ACH2024

Aula 15

Organização de arquivos: Alocação sequencial ordenada alocação ligada

Profa. Ariane Machado Lima



Aulas passadas

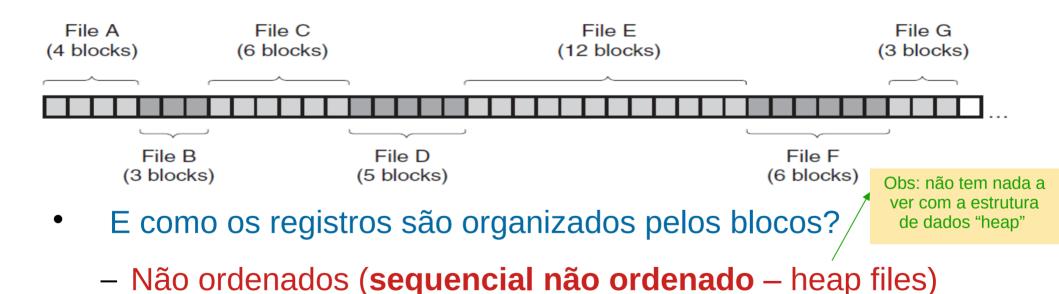
Algoritmos e estruturas de dados para lida com memória secundária

- Organização interna de arquivos
- Acesso à memória secundária (por blocos seeks)
- Tipos de alocação de arquivos na memória secundária:
 - Sequencial (ordenado e não ordenado)
 - Ligada
 - Indexada
 - Árvores-B
 - Hashing (veremos também hashing em memória principal)
- Algoritmos de processamento cossequencial e ordenação em disco



Aula passada - Alocação sequencial

Blocos alocados sequencialmente no disco (pelos cilindros)



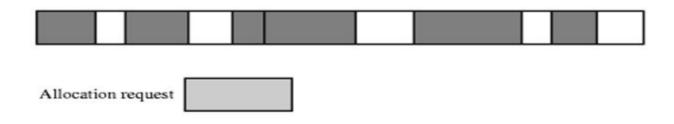
 Ordenados por um campo chave (sequencial ordenado sorted files)

EACH

Alocação sequencial

Fragmentação externa:

 Com o tempo (após alocações/desalocações sucessivas), o disco pode ficar fragmentado, isto é, com vários trechos disponíveis intercortados por trechos utilizados



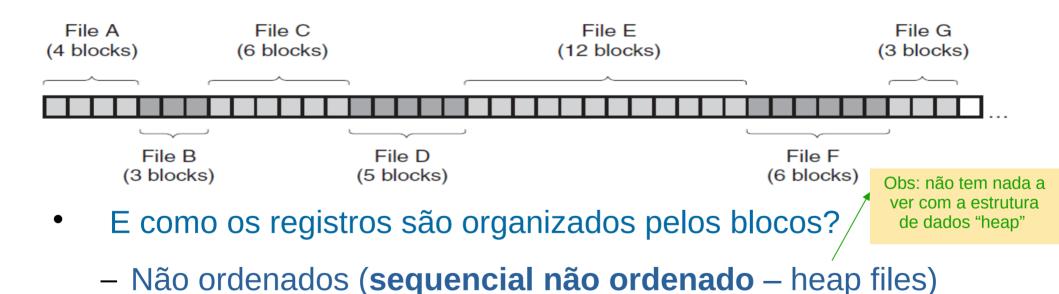


	Sequencial		
	Não-Ordenado		
Busca	O(b)		
Inserção**	O(1) se tiver espaço no final; O(b) c.c.		
Remoção* · **	O(1)		
Leitura ordenada	ω (b) (depende do alg de ord. externa)		
Mínimo/máximo	O(b)		
Modificação**	O(1)		
* considerando uso	de bit de validade		
** considerando que	e já se sabe a localizaçã	ão do registro (busc	a já realizada)



Aula de hoje - Alocação sequencial

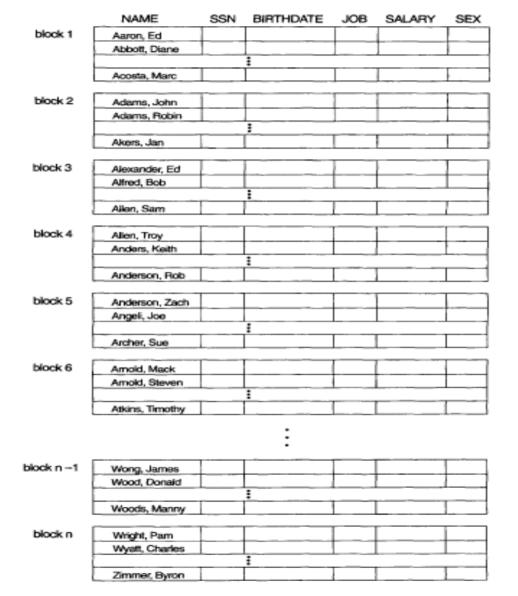
Blocos alocados sequencialmente no disco (pelos cilindros)



 Ordenados por um campo chave (sequencial ordenado sorted files)

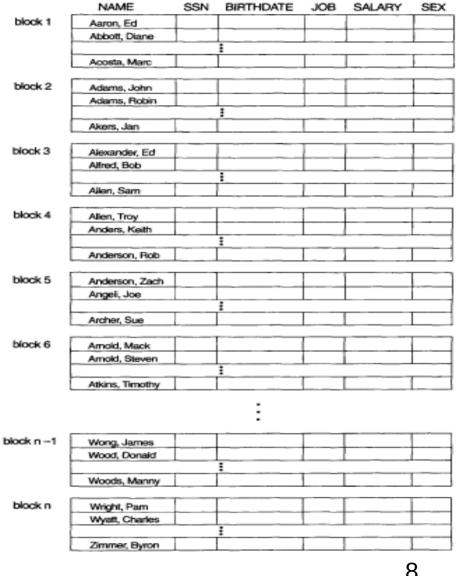
EACH

Os r registros estão ordenados por um campo específico - a chave





Leitura ordenada





Pego os nrs do primeiro e último bloco no cabeçalho do arquivo (que está em memória), e

```
for (int i = primeiro bloco; i <= ultimo bloco; i++) {
     read disk(i);
     ... /* leio os registros do bloco */
```

Leitura ordenada eficiente (sequencial) – O(b) seeks,

mas se fizer a leitura toda de uma vez, apenas O(1) seeks!

(considerando que os blocos estarão sequenciamente dispostos pelo(s) cilindros)

block 1	Aaron, Ed			
	Abbott, Diane			
		1		
	Acosta, Marc			
block 2	Adams, John			
	Adams, Robin			
		- :		
	Akers, Jan			
block 3	Alexander, Ed			
	Alfred, Bob			
		.		
	Allen, Sam			
block 4	Allen, Troy			
	Anders, Keith			
		1		
	Anderson, Rob			
block 5	Anderson, Zach			
	Angeli, Joe			
		ŧ		
	Archer, Sue			
block 6	Arnold, Mack			1
	Arnold, Steven			
		i		
	Atkins, Timothy			
		:		
block n -1	Wong, James			
Dioce II I	Wood, Donald		- - -	
	VVCCU, Conard			
	Woods, Manny	<u>-</u> -		
	WOODS, Mariny			
block n	Wright, Pam			
Diogram	Wyatt, Charles			
	TIYON, CHEROS	- i		
	Zimmer, Byron	<u> </u>		
	Zittiner, byroti			
				9

NAME

Abbott, Diane

Acosta, Marc

Adams, John

block 1

block 2

block 3

block 4

block 5

NAME

Aaron, Ed

Alexander, Ed Alfred, Bob

Amold, Steven Atkins, Timothy

Zimmer, Byron

Adams, John
Adams, Robin

BIRTHDATE

 Mínimo (menor chave) / Máximo (maior chave): Allen, Troy Anders, Keith Anderson, Ro

Anderson, Rob

Anderson, Zach
Angeli, Joe

Archer, Sue

Amold, Mack

block 6

:

SALARY

SEX

block n -1

Wood, Donald

Woods, Manny

Wright, Pam
Wyatt, Charles



block n

- Mínimo (menor chave) / Máximo (maior chave): basta obter o nr do primeiro/último bloco, que está no cabeçalho (0 seeks) e acessar o bloco – 1 seek
 - Isto podemos fazer para todos os tipos de alocação em que os registros estejam ordenados

Aaron, Ed			
Abbott, Diane			
	i		
Acosta, Marc			
Adams, John			
Adams, Robin			
	1		
Akers, Jan			
Alexander, Ed	1		
Alfred, Bob			
	1		
Allen, Sam	1		
Allen, Troy			
Anders, Keith			
P d Halo cop c control	- i		
Anderson, Rob	Ť		
Paradison, room			
Anderson, Zach			
Angeli, Joe			
Aliger, Joe	<u>+</u>		
Archer, Sue			
Aldren, ode			
Amold, Mack		\neg	
Arnold, Steven	-		
Pariou, oicron			
Atkins, Timothy	Ť		
Addis, Intolly			
	:		
Wong, James			
Wood, Donald			
	E		
Woods, Manny			L
Wright, Pam			
Wyatt, Charles			
VIYOU, CHEERGS			
viyes, Granca	i		

block t

block 2

block 3

block 4

block 5

block 6

block n

block 1

Aaron, Ed Abbott, Diane

Acosta, Marc

Adams, John

NAME

BIRTHDATE

SEX SALARY

block 2

block 3

błock 4

Adams, Robin Akers, Jan

Allen, Sam

Alien, Troy Anders, Keith

Anderson, Rob

Anderson, Zach

Alexander, Ed Alfred, Bob

Busca:

block 5

block 6

block n -1

block n

Angeli, Joe Archer, Sue

Atkins, Timothy

Wong, James Wood, Donald Woods, Manny

Amold, Mack Amold, Steven

Wright, Pam Wyatt, Charles

Zimmer, Byron

 Busca: dá para usar busca binária (baseada nos blocos!)

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
			:			
	Acosta, Marc		1			
block 2	Adams, John	i				
	Adams, Robin					
			1	-		
	Akers, Jan					
block 3	Alexander, Ed		1)		
	Alfred, Bob					
			1			
	Allen, Sam		1			
block 4	Allen, Troy					
	Anders, Keith					
	7 4 43 6 1 4 5 6 6 1		1			
	Anderson, Rob					-1
	7410010011,1100					
block 5	Anderson, Zach					
	Angeli, Joe					
	reigen, see		ŧ			
	Archer, Sue		·			
	740101,000					
block 6	Amold, Mack					
	Arnold, Steven					
	Parious, Gravers		1			
	Atkins, Timothy					
	Parato, Intioning					
			•			
			:			
block n -1	Wong, James					
	Wood, Donald					
	L		<u>:</u>			
	Woods, Manny					
					,	
block n	Wright, Pam					
	Wyatt, Charles					
			:			
	Zimmer, Byron					



 Busca: dá para usar busca binária (baseada nos blocos!)

Como é mesmo o algoritmo de busca binária (em memória)?

	Abbott, Diane			
		1		
	Acosta, Marc			
block 2	Adams, John			
	Adams, Robin			
	Akers, Jan			
	Property Court			
block 3	Alexander, Ed	j		
0100110	Alfred, Bob			
	Alieu, Dou	-		
	Allen, Sam			
	Alleri, Surri			
block 4	Alban Torri			
DRUCK 4	Alien, Troy			
	Anders, Keith			
	L	<u>i</u>		
	Anderson, Rob			
block 5	Anderson, Zach			
	Angeli, Joe			
	Archer, Sue			
block 6	Amold, Mack			
	Arnold, Steven		1	
		Ė		
	Atkins, Timothy			
		:		
ock n1	Wong, James			
	Wood, Donald			
	Woods, Manny			
block n	Wright, Pam			
	Wyatt, Charles			
	Zimmer, Byron			
				1 /



Busca binária em vetor - em memória (assunto de AED 1)

```
// Busca binaria em lista ordenada
int buscaBin (TIPOCHAVE ch, LISTA 1)
      int inf, sup, meio;
      inf = 0;
      sup = 1.nroElem - 1;
      while (inf <= sup)
                                                                        Ex: busca do nr 3
                                                                        [encurtador.com.br/uvBCO0]
                   meio = ((inf + sup) / 2);
                   if(l.A[meio].chave == ch) return(meio); // achou
                   else
                         if(1.A[meio].chave < ch) inf = meio + 1;
                         else \sup = meio - 1;
      return (-1);
```



Busca binária em vetor - em memória (assunto de AED 1)

```
// Busca binaria em lista ordenada
int buscaBin (TIPOCHAVE ch, LISTA 1)
      int inf, sup, meio;
      inf = 0;
      sup = 1.nroElem - 1;
      while (inf <= sup)
                                                                        Ex: busca do nr 3
                                                                        [encurtador.com.br/uvBCO0]
                   meio = ((inf + sup) / 2);
                   if(l.A[meio].chave == ch) return(meio); // achou
                   else
                         if(1.A[meio].chave < ch) inf = meio + 1;
                         else sup = meio - 1;
      return (-1);
```



Complexidade: ?

Busca binária em vetor - em memória (assunto de AED 1)

// Busca binaria em lista ordenada

```
Ex: busca do nr 3
[encurtador.com.br/uvBCO0]
```

```
int buscaBin (TIPOCHAVE ch, LISTA 1)
     int inf, sup, meio;
      inf = 0:
      sup = 1.nroElem - 1;
     while (inf <= sup)
                  meio = ((inf + sup) / 2);
                  if(l.A[meio].chave == ch) return(meio); // achou
                  else
                       if(1.A[meio].chave < ch) inf = meio + 1;
                       else sup = meio - 1;
     return (-1);
```



Complexidade: O(lg n) – n elementos no vetor

/* busca registro com chave k em um arquivo com b blocos */

buscaBinaria (k) {

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
			1			
	Acosta, Marc		L			
block 2	Adams Island	1				
DIOCK 2	Adams, John	ļ				
	Adams, Robin		-		L	\vdash
	Akers, Jan		<u>.</u>			
	Akers, Jan					
block 3	Alexander, Ed					
brour.	Alfred, Bob					
	Pellou, Lou		1			
	Allen, Sam	F				
	Thomas was a					
block 4	Allen, Troy					
	Anders, Keith					
			1			
	Anderson, Rob					
block 5	Anderson, Zach					
	Angeli, Joe					
			i .			
	Archer, Sue					
block 6	Amold, Mack					
	Arnold, Steven		Ļ			
			<u>:</u>			
	Atkins, Timothy					
			:			
			•			
olock n -1	Wong, James					
	Wood, Donald					
			:			
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
			:			
	Zimmer, Byron					
						18

/* busca registro com chave k em um arquivo com b blocos */

buscaBinaria (k) {

inf ← primeiro bloco; /* está no cabeçalho */

sup ← ultimo bloco; /* está no cabeçalho */

enquanto (inf <= sup){

 $m \leftarrow floor((inf+sup)/2);$

lê bloco m do disco para o buffer (read_disk(m))

NAME Aaron, Ed Abbott, Diane Acosta, Marc Adams, John Adams, Robin Akers, Jan

block 1

block 2

block 3

block 4

block 5

block 6

block n -1

block n

Alexander, Ed Alfred, Bob Allen, Sam

Alien, Troy Anders, Keith Anderson, Rob

Angeli, Joe Archer, Sue

Amold, Mack Amold, Steven

Atkins, Timothy

Wong, James Wood, Donald Woods, Manny

Anderson, Zach

BIRTHDATE

SALARY

SEX

Wright, Pam Wyatt, Charles

Zimmer, Byron

Profa. Ariane Machado Lima

/* busca registro com chave k em um arquivo com b blocos */

inf ← primeiro bloco; /* está no cabeçalho */

sup ← ultimo bloco; /* está no cabeçalho */

enguanto (inf <= sup){

 $m \leftarrow floor((inf+sup)/2);$

buscaBinaria (k) {

lê bloco m do disco para o buffer (read_disk(m))

se (k < chave do **primeiro** registro do bloco m) sup \leftarrow m - 1;

senão se (k > chave do **último** registro do bloco m) inf \leftarrow m+1;

senão se (k = chave de **algum** registro do bloco m) retorma TRUE

senão retorna FALSE

Akers, Jan

NAME

Aaron, Ed Abbott, Diane

Acosta, Marc

Adams, John

Adams, Robin

Alexander, Ed

BIRTHDATE

SALARY

SEX

Alfred, Bob Allen, Sam

block 4 Alien, Troy Anders, Keith

block 1

block 2

block 3

block n

Anderson, Rob block 5 Anderson, Zach

Archer, Sue

block 6 Amold, Mack Amold, Steven

Angeli, Joe

Atkins, Timothy

Wong, James

Wood, Donald Woods, Manny

Wright, Pam Wyatt, Charles

Zimmer, Byron

/* busca registro com chave k em um arquivo com b blocos */

buscaBinaria (k) {

inf ← primeiro bloco; /* está no cabeçalho */

sup ← ultimo bloco; /* está no cabeçalho */

enguanto (inf <= sup){

 $m \leftarrow floor((inf+sup)/2);$

lê bloco m do disco para o buffer

se (k < chave do **primeiro** registro do bloco m) sup \leftarrow m - 1;

senão se (k > chave do **último** registro do bloco m) inf \leftarrow m+1;

senão se (k = chave de **algum** registro do bloco m) retorma TRUE

senão retorna FALSE

Complexidade: ?

Adams, Robin Akers, Jan

block 1

block 2

block n

block 3 Alexander, Ed Alfred, Bob

NAME

Aaron, Ed Abbott, Diane

Acosta, Marc

Adams, John

BIRTHDATE

SALARY

SEX

Allen, Sam block 4 Alien, Troy

Anders, Keith

Angeli, Joe

Anderson, Rob block 5 Anderson, Zach

Archer, Sue block 6 Amold, Mack

Amold, Steven Atkins, Timothy

Wong, James

Woods, Manny

Wood, Donald

Wright, Pam Wyatt, Charles

Zimmer, Byron

/* busca registro com chave k em um arquivo com b blocos */

buscaBinaria (k) {

inf ← primeiro bloco; /* está no cabeçalho */

sup ← ultimo bloco; /* está no cabeçalho */

enguanto (inf <= sup){

 $m \leftarrow floor((inf+sup)/2)$:

lê bloco m do disco para o buffer

se (k < chave do **primeiro** registro do bloco m) sup \leftarrow m - 1;

senão se (k > chave do último registro do bloco m) inf \leftarrow m+1;

senão se (k = chave de algum registro do bloco m) retorma TRUE Procura em memória (sequencial),

senão retorna FALSE Confere bit de validade

Complexidade: O(lg b)

block 2

block 1

block 3

block 4

block 5

block 6

block n -1

block n

Adams, Robin Akers, Jan

Acosta, Marc

Adams, John

NAME

Aaron, Ed Abbott, Diane

Alexander, Ed Alfred, Bob

Allen, Sam

Alien, Troy Anders, Keith Anderson, Rob BIRTHDATE

SALARY

SEX

Anderson, Zach Angeli, Joe

Archer, Sue Amold, Mack

Amold, Steven Atkins, Timothy

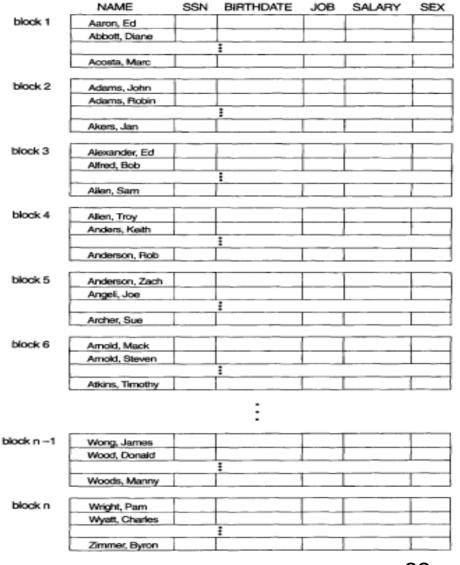
Wong, James Wood, Donald

Woods, Manny

Wright, Pam Wyatt, Charles

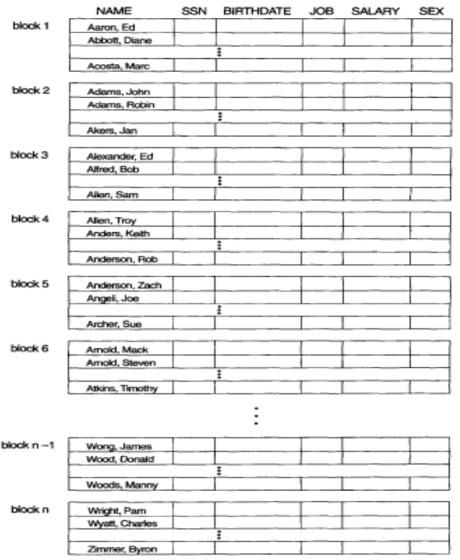
Zimmer, Byron

Inserção: assumindo que já achou o bloco





- Inserção: assumindo que já achou o bloco
 - Se tiver espaço no bloco (algum registro com bit de validade = false) é inserir ordenado no bloco (O(1) seeks)
 - Se não, tem que abrir espaço para o registro (deslocar todos os registros com chave maior para frente) : O(b)
 - Trazer o respectivo bloco para o buffer
 - Deslocar os registros (o último registro vai para o próximo bloco) (O(b) no pior caso)
 - Salvar o bloco no disco
 - Se no último bloco não caber mais um registro, tem que realocar o arquivo todo em outro lugar (O(b) de novo)





- Inserção: assumindo que já achou o bloco
 - Se tiver espaço no bloco (algum registro com bit de validade = false) é inserir ordenado no bloco (O(1) seeks)
 - (deslocar todos os registros com chave maior para frente) : O(b)
 - Trazer o respectivo bloco para o buffer

Se não, tem que abrir espaço para o registro

- Deslocar os registros (o último registro vai para o próximo bloco) (O(b) no pior caso)
- Salvar o bloco no disco
- Se no último bloco não caber mais um registro, tem que realocar o arquivo todo em outro lugar (O(b) de novo)

Abbott, Diane Acosta, Marc Adams, John Adams, Robin Akers, Jan Alexander, Ed Alfred, Bob Allen, Sam Alien, Troy Anders, Keith Anderson, Rob Anderson, Zach Angeli, Joe Archer, Sue Amold, Mack Amold, Steven Atkins, Timothy Wong, James Wood, Donald Woods, Manny

BIRTHDATE

SALARY

NAME

Aaron, Ed

block 1

block 2

block 3

block 4

block 5

block 6

block n -1

block n

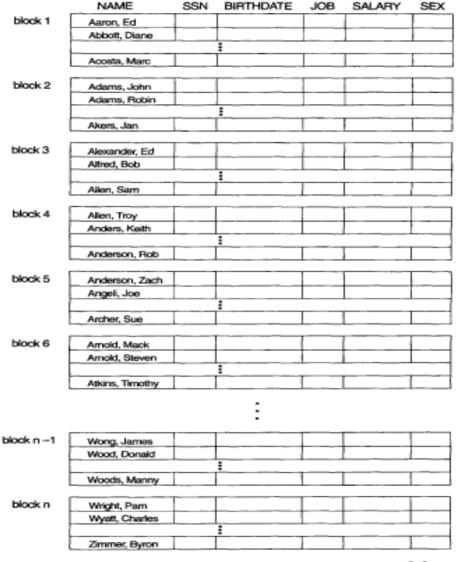
Wright, Pam

Wyatt, Charles

Zimmer, Byron

Escrevam um código para isso!

 Exclusão: assumindo que já achou o bloco



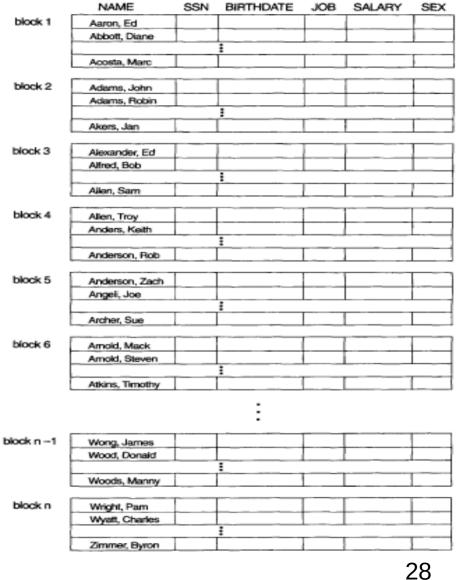


- **Exclusão**: assumindo que já achou o bloco
 - O(1) se usar bit de validade
 - O(b) caso contrário

block 1	Aaron, Ed		
	Abbott, Diane		
		i	
	Acosta, Marc		
block 2	Adams, John		
	Adams, Robin		
	Akers, Jan		
block 3	Alexander, Ed		
	Alfred, Bob		
	Allen, Sam		
block 4	Allen, Troy		
	Anders, Keith		
		_ i	
	Anderson, Rob		
block 5	Anderson, Zach		
	Angeli, Joe		
	Archer, Sue		
block 6			7
DIOCK 6	Amold, Mack		
	Arnold, Steven		
	Address Townston	-	
	Atkins, Timothy		
		:	
block n -1	Wong, James		
	Wood, Donald		
	L	<u>.</u>	
	Woods, Manny		
block n	Wright, Pam		
	Wyatt, Charles		
	-	<u> </u>	
	Zimmer, Byron		
			27



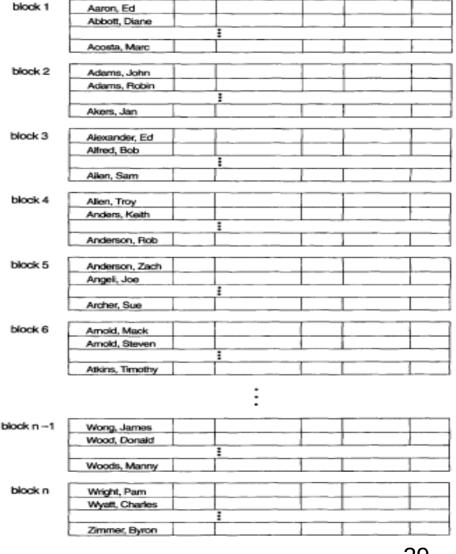
Modificação:





Modificação:

- Ok se for em um campo comum (não chave) - O(1)
- Caro se:
 - a atualização alterar o tamanho do registro (no caso de tamanho variável) - O(b)
 - a atualização for na chave: uma remoção e uma inserção - O(b)



BIRTHDATE

NAME

SALARY

SEX

Complexidades (sempre em termos de nr de seeks...)

	Sequencial			
	Não-Ordenado	Ordenado		
Busca	O(b)	O(lg b)		
Inserção**	O(1) se tiver espaço no final, O(b) c.c.	O(1) se tiver espaço no bloco, O(b) c.c.		
Remoção* , **	O(1)	O(1)		
Leitura ordenada	ω (b) (depende do alg de ord. externa)	O(b), O(1) se fizer a leitura toda de uma vez		
Mínimo/máximo	O(b)	O(1)		
Modificação**	O(1)	O(b) se no campo chave, O(1) c.c.		

^{*} considerando uso de bit de validade



^{**} considerando que já se sabe a localização do registro (busca já realizada)

Aulas passadas

Algoritmos e estruturas de dados para lida com memória secundária

- Organização interna de arquivos
- Acesso à memória secundária (por blocos seeks)
- Tipos de alocação de arquivos na memória secundária:
 - Sequencial (ordenado e não ordenado)
 - Ligada
 - Indexada
 - Árvores-B
 - Hashing (veremos também hashing em memória principal)
- Algoritmos de processamento cossequencial e ordenação em disco



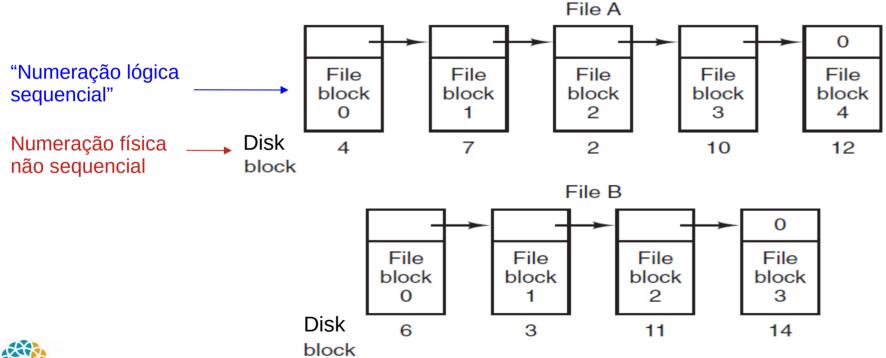
Nosso conteúdo

- Organização interna de arquivos
- Acesso à memória secundária (por blocos seeks)
- Tipos de alocação de arquivos na memória secundária:
 - Sequencial (ordenado e não ordenado)
 - Ligada
 - Indexada
 - Árvores-B

- Vamos sempre assumir arquivo ordenado pelo campo chave a partir daqui
- Hashing (veremos também hashing em memória principal)
- Algoritmos de processamento cossequencial e ordenação em disco

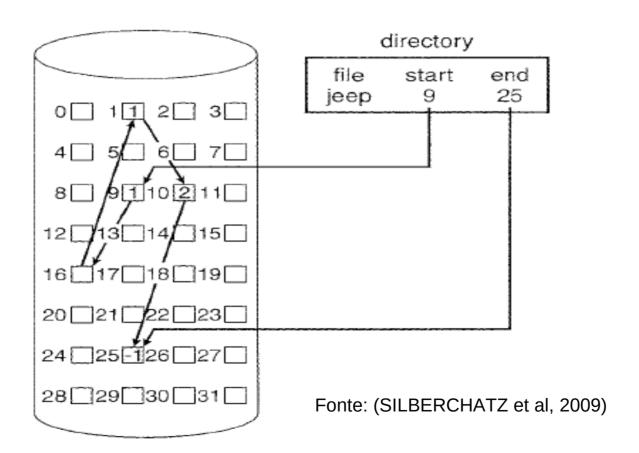
EACH USP

Cada arquivo é uma lista ligada de blocos





Fonte: (TANEMBAUM, 2015)





- Uso de espaço:
 - Vantagem:
 - Desvantagem:
- Leitura sequencial:
- Leitura aleatória / Busca:
- Inserção:
- Remoção:



- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem:
- Leitura sequencial:
- Leitura aleatória / Busca:
- Inserção:
- Remoção:



- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem: perda de espaço com os ponteiros
- Leitura sequencial:
- Leitura aleatória / Busca:
- Inserção:
- Remoção:



- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem: perda de espaço com os ponteiros
- Leitura sequencial: O(b) (mesmo que em uma transação)
- Leitura aleatória / Busca:
- Inserção:
- Remoção:



- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem: perda de espaço com os ponteiros
- Leitura sequencial: O(b) (mesmo que em uma transação)
- Leitura aleatória / Busca: O(b)
- Inserção:
- Remoção:



- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem: perda de espaço com os ponteiros
- Leitura sequencial: O(b) (mesmo que em uma transação)
- Leitura aleatória / Busca: O(b)
- **Inserção:** O(1) assumindo que sei onde inserir
- Remoção:



- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem: perda de espaço com os ponteiros
- Leitura sequencial: O(b) (mesmo que em uma transação)
- Leitura aleatória / Busca: O(b)
- Inserção: O(1) assumindo que sei onde inserir
- Remoção: O(1) para remoção de fato (não apenas ressetar bit de validade) assumindo que sei onde remover, senão + O(b) para buscar

Modificação:

- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem: perda de espaço com os ponteiros
- Leitura sequencial: O(b) (mesmo que em uma transação)
- Leitura aleatória / Busca: O(b)
- **Inserção**: O(1) assumindo que sei onde inserir
- Remoção: O(1) para remoção de fato (não apenas ressetar bit de validade) assumindo que sei onde remover, senão + O(b) para buscar
 - **Modificação**: O(b) se for no campo chave (tem que achar onde inserir), O(1) c.c.

EACH USP

- Uso de espaço:
 - Vantagem: resolve o problema de fragmentação externa
 - Desvantagem: perda de espaço com os ponteiros
- Leitura sequencial: O(b) (mesmo que em uma transação)
- Leitura aleatória / Busca: O(b)

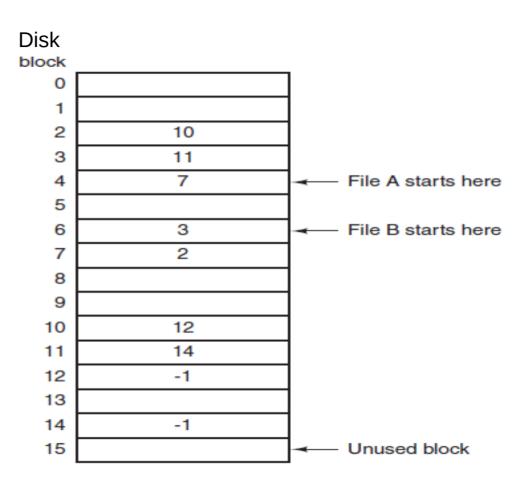


- **Inserção:** O(1) assumindo que sei onde inserir
- Remoção: O(1) para remoção de fato (não apenas ressetar bit de validade) assumindo que sei onde remover, senão + O(b) para buscar
 - **Modificação**: O(b) se for no campo chave (tem que achar onde inserir), O(1) c.c.

EACH USP

Alocação por listas ligadas com uso de uma File Allocation Table (FAT) em memória

• t[i] armazena o próximo bloco do bloco





FAT

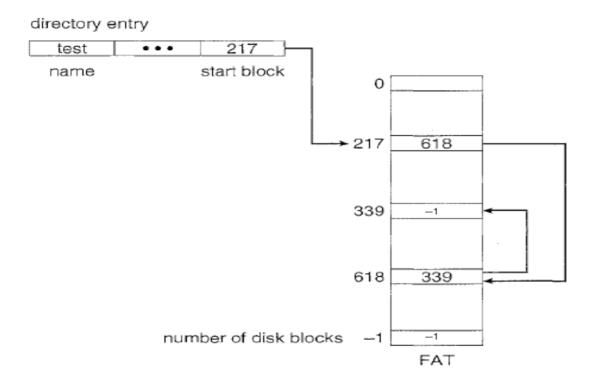


Figure 11.7 File-allocation table.

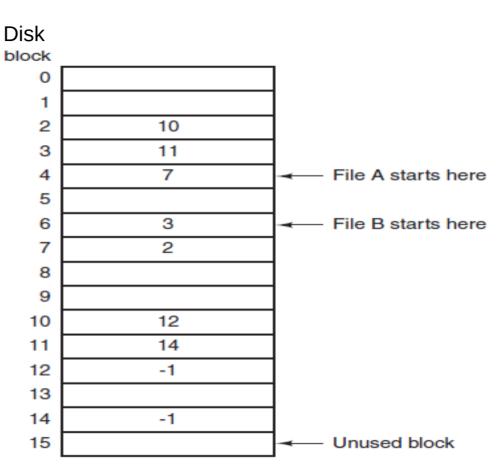


Fonte: (SILBERCHATZ et al, 2009)

Alocação por listas ligadas com uso de uma File Allocation Table (FAT) em memória

- t[i] armazena o próximo bloco do bloco
- Vantagens:
 - Economiza espaço nos blocos de dados (que só conterão dados e não ponteiros) → preciso de menos blocos → b menor impacta nas velocidades dependentes de b
 - Para uma leitura aleatória (dado um deslocamento em relação ao início do arquivo), o encadeamento (para achar o bloco certo) é seguido apenas sobre a

tabela (que está toda em memória) \rightarrow O(1)



Fonte: (TANEMBAUM, 2015)

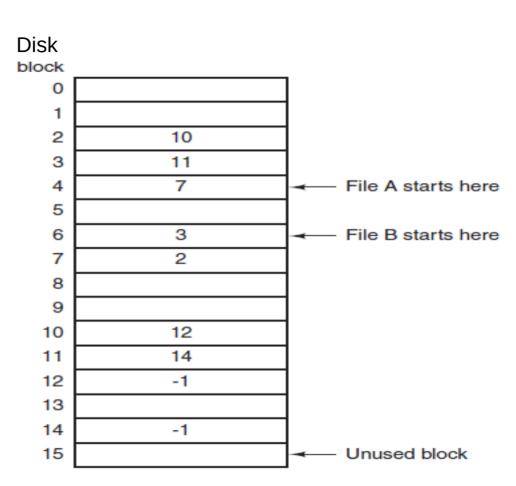
Daria para implementar uma busca binária???

Alocação por listas ligadas com uso de uma File Allocation Table (FAT) em memória

- t[i] armazena o próximo bloco do bloco
- Desvantagens:
 - Não escalável para grandes discos

Ex: para um disco de 1TB e blocos de 1KB, a tabela ocuparia 3GB de memória

 Sistema de arquivos FAT32, por ex, impõe tamanho máximo de disco de 2TB





Complexidades (sempre em termos de nr de seeks...)

	Sequencial		Ligada
	Não-Ordenado	Ordenado	Ordenado
Busca	O(b)	O(lg b)	O(b), O(lg b) só se usar FAT (discos pequenos)
Inserção**	O(1) se tiver espaço no final, O(b) c.c.	O(1) se tiver espaço no bloco, O(b) c.c.	O(1)
Remoção* · **	O(1)	O(1)	O(1)
Leitura ordenada	ω (b) (depende do alg de ord. externa)	O(b), O(1) se fizer a leitura toda de uma vez	O(b)
Mínimo/máximo	O(b)	O(1)	O(1)
Modificação**	O(1)	O(b) se no campo chave, O(1) c.c.	O(1)
* considerando uso de bit de validade			
** considerando que já se sabe a localização do registro (busca já realizada)			



Outra alternativa?

- Lista ligada aproveita espaço (resolve fragmentação externa), mas leitura aleatória fica horrível (sem tabela de alocação)
- Tabela de alocação acelera a leitura aleatória mas gasta muita memória
- Como diminuir esse último problema?



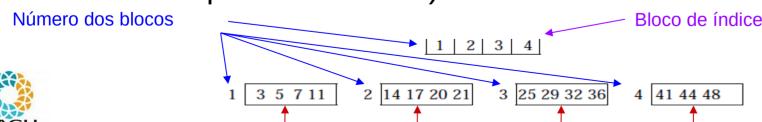
Outra alternativa?

- Lista ligada aproveita espaço (resolve fragmentação externa), mas leitura aleatória fica horrível
- Tabela de alocação acelera a leitura aleatória quando quero o i-ésimo bloco, mas gasta muita memória
- Como diminuir esse último problema?
- Por que manter em memória as informações de arquivos que não foram abertos? Que tal "uma tabela" por arquivo?



Alocação indexada

- Um ou mais blocos de índices contém ponteiros para os blocos de fato
- Blocos de índices são como uma tabela de alocação específica daquele arquivo
 - i-ésima entrada do primeiro bloco de índice contém o número do i-ésimo bloco de dado do arquivo
- Blocos de índice carregados na memória sob demanda (assim como arquivos de dados)



Profa Ariane Machado I ma

Alocação indexada

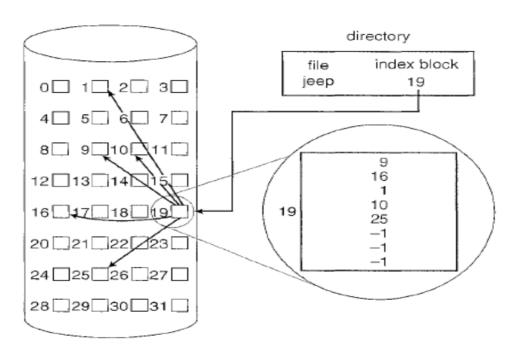


Figure 11.8 Indexed allocation of disk space.



Fonte: (SILBERCHATZ et al, 2009)

Cabeçalhos de arquivo do tipo I-nodes (index-nodes): visão geral

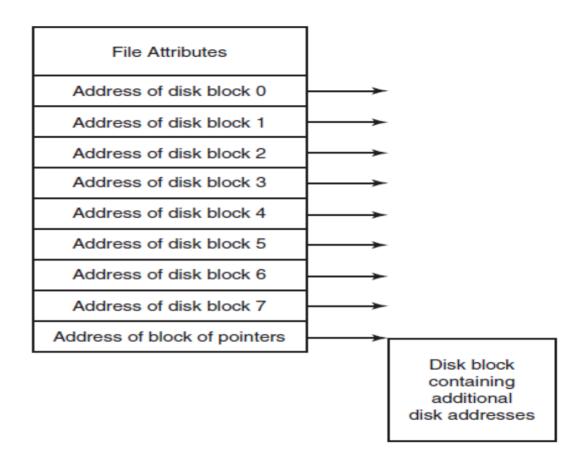
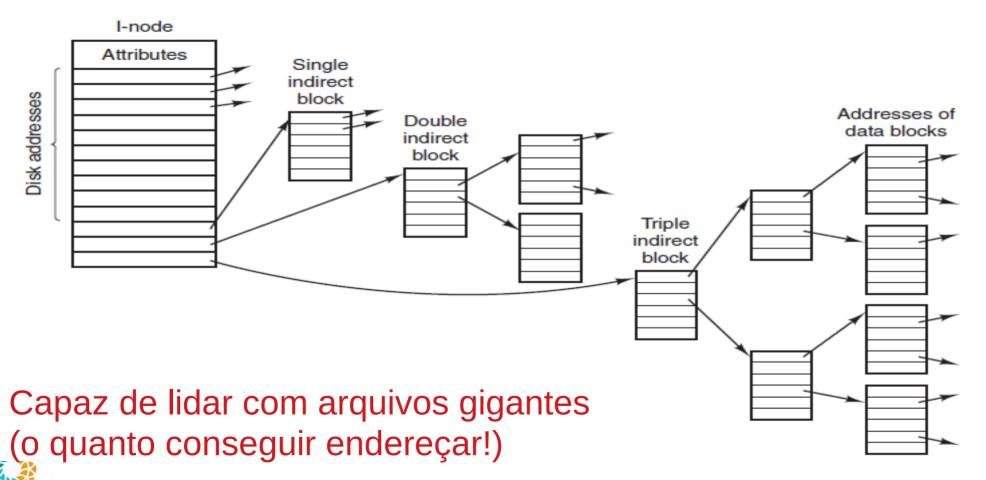


Figure 4-13. An example i-node.



I-nodes (index-nodes) do UNIX/LINUX:



Leitura complementar

 Mais detalhes sobre o sistema de arquivos do Linux e Windows: cap 4, 10 e 11 do livro do Tanenbaum (referência no último slide)



Referências

- ELMASRI, R.; NAVATHE, S. B. Fundamentals of Database Systems. 4 ed. Ed. Pearson-Addison Wesley. Cap 13 (até a seção 13.7).
- GOODRICH et al, Data Structures and Algorithms in C++. Ed. John Wiley & Sons, Inc. 2nd ed. 2011. Seção 14.2
- RAMAKRISHNAN, R.; GEHRKE, J. **Data Management Systems** 3ed. Ed McGraw Hill. 2003. cap 8 e 9.
- SILBERSCHATZ, A.; GALVIN, p. B.; GAGNE, G. Operating Systems Concepts. 8
 ed. Ed. John Wiley \$ Sons. 2009. Cap 11
- TANEMBAUM, A. S. & BOS, H. Modern Operating Systems. Pearson, 4th ed. 2015. Cap 4

