Data Science & Big Data

Infraestrutura Computacional
Parte 1: Linux e Shell

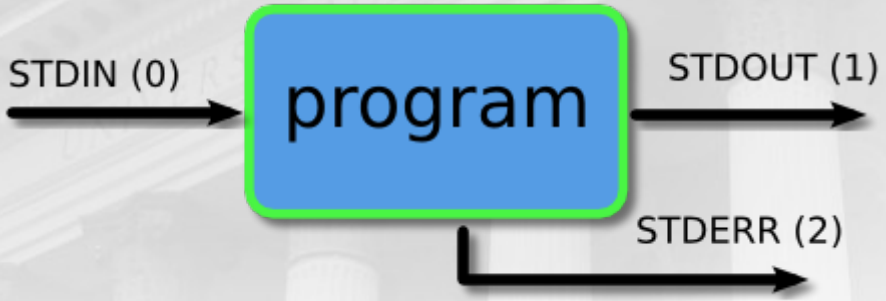
DSBD

Entrada/saída, Redirecionamento e *Piping*

Entrada e Saída

Todo programa em GNU/Linux possui 3 fluxos de dados (arquivos) conectados a ele

- STDIN (0) – entrada padrão (dados de entrada do programa)
- STDOUT (1) – saída padrão (dados produzidos pelo programa)
- STDERR (2) – saída de erros (mensagens de erro produzidas pelo programa)



Redirecionamento

Os fluxos podem ser redirecionados:

- Para arquivos
  - (>), (>>) - redireciona STDOUT

```
user@host: ls -l > saida.txt
user@host: ls -l $HOME >> saida.txt
user@host: cat > saida.txt
```

- De arquivos
  - (<) - redireciona STDIN

```
user@host: wc -l < dados.txt
user@host: wc -l < dados.txt > saida.txt
```

Redirecionamento

Redirecionamento da saída de erros

- (**2>**), (**2>>**) - redireciona STDERR

```
user@host: ls -l naoexiste 2> erros.txt

user@host: ls -l $HOME bla > saida.txt 2> erros.txt

user@host: ls -l $HOME bla > saida.txt 2>&1
```

Redirecionamento

Quatro regras para redirecionamento de saída

- Redirecionamento é feito **antes** de executar o comando
- Só é possível redirecionar saída que você pode (ou poderia) ver
- Redirecionamento vai apenas para um lugar
- Por padrão, somente a saída é redirecionada (não o erro)

Piping

Ao invés de enviar/receber dados de arquivos, é possível redirecionar dados da saída de um programa diretamente para a entrada de outro

- (|) - *pipe* redireciona STDOUT do comando à esquerda para a STDIN do comando à direita

```
user@host: ls -l $HOME | head -3

user@host: ls -l $HOME | head -3 | tail -1
```

- Argumentos de cada comando devem ser providos
  - DICA: construa seu *pipe* aos poucos, com um comando de cada vez

Pipe e Redirecionamento

O resultado final de um *pipe* pode ser redirecionado normalmente, como visto anteriormente

```
user@host: ls -l $HOME | head -3 | tail -1 > saida.txt

user@host: ls -l $HOME | tee saida.txt | head -3

user@host: ls -l $HOME | tail -n +8 | sort
```

Piping

Três regras para *pipes*:

- Redirecionamento por *pipe* é feito pelo shell, por primeiro, antes do redirecionamento de arquivos
- O comando à esquerda precisa produzir dados na saída padrão
- O comando à direita precisa ler da entrada padrão

Perguntas?

Controle de Processos

Processos

O que são processos?

- Uma instância em execução de um programa
- Possui uma área de memória própria, isolada de outros processos
- Diversos processos são executados simultaneamente num sistema GNU/Linux
  - Quando há mais processos ativos do que processadores, o SO dedica um pouco de tempo de processamento a cada processo, criando a ilusão de que eles são executados simultaneamente

Gerenciamento de Processos

Processos podem ser visualizados, mortos, suspensos, executados em segundo plano

Gerenciamento de processos é o trabalho de distribuir tempo de CPU entre vários programas

- Ex: Processo A é executado/interrompido, B é executado/interrompido, A volta a ser executado, …

Cada processo possui uma CPU virtual, e seus dados são carregados para o processador quando o processo está ativo

O usuário pode dar maior prioridade a determinados processos

Gerenciamento de Processos

Comando	Significado
`top`	mostra iterativamente os processos do sistema
`ps [aux]`	mostra os processos do sistema
`kill`	mata processos (sinais: -HUP, -9)
`pkill`	mata processos pelo nome
`strace`	mostra chamadas de sistema de um processo
`nice`	define a prioridade de um processo
`Ctrl+Alt+F?`	abre terminal em modo texto
`Ctrl + z`	suspende processo atual
`Ctrl + c`	mata processo atual
`bg ou &`	coloca processo em segundo plano
`fg`	coloca processo em primeiro plano

Exemplos

Comando
`ls /bin | sort | tee /tmp/lista | wc -l`
`ls -l /home/espinf | tail -n +2 | sed 's/\s\s*/ /g' | cut -d ' ' -f 3 | sort | uniq -c`
`ps aux | grep danielw`
`pkill firefox`
`strace ls $HOME | less`
`find /usr/share/ -iname '*.html' > arq.txt 2>/dev/null`

Referências

- Anatomy of the Linux kernel
- Linux OS Tutorial
- Introduction to UNIX
- Introduction to Linux
- Ryans Linux Tutorial