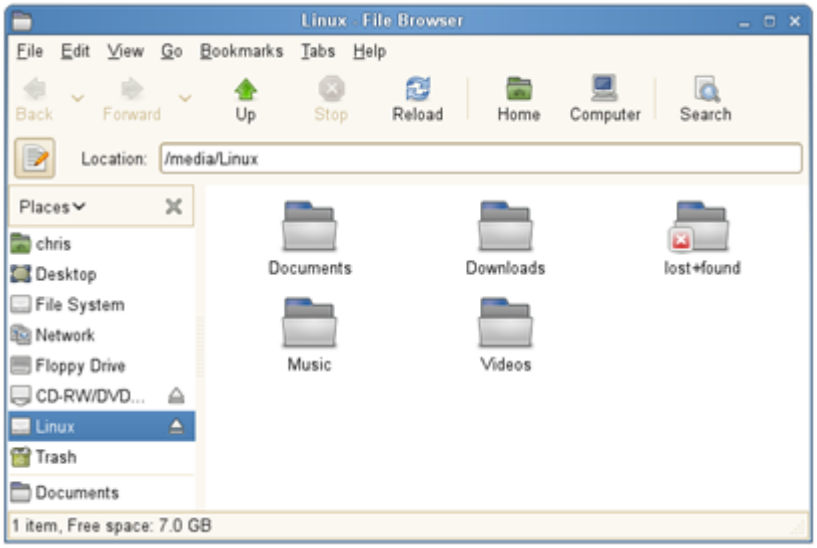
Em termos de sistema de arquivos, o Linux usa uma hierarquia onde tudo fica abaixo do diretório / (root, ou raiz). Ao instalar uma distribuição, você deve criar ao menos uma partição, que será usada como o sistema de arquivos root.

Se você veio do Windows, pode considerar o root mais ou menos (BEM mais ou menos) como se fosse a unidade C. Muitas distribuições Linux criam apenas uma partição para todos os dados. Segue abaixo um exemplo das informações sobre as partições de um HD. É uma unidade de 8 GB (/dev/sde) com uma única partição (/dev/sde1) do tipo “Linux”.



Para ter acesso aos dados desse dispositivo, é preciso atribuir um ponto de montagem a ele. Um ponto de montagem é um diretório do computador que o Linux associa a uma partição.

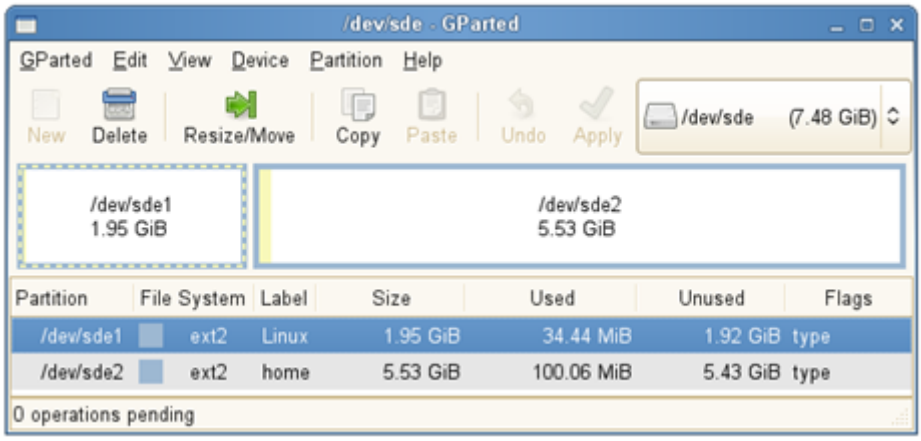
Tudo o que for gravado nesse ponto de montagem será gravado fisicamente na partição à qual ele corresponde. O dispositivo /dev/sde1 está montado em /media/Linux, como se pode ver na foto.



Estamos todos razoavelmente familiarizados com esse conceito quando usamos mídias removíveis, como memory sticks, mas isso também pode ser aplicado a todo o sistema. Uma das grandes vantagens de se usar um sistema hierárquico desses é que é possível atribuir partições extras a diretórios mais baixos para preservar seus dados.

O Linux precisa no mínimo de uma partição root (/), mas você pode criar uma segunda partição para pastas como /home, onde todos os dados dos usuários são armazenados. E por que você criaria uma partição separada só para isso?

Bom, se for preciso reinstalar sua distribuição Linux por algum motivo, não será preciso fazer backup dos dados, bastando limpar a primeira partição e montar a segunda como /home, sem formatá-la. Feito isso, todos os dados e configurações permanecerão exatamente como eram! Eis um exemplo de HD com uma partição para / e outra para /home.



Esse excelente recurso é muito útil, mas como saber quanto espaço atribuir às partições? E se você ficar sem espaço depois? Você poderia excluir dados ou movê-los para outras partições, mas há uma maneira muito mais poderosa e flexível. Chama-se LVM – Gerenciamento de Volumes Lógicos O LVM é uma maneira de criar, excluir, redimensionar e expandir partições do computador.

Ele é ótimo não apenas para servidores, mas também para desktops! Como o LVM funciona? Ao invés de abrigar as informações sobre as partições na tabela de partições, o LVM escreve suas próprias informações em separado e mantém o controle sobre a localização das partições, quais dispositivos são partes delas e o tamanho de cada uma.

Com ele você pode ter uma partição root e outra partição home, mas se ficar sem espaço é só dizer ao LVM para expandir a partição desejada e pronto, você terá mais espaço disponível. E você ainda pode adicionar outros discos rígidos ao sistema e dizer ao LVM para incluí-los também!

A maioria das distribuições Linux modernas suportam dispositivos LVM durante a instalação, e boa parte desse processo é gerenciada automaticamente. Nos exemplos acima temos uma partição do tipo Linux, que é formatada com um sistema de arquivos e montada em um diretório do sistema. Bem básico.

Ao usar o LVM, defina o tipo de partição como Linux LVM e depois use as ferramentas em espaço de usuário para criar e gerenciar os dispositivos. É importante destacar que o GParted não tem suporte a partições LVM, logo, você terá que usar outras ferramentas.

Se estiver usando o openSUSE, ele traz uma excelente ferramenta de gerenciamento para LVM como parte do YaST. Você vai precisar de um módulo do kernel, bem como de ferramentas em espaço de usuário. Instale tudo isso pela ferramenta de gerenciamento de pacotes do seu sistema. Segue um exemplo para o Debian.

modprobe dm-mod ; apt-get install lvm2

Criação De Dispositivos LVM

O LVM consiste em alguns elementos. Primeiro, é necessária uma partição física do tipo Linux LVMque será o volume físico do LVM. A seguir, nós criamos o grupo lógico ao qual atribuiremos a partição física. Depois criamos as partições individuais, chamadas de volumes lógicos. Vamos dar uma olhada nisso tudo como usuário root.

Etapa 1

Crie uma partição em um HD vazio e defina-a com o tipo LVM Linux (8e).

Observação: meu dispositivo é /dev/sde mas o seu provavelmente é diferente!

fdisk /dev/sde
n
p
1
[Enter] [Enter] t
8e
w

O dispositivo deve aparecer mais ou menos assim.

fdisk -l /dev/sde

Disk /dev/sde: 8040 MB, 8040480256 bytes
255 heads, 63 sectors/track, 977 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000bf9ae

Device Boot Start End Blocks Id System
/dev/sde1 1 977 7847721 8e Linux LVM

Aqui vemos uma única partição, do tipo LVM Linux (8e).

Etapa 2

Agora temos que dizer ao LVM para usar essa partição como volume físico.

pvcreate /dev/sde1

No physical volume label read from /dev/sde1
Physical volume “/dev/sde1” successfully created

This device has now been added to the LVM pool.

Etapa 3

Crie um grupo de volume no volume físico, que chamaremos de ‘sistema’.

vgcreate system /dev/sde1
Volume group “system” successfully created

Etapa 4

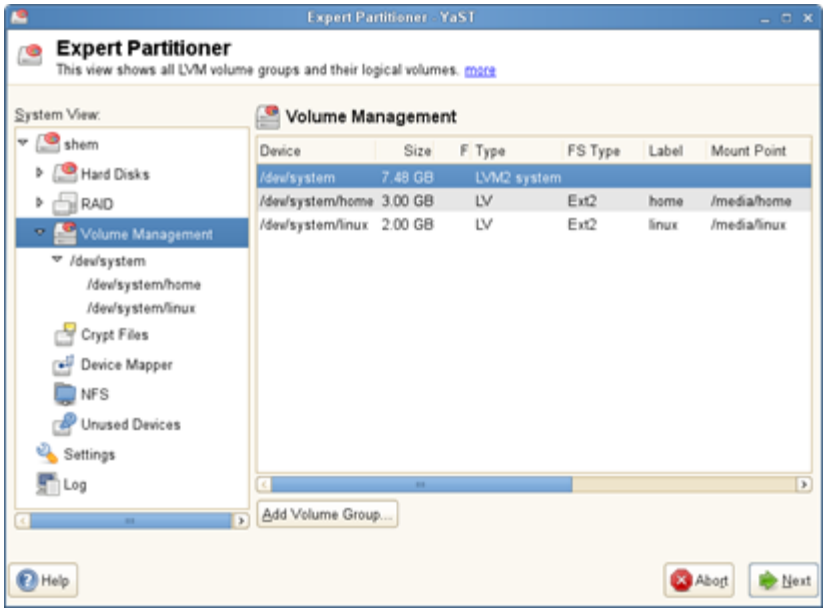
Agora que criamos um volume físico e um grupo de volume, é hora de criar um volume lógico. Vamos criar um para o root (/) chamado linux e outro para a home.

lvcreate -n linux -L 2G system
Logical volume “linux” created

lvcreate -n home -L 3G system
Logical volume “home” created

Pronto! Agora temos duas partições novas que podem ser formatadas como se fossem partições comuns. Se olharmos no diretório /dev veremos nossos novos dispositivos.

ls -l /dev/system/
total 0
lrwxrwxrwx 1 root root 23 Mar 9 17:18 home -> /dev/mapper/system-home
lrwxrwxrwx 1 root root 24 Mar 9 17:17 linux -> /dev/mapper/system-linux



Ferramentas Para Consulta

Antes de irmos em frente, vamos ver mais algumas ferramentas LVM para exibir o status dos nossos dispositivos. Lembre-se de que temos três componentes diferentes que compõem uma partição LVM completa, o volume físico (PV, ou Physical Volume), o grupo de volume (VG, ou Volume Group) e o volume lógico (LV, ou Logical Volume).

Vamos dar uma olhada no volume físico (PV).

pvdisplay
— Physical volume —
PV Name /dev/sde1
VG Name system
PV Size 7.48 GB / not usable 3.79 MB
Allocatable yes
PE Size (KByte) 4096
Total PE 1915

Free PE 1915
Allocated PE 0
PV UUID 7vkgGI-e402-K3hE-XGJz-kl4C-nI7o-oFqwA8

Aqui podemos ver o nome do volume físico (a partição física que criamos), o grupo de volume ao qual a partição foi atribuída (que chamamos de sistema) e outras informações relativas ao tamanho do volume.

Vamos dar uma olhada no grupo de volume (VG).

vgdisplay
— Volume group —
VG Name system
System ID
Format lvm2
Metadata Areas 1
Metadata Sequence No 3
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 2
Open LV 0
Max PV 0
Cur PV 1
Act PV 1
VG Size 7.48 GB
PE Size 4.00 MB
Total PE 1915
Alloc PE / Size 1280 / 5.00 GB
Free PE / Size 635 / 2.48 GB
VG UUID Z6TSXO-0DQ3-7Jiz-67k2-dEkY-dYR2-RNJE85

Aqui temos o nome do grupo de volume (que chamamos de sistema), seu tipo (lvm2), seu espaço total e o espaço que já foi atribuído (lembre-se de que criamos dois volumes lógicos, root e home).

Finalmente, vejamos o volume lógico (LV).

lvdisplay
— Logical volume —
LV Name /dev/system/linux
VG Name system
LV UUID L0qrZu-bwCp-rnEu-uJry-4j3n-XBLB-OWsXVx
LV Write Access read/write
LV Status available
open 0
LV Size 2.00 GB
Current LE 512
Segments 1
Allocation inherit
Read ahead sectors auto
– currently set to 256
Block device 253:0

Logical volume —
LV Name /dev/system/home
VG Name system
LV UUID AScHe0-q5sJ-F8eH-bYRy-3URL-Nt7m-0UFduW
LV Write Access read/write
LV Status available
open 0
LV Size 3.00 GB

Current LE 768
Segments 1
Allocation inherit
Read ahead sectors auto
– currently set to 256
Block device 253:1

Aqui temos as duas partições que criamos, home e Linux. Observe que o volume físico (PV) e o grupo de volume (VG) também possuem um identificador único, que o Linux usa para detectar e controlar os dispositivos.

Formatação

Agora que temos nossas duas partições, podemos formatá-las como qualquer outro dispositivo físico.

mke2fs /dev/system/linux -L linux
mke2fs 1.41.1 (01-Sep-2008)
Filesystem label=linux
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
131072 inodes, 524288 blocks
26214 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=536870912
16 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 36 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.

Elas também podem ser montadas como qualquer outro dispositivo.

Expansão de Partições

Agora que está usando o LVM, se você ficar sem espaço em uma partição, só terá que dizer ao LVM para atribuir mais espaço ao dispositivo em particular e redimensionar o sistema de arquivos. Embora você possa encolher partições, é bem mais seguro aumentá-las. Por isso eu recomendo a você nunca atribuir o tamanho total do volume físico aos volumes lógicos, mas sim começar pequeno e ir aumentando conforme a necessidade. Se estiver usando um sistema de arquivos ext, isso pode ser feito mesmo com as partições montadas (alguns outros sistemas de arquivos também permitem fazer isso).

A situação atual é esta, com 100% de utilização.

df -h
/dev/mapper/system-linux 2.0G 2.0G 0 100% /media/linux

Primeiro, aumente o volume lógico em 1 GB.

lvresize -L +1G /dev/system/linux
Extending logical volume linux to 3.00 GB
Logical volume linux successfully resized

Agora que aumentamos o dispositivo, temos que redimensionar o sistema de arquivos

resize2fs /dev/system/linux
Filesystem at /dev/system/linux is mounted on /media/linux; on-line resizing required

Resizing the filesystem on /dev/system/linux to 786432 (4k) blocks.
The filesystem on /dev/system/linux is now 786432 blocks long.

Depois de redimensionar o sistema de arquivos, eis o resultado.

df -h
/dev/mapper/system-linux 3.0G 2.0G 855M 71% /media/linux

Se quiser adicionar outro HD ao computador, basta instalá-lo no computador e repetir as etapas 1 e 2. Como o grupo de volume já existe, não é preciso criá-lo como na etapa 3. Em vez disso, adicione-o ao grupo existente.

vgextend system /dev/sdf1
Volume group “system” successfully extended

Agora você tem um disco inteiro novinho como espaço LVM adicional, pronto para ser atribuído a qualquer volume lógico que você quiser.

É assim que você pode usar o LVM para expandir e atribuir espaço na mesma hora, sem ter que mover dados ou desligar o computador!

Iniciando pelo LVM

É possível usar o LVM para o sistema inteiro. O único problema é que você vai precisar de uma partição adicional para o diretório /boot e vai ter que usar um disco de RAM inicial (initrd). A partição boot precisa de apenas 100 MB e irá armazenar o kernel do Linux e o initrd. O initrd vai abrigar as ferramentas em espaço de usuário para o LVM necessárias à detecção e ativação dos dispositivos LVM durante a inicialização, para que o resto do sistema seja carregado. O instalador da distribuição vai cuidar disso para você automaticamente.