Universidade de São Paulo
Brasil

Aula 11
arquivos

MAC0216 - Técnicas de Programação I

Professores: Alfredo, Daniel, Fabio e Kelly

Departamento de Ciência da Computação
Instituto de Matemática e Estatística

1.
Arquivos - classificações

Arquivos

Texto × Binário; Dados × Executável

- **Arquivo Texto** – é estruturado como uma sequência de linhas de texto. De forma geral, podem conter apenas caracteres "imprimíveis", tabulações horizontais e quebras de linhas.
- **Arquivo Binário** – podem conter qualquer tipo de dados, codificados em formato binário, para propósitos de armazenamento ou processamento computacional. Frequentemente, contém bytes que não devem ser interpretados como caracteres de texto.

Arquivos

Arquivo Texto

- Caracteres de quebra de linha

  Carriage return (CR) [ASCII 13] e Line Feed (LF) [ASCII 10]
- Cada SO usa uma quebra de linha diferente
  - Windows: CR+LF
  - Unix: LF
  - Mac (algumas versões): CR
- Comando `hexdump` mostra os bytes do arquivo (qualquer tipo)

Arquivos

Texto × Binário; Dados × Executável

- **Arquivo de Dados –** contém informações que são entrada ou saída de programas de computadores.
- **Arquivo Executável –** contém código ou instruções a serem executadas.

Arquivos de dispositivos

O Unix e seus derivados são sistemas orientados a arquivos:

- Os dispositivos periféricos também são tratados como um tipo especial de arquivo
- Esses arquivos especiais possibilitam que programas interajam com qualquer dispositivo por meio de chamadas ao sistema padronizadas para operações de E/S
- Exemplos de arquivos de dispositivos de entrada e saída:
  - impressora – `/dev/lp0`
  - hard disk ou SSD – `/dev/sd[x]` ou `/dev/hd[x]` ...
  - cdrom – `/dev/cdrom`
  - `/dev/null` [não está associado a um dispositivo físico!]
- Arquivos de dispositivos podem ser de três tipos: de caracter, de bloco e pseudo-dispositivo

“Brincando” com os arquivos de dispositivos

Faça este teste e observe o resultado:

- em um terminal, executar o comando `tty`. O resultado será algo como `/dev/pts/3`.
- em outro terminal, executar o comando `cat arq >/dev/pts/3` (onde `arq` é um arquivo texto presente no diretório atual).
- Executar também `cat >/dev/pts/3`. Digite caracteres e pressione [Ctrl-D] para efetuar a “transmissão”. [CRTL+D] em uma linha vazia encerra a “transmissão”.

“Curiosidade” sobre o cat

- Quando nenhum arquivo de entrada é passado para o `cat`, ele lê caracteres da entrada padrão até que as teclas CRTL-D sejam pressionadas em uma linha vazia.
- Exemplo: o comando abaixo cria um arquivo de nome “`meu_arq.txt`”, gravando nele tudo o que o usuário digitar até que CTRL-D seja pressionado em uma linha vazia

```
cat > meu_arq.txt
```

Extensões e "magic number"

Diferente das várias versões do Windows, em derivados do Unix não é necessário colocar extensão (.txt, .pdf, etc...) em arquivos (mas é bom colocar para facilitar a vida)

Antes de abrir / executar qualquer arquivo é bom saber o que ele é:

```
$ file aula11.pdf
aula11.pdf: PDF document, version 1.5
$ file script
script: Bourne-Again shell script text executable, ASCII text
```

O programa `file` conhece o formato de vários tipos de arquivos com base nos seus "magic numbers" (`man magic`)

Streams

- No Unix e seus derivados, um stream é um fluxo de dados (bytes ou caracteres) , que pode ser tanto a entrada quanto a saída de um programa
- No fluxo de dados, os dados são acessados sequencialmente, um a um
- Todo processo tem ao menos três streams, os chamados

  0 Entrada padrão

  1 Saída padrão

  2 Saída de erro

- Geralmente, os streams da entrada padrão e saída padrão estão conectados ao teclado e ao monitor, respectivamente
- Os shells dos derivados do Unix possuem também as operações de redirecionamento, que conectam os stream de entrada e saída padrão a arquivos

Streams em C

- Antes de poder ler ou escrever em um arquivo, é preciso estabelecer com ele uma conexão (ou canal de comunicação). Isso é feito na abertura do arquivo.
- Existem dois mecanismos diferentes para se representar conexões com arquivos: os descritores de arquivos (objetos do tipo `int`) e os streams (objetos do tipo `FILE*`)
- O conjunto de funções que realizam operações de escrita e leitura é muito mais rico e poderoso para streams
- Para descritores de arquivos, as funções se limitam a transferência de blocos de bytes
- Para streams, existem funções para a leitura e escrita formatada (`printf` e `scanf`), além de funções específicas para a leitura e escrita de caracteres e strings (`fgetc`, `fputs`, `getline`, etc.)

Pipes – mais detalhes sobre o seu funcionamento

- Pipes dependem da convenção de que todo programa tem inicialmente disponível para si de (pelo menos) dois streams de E/S: a entrada padrão e a saída padrão
- A operação de pipe conecta a saída padrão de um programa à entrada padrão de outro. A cadeia de programas conectados desta forma é chamada de pipeline
- Pipes são um mecanismo de comunicação inter-programas

Named pipes

`mkfifo` cria FIFOs (também chamados de "canais nomeados") com o nome de arquivo especificado.

- Um "FIFO" é um tipo de arquivo especial que permite comunicação entre processos. Um processo abre o arquivo FIFO para escrita, e outro para leitura.

Em um terminal:

```
$ mkfifo /tmp/canal
$ cat > /tmp/canal
olah vc
Tchau
CTRL+D
```

Em outro terminal:

```
$ tail -f /tmp/canal
olah vc
Tchau
```

Material recomendado

- Filosofia do Unix e pipes – livro: The Art of Unix Programming, de E.S. Raymond
  http://www.catb.org/esr/writings/taoup/html/