Programação de Sistemas

Introdução aos Sistemas de ficheiros

Conceitos base (1)

[Def] Um **ficheiro** é um contentor de dados, acedidos sequencialmente ou alteatoriamente.

[Def] Num **sistema de ficheiros** são determinados:

- A organização dos vários ficheiros existentes no meio (disco, memória RAM,…),
- Os mecanismos de localização de um ficheiro no meio.

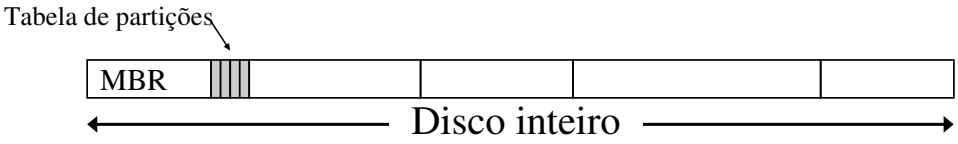
Exemplos de sistemas de ficheiros (✓ a estudar neste capítulo)

- ✓ ext2 e ext3, para Linux (número mágico 0xEF53)
- ✓ iso9660, para CDs
- ✓ nfs, para acesso por rede
- fat, para MSDOS e Windows 95/98 (versões 16 e 32 bits)
- ntfs, para Windows NT/2000/XP/Vista
- swap, em Unix

Conceitos base (2)

[Def] Uma **partição**, por vezes designado volume, é um grupo contínuo de cilindros, vista pelo sistema operativo como um disco lógico.

- Um disco inteiro é dividido em:
  - Sector 0 : MBR-"master boot record", com código para seleccionar a partição de onde será carregado o sistema operativo.
    Nota: No final do MBR existe a tabela das partições no disco.
  - Uma, ou mais, partições (limite máximo de 4 em discos ATA-"Advanced Technology Attachment" , designados por IDE-"Integrated Drive Electronis" pela Western Digital) .



Conceitos base (3)

- No Linux, a gestão de partições (criação, eliminação, listagem) é feita pelo comando
  `/sbin/fdisk dispositivo`
- O comando é interactivo com diversas opções
  - `l` – lista códigos dos sistemas de ficheiros
  - `m` - lista opções
  - `p` – imprime tabela de partição
  - `t` – altera sistema de ficheiros na partição
  - `w` – escreve no disco a tabela de partição
    Nota: as alterações têm efeito só depois de executada esta opção

Conceitos base (4)

Exemplo

```
charlie.ist.utl.pt> /sbin/fdisk -l /dev/hda

Disk /dev/hda: 20.0 GB, 20003880960 bytes
255 heads, 63 sectors/track, 2432 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End      Blocks   Id  System
/dev/hda1             1        13      104391   83  Linux
/dev/hda2            14      1656    13197397+  8e  Linux LVM
/dev/hda3   *      1657      2432     6233220    7  HPFS/NTFS
```

Disco principal

Partição

Conceitos base (5)

- O sistema de ficheiros de um disco tem de resolver vários problemas
  A. Que tipos de ficheiros podem existir?
  B. Como determinar, de forma eficiente, a localização de cada um dos blocos constituintes de um ficheiro?
  C. Que estrutura de directórios é aceite, e como implementá-la?
  D. Como identificar rapidamente o estado (livre ou ocupado) de cada um dos blocos disponíveis?

Estrutura de ficheiros (1)

- Um ficheiro guarda informação de forma persistente em dispositivos de memória de massa (disco, CDROM,…)
- Ficheiros podem ser estruturados em diversas formas:

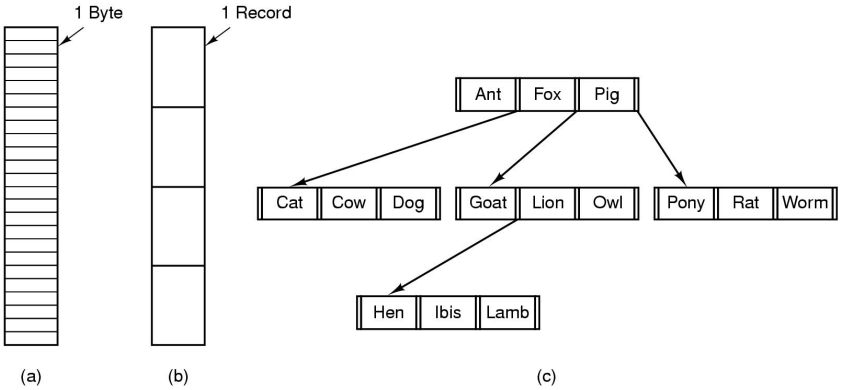


Figura 6-2, Modern Operating Systems

Estrutura de ficheiros (2)

a) Sequência de Bytes
O sistema operativo desconhece estrutura interna dos ficheiros - solução adoptado pelo Linux e Windows.

b) Sequência de registos
Os registos possuem comprimento fixo – solução adoptada pelos computadores antigos de grande porte (ex: registo corresponde a um cartão perfurado de 80 caracteres, ou linha de impressora de 132 caracteres)

c) Árvore de registos
Cada registo pode ter comprimento distinto mas contendo uma chave - solução adoptada nas bases de dados, que facilita o rápido acesso ao registo.

Estrutura de ficheiros (3)

Curiosidade, para avaliação avançada apenas

- Ficheiros divididos pelo seu tipo, indicado pelo primeiro caractere no comando `ls -l`
  - `-` – Regular (texto, programa executável,…)
  - `b` – Dispositivo de blocos
  - `c` – Dispositivo de caracteres
  - `d` – Directório
  - `l` – Ligação simbólica
  - `p` – Tubos (“pipes”)
  - `s` – Socket

Nota: Informação sobre ficheiro obtida pelo comando `stat fich`

Alocação de ficheiros (1)

- Um ficheiro ocupa um, ou mais, blocos. Cada bloco corresponde a um, ou mais, sectores de disco (512B).

  Nota: no Linux, cada bloco ocupa 4KB.

  Existem 4 estratégias de alocação dos blocos de um ficheiro pela partição:

  1. Alocação contínua : usada em CDROM, DVD e “smart-cards”. Para cada ficheiro o directório contém apenas posição do primeiro bloco e o tamanho.
  - Vantagens:
    - simplicidade de implementação
    - leitura de um ficheiro feita numa única operação.
  - Inconvenientes: fragmentação de disco após múltiplas operações de inserção e eliminação de ficheiros.

Alocação de ficheiros (2)

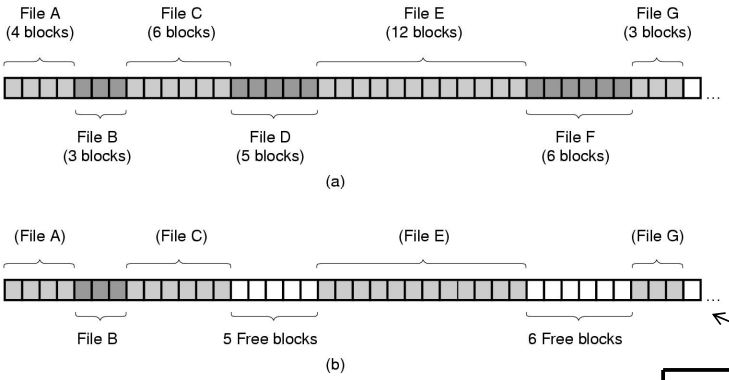


Figura 6-12, Modern Operating Systems

Directório

Ficheiro	Início	Comprimento
A	0	4
B	4	3
C	7	6
E	18	12
G	36	3

Alocação de ficheiros (3)

2. Alocação por lista ligada : em cada bloco é indicada a localização do bloco seguinte.
   Para cada ficheiro o directório contém apenas posição do primeiro bloco e último blocos.

– Vantagens:
  - Evita fragmentação de disco, porque todos os blocos livres podem ser usados independentemente da sua posição

– Inconvenientes:
  - Acesso aleatório mais lento, por em cada acesso ser necessário posicionar-se desde início.
  - Espaço extra ocupado pela localização do bloco seguinte.

Os inconvenientes são resolvidos por transferência das localizações para uma tabela.

Alocação de ficheiros (4)

Ex: alocação por lista de ficheiros A (5 blocos) e B (2 blocos)

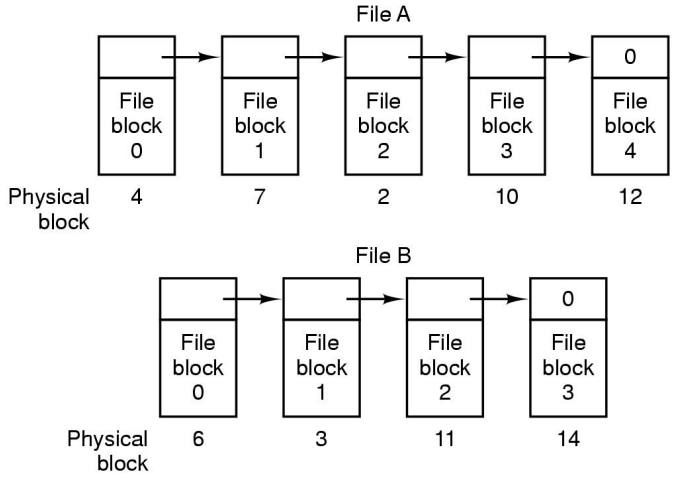


Figura 6-13, Modern Operating Systems

Directório

Ficheiro	Início	Fim
A	4	12
B	6	14

Alocação de ficheiros (5)

3. Alocação por tabela : localizações dos blocos armazenadas num única tabela residente na memória.
   - Em cada índice é indicada a localização do bloco seguinte.
     Ex : ficheiro A, iniciado no bloco 4, prossegue nos blocos 7,2,10 e 12.
   - Último bloco identificado por marcador (ex: -1).

Directório

Ficheiro	Índice início
A	4
B	6

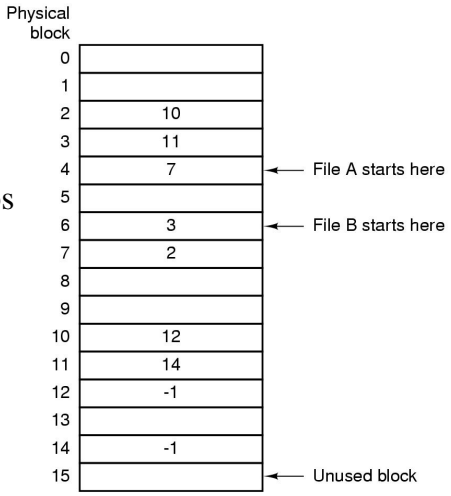


Figura 6-14, Modern Operating Systems

Alocação de ficheiros (6)

4. Alocação por bloco indexado : localizações dos blocos armazenadas num bloco com índices.
   - O directório apenas indica localização do bloco dos índices.
   - Entradas irrelevantes por o ficheiro ser mais curto, indicadas por marcadores (-1).
   - Solução adoptada pelo UNIX

Exemplo: para ficheiros da figura 6-14

Directório

Ficheiro	Bloco índices
A	32
B	47

Bloco 32

4
7
2
10
12
-1
-1

Alocação de ficheiros (6)

Curiosidade, para avaliação avançada apenas

Exemplo : seja um simples ficheiro de teste

```
[rgc@asterix ~]$ cat > test.txt
Ola

[rgc@asterix ~]$ ls -l test.txt
-rw-r--r-- 1 rgc docentes 4 2008-11-26 11:03 test.txt
[rgc@asterix ~]$ du -s test.txt
4       test.txt
```

Dimensão do ficheiro = 4B

Espaço ocupado em disco = 4 KB

Alocação de directorias (1)

- Nas directorias são armazenados atributos diversos:
  - Acessíveis a utilizadores : identificador, dimensão, datas (criação,…)
  - Internas ao sistema operativo : localização em disco, espaço ocupado em disco, …
- Existem 2 estratégias de alocação dos atributos dos ficheiros:
  a) No próprio directório (estratégia adoptada pelo Windows)
  b) Num ficheiro especial, designado por i-node (estratégia adoptada pelo Linux)

Alocação de directorias (2)

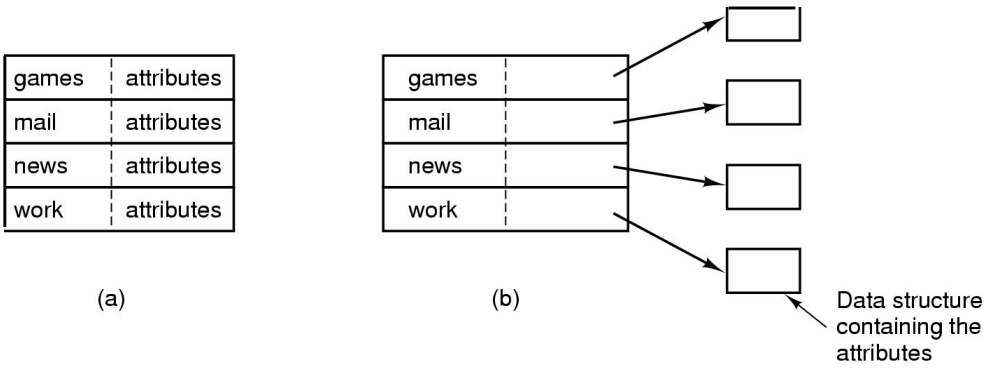


Figura 6-16, Modern Operating Systems

Ficheiros partilhados (1)

- Pode haver conveniência de um ficheiro ser partilhado ("shared") por vários utilizadores.
  - Membros da mesma equipa de projecto.
  - Entrada de página WWW reside noutro directório de trabalho (por exemplo, a entrada `ec-ps` do nó comp.ist.utl.pt está no ficheiro /var/www/html/ec-ps da máquina `comp`, mas reside no ficheiro /home/ec-ps/public_html/index.html).
- Idealmente, o ficheiro reside num directório mas é listado como fazendo parte noutro directório.
- O ficheiro é acedido, se o dono tiver privilégios para tal, através de uma ligação simbólica ("soft link").

Ficheiros partilhados (2)

- A estrutura dos directórios em árvore passa a grafo directo acíclico (DAG-Directed Acyclic Graph)

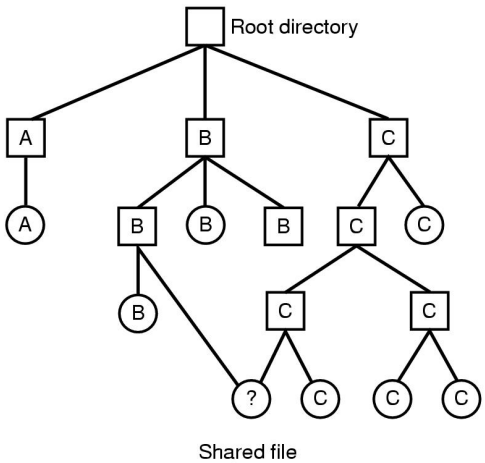


Figura 6-18, Modern Operating Systems

Ficheiros partilhados (3)

No Linux, a ligação simbólica é criada pelo comando
`ln -s ficheiro_alvo ligação`

1º Problema : se um dos utilizadores acrescentar dados ao ficheiro, como evitar que os novos blocos sejam apenas acrescentados na directoria do utilizador que efectuou a operação?

Solução A (adoptada pelo Linux) : Os endereços dos blocos são armazenados no i-node.
Como os directórios dos dois utilizadores referenciam a mesma estrutura, qualquer alteração ao ficheiro passa a ser acessível aos dois utilizadores.

Ficheiros partilhados (4)

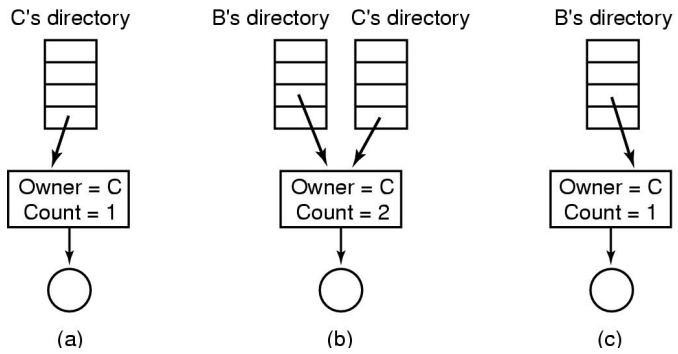


Figura 6-19, Modern Operating Systems

No i-node é mantido um contador das ligações para o ficheiro, que só é eliminado quando o contador chega a 0.

Solução B : Uma ligação simbólica é um tipo especial de ficheiro LINK.

Ficheiros partilhados (5)

2º Problema : quando se copia uma ligação, o que se transfere?

- a referência (e para onde, o caminho absoluto para o ficheiro ou o caminho relativo?) - inconveniente: noutro computador o caminho ou o ficheiro referenciado podem não existir.
- o ficheiro referenciado? - inconvenientes: duplicação do espaço em disco e a nova referência passa a ser um ficheiro independente.

No Linux:

- A cópia transfere a referência se o ficheiro destino tiver o mesmo identificador, caso contrário transfere o ficheiro referenciado.
- As operações de abertura e leitura sobre uma ligação actuam sobre o ficheiro referenciado, a eliminação actua apenas sobre a ligação.

Ficheiros partilhados (6)

Curiosidade, para avaliação avançada apenas

- As funções podem seguir, ou não, as ligações.
- A criação e leitura de ligações simbólicas definidas no POSIX.

POSIX:
```
#include <unistd.h>
int *symlink(char *,char *);
```
O 1º parâmetro é o caminho actual
```
#include <unistd.h>
int readlink(char *,char *,size_t);
```
O 2º parâmetro é a localização do *buffer*

O 3º parâmetro é a dimensão do *buffer*

Função	Segue? (S/N)
access	S
chdir	S
chmod	S
chown	N(até ver 2.1.81)/S
creat	S
exec	S
link	S
lstat	N
open,opendir	S
readlink	N
remove,rename	N
readlink	N
stat	S
unlink	S

Figura 4-17, Advanced Programming UNIX Environment

Estrutura de directorias (1)

[Def] **Directoria** é um ficheiro especial que contém as referência a ficheiros (programas, texto, subdirectorias,…) nele contidos.

- A ligação da directoria a um ficheiro nele contido é designada por ligação dura (“hard link”).
- As directorias são organizadas em árvore, com a raíz no Linux designada por **/** - “root”.
  - Em cada directoria todos os ficheiros devem ter identificadores distintos (directorias distintas podem conter ficheiros com mesmo identificador)
  - A cada utilizador é atribuído um directório quando ele(a) entra em sessão. O directório de entrada é referido por `$HOME` ou `~`.
  - Em cada directório existem duas referências:
    . Próprio directório
    . . Directório ascendente na hierarquia

Estrutura de directorias (2)

Curiosidade, para avaliação avançada apenas

A. Sistema de directoria única
  - Vantagens: estrutura muito simples
  - Inconvenientes: obriga identificadores distintos

Exemplo: directoria única com 4 ficheiros de 3 donos (A,B e C)

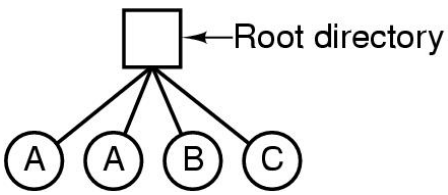


Figura 6-7, Modern Operating Systems

Estrutura de directorias (3)

Curiosidade, para avaliação avançada apenas

B. Sistema de directoria de dois níveis: todos os ficheiros colocados nas folhas, com directoria ascendente atribuído a dono dos ficheiros.
   - Vantagens: permite utilizadores distintos possuirem ficheiros com mesmo identificador.
   - Inconvenientes: obriga cada utilizador ter cópia dos ficheiros binários de sistema e é incómodo para utilizadores com muitos ficheiros.

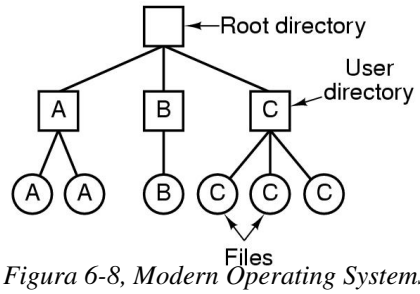


Figura 6-8, Modern Operating Systems

Estrutura de directorias (4)

Curiosidade, para avaliação avançada apenas

C. Sistema de directoria hierárquica: cada utilizador define a sua própria organização hierárquica de directorias, por onde os ficheiros são distribuídos.

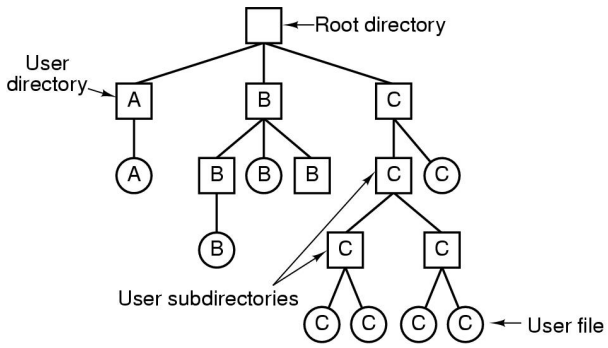


Figura 6-9, Modern Operating Systems

APUE 4.21

Acesso a directorias (1)

Curiosidade, para avaliação avançada apenas

- O POSIX define várias funções de manipulação de directorias.

POSIX:
```
#include <sys/types.h>
#include <sys/dirent.h>
DIR *opendir(const char *);
```

Em caso de erro devolve `NULL`.

POSIX:
```
#include <sys/types.h>
#include <sys/dirent.h>
int closedir(DIR *);
```

Em caso de sucesso devolve `0`, em caso de erro devolve `-1`.

Acesso a directorias (2)

Curiosidade, para avaliação avançada apenas

POSIX:
```
#include <sys/types.h>
#include <sys/dirent.h>
struct dirent *readdir(const char *);
```

A estrutura `dirent` deve conter campo `char d_name[]` com o identificador da entrada (ficheiro regular, directório ou outro).

```
struct dirent {
  ino_t d_ino;
  char d_name[NAME_MAX + 1]; }
```

POSIX:
```
#include <sys/types.h>
#include <sys/dirent.h>
void rewinddir(DIR *);
```

Retorna início a posição do directório.

Tipo de ficheiros no ls (1)

Curiosidade, para avaliação avançada apenas

- Cada interpretador de comandos ("shell") tem uma forma particular para indicar o tipo de ficheiros no comando `ls`.

A. O `csh` acrescenta um caractere no prefixo

Directorias:	/
Executáveis:	*
Ligações:	@
Tubos:	|
Texto:	

```
asterix.ist.utl.pt> ls /etc/X11
applnk/  lbxproxy/   serverconfig/  twm/  xinit/   xorg.conf   xserver/
fs/      prefdm*     starthere/     X@    xkb@     Xresources  xsm/
gdm/     proxymngr/  sysconfig/     xdm/  Xmodmap  X.rpmsave@
```


Tipo de ficheiros no ls (2)

Curiosidade, para avaliação avançada apenas

B. O `bash` utiliza cores no identificador, definidas na variável LS_COLORS no `~/.bashrc`

```
alias ls='ls --color'
LS_COLORS='di=1:fi=0:ln=31:pi=5:so=5:bd=5:cd=5:or=31:mi=0:ex=35:*.rpm=90'
export LS_COLORS
```

- Letras determinam tipo de ficheiro (di=directory, fi=file, ln=symbolic link, ex=executable file,…)
- Inteiro indica código de cor (0-omissão, 1-bold, 31-red, 90-dark gray,…)

```
[rgc@asterix ~]$  ls /etc/X11
applnk  lbxproxy   serverconfig  twm  xinit    xorg.conf   xserver
fs      prefdm     starthere     X    xkb      Xresources  xsm
gdm     proxymngr  sysconfig     xdm  Xmodmap  X.rpmsave
```

Gestão espaços de disco (1)

A. Dimensão dos blocos
   - Blocos de grande dimensão levam a desperdício de disco (por exemplo, blocos de dimensão 4KB levam um ficheiro de 1KB a despediçar 75% do espaço)
   - Blocos de reduzida dimensão diminuem capacidade de disco, medida por N * D_{bloco} (N – número de entradas da tabela de disco)
   - Blocos de reduzida dimensão levam a perdas de tempo na transferência do ficheiro entre memória e disco. k bits demoram

$$S_t + \frac{R_t}{2} + \frac{k}{d} \cdot R_t$$

- S_t - tempo de posicionamento ("seek")
- R_t - período rotacional
- d - densidade bits na faixa

Gestão espaços de disco (2)

- Mediana da dimensão dos ficheiros ronda 1K7 Bytes.
- Seja um ficheiro de 2KB e diferentes tamanhos de bloco
  - A tracejado é representada a eficiência da ocupação de disco, que é máxima até o bloco se tornar maior que o ficheiro.
  - A contínuo é representada a taxa de transferência de dados, que aumenta com a dimensão do bloco devido ao peso do posicionamento

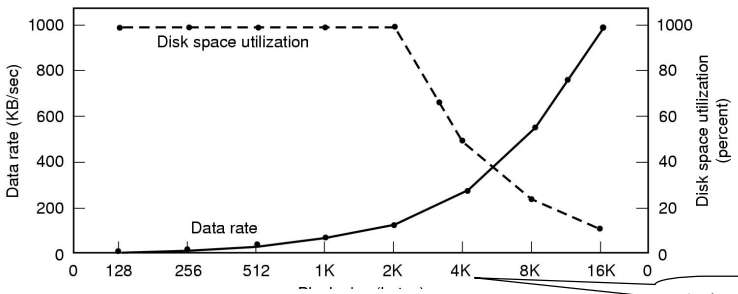


Figura 6-20, Modern Operating Systems

Adoptado pelo Linux

Gestão espaços de disco (3)

B. Acompanhamento dos blocos livres
Existem duas técnicas de registo dos blocos livres, listas ligadas e mapas de bits

Listas ligadas: num bloco armazenada tabela com endereços de blocos livres.
Para referências de E Bytes, um bloco de dimensão D armazena D/E-1 localizações de blocos livres (-1 destinado a ponteiro para bloco seguinte)

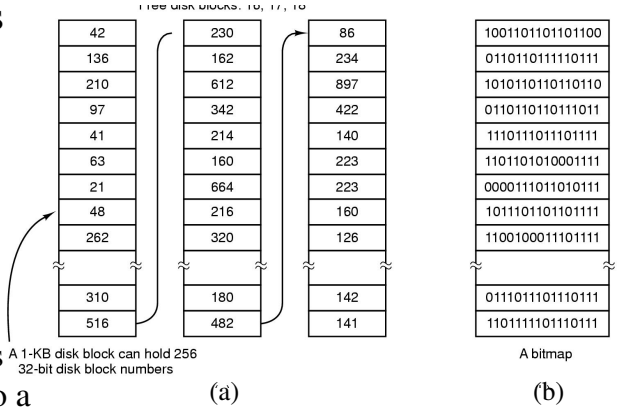


Figura 6-21, Modern Operating Systems

Ex: Para D=1KB e B=4, um bloco armazena 1KB/4-1=255 localizações

Gestão espaços de disco (4)

- Nas listas ligadas, os blocos de localizações livres são recolhidos entre os blocos livres. Quando um bloco de localizações livres é esvaziado, naturalmente é liberto.

Mapas de bits: o estado do bloco i determinado pelo valor do bit (0-livre, 1- ocupado).
Um bloco de dimensão D armazena estado de 8.D blocos.

Ex : Para D=1KB, o bloco armazena estados de 8 KB blocos

$$bit[i] = \begin{cases} 0 \rightarrow bloco[i] & \text{livre} \\ 1 \rightarrow bloco[i] & \text{ocupado} \end{cases}$$

Gestão espaços de disco (5)

C. Quota de disco

- Limites a observar, por cada utilizador, no número de ficheiros e espaço de disco
  - "soft": pode ser ultrapassado. No log-in seguinte recebe um aviso, reduzindo em 1 o número de avisos disponíveis.
    O utilizador deve eliminar ficheiros em excesso.
    O sistema operativo recusa admissão de utilizadores com 0 avisos disponíveis.
  - "hard": nunca pode ser ultrapassado. Tentativa de escrita em ficheiros com limite "hard" atingido gera erro.

Gestão espaços de disco (6)

- Abertura de ficheiro, mesmo doutro utilizador, leva sistema operativo instala tabela de quotas na memória.
- A tabela de quotas é salva em disco quando último ficheiro é fechado.

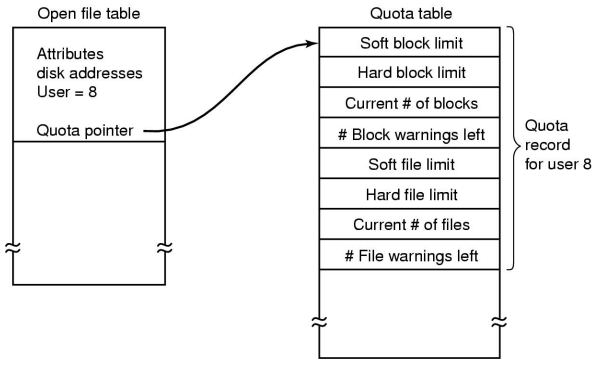


Figura 6-23, Modern Operating Systems

Gestão espaços de disco (7)

Curiosidade, para avaliação avançada apenas

- No Linux, a determinação de quotas de disco para utilizadores segue os seguintes passos (exemplo para partição `/home`):
  1. Entrar em modo de utilizador único, através do comando
     `init 1`
  2. Alterar sistema de ficheiros por forma a admitir quotas de disco, adicionando a opção `usrquota` no ficheiro `/etc/fstab`
     /etc/fstab anterior
     `LABEL=/home /home ext3 defaults 1 2`
     /etc/fstab alterado
     `LABEL=/home /home ext3 defaults,usrquota 1 2`
  3. Remontar sistema de ficheiro, através do comando
     `mount -o remount /home`

Gestão espaços de disco (8)

Curiosidade, para avaliação avançada apenas

4. Criar ficheiro `aquota.user` no topo da partição, através dos comandos
   `touch /home/aquota.user`
   `chmod 600 /home/aquota.user`
5. Instruir Linux a ler ficheiro `aquota.user`
   `quotacheck -vagum`
6. Alterar quotas de utilizador com comando `edquota -u`, que invoca o editor por omissão (por exemplo, `vi`)
   `Disk quota for user rgc (uid 500):`

limites blocos

limites inodes

```
Filesystem blocks soft hard inodes soft hard
/dev/hda3      24 5000    0      7    0    0
```

Blocos correntes (1KB) Limite "soft" i-nodes correntes

Gestão espaços de disco (9)

Curiosidade, para avaliação avançada apenas

7. Terminar modo de utilizador único, através do comando
   `init 3`

- O Linux não verifica as quotas quando o ficheiro é aberto, pelo que deve ser lançado periodicamente essa verificação com `cron`

```
#!/bin/bash
quotacheck -vagu
```

- Relatórios de utilização podem ser gerados pelo comando
  `repquota partição`

Agendar comandos (1)

Curiosidade, para avaliação avançada apenas

- `cron` é um processo Lunix que agenda comandos para serem executados automaticamente.

A. Os programas e as alturas de execução são indicadas num ficheiro, com linhas na forma
   min hour day-of-month month day-of-week program-to-be-run
   - min: (* indica uma vez por minuto)
   - day-of-week: com 0=domingo, 1=2ª feira (* indica todos os dias)
   - program-to-be-run]: indicar caminho desde /

   Nota: vários valores separados por vírgulas, gama indicada por -

[Exemplo] remover todos os ficheiros em /tmp às 18:30
   30 18 * * * rm /tmp/*

Agendar comandos (2)

Curiosidade, para avaliação avançada apenas

- Por omissão, sempre que o comando é executado o Linux envia uma mensagem por Email. Para que tal não suceda, inserir a seguinte directiva no final da linha de comando
  >/dev/null 2>&1
  - Cada utilizador pode ter o seu agendamente no ficheireo`/var/spool/user`
  - Os ficheiros de agendamento estabelecidos no arranque são colocados em `/etc/cron.d`

B. O agendamento é fixado pelo comando
`crontab fich`