TRYBE - 25/06/20 – AULA 4 BLOCO 1

TEMA: UNIX & BASH | PARTE 2

- **REDIRECIONAMENTO**

A maior parte dos processos iniciados por comandos UNIX tem retorno padrão na tela do terminal a muitas recebem dados pelo processo padrão (comandos do teclado) e também existem mensagens de erro padrão que são retornada para a tela do terminal.

Quando rodamos o comando cat sem associa-lo a um arquivo especifico (ex: cat readme.txt) ele recebe o input e retorna o mesmo valor preenchido ao finalizar o comando com ctrl + d.

No UNIX podemos redirecionar tanto o input (entrada – geralmente pelo teclado) quanto o output (saída – geralmente na tela) dos comandos.

- **REDIRECIONANDO O OUTPUT (SAÍDA)**

Para redirecionar o output (saída) de um comando utilizamos a simbologia **>**

Ex: Criando uma lista através do comando cat.

1. **cat > lista** (enter)
2. Digitar as informações que **vão compor** o arquivo
3. **Ctrl + d** para finalizar a edição.
4. O **output** será um arquivo criado com o nome de lista contendo o conteúdo digitado.

- **ADICIONANDO INFORMAÇÕES EM UM ARQUIVO**

Para adicionar informações a um arquivo existente utilizamos a simbologia **>>**

Ex: Adicionando informações ao arquivo lista

1. **cat >> lista** (enter)
2. Digitar as informações que **serão adicionadas** ao arquivo.
3. **Ctrl + d** para finalizar a edição.
4. O **output** será a modificação no arquivo lista existente.
5. Podemos checar se o conteúdo foi adicionado através do comando **cat lista.**

- **CONCATENAR OU UNIFICAR ARQUIVOS**

Para concatenar (juntar / unir) dois arquivos utilizamos a simbologia **>**

Ex: Criar uma nova lista2 e unir a lista existente.

1. **cat > lista2** (enter).
2. Digitar as informações que **vão compor** o arquivo
3. **Ctrl + d** para finalizar a edição.
4. **cat lista lista2 > lista_unificada** (enter).
5. O output será a **união das duas listas** sem perder os arquivos originais.

- **REDIRECIONANDO O INPUT (ENTRADA)**

Para redirecionar o input (entrada) utilizamos a simbologia **<**

O comando **sort** organiza uma lista em ordem alfabética.

Ex: Utilizar o comando sort para organizar em ordem alfabética uma lista existente.

1. **sort < lista_unificada** (enter).
2. O **output** será o conteúdo da lista unificada em ordem alfabética retornado na tela.

Ex2: Criar uma nova_lista com o conteúdo da lista_unificada em ordem alfabética.

1. **sort < lista_unificada > nova_lista** (enter).
2. O **output** será um arquivo chamado **nova_lista.**

- **PIPES (PEGAM SAIDA DE UM COMANDO E ADICIONA NA ENTRADA DE OUTRO COMANDO)**

Para ver quem está utilizando o sistema utiliza-se o comando **who**.

Uma forma de gerar uma lista em ordem alfabética dos usuários ativos no sistema é :

1. **who > names.txt** (enter) – Redireciona o output (saída) do comando **who** para o arquivo names.txt
2. **sort < names.txt** (enter) – Redireciona o input (entrada) do comando **sort** para o arquivo names.txt
3. O **output** será uma lista de nomes organizada em ordem alfabética na tela do terminal.

Uma forma mais simples e "limpa" de se obter o mesmo output (saída) é utilizando **pipe**, representado por **|.**

1. **who | sort** (enter)
2. O **output** será uma lista de nomes organizada em ordem alfabética na tela do terminal.

- **CRIANDO DIRETÓRIOS E ARQUIVOS**

Para contar quantos usuários ativos no momento podemos também utilizar pipe:

1. **who | wc –l** (enter) – **wc** (Word Counter , **-l** conta linhas).
2. **O output** (saída) será o número de linhas/usuários ativos no sistema na tela do terminal.

Ex: Exibindo no terminal todas as palavras com a de duas listas (lista e lista2) em ordem alfabética.

1. cat lista lista2 | grep 'a' | sort
2. O output (saída) será exibido no terminal.

Explicação: neste exemplo o comando **cat** irá exibir o **output** (saída) que são as palavras começando com 'a' selecionadas através do comando **grep 'a'** e organizadas em ordem alfabética pelo comando **sort**.

- **SISTEMA DE SEGURANÇA DE ARQUIVOS**

Cada arquivo e diretório possui permissões de acesso que podem ser encontradas através do comando **ls –l** ou **ls –lg**.

-rwxrw-r-- 1 ee51ab beng95 2450 Sept29 11:52 file1

Na coluna em **vermelho** encontra-se uma string (conjunto de letras, símbolos e números) contendo 10 caracteres (***-rwxrw-r--***).

O sinal – da cor preta indica que este é um arquivo. Quando iniciar com a letra **D** quer dizer que é um diretório.

O grupo em **vermelho** concede 3 permissões ao owner (dono/usuário/quem está logado) do arquivo ou diretório **r** (read) **w** (write) **x** (execute).

O grupo em **azul** concede 2 permissões ao grupo (ex: administrador, staff, etc) que o arquivo ou diretório pertence **r** (read) **w** (write).

O grupo em **verde** concede 1 permissão para qualquer outra pessoa que acessar o arquivo ou diretório **r** (read).

- **SISTEMA DE SEGURANÇA EM DIRETÓRIOS**

Os diretórios têm algumas características que se diferenciam dos arquivos, a primeira é que no início da coluna em vermelho temos um **d** e não um **–**

r – permite que usuários listem arquivos no diretório

w – usuários podem deletar e mover arquivos para o diretório

x – usuários tem permissão de acesso aos arquivos no diretório, indica que você também pode ler os arquivos dentro do diretório.

- **CHMOD – (CHANGING FILE MODE / MODIFICANDO PERMISSÕES)**

Symbol	Meaning
u	user
g	group
o	other
a	all
r	read
w	write (and delete)
x	execute (and access directory)
+	add permission
-	take away permission

(FONTE: http://www.ee.surrey.ac.uk/Teaching/Unix/unix5.html**)**

Para modificar permissões utiliza-se a sintaxe:

(Comando chmod) + (grupo alvo) + (+ adicionar / - remover / = atribuir) + (nome do arquivo/diretório)

Ex: **chmod a+rw lista.txt** – Neste exemplo todos/all **(a)** receberam **(+)** permissão para ler **(r)** e escrever **(w)** no arquivo **lista.txt**

- **PROCESSOS & JOBS**

Um processo é um programa em execução identificado por um identificador de processo único (**PID** / Process Identifier), pra obter informação com relação a processos e seus respectivos **PID** e status, utiliza-se o comando **ps**.

Um processo pode estar no foreground (rodando direto no terminal impedindo outras ações **fg**), no backgound (rodando em segundo plano **bg**), suspenso (pausado momentaneamente).

Alguns processos demoram para rodar no terminal tornando extremamente útil roda-lo no background e liberar o terminal para utilização.

Ex: rodando um processo em segundo plano ou em backgound (bg):

1. **sleep 100** (o terminal ficará 100 segundos aguardando o processo rodar para que você volte a utiliza-lo.)
2. **ctrl + z** (interrompe o processo sleep 100 – status: parado)
3. **bg** (enter) (sleep volta a executar em segundo plano)
4. **jobs** (enter) (comando para checar o status dos processos ativos)

Ex2: rodando um processo em segundo plano ou em backgound (bg):

1. **sleep 100&** (A utilização da simbologia 'and' & faz o processo rodar diretamente em background)
2. **jobs** (enter) (comando para checar o status dos processos ativos)

Ex3: Trazendo um processo para foreground (fg):

1. **jobs** (enter) (comando para checar o status dos processos ativos)
2. **[1] executando sleep 100 &** (processo ativo em bg [1])
3. **[2] executando sleep 100 &** (processo ativo em bg [2])
4. **fg %numero_do_job** (enter) (trará o processo escolhido para o foreground)

Ex5: Finalizando ou matando um processo:

Às vezes é necessário finalizar um processo (por exemplo quando um programa entra em loop), para isto utiliza-se o comando **kill**.

1. Processo em foreground: **ctrl + c** (finaliza o processo em foreground)
2. Processo em backgound: **kill %numero_do_job** (mata o processo em background)
3. Finalizando processo por PID/Process Identifier: **kill %numero_do_PID** (finaliza o processo através do PID / identificador de processo)

Command	Meaning
`ls -lag`	list access rights for all files
`chmod [options] file`	change access rights for named file
`command &`	run command in background
`^C`	kill the job running in the foreground
`^Z`	suspend the job running in the foreground
`bg`	background the suspended job
`jobs`	list current jobs
`fg %1`	foreground job number 1
`kill %1`	kill job number 1
`ps`	list current processes
`kill 26152`	kill process number 26152

(FONTE: http://www.ee.surrey.ac.uk/Teaching/Unix/unix5.html**)**