Proposição
Manipulando E/S e formatando dados

- Padrão para valores de saída
- Operadores para encaminhamento de E/S
- Pipes
- Testes lógicos
- Formatação e substituição de textos
- Aplicativos: cut, tr e sed
- Processador de texto com recursos avançados: awk

Redirecionamento de Entrada e Saída

- Como é sabido o SHELL trabalha com terminais que definem entradas e saídas
- Dispositivos de entrada e saídas padronizados
- Terminal de sessão ou processo em execução deverá possuir valores para entrada e saída padrões.

Existem “file descriptors” no sistema que definem a saída e entrada padrão, bem como seus valores:

- /dev/stdout
- /dev/stdin
- /dev/stderr

STDIN → valor igual a 0

STDOUT → valor igual a 1

STDERR → valor igual a 2

Redirecionamento de Entrada e Saída

É possível alterar o funcionamento padrão do encaminhamento da entrada e saída. Para tanto o SHELL dispõe de operadores de redirecionamento.

> : redireciona saída para a variável à direita do sinal

< : redireciona a entrada para à esquerda do sinal

>> : redireciona concatenando para a variável à direita do sinal

<< : redireciona concatenando para à esquerda do sinal

| : faz a junção de dois comandos. Fornece a saída de um processo como entrada para outro

Redirecionamento de Entrada e Saída

Exemplos de uso:

#> cat arquivo 1> sucesso.log

#> cat arquivo 2> erro.log

#> cat arquivo 2>erro.log 1>&2

#> echo $var >> sucesso.log

#> echo “zera tudo” > sucesso.log

#> tail -f arquivo >> /dev/ttys0

#> watch -n 2 df -h >> /dev/tty1

ATIVIDADE

Execute os comandos:

```
#> ls       1>&2    2>erro.txt
#> ls     /etc/bazinga  2>&1      1>acerto.txt
```

A saída resultante é aquela esperada? Por quê?

Redirecionamento de E/S

#> cat < /etc/hosts

#> tail < /var/log/messages

#> cat<<pare > arquivo

#> ftp -in server << exec

Condicionais

&& - Teste condicional AND

|| - Teste condicional OR

#> comando1 && comando2

#> comando1 || comando2

#> cat nada && ls

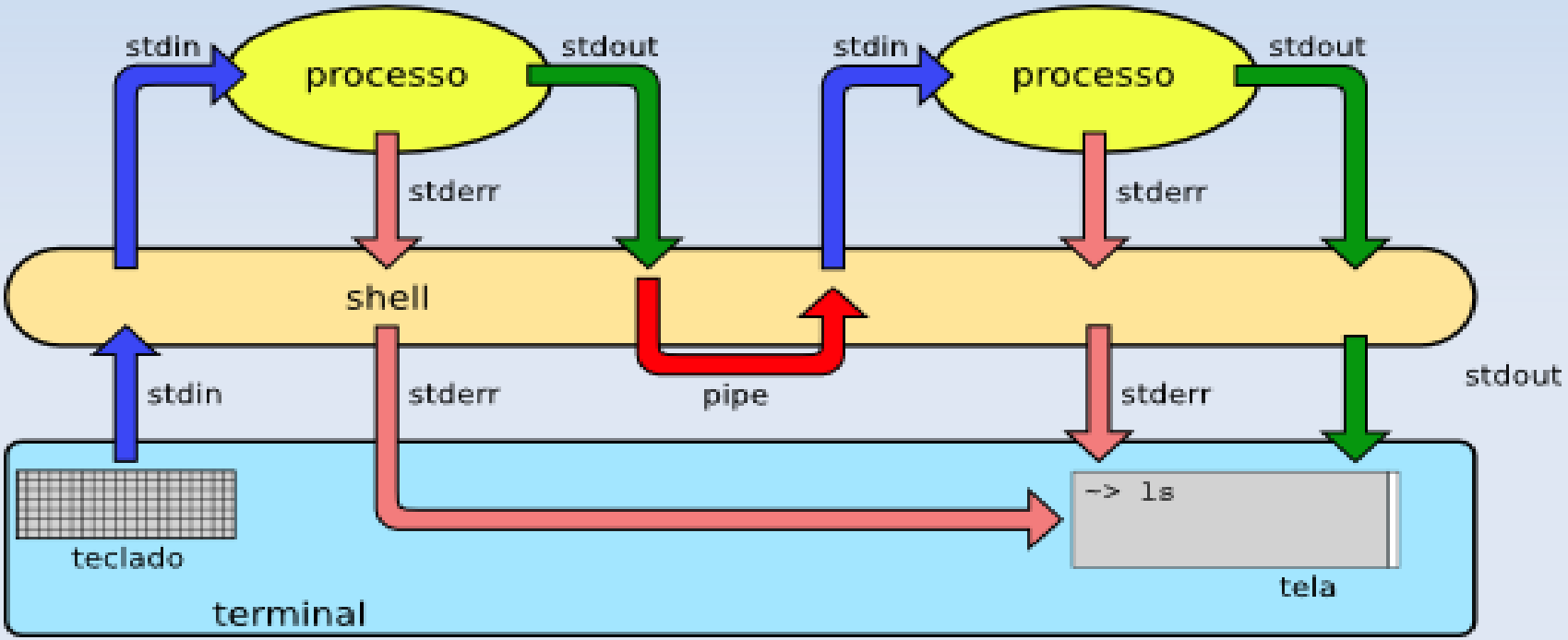
#> cat nada || ls

Atividade

Utilizando condicionais lógicos (AND ou OR), escreva em um único comando, uma verificação que valida a existência de um arquivo qualquer, atribuindo o valor 0 ou 1 para um arquivo de controle (arquivo de nome ”existe”)

Isto é, se o arquivo /etc/bazinga existir, então o arquivo existe.dat tem valor de 1, caso contrário, existe.dat terá valor de 0

Pipes



*

#> ls -la | wc -l

#> cat /etc/passwd | grep bash | wc -l > arquivo

* retirado de http://dainf.ct.utfpr.edu.br/~maziero/doku.php/unix:shell_avancado

Formatando dados
Importância e uso

Formatação de dados contidos em: variáveis ou arquivos texto.

Utilização de comandos/aplicativos invocados no shell ou através de linguagens interpretadas.

Criação de scripts (comandos em lote) para a tradução ou formatação dos dados.

Ferramentas e métodos

- Nesta seção serão abordados:

- Tradutores e interpretadores complexos: AWK e sed

- Comandos simples, técnicas de redirecionamento: tr e cut

Formatação com comandos auxiliares

tr - utilizado para traduzir, entrada padrão é transformada em outro caractere na saída padrão

sed – muito utilizado para substituição e reconhecimento de padrões. Entrada padrão recebe a ação específica e exibe na saída padrão

cut – formata o dado de entrada, mantendo-o apresentado conforme condição estipulada. O resultado vai para a saída padrão

Formatação com comandos auxiliares

TR

tr -s [:lower:] [:upper:] < text > words

echo "Abc123d56E" | tr -d [:digit:]

tr -d '\n' < textfile > newfile

Formatação com comandos auxiliares

CUT

cut -d" " -f1,2,3,4,5 filename > filename2

cut -d”:” -f1 /etc/passwd

Formatação com comandos auxiliares

SED

```
sed 's/\/usr\/local\/bin/\/common\/bin/' <old >new
```

```
sed 's/[^ ]*/(&)/g' < /tmp/usb
```

```
echo "123 abc" | sed 's/[0-9]*//'
```

```
sed 's/bash/&/p' /etc/passwd
```

```
free | tr -s ' ' | sed '/^Mem/!d' | cut -d" " -f2-4
```

Atividade

1. Determine o número de linhas da página de manual do shell Bash.
2. Determine quanto arquivos normais (não diretórios nem links) existem em /usr.

3 Monte uma linha de comandos usando pipes para identificar todos os usuários proprietários de arquivos ou diretórios a partir de /tmp, colocando o resultado no arquivo users-tmp.txt. Siga os seguintes passos:

Use o comando find para listar os proprietários de todos os arquivos dentro de /tmp (dica: use a opção -printf do comando find).

* Ordene a listagem obtida, usando o comando sort
* Remova as linhas repetidas, usando o comando uniq
* Direcione a saída para o arquivo indicado users-tmp.txt.

4. O arquivo /var/log/secure contém um registro de conexões de usuários ao servidor. Monte uma linha de comandos usando pipes para gerar uma listagem constando quantas vezes cada usuário se logou via SSH (shell seguro) no servidor durante este mês, ordenada por número de acessos, e armazene o resultado no arquivo users-ssh.txt. Siga os seguintes passos:

* Selecione as linhas referentes ao mês corrente no arquivo de log.
* Identifique e selecione as linhas referentes a inícios de conexões SSH.
* Recorte as colunas referentes aos nomes dos usuários.
* Ordene a lista de nomes de usuários.
* Remova as linhas repetidas, contando as repetições.
* Ordene a lista obtida por número de acessos.
* Desvie a saída para o arquivo desejado.

Linguagem AWK

AWK é definida como uma linguagem processadora de texto.

Suas principais finalidades são:

- Gerar relatórios ou recortar dados importantes
- Adicionar funções a editores de texto
- Transformar padrões de arquivos
- Criar pequenas base de dodos
- Efetuar operações matemáticas sobre arquivos com dados numéricos

Linguagem AWK

Execuções simples do AWK , em uma única linha de comando

awk '/root/' /etc/passwd

Filtra as linhas que contém o padrão “root” dentro do arquivo passwd e as exibe na saída padrão

Linguagem AWK

A lógica empregada no processador do AWK pode ser simplificada na seguinte expressão:

awk '<padrão buscado> {<ação do programa>}' <dado entrada>

Exemplo:

awk ' /root/ { print $2 }' /etc/passwd

Linguagem AWK

A grande diferença é que AWK reconhece uma sequência diferente daquela utilizada pelas linguagens estruturadas que estamos acostumados

```
awk ' {print NR,”bash”}'  /etc/passwd
```

```
awk ' END  {print NR,”bash”}' /etc/passwd
```

Linguagem AWK

Para tanto, deve-se notar a existência não obrigatória das cláusulas BEGIN e END

```
awk ' BEGIN {<inicialização>}
        <padrao1>  {<ação 1>}
              <padrao2>  {<ação 2>}
              . . .
             END  {<ação final>} '  <dado entrada>
```

Linguagem AWK

É possível, através de ações, executar comandos condicionais ou operações aritméticas:

```
awk -F":" '{if ($3==0) print "esse é o root"}'
  /etc/passwd
```

```
awk -F”:” '{sum_id +=$3}
END {printf (”%d\n”),sum_id }' /etc/passwd
```

Linguagem AWK

```
# This is an awk program that summarizes a coin collection.
  #
  /gold/    { num_gold++; wt_gold += $2 }      # Get weight of gold.
  /silver/  { num_silver++; wt_silver += $2 }  # Get weight of silver.
  END { val_gold = 485 * wt_gold;              # Compute value of gold.
        val_silver = 16 * wt_silver;           # Compute value of silver.
        total = val_gold + val_silver;
        print "Summary data for coin collection:";  # Print results.
        printf ("\n");
        printf ("   Gold pieces:                %2d\n", num_gold);
        printf ("   Weight of gold pieces:      %5.2f\n", wt_gold);
        printf ("   Value of gold pieces:      %7.2f\n",val_gold);
        printf ("\n");
        printf ("   Silver pieces:              %2d\n", num_silver);
        printf ("   Weight of silver pieces:    %5.2f\n", wt_silver);
        printf ("   Value of silver pieces:    %7.2f\n",val_silver);
        printf ("\n");
        printf ("   Total number of pieces:     %2d\n", NR);
        printf ("   Value of collection:       %7.2f\n", total); }
```

Linguagem AWK

```
gold     1    1986  USA              American Eagle
gold     1    1908  Austria-Hungary  Franz Josef 100 Korona
gold     1    1984  Switzerland      ingot
gold     1    1979  RSA              Krugerrand
gold     0.5  1981  RSA              Krugerrand
gold     0.1  1986  PRC              Panda
gold     0.25 1986  USA              Liberty 5-dollar piece
gold     0.25 1987  USA              Constitution 5-dollar piece
gold     1    1988  Canada           Maple Leaf
```

Linguagem AWK

```
awk -f summary.awk coins.txt
##########################################################
Summary data for coin collection:
   Gold pieces:                9
   Weight of gold pieces:      6.10
   Value of gold pieces:       2958.50
   Silver pieces:              4
   Weight of silver pieces:    12.50
   Value of silver pieces:     200.00
   Total number of pieces:     13
   Value of collection:        3158.50
```

Atividade

- Dado um arquivo de alunos, formatado conforme o seguinte padrão

Nome,nota,matricula

José da Silva Solteiro, 10, 1023

Mauro da Silva Sauro, 4, 1302

Tulio Maravilha Alberto, 6, 1245

Vania Orleans e Bragança, 6, 1231

Atividade

- Crie um script awk que vai ler esse arquivo de entrada e ao seu final ele deverá imprimir na tela:
- A quantidade de alunos reprovados (nota menor que 5)
- A quantidade de alunos aprovados (nota maior que 5)
- Media geral da turma, quantidade de alunos
- Maior número de matrícula com o nome desse aluno