**Ministério da Educação**
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
UNED Nova Friburgo
Curso Técnico em Informática Integrado ao Ensino Médio

# Arquivos e redirecionamentos

Sistemas Operacionais


Prof. Bruno Policarpo Toledo Freitas
bruno.freitas@cefet-rj.br

# Objetivos

- **Redirecionar (salvar) saídas e erros de comandos para arquivos**
- **Introduzir o conceito de utilizar saídas de comandos como parte da execução de outros**
- **Introduzir o conceito de *pipes* e executar operações rotineiras com eles**

# Arquivos

- **Tudo em um sistema operacional GNU/Linux é baseado em arquivos**
  - Exemplo 1: pasta */dev* contém arquivos que representam dispositivos do sistema
  - Exemplo 2: dispositivos *plug & play* (ex.: pen drives) são pastas dentro do sistema de arquivos raiz ( **/** )
    - Normalmente em /media
    - Veremos isso em sistemas de arquivos
- **Quando um arquivo é aberto por um programa, é associado um número *identificador* do arquivo para o programa.**
  - Cada programa possui um identificador diferente para arquivos iguais

# Arquivos
# Canais de comunicação

- **Sistemas Operacionais GNU/Linux possuem 3 arquivos que representam canais de comunicação dos programas com os usuários:**
  - Entrada padrão (*stdin*)
    - */dev/*stdin
    - Identificador **0**
  - Saída padrão (*stdout*)
    - */dev/*stdout
    - Identificador **1**
  - Saída de erros padrão (*stderr*)
    - */dev/*stderr
    - Identificador **2**

# Arquivos
# Canais de comunicação

exemplo1 > C exemplo1.c > main()

```c
#include <stdio.h>
#include <string.h>

// Quando voces usam o printf() em C em sistemas GNU/Linux, vocês fazem isso:
int main() {

    char string[] = "Eu sou o stdout\n";

    // Abre o arquivo /dev/stdout ...
    FILE *identificador = fopen( "/dev/stdout" , "w"  );

    // ... e escreve nele
    fwrite( string , strlen(string), sizeof(char) , identificador );

    fclose(identificador);

    return 0;
}
```

TERMINAL PROBLEMS DEBUG CONSOLE

TERMINAL

```
bruno@Bruno-Desktop:$gcc exemplo1.c -o exemplo1
bruno@Bruno-Desktop:$./exemplo1
Eu sou o stdout
bruno@Bruno-Desktop:$
```

# Redirecionamentos com arquivos

- **Existem 3 maneiras principais de se redirecionar *saídas* e *entradas* de programas para arquivos:**
  - comando **>** arquivo : salva a saída de *comando* em *arquivo*
  - comando **<** arquivo : entrada de *comando* é o conteúdo de *arquivo*
  - comando **>>** arquivo *:* anexa a saída de *comando* em *arquivo*

# Redirecionamento *saída-padrão* para *arquivo*

- **Salva a saída-padrão de comandos em arquivos**
- **Trunca o arquivo-destino**
- **Exemplos:**

  ls ~ > arquivos.txt

  pwd > pasta.txt

# Redirecionamento *arquivo* para *entrada-padrão*

- **A entrada do *comando* é o conteúdo em *arquivo***
- **Exemplo:**

1)read NUMERO

echo “2” > numero.txt

read NUMERO < numero.txt

2)cd < pasta.txt

# Redirecionamento *saída-padrão* anexa a *arquivo*

- **Anexa a saída-padrão de um comando em um arquivo existente**
- **Cria o arquivo se ele não existir**
- **Exemplo:**

```
ls *.txt > arquivos.txt
ls /etc >> arquivos.txt
ls ~/Downloads >> arquivos.txt
```

# Exercício de fixação

1) Salve todos os processos em execução pelo seu usuário no arquivo “*meus_processos.txt*”
2) Salve todos os processos do usuário root no arquivo “*processos_root.txt*”
3) Usando o cat, concatene os arquivos acima e salve-os no arquivo *processos.txt*
4) Repita o exercício 3 mas agora usando o >> e sem usar o *cat*

# Redirecionamento dos *erros* para *arquivo*

- **Se redirecionarmos a saída (stdout) com o >, mensagens de erro ainda irão aparecer no terminal**

```
ls /eunaoexisto > ~/saida.txt
```

- **Para redirecionar os erros (stderr), devemos utilizar:**

```
ls /eunaoexisto 2> ~/erros.txt
```

# Redirecionamento dos *erros* e *saída-padrão* para *arquivos*

- **É possível redirecionar saída-padrão e os erros ao mesmo tempo em arquivos separados**

  firefox 1> saida.txt 2> erros.txt

# Redirecionamento dos *erros* para a *saída-padrão*

- **Para capturar a saída e os erros em um mesmo arquivo, devemos utilizar a seguinte sintaxe:**

```
libreoffice --writer  > saida_e_erros.txt 2>&1
```

# Descartando saídas e erros

- **Para ignorar saídas e erros de programas, utiliza-se o dispositivo nulo**

  /dev/null

- **Apenas é necessário redirecionar para esse dispositivo**
- **Exemplo:**

  libreoffice  --writer 1> /dev/null 2>&1

# Redirecionamento *saída-padrão* para *comando*

- **Utiliza a saída de um programa como parte da execução de outro programa**
  - *comando1* `*comando2*`:
    1) Executa *comando2* ;
    2) Substitui saída de *comando2* e então executa *comando1*
- **Muito usado em *scripts***
- **Exemplos:**

```
cd `pwd`
echo “A data hoje é: `date`”
```

# Pipes

- **Filosofia UNIX:**

  *“Tarefas complexas são realizadas pela concatenação de várias tarefas simples”*

- **Pipes: utilizados para redirecionar *a saída de um programa* para ser *entrada de um outro***

- **Sintaxe:**

  programa1 | programa2

# Pipesclear

- **Exemplos:**
  - ps -e | less
  - apt-cache search codeblocks | grep ‘codeblocks’
  - ps -e | grep ‘firefox’

# Programas para Pipes

- **Pipes são parte essencial de sistemas GNU/Linux**
- **Muitos programas são feitos para se utilizar junto com pipes:**
  - head / tail
  - sort
  - uniq
  - cut
  - join / paste
  - grep / egrep
  - wc
  - **find**
  - **sed**
  - **awk**
  - **...**
- **Ou seja: iremos voltar a esse assunto ...**

# Pipes & redirecionamentos para arquivos

- **Pipes podem ser combinados com redirecionamentos para arquivos**
  - ps -e | grep ‘root’ > processos_root.txt

# O programa tee

- **Alternativa para redirecionamentos em arquivos**
- **Usado quando queremos:**
  - Redirecionar a saída de um programa mas ainda ver sua saída.
  - Redirecionar a saída e salvar em arquivos com permissão de superusuários
- **Exemplo:**

  ls ~ > /home/bruno/meus_arquivos.txt (**ok**)

  ls ~ > /etc/meus_arquivos.txt (**erro**)

  ls ~ | sudo tee /etc/meus_arquivos.txt (**ok**)

# Exercícios

1) **Salve os nomes de todos os arquivos .conf da pasta /etc no arquivo $HOME/*todasConfiguracoes.conf***

2) **Quantos arquivos .conf há na pasta /etc?**

   Dica: utilize pipe com o programa *wc*

3) **Procure as informações do seu usuário no arquivo /etc/passwd.**

   Dica: utilize grep com *pipes*

4) **Imprima o UID do seu usuário a partir da questão 3**

   Dica: use o *cut* para imprimir a coluna da questão anterior em que essa informação se encontra

5) **Quantos usuários existem no sistema?**

   Dica:  conte quantas linhas possui o arquivo /etc/passwd com *pipes*

# Referências


- **FILHO, João Eriberto Mota. Descobrindo o Linux: entenda o sistema operacional GNU/Linux. 3a. ed. São Paulo: Novatec Editora, 2012.**
  - Capítulo 24

# Referências

- **e-book “Introduction to the command-line”**
  - http://write.flossmanuals.net/command-line/standard-files/
  - http://write.flossmanuals.net/command-line/file-structure/
- **The Linux Documentation Project (tldp.org)**
  - https://tldp.org/LDP/abs/html/io-redirection.html