

# *Fundamentos de Sistemas de Operação*

*Unix*   *Windows NT*   *Netware*   *MacOS*   *DOS/VIS*   *Vax/VMS*  
*Linux*   *Solaris*   *HP/UX*   *AIX*   *UX*   *Mach*  
*Chorus*

*Sistemas de Ficheiros:*  
*Recuperação de falhas*

# *Alguns cenários...*

- **Cenário 1:** Todos ☺ os processos a ler ficheiros
  - Aqui, assume-se que *nenhum processo* está a escrever em ficheiros.
- **Cenário 2:** Alguns processos a escrever em ficheiros
  - 2a) Os processos escrevem, mas a dimensão dos ficheiros não se altera.
  - 2b) A dimensão de alguns ficheiros cresce.
- **Cenário 3:** Um cenário real!
  - Há ficheiros a serem lidos e outros lidos/escritos; há-os a serem criados e apagados...

# *Anomalias: crashes e falhas de energia (1)*

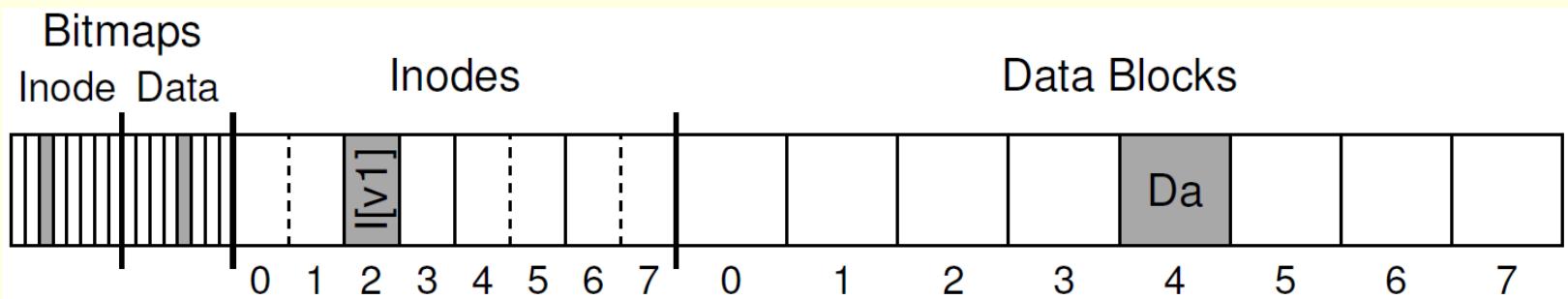
- Que se pode esperar num SF, como resultado das anomalias, nos diferentes cenários?
  - Passo 1: Que estruturas de dados estão a ser manipuladas?
  - Passo 2: De que forma estão a ser manipuladas?
  - Passo 3: Quais os efeitos da perda de informação, i.e., dessas estruturas não reflectirem o estado existente em memória na altura da anomalia?
    - a) Sobre o funcionamento do SF
    - b) Sobre os dados dos utilizadores
- Podemos fazer alguma coisa para recuperar o SF?

# Anomalias: crashes e falhas de energia (2)

- Como é que se sabe que houve um crash ou falta de energia (abrev. CFE)?
  - Antes de um volume (“disco”) poder ser acedido o SF tem de ser montado. Essa operação carrega informações para as estruturas *in-core*, e deixa uma flag (força a sua gravação imediata) no superbloco que indica “disco em uso”.
  - Essa flag é “removida” quando o disco é desmontado, o que pode acontecer por comando do administrador de sistema, ou no shutdown do SO.
  - Se, ao montar um volume, o SO encontra a flag no estado que indica “disco em uso”, isso significa que houve um CFE.

# Considerações preliminares

- As estruturas, e as operações que as manipulam...
  - Ainda que as estruturas em memoria (a.k.a. *in-core*) possam ser (*muito até*) diferentes das estruturas no disco, derivam destas...
  - Portanto, “cada caso é um caso” ☺ e a recuperação do estado do SF é fortemente dependente do SF particular, pelo que...
- No resto da discussão tomaremos como exemplo um SF simples, do tipo do ext2 (mas ainda mais simples: VSFS):



# *VFSF: estruturas de dados manipuladas (1)*

- As estruturas, e para que servem (relembrando)
  - **Bitmaps**: gerem o “espaço” livre/ocupado
  - **inodes**: mantêm informação sobre os ficheiros (incluindo directorias, que são ficheiros “especiais”)
  - **Blocos de dados**: armazenam o conteúdo dos ficheiros
  - **Directorias**: são uma estrutura absolutamente fundamental, pois mapeiam um nome num inode
- Como/quando são usadas (relembrando)
  - **Bitmaps**: quando se necessita de espaço (blocos) para um ficheiro crescer, ou quando é necessário um inode para um novo ficheiro. Ou quando se libertam blocos ou inodes. Se não há atribuição ou libertação, os bitmaps não são lidos nem modificados.

(continua)

# *VFSF: estruturas de dados manipuladas (2)*

- *Como/quando são usadas* (continuação)
  - **Directorias:** são ficheiros, pelo que comungam de todas as estruturas anteriores (bitmaps, blocos). Contudo, são ficheiros cujo conteúdo é crítico, pois estabelece a ligação entre os nomes e os inodes.

# VFSF: Cenário 1

- Consequências de um CFE (*crash ou falha de energia*)
  - Se todas as operações são leituras, um CFE não provoca danos de maior. A única informação que pode ficar desactualizada são os timestamps com a informações das “datas” do últimos acessos.
- Nota: este cenário, muito simplista,
  - Geralmente não se verifica, pois é comum o próprio SO escrever informações de log, etc., em ficheiros para análise de problemas, potenciais quebras de segurança, etc.

# VFS: Cenário 2a

## □ Consequências de um CFE

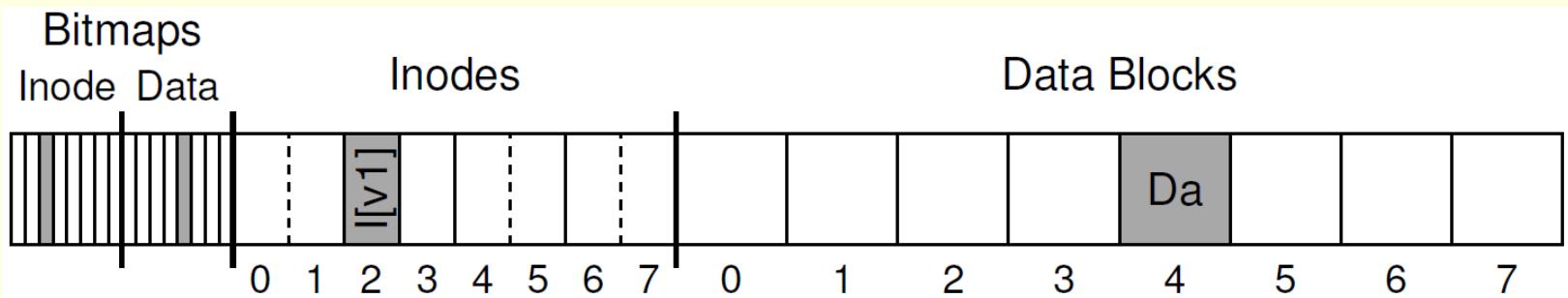
- Se as operações de escrita não alteram a dimensão dos ficheiros, apenas o *inode* é actualizado – timestamps.
- Note-se que, se as escritas fizerem aumentar a dimensão dos ficheiros, mas não a ponto de haver necessidade de atribuição de novos blocos, apenas a dimensão (em bytes) do ficheiro tem de ser actualizada no *inode*. Neste caso, na sequência de um CFE pode ficar-se com o valor antigo da dimensão do ficheiro...

## □ E os dados em cache aquando do CFE?

- Podem perder-se, mas a estrutura do SF continua íntegra...

# VFSF: Cenário 2b (1)

- Consideremos o estado *inicial* do SF:



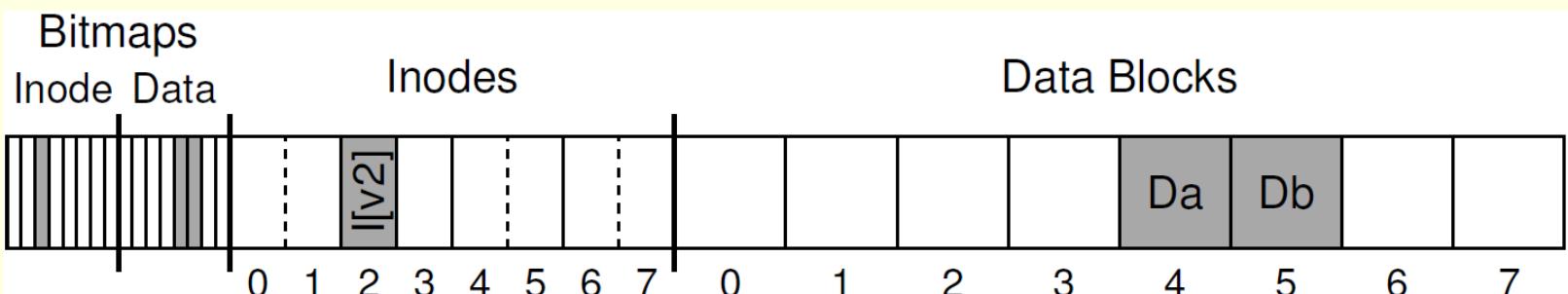
Existe um ficheiro, que ocupa o inode 2, e gasta um data block, informação que está reflectida nos dois bitmaps. O inode tem:

```
dono: remzi  
permissões: rw  
comprimento: 500  
ptr1: 4  
ptr2: -
```

(no livro o comprimento é em blocos; preferimos bytes)

# VFSF: Cenário 2b (2)

- *O(s) write(s) fazem o ficheiro crescer para além de 512 bytes*



*Foi necessário arranjar um bloco mais, e encontrou-se o 5 disponível, marcando-se no bitmap como ocupado. O inode passou a ter:*

```
dono: remzi
permissões: rw
comprimento: 840
ptr1: 4
ptr2: 5
```

# VFSF: Cenário 2b (3)

## □ Consequências de um CFE

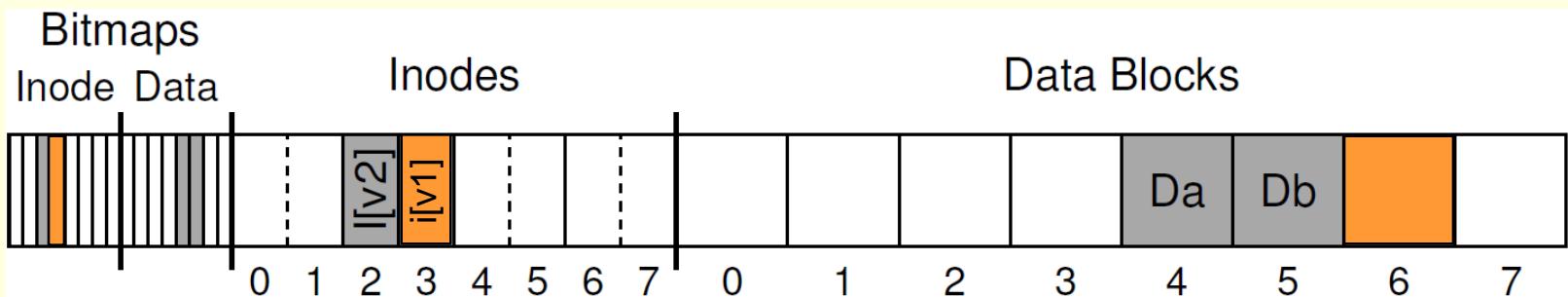
- Há agora 2 estruturas distintas a actualizar: o *bitmap* de dados e o *inode* (*ptr2*). Duas inconsistências podem surgir:
  - Consegue-se actualizar o *bitmap*, mas não o *inode*: consequência - um bloco ocupado, potencialmente com dados úteis, mas que não se sabe a que ficheiro pertence.
  - Consegue-se actualizar o *inode*, mas não o *bitmap*: consequência - um bloco, potencialmente com dados úteis, marcado como livre, mas referenciado num *inode*.

## □ E os dados em cache aquando do CFE?

- Podem perder-se...

# *VFSF: Cenário 3 (1)*

- Existe uma directória, descrita pelo inode 3 e cujas entradas (descrevendo ficheiros nessa directória) estão no data block 6



# VFSF: Cenário 3

## □ Consequências de um CFE

- *Todas as estruturas podem agora ter de ser modificadas, e várias inconsistências podem surgir:*
  - Criou-se um ficheiro na directória e escreveram-se uns bytes. Houve um CFE e os bitmaps e inode do novo ficheiro estão correctos, mas falhou a actualização da entrada na directória – não há nome para o ficheiro...
  - Apagou-se um ficheiro na directória. Consegiu-se marcar os blocos livres no bitmap, mas...
  - Todos os casos 2a e 2b...

## □ E os dados em cache aquando do CFE?

- *Podem perder-se...*

# *fsck: file system checker* (1)

- Aplicação que corre antes do volume ser montado
  - Tenta (é raro não conseguir) garantir que o estado final dos metadados é consistente. **Não recupera conteúdos de ficheiros do utilizador ou directorias**, etc.
- Corre em passos; verifica/recupera a consistência
  1. do superbloco
  2. dos bitmaps
  3. da informação nos inodes
  4. apontadores duplicados ou que apontam para sítios inválidos
  5. das directorias

# *fsck: file system checker* (2)

## □ Sumário das operações

### 1. Consistência do superbloco

1. Verificações simples de consistência: o “magic number” está correcto? A dimensão do SF é consistente com a dimensão do volume (disco)? Se há inconsistências graves, o fsck vai buscar uma cópia do superbloco ou sugere ao administrador que o faça.

### 2. Consistência dos bitmaps

1. Percorre a tabela de inodes, e para cada um constroi a lista de data blocks pertencentes a esse inode. Verifica se o bitmap de dados tem esses blocos marcados como ocupados – senão, acredita na informação do inode e corrige o bitmap de dados
2. Para cada inode, vê se o bitmap o tem marcado como ocupado – senão, acredita no inode...

# *fsck: file system checker* (3)

(continuação)

## 3. Consistência da informação no inode

1. Verificações simples de consistência: o “tipo de ficheiro” está correcto, i.e., refere um ficheiro regular, directória, pipe (FIFO), etc? Se há inconsistências graves, o fsck apaga o inode e liberta os blocos nos bitmaps.
2. Verifica os “link counts”

## 4. Consistência dos apontadores

1. Se um bloco de dados é apontado por 2 ou mais inodes...
2. Se um apontador de blocos aponta para um bloco que não existe (o volume é mais pequeno que o nº do bloco) ou para uma zona inválida (superbloco, bitmaps, inodes)...

# *fsck: file system checker* (3)

(continuação)

## 5. *Consistência da informação nas directorias*

1. Os *inodes* apontados pelos *nomes* são válidos?
2. Há *inodes* sem *nome*? Se sim, são colocados na *lost+found*

## □ *Inconvenientes dos FS checkers*

- *Lentidão... para volumes grandes contendo FSs com muitos milhares de ficheiros, pode demorar horas*
- *Algumas soluções são muito “radicais” – apagar integralmente um ficheiro ou, pior, uma directória...*