

SSC 140 - SISTEMAS OPERACIONAIS I

Turmas A e B

Aula 21 – Sistemas de Arquivos

Profa. Sarita Mazzini Bruschi

Slides de autoria de
Luciana A. F. Martimiano baseados no livro
Sistemas Operacionais Modernos de A. Tanenbaum

Implementando o Sistema de arquivos - Arquivos

- ❑ Armazenamento de arquivos → como os arquivos são alocados no disco;
- ❑ Diferentes técnicas são implementadas por diferentes Sistemas Operacionais;
 - Alocação contínua;
 - Alocação com lista encadeada;
 - Alocação com lista encadeada utilizando uma tabela na memória (FAT);
 - *I-Nodes*;

2

Implementando o Sistema de Arquivos - Diretórios

- ❑ Quando um arquivo é aberto, o Sistema Operacional utiliza o caminho fornecido pelo usuário para localizar o diretório de entrada;
- ❑ O diretório de entrada provê as informações necessárias para encontrar os blocos no disco nos quais o arquivo está armazenado:
 - Endereço do arquivo inteiro (alocação contínua);
 - Número do primeiro bloco do arquivo (alocação com listas encadeadas);
 - Número do *i-node*;
- ❑ O serviço de diretório é responsável por mapear o nome ASCII do arquivo na informação necessária para localizar os dados

3

Implementando o Sistema de Arquivos - Diretórios

- ❑ O serviço de diretório também é responsável por manter armazenados os atributos relacionados a um arquivo:
 - a) Entrada do Diretório: Diretório consiste de uma lista de entradas com tamanho fixo (uma para cada arquivo) contendo um nome de arquivo (tamanho fixo), uma estrutura de atributos de arquivos, e um ou mais endereços de disco; MS/DOS e Windows;

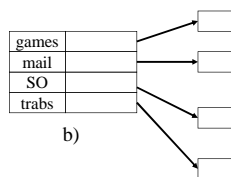
4

Implementando o Sistema de Arquivos - Diretórios

- b) *I-node*: nesse caso, o diretório de entrada é menor, armazenando somente o nome de arquivo e o número do *i-node* que contém os atributos; UNIX

games	atributos
mail	atributos
SO	atributos
trabs	atributos

a)



b)

Estrutura de dados contendo atributos (*i-nodes*)

5

Implementando o Sistema de Arquivos - Diretórios

- ❑ Tratamento de nomes de arquivos:
 - **Maneira mais simples:** limite de 255 caracteres reservados para cada nome de arquivo:
 - ❑ Toda entrada de diretório tem o mesmo tamanho;
 - ❑ Desvantagem: desperdício de espaço, pois poucos arquivos utilizam o total de 255 caracteres;
 - **Maneira mais eficiente:** tamanho do nome do arquivo é variável;

6

Implementando o Sistema de Arquivos - Diretórios

Tamanho do nome de arquivo variável

Tamanho da entrada do A1			
Atributos A1			
p	r	o	j
e	c	t	-
b	u	d	☒
Tamanho da entrada do A2			
Atributos A2			
p	e	r	s
o	n	n	e
l	☒		

- Cada nome do arquivo é preenchido de modo a ser composto por um número inteiro de palavras

- Quando não ocupar toda a palavra, preenche de modo a completar a palavra (parte sombreada);

Problema: se arquivo é removido, um espaço em branco é inserido e pode ser que o nome de outro arquivo não possua o mesmo tamanho

a)

7

Implementando o Sistema de Arquivos - Diretórios

Tamanho do nome de arquivo variável

Tamanho da entrada do A1			
Atributos A1			
p	r	o	j
e	c	t	-
b	u	d	☒
Tamanho da entrada do A2			
Atributos A2			
p	e	r	s
o	n	n	e
l	☒		

a)

Ponteiro p/ nome A1			
Atributos A1			
Ponteiro p/ nome A12			
Atributos A2			
p	r	o	j
e	c	t	-
b	u	d	☒
p	e	r	s
o	n	n	e
l	☒		

b)

8

Implementando o Sistema de Arquivos - Diretórios

Busca em diretório:

- Linear → lenta para diretórios muito grandes;
- Uma tabela *Hash* para cada diretório:
 - O nome do arquivo é submetido a uma função *hash* para selecionar uma entrada na tabela *hash*;
 - Cria-se uma lista encadeada para todas as entradas com o mesmo valor *hash*;
 - Vantagem: busca mais rápida;
 - Desvantagem: gerenciamento mais complexo;
- Cache de busca → ótima para poucas consultas de arquivos;

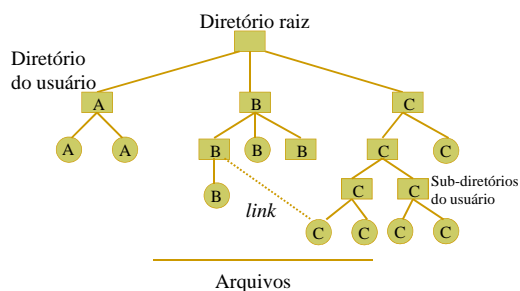
9

Implementando o Sistema de Arquivos – Arquivos Compartilhados

- Normalmente, o sistema de arquivos é implementado com uma árvore;
- Mas quando se tem arquivos compartilhados, o sistema de arquivos passa a ser um grafo acíclico direcionado (*directed acyclic graph – DAG*);
 - Links são criados;

10

Implementando o Sistema de Arquivos – Arquivos Compartilhados



11

Implementando o Sistema de Arquivos – Arquivos Compartilhados

- Compartilhar arquivos é sempre conveniente, no entanto, alguns problemas são introduzidos:
 - Se os diretórios tiverem **endereços de disco**, então deverá ser feita uma cópia dos endereços no diretório de B;
 - 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;

12

Implementando o Sistema de Arquivos – Arquivos Compartilhados

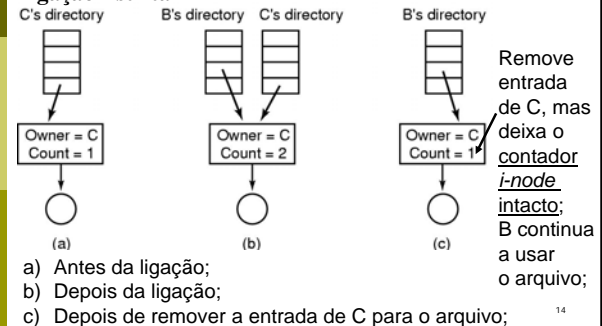
□ Soluções:

- **Primeira solução:** os **endereços de disco** não estão relacionados nos diretórios, mas em uma estrutura de dados (*i-node*) associada ao próprio arquivo. Assim, os diretórios apontam para essa estrutura; (UNIX)
 - **Ligação Estrita (*hard link*)**;
 - **Problema com essa solução:** o dono do arquivo que está sendo compartilhado apaga o arquivo;

13

Implementando o Sistema de Arquivos – Arquivos Compartilhados

Ligação Estrita



14

Implementando o Sistema de Arquivos – 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;
 - Somente o dono do arquivo tem o ponteiro para o *i-node*;
 - O arquivo **link** contém apenas o caminho do arquivo ao qual ele está ligado;
 - Assim, remoções não afetam o arquivo;
 - **Problema:**
 - Sobrecarga;
 - Geralmente um *i-node* extra para cada ligação simbólica;

15

Implementando o Sistema de Arquivos – 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;
 - Arquivo é espalhado por um número de blocos não necessariamente contínuos → blocos com tamanho fixo;
 - A maioria dos sistemas de arquivos utilizam essa estratégia;

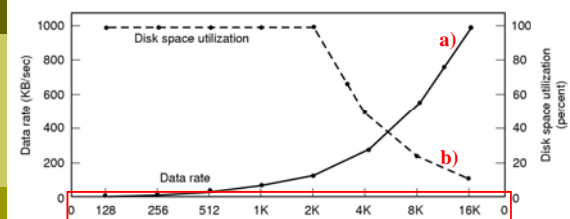
16

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco

- **Questão importante:** Qual é o tamanho ideal para um bloco?
 - Se for muito grande, ocorre desperdício de espaço;
 - Se for muito pequeno, um arquivo irá ocupar muitos blocos, tornando o acesso/busca lento;
- Assim, o tamanho do bloco tem uma grande influência na eficiência de utilização do espaço em disco e no acesso ao disco (desempenho);

17

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco



a) Taxa de Dados (curva contínua) X Tamanho do Bloco
b) Utilização do disco (curva tracejada) X Tamanho do Bloco

18

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco

- ❑ Conflito entre performance (desempenho) e utilização do disco → blocos pequenos contribuem para um baixo desempenho, mas são bons para o gerenciamento de espaço em disco;
- ❑ UNIX → 1Kb;
- ❑ MS-DOS → 512 bytes a 32 Kb (potências de 2);
 - Tamanho do bloco depende do tamanho do disco;
 - Máximo número de blocos = 2^{16} ;
- ❑ WinNT → 2Kb; WINXP → 4Kb;
- ❑ Linux → 1Kb, 2Kb, 4Kb;

19

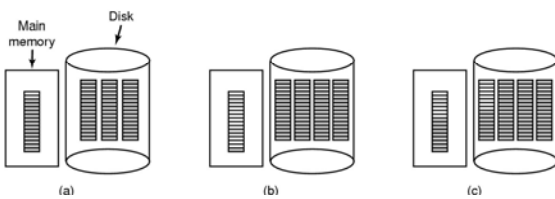
Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco

- ❑ Controle de blocos livres → dois métodos:
 - **Lista ligada de blocos livres**: 32 bits para endereçar cada bloco; mantida no disco;
 - ❑ Somente um bloco de ponteiros é mantido na memória principal → quando bloco está completo, esse bloco é escrito no disco;
 - ❑ **Vantagens**:
 - 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 contínua;
 - Não ordenação;

20

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco

Situação: três blocos são liberados



Blocos de ponteiros

- Entradas sombreadas representam ponteiros para blocos livres no disco

21

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco

- **Mapa de bits (*bitmap*)**: depende do tamanho do disco:
 - ❑ Um disco com n blocos, possui um mapa de bits com n bits, sendo um bit para cada bloco;
 - ❑ Mapa é mantido na memória principal;
 - ❑ **Vantagens**:
 - Requer menos espaço;
 - Facilita alocação contínua;
 - ❑ **Desvantagens**:
 - Torna-se lento quando o disco está quase cheio;

22

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco

Mapa de bits	
10011011	Blocos livres = 1 Blocos ocupados = 0 ou vice-versa;
01101101	
10101101	
01101101	
11101110	
...	
11011111	

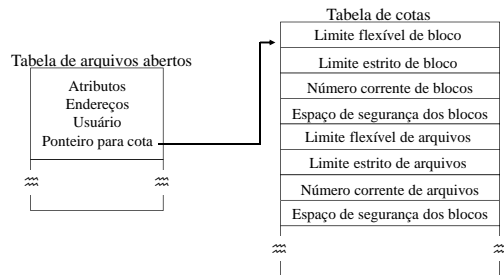
23

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco

- ❑ Controle de cotas do disco: feito para que um usuário não ocupe muito espaço do disco;
 - Idéia → administrador do sistema atribui para cada usuário uma cota máxima de espaço;
- ❑ Na memória principal:
 - Tabela de arquivos abertos com ponteiro para uma tabela que mantém registro de todas as cotas do usuário;

24

Implementando o Sistema de Arquivos – Gerenciamento de espaço em disco



25

Implementando o Sistema de Arquivos

Algumas características importantes:

- **Confiabilidade:**
 - Backups;
 - Consistência;
- **Desempenho:**
 - Caching;

26

Implementando o Sistema de Arquivos – Confiabilidade

- Danos causados ao sistema de arquivos podem ser desastrosos;
- Restaurar informações pode, e geralmente é, ser custoso, difícil e, em muitos casos, impossível;
- Sistemas de arquivos são projetados para proteger as informações de danos lógicos e não físicos;

27

Implementando o Sistema de Arquivos – Confiabilidade

Backups

- Cópia de um arquivo ou conjunto de arquivos mantidos por questão de segurança;
 - Midia mais utilizada → fitas magnéticas;
- Por que fazer *backups*?
 - Recuperar de desastres: problemas físicos com disco, desastres naturais;
 - Recuperar de “acidentes” do usuários que “acidentalmente” apagam seus arquivos;
 - Lixeira (diretório especial – *recycle bin*): arquivos não são realmente removidos;

28

Implementando o Sistema de Arquivos – Confiabilidade

- Backups podem ser feitos automaticamente (horários/dias programados) ou manualmente;
- Backups demoram e ocupam muito espaço → eficiência e conveniência;
- Questões:
 - O que deve ser copiado → nem tudo no sistema de arquivos precisa ser copiado;
 - Diretórios específicos;

29

Implementando o Sistema de Arquivos – Confiabilidade

- Não fazer *backups* de arquivos que não são modificados há um certo tempo;
 - Backups semanais/mensais seguidos de *backups* diários das modificações → *incremental dumps*;
 - Vantagem: minimizar tempo;
 - Desvantagem: recuperação é mais complicada;
- Comprimir os dados antes de copiá-los;
- Dificuldade em realizar *backup* com o sistema de arquivos ativo:
 - Deixar o sistema *off-line*: nem sempre possível;
 - Algoritmos para realizar *snapshots* no sistema: salvam estado atual do sistema e suas estruturas de dados;
- As fitas de *backup* devem ser deixadas em locais seguros;

30

Implementando o Sistema de Arquivos – Confiabilidade

□ Estratégias utilizadas para *backup*:

- **Física:** cópia se inicia no bloco 0 e pára somente no último bloco, independentemente se existem ou não arquivos nesses blocos;
 - Desvantagens:
 - Copiar blocos ainda não utilizados não é interessante;
 - Possibilidade de copiar blocos com defeitos;
 - Difícil restaurar diretórios/arquivos específicos;
 - Incapacidade de saltar diretórios específicos;
 - Não permite cópias incrementais;
 - Vantagens:
 - Simples e rápida;

31

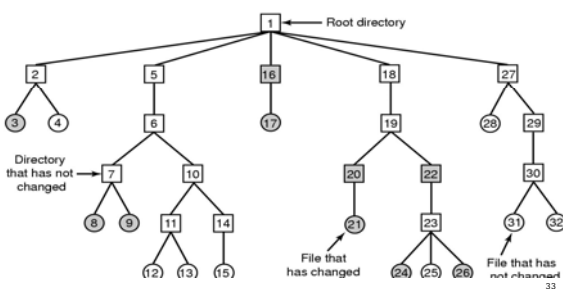
Implementando o Sistema de Arquivos – Confiabilidade

- **Lógica:** inicia-se em um diretório específico e recursivamente copia seus arquivos e diretórios; A idéia é copiar somente os arquivos (diretórios) que foram modificados;
 - Vantagem:
 - Facilita a recuperação de arquivos ou diretórios específicos;
 - Forma mais comum de *backup*;
 - Cuidados:
 - *Links* devem ser restaurados somente uma vez;
 - Como a lista de blocos livres não é copiada, ela deve ser reconstruída depois da restauração;

32

Implementando o Sistema de Arquivos – Confiabilidade

Algoritmo para Cópia Lógica



34

Implementando o Sistema de Arquivos – Confiabilidade

□ Algoritmo para cópia lógica:

- Fase 1 (a): marcar todos os arquivos modificados e os diretórios modificados ou não;
 - Diretórios marcados: 1, 2, 5, 6, 7, 10, 11, 14, 16, 18, 19, 20, 22, 23, 27, 29, 30;
 - Arquivos marcados: 3, 8, 9, 17, 21, 24, 26;
- Fase 2 (b): desmarcar diretórios que não tenham arquivos/sub-diretórios abaixo deles modificados;
 - Diretórios desmarcados: 10, 11, 14, 27, 29, 30;
- Fase 3 (c): varrer os *i-nodes* (em ordem numérica) e copiar diretórios marcados;
 - Diretórios copiados: 1, 2, 5, 6, 7, 16, 18, 19, 20, 22, 23;
- Fase 4 (d): arquivos marcados são copiados.
 - Arquivos copiados: 3, 8, 9, 21, 24, 26;

Implementando o Sistema de Arquivos – Confiabilidade

Algoritmo para Cópia Lógica



Mapa de bits indexado pelo número do *i-node*

35

Implementando o Sistema de Arquivos – Confiabilidade

- Consistência → dados no sistema de arquivos devem estar consistentes;
- Crítico: blocos de *i-nodes*, blocos de diretórios ou blocos contendo a lista de blocos livres/mapa de bits de blocos livres;
- Diferentes sistemas possuem diferentes programas utilitários para lidar com inconsistências:
 - UNIX: *fsck*;
 - Windows: *scandisk*;

36

Implementando o Sistema de Arquivos – Confiabilidade

FSCK (*file system checker*):

- **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;
- Lendo o *i-node*, o programa constrói uma lista com todos os blocos utilizados por um arquivo (incrementa contadores da 1ª tabela);
- Lendo a lista de bloco livres ou *bitmap*, o programa verifica quais blocos não estão sendo utilizados (incrementa contadores da 2ª tabela);
- Assim, se o sistema de arquivos estiver consistente, cada bloco terá apenas um bit 1 em uma das tabelas (a);

37

Implementando o Sistema de Arquivos – Confiabilidade

- a)
- | | | | | | | | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 15 |
| Blocos em uso | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| Blocos livres | | | | | | | | | | | | | | | | | |

Se problemas acontecerem, podemos ter as seguintes situações:

- b)
- | | | | | | | | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 0 | 2 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 15 |
| Blocos em uso | | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| Blocos livres | | | | | | | | | | | | | | | | | |

Bloco 2 perdido (*missing block*)

Solução: colocá-lo na lista de livres

- | | | | | | | | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 0 | 2 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 15 |
| Blocos em uso | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| Blocos livres | | | | | | | | | | | | | | | | | |

38

Implementando o Sistema de Arquivos – Confiabilidade

- a)
- | | | | | | | | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 15 |
| Blocos em uso | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| Blocos livres | | | | | | | | | | | | | | | | | |

Se problemas acontecerem, podemos ter as seguintes situações:

- c)
- | | | | | | | | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 15 |
| Blocos em uso | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| Blocos livres | | | | | | | | | | | | | | | | | |

Bloco 4 duplicado na lista de livres

Solução: reconstruir a lista

Essa situação só ocorre se existir uma lista encadeada de blocos livres ao invés de um mapa de bits.

39

Implementando o Sistema de Arquivos – Confiabilidade

- a)
- | | | | | | | | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 15 |
| Blocos em uso | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| Blocos livres | | | | | | | | | | | | | | | | | |

Se problemas acontecerem, podemos ter as seguintes situações:

- d)
- | | | | | | | | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
| 0 | 1 | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 15 |
| Blocos em uso | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| Blocos livres | | | | | | | | | | | | | | | | | |

Bloco 5 duplicado na lista de "em uso" (dois arquivos)

Problemas:

- Se um arquivo for removido, o bloco vai estar nas duas listas;
- Se ambos forem removidos, o bloco vai estar na lista de livres duas vezes;

Solução: alocar um bloco livre, copiar para esse bloco o conteúdo do bloco 5 e atribuir esse bloco a um dos arquivos, avisando o administrador/usuário do problema;

40

Implementando o Sistema de Arquivos – Confiabilidade

FSCK (*file system checker*)

- **Arquivos:** Além do controle de blocos, o verificador também armazena em um contador o uso de um arquivo → tabela de contadores por arquivos:
 - *Links* simbólicos não entram na contagem;
 - *Links* estritos (*hard link*) entram na contagem (arquivo pode aparecer em dois ou mais diretórios);
 - Cria uma lista indexada pelo número do *i-node* indicando em quantos diretórios cada arquivo aparece (contador de arquivos);
 - Compara esses valores com a contagem de ligações existentes (começa em 1 quando arquivo é criado);

41

Implementando o Sistema de Arquivos – Confiabilidade

FSCK (*file system checker*)

Arquivos:

- Se o sistema estiver consistente, os contadores devem ser iguais;
- Se a contagem de ligações no *i-node* for maior que o valor contado (contador de arquivo):
 - Problema: *i-node* não será removido quando o(s) arquivo(s) for(em) apagado(s)
- Se for menor:
 - Problema: quando chegar em zero, o sistema marca o *i-node* como não usado e libera os blocos, mas ainda tem arquivo apontando para aquele *i-node*
- Solução para ambos: atribuir o valor do contador de arquivos à contagem de ligações do *i-node*;

42