

# Pipes, Redirecionamento

Prof: Diógenes



Instituto Federal  
de Brasília



# Objetivos

- Usar o Pipe e Redirecionamento
- Expressões Regulares - REGEX



# Comando de Linha e Redirecionamento



# Pipes em Linha de Comando

- O caractere pipe | pode ser usado entre dois comandos para enviar a saída do primeiro comando como entrada para o segundo comando:
  - `ls /etc | head`
- A saída de `ls /etc` é enviada para a entrada do comando `head`



# Pipelines em linhas de comando

- Vários comandos podem ser combinados para formar pipelines. A ordem em que os comandos são adicionados ao pipeline pode afetar a saída:

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls -l /etc/ppp | nl | tail -5  
7 -rwxr-xr-x. 1 root root 1687 Apr 27 2012 ipv6-down  
8 -rwxr-xr-x. 1 root root 3196 Apr 27 2012 ipv6-up  
9 -rw-r--r--. 1 root root 5 Aug 22 2010 options  
10 -rw-----. 1 root root 77 Aug 22 2010 pap-secrets  
11 drwxr-xr-x. 2 root root 4096 Jun 22 2012 peers  
[sysadmin@localhost ~]$
```

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls -l /etc/ppp | tail -5 | nl  
1 -rwxr-xr-x. 1 root root 1687 Apr 27 2012 ipv6-down  
2 -rwxr-xr-x. 1 root root 3196 Apr 27 2012 ipv6-up  
3 -rw-r--r--. 1 root root 5 Aug 22 2010 options  
4 -rw-----. 1 root root 77 Aug 22 2010 pap-secrets  
5 drwxr-xr-x. 2 root root 4096 Jun 22 2012 peers  
[sysadmin@localhost ~]$
```



# Redirecionamento de I/O

- Três fluxos de entrada/saída (I/O) associados a cada comando:
  - A entrada padrão (STDIN) é normalmente fornecida pelo usuário através do teclado.
  - A saída Padrão (STDOUT) é a saída produzida pelo comando quando operando corretamente. STDOUT normalmente aparece na mesma janela de onde o comando foi executado.
  - O erro padrão (STERR) é o resultado produzido pelo comando quando ocorre um erro. STDERR normalmente aparece na mesma janela de onde o comando foi executado.



# Símbolos de redirecionamento de I/O

- Resumo dos possíveis direcionamentos possíveis com o shell bash:
  - **< /path/to/file** (redireciona STDIN do arquivo)
  - **> /path/to/file** (redireciona STDOUT sobrescrevendo o arquivo)
  - **>> /path/to/file** (redireciona STDOUT adicionando ao arquivo)
  - **2 > /path/to/file** (redireciona STDERR sobrescrevendo o arquivo)
  - **2 >> /path/to/file** (redireciona STDERR adicionando ao arquivo)
  - **&> /path/to/file** (redireciona STDERR e STDOUT sobrescrevendo o arquivo)
  - **&>> /path/to/file** (redireciona STDERR e STDOUT adicionando o arquivo)



# O Device Null

- O device null é representado pelo arquivo `/dev/null`. (Também conhecido como "Bit Bucket")
- Este arquivo é muito útil no redirecionamento de entrada e saída.
- Este arquivo serve dois propósitos:
  - qualquer saída redirecionada para `/dev/null` é descartada.
  - `/dev/null` pode ser usado para entrada para fornecer um fluxo de valores nulos.





# STDIN, STDOUT e STDERR



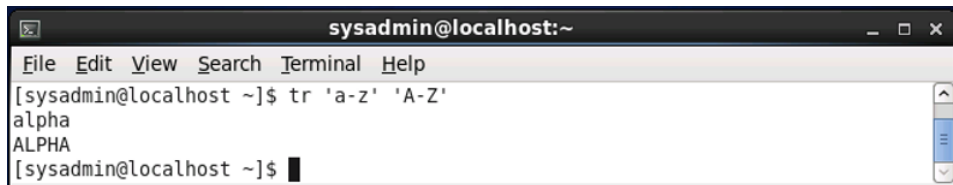
# STDIN ou 0

- A entrada padrão (STDIN) normalmente é fornecida pelo teclado, mas pode ser redirecionada com o símbolo `<`.
- STDIN pode ser lido por programas para obter dados para serem processados.
- Para sinalizar um programa que você deseja parar de fornecer dados pelo teclado via STDIN, digite **CTRL-D**.
- O comando `tr` lê seus dados de STDIN. Ele traduz de um conjunto de caracteres para outro.
- Se você fosse o usuário digitando os dados a serem traduzidos pelo comando `tr`, você digitaria **CTRL-D** quando terminar.



# STDIN do teclado

- No exemplo a seguir, o comando **tr** converte de minúsculas para maiúsculas depois que o usuário digitou o comando e pressionou **Enter**.
- Então, "alpha" foi digitado e **Enter** pressionado. Finalmente, o usuário digitou **CTRL-D**.

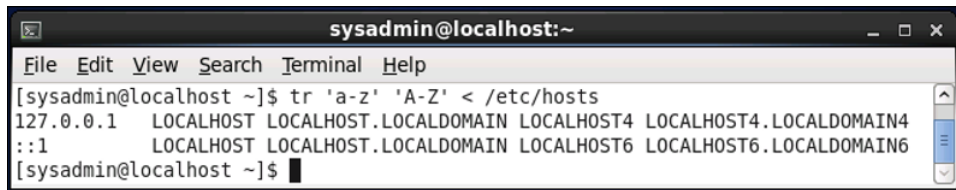


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tr 'a-z' 'A-Z'  
alpha  
ALPHA  
[sysadmin@localhost ~]$
```



## Redirecionando STDIN de arquivo

- O comando **tr** traduz de minúsculas para maiúsculas com o STDIN sendo redirecionado do arquivo `/etc/hosts`:



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tr 'a-z' 'A-Z' < /etc/hosts  
127.0.0.1    LOCALHOST LOCALHOST.LOCALDOMAIN LOCALHOST4 LOCALHOST4.LOCALDOMAIN4  
::1         LOCALHOST LOCALHOST.LOCALDOMAIN LOCALHOST6 LOCALHOST6.LOCALDOMAIN6  
[sysadmin@localhost ~]$
```



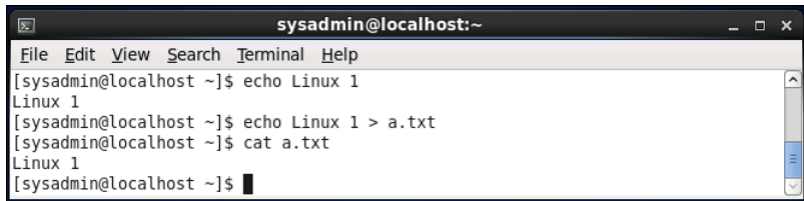
# STDOUT ou 1

- Standard Out (STDOUT) é a saída do comando quando operando corretamente.
- Geralmente é exibido na mesma janela em que o comando é executado.
- O comando **echo** é usado para imprimir mensagens no STDOUT.
- Ele pode ser usado para demonstrar como o STDOUT pode ser redirecionado, conforme mostrado no slide a seguir.



# Redirecionando STDOUT

- No exemplo abaixo, o comando `echo Linux 1` é executado e a saída aparece no STDOUT.
- Então, o comando `echo Linux 1 > a.txt` redireciona a saída para o arquivo `a.txt`.
- Finalmente, o comando `cat a.txt` envia o conteúdo do arquivo para STDOUT, então a saída é mostrada.

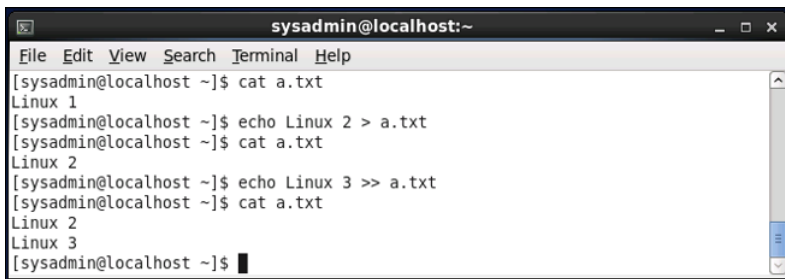
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following sequence of commands and output:

```
[sysadmin@localhost ~]$ echo Linux 1
Linux 1
[sysadmin@localhost ~]$ echo Linux 1 > a.txt
[sysadmin@localhost ~]$ cat a.txt
Linux 1
[sysadmin@localhost ~]$
```



# Adicionando no Redirecionando STDOUT

- Usar um único símbolo `>` para o redirecionamento STDOUT irá sobrescrever ou substituir o arquivo especificado.
- Usar um duplo símbolo `>>` para redirecionamento STDOUT criará um novo arquivo ou anexará (adiciona conteúdo) em um existente:

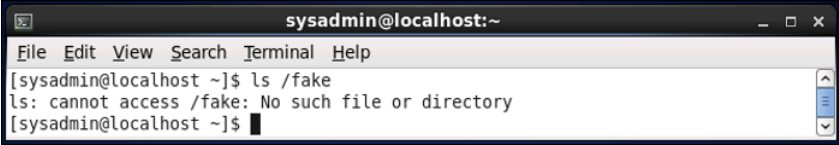


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ cat a.txt  
Linux 1  
[sysadmin@localhost ~]$ echo Linux 2 > a.txt  
[sysadmin@localhost ~]$ cat a.txt  
Linux 2  
[sysadmin@localhost ~]$ echo Linux 3 >> a.txt  
[sysadmin@localhost ~]$ cat a.txt  
Linux 2  
Linux 3  
[sysadmin@localhost ~]$
```



# STDERR ou 2

- Erro Padrão (STDERR) é a saída de um comando após um erro ter ocorrido.
- Normalmente é enviado para o console/terminal onde o comando é executado.
- `ls /fake` é um comando que fará com que um erro seja enviado para STDERR porque o arquivo `/fake` não existe.

A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is '[sysadmin@localhost ~]\$'. The command 'ls /fake' has been entered, and the output is 'ls: cannot access /fake: No such file or directory'. The prompt '[sysadmin@localhost ~]\$' is shown again with a cursor.

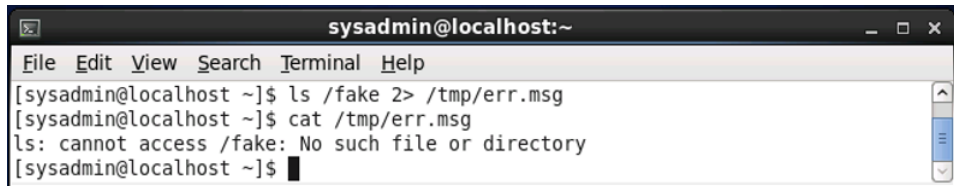
```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /fake  
ls: cannot access /fake: No such file or directory  
[sysadmin@localhost ~]$
```





## Redirecionando STDERR

- `ls /fake 2> /tmp/err.msg` é um comando que faria com que um erro fosse enviado para STDERR, que seria então redirecionado para o arquivo `/tmp/err.msg`.
- O comando `cat /tmp/err.msg` envia o conteúdo do arquivo para STDOUT para exibir o arquivo:

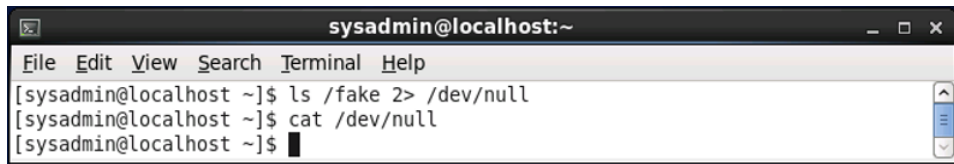
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'ls /fake 2> /tmp/err.msg' followed by 'cat /tmp/err.msg', which outputs the error message 'ls: cannot access /fake: No such file or directory'.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /fake 2> /tmp/err.msg  
[sysadmin@localhost ~]$ cat /tmp/err.msg  
ls: cannot access /fake: No such file or directory  
[sysadmin@localhost ~]$
```



# Eliminação de STDERR

- `ls /fake 2> /dev/null` é um comando que faria com que o STDERR fosse redirecionado para o arquivo `/dev/null`, com efeito eliminando a mensagem de erro.
- Observe: `cat /dev/null` não exibe saída visível.

A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows three lines of command execution: 1. '[sysadmin@localhost ~]\$ ls /fake 2> /dev/null' 2. '[sysadmin@localhost ~]\$ cat /dev/null' 3. '[sysadmin@localhost ~]\$' followed by a cursor. The second command produces no visible output, demonstrating redirection to /dev/null.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /fake 2> /dev/null  
[sysadmin@localhost ~]$ cat /dev/null  
[sysadmin@localhost ~]$
```



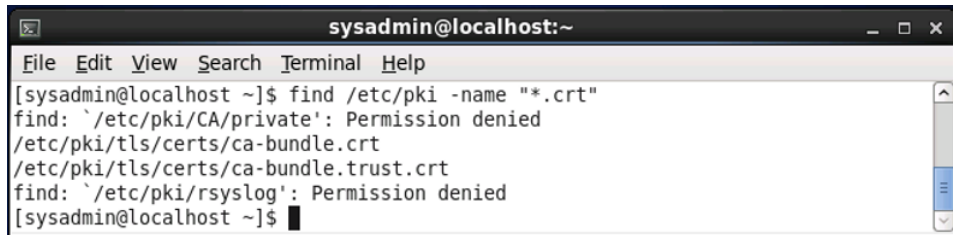
# Trabalhando com STDERR e STDOUT

- `find` é um comando que pesquisa o sistema de arquivos.
- Envia a saída para STDOUT quando localiza com êxito um arquivo que corresponda aos seus critérios.
- Envia a saída para STDERR quando não consegue acessar um diretório.
- O comando `find` será usado para demonstrar o redirecionamento de STDOUT e STDERR nos slides a seguir.
- Mais detalhes sobre o comando `find` aparecem mais adiante neste capítulo.



## Exemplo: STDERR e STDOUT

- O exemplo a seguir demonstra o comando **find** pesquisando recursivamente o diretório `/etc/pki` para quaisquer arquivos correspondentes a `"*.crt"`.
- Duas linhas de STDERR e duas linhas de mensagens STDOUT aparecem:

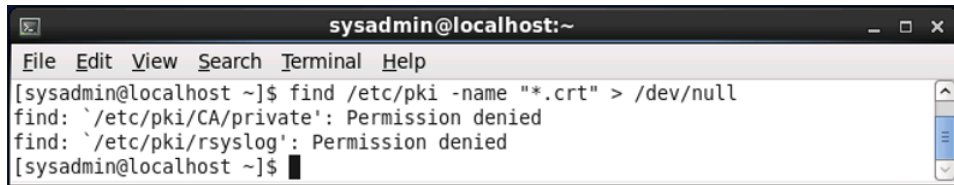


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc/pki -name "*.crt"  
find: `/etc/pki/CA/private': Permission denied  
/etc/pki/tls/certs/ca-bundle.crt  
/etc/pki/tls/certs/ca-bundle.trust.crt  
find: `/etc/pki/rsyslog': Permission denied  
[sysadmin@localhost ~]$
```



# Isolando STDERR

- No próximo exemplo, a saída STDOUT é redirecionada para o arquivo `/dev/null`, portanto, apenas a saída STDERR é enviada para a janela do terminal:

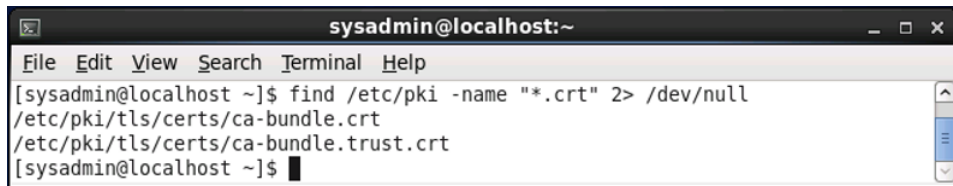


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc/pki -name "*.crt" > /dev/null  
find: `/etc/pki/CA/private': Permission denied  
find: `/etc/pki/rsyslog': Permission denied  
[sysadmin@localhost ~]$
```



# Isolando STDOUT

- No próximo exemplo, a saída STDERR é agora redirecionada para o arquivo `/dev/null`, logo apenas a saída STDOUT é enviada para a janela do terminal:

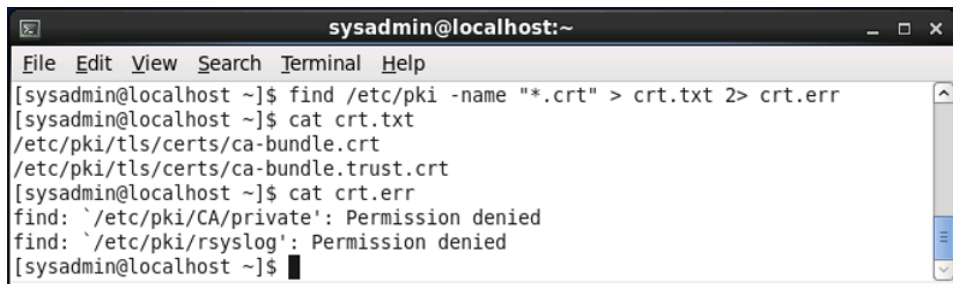
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'find /etc/pki -name "\*.cert" 2> /dev/null' being executed. The output lists two files: '/etc/pki/tls/certs/ca-bundle.crt' and '/etc/pki/tls/certs/ca-bundle.trust.crt'. The prompt '[sysadmin@localhost ~]\$' is shown at the bottom with a cursor.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc/pki -name "*.cert" 2> /dev/null  
/etc/pki/tls/certs/ca-bundle.crt  
/etc/pki/tls/certs/ca-bundle.trust.crt  
[sysadmin@localhost ~]$
```



## Redirecionando Múltiplos Fluxos Separadamente

- No próximo exemplo, a saída STDERR é enviada para o arquivo `crt.err` e a saída STDOUT é enviada para o arquivo `crt.txt`:

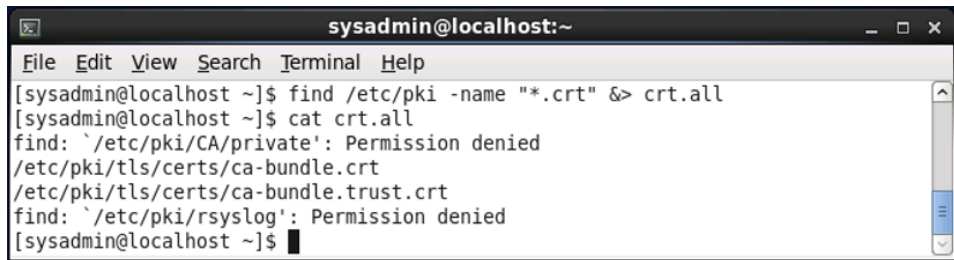


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc/pki -name "*.crt" > crt.txt 2> crt.err  
[sysadmin@localhost ~]$ cat crt.txt  
/etc/pki/tls/certs/ca-bundle.crt  
/etc/pki/tls/certs/ca-bundle.trust.crt  
[sysadmin@localhost ~]$ cat crt.err  
find: `/etc/pki/CA/private': Permission denied  
find: `/etc/pki/rsyslog': Permission denied  
[sysadmin@localhost ~]$
```



## Redirecionando Múltiplos Fluxos Combinados

- Neste exemplo, tanto STDOUT quanto STERR são redirecionados para o mesmo arquivo, crt.all:



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc/pki -name "*.cert" &> crt.all  
[sysadmin@localhost ~]$ cat crt.all  
find: `/etc/pki/CA/private': Permission denied  
/etc/pki/tls/certs/ca-bundle.crt  
/etc/pki/tls/certs/ca-bundle.trust.crt  
find: `/etc/pki/rsyslog': Permission denied  
[sysadmin@localhost ~]$
```





## Comando find



# Pesquisando com o comando find

- O sistema de arquivos tem centenas de diretórios com milhares de arquivos, tornando a localização de arquivos um desafio.
- O comando **find** é uma ferramenta poderosa para procurar arquivos de maneiras diferentes, incluindo:
  - nome
  - tamanho
  - data
  - proprietário



# Sintaxe do comando find

- O comando **find** possui a seguinte sintaxe:
  - **find [dir\_inicio] [op\_pesquisa] [critérios] [resultado]**
- Se o diretório inicial (dir\_inicio) não for especificado, o diretório atual será assumido.
- A opção de pesquisa (op\_pesquisa) é como a pesquisa será feita. Por exemplo, use a opção **-name** para pesquisar por nome.



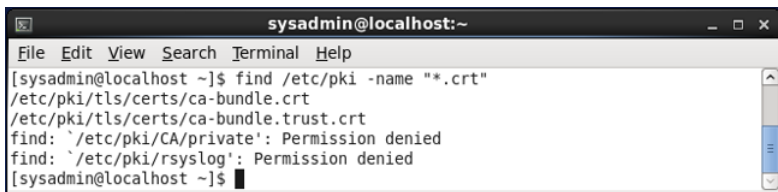
# Sintaxe do comando find (continuação)

- Os critérios de pesquisa (critérios) são os dados a serem usados com a opção de pesquisa. Portanto, se a opção de pesquisa fosse `-name`, o critério de pesquisa seria o nome do arquivo a ser encontrado.
- A opção de resultado (resultado) é padronizada como `-print`, que mostrará os nomes dos arquivos encontrados. Outras opções de resultado podem executar ações nos arquivos encontrados.



# Pesquisando pelo nome do arquivo

- Considere o seguinte comando:
  - `find /etc/pki -name '*.crt'`
- Começa a procurar recursivamente a partir do diretório `/etc/pki`
- Obtém como saída quaisquer nomes de arquivo que correspondam a `"*.crt"` (qualquer coisa que termine em `"*.crt"`).

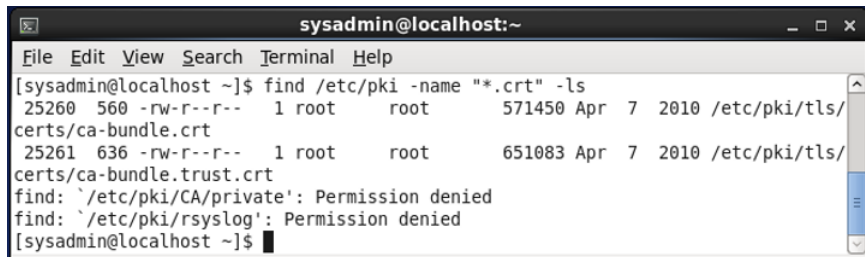
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command `find /etc/pki -name "*.crt"` being executed. The output lists two files: `/etc/pki/tls/certs/ca-bundle.crt` and `/etc/pki/tls/certs/ca-bundle.trust.crt`. It also shows two permission denied messages: `find: '/etc/pki/CA/private': Permission denied` and `find: '/etc/pki/rsyslog': Permission denied`. The prompt `[sysadmin@localhost ~]$` is shown at the end.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc/pki -name "*.crt"  
/etc/pki/tls/certs/ca-bundle.crt  
/etc/pki/tls/certs/ca-bundle.trust.crt  
find: '/etc/pki/CA/private': Permission denied  
find: '/etc/pki/rsyslog': Permission denied  
[sysadmin@localhost ~]$
```



## Mostrando os detalhes do arquivo

- A opção `-ls` criará uma saída semelhante ao comando `ls -l`.
- As colunas de saída são: inode, blocos usados, permissões, contagem de links, proprietário do usuário, proprietário do grupo, tamanho, data/hora e nome do arquivo.

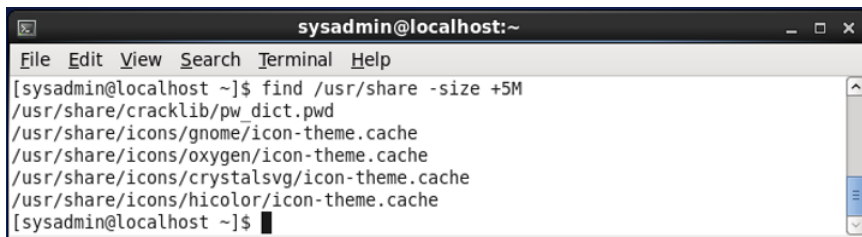


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc/pki -name "*.crt" -ls  
25260 560 -rw-r--r-- 1 root root 571450 Apr 7 2010 /etc/pki/tls/  
certs/ca-bundle.crt  
25261 636 -rw-r--r-- 1 root root 651083 Apr 7 2010 /etc/pki/tls/  
certs/ca-bundle.trust.crt  
find: `/etc/pki/CA/private': Permission denied  
find: `/etc/pki/rsyslog': Permission denied  
[sysadmin@localhost ~]$
```



## Pesquisando pelo tamanho do arquivo

- A opção **-size** pode ser usada por **find** para pesquisar por seu tamanho.
- Unidades grandes podem ser especificadas como K, M, G etc.
- Usando **+1M** significa mais de um megabyte.
- Usando **-1M** significa menos de um megabyte.



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /usr/share -size +5M  
/usr/share/cracklib/pw_dict.pwd  
/usr/share/icons/gnome/icon-theme.cache  
/usr/share/icons/oxygen/icon-theme.cache  
/usr/share/icons/crystalsvg/icon-theme.cache  
/usr/share/icons/hicolor/icon-theme.cache  
[sysadmin@localhost ~]$
```



# Opções úteis para o comando find

Opção	Exemplo	Significado
<code>-maxdepth</code>	<code>-maxdepth 1</code>	Somente pesquisa no diretório especificado e nos subdiretórios imediatos
<code>-group</code>	<code>-group payroll</code>	Encontra quaisquer arquivos pertencentes ao grupo payroll
<code>-iname</code>	<code>-iname hosts</code>	Pesquisa nomes de arquivos sem diferenciar maiúsculas e minúsculas
<code>-mmin</code>	<code>-mmin -10</code>	Encontra quaisquer arquivos modificados nos últimos 10 minutos ou menos
<code>-type</code>	<code>-type f</code>	Encontra somente arquivos regulares
<code>-user</code>	<code>-user bob</code>	Encontra quaisquer arquivos pertencentes ao usuário bob





## Comando less



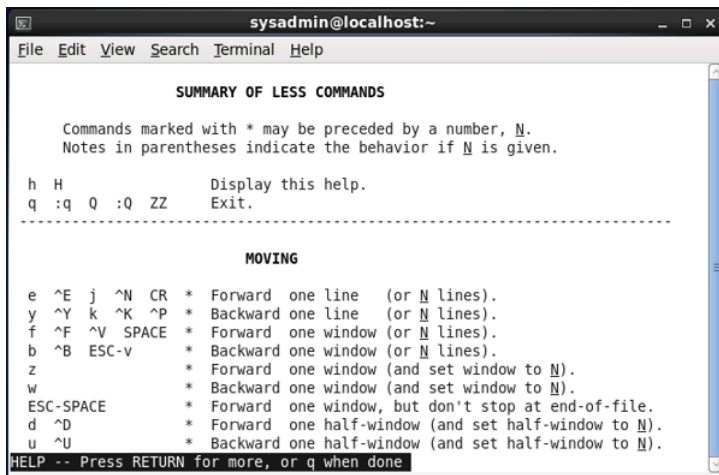
# Exibindo arquivos com o comando less

- O comando **less** é um comando de pager projetado para exibir apenas uma página de dados por vez.
- O comando
- **more** é outro comando de pager que possui menos recursos que o comando **less**.
- Ambos os comandos permitem ao usuário mover-se para frente e para trás com comandos de movimento para visualizar uma página por vez.



# A tela de ajuda em less

- Uma vez no programa **less**, pressionar a tecla “**h**” exibirá a tela de ajuda:



The screenshot shows a terminal window titled 'sysadmin@localhost:~'. The window contains the 'less' command help text. At the top is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The main text is titled 'SUMMARY OF LESS COMMANDS' and explains that commands marked with an asterisk can be preceded by a number 'N'. It lists two basic commands: 'h H' for displaying help and 'q :q Q :Q ZZ' for exiting. A dashed line separates this from the 'MOVING' section, which lists navigation commands like '^E j ^N CR' for forward one line, '^Y k ^K ^P' for backward one line, '^F ^V SPACE' for forward one window, and '^B ESC-v' for backward one window. It also includes commands for setting window size (z, w) and half-window size (d, u). The bottom line says 'HELP -- Press RETURN for more, or q when done'.

```
sysadmin@localhost:~
File Edit View Search Terminal Help

SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, N.
Notes in parentheses indicate the behavior if N is given.

h H          Display this help.
q :q Q :Q ZZ  Exit.
-----

MOVING

e ^E j ^N CR * Forward one line (or N lines).
y ^Y k ^K ^P * Backward one line (or N lines).
f ^F ^V SPACE * Forward one window (or N lines).
b ^B ESC-v    * Backward one window (or N lines).
z             * Forward one window (and set window to N).
w             * Backward one window (and set window to N).
ESC-SPACE    * Forward one window, but don't stop at end-of-file.
d ^D         * Forward one half-window (and set half-window to N).
u ^U         * Backward one half-window (and set half-window to N).
HELP -- Press RETURN for more, or q when done
```



# Comandos de Movimento do less

- Como visto na tela de ajuda, o comando **less** possui muitos comandos de movimento. Os comandos mais comuns são:

Movimento	Tecla
Avançar janela	Barra de espaço
Voltar janela	B
Avançar linha	Enter
Sair	q
Ajuda	h



# Comandos de pesquisa do less

- Digite **/** para pesquisar do cursor até o final do arquivo.
- Digite **?** para pesquisar do cursor ao início do arquivo.
- Digite o padrão de pesquisa para pesquisar e pressione **Enter**.
- Se mais de uma correspondência for encontrada, pressione **n** para ir para a próxima correspondência ou **N** para ir para a correspondência anterior.

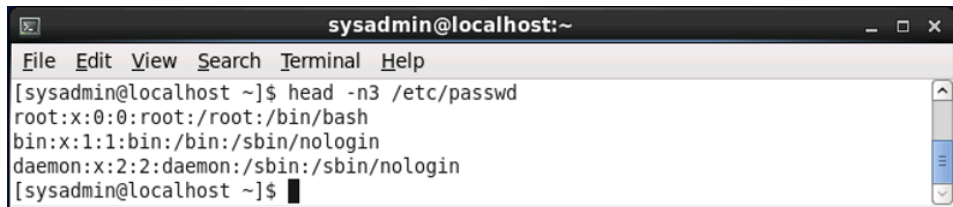


head ou tail



# Filtrando com o head

- O comando **head** exibe as dez primeiras linhas de um arquivo por padrão.
- A opção **-n** permite que o número de linhas a serem exibidas seja especificado.

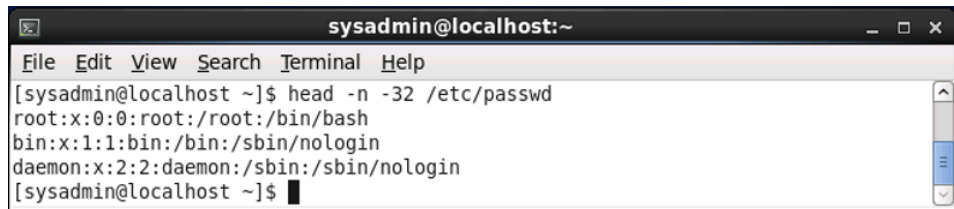
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[sysadmin@localhost ~]\$ head -n3 /etc/passwd' has been executed, resulting in the following output:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

The prompt '[sysadmin@localhost ~]\$' is visible at the bottom of the terminal.

## head com linhas negativas

- Normalmente, o comando **head** exibe o número de linhas especificadas na parte superior do arquivo.
- Usando **-n** com um valor negativo, indica quantas linhas da parte inferior para não mostrar.
- Este exemplo mostra todas as linhas de `/etc/passwd`, exceto as últimas trinta e duas.



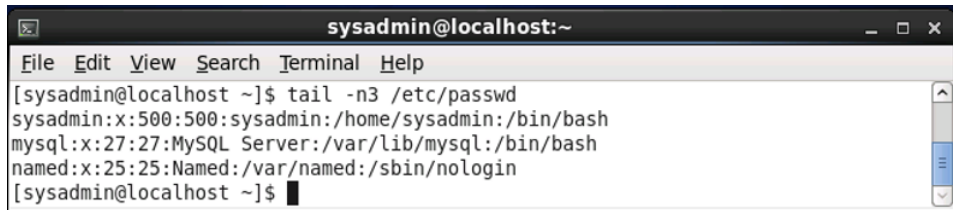
```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ head -n -32 /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin  
daemon:x:2:2:daemon:/sbin:/sbin/nologin  
[sysadmin@localhost ~]$
```





# Filtrando com tail

- O comando **tail** exibe as últimas dez linhas de um arquivo por padrão.
- A opção **-n** permite que o número de linhas a serem exibidas seja especificado:

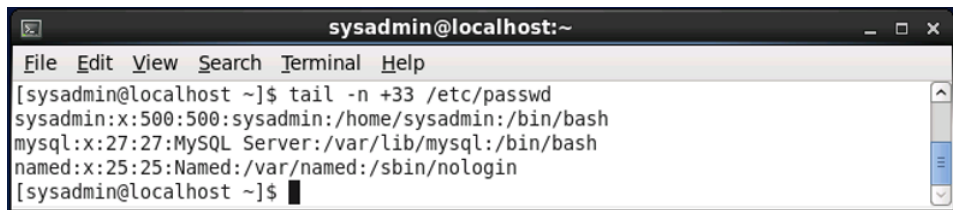


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tail -n3 /etc/passwd  
sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash  
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash  
named:x:25:25:Named:/var/named:/sbin/nologin  
[sysadmin@localhost ~]$
```



## tail com linhas positivas

- Se a opção **-n** especifica o número de linhas a serem exibidas com um prefixo mais (+), o comando **tail** interpreta que significa exibir desse número de linha até o final do arquivo:

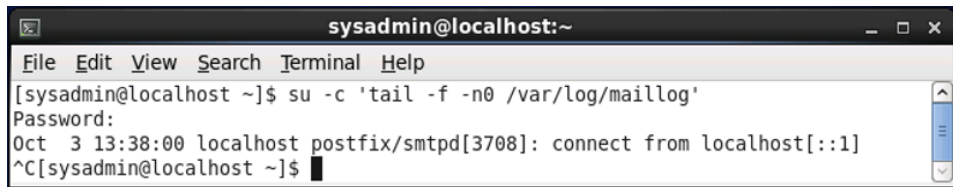
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[sysadmin@localhost ~]\$ tail -n +33 /etc/passwd' has been executed. The output shows the last three lines of the /etc/passwd file: 'sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash', 'mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash', and 'named:x:25:25:Named:/var/named:/sbin/nologin'. The prompt '[sysadmin@localhost ~]\$' is visible at the bottom.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tail -n +33 /etc/passwd  
sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash  
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash  
named:x:25:25:Named:/var/named:/sbin/nologin  
[sysadmin@localhost ~]$
```



# Monitorando com tail

- O comando **tail** é capaz de monitorar as alterações de um arquivo e imprimi-las conforme elas ocorrem usando a opção **-f**.
- Os administradores de sistema freqüentemente monitoram arquivos de log para solucionar problemas do sistema.
- O usuário deve terminar o comando **tail** ao seguir com a opção **{f}** usando **CTRL-C**.



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ su -c 'tail -f -n0 /var/log/maillog'  
Password:  
Oct  3 13:38:00 localhost postfix/smtpd[3708]: connect from localhost[::1]  
^C[sysadmin@localhost ~]$
```



## Comando sort



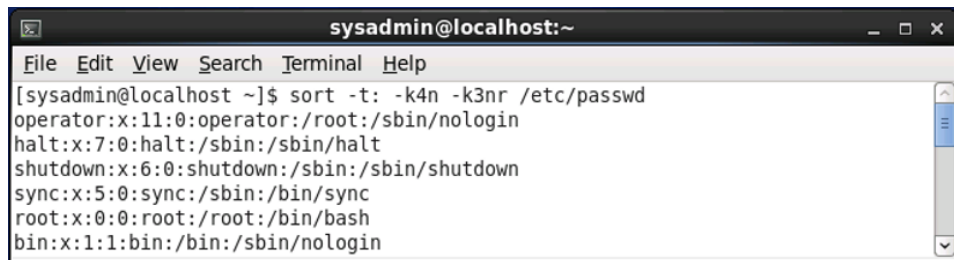
# Classificando arquivos ou entrada

- O comando **sort** irá reorganizar suas linhas de saída de acordo com um ou mais campos que você especificar para classificação.
- Os campos são separados por espaço em branco, embora com a opção **-t** você possa especificar o delimitador.
- A classificação padrão está em ordem crescente, mas você pode usar a opção **-r** para reverter a classificação de um campo.
- A classificação padrão é uma classificação de dicionário, mas você pode usar a opção **-n** para torná-la uma classificação numérica.



## Exemplo do sort

- No exemplo a seguir, o arquivo `/etc/passwd` é classificado usando um caractere `:` como um delimitador, pelo quarto campo numericamente e depois o terceiro campo numericamente ao contrário:



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ sort -t: -k4n -k3nr /etc/passwd  
operator:x:11:0:operator:/root:/sbin/nologin  
halt:x:7:0:halt:/sbin:/sbin/halt  
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown  
sync:x:5:0:sync:/sbin:/bin/sync  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin
```



## Estatísticas do arquivo



# Estatísticas de arquivos com o comando wc

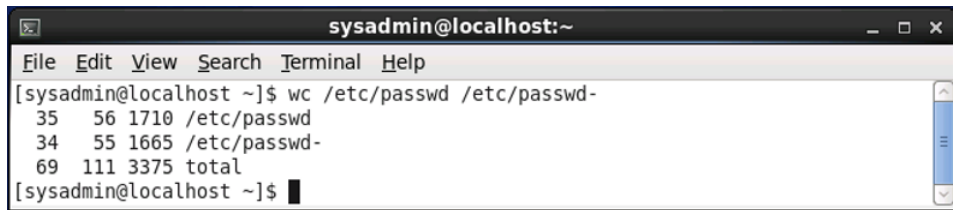
- O comando **wc** gera até três estatísticas para cada arquivo que é dado como argumento.
- Por padrão, o **wc** exibe as linhas, palavras e bytes contidos em cada arquivo.
- Se fornecido mais de um arquivo, ele também calcula os totais de todos os arquivos.
- Para visualizar estatísticas individuais, especifique **-l** para linhas, **-w** para palavras ou **-c** para bytes.





## Exemplo do comando wc

- Para analisar o número de linhas, palavras e bytes nos arquivos `/etc/passwd` e `/etc/passwd-`, o seguinte comando **wc** pode ser executado:



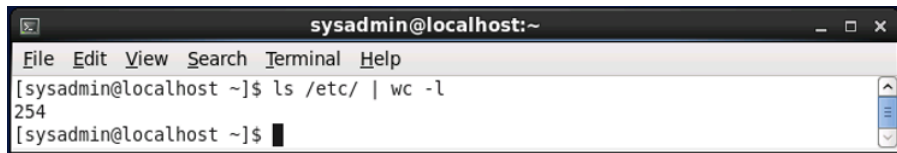
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command '[sysadmin@localhost ~]\$ wc /etc/passwd /etc/passwd-' has been executed, resulting in the following output:

```
[sysadmin@localhost ~]$ wc /etc/passwd /etc/passwd-
 35   56 1710 /etc/passwd
 34   55 1665 /etc/passwd-
 69  111 3375 total
[sysadmin@localhost ~]$
```



## Usando wc com pipes

- O comando **wc** é freqüentemente usado com pipes para que a saída de um comando possa ser analisada.
- Usando **wc -l** como o comando final no pipe contará quantas linhas de saída foram produzidas.
- Por exemplo, para determinar quantos arquivos e diretórios estão no diretório **/etc**, você poderia executar: **ls /etc | wc -l**



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /etc/ | wc -l  
254  
[sysadmin@localhost ~]$
```



## Comando cut



# Filtrando com o comando cut

- Se você quiser extrair colunas de texto, o comando **cut** fornecerá duas técnicas simples:
  - Por delimitador, onde o espaço em branco é o padrão. A opção **-d** permite especificar outros delimitadores e **-f** é usado para indicar quais campos serão extraídos.
  - Por posição de caractere, usando a opção **-c** com o intervalo da coluna a ser extraída.

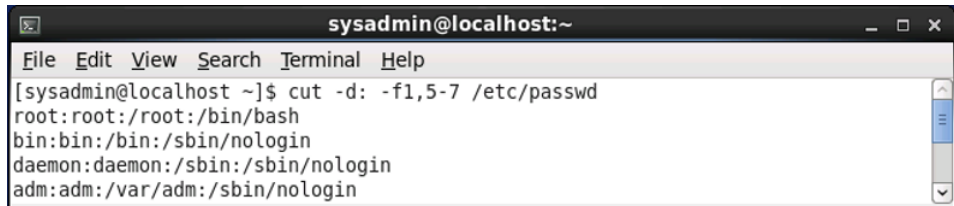


## Exemplo do comando cut

- O arquivo `/etc/passwd` é delimitado por dois-pontos com estes campos:

`conta:senha:UID:GID:GECOS:diretório:shell`

- Para extrair o primeiro e o quinto a sétimo campos:



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ cut -d: -f1,5-7 /etc/passwd  
root:root:/root:/bin/bash  
bin:bin:/bin:/sbin/nologin  
daemon:daemon:/sbin:/sbin/nologin  
adm:adm:/var/adm:/sbin/nologin
```

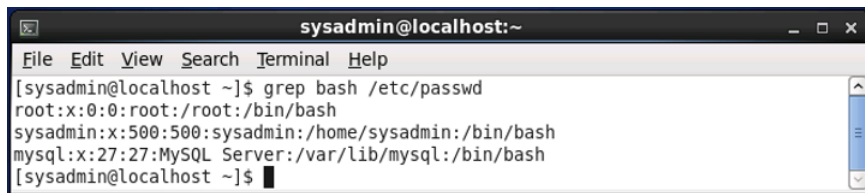


# Comando grep



## Filtrando com o comando grep

- O comando **grep** pode ser usado para filtrar a entrada padrão ou o conteúdo de um arquivo para linhas que correspondam a um padrão especificado.
- Se você quiser ver onde um padrão, ou talvez uma palavra, aparece em um arquivo, o comando **grep** é útil para essa finalidade.

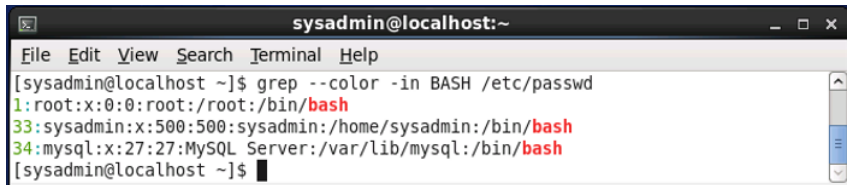


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep bash /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash  
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash  
[sysadmin@localhost ~]$
```



# Opções do comando grep

Opção	Propósito
--color	Colore as correspondências encontradas
-v	Reverte (nega) a correspondência
-c	Conta as correspondências
-n	Enumera as linhas das correspondências
-l	Lista os arquivos das correspondências
-i	Ignora maiúsculas/minúsculas
-w	Padrão de correspondência como uma palavra



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color -in BASH /etc/passwd  
1:root:x:0:0:root:/root:/bin/bash  
33:sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash  
34:mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash  
[sysadmin@localhost ~]$
```





# Expressões regulares básicas



# Expressões Regulares Básicas

- As Expressões Regulares Básicas (Basic Regular Expression - BRE) podem ser usadas com o comando **grep** sem precisar de uma opção para usá-las (diferentemente das Expressões Regulares Estendidas mostradas mais tarde).
- As expressões regulares mais simples são apenas caracteres alfabéticos ou numéricos correspondentes.
- A barra invertida \ pode ser usada para escapar do significado de metacaracteres de expressões regulares, incluindo a própria barra invertida.



# BRE - O sinal de ponto

- O caractere `.` (ponto) corresponde exatamente a um caractere.
- O exemplo abaixo mostra o comando `grep` que corresponde ao `'a'` seguido por dois caracteres.
- Os resultados mostram que corresponde a `'abc'`.

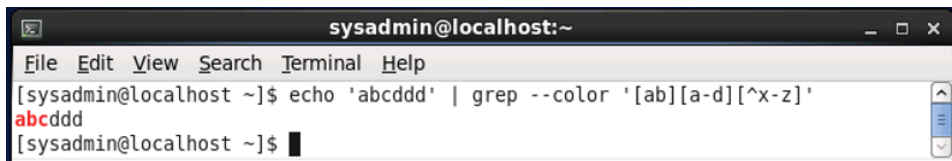


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo 'abcddd' | grep --color 'a..'br/>abcddd  
[sysadmin@localhost ~]$
```



# BRE - Colchetes

- Os caracteres `[ ]` (colchetes) são usados para corresponder exatamente a um caractere.
- Os caracteres podem ser listados ou dados como um intervalo.
- Se o primeiro caractere listado for `^` (circunflexo), significa que não são os caracteres entre colchetes.

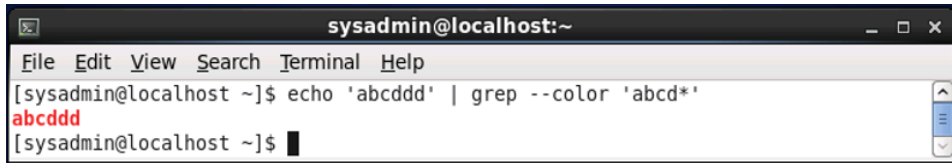


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo 'abcddd' | grep --color '[ab][a-d][^x-z]'  
abcddd  
[sysadmin@localhost ~]$
```



# BRE - Asterisco

- O caractere **\*** (asterisco) corresponderá a zero ou mais do caractere anterior.
- A correspondência “a\*” não é muito útil porque pode corresponder a zero a (corresponde a todas as linhas).
- Combinar “abcd\*” seria mais útil, já que você precisaria de um “abc” seguido por zero ou mais d’s.

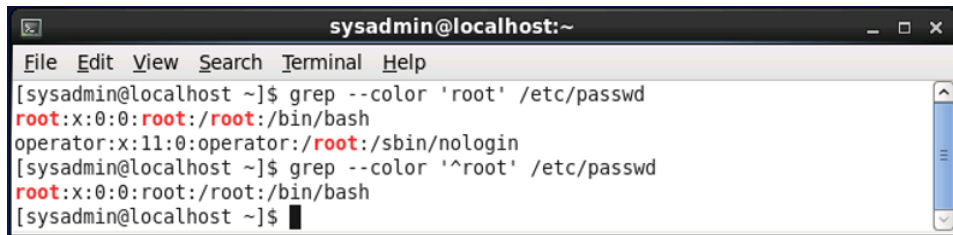


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo 'abcddd' | grep --color 'abcd*'  
abcddd  
[sysadmin@localhost ~]$
```



# BRE - Circunflexo

- O caractere `^` (circunflexo), quando aparece no início do padrão, significa que o padrão deve aparecer no início da linha.
- O `^` se não estiver no início de um padrão corresponde a si mesmo.

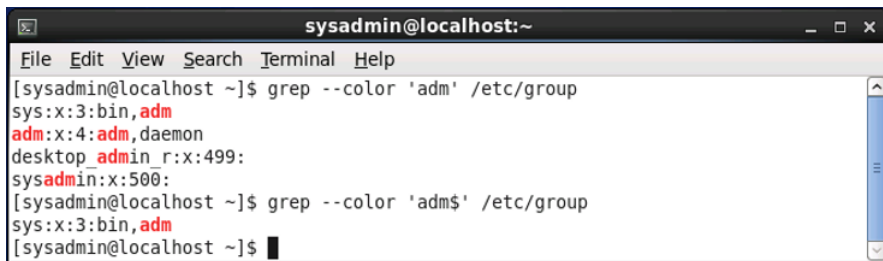


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color 'root' /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
operator:x:11:0:operator:/sbin/nologin  
[sysadmin@localhost ~]$ grep --color '^root' /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
[sysadmin@localhost ~]$
```



# BRE - Cifrão

- O **\$** (cifrão), quando aparece no final do padrão, significa que o padrão deve aparecer no final da linha.
- O **\$** se não estiver no fim de um padrão corresponde a si mesmo.

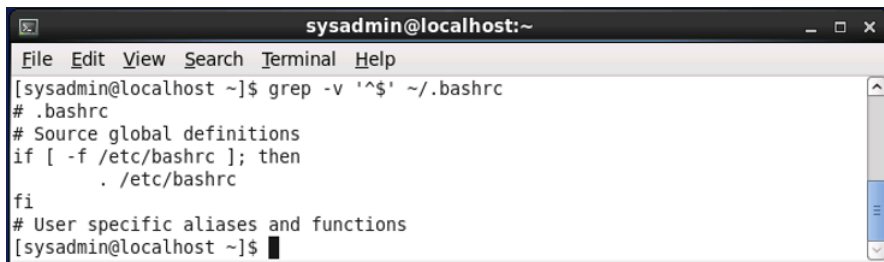


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color 'adm' /etc/group  
sys:x:3:bin,adm  
adm:x:4:adm,daemon  
desktop_admin_r:x:499:  
sysadmin:x:500:  
[sysadmin@localhost ~]$ grep --color 'adm$' /etc/group  
sys:x:3:bin,adm  
[sysadmin@localhost ~]$
```



# BRE - Combinando circunflexo e cifrão

- Combinando ambos caracteres `^` e `$` permite duas correspondências especiais:
  - `'^$',` corresponde a uma linha em branco.
  - `^padrão$` corresponde se uma linha contém somente o "padrão".



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep -v '^$' ~/.bashrc  
# .bashrc  
# Source global definitions  
if [ -f /etc/bashrc ]; then  
    . /etc/bashrc  
fi  
# User specific aliases and functions  
[sysadmin@localhost ~]$
```





# Expressões regulares estendidas



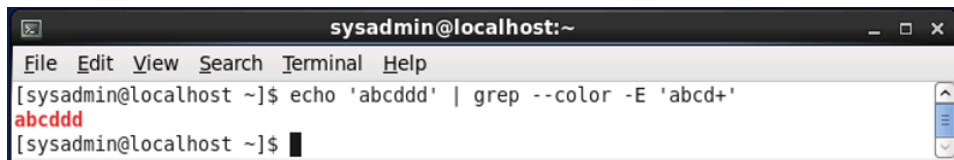
# Expressões Regulares Estendidas

- O uso de expressões regulares estendidas (Extended Regular Expression - ERE) requer a opção **-E** ao usar o comando **grep**.
- Expressões regulares estendidas podem ser combinadas com expressões regulares básicas.
- Os caracteres seguintes são ERE: **?**, **+** e **|**



## ERE - Caractere +

- O caractere **+** (mais) corresponderá a um ou mais do caractere anterior.
- A correspondência **‘a+’** é útil porque pode corresponder a um ou mais a's, garantindo que apenas as linhas com pelo menos um "a" são correspondidas.

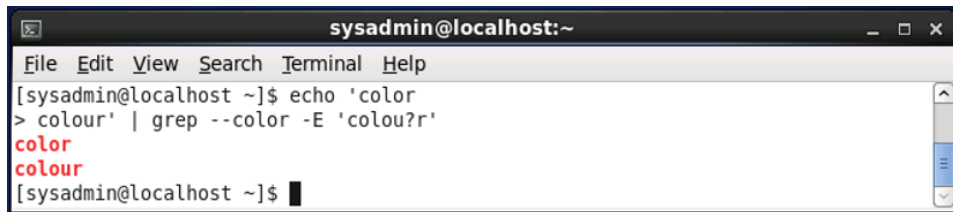


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo 'abcddd' | grep --color -E 'abcd+'  
abcddd  
[sysadmin@localhost ~]$
```



# ERE - Caractere ?

- O **?** (ponto de interrogação) irá opcionalmente corresponder a um dos caracteres anteriores.
- O **?** caractere é útil para caracteres correspondentes que aparecem apenas ocasionalmente em uma palavra. O exemplo a seguir ilustra isso:

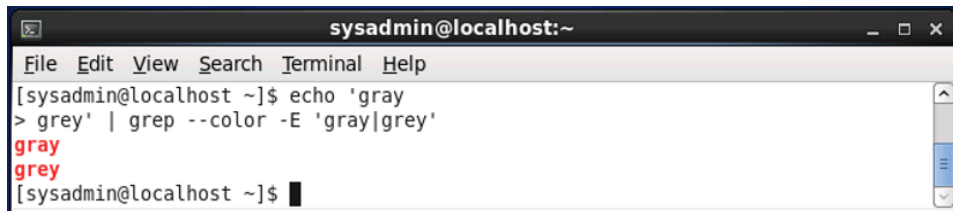


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo 'color  
> colour' | grep --color -E 'colou?r'  
color  
colour  
[sysadmin@localhost ~]$
```



# ERE - Caractere barra vertical

- O `|` (barra vertical) irá agir como um operador “ou” entre duas expressões regulares.
- Esse operador de alternância é útil para poder corresponder a vários padrões:

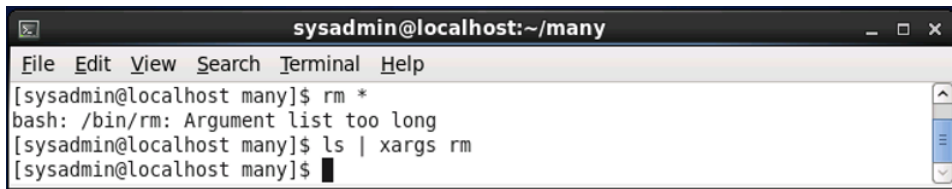


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo 'gray  
> grey' | grep --color -E 'gray|grey'  
gray  
grey  
[sysadmin@localhost ~]$
```



# O comando xargs

- O comando **xargs** ajuda os conjuntos de comandos com pipes complexos a executarem de forma mais eficiente
- Ele tenta construir a linha de comando mais longa possível com tantos argumentos quanto possível
- É comum precisarmos passar uma lista de itens para um comando que é maior do que o shell pode trabalhar.
- O comando **xargs** pode ser utilizado para efetuar uma divisão dessa lista em sub-listas menores, e passar os argumentos ao comando requisitado, em partes.



```
sysadmin@localhost:~/many
File Edit View Search Terminal Help
[sysadmin@localhost many]$ rm *
bash: /bin/rm: Argument list too long
[sysadmin@localhost many]$ ls | xargs rm
[sysadmin@localhost many]$
```

The image shows a terminal window titled 'sysadmin@localhost:~/many'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal output shows a failed command 'rm \*' with the error 'bash: /bin/rm: Argument list too long'. This is followed by the successful command 'ls | xargs rm', which removes all files in the directory. The prompt returns to '[sysadmin@localhost many]\$'.