Curso de Engenharia de Computação
Sistemas Operacionais

Sistemas de Arquivo – Parte II

Slides da disciplina Sistemas Operacionais
Curso de Engenharia de Computação
Instituto Mauá de Tecnologia – Escola de Engenharia Mauá
Prof. Marco Antonio Furlan de Souza

Operações em arquivos

INSTITUTO MAUÁ DE TECNOLOGIA

- **Chamadas de sistema mais comuns**
  - `Create`; `Delete`;
  - `Open`; `Close`;
  - `Read`; `Write`; `Append`;
  - `Seek`;
  - `Get attributes`; `Set attributes`;
  - `Rename`.

Operações em arquivos

INSTITUTO MAUÁ DE TECNOLOGIA

- **Chamadas de sistema mais comuns**
  - Exemplo no Linux

```
/*
 * copyfile.c - cópia de arquivos
 * Para compilar na linha de comando:
 *      gcc -o copyfile copyfile.c
 * Para executar, na pasta do executável, por exemplo:
 *      copyfile abc xyz
 * */
#include <sys/types.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

#define BUF_SIZE      4096    /* buffer de 4096 bytes */
#define OUTPUT_MODE   0700    /* modo de bits para arquivo de saída */
#define TRUE          1
```

Operações em arquivos

INSTITUTO MAUÁ DE TECNOLOGIA

- **Chamadas de sistema mais comuns**
  - Exemplo no Linux (cont.)

```
int main(int argc, char *argv[]) {
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];
    if (argc != 3)
        /* argc deve ser 3: copyfile nome_arq1 nome_arq_2 */
        exit(EXIT_FAILURE);
    /* Abrir o  arquivo de entrada e criar o arquivo de saída */
    in_fd = open(argv[1], O_RDONLY); /* abrir o arquivo de entrada */
    if (in_fd < 0)
        exit(EXIT_FAILURE); /* termina o programa se não conseguiu abrir*/
    out_fd = creat(argv[2], OUTPUT_MODE); /* cria o arquivo de destino */
    if (out_fd < 0)
        exit(EXIT_FAILURE); /* termina o programa se não conseguir */
```

Operações em arquivos

INSTITUTO MAUÁ DE TECNOLOGIA

- **Chamadas de sistema mais comuns**
  - Exemplo no Linux (cont.)

```
    /* laço de cópia */
    while (TRUE) {
        rd_count = read(in_fd, buffer, BUF_SIZE); /* lê um bloco de dados */
        if (rd_count <= 0)
            break; /* chegou no final de arquivo ou tem erro - termina */
        wt_count = write(out_fd, buffer, rd_count); /* escreve um bloco de dados */
        if (wt_count <= 0)
            exit(EXIT_FAILURE); /* não consegui escrever - termina */
    }
    /* fechar os arquivos */
    close(in_fd);
    close(out_fd);
    if (rd_count == 0) /* se não houve erro na última leitura – termina ok */
        exit(EXIT_SUCCESS);
    else
        exit(EXIT_FAILURE); /* houve erro - termina */
}
```

Diretórios

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - São arquivos **responsáveis** por manter a **estrutura** do **sistema** de **arquivos**;
    - **Organização**:
      - **Um nível**: monousuário, eficiente, simples. Ex.: CDC6600
      - **Dois níveis**: multiusuário, cada usuário tem diretório privado, compartilham programas do sistema → usuários com muitos arquivos;
      - **Hierárquico**: árvore de diretórios vários níveis – melhor organização de arquivos e diretórios – flexível e utilizado pelos sistemas operacionais modernos.
        - **Caminho**: conceito para localizar um arquivo ou diretório
          - **Absoluto**: `/home/marco/Downloads`
          - **Relativo**: `cp arquivos/teste.txt /tmp`
          - **Atalhos**: `.` (diretório atual), `..` (diretório-pai)

Diretórios

INSTITUTO MAUÁ DE TECNOLOGIA

- **Operações**
  - `Create`; `Delete`;
  - `OpenDir`; `CloseDir`;
  - `ReadDir`;
  - `Rename`;
  - `Link` (criar um atalho para o arquivo);
  - `Unlink`.

Sistema de arquivos – Implementação

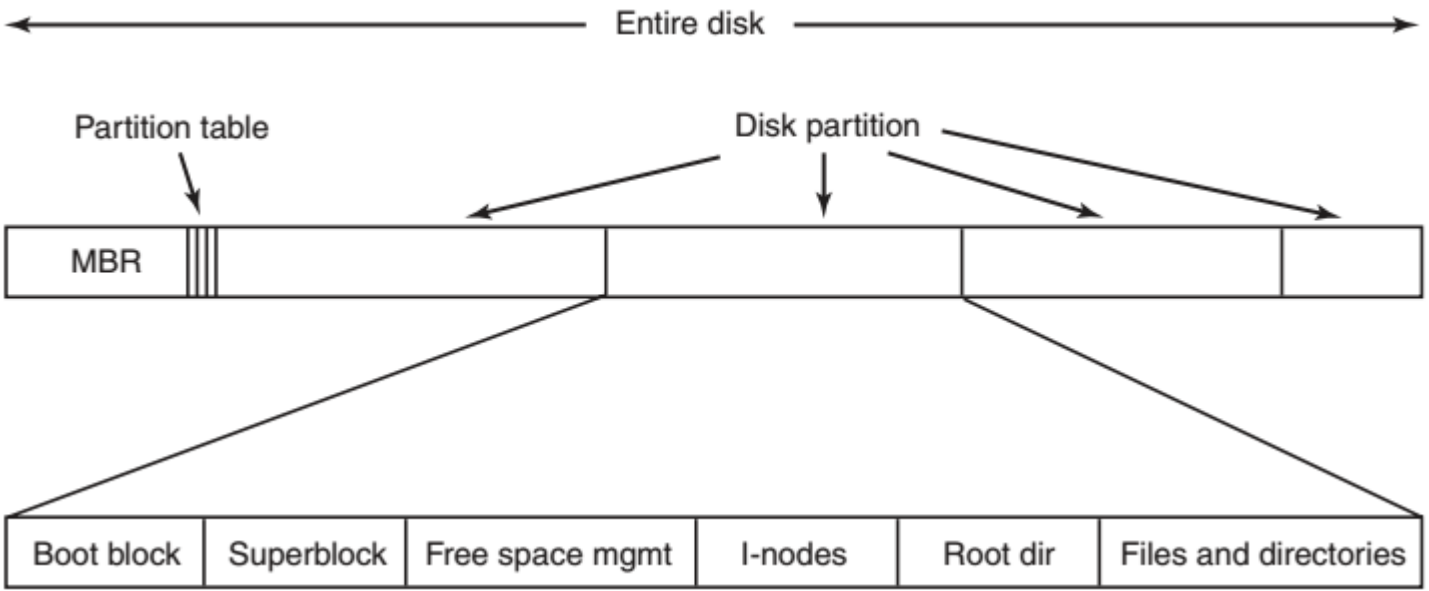
INSTITUTO MAUÁ DE TECNOLOGIA

- **MBR**
  - **MBR** = *Master Boot Record* – área do disco em que estão armazenados dados que são utilizados no *boot* para:
    1) Localizar a partição ativa;
    2) Ler o primeiro bloco dessa partição, chamado bloco de boot (boot block);
    3) Executar o bloco de *boot*.

Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Leiaute do disco**
  - **Uma possível organização**



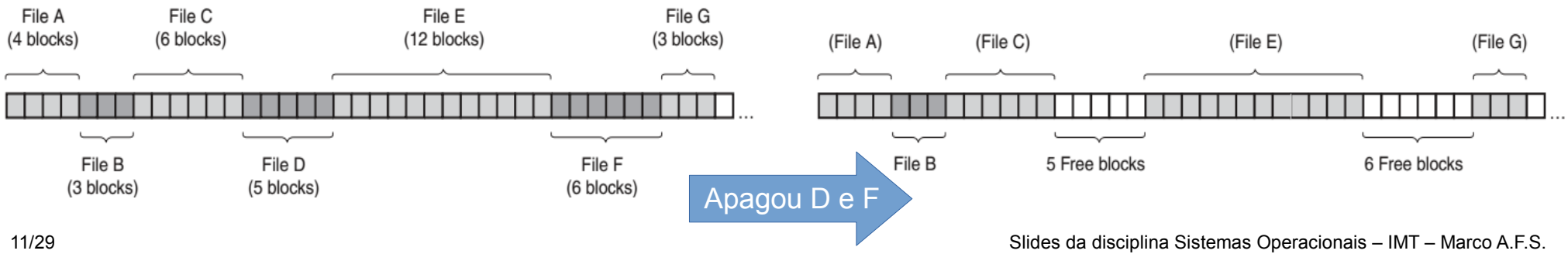
Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Leiaute do disco**
  - **Superbloco**: contém parâmetros (tipo do sistema de arquivos, número de blocos) – carregado na memória;
  - **Gerenciamento do espaço livre**: contém informações sobre os blocos livres do disco (mapa de bits ou lista encadeada);
  - **I-nodes**: estruturas de dados (vetor) contendo informações sobre os arquivos ;
  - **Diretório raiz**: árvore de diretórios;
  - **Arquivos e diretórios**: são os objetos criados no sistema de arquivos.

Sistema de arquivos – Implementação

- **Armazenamento de arquivos**
  - **Alocação contínua**
    - **Técnica** mais **simples**;
    - **Armazena arquivos** de **forma contínua** no disco;
    - **Exemplo**: disco com blocos de 1kB um arquivo com 50kB será alocado em 50 blocos consecutivos;



Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Armazenamento de arquivos**
  - **Alocação contínua**
    - **Vantagens**:
      - **Simplicidade**: somente o endereço do primeiro bloco e número de blocos no arquivo são necessários;
      - **Desempenho** para o **acesso** ao arquivo: **sequencial**;
    - **Desvantagens** (discos rígidos):
      - **Fragmentação** externa:
        - **Compactação**: alto custo;
        - **Reuso** de espaço: **atualização** da **lista** de **espaços** livres;
        - **Conhecimento** prévio do tamanho do arquivo para alocar o espaço necessário;
    - Usado em CD-ROM e DVD-ROM (apenas escrita).

Sistema de arquivos – Implementação

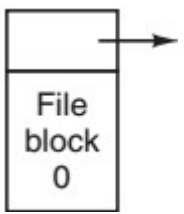
INSTITUTO MAUÁ DE TECNOLOGIA

- **Armazenamento de arquivos**
  - **Lista encadeada**
    - A **primeira palavra** de cada bloco é um **ponteiro** para o **bloco seguinte**;
    - O **restante** do **bloco** é destinado aos **dados**;
    - Apenas o **endereço** em **disco** do **primeiro bloco** do arquivo é armazenado;
      - **Serviço de diretório** é responsável por manter esse endereço que **localiza** os **arquivos**.

Sistema de arquivos – Implementação

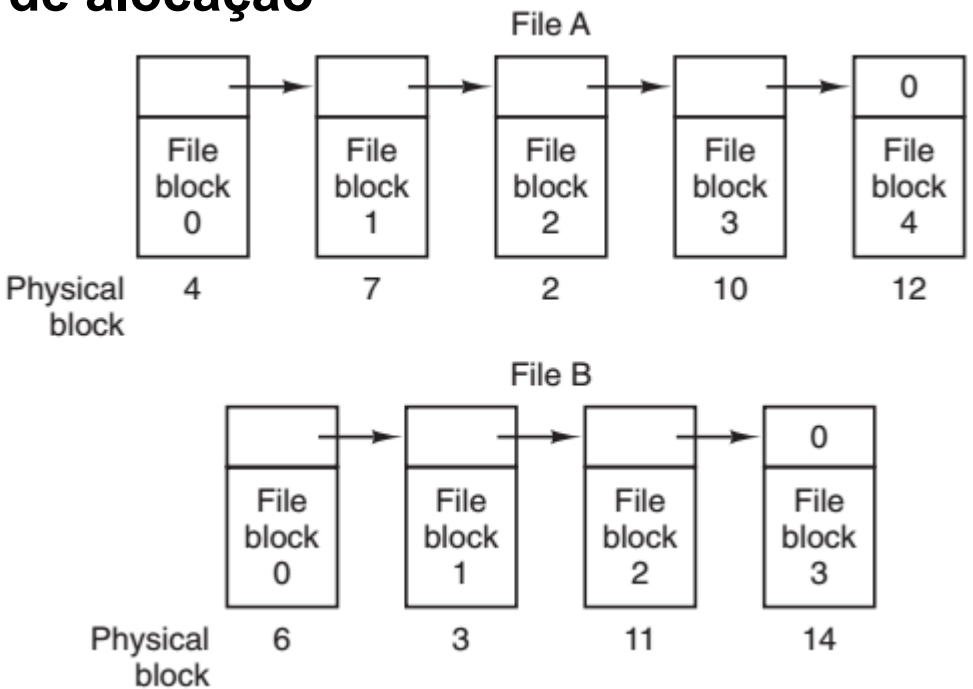
INSTITUTO MAUÁ DE TECNOLOGIA

- **Armazenamento de arquivos**
  - **Lista encadeada**
    - **Vantagens**
      - Não se perde espaço com fragmentação externa
    - **Desvantagens**
      - **Acessos aleatórios** a arquivos – torna o processo mais lento;
      - A **informação armazenada em um bloco não** é mais uma **potência** de **dois**, pois é necessário **armazenar** o **ponteiro** para o **próximo bloco**:



Sistema de arquivos – Implementação

- **Armazenamento de arquivos**
  - **Lista encadeada**
    - **Exemplo de alocação**



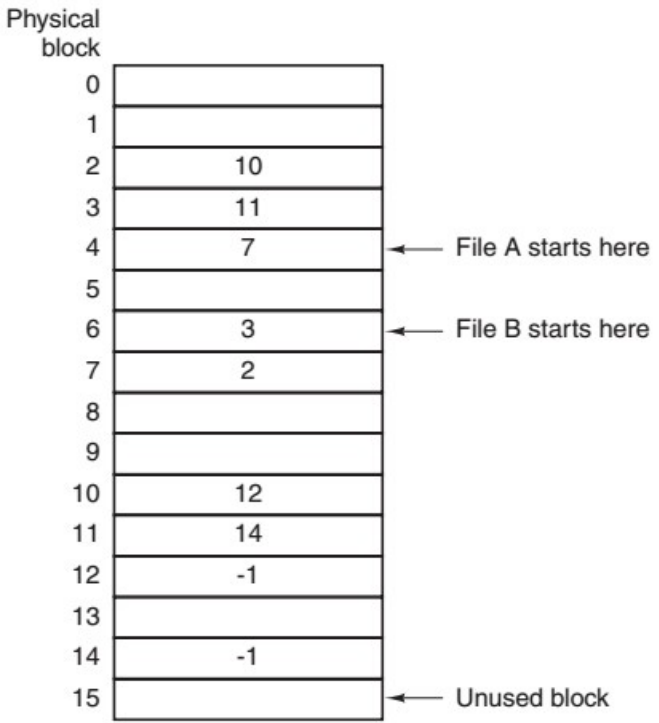
Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Armazenamento de arquivos**
  - **Alocação com lista encadeada utilizando uma tabela na memória**
    - O **ponteiro** é colocado em uma **tabela** na **memória** e não no bloco;
      - **FAT**: tabela de alocação de arquivos (*File Allocation Table*);
      - **Todo** o **espaço** do bloco está disponível para dados.
    - **Serviço de diretório** é responsável por manter o **início** do **arquivo** (bloco inicial);
      - Uso em MS-DOS e família Windows 9x
    - Acesso aleatório mais eficiente (sem acesso a disco) pelo uso da memória;
    - **Desvantagem**: toda a tabela deve estar armazenada na memória;
      - **Exemplo**: um disco com 1TB e com tamnaho de bloco de 1KB a tabela precisará de 1 bilhão de entradas, uma para cada um dos 1 bilhão de blocos de disco. Cada entrada precisa de, no mínimo, de 3 bytes (3 x 8 bits = 24 bits = 2^{24} endereços). Para melhorar a velocidade de busca, pode-se utilizar 4 bytes levando a uma tabela de 3GB fixa na memória.

Sistema de arquivos – Implementação

- **Armazenamento de arquivos**
  - **Alocação com lista encadeada utilizando uma tabela na memória**



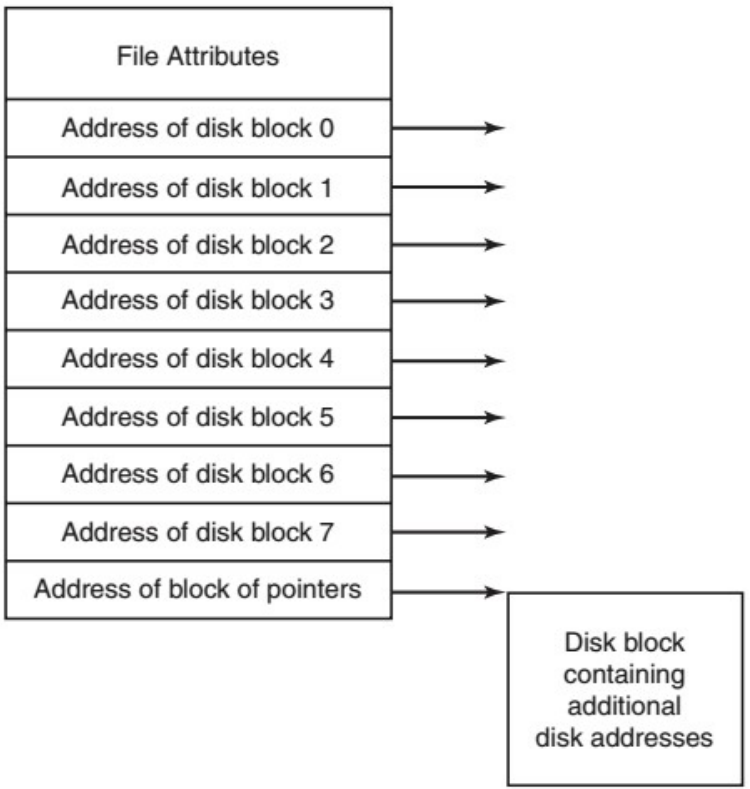
Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Armazenamento de arquivos**
  - **I-Nodes**
    - **Cada arquivo** possui uma **estrutura de dados** chamada **i-node** (*index-node*) que lista os atributos e endereços em disco dos blocos do arquivo – **dado o i-node de um arquivo** é possível **encontrar** todos os blocos desse arquivo;
    - **Se** cada **i-node** ocupa **n bytes** e **k arquivos** estão abertos ao **mesmo tempo**, portanto o total de **memória ocupada** é **k x n bytes**;
    - Usado no UNIX e Linux;
    - **Vantagem**: o i-node somente é **carregado** na **memória** quando o seu **respectivo arquivo** está **aberto** (em uso);
    - **Desvantagem**: o **tamanho** do arquivo pode **aumentar** muito – **solução**: **reservar** o **último endereço** para outros endereços de blocos.

Sistema de arquivos – Implementação

- **Armazenamento de arquivos**
  - **I-Nodes**



Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Diretórios**
  - **Entradas** de **diretório** provêm **informações** para **encontrar blocos** de **arquivos**;
  - **Dependendo** do **sistema**, estas informações **podem ser**:
    - **Endereço** no **disco** do **arquivo inteiro** (alocação contínua);
    - **Número** do **primeiro bloco** (esquema com lista ligada);
    - **Número do i-node**.
  - Um sistema de diretório mapeia o nome (ASCII ou outro) do arquivo para a informação necessária para localizar os dados.

Sistema de arquivos – Implementação

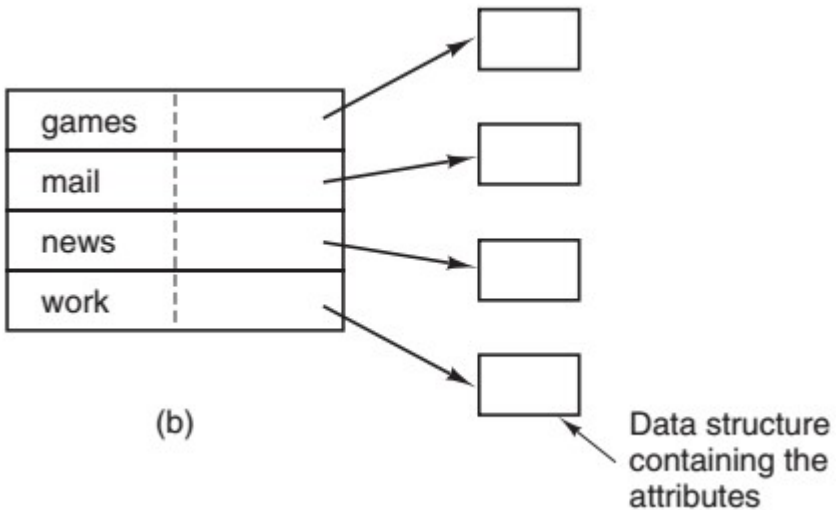
INSTITUTO MAUÁ DE TECNOLOGIA

- **Diretórios**
  - **Entradas fixas de diretório**
    - Um **diretório consiste** de uma **lista** de **entradas** de **tamanho fixo**, **uma por arquivo**, contendo o **nome** (**tamanho fixo**), uma **estrutura** de **atributos** de arquivo e **um** ou **mais endereços** (até um valor máximo) indicando onde os **blocos estão**.

games	attributes
mail	attributes
news	attributes
work	attributes

Sistema de arquivos – Implementação

- **Diretórios**
  - **Entradas de i-nodes**
    - Neste caso, o **diretório** contém apenas o **nome** do **arquivo** e seu **número** de **i-node** (economiza espaço).



Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Diretórios**
  - **Tamanho de nomes de arquivos**
    - Uma **opção** é manter um **espaço fixo** – exemplo: **255 caracteres** – mas gera-se um **gasto desnecessário** de espaço;
    - **Outra opção**: manter **cada diretório** com **entradas** de cabeçalho de **tamanho fixo** (**tamanho** da **entrada**, **seguido** de **estrutura** de **dados** de **tamanho fixo** como **dono**, **hora** da criação, **informação** de **proteção** entre **outros**);
    - Este **cabeçalho** de **tamanho fixo** é então **seguido** pelo **nome real** do **arquivo**, **não importando** o **tamanho**.

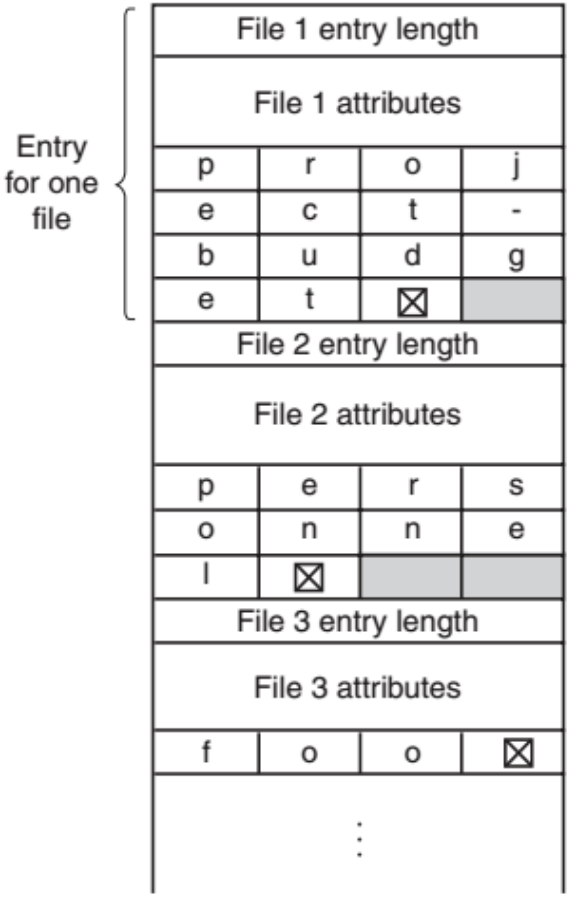
Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Diretórios**
  - **Tamanho de nomes de arquivos**
    - **Armazenamento em linha**
    - Uma **opção** é manter um **espaço fixo** – exemplo: **255 caracteres** – mas gera-se um **gasto desnecessário** de espaço;
    - **Outra opção**: manter **cada diretório** com **entradas** de cabeçalho de **tamanho fixo** (**tamanho** da **entrada**, **seguido** de **estrutura** de **dados** de **tamanho fixo** como **dono**, **hora** da criação, **informação** de **proteção** entre **outros**);
    - Este **cabeçalho** de **tamanho fixo** é então **seguido** pelo **nome real** do **arquivo**, **não importando** o **tamanho** (string terminada por '\0');
    - Os nomes devem se iniciar em limites de palavra (word) para melhor eficiência na reutilização do espaço.

Sistema de arquivos – Implementação

- **Diretórios**
  - **Tamanho de nomes de arquivos**
    - **Armazenamento em linha**
      - **Desvantagem**: ao se remover um arquivo cria-se uma lacuna de entrada variável – se for reutilizada por outro arquivo, talvez o nome não caiba.



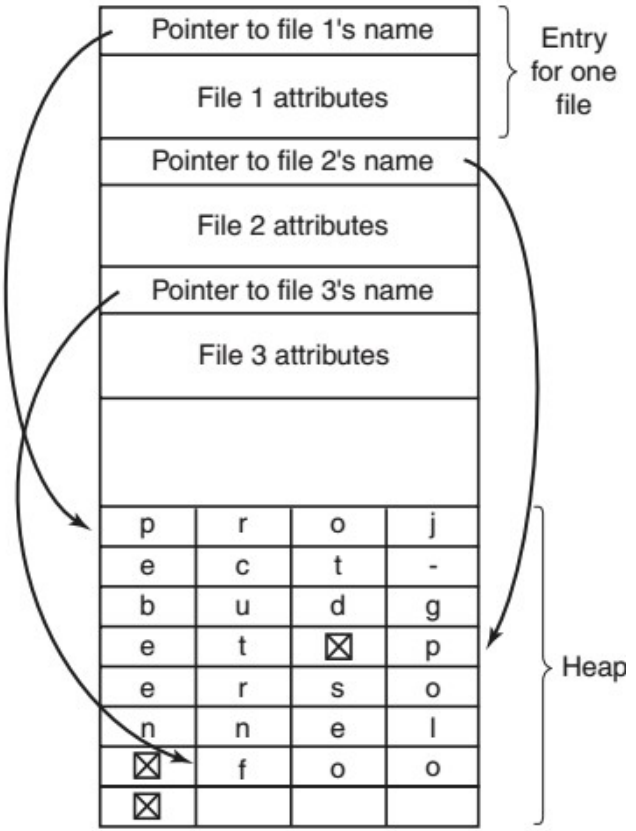
Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Diretórios**
  - **Tamanho de nomes de arquivos**
    - **Armazenamento em heap**
      - Neste tipo de solução, os **nomes** de **arquivos** são mantidos juntos em um **heap** no **final** do **arquivo**;
      - Este método tem a **vantagem** que quando uma **entrada** é **removida**, o **próximo arquivo** a ser inserido sempre **caberá lá** (o nome fica separado no heap);
      - Uma **desvantagem** é que o heap precisa ser **gerenciado** e que podem ocorrer **faltas** de **página** enquanto **processar nomes** de arquivos;
      - Um **pequeno ganho** de desempenho ocorre pois **não** há **necessidade** de se **iniciar nomes** em **limites** de **palavra**.

Sistema de arquivos – Implementação

- **Diretórios**
  - **Tamanho de nomes de arquivos**
    - **Armazenamento em heap**



Sistema de arquivos – Implementação

INSTITUTO MAUÁ DE TECNOLOGIA

- **Diretórios**
  - **Localização de nomes de arquivos**
    - Para **diretórios** muito **longos**, realizar **busca linear** por **nome de arquivo** pode ser **custoso** em **tempo**;
    - Se o **tamanho** do **diretório** é **n**, pode-se utilizar uma **tabela de hash** assim: com o **nome** do **arquivo** e com uma **função** de **hash** localiza-se diretamente a **posição** das **informações arquivo** na **tabela** (posição >=0 e posição< n);
    - Ao **se** deseja **inserir** um **arquivo**, **obtém-se** a **posição correspondente função** de **hash**. Se a **posição** estiver **livre**, **inserem-se** as **informações** do arquivo **ali**; **Senão**, **cria**-se uma **lista ligada** a **partir daquela posição** e **inserem-se** as **informações nela** (ou no fim dela, se já existir).

Referências bibliográficas

TANENBAUM, Andrew S. **Sistemas operacionais modernos**. 3. ed. São Paulo: Pearson, 2013. 653 p.