

# Aula 21 – Sistema de Arquivos II

Norton Trevisan Roman  
Clodoaldo Aparecido de Moraes Lima

5 de dezembro de 2014

# I-Nodes

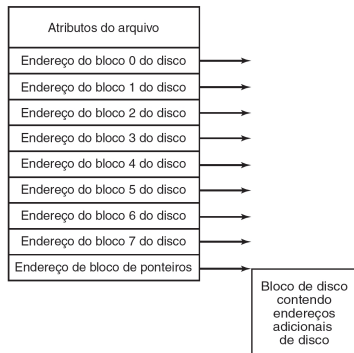
- Associa-se a cada arquivo uma estrutura de dados chamada i-node (index-node)
  - Listando os atributos e endereços em disco dos blocos do arquivo
  - Dado o i-node de um arquivo, é possível encontrar todos os blocos desse arquivo
  - Um único i-node por arquivo
- SOs: UNIX e Linux

# I-Nodes

- Vantagens:
  - O i-node somente é carregado na memória quando o seu respectivo arquivo estiver aberto (em uso)
    - Espaço de memória ocupado pelos i-nodes é proporcional ao número máximo de arquivos que podem ser abertos ao mesmo tempo
    - Enquanto o espaço de memória ocupado pela tabela de arquivo (FAT) é proporcional ao tamanho do disco
    - NTFS é proporcional ao número de arquivos no dispositivo
  - Se cada i-node ocupa  $n$  bytes, e no máximo  $k$  arquivos podem estar abertos ao mesmo tempo, o total de memória ocupada pelo arranjo contendo os i-nodes é  $kn$  bytes;

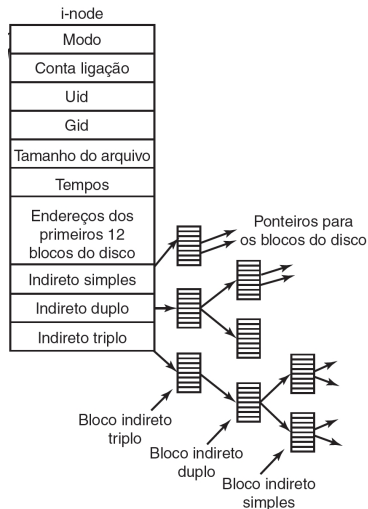
# I-Nodes

- Desvantagem:
  - Cada i-node tem espaço para um número fixo de endereços de disco
    - O tamanho do arquivo pode aumentar além deste limite
- Solução:
  - Reservar o último endereço para um bloco contendo outros endereços de blocos



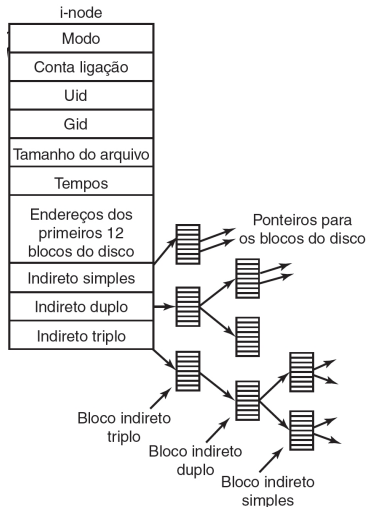
# I-Nodes

- Solução:
  - Pode-se ter até 2 ou mais desses blocos contendo endereços de disco
  - Ou mesmo blocos apontando a outros blocos de endereços
  - Ex: I-node no Linux e UNIX V7



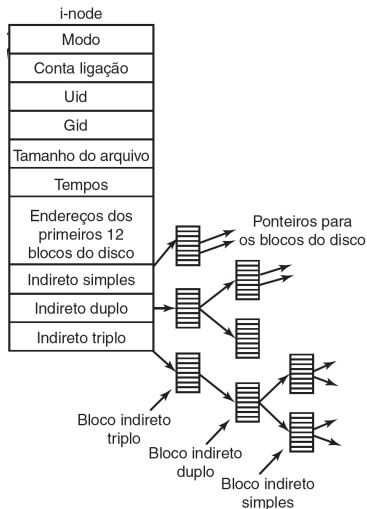
# I-Nodes: Padrão POSIX

- Atributos de arquivos regulares:
  - Tamanho do arquivo (B)
  - Device ID (dispositivo contendo o arquivo)
  - User ID do dono do arquivo
  - Group ID do arquivo
  - Modo (tipo de arquivo e como pode ser acessado)
  - Flags para proteção



# I-Nodes: Padrão POSIX

- Atributos de arquivos regulares:
  - Timestamps (quando o i-node foi modificado, quando o conteúdo do arquivo foi modificado, tempo do último acesso)
  - Contador de links (quantos links apontam para o i-node)
  - Ponteiros para os blocos de disco com o arquivo



# I-Nodes: Comandos Linux

- Descobrir o número do i-node

```
$ ls -li temp  
1320187 temp
```

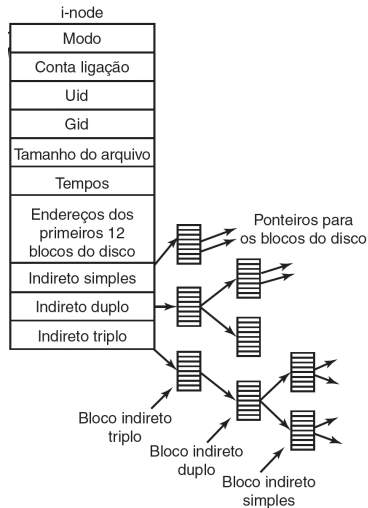
- Obter as informações de um arquivo

```
$ stat temp  
Arquivo: "temp"  
Tamanho: 4 Blocos: 8 Bloco IO: 4096 arquivo  
comum  
Dispositivo: 801h/2049d Inode: 1320187 Links: 1  
  
Acesso: (0664/-rw-rw-r--) Uid: ( 1530/ norton)  
Gid: ( 1205/ norton)  
Acessar: 2012-11-09 14:41:20.289825865 -0200  
Modificar: 2012-11-09 14:41:19.169825830 -0200  
Alterar: 2012-11-09 14:41:19.169825830 -0200  
Nascimento: -
```



# I-Nodes: Dispositivos

- No Linux, dispositivos em geral são representados como arquivos
  - I-node especial, contendo:
    - Major device number → usado para localizar o driver apropriado
    - Minor device number → passado como parâmetro ao driver, de modo a especificar a unidade específica a ser lida ou escrita



# Gerenciamento de Espaço em Disco

- Duas estratégias são possíveis para armazenar um arquivo de  $n$  bytes:
  - São alocados ao arquivo  $n$  bytes consecutivos do espaço disponível em disco
    - Se o arquivo crescer, provavelmente terá que ser movido para outra parte do disco
  - Arquivo é quebrado em um número de blocos não necessariamente contíguos
    - Blocos com tamanho fixo
    - Não necessariamente do mesmo tamanho usado pelo hardware do dispositivo (chamados de clusters, na nomenclatura da FAT)
    - A maioria dos sistemas de arquivos utilizam essa estratégia

# Implementação de Sistemas de Arquivos

- Retomando...
  - Antes que possa ser lido, um arquivo tem que ser aberto
  - Quando “abrimos” um arquivo, o SO usa o caminho fornecido para localizar a entrada no diretório
  - Essa entrada fornece a informação necessária para encontrar os blocos no disco
  - Dependendo do sistema, será
    - O endereço em disco do arquivo inteiro (alocação contígua)
    - Número do primeiro bloco (lista ligada ou FAT)
    - Índice do arquivo na MFT (NTFS)
    - Número do i-node

# Implementação de Diretórios

- Função do diretório: mapear o nome do arquivo à informação necessária para localizar os dados
- Onde armazenar os atributos relacionados a um arquivo?
  - Diretamente na entrada do Diretório:
    - Nesse caso, um diretório consiste de uma lista de entradas com tamanho fixo (uma para cada arquivo), contendo um nome de arquivo, uma estrutura de atributos de arquivos, e um ou mais endereços de disco para os blocos (ou ponteiro para a FAT)
  - MS/DOS e Windows (FAT)

jogos	atributos
correio eletrônico	atributos
notícias	atributos
trabalho	atributos

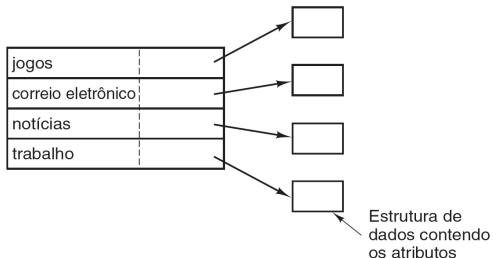
# Implementação de Diretórios

- NTFS
  - Tudo está na MFT (diretórios, subdiretórios e arquivos)
    - Os atributos do arquivo são mantidos no próprio arquivo
    - Um diretório guarda tão somente informação sobre si mesmo, não sobre os arquivos nele contidos
    - Buscas se dão na MFT
  - Registro da MFT para um diretório pequeno (diretórios grandes usam uma árvore B+ para listar os arquivos)



# Implementação de Diretórios

- Onde armazenar os atributos relacionados a um arquivo?
  - No i-node:
    - Nesse caso, a entrada no diretório é menor, armazenando somente o nome de arquivo e o número do i-node
  - UNIX e Linux



# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”

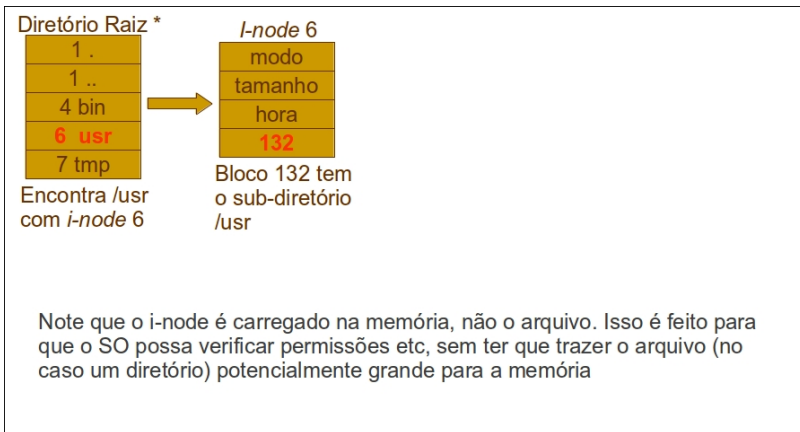
Diretório Raiz \*

1 .
1 ..
4 bin
<b>6 usr</b>
7 tmp

Encontra /usr  
com *i-node* 6

# I-Nodes

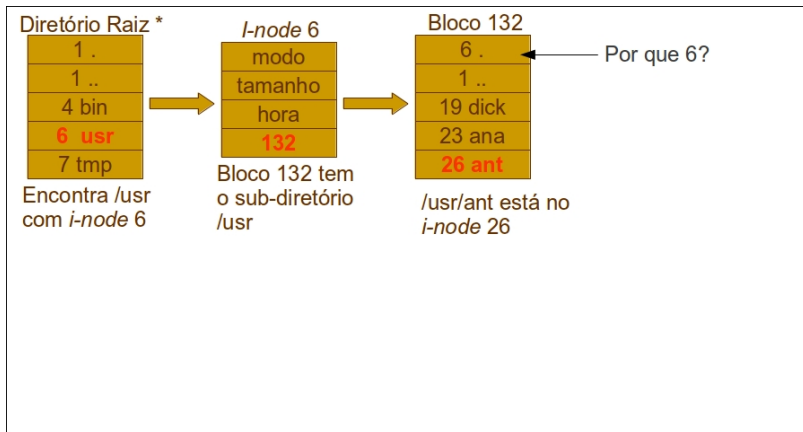
- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”





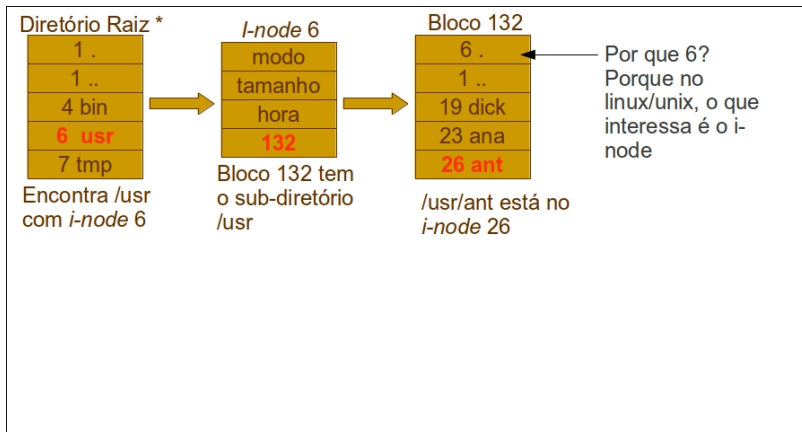
# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”



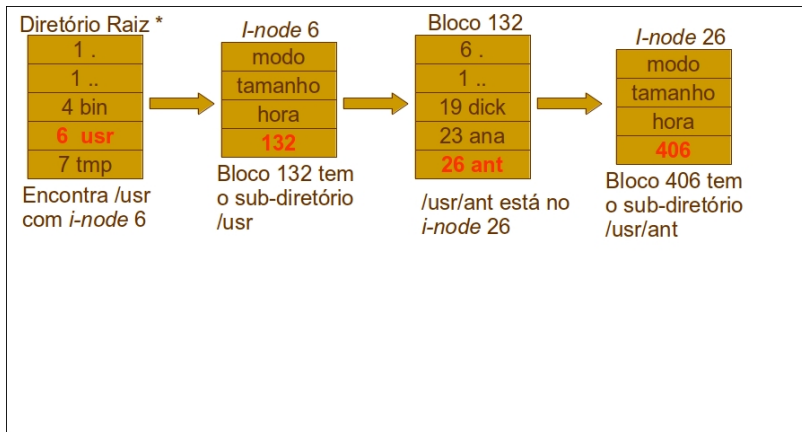
# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”



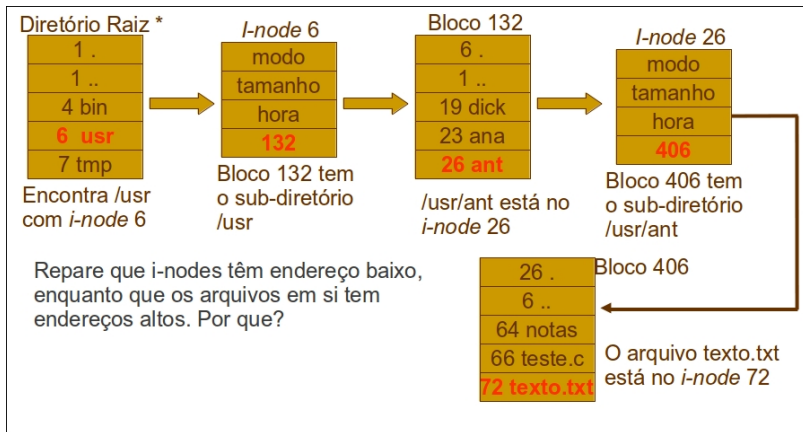
# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”



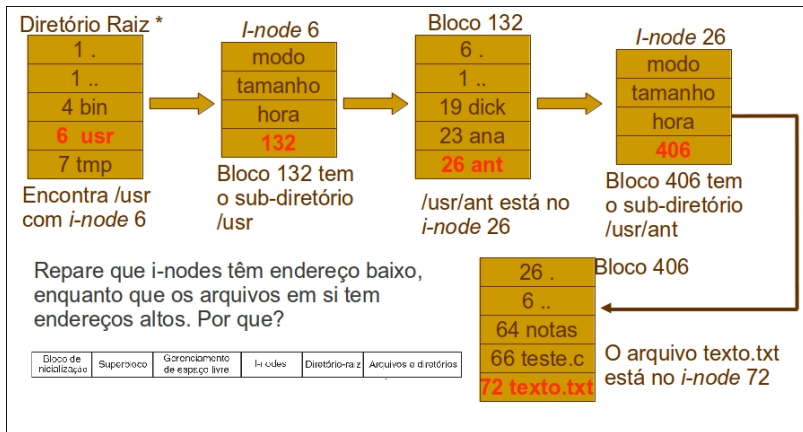
# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”



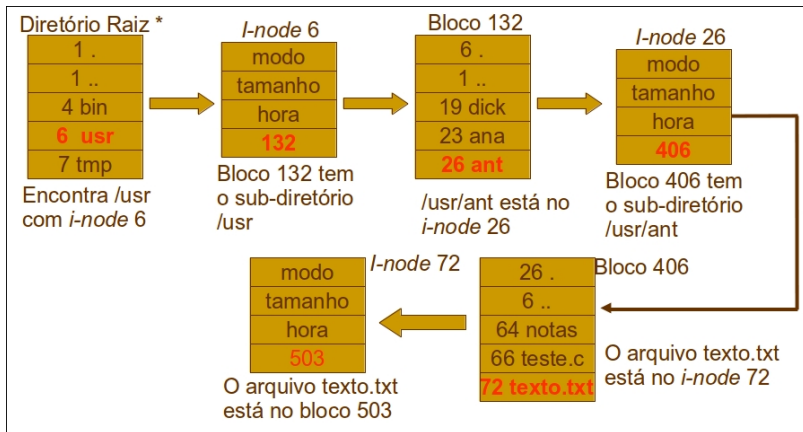
# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”



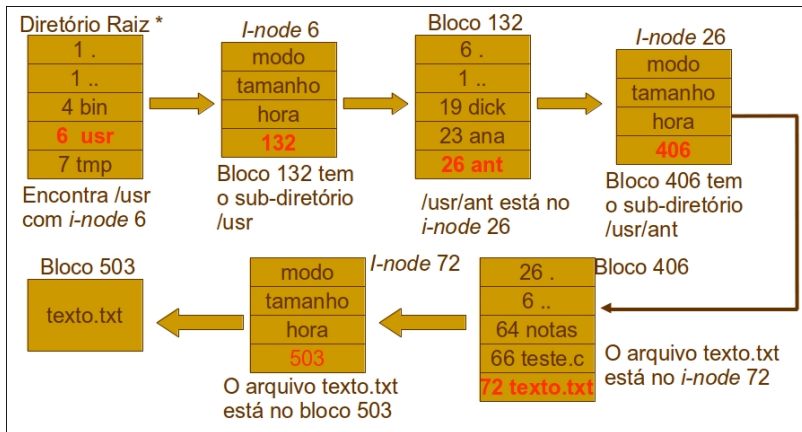
# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”



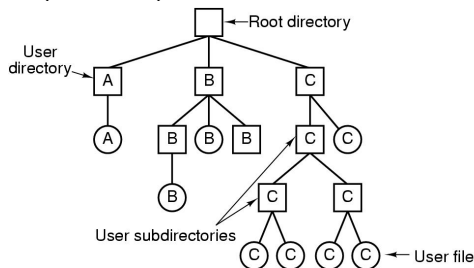
# I-Nodes

- Exemplo prático: ler o arquivo “/usr/ant/texto.txt”



# Arquivos Compartilhados

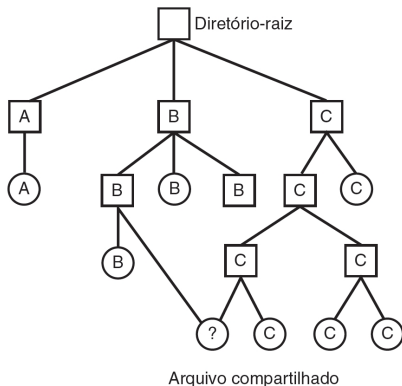
- Usuários frequentemente precisam acessar os mesmos arquivos
  - Muitas vezes, é conveniente que arquivos apareçam em diferentes diretórios (via links)
- Normalmente, o sistema de arquivos é implementado como uma árvore





# Arquivos Compartilhados

- Mas quando se tem arquivos compartilhados...
- O sistema de arquivos é implementado como um grafo acíclico direcionado
- Links são criados
  - Conexão entre um diretório (B) e o arquivo compartilhado presente em outro (C)

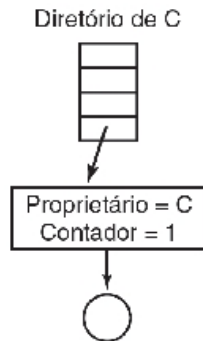


# Arquivos Compartilhados

- Links podem ser feitos a qualquer tipo de arquivo (regulares, diretórios etc)
- Compartilhar arquivos é conveniente. No entanto, alguns problemas são introduzidos:
  - Se os diretórios contiverem diretamente os endereços de disco, então deverá ser feita uma cópia dos endereços no diretório B, quando do link
    - Se B ou C adicionar blocos ao arquivo (append), os novos blocos serão relacionados somente no diretório do usuário que está fazendo a adição
    - Mudanças não serão visíveis ao outro usuário, comprometendo o propósito do compartilhamento

# Arquivos Compartilhados

- Primeira solução:
  - Os blocos de disco não estão listados nos diretórios, mas em uma estrutura de dados associada ao próprio arquivo (i-node ou linha da MFT)
  - Assim, os diretórios apontam para essa estrutura (UNIX)
  - A estrutura contém informação sobre o arquivo
    - Dono
    - Localização no disco
    - Número de ligações
    - etc

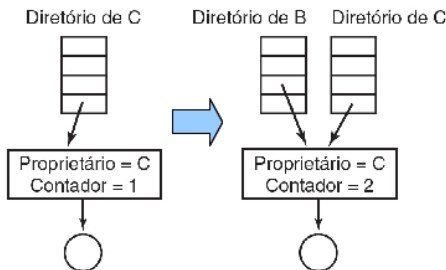


# Arquivos Compartilhados

- Primeira solução:

- Ligação Estrita (hard link)

- No momento em que B faz a ligação com o arquivo compartilhado, o i-node está registrando que o dono do arquivo é C
    - Criar o link não muda seu dono
    - O contador de links no i-node é incrementado
    - O sistema sabe quantas entradas de diretório apontam para o arquivo naquele momento

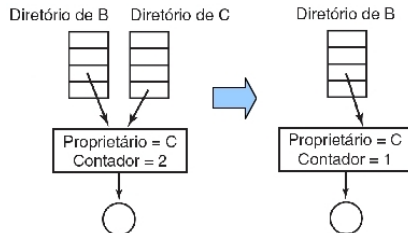


# Arquivos Compartilhados

- Primeira solução:

- Ligação Estrita (hard link)

- Quando C remove o arquivo, há um problema: Se o arquivo e seu i-node forem removidos, B terá uma entrada apontando para um i-node inválido
    - Solução: Remover a entrada de C, mas deixando o i-node intacto, decrementando seu contador de links. Se o contador for 0, aí sim remove o arquivo e limpa o i-node
    - Problema: B é o único usuário com uma entrada em diretório para um arquivo cujo dono é C – Se o sistema tiver quotas, ocupará espaço de C



# Arquivos Compartilhados

- Segunda solução: Ligação Simbólica
  - B se liga ao arquivo de C criando um arquivo do tipo link e inserindo esse arquivo em seu diretório (B)
    - O arquivo link contém apenas o caminho absoluto do arquivo ao qual ele está ligado
    - Remover o link não afeta o arquivo
    - Quando B lê do arquivo, o SO vê que é um link simbólico, olha o nome do arquivo no link e lê este arquivo
  - Não incrementa contador nenhum
  - Também presente em Windows (Atalhos)

# Arquivos Compartilhados

- Segunda solução: Ligação Simbólica
  - Somente o dono do arquivo tem o ponteiro para o i-node
    - Resolve o problema da quota
  - Vantagem:
    - O arquivo alvo pode estar, por exemplo, em outro disco
    - Podem se ligar a qualquer máquina no mundo. Basta dar o endereço de rede da máquina

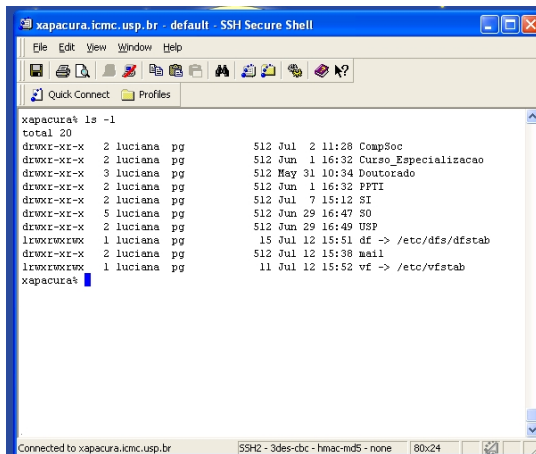
# Arquivos Compartilhados

- Ligação Simbólica: Problemas
  - Remover o arquivo original, sem remover o link, criará um link que não pode ser seguido
    - Tentativas subsequentes de acessá-lo falharão
  - Overhead:
    - O arquivo com o caminho deve ser lido, o caminho sintaticamente analisado e seguido, componente por componente, até o i-node final
    - Vários acessos adicionais ao disco
  - Deve haver um i-node extra para cada ligação simbólica
    - Além de um bloco de disco extra, para guardar o caminho



# Arquivos Compartilhados

- Ligação Simbólica



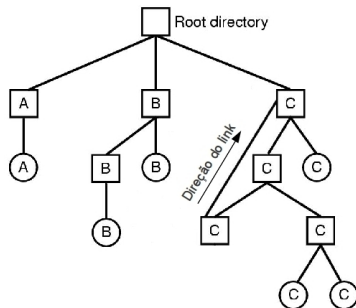
The screenshot shows an SSH terminal window titled "xapacura.icmc.usp.br - default - SSH Secure Shell". The terminal displays the output of the command "ls -l", which lists files and directories with their permissions, owner, group, size, date, and name. The output is as follows:

```
xapacura% ls -l
total 20
drwxr-xr-x  2 luciana  pg          512 Jul  2 11:28 CompSoc
drwxr-xr-x  2 luciana  pg          512 Jun  1 16:32 Curso_Especializacao
drwxr-xr-x  3 luciana  pg          512 May 31 10:34 Doutorado
drwxr-xr-x  2 luciana  pg          512 Jun  1 16:32 PPTI
drwxr-xr-x  2 luciana  pg          512 Jul  7 15:12 SI
drwxr-xr-x  5 luciana  pg          512 Jun 29 16:47 S0
drwxr-xr-x  2 luciana  pg          512 Jun 29 16:49 USP
lrwxrwxrwx  1 luciana  pg           15 Jul 12 15:51 df -> /etc/dfs/dfstab
drwxr-xr-x  2 luciana  pg          512 Jul 12 15:38 mail
lrwxrwxrwx  1 luciana  pg           11 Jul 12 15:52 vf -> /etc/vfstab
```

The terminal status bar at the bottom indicates the connection details: "Connected to xapacura.icmc.usp.br", "SSH2 - 3des-cbc - hmac-md5 - none", and "80x24".

# Arquivos Compartilhados

- Problema com links em geral: um arquivo pode ter 2 ou mais caminhos
  - Buscas poderão retornar o mesmo arquivo várias vezes
  - Programas devem também levar em conta a possível existência de ciclos, quando efetuam busca em diretório, seguindo links
  - Por esse motivo o linux não permite hard links a diretórios
  - Links simbólicos são permitidos a diretórios tanto no linux quanto no windows, mas são ignorados em buscas



# Journaling

- Considere a remoção de um arquivo no Unix
  - Passos (Windows é semelhante):
    - Remoção do arquivo de seu diretório
    - Liberação do i-node para o conjunto de i-nodes livres
    - Devolução dos blocos pertencentes ao arquivo para o conjunto de blocos livres no disco
- E se o sistema parar ao final da primeira etapa?
  - Teremos i-nodes com blocos não acessíveis, mas também não disponíveis para realocação
- E ao final da segunda etapa?
  - Os blocos serão perdidos

# Journaling

- Solução: Journaling
  - Técnica para criar uma certa robustez diante de falhas
- Mantém-se um registro (em um log – o journal) do que o sistema de arquivos irá fazer antes que ele efetivamente o faça
  - Se o sistema falhar antes de executar o trabalho, pode-se descobrir, após a reinicialização, o que deveria ter feito
  - Sistema de Arquivos Journaling
    - Usado no NTFS (Windows) e ext3 em diante (Linux)

# Journaling – Funcionamento

- Escreve uma entrada no log listando as 3 operações a serem concluídas
  - Remoção da entrada no diretório, liberação dos i-nodes e blocos
- Grava o log no disco
  - É lido de volta para a memória, para verificar sua integridade
- Só então as operações têm início
  - Quando uma é concluída, sua entrada do log é marcada
  - Entradas são excluídas de tempos em tempos ou quando o log estiver quase cheio, dependendo da política

# Journaling

- Se houver alguma parada, o sistema pode, após a recuperação, verificar o log para saber se há operação pendente
  - Se for o caso, pode executá-la
  - Isso pode ser repetido até que o arquivo seja removido corretamente, caso haja várias paradas
- Para que journaling funcione, as operações no log devem ser idempotentes
  - Devem poder ser repetidas sempre que necessário sem causarem nenhum dano

# Journaling

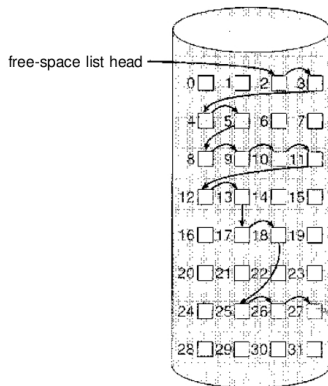
- Ex: Incluir novos blocos ao final da lista de blocos livres não é idempotente → os blocos podem já estar lá
  - Deve-se antes pesquisar se já não está lá e, em caso negativo, incluir → idempotente
- Sistemas de arquivo journaling devem organizar suas estruturas de dados e operações no log de forma que todos sejam idempotentes

# Controle de Blocos Livres

- Dois métodos:

- Lista ligada de blocos de disco

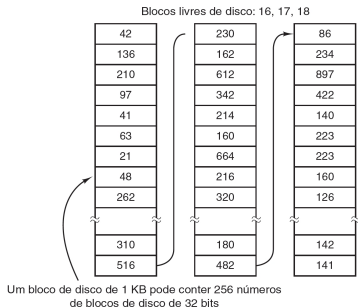
- Um ponteiro para o primeiro bloco livre é mantido em um local específico do disco e em cache (na memória)
- Este primeiro bloco contém o endereço do segundo, e assim por diante
- Problema: pouco eficiente, caso precisemos varrer a lista





# Controle de Blocos Livres

- Lista ligada de blocos de disco – alternativa
  - Armazenar, no primeiro bloco, o endereço de n-1 blocos livres
  - Guardar a última posição para o ponteiro do próximo bloco de endereços de blocos livres
    - Com blocos de 1KB e um número de 32 bits para cada bloco, cada bloco na lista manterá os números de 255 blocos livres
  - Endereços de um grande número de blocos podem ser encontrados rapidamente agora

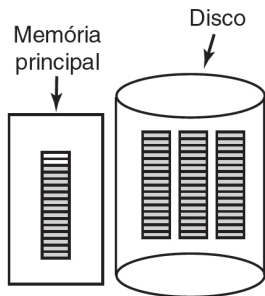


# Controle de Blocos Livres

- Lista ligada de blocos de disco
  - Somente um bloco de ponteiros precisa ser mantido na memória principal
  - Quando um arquivo é criado, os blocos necessários são tirados desse bloco
    - Quando o bloco se esgotar, um novo bloco de ponteiros é lido do disco
  - Quando um arquivo é apagado, os blocos são liberados e adicionados ao bloco de ponteiros na memória
    - Quando bloco está completo, esse bloco é escrito no disco

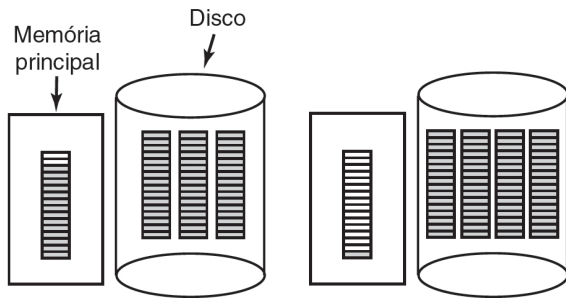
# Controle de Blocos Livres

- Lista ligada de blocos de disco: Problema
  - O bloco de ponteiros na memória tem espaço para mais 2 entradas



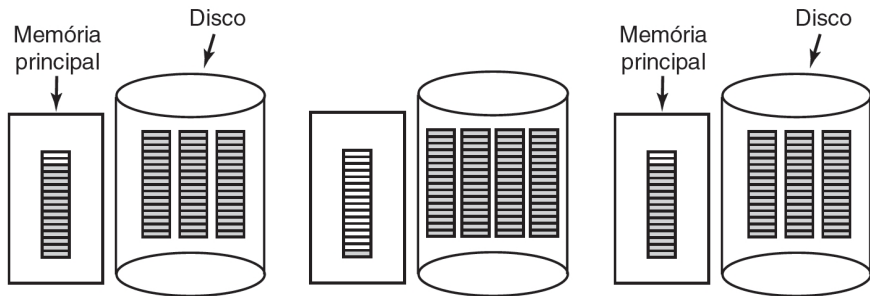
# Controle de Blocos Livres

- Lista ligada de blocos de disco: Problema
  - Um arquivo de 3 blocos é liberado → o bloco de ponteiros se enche, devendo ser escrito em disco



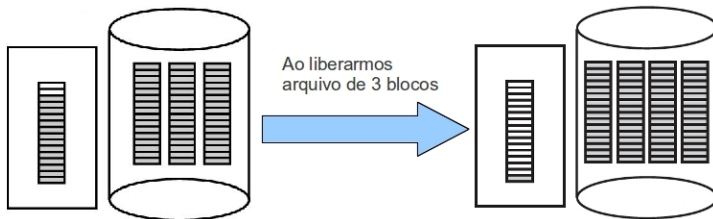
# Controle de Blocos Livres

- Lista ligada de blocos de disco: Problema
  - Um arquivo de 3 blocos é escrito → o bloco cheio deve ser lido novamente (Ex: arquivo temporário)



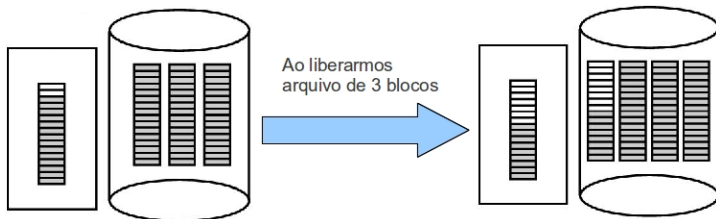
# Controle de Blocos Livres

- Lista ligada de blocos de disco: Problema
  - Quando o bloco de ponteiros está quase vazio, uma série de arquivos temporários pequenos podem causar uma grande quantia de E/S



# Controle de Blocos Livres

- Lista ligada de blocos de disco: Alternativa
  - Quebrar o bloco de ponteiros cheio
    - Se o bloco na memória se encher, é quebrado ao meio, e metade escrita em disco, restando metade na memória



# Controle de Blocos Livres

- Lista ligada de blocos de disco
  - Vantagens do uso da lista:
    - Requer menos espaço se existem poucos blocos livres (disco quase cheio)
    - Armazena apenas um bloco de ponteiros na memória
  - Desvantagens:
    - Requer mais espaço se existem muitos blocos livres (disco quase vazio)
    - Dificulta alocação contígua
    - Não ordenação



# Controle de Blocos Livres

- Dois métodos:
  - Mapa de bits
    - Depende do tamanho do disco
    - Um disco com  $n$  blocos requer um mapa com  $n$  bits, sendo um bit para cada bloco
    - $1 \rightarrow$  bloco livre
    - $0 \rightarrow$  bloco alocado
    - O mapa é mantido na memória principal

1001101101101100
0110110111110111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
0000111011010111
1011101101101111
1100100011101111
⋮
0111011101110111
1101111101110111

Cada bit representa um bloco (livre = 1, ocupado = 0)

# Controle de Blocos Livres

- Mapa de bits

- Vantagens:

- Requer menos espaço, com disco mais vazio (grande número de blocos livres)
    - Facilita alocação contígua

- Desvantagens:

- Torna-se lento quando o disco está quase cheio (e o número de blocos livres é pequeno)

1001101101101100
0110110111110111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
0000111011010111
1011101101101111
1100100011101111
~ ~
0111011101110111
1101111101110111

Cada bit representa um bloco (livre = 1, ocupado = 0)

# Confiabilidade: Consistência

- Dados no sistema de arquivos devem ser consistentes
  - Se o sistema cair antes que todos os blocos de arquivo modificados tenham sido guardados, pode ficar em estado inconsistente
  - Crítico: blocos de i-nodes, blocos de diretórios ou blocos contendo a lista de blocos livres (ou mapa de bits de blocos livres)
- Diferentes sistemas possuem diferentes programas utilitários para lidar com inconsistências:
  - UNIX: fsck
  - Windows: scandisk

# Confiabilidade: Consistência (fsck)

- Existem 2 tipos de verificação de consistência:
  - Blocos e arquivos (são feitas as 2)
  - Blocos
    - O programa constrói duas tabelas, cada qual com um contador (inicialmente com valor 0) para cada bloco
    - Os contadores da primeira tabela registram quantas vezes cada bloco está presente em um arquivo
    - Os contadores da segunda tabela registram quantas vezes cada bloco está presente na lista de blocos livres
    - O programa lê todos os i-nodes, ignorando a estrutura de arquivos, retornando todos os blocos de disco, começando do 0
    - A partir de um i-node, é possível construir uma lista com todos os números de blocos utilizados por um arquivo

# Verificação de Consistência (fsck)

- Blocos

- À medida em que os números de blocos são lidos, seu contador na primeira tabela é incrementado
- O programa então examina a lista de blocos livres (ou bitmap), buscando todos os blocos que não estão em uso
  - Cada ocorrência de um bloco na lista de livres incrementa seu contador na segunda tabela

- Se o sistema de arquivos estiver consistente, cada bloco terá 1 ou na primeira ou na segunda tabela

Número de bloco															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Blocos em uso															
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
Blocos livres															

# Fsck – E se problemas ocorrerem?

- Missing block (bloco 2):
  - Desperdício de espaço, reduzindo a capacidade do disco

Número do bloco

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Blocos em uso

0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Blocos livres

- Adicione-os à lista de livres

Número do bloco

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Blocos em uso

0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Blocos livres

# Fsck – E se problemas ocorrerem?

- Duplicatas na lista de livres (bloco 4):
  - Não podem ocorrer se a lista de livres for bitmap (apenas se for lista ligada)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Blocos em uso

0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Blocos livres

- Reconstrua a lista de livres

# Fsck – E se problemas ocorrerem?

- Duplicatas na lista de blocos em uso (bloco 5):
  - O mesmo bloco está presente em 2 ou mais arquivos

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

1	1	0	1	0	2	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 Blocos em uso

0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 Blocos livres

- Problemas
  - Se algum dos arquivos for removido, o bloco irá para a lista de livres → o mesmo bloco está tanto livre quanto em uso
  - Se ambos os arquivos forem removidos, o bloco será colocado 2 vezes na lista de livres

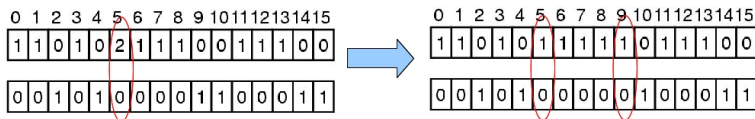


# Fsck – E se problemas ocorrerem?

- Duplicatas na lista de blocos em uso:

- Solução:

- Alocar um bloco livre
    - Copiar o conteúdo do bloco 5 nele
    - Inserir a cópia em um dos arquivos



- Embora mantenha o sistema consistente, certamente há erro
    - Relatar ao usuário, para que ele possa inspecionar o arquivo

# Confiabilidade: Consistência (fsck)

- Checagem do Sistema de Arquivos
  - Além do controle de blocos, o verificador também armazena em um contador o uso de cada arquivo
    - Tabela com um contador por arquivo
  - Começando da raiz, desce a árvore recursivamente, inspecionando cada diretório
  - Para cada arquivo no diretório, incrementa o contador correspondente
    - Devido a hard links, um arquivo pode aparecer em mais de um diretório → entra na contagem
    - Links simbólicos não são contados para o arquivo-alvo

# Confiabilidade: Consistência (fsck)

- Checagem do Sistema de Diretórios
  - A tabela é construída dinamicamente, à medida que corre a árvore
    - Será uma lista indexada pelo número do i-node
  - Ao final, a lista indicará em quantos diretórios cada arquivo aparece (contador de arquivos)
  - Esses valores são então comparados com a contagem de links presente no próprio i-node
    - Começa em 1 quando arquivo é criado
    - Incrementada toda vez que um hard link é feito
  - Em um sistema consistente, ambos valores devem bater

# Checagem do Sistema de Diretórios (fsck)

- Possíveis erros:
  - A contagem de links no i-node pode estar mais alta que o número de entradas nos diretórios
    - Mesmo que removamos de todos os diretórios, o i-node não é realmente removido
  - A contagem de links no i-node pode estar mais baixa
    - Arquivos podem ser removidos do disco, mesmo havendo ainda entradas para eles em algum diretório
  - Qualquer que seja o caso, corrija o contador no i-node
    - Sempre o que está registrado nos diretórios é o que vale

# Referências Adicionais

- <http://en.wikipedia.org/wiki/Inode>
- <http://www.cyberciti.biz/tips/understanding-unixlinux-filesystem-inodes.html>
- [http://cnx.org/contents/3c560b10-ed9c-4238-9a25-b50059a8ee67@1/I/O\\_devices\\_and\\_File\\_systems](http://cnx.org/contents/3c560b10-ed9c-4238-9a25-b50059a8ee67@1/I/O_devices_and_File_systems)
- <http://www.infowester.com/fat.php>
- <http://homepage.cs.uri.edu/courses/fall2004/hpr108b/FAT.htm>
- <http://pcsupport.about.com/od/termsf/g/fat.htm>

# Referências Adicionais

- <http://www.ibm.com/developerworks/library/l-journaling-filesystems/index.html>
- [http://en.wikipedia.org/wiki/Journaling\\_file\\_system](http://en.wikipedia.org/wiki/Journaling_file_system)