

# Sistemas de Arquivos: Arquivos e Diretórios

**Prof. Dr. Márcio Castro**  
[marcio.castro@ufsc.br](mailto:marcio.castro@ufsc.br)



UNIVERSIDADE FEDERAL  
DE SANTA CATARINA

1

# Introdução

# Introdução

- Processos armazenam e recuperam dados no seu próprio espaço de endereçamento virtual
- Problemas relacionados ao armazenamento em RAM
  - Capacidade de armazenamento restrita ao tamanho do espaço de endereçamento virtual
    - Insuficiente para grandes sistemas (e.g., reservas de passagens aéreas e sistemas bancários)
  - Armazenamento em memória volátil
    - Insuficiente para sistemas de bancos de dados
  - Dados armazenados pertencem necessariamente a um processo
    - Não há desacoplamento entre dados e processos

- **Três requisitos fundamentais para armazenamento de longo prazo**

- ① Deve ser possível armazenar uma grande quantidade de dados
- ② Os dados devem permanecer disponíveis após o término dos processos
- ③ Múltiplos processos devem poder acessar os dados ao mesmo tempo

# Introdução

- **Anteriormente vimos que:**

- O SO abstrai o conceito de **processador** criando uma **abstração de processo**
- O SO abstrai o conceito de **memória física** criando uma **abstração de espaço de endereçamento virtual**

- **Armazenamento permanente de dados**

- Abstrai o conceito de **disco**, criando a **abstração de arquivos e diretórios**

# Introdução

## ■ Questões a serem gerenciadas pelo SO

- Como encontrar uma informação?
- Como impedir que um usuário tenha acesso a informações de outro usuário?
- Como gerenciar o espaço livre?

2

# Arquivos

---

- **Conceito de arquivo**

- Um arquivo é uma **unidade lógica de informação** que pode ser **criada, alterada e removida** por um processo

- **Persistência**

- A informação armazenada em arquivos deve ser **persistente**
- Não pode ser afetada pela criação ou pelo término de um processo

- O SO define como os arquivos são:
  - Nomeados
  - Estruturados
  - Acessados
  - Protegidos
  - Implementados
- Estas e outras definições são feitas pelo sistema de arquivos

# Tipos de arquivos

## ■ Arquivos de texto

- São constituídos de **caracteres**
- Usa-se um **sistema de codificação** (e.g., **ASCII ou UTF-8**)
- Caracteres especiais são usados para indicar **espaço, nova linha, e outros caracteres de controle**
- Podem ser editados com qualquer editor de texto

## ■ Arquivos binários:

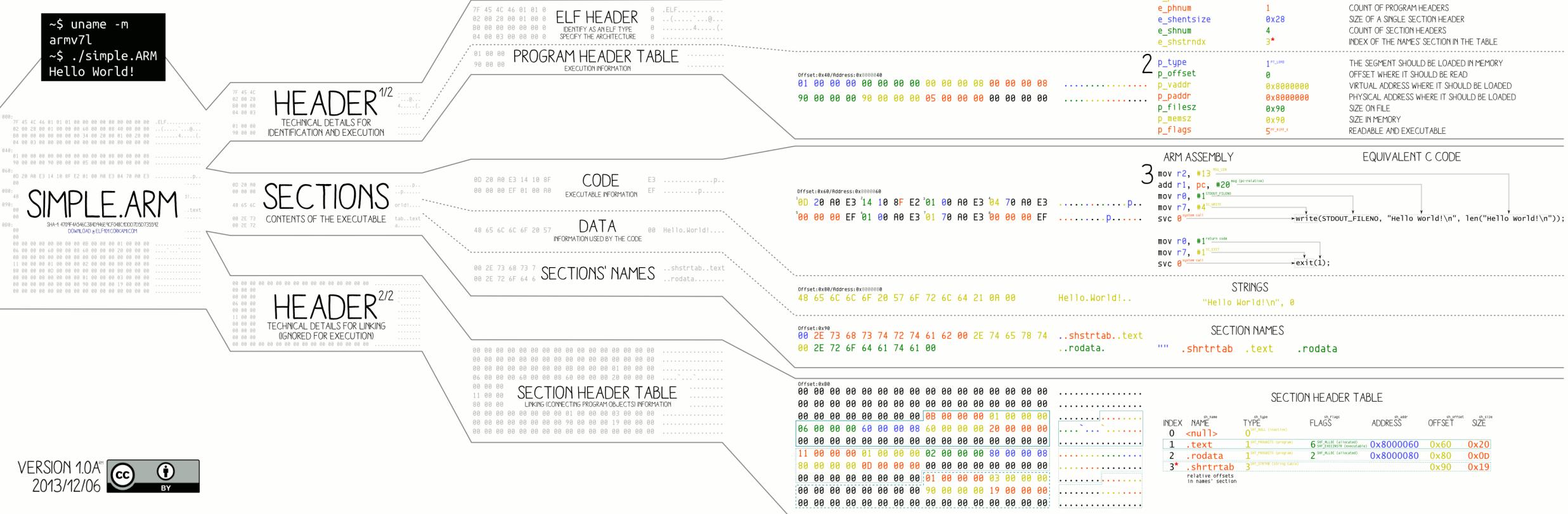
- São arquivos que não estão em formato texto
- Possuem uma estrutura interna bem definida
- **Exemplos: programa** (e.g., **ELF**), **imagem** (e.g., **PNG**), **áudio** (e.g., **MP3**), ...

# ELF<sup>101</sup> a Linux executable walkthrough

ANGE ALBERTINI  
CORKAMI.COM

## DISSECTED FILE

```
~$ uname -m
armv7l
~$ ./simple.ARM
Hello World!
```

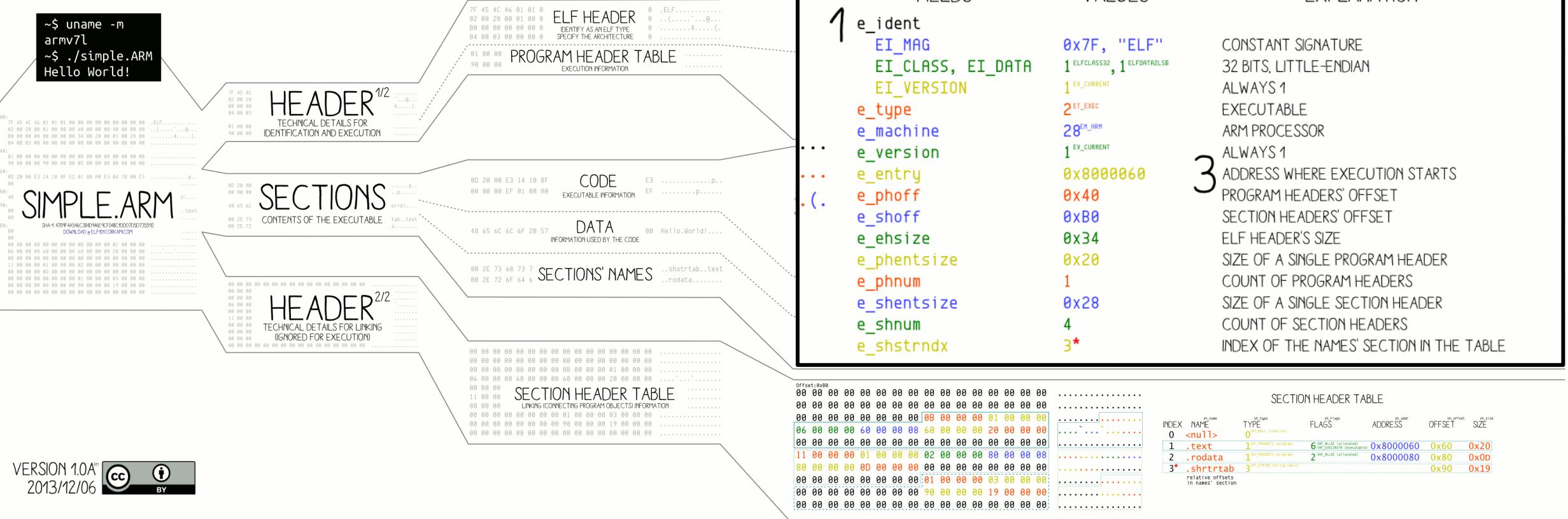


# ELF<sup>101</sup> a Linux executable walkthrough

ANGE ALBERTINI  
CORKAMI.COM

## DISSECTED FILE

```
~$ uname -m
armv7l
~$ ./simple.ARM
Hello World!
```



VERSION 1.0A<sup>TM</sup>  
2013/12/06

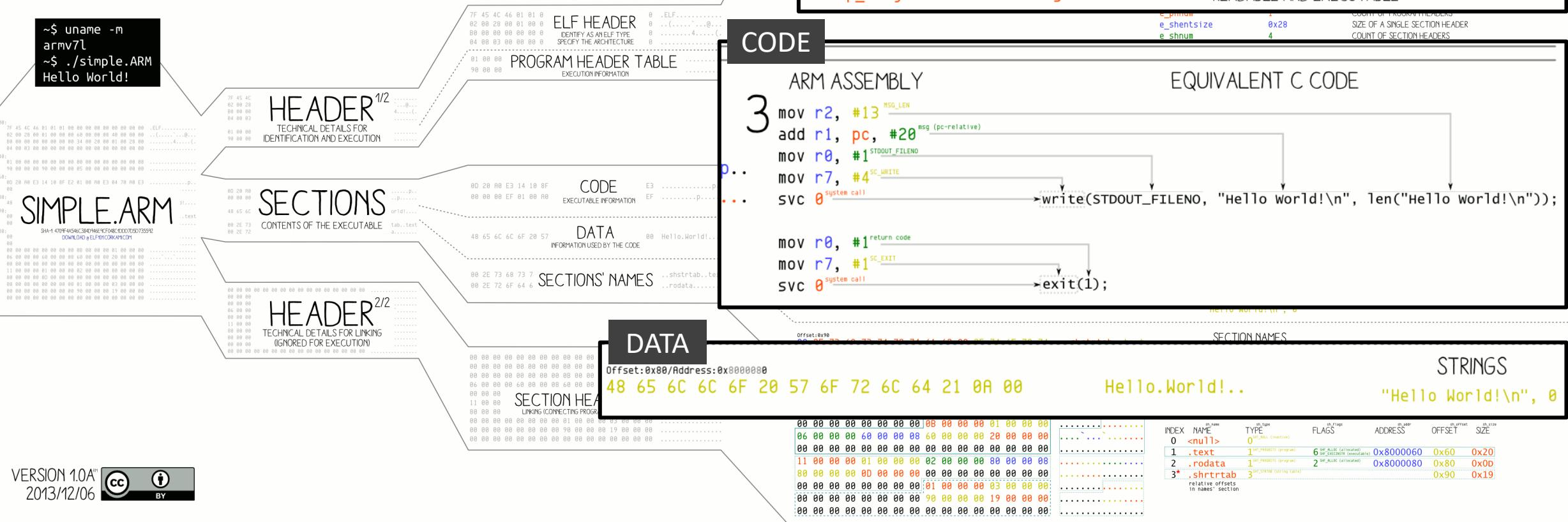


# ELF<sup>101</sup> a Linux executable walkthrough

ANGE ALBERTINI  
CORKAMI.COM

## DISSECTED FILE

```
~$ uname -m
armv7l
~$ ./simple.ARM
Hello World!
```



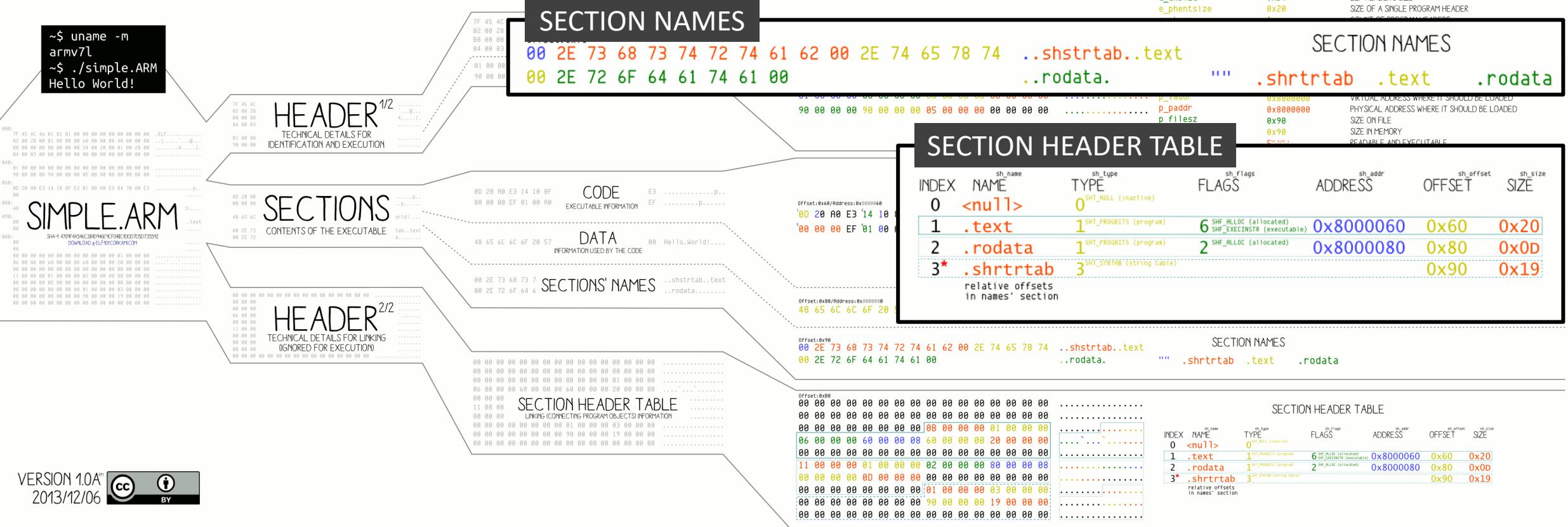
VERSION 1.0A™  
2013/12/06

# ELF<sup>101</sup> a Linux executable walkthrough

ANGE ALBERTINI  
CORKAMI.COM

## DISSECTED FILE

```
~$ uname -m
armv7l
~$ ./simple.ARM
Hello World!
```



# Acesso aos arquivos

- **Arquivos de acesso sequencial**

- Os bytes são lidos **sequencialmente** (em ordem) partindo-se do início
- Saltos ou acessos fora de ordem não são permitidos
- **Exemplo:** fitas magnéticas

- **Arquivos de acesso aleatório**

- Os bytes podem ser lidos **em qualquer ordem**
- Saltos são permitidos
- **Exemplo:** discos rígidos e SSDs

# Atributos de arquivos

- Arquivos possuem um **nome** e **dados armazenados** neles
- Além disso, arquivos possuem **atributos** (ou **metadados**)
  - Atributos são definidos pelo sistema de arquivos

Atributo	Significado
Criador	Usuário que criou o arquivo.
Proprietário	Proprietário atual do arquivo.
Flag de somente leitura	Leitura/escrita (0) ou somente leitura (1).
Flag de tipo	Arquivo texto (0) ou binário (1).
Momento de criação	Data e hora em que o arquivo foi criado.
Momento de última alteração	Data e hora em que o arquivo foi alterado pela última vez.
Tamanho atual	Tamanho do arquivo em bytes.
Proteção	Indicam quem tem acesso e de que modo.

# Operações com arquivos

- Operações em **arquivos** são feitas através de **chamadas de sistema**

## Operação Significado

<b>Create</b>	Cria um arquivo <b>vazio</b> e <b>inicializa alguns atributos</b> .
<b>Delete</b>	Remove um arquivo.
<b>Open</b>	Abre um arquivo, trazendo para a RAM seus metadados. Um <b>ponteiro de arquivo</b> aponta para o <b>primeiro byte do arquivo</b> .
<b>Close</b>	Fecha um arquivo e remove os seus metadados da memória.
<b>Read</b>	Lê um conjunto de bytes a partir da posição indicada pelo <b>ponteiro de arquivo</b> .
<b>Write</b>	Escreve um conjunto de bytes a partir da posição indicada pelo <b>ponteiro de arquivo</b> . A escrita <b>sobrescreve</b> dados existentes no arquivo.
<b>Append</b>	Escreve um conjunto de bytes a partir do <b>fim</b> do arquivo.
<b>Seek</b>	Modifica a posição do <b>ponteiro de arquivo</b> .
<b>Rename</b>	Altera o nome do arquivo.

Avançam  
ponteiro  
de arquivo

```
int main(int argc, char *argv[ ]) {
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit(1);

    if (in_fd = open(argv[1], O_RDONLY) < 0) exit(2);
    if (out_fd = creat(argv[2], OUTPUT_MODE) < 0) {
        close(in_fd); exit(3);
    }

    while (1) {
        if (rd_count = read(in_fd, buffer, BUF_SIZE) <= 0) break;
        if (wt_count = write(out_fd, buffer, rd_count) <= 0) {
            close(in_fd); close(out_fd); exit(4);
        }
    }

    close(in_fd);
    close(out_fd);
    if (rd_count == 0) exit(0);
    else exit(5);
}
```

3

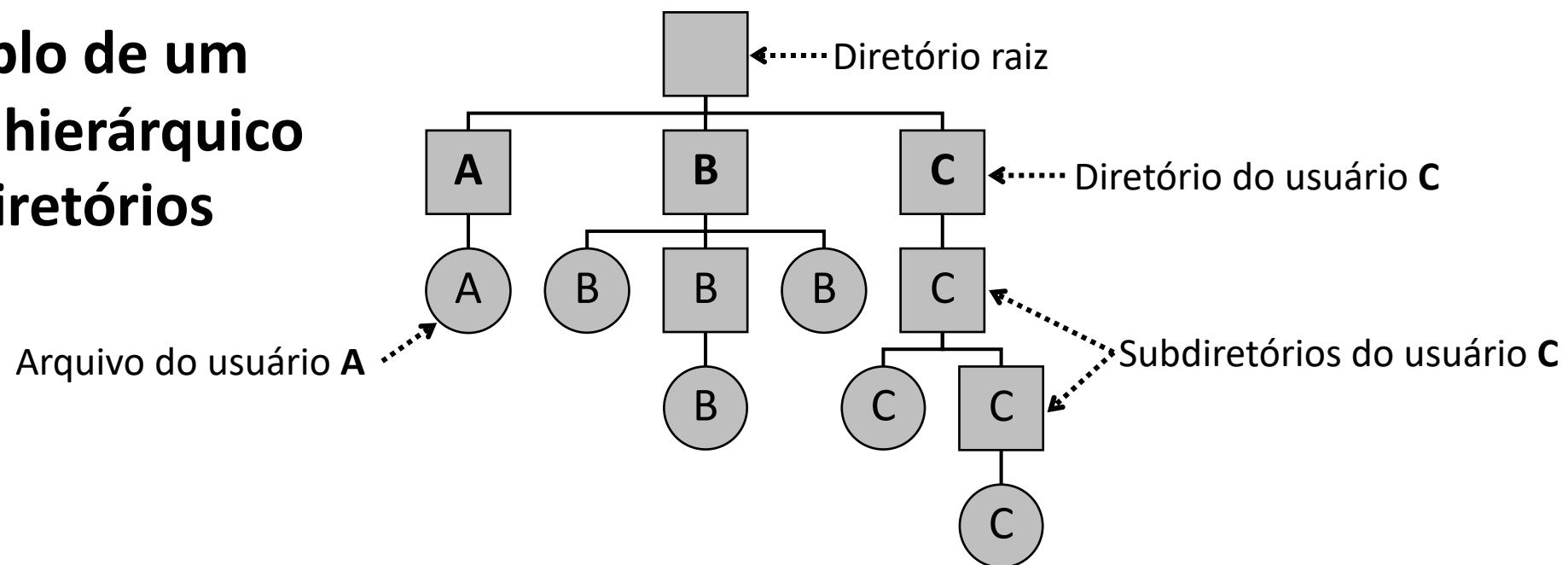
# Diretórios

---

# Diretórios

- Diretórios são unidades lógicas de armazenamento que permitem organizar arquivos de maneira hierárquica
  - O diretório é um arquivo
  - Possui apenas metadados no conteúdo do arquivo

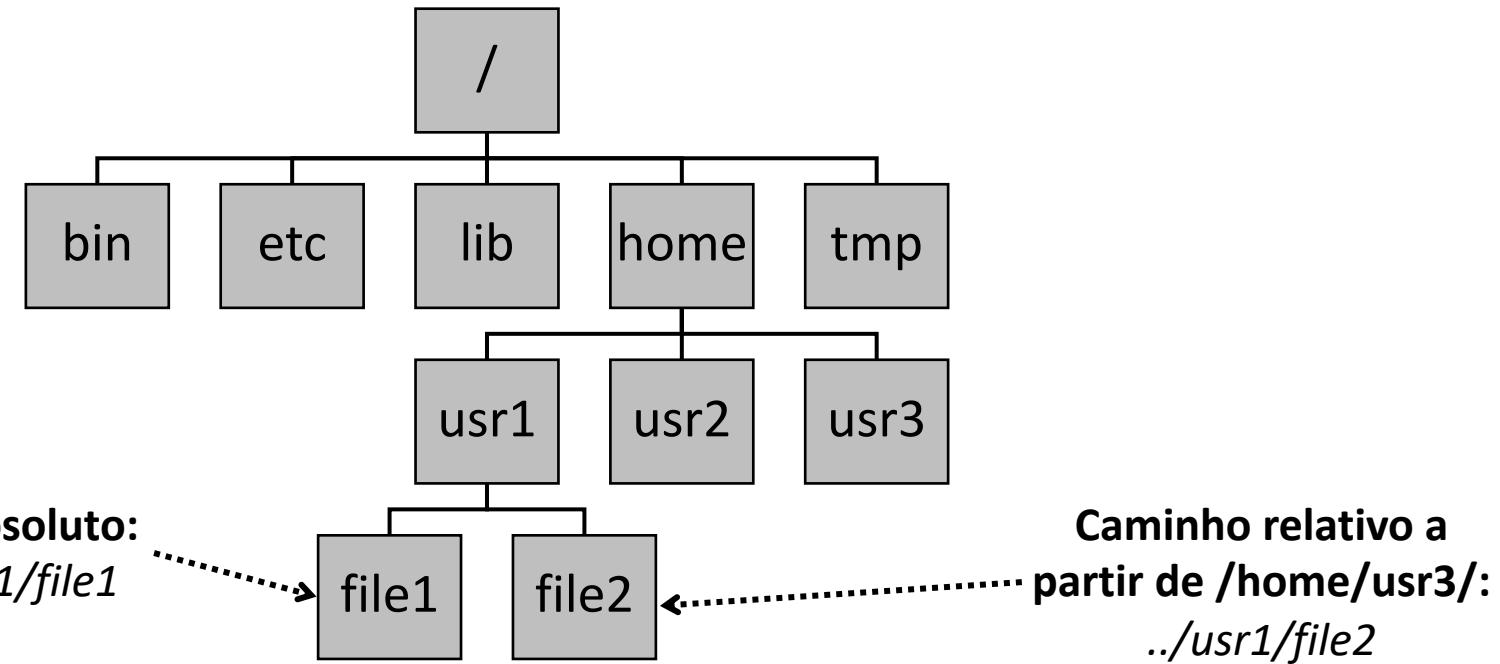
**Exemplo de um sistema hierárquico de diretórios**



Adaptado de TANENBAUM, Andrew S. *Sistemas operacionais modernos*. 3. ed. Rio de Janeiro: Pearson Prentice Hall, 2010. xiii, 653 p. ISBN 9788576052371.

# Caminhos em diretórios: exemplo UNIX

- Entradas especiais em cada diretório: “.” (diretório atual) e “..” (diretório pai)
- Caminho **absoluto**: a partir do diretório raiz
- Caminho **relativo**: a partir do diretório atual de trabalho



# Operações com diretórios

- Operações em diretórios são feitas através de **chamadas de sistema**

---

Operação	Significado
----------	-------------

---

Create	Cria um diretório <b>vazio</b> e adiciona as entradas “.” e “..”.
--------	---

Delete	Remove um diretório (normalmente precisa estar vazio).
--------	--

Opendir	Abre um diretório, trazendo para a RAM seus metadados.
---------	--

Closedir	Fecha um diretório e remove os seus metadados da memória.
----------	---

Readdir	Lê a próxima entrada em um diretório já aberto.
---------	---

Rename	Altera o nome do diretório.
--------	-----------------------------

Link	Adiciona uma entrada no diretório. Cria uma ligação entre um diretório e um arquivo, fazendo com que o arquivo “pertença” a um diretório ( <b>hard link</b> ).
------	--

Unlink	Remove uma ligação entre um diretório e um arquivo.
--------	---

---

!

# Obrigado pela atenção!



Dúvidas? Entre em contato:

- [marcio.castro@ufsc.br](mailto:marcio.castro@ufsc.br)
- [www.marciocastro.com](http://www.marciocastro.com)



Distributed Systems Research Lab  
[www.lapesd.inf.ufsc.br](http://www.lapesd.inf.ufsc.br)