

ACH2024

Aula 15

Organização de arquivos:
Alocação sequencial ordenada
alocação ligada

Profa. Ariane Machado Lima

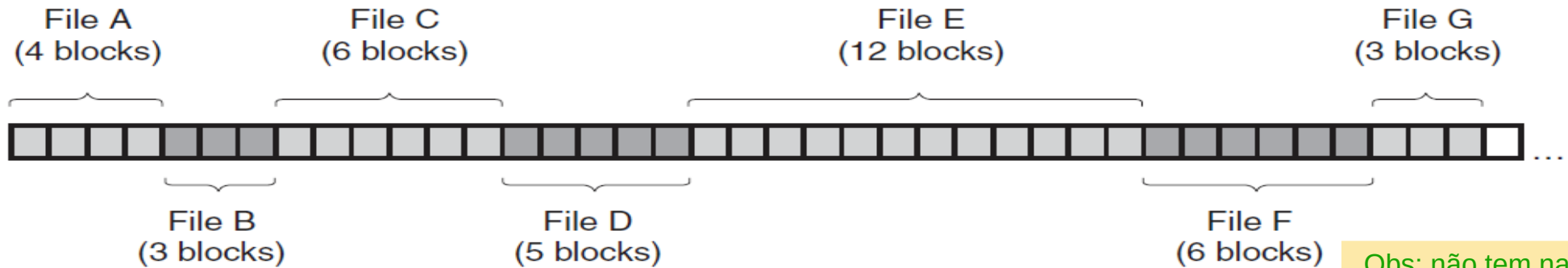
Aulas passadas

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)



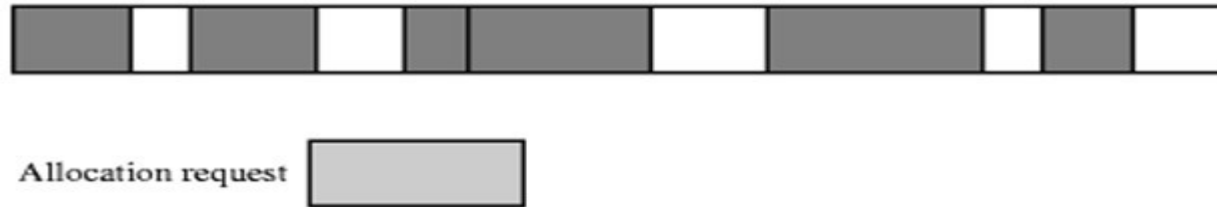
- E como os registros são organizados pelos blocos?
 - Não ordenados (**sequencial não ordenado** – heap files)
 - Ordenados por um campo chave (**sequencial ordenado** - sorted files)

Obs: não tem nada a ver com a estrutura de dados "heap"

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

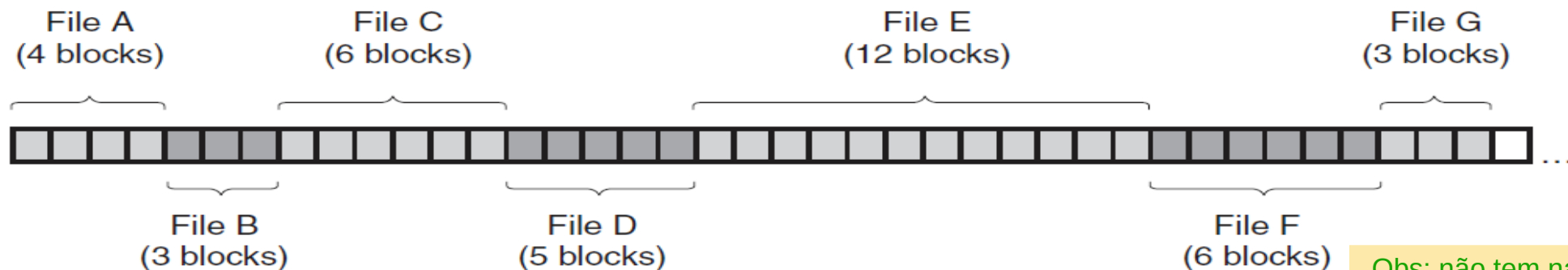


Alocação sequencial (não ordenado)

	Sequencial		
	Não-Ordenado		
Busca	$O(b)$		
Inserção**	$O(1)$ se tiver espaço no final; $O(b)$ c.c.		
Remoção* , **	$O(1)$		
Leitura ordenada	$\omega(b)$ (depende do alg de ord. externa)		
Mínimo/máximo	$O(b)$		
Modificação**	$O(1)$		
* considerando uso de bit de validade			
** considerando que já se sabe a localização do registro (busca já realizada)			

Aula de hoje - Alocação sequencial

- Blocos alocados sequencialmente no disco (pelos cilindros)



- E como os registros são organizados pelos blocos?
 - Não ordenados (**sequencial não ordenado** – heap files)
 - Ordenados por um campo chave (**sequencial ordenado** - sorted files)

Obs: não tem nada a ver com a estrutura de dados "heap"

Alocação sequencial (ordenado)

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

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	Atkins, Timothy					
block n - 1	Wong, James					
	Wood, Donald					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **Leitura ordenada**

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

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 sequencialmente dispostos pelo(s) cilindros)

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n-1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **Mínimo** (menor chave) / **Máximo** (maior chave):

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **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

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n-1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **Busca:**

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

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

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

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

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

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

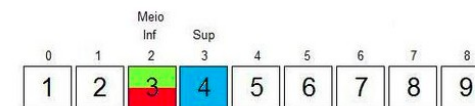
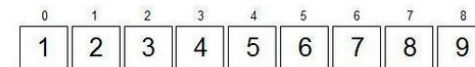
Busca binária em vetor - em memória (assunto de AED 1)

```
// Busca binaria em lista ordenada
int buscaBin(TIPOCHAVE ch, LISTA l)
{
```

```
    int inf, sup, meio;
    inf = 0;
    sup = l.nroElem - 1;
    while(inf <= sup)
    {
```

```
        meio = ((inf + sup) / 2);
        if(l.A[meio].chave == ch) return(meio); // achou
        else
        {
            if(l.A[meio].chave < ch) inf = meio + 1;
            else sup = meio - 1;
        }
    }
```

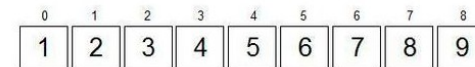
```
    return(-1);
}
```



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

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

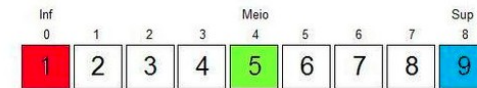
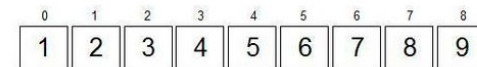
```
// Busca binaria em lista ordenada
int buscaBin(TIPOCHAVE ch, LISTA l)
{
    int inf, sup, meio;
    inf = 0;
    sup = l.nroElem - 1;
    while(inf <= sup)
    {
        meio = ((inf + sup) / 2);
        if(l.A[meio].chave == ch) return(meio); // achou
        else
        {
            if(l.A[meio].chave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



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

Complexidade: ?

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



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

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

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

Alocação sequencial (ordenado)

/* 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					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

/ 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 ← floor((inf+sup)/2);

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

}

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

/ 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 ← floor((inf+sup)/2);

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

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

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

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

 senão retorna FALSE

}

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n-1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

/ 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 ← floor((inf+sup)/2);

lê bloco m do disco para o buffer

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

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

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

senão retorna FALSE

}

Complexidade: ?

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n-1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

/ 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 ← floor((inf+sup)/2);

lê bloco m do disco para o buffer

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

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

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

Procura em memória (sequencial),

senão retorna FALSE

Confere bit de validade

}

Complexidade: $O(\lg b)$

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n-1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **Inserção:** assumindo que já achou o bloco

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **Inserção:** assumindo que já achou o bloco
 - Se tiver espaço no bloco (algun 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)

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n-1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **Inserção:** assumindo que já achou o bloco
 - Se tiver espaço no bloco (algun 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)

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Escrevam um código para isso!

Alocação sequencial (ordenado)

- **Exclusão:** assumindo que já achou o bloco

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

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

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **Modificação:**

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n - 1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

Alocação sequencial (ordenado)

- **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 - $O(b)$

	NAME	SSN	BIRTHDATE	JOB	SALARY	SEX
block 1	Aaron, Ed					
	Abbott, Diane					
	⋮					
	Acosta, Marc					
block 2	Adams, John					
	Adams, Robin					
	⋮					
	Akers, Jan					
block 3	Alexander, Ed					
	Alfred, Bob					
	⋮					
	Allen, Sam					
block 4	Allen, Troy					
	Anders, Keith					
	⋮					
	Anderson, Rob					
block 5	Anderson, Zach					
	Angel, Joe					
	⋮					
	Archer, Sue					
block 6	Arnold, Mack					
	Arnold, Steven					
	⋮					
	Atkins, Timothy					
	⋮					
block n-1	Wong, James					
	Wood, Donald					
	⋮					
	Woods, Manny					
block n	Wright, Pam					
	Wyatt, Charles					
	⋮					
	Zimmer, Byron					

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	$\omega(b)$ (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

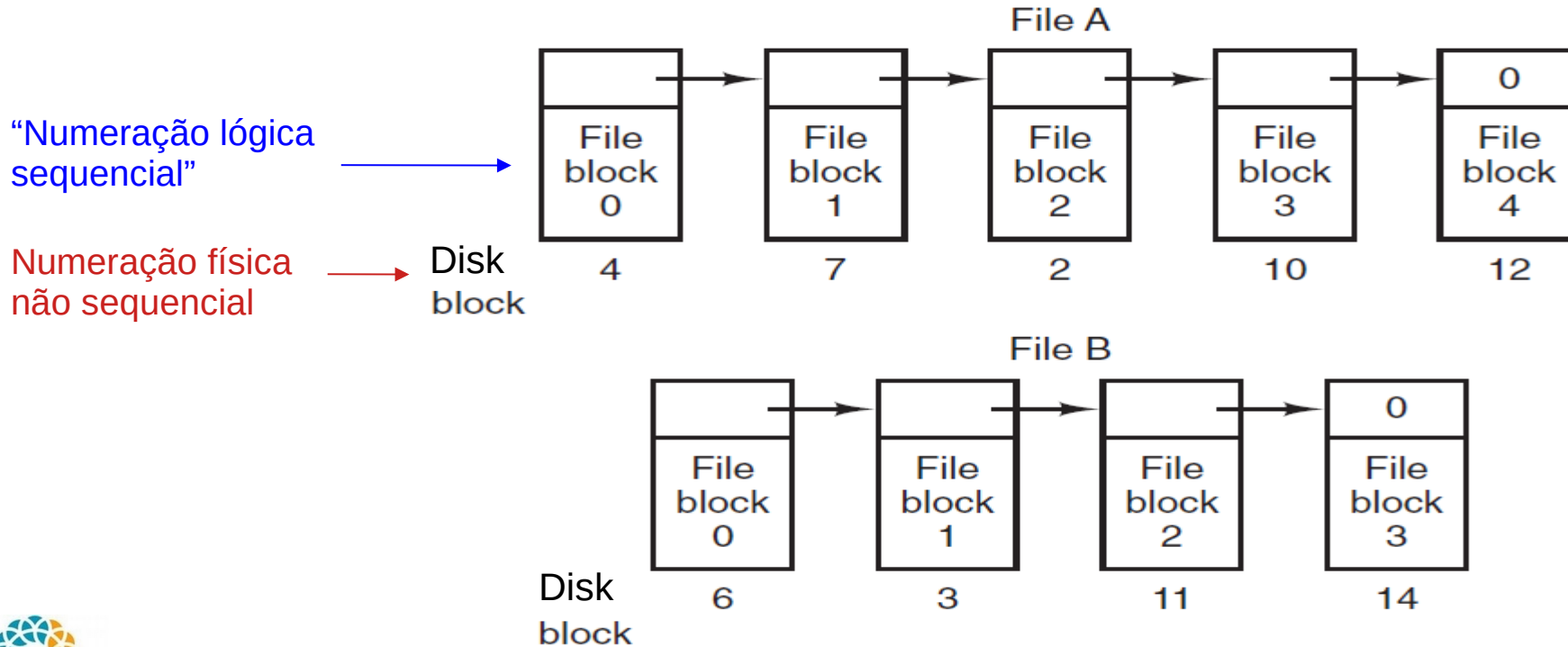
Nosso conteúdo

- 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
 - Hashing (veremos também hashing em memória principal)
- Algoritmos de processamento cossequencial e ordenação em disco

Vamos sempre assumir arquivo ordenado pelo campo chave a partir daqui

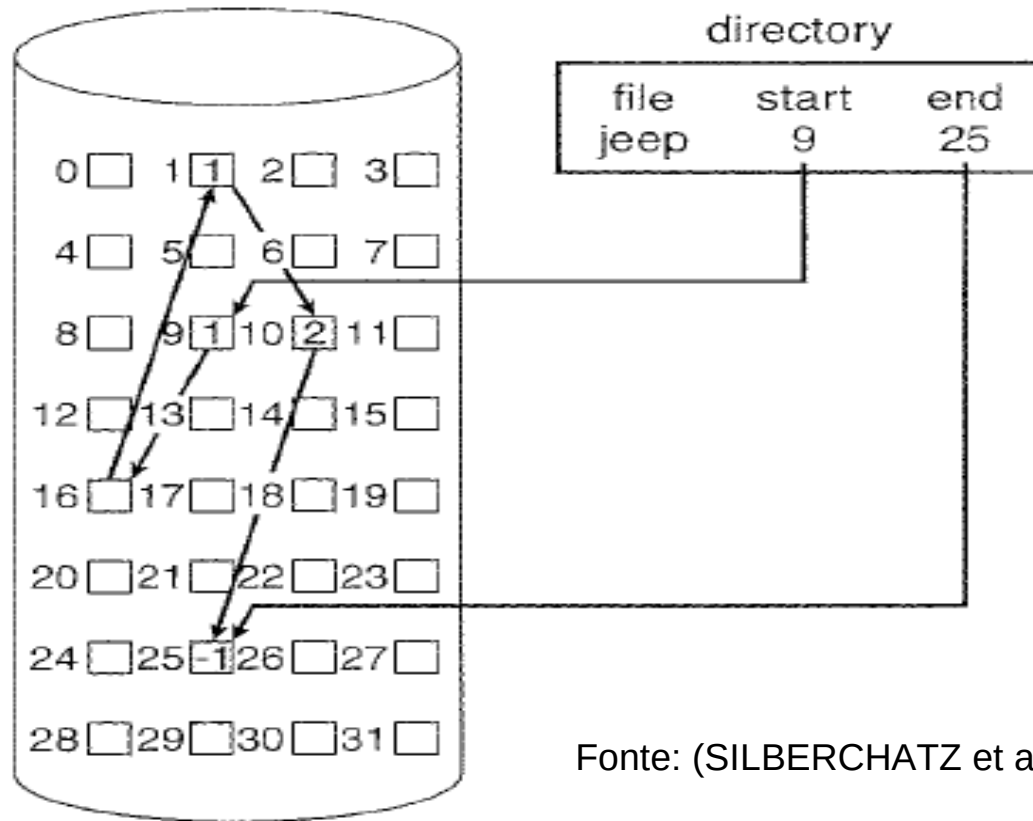
Alocação por listas ligadas

Cada arquivo é uma lista ligada de blocos



Fonte: (TANEMBAUM, 2015)

Alocação por listas ligadas



Fonte: (SILBERCHATZ et al, 2009)

Alocação por listas ligadas

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

Alocação por listas ligadas

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

Alocação por listas ligadas

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

Alocação por listas ligadas

- **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:**

Alocação por listas ligadas

- **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:**

Alocação por listas ligadas

- **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:**

Alocação por listas ligadas


- **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:

Alocação por listas ligadas

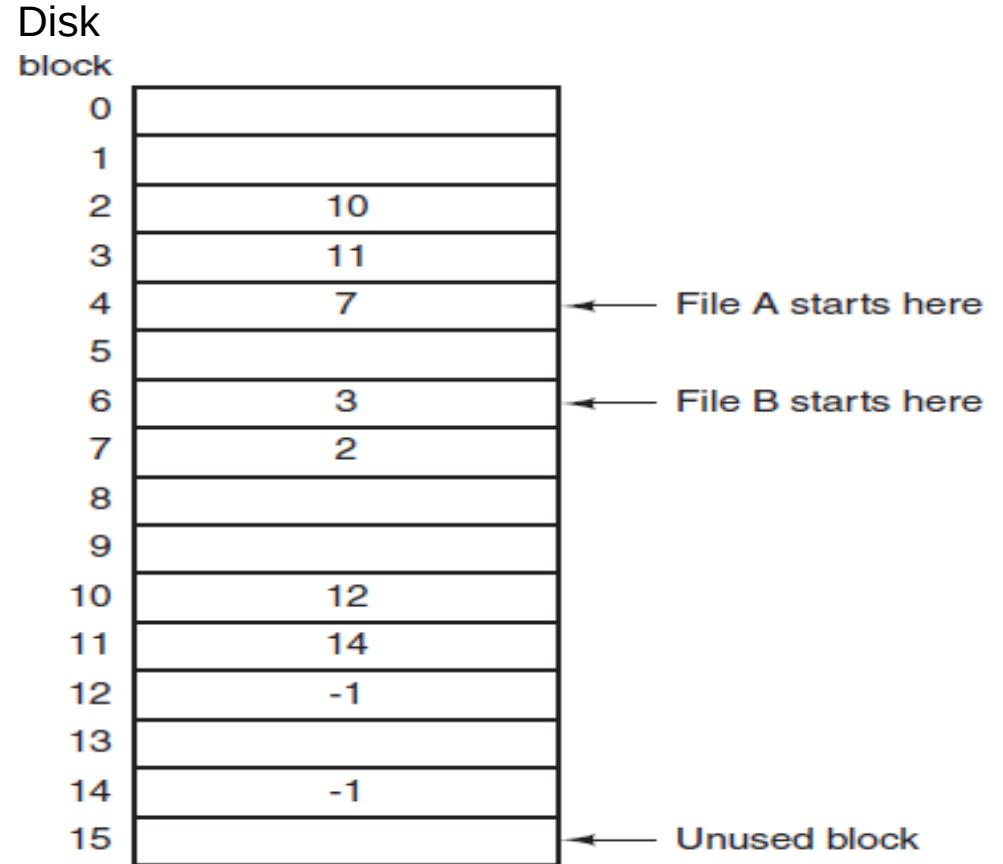
- **Uso de espaço:**
 - **Vantagem:** resolve o problema de fragmentação externa
 - **Desvantagem:** perda de espaço com os ponteiros
- **Leitura sequencial:** $O(b)$ (mesmo que em uma transação)
- **Leitura aleatória / Busca:** $O(b)$
- **Inserção:** $O(1)$ – assumindo que sei onde inserir
- **Remoção:**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

Alocação por listas ligadas

- **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.

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

- $t[i]$ armazena o próximo bloco do bloco



Fonte: (TANEMBAUM, 2015)

FAT

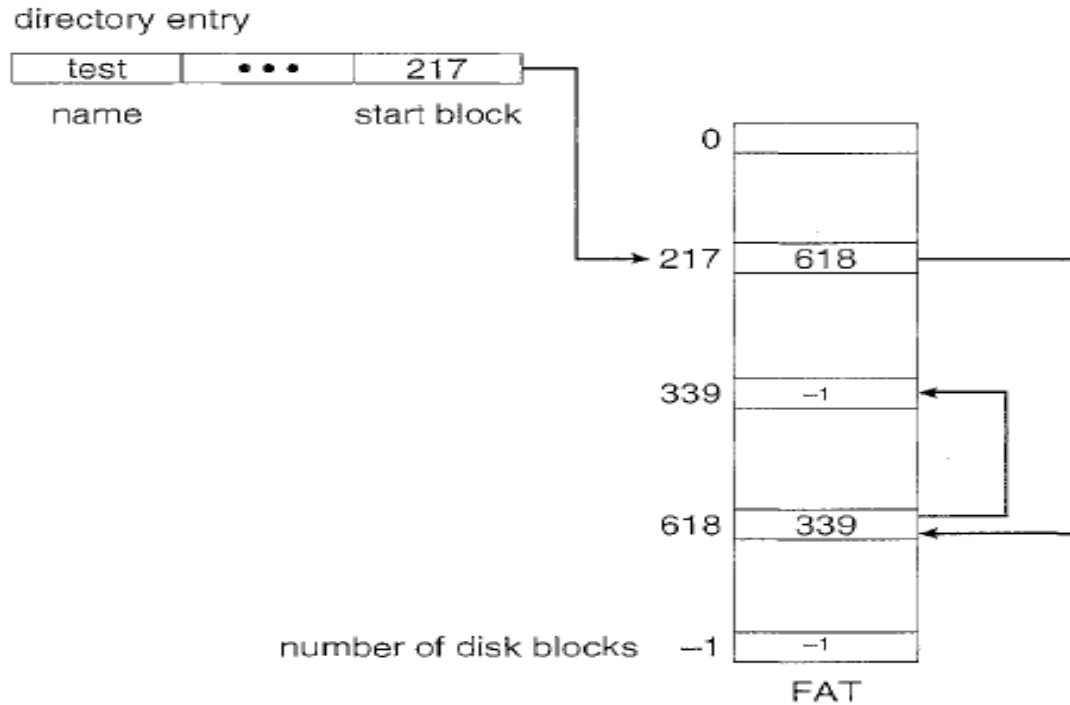
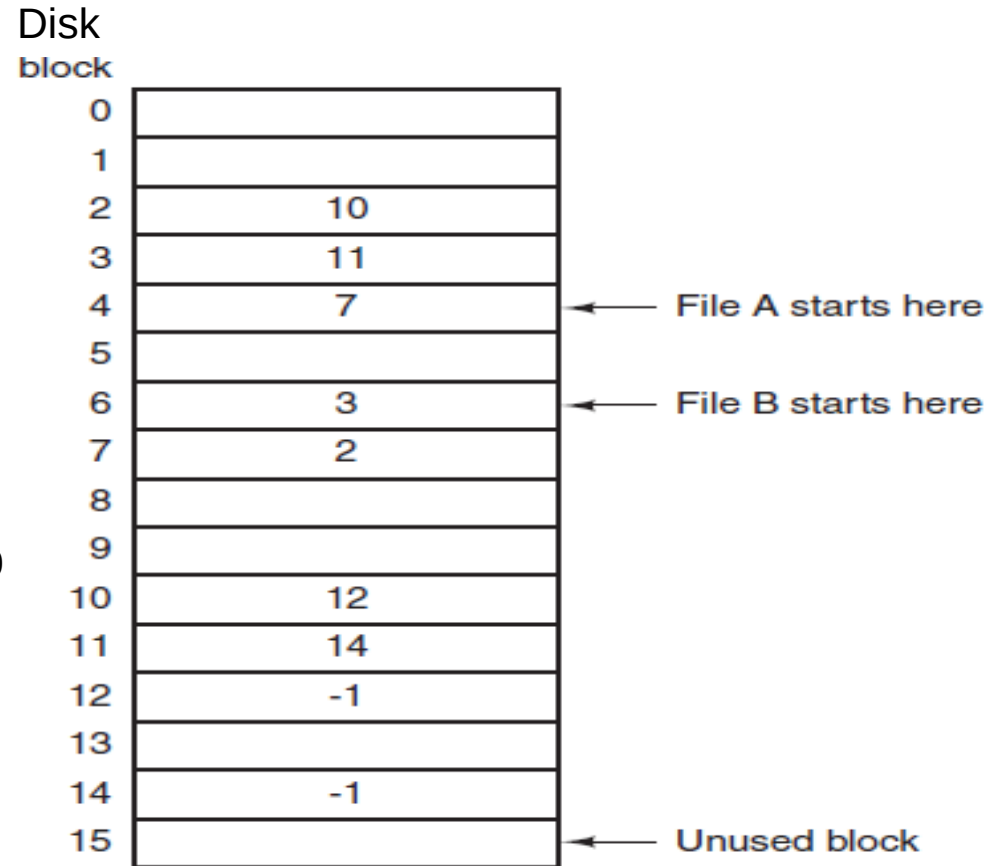


Figure 11.7 File-allocation table.

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) → $O(1)$

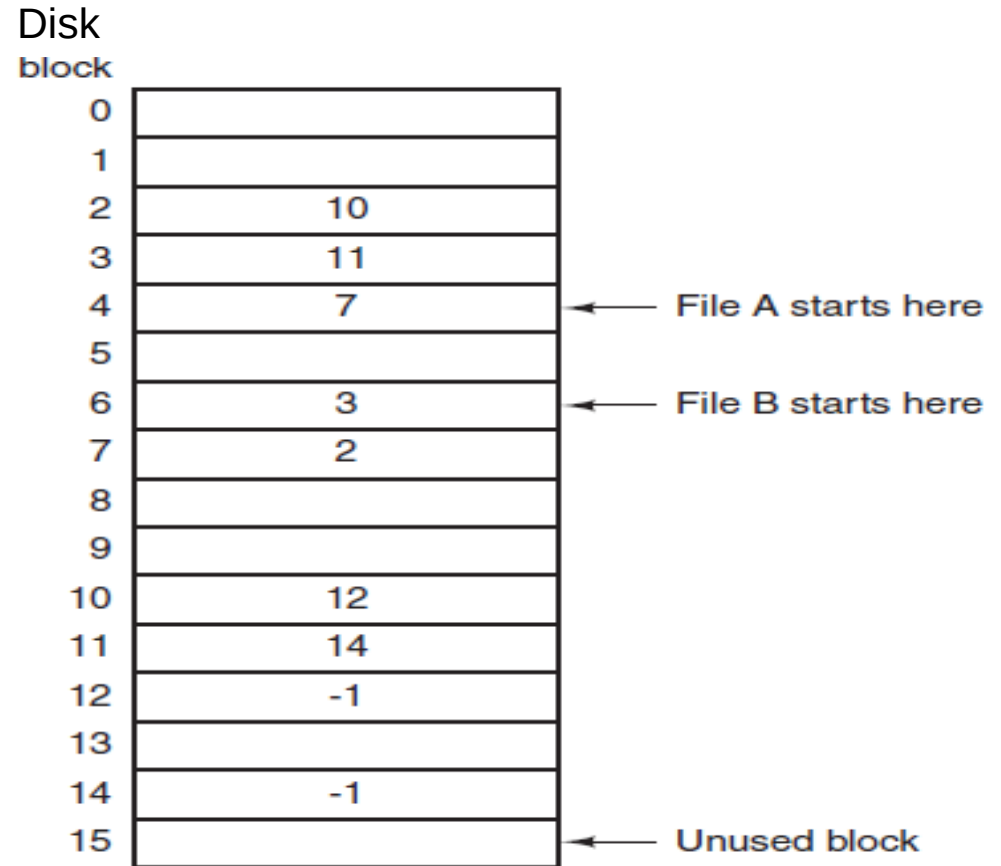


Fonte: (TANEMBAUM, 2015)

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



Fonte: (TANEMBAUM, 2015)

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	$\omega(b)$ (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?

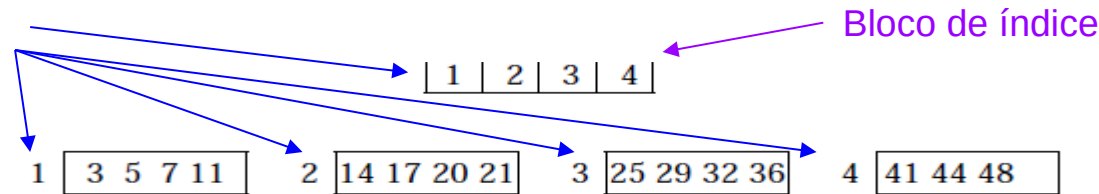
Outra alternativa?

- 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)

Número dos blocos



Blocos de dados contendo os registros (apenas a chave de cada registro aqui representada)

Alocação indexada

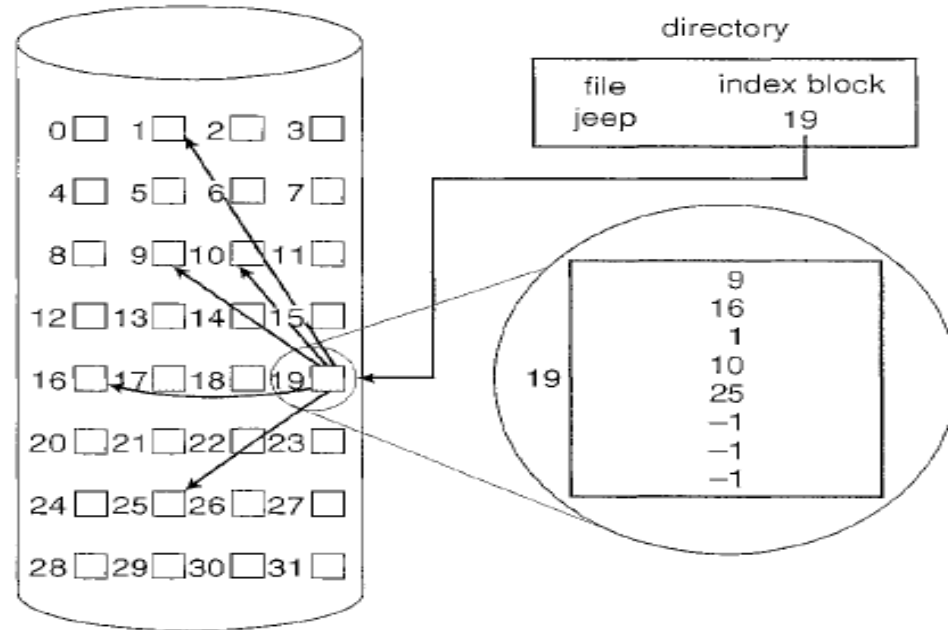


Figure 11.8 Indexed allocation of disk space.

Fonte: (SILBERCHATZ et al, 2009)

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

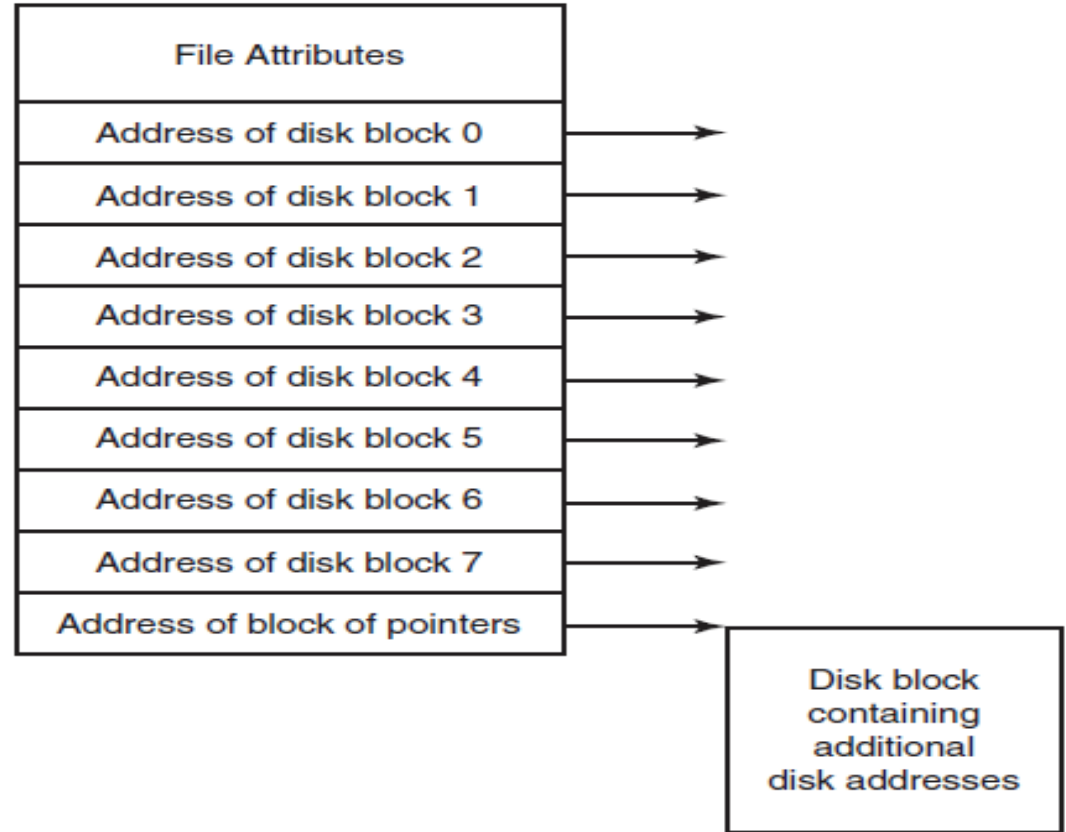
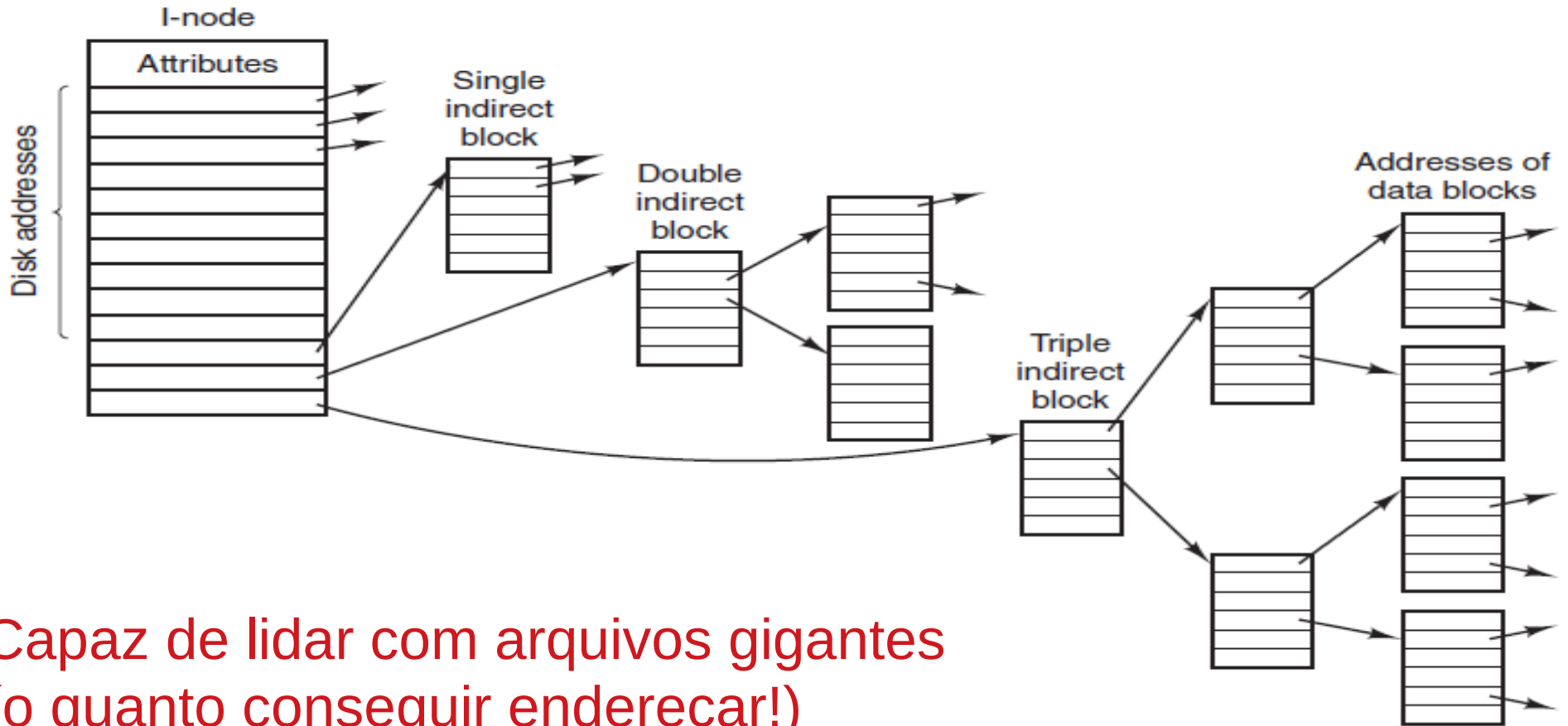


Figure 4-13. An example i-node.

Fonte: (TANEMBAUM, 2015)

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



Capaz de lidar com arquivos gigantes
(o quanto conseguir endereçar!)

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