Sistemas Operacionais II

- Sistemas de Arquivos -

Armazenamento a Longo Prazo

- Em muitas aplicações o arquivo é o elemento central
- 3 requisitos essenciais
  - Deve ser possível armazenar uma quantidade muito grande de informações
  - A informação deve sobreviver ao término do processo que a usa
  - Múltiplos processos tem que ser capazes de acessar as informações concorrentemente

Arquivos

- Do ponto de vista do usuário o sistema de arquivos é uma das partes principais de um SO
- Usuários desejam acessar arquivos, ler , escrever, salvar, etc.
  - Para ajudar nestes objetivos os sistemas operacionais oferecem os sistemas gerenciadores de arquivos
- O Sistema Operacional gerencia dispositivos de armazenamento
  - Esconde as dificuldades impostas pelo hardware
  - Define uma unidade lógica de armazenamento: Arquivo

Arquivos

- Um sistema de arquivos não fornece apenas uma maneira de organizar os arquivos estruturadamente
- Fornece também uma coleção de funções que podem ser executadas em arquivos
- As operações com arquivos são realizadas através de chamadas ao sistema
- As operações mais comuns são: criar, apagar, abrir, fechar, ler e escrever

Operações

- Criar: cria o arquivo sem dados, estabelecendo alguns de seus atributos
- Apagar: libera o espaço em disco ocupado pelo arquivo
- Abrir: permite ao SO buscar e colocar na memória principal os atributos e a lista de endereços do disco, para tornar mais rápido o acesso a chamadas posteriores
- Fechar: Libera o espaço ocupado com as informações obtidas com o comando de abrir
- Ler/Escrever: Lê/escreve dados a partir da posição corrente do arquivo

Atributos do Arquivo

- Um sistema de arquivos mantém uma série de atributos associados aos arquivos
  - Nome
    - Uma característica muito importante em mecanismos de abstração é o modo como os objetos são gerenciados e nomeados
    - As regras para se dar nomes aos arquivos variam de sistema para sistema
  - Tipo
    - Extensões podem oferecer dicas sobre o tipo do arquivo
    - Uma consideração importante é se o SO fornecerá suporte a tipos de arquivo
    - Se o SO reconhecer o tipo de arquivo ele poderá operá-lo de forma razoável → Ex: imprimir arquivo executável

Extensões de Arquivos - Exemplos

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

Atributos do Arquivo

- Tamanho
  - Indica o espaço que o arquivo ocupa
- Proteção
  - Informa quem pode acessar o arquivo. Senhas. Modos de acesso.
- Data/Hora
  - Data de criação, do último acesso, da última modificação
- *Flags*
  - Usados para controlar ou habilitar alguma característica específica

Atributos

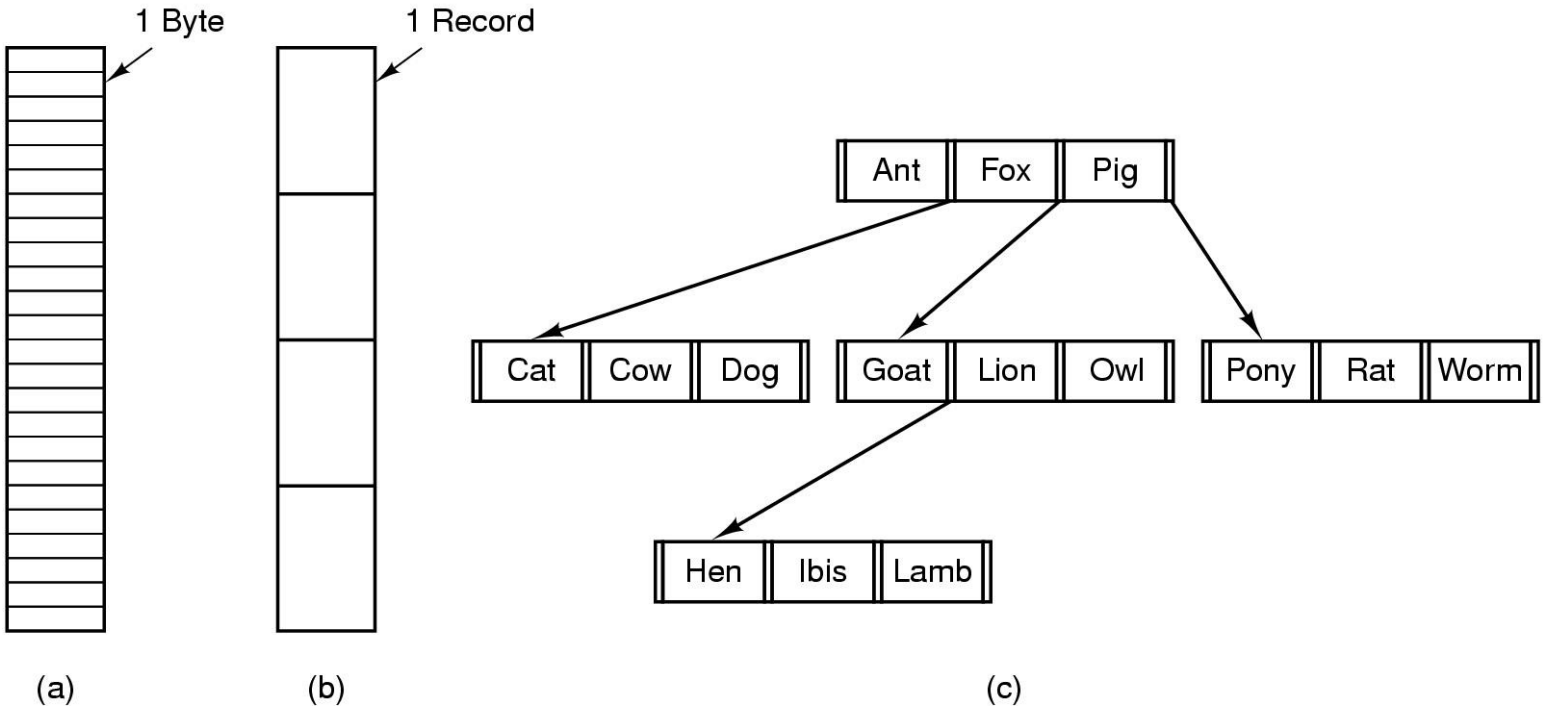
Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Estrutura de Arquivos

a) Sequência de bytes

b) Sequência de registros

c) Árvore



Acesso aos Arquivos

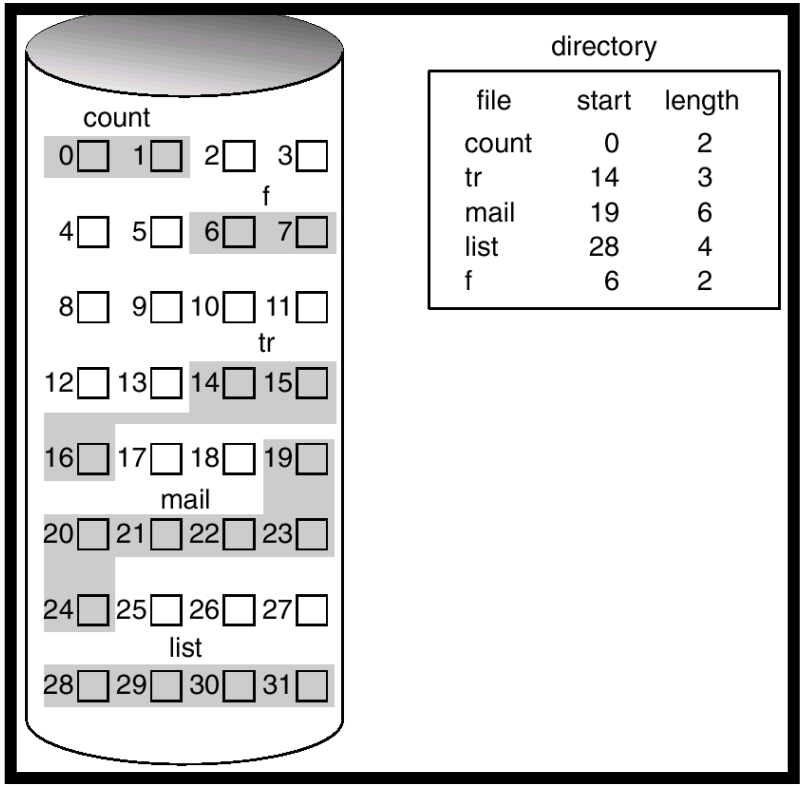
- Acesso Sequencial
  - Os primeiros tipos de sistema ofereciam somente este tipo de acesso → fitas
  - Lê uma posição do arquivo e avança para a próxima → não é possível ler fora de ordem
- Acesso Direto ou Randômico
  - Baseado no modelo de arquivo em disco
  - Permite que blocos sejam lidos e gravados em qualquer ordem
  - Essencial para vários tipos de aplicações
  - É o método mais usado

Implementação de Arquivos

- É importante saber quais blocos do disco estão relacionados a quais arquivos
  - Qual a estratégia de alocação usada?
    - Alocação Contígua
    - Alocação Encadeada
    - Alocação Indexada
- Ao se escolher uma estratégia de alocação, diversos critérios devem ser considerados:
  - Tempo de acesso, espaço em disco, facilidade de atualização, manutenção, etc.

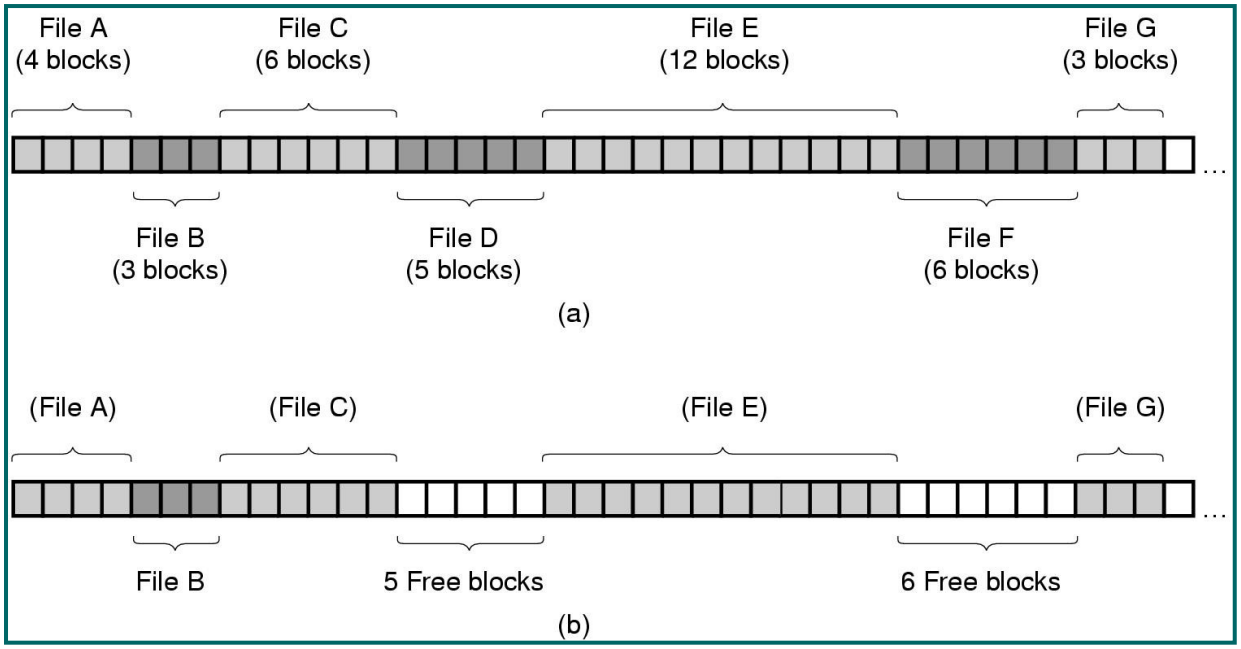
Alocação Contígua

- É o esquema mais simples de alocação
- O arquivo é armazenado em blocos contíguos do disco
- Na tabela de diretórios há a indicação do endereço inicial e o tamanho da área alocada
- Simples implementação
- Bom desempenho



Alocação Contígua

- Desvantagens
  - Com o tempo o disco fica fragmentado
    - Quando o arquivo é removido é deixada uma lacuna de blocos livres
    - Compactação é custosa
  - Dificuldade de se estimar o tamanho máximo do arquivo

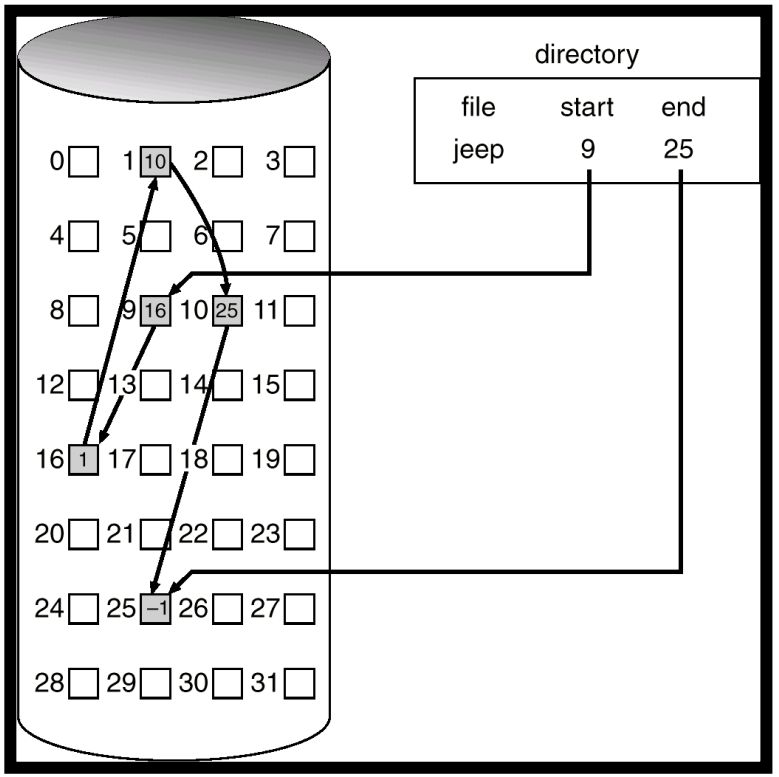


(a) Alocação contígua de 7 arquivos no disco

(b) D e E são removidos

Alocação Encadeada

- Cada arquivo é mantido como uma lista encadeada de blocos de disco
- A primeira palavra de cada bloco é usada como ponteiro para o próximo bloco.
  O restante do bloco é usado para dados
- Não existe o problema da fragmentação externa
- Não é necessário definir o tamanho máximo do arquivo



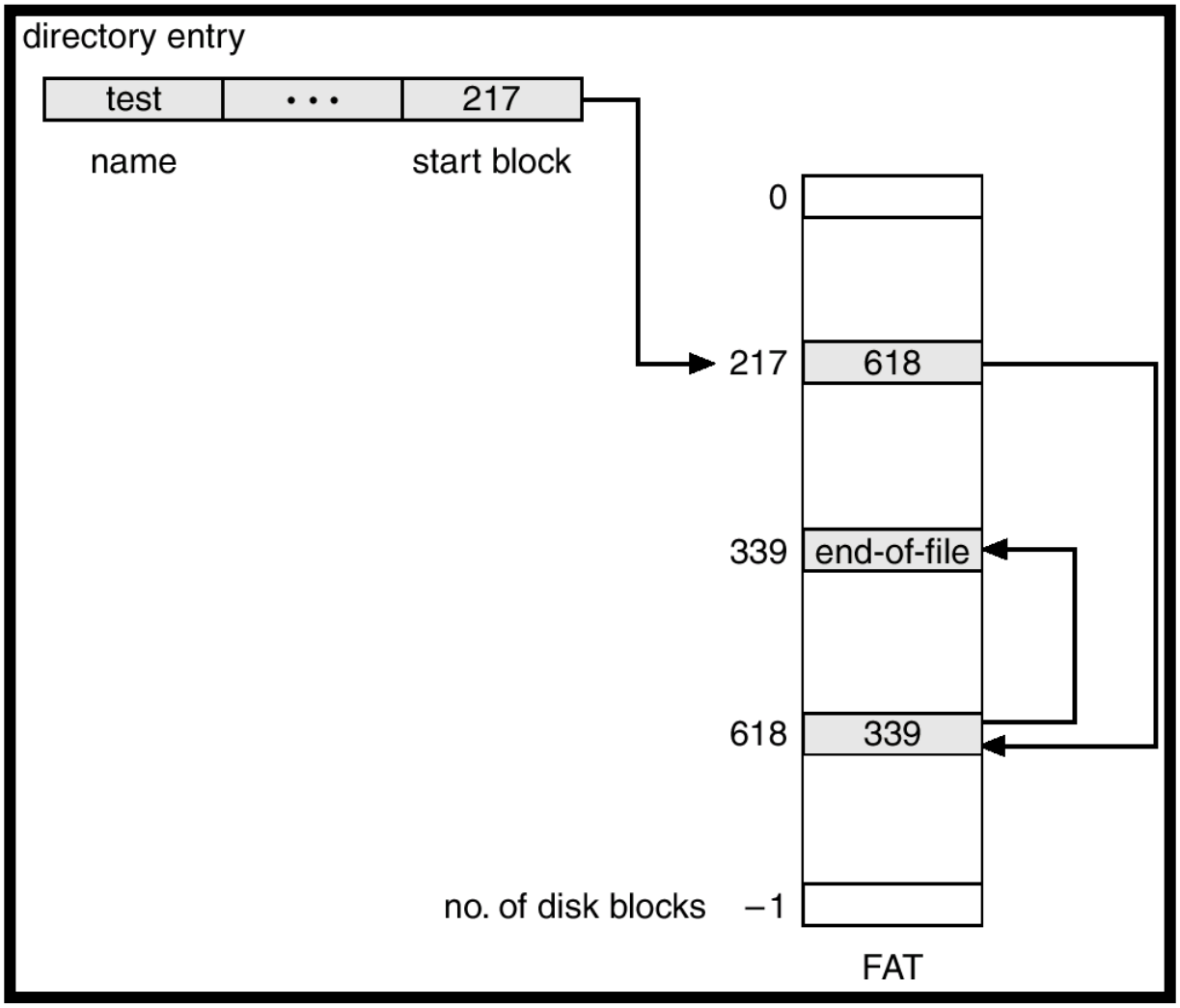
Alocação Encadeada

- Desvantagens
  - Não existe acesso direto
    - Quando se deseja acessar um bloco i, é preciso começar no primeiro bloco do arquivo e seguir os ponteiros até o i-ésimo bloco
  - Gasta espaço para armazenar os ponteiros
  - Menos confiável → se um ponteiro for danificado
- Uma variação importante do método de alocação encadeada é o uso de uma tabela de alocação de arquivos: FAT (File Allocation Table)

FAT

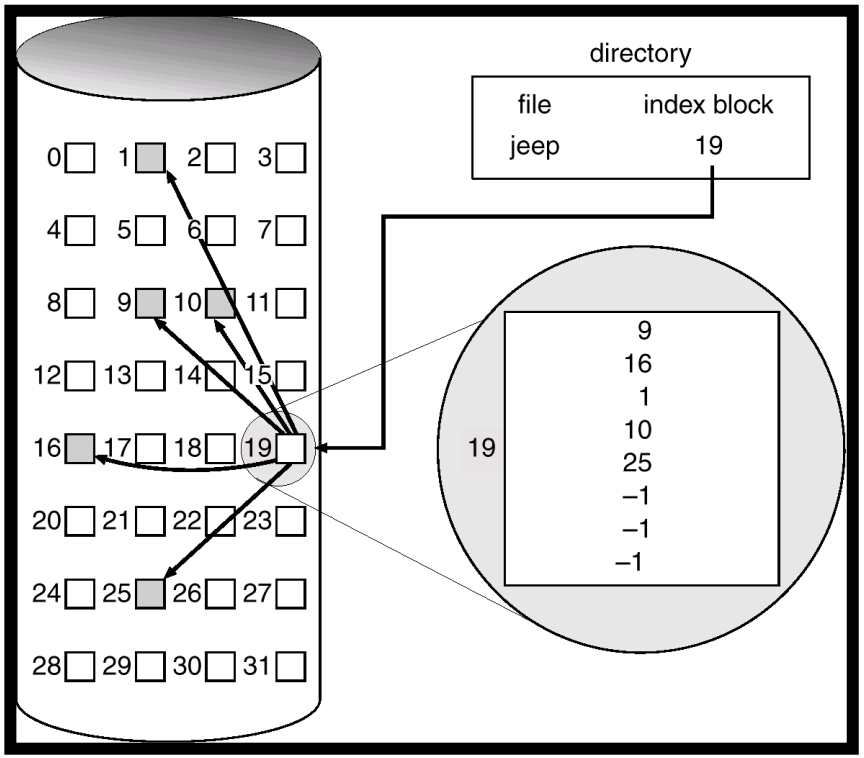
- Uma seção de disco no início de cada partição é reservada para controlar a tabela
- A tabela contém uma entrada para cada bloco de disco
  - A tabela é indexada pelo número do bloco
- A FAT é usada como uma lista encadeada
- É carregada na memória principal
  - O encadeamento é feito na MP, sem a necessidade de referência ao disco
  - A desvantagem é que toda a tabela tem que estar na memória

FAT


directory entry
test
• • •
217
name
start block
0
217
618
339
end-of-file
618
339
no. of disk blocks
−1
FAT

Alocação Indexada

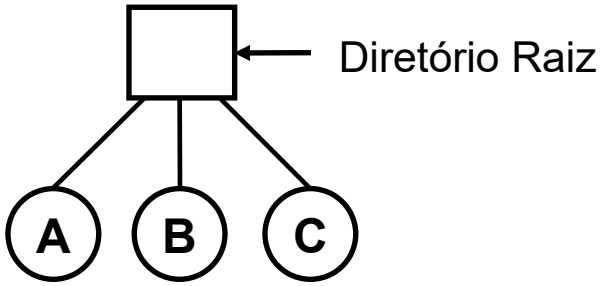
- Usa um bloco de índice que possui ponteiros para todos os blocos do arquivo
  - A entrada i do bloco de índice aponta para o bloco i do arquivo
- Cada arquivo possui seu próprio bloco de índice
- Na tabela de diretórios é guardado o endereço do bloco de índices



Alocação Indexada

- Suporta acesso direto aos blocos
- Assim como a alocação encadeada, não sofre de fragmentação externa
- Pode ocorrer um desperdício de espaço com a alocação do bloco de índices
  - Se for escolhido um bloco de índices pequeno
    - Diminui o desperdício
    - Não existirão ponteiros suficientes para um arquivo grande → uma solução é utilizar mais de um nível de blocos de índices

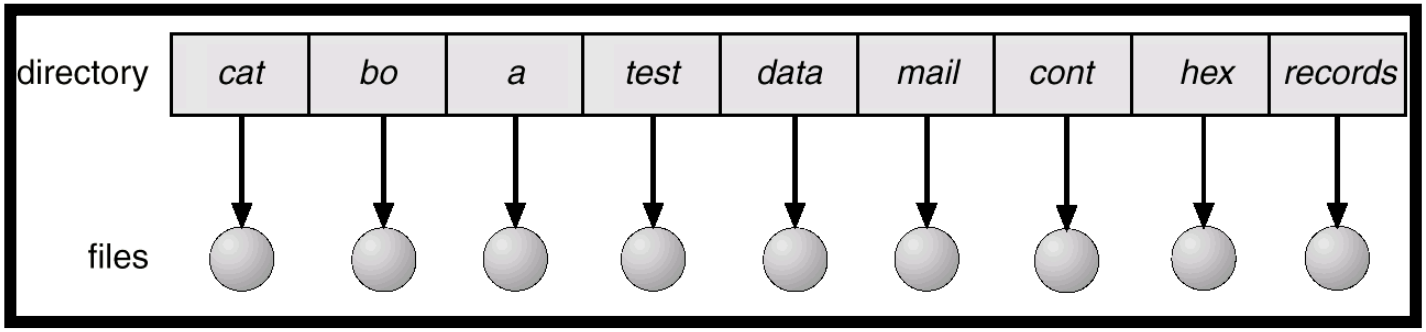
Estrutura de Diretórios



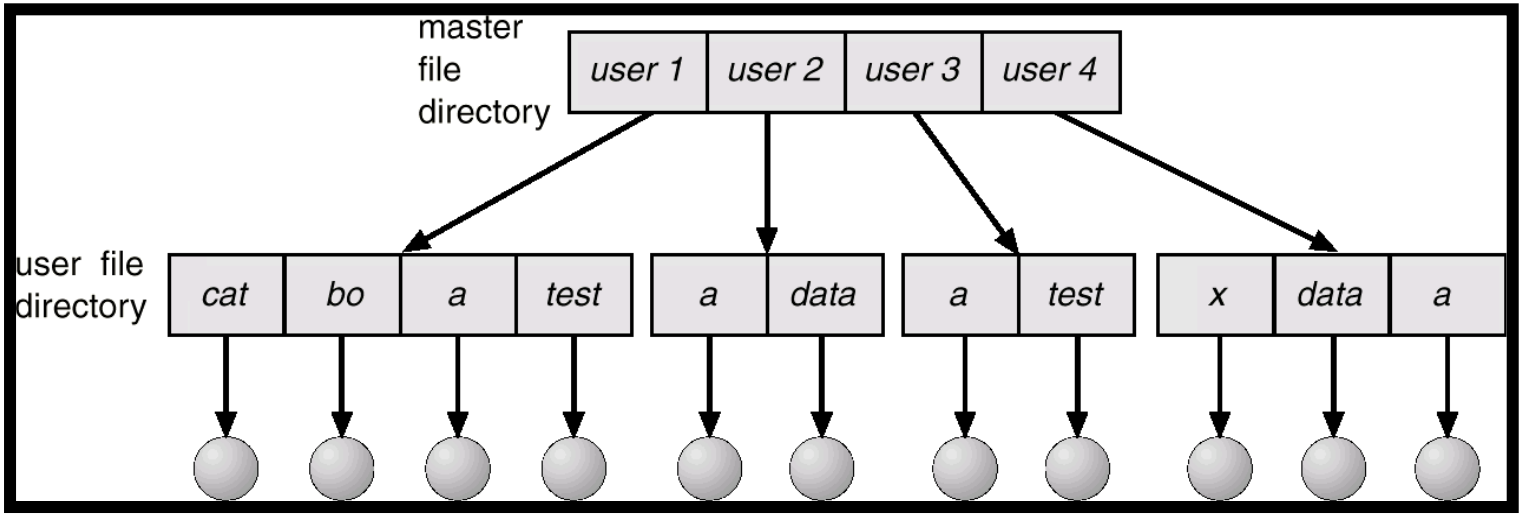
- Diretório de nível único
  - É a maneira mais simples
  - Existe apenas um diretório raiz que contém todos os arquivos
  - É difícil para o usuário gerenciar um número muito grande de arquivos
    - Muitos usuários o problema é ainda maior
    - É um problema nomear e agrupar arquivos
  - Normalmente usado em sistemas embarcados simples como telefones e câmeras digitais

Estrutura de Diretórios

- Diretório de nível único



- Diretório de dois níveis

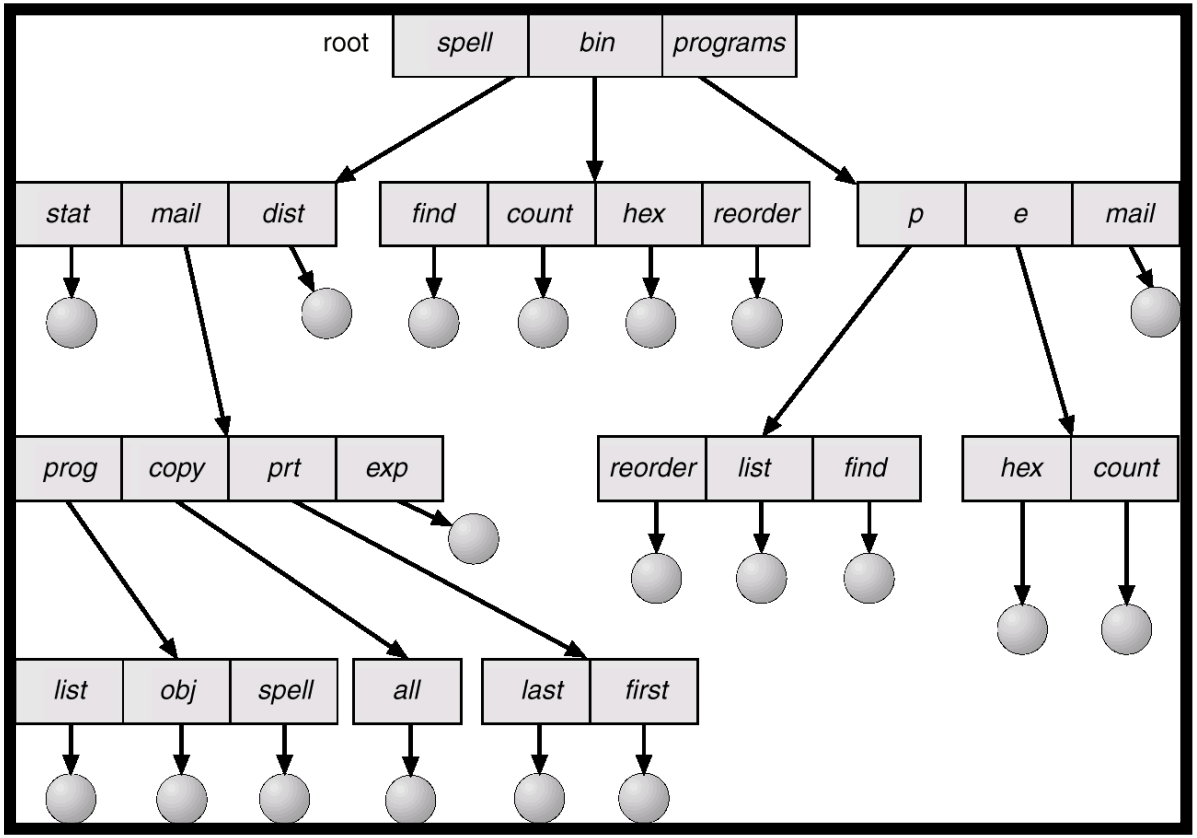


Estrutura de Diretórios

- Diretório de dois níveis
  - Cria um diretório para cada usuário (User File Directory – UFD)
  - Raiz: diretório de arquivos mestre (Master File Directory – MFD)
    - O MFD é indexado pelo nome do usuário ou número de conta.
    - Cada entrada do MFD aponta para o diretório de um usuário
  - Elimina o problema de conflitos de nomes entre arquivos de de diferentes usuários
  - Não permite que o usuário crie uma hierarquia em seu UFD
    - Ainda existem problemas para nomear e agrupar arquivos no diretório do usuário

Estrutura de Diretórios

- Diretório em árvores
  - Um diretório contém arquivos e/ou sub-diretórios

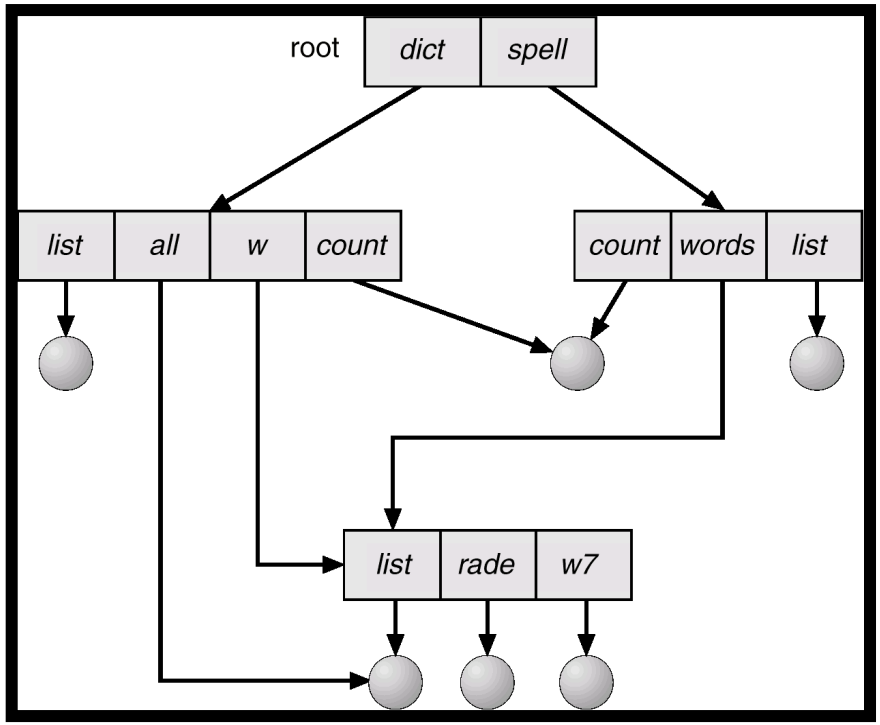


Estrutura de Diretórios

- Diretório em árvores
  - Cada usuário pode ter tantos diretórios quanto necessário, podendo agrupar seus arquivos da melhor maneira
  - Cada usuário tem um diretório corrente
    - Caso um arquivo não esteja no diretório corrente, deve se especificar um caminho
    - Nomes de caminhos podem ser absolutos (começa no diretório raiz) ou relativos (a partir do diretório corrente)
    - Um usuário pode acessar arquivos de outros usuários

Estrutura de Diretórios

- Diretório em grafos orientados acíclicos
  - É uma generalização da estrutura de árvores
  - Arquivos e diretórios podem ser compartilhados por diferentes usuários
  - Uma maneira de implementar é usando links simbólicos
  - Gerenciamento é mais complexo



Diretórios

- Operações importantes que acontecem em um diretório
  - Procurar arquivos
    - O diretório deve ser procurado para que se encontre a entrada correspondente ao arquivo
  - Criar arquivos
    - Deve ser criada uma nova entrada no diretório
  - Apagar arquivos
    - A entrada correspondente ao arquivo deve ser removida
  - Listar Diretórios
  - Atualizar Diretórios

Implementação de Diretórios

- Lista linear
  - O diretório constitui uma lista de entradas, uma para cada arquivo
    - Pode ser representada como um arquivo sequencial simples
  - Simples de programar porém perde em eficiência
    - A pesquisa linear para se encontrar um arquivo é demorada
- Tabela Hash
  - O diretório é implementado usando uma estrutura de tabela hash
  - A busca é mais rápida, porém o gerenciamento é mais complicado

Gerenciamento do Sistema de Arquivos

- Algumas questões devem ser consideradas
  - Gerenciamento do espaço em disco
  - Monitoramento dos blocos livres
  - Cópia do sistema de arquivos
  - Desempenho do sistema de arquivos
  - Redução do movimento do braço do disco

Gerenciamento do Espaço em Disco

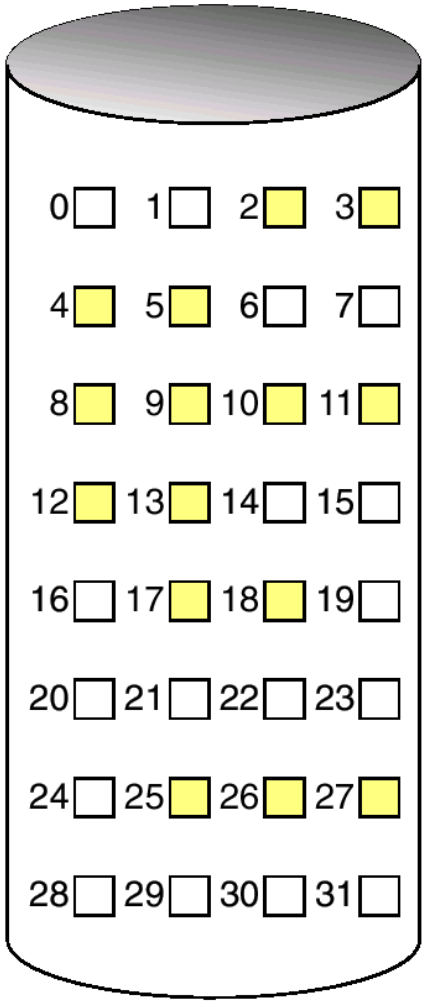
- Os arquivos são divididos em blocos de tamanho fixo
  - Qual o tamanho ideal do bloco?
- Blocos muito grandes significam que arquivos pequenos irão desperdiçar espaço significativo no disco
- Blocos muito pequenos indicam que a maioria dos arquivos irá ocupar mais de um bloco
  - Múltiplas buscas → reduzem o desempenho
- Bloco muito grande → desperdício de espaço
- Bloco muito pequeno → desperdício de tempo

Monitoramento dos Blocos Livres

- Para controlar a quantidade de espaço livre em disco o sistema mantém uma lista de blocos livres
- Mapa de bits
  - A lista de blocos livres é implementada como um vetor de bits
  - Cada bloco é representado por um bit
    - Bloco livre → bit 1
    - Bloco alocado → bit 0
  - Para encontrar um bloco livre basta encontrar o primeiro bit 1 do vetor

Monitoramento dos Blocos Livres

- Exemplo de Mapa de bits

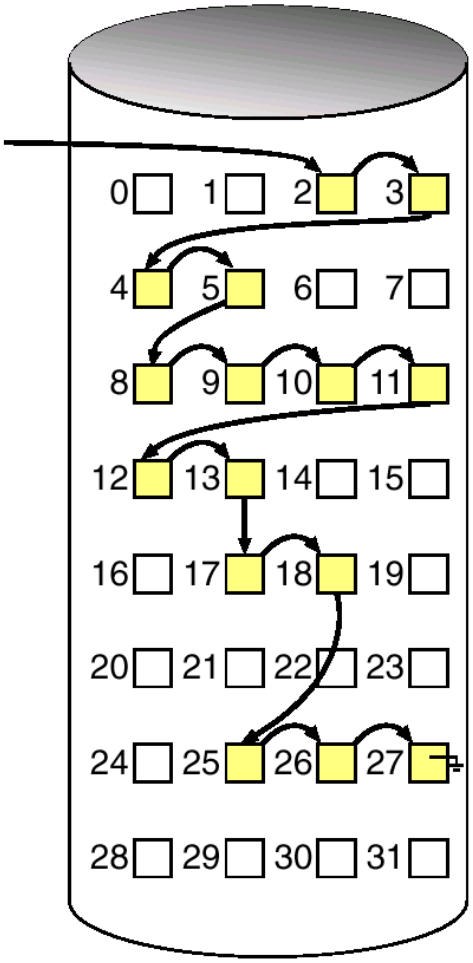


- Os blocos amarelos estão livres
- Um disco com *n* blocos requer um mapa com *n* bits

  00111100111111000110000001110000
- Com o vetor de bits carregado na MP, os blocos livres podem ser procurados sem acesso ao disco
  - Em discos muito grandes o vetor de bits pode consumir muito espaço de memória

Monitoramento dos Blocos Livres

- Lista Encadeada

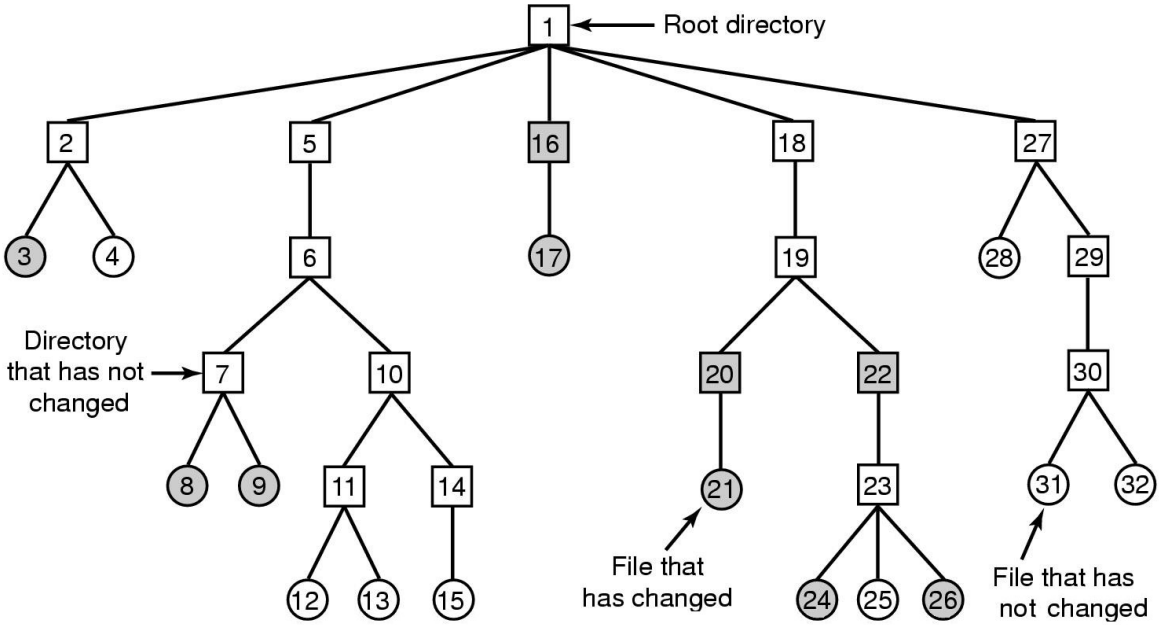


- Os blocos livres são mantidos em uma lista encadeada
  - Mantém apenas o endereço do primeiro bloco
- Não há problema de espaço na MP pois não precisa de uma tabela de alocação
- Requer maior tempo para achar blocos livres, pois a lista deve ser percorrida
  - Perde mais tempo com E/S

Cópia do Sistema de Arquivos

- Como os dispositivos de armazenamento podem falhar é necessário que se façam cópias de segurança
- Uma cópia completa pode consumir muito tempo
- Para minimizar a quantidade de cópia necessária pode se adotar a cópia incremental
  - Faz uma cópia completa (semanalmente/mensalmente)
  - Faz uma cópia diária apenas dos arquivos modificados

Cópia do Sistema de Arquivos - Exemplo



(a) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

(b) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

(c) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

(d) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Cópia do Sistema de Arquivos

- Restauração
  - Inicialmente a cópia completa mais recente é restaurada
  - Como os diretórios aparecem primeiro na fita, cria-se um esqueleto do sistema de arquivos
  - A seguir são copiados os arquivos
  - O processo de restauração é repetido para a primeira cópia incremental feita após a cópia completa, e assim por diante

Desempenho do Sistema de Arquivos

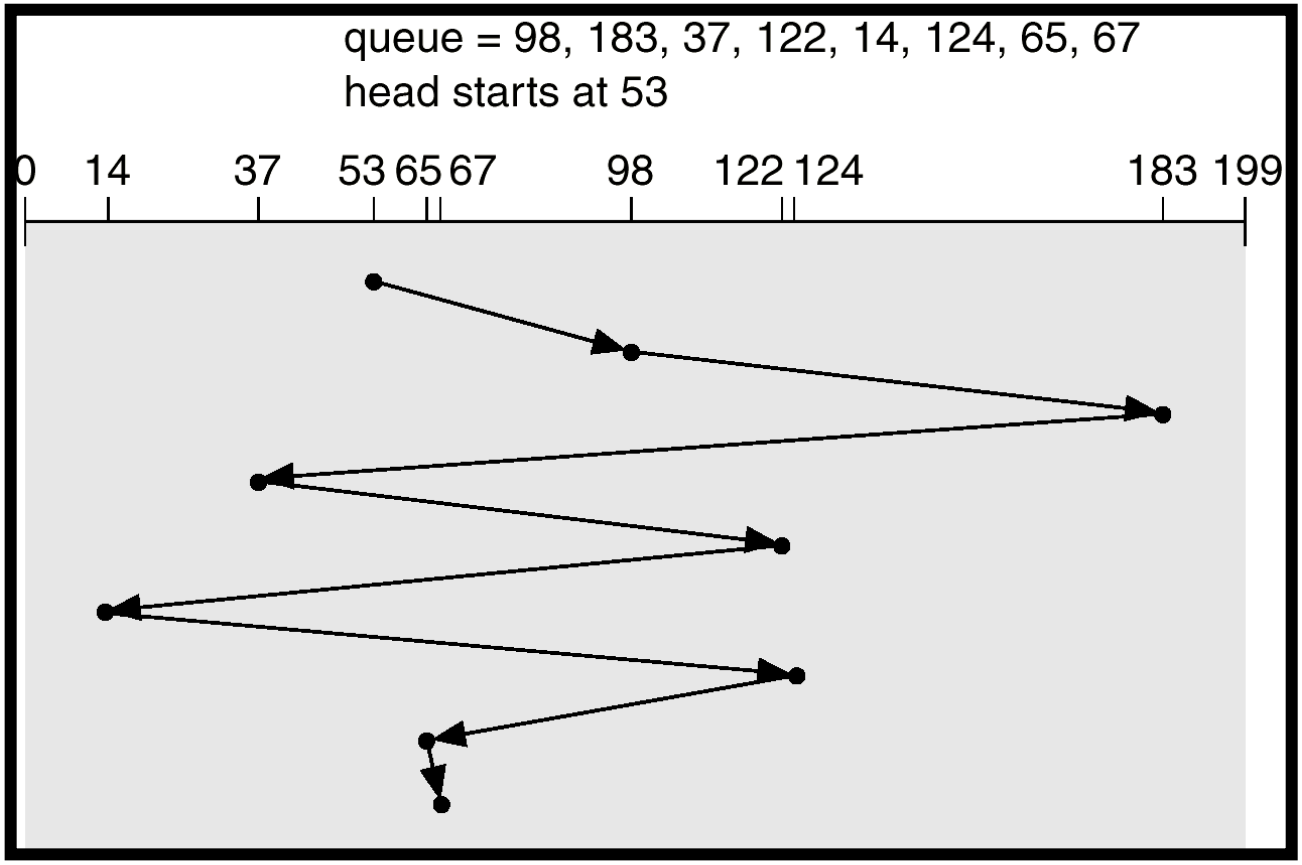
- O acesso ao disco é mais lento que o acesso a memória principal
- A técnica mais usada para reduzir o acesso ao disco é usar uma *cache* de blocos
  - A *cache* de blocos é uma coleção de blocos que pertencem ao disco mas que são mantidos na memória principal para aumentar o desempenho
  - Alguns sistemas otimizam sua *cache* de disco utilizando diferentes algoritmos de substituição de blocos, dependendo do tipo de acesso ao arquivo
    - Leitura antecipada de blocos

Escalonamento de Disco

- O tempo necessário para ler ou escrever um bloco no disco é determinado por três fatores
  - Tempo de posicionamento
  - Atraso de rotação
  - Tempo de transferência real do dado
- A redução do tempo médio de posicionamento pode melhorar substancialmente o desempenho do sistema
- Diferentes algoritmos de escalonamento de disco podem ser usados para selecionar os pedidos pendentes

FCFS – *First Come First Served*

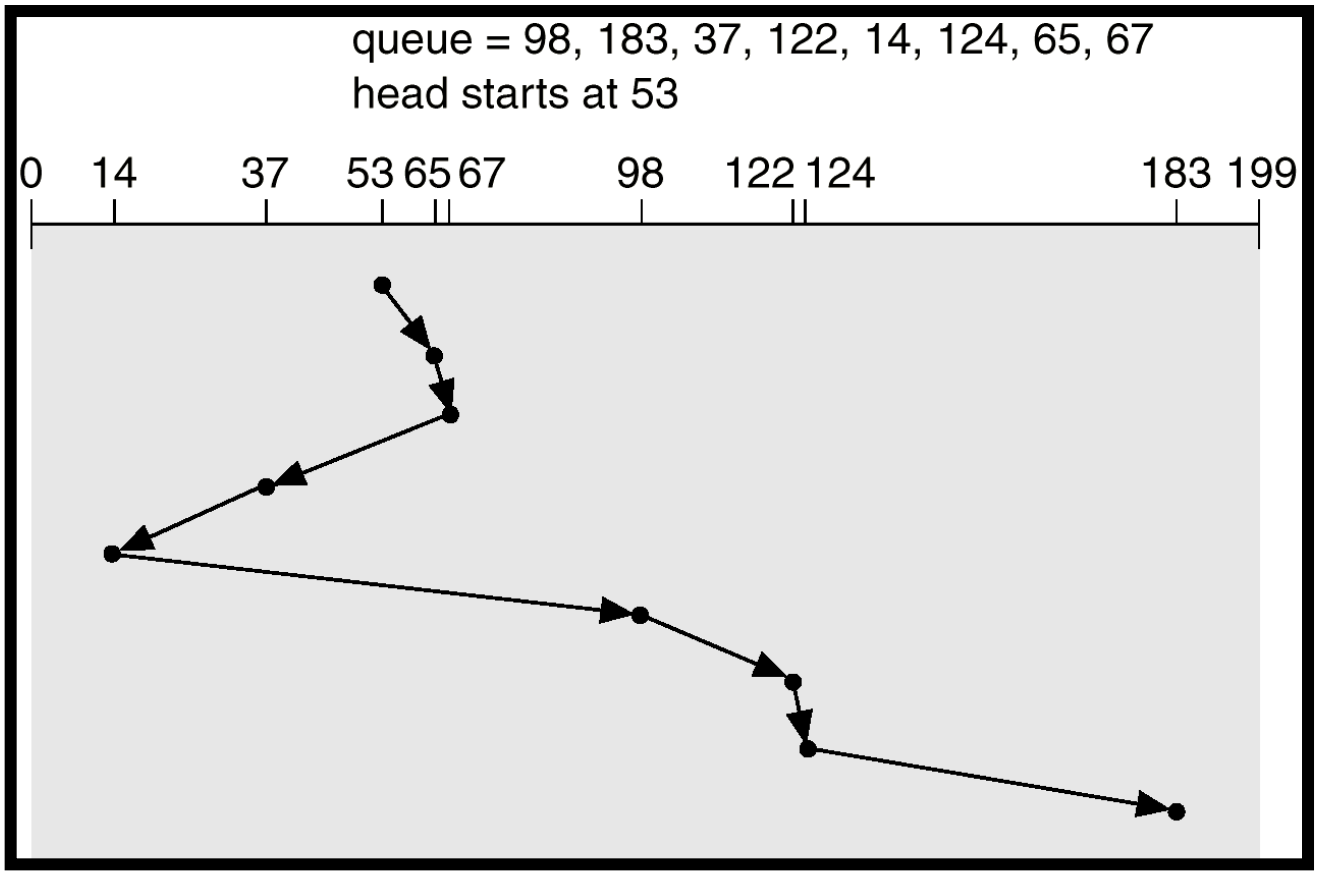
- A política de escalonamento mais simples
- Algoritmo justo, mas não necessariamente o mais rápido



Movimento
de 640 cilindros

SSF – *Shortest Seek First*

- Escolhe o pedido que seja o mais próximo da posição atual do cabeçote, afim de minimizar o tempo de posicionamento

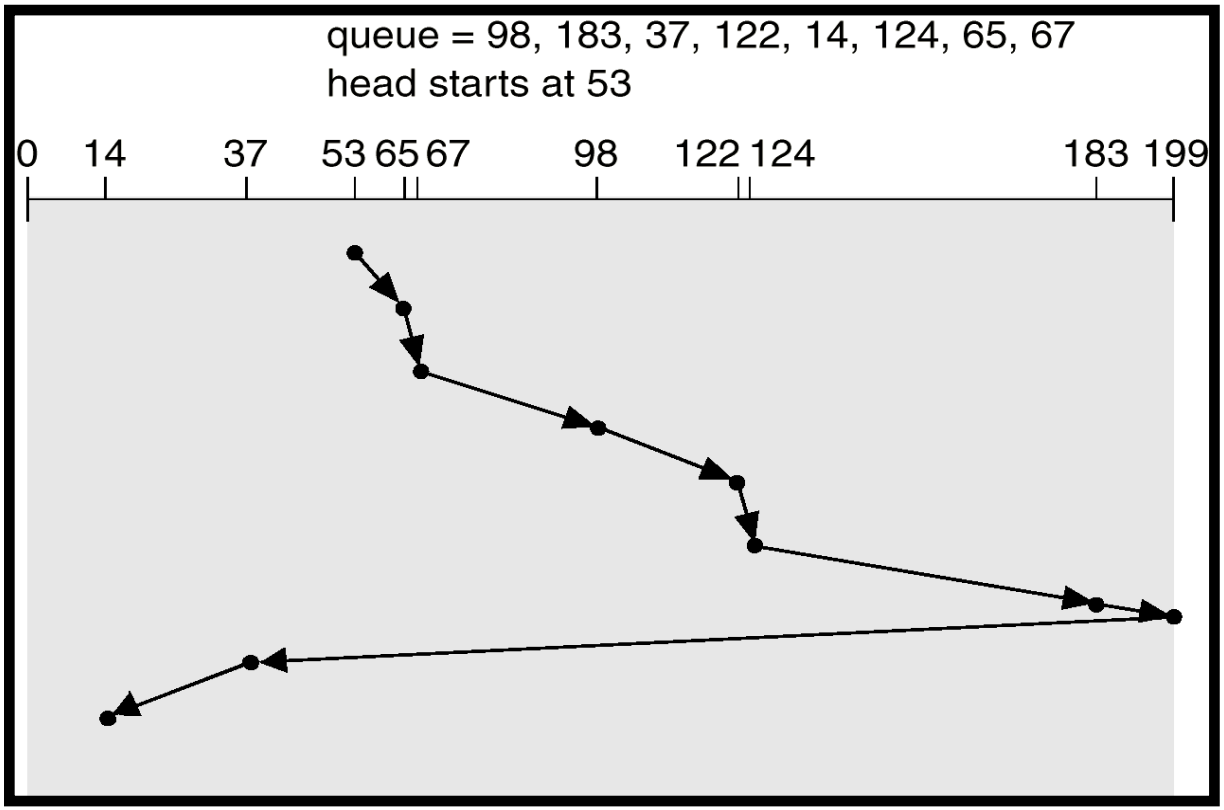


Movimento
de 236 cilindros

Scan

- O braço do disco se move em uma única direção atendendo os pedidos até atingir uma “ponta” do disco
  → A direção é invertida e o serviço continua

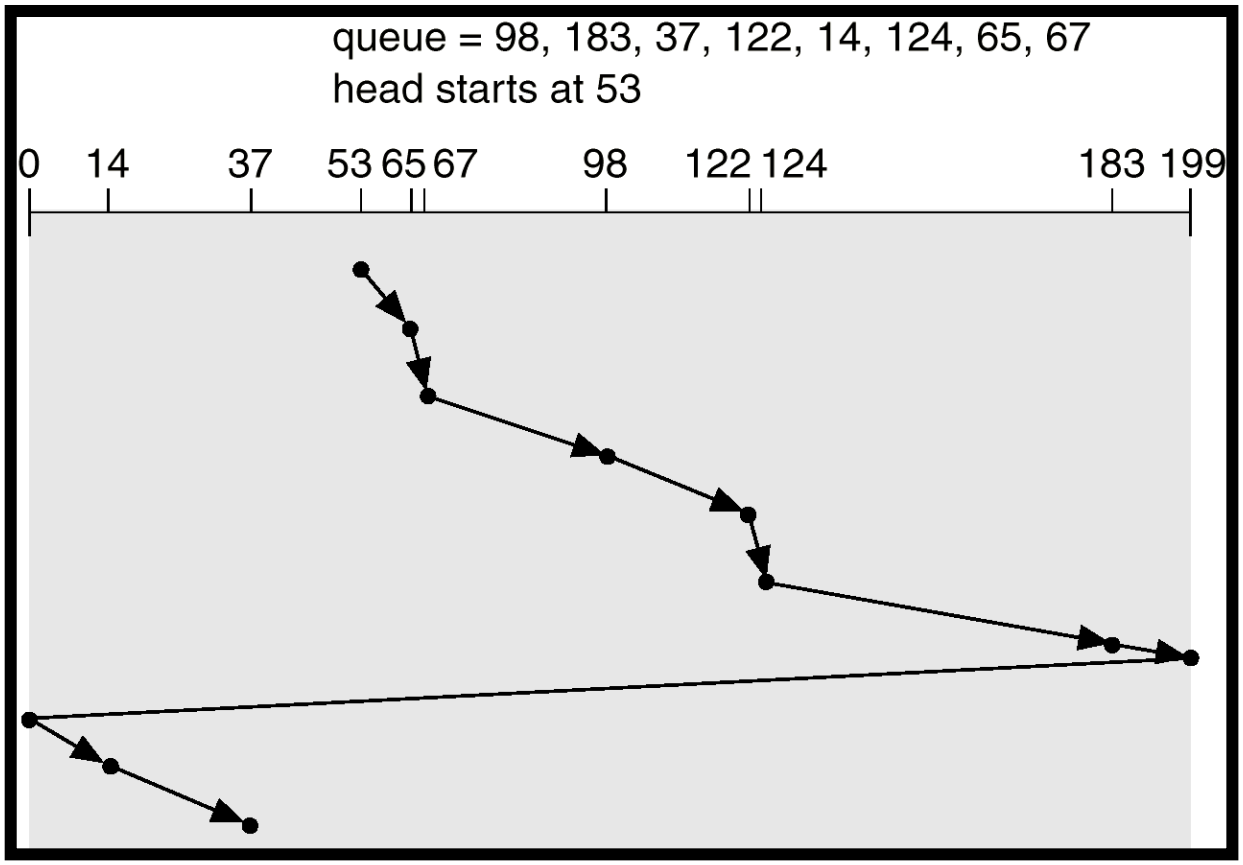
- Algoritmo do elevador



Movimento
de 331 cilindros

C-Scan

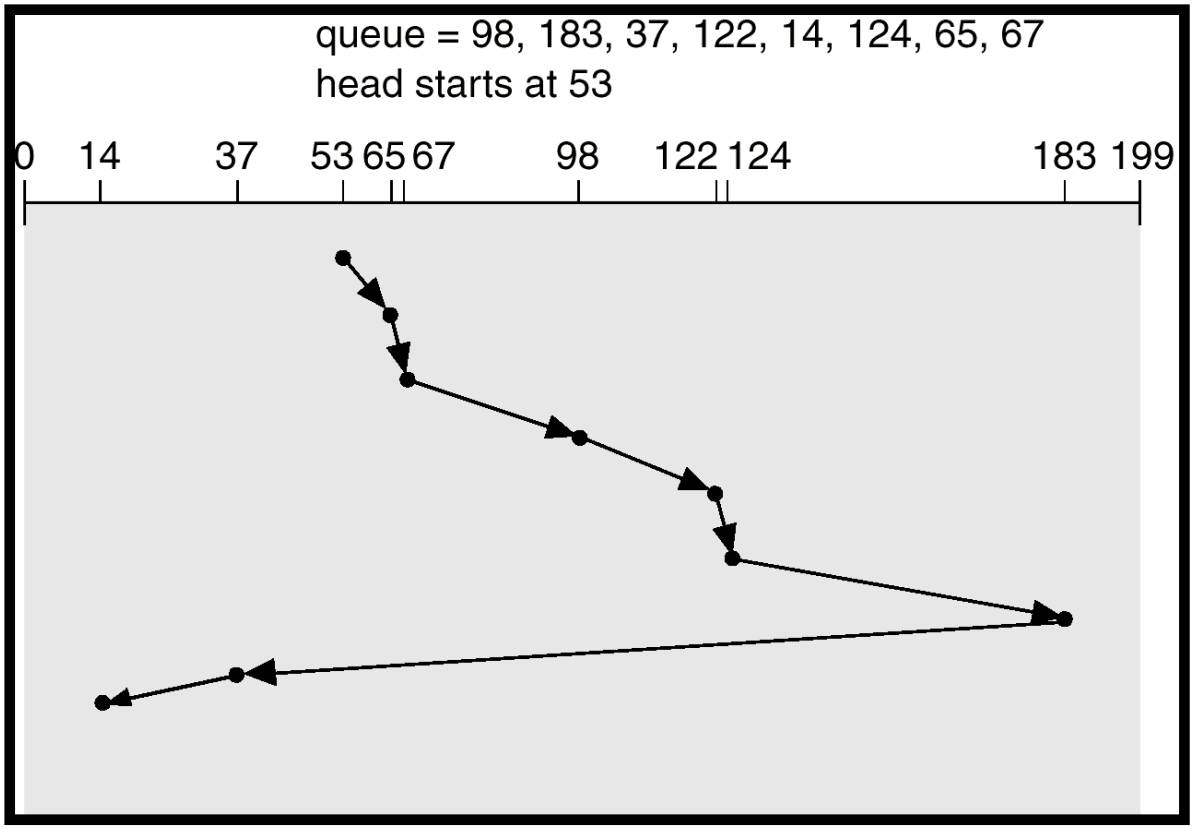
- Oferece um tempo de espera mais regular
- Similar ao algoritmo Scan, porém os pedidos são atendidos apenas em uma direção.



Movimento
de 382 cilindros

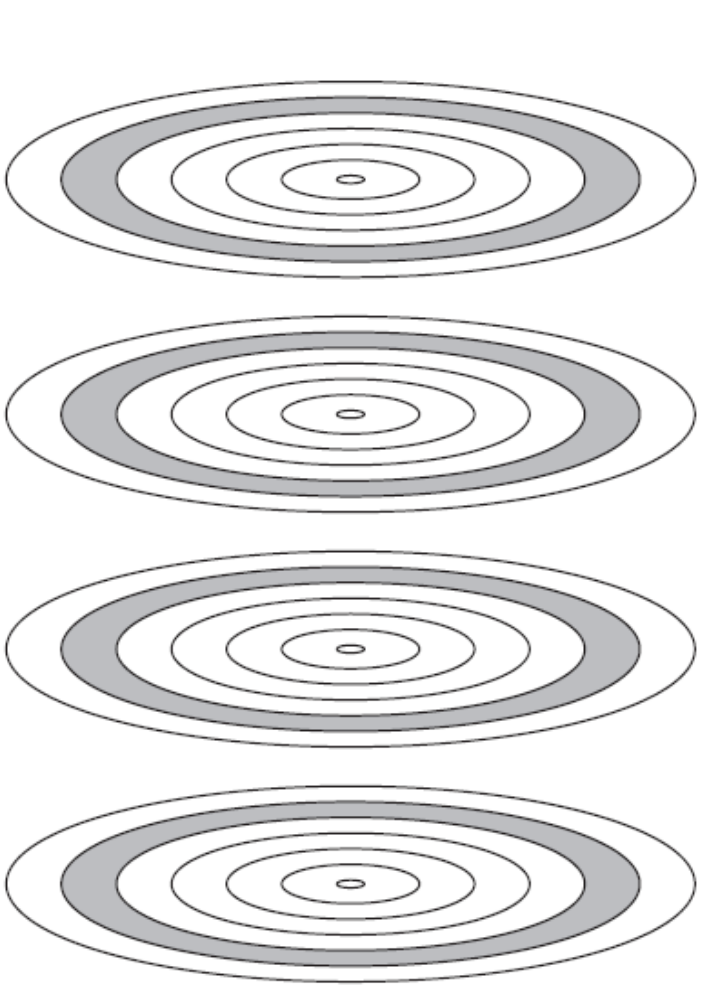
Look

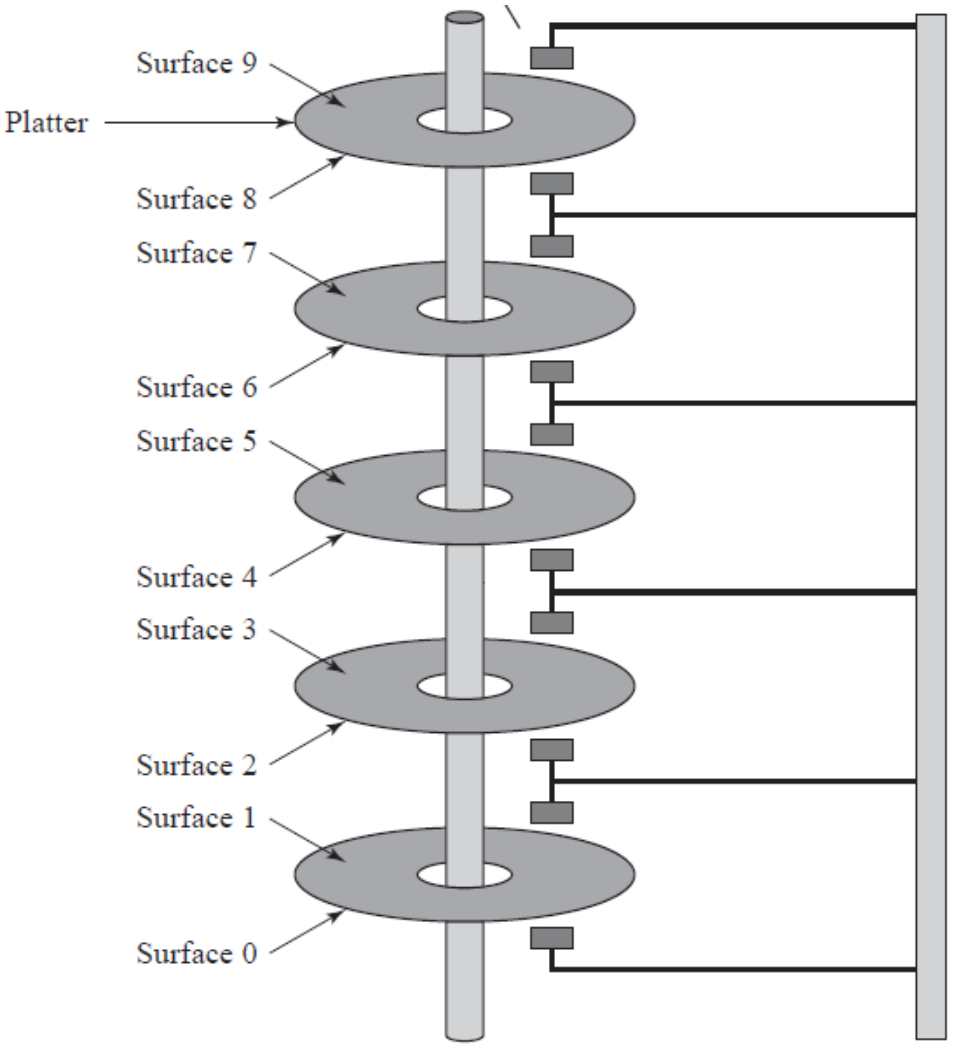
- É uma variação do algoritmo Scan
- Invertem a direção do movimento da cabeça do disco quando chega ao último pedido → Não é necessário ir até a “ponta” do disco



Movimento
de 299 cilindros

Trilhas e Cilindros

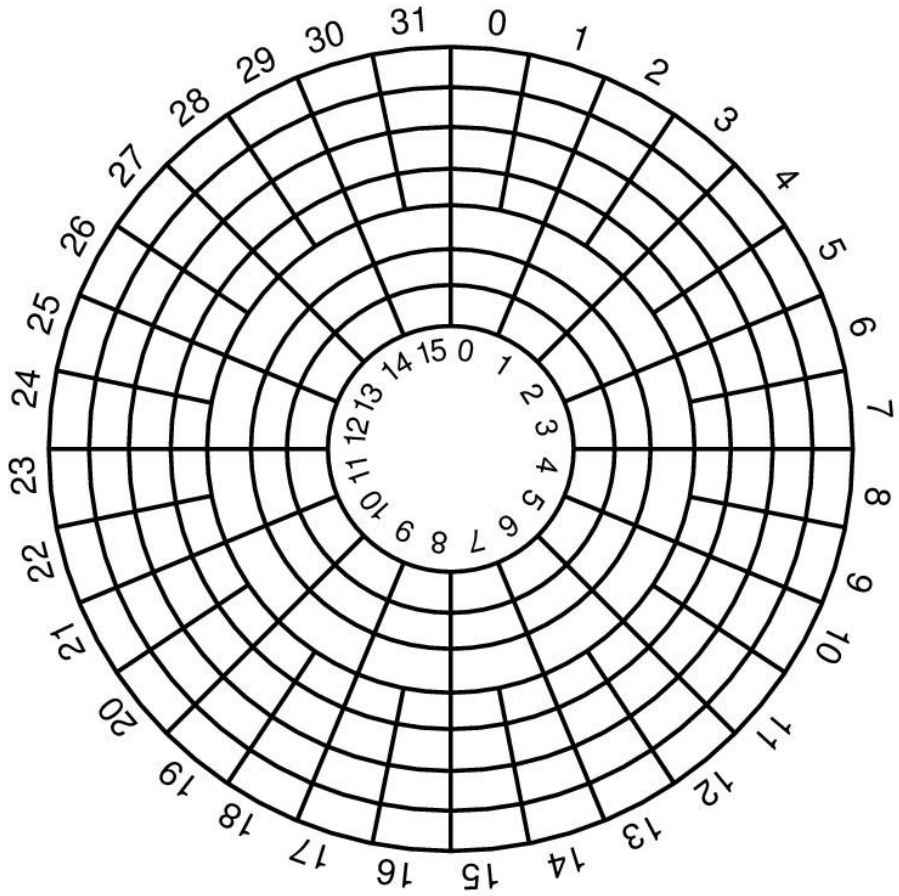


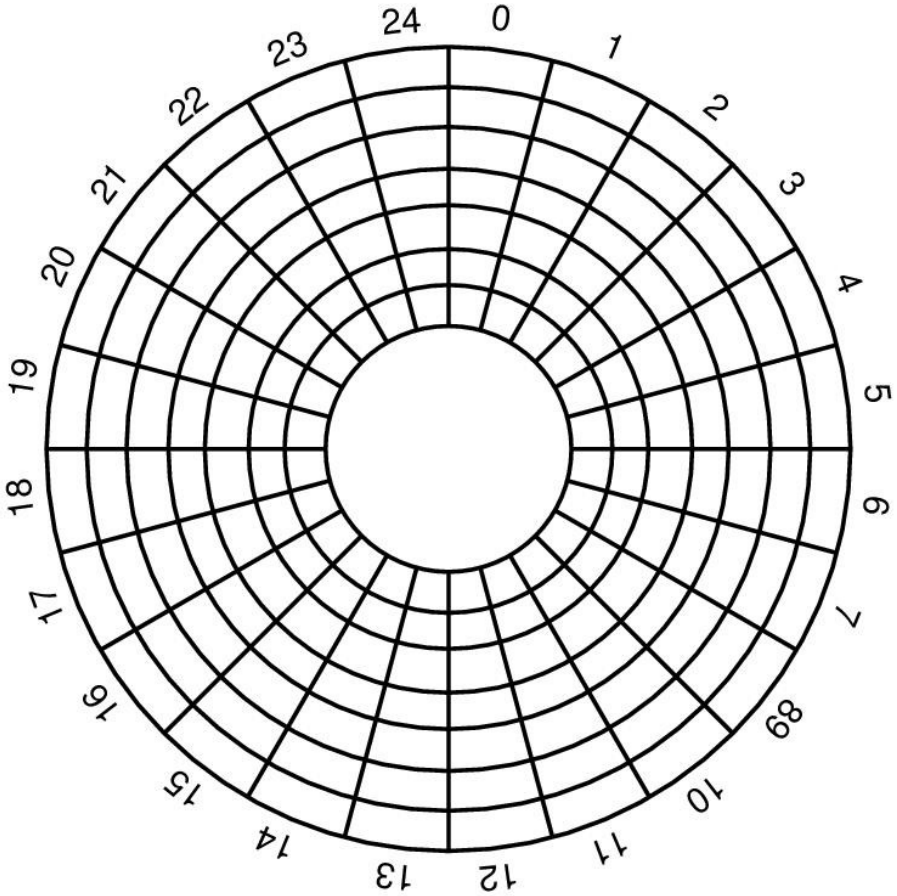


Formatação de Disco

- Antes que o disco possa ser usado, cada prato deve receber uma formatação de baixo nível
  - A formatação é feita como parte do processo de fabricação
- A formatação consiste em criar uma série de trilhas concêntricas, cada uma com um certo número de setores
- Formato do setor

Formatação de Disco

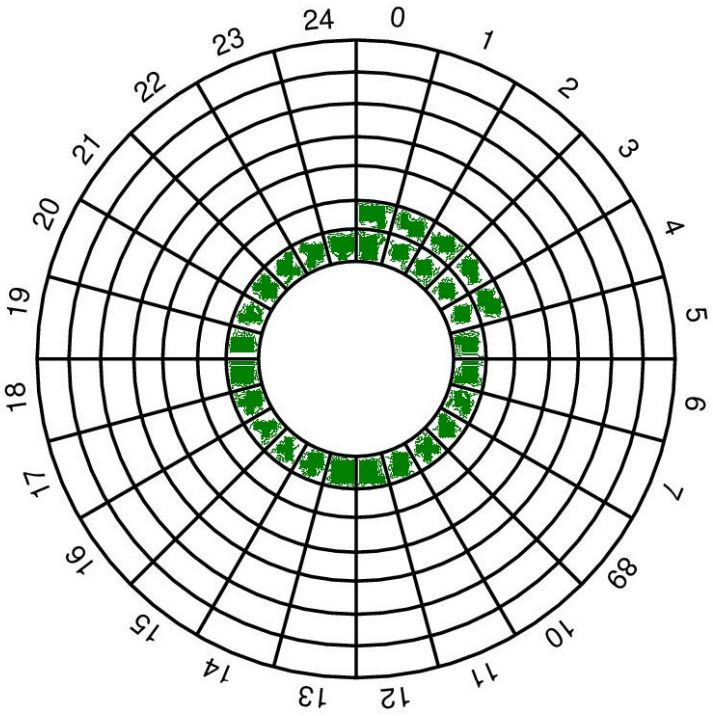




Formatação de Disco

- Na formatação de baixo nível, a posição do setor 0 em cada trilha é deslocada com relação à trilha anterior

- O Problema



- Solução