Instituto de Informática - UFRGS

Sistemas Operacionais

Sistema de arquivos

Montagem

Virtual File System

Sistemas de arquivos jornalizados

Aula 28

Introdução

- Um sistema computacional pode ter vários sistemas de arquivos
  - e.g.: NTFS, FAT32, ext3, ReiserFS, ISO 9660 (cdrom), Universal Disk Format (dvd), etc
- Cada sistema de arquivos é um disco lógico (partição)
- Arquivos só podem ser acessados depois que o sistema de arquivo ser montado
  - Montagem: combina vários sistemas de arquivos em um único espaço de nomes
- Sistema de arquivos pode estar em outro dispositivo ou em outra máquina (montagem remota)
  - e.g.: Network File System (NFS)
    - Conceito de exportação e de importação

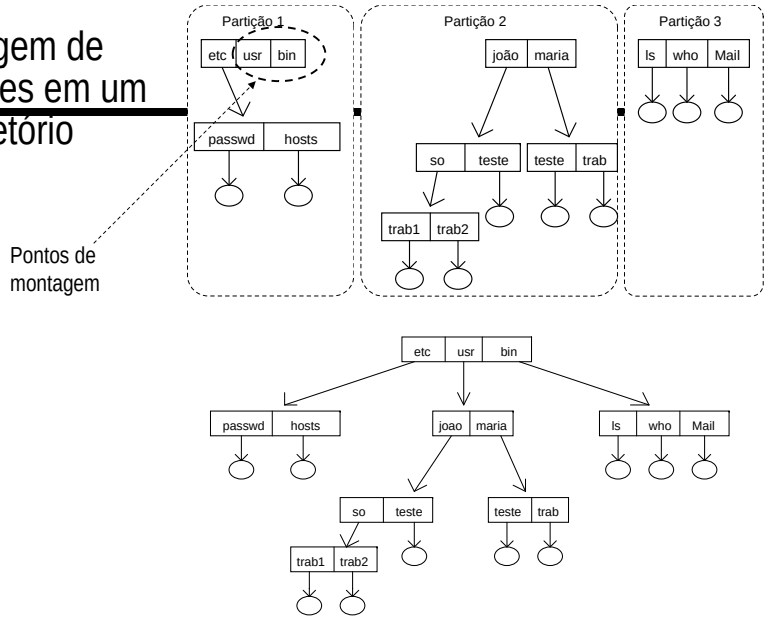
Instituto de Informática - UFRGS
A. Carissimi -

O conceito de montagem

- Montar um sistema de arquivo significa integrá-lo a uma hierarquia já existente de um outro sistema de arquivos
- Montagem é composta por:
  - Nome do dispositivo
  - Ponto de montagem
- Semânticas possíveis:
  - Ponto de montagem ser um diretório vazio ou não
  - Mesmo sistema de arquivos ser montado mais de uma vez
- Operações *mount* e *unmount*

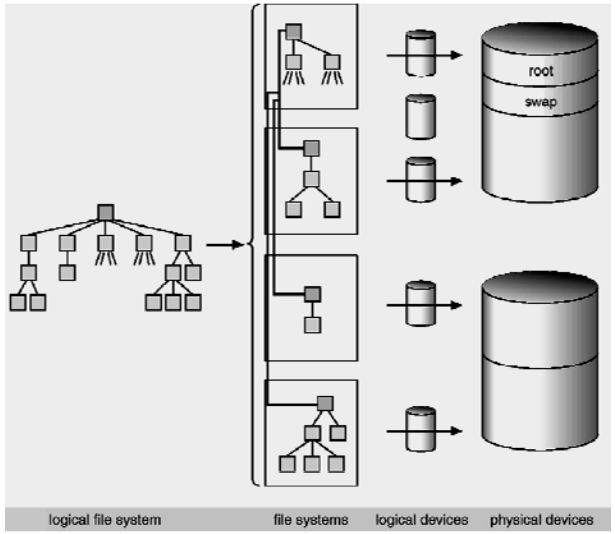
Instituto de Informática - UFRGS
A. Carissimi -

Montagem de partições em um subdiretório



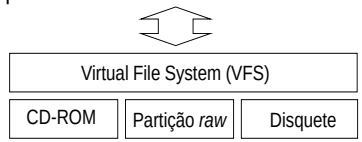
Instituto de Informática - UFRGS
A. Carissimi -

Exemplo de montagem

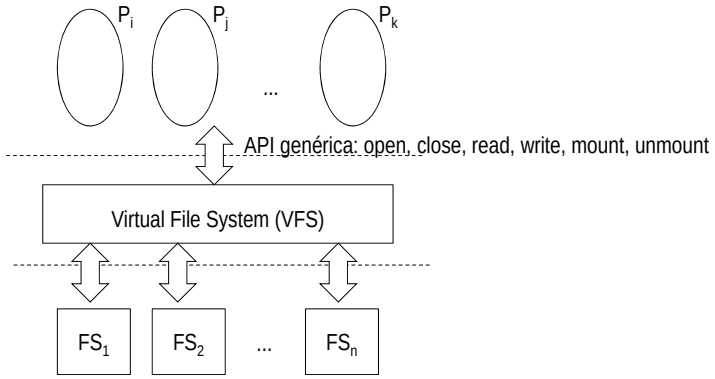


Suporte a múltiplos sistemas de arquivos

- Fazer com que o sistema operacional suporte diversos sistemas de arquivos diferentes simultaneamente
  - Cada partição possui um sistema de arquivos auto-contido
- Solução inspirada na gerência de periféricos
  - Parte independente do dispositivo
    - Serviços idênticos independente do tipo de sistema de arquivos
  - Parte dependente do dispositivo
    - Interface padrão



Virtual File System



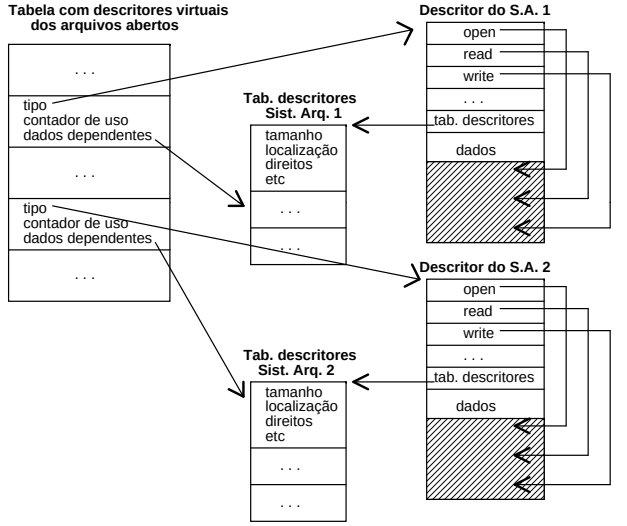
Princípio de implementação

- Baseado em técnicas de orientação a objetos
  - Estrutura de dados e procedimentos são usados para isolar a funcionalidade básica de sua implementação
- Três camadas
  - API: interface entre os processos de usuário e o sistema de arquivos
  - VFS
    - Separação entre as operações genéricas da implementação
    - Identificador único (v-node)
  - Implementação do sistema de arquivos específicos

Implementação de múltiplos sistemas de arquivos

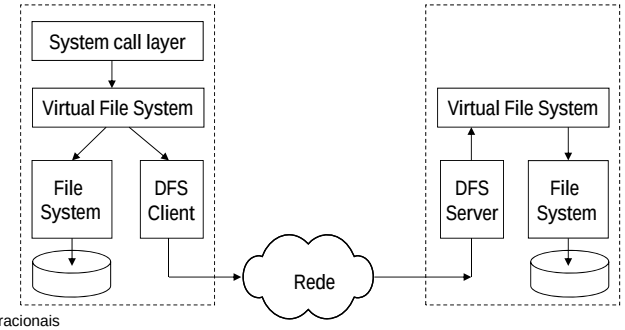
- Tabela com descritores virtuais de arquivos abertos
  - Parte independente do sistema de arquivos
  - Uma entrada ocupada para cada arquivo aberto (descritor virtual)
- Descritor virtual
  - Informações comuns a todo sistema de arquivo (proteção, nro de acessos, ...)
  - Apontador para uma estrutura “Tipo do sistema de arquivos”
  - Apontador para o descritor do sistema de arquivos real
    - Lista de ponteiros para rotinas que implementam o código necessário a execução de uma dada chamada de sistema (*read, write, close*,...)
    - Informações sobre a gerência desse sistema de arquivos (blocos livres, ocupados, estrutura de diretórios, ...)

Múltiplos sistemas de arquivos: estrutura de dados



Implementação de sistemas de arquivos distribuídos

- Coleção de clientes e de servidores
  - Servidores: armazenamento do sistema de arquivos (recursos)
  - Clientes: utilização de recursos remotos
- Arquitetura geral: (válida para o Network File System – NFS)



Confiabilidade em sistemas de arquivos

- Modificações no sistema de arquivos são mantidas em memória (cache de disco) por questões de desempenho
  - Blocos de dados e estruturas de controle
- Em caso de panes ou desligamento inadequado da máquina o sistema de arquivos pode ficar inconsistente
  - Necessário reconstruir a consistência
  - Tempo depende do número de arquivos e de diretórios
    - Discos atuais pode chegar a horas
- Sistemas de arquivos jornalizados visam otimizar a verificação de consistência

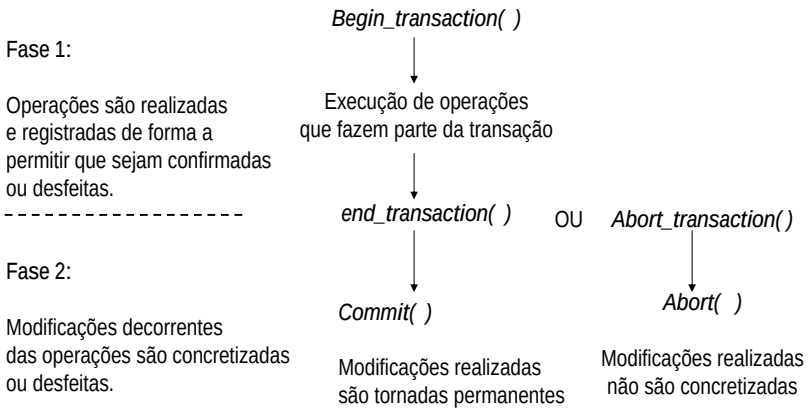
Confiabilidade em sistemas de arquivos

- Três princípios básicos
- Preservação
  - Dados estáveis no disco não podem ser afetados por uma pane
- Predição
  - Comportamento de recuperação após a pane deve apresentar resultados previsíveis
- Atomicidade
  - Uma operação é *completamente realizada* ou é considerada como nunca iniciada

O conceito de transação

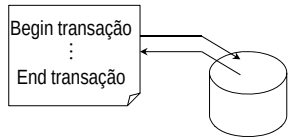
- Analogia: “um negócio é considerado feito somente após o acordo entre as partes e a assinatura do contrato”
- Devem oferecer a propriedade ACID
  - Atomicidade: ser indivisível sob o ponto de vista externo
  - Consistência: não violar invariantes* do sistema
  - Isolamento: transações concorrentes não devem interferir uma nas outras
  - Durabilidade: uma vez concluída, os efeitos são permanentes
- Primitivas básicas: *begin_transaction, end_transaction*, *commit* e *abort*
- Procedimento de *Roll-back*
  - Uma transação não concluída deve manter inalterado o estado do sistema

Fases de uma transação



Sistemas de arquivos jornalizados

- Emprega tecnologia proveniente de sistemas de banco de dados
- Mecanismo de base é realizar as atualizações no sistema de arquivos através do emprego de transações
  - As operações (transações) são mantidas em um jornal (arquivo *log*)
  - Efetua um rastreamento das modificações feitas no sistema de arquivos
- Em caso de pane utiliza as informações do jornal para deixar o sistema em um estado consistente

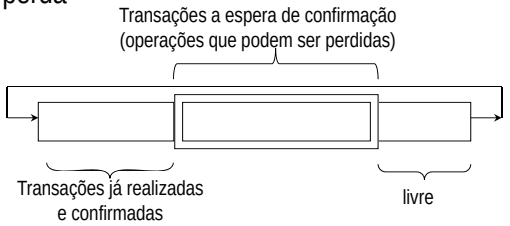


Implementação de ação atômica

1. *Execução da ação atômica A_i*
   a. Na execução de **begin atomic action** criar uma lista de intenções e *um flag commit commit flag* = (A_i, "not commited")
      Lista de intenções = "vazia"
   b. Para cada atualização feita por uma subação, adicionar um par (*d, v*) na lista de intenções, onde *d* é nro de um bloco e *v* é novo valor para este bloco
   c. Na execução de **end atomic action**, colocar o valor da ação A_i para *commit*, ir passo 2.
2. *Processamento do commit*
   1. Para cada par (*d,v*) na lista de intenções, escrever *v* no bloco do disco *d*
   2. Apagar o *flag commit* e a lista de intenções
3. *Na recuperação após uma falha*: se o flag commit da ação A_i existir
   1. Se o valor do flag é "not commited": apagar o *flag commit* e a lista de intenções e reiniciar a ação Ai
   2. Realizar o passo 2 se o valor do flag é *commit*

Arquivo de jornal

- Composto por um registro
  - Identificador da transação, identificador do arquivo/bloco, valores novo/antigo
- Possui um tamanho fixo
  - Esquema de "*rotate*" apagando informações mais antigas
- Otimizado para escrita
- "janela" de perda



Modos de jornalização

- Write behind
  - Garante a consistência apenas dos metadados
  - As operações de atualização de blocos de dados podem ser perdidas
- Ordered data
  - Garante a consistência dos metadados e dos dados, fazendo com que os metatados sejam confirmados somente após a escrita do bloco de dados
    - Pode acontecer de um bloco novo ter sido escrito no disco, mas ainda não ter sido atualizado como pertencente ao arquivo
- Full data
  - Garante a consistência dos metadados e dos dados por jornalização

Prós e contras da jornalização

- Tendência a fragmentar o disco
- Custo computacional
- Maior quantidade de operações de entrada e saída no disco
- Atualização do jornal no disco é sem uso de cache
- Paliativo:
  - Nem toda partição necessita ser jornalizada

Sistemas de arquivos jornalizados em moda

- XFS (SGI-irix/linux)
- JFS (IBM-aix/linux)
- ReiserFS (linux)
- ext3fs (linux)
- NTFS (windows NT, XP)

Estudo de caso: Ext3fs

- Objetivos:
  - Ser sistema de arquivos jornalizado
  - Ser compatível com o ext2 (usa mesmas estruturas de controle)
- Permite ser configurado para metadados e dados regulares
  - Problema é o desempenho
- Três tipos de jornalização:
  - Jornal: dados e metadados
  - Ordenado: metadados porém os atualiza após ter escrito os dados regulares
    - Método *default*
  - Writeback: metadados
- Arquivo de jornal é *.jornal* no raiz do sistema de arquivos ext3

Estudo de caso: ext3fs (*cont*.)

- Metadados:
  - Superblocos, descritor de grupos, i-nodes, blocos de indireção, bitmap (blocos e i-nodes)
- Dois passos:
  - Metadados modificados são copiados para o arquivo de jornal (*commit1*)
  - Blocos do jornal são escritos no sistema de arquivos (*commit 2*)
- Situações na recuperação (usa dois commits):
  - Falha acontece antes do commit 1
    - Modificações são ignoradas (volta-se ao estado anterior)
  - Falha acontece depois do commit 1 e antes do commit 2
    - Cópias no jornal estão atualizadas, as transfere para o sistema de arquivos

Estudo de caso: NTFS

- Transações são mantidas no metafile ($LogFile)
- Armazena apenas metadados
  - Garante a consistência do sistema de arquivos não de dados de usuário

Leituras complementares

- Silberchatz, A.; Galvin, P.B. *Operating Systems Concepts*. Adisson wesley, 5th edition, 1994.
  - Capítulos 13 e 17
- Silberchatz, A.; Galvin, P.B; Gagne, G. *Sistemas Operacionais: conceitos e aplicações*. Campus, 1ª edição, 2000.
  - Capítulos 17 e 18