



Comandos

para Execução de Programas

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro

Prof. Edwar Saliba Júnior

Agosto / 2017



Executando um Programa/Comando

- Para executar um comando, é necessário que se tenha permissão de execução e que este esteja no caminho de procura de arquivos, `path`;
- No aviso de comando `#` (`root`) ou `$` (`usuário`), digite o nome do comando e tecla `Enter`. O programa/comando é executado e receberá um número de identificação (chamado de `PID - Process Identification`), este número é útil para identificar o processo no sistema e assim ter um controle sobre sua execução;
- Todo o programa executado no GNU/Linux roda sob o controle das permissões de acesso;
- Exemplos de comandos:
 - `ls`
 - `df`
 - `pwd`



path

- `path` é o caminho de procura dos arquivos/comandos executáveis. O `path` (caminho) é armazenado na variável de ambiente `PATH`. Você pode ver o conteúdo desta variável com o comando `echo $PATH`;
- Por exemplo, o caminho `/usr/local/bin:/usr/bin:/bin:/usr/bin/x11` significa que se você digitar o comando `ls`, o interpretador de comandos iniciará a procura do programa `ls` no diretório `/usr/local/bin`, caso não encontre o arquivo neste diretório, então, ele inicia a procura em `/usr/bin` e assim por diante, até que encontre o arquivo procurado;
- Caso o interpretador de comandos chegue até o último diretório do `path` e não encontre o arquivo/comando digitado, é mostrada a seguinte mensagem:

```
bash: ls: command not found (comando não encontrado).
```



path

- O caminho de diretórios vem configurado na instalação do GNU/Linux, mas pode ser alterado no arquivo `/etc/profile`. Caso deseje alterar o caminho para todos os usuários, este arquivo é o melhor lugar, pois ele é lido por todos os usuários no momento do login;
- Caso um arquivo/comando não esteja localizado em nenhum dos diretórios do `path`, você deve executá-lo usando um `./` na frente do comando;
- Se deseja alterar o `path` para um único usuário, modifique o arquivo `.bash_profile` ou `.profile` (depende da versão do S.O. e da distribuição que estiver sendo usada) em seu diretório de usuário (home).



Tipos de Execução

- Um programa pode ser executado de duas formas:
 - Primeiro Plano - também chamado de *foreground*. Quando você deve esperar o término da execução de um programa para executar um novo comando. Somente é mostrado o aviso de comando após o término de execução do comando/programa;
 - Segundo Plano - também chamado de *background*. Quando você não precisa esperar o término da execução de um programa para executar um novo comando. Após iniciar um programa em *background*, é mostrado um número `PID` (identificação do Processo) e o aviso de comando é novamente mostrado, permitindo o uso normal do sistema;
- O programa executado em *background* continua sendo executado internamente. Após ser concluído, o sistema retorna uma mensagem de pronto acompanhado do número `PID` do processo que terminou.



Tipos de Execução

- Para iniciar um programa em primeiro plano, basta digitar seu nome normalmente. Para iniciar um programa em segundo plano, acrescente o caractere "&" após o final do comando;
- Existem também os seguintes comandos para manipular processos em segundo plano:

`fg` - coloca o processo em primeiro plano (*foreground*);

`bg` - coloca o processo em segundo plano (*background*);

`jobs` - lista os processos correndo em segundo plano, ou apenas pausados/parados;

`&` - o caractere "e comercial" colocado ao final de um comando, faz com que o mesmo rode em segundo plano desde o início;

`ctrl+c` :: mata o processo atual ativo e

`ctrl+z` :: pausa/para o processo atual ativo.

- OBS: mesmo que um usuário execute um programa em segundo plano e saia do sistema, o programa continuará sendo executado até que seja concluído ou finalizado pelo usuário que iniciou a execução (ou pelo usuário root);
- Exemplo: [Execução em primeiro e segundo plano..](#)



Executando Programas em Sequência

- Os comandos podem ser executados em sequência, um após o término do outro, se os separarmos com ";".

Por exemplo: `echo primeiro;echo segundo;echo terceiro`



Comando `ps`

- Algumas vezes é útil ver quais processos estão sendo executados no computador. O comando `ps` faz isto, e também nos mostra qual usuário executou o programa, hora que o processo foi iniciado, etc.

`ps [opções]`

- Onde:

`opções`

`a`

Mostra os processos criados por você e de outros usuários do sistema;

`x`

Mostra processos que não são controlados pelo terminal;

`u`

Mostra o nome de usuário que iniciou o processo e hora em que o processo foi iniciado;

`m`

Mostra a memória ocupada por cada processo em execução;

`f`

Mostra a árvore de execução de comandos (comandos que são chamados por outros comandos).



Comando `ps`

- Continua...

`opções`

`e`

Mostra variáveis de ambiente no momento da inicialização do processo.

`w`

Mostra a continuação da linha atual na próxima linha ao invés de cortar o restante que não couber na tela.

- As opções acima podem ser combinadas para resultar em uma listagem mais completa. Você também pode usar pipes "`|`" para filtrar a saída do comando `ps`;
- Ao contrário de outros comandos, o comando `ps` não precisa do hífen "`-`" para especificar as opções. Isto porque ele não utiliza opções longas e não usa parâmetros;
- Exemplos:

```
ps
```

```
ps ax|grep inetd
```

```
ps auxf
```

```
ps auxw
```



Comando top

- Mostra os programas em execução ativos, parados, tempo usado na CPU, detalhes sobre o uso da memória RAM, Swap, disponibilidade para execução de programas no sistema, etc.;
- `top` é um programa que continua em execução mostrando continuamente os processos que estão rodando em seu computador e os recursos utilizados por eles. Para sair do `top`, pressione a tecla `q`;

`top [opções]`

- Onde, `opções` pode ser substituído por:

`-d [tempo]`

Atualiza a tela após o `[tempo]` (em segundos);

`-s`

Diz ao `top` para ser executado em modo seguro;

`-i`

Inicia o `top` ignorando o tempo de processos zumbis;

`-c`

Mostra a linha de comando ao invés do nome do programa.



Comando top

- A ajuda sobre o `top` pode ser obtida dentro do programa pressionando a tecla `h` ou pela página de manual (`man top`);
- Abaixo algumas teclas úteis:
 - espaço - atualiza imediatamente a tela;
 - CTRL+L - apaga e atualiza a tela;
 - h - mostra a tela de ajuda do programa. É mostrado todas as teclas que podem ser usadas com o `top`;
 - i - ignora o tempo ocioso de processos zumbis;
 - q - sai do programa;
 - k - finaliza um processo - semelhante ao comando `kill`. Você será perguntado pelo número de identificação do processo (PID). Este comando não estará disponível caso esteja usando o `top` com a opção `-s`.
 - n - muda o número de linhas mostradas na tela. Se 0 for especificado, será usada toda a tela para listagem de processos.



Interrupção de Processos

- Para cancelar a execução de algum processo rodando em primeiro plano, basta pressionar as teclas `CTRL+C`;
- A execução do programa será cancelada e será mostrado o aviso de comando;
- Você também pode usar o comando `kill`, para interromper um processo sendo executado.



Interrupção Momentânea de um Processo

- Para parar a execução de um processo rodando em primeiro plano, basta pressionar as teclas CTRL+Z. O programa em execução será pausado e será mostrado o número de seu `job` e o aviso de comando;
- Para retornar a execução de um comando pausado, use `fg` OU `bg`;
- O programa permanece na memória no ponto de processamento em que parou quando ele é interrompido. Você pode usar outros comandos ou rodar outros programas enquanto o programa atual está interrompido.



Comando jobs

- O comando `jobs` mostra os processos que estão parados ou rodando em segundo plano. Processos em segundo plano são iniciados usando o símbolo "&" no final da linha de comando ou através do comando `bg`;

`jobs`

- O número de identificação de cada processo parado ou em segundo plano (`job`), é usado com os comandos `fg` e `bg`. Um processo interrompido pode ser finalizado usando-se o comando `kill %[num]`, onde `[num]` é o número do processo obtido pelo comando `jobs`.



Comando jobs

- Exemplo:
 - abra o terminal e digite o comando os comandos a seguir:
jobs
top
CTRL+Z
jobs
bg
- O que aconteceu?



Comando `fg`

- Permite fazer um programa rodando em segundo plano ou parado, rodar em primeiro plano. Você deve usar o comando `jobs` para pegar o número do processo rodando em segundo plano ou interrompida, este número será passado ao comando `fg` para ativá-lo em primeiro plano;

`fg [número]`

- Onde `[número]` é o número obtido através do comando `jobs`;
- Caso seja usado sem parâmetros, o `fg` utilizará o último programa interrompido (o maior número obtido com o comando `jobs`);
- Exemplo: `fg 1`



Comando `bg`

- Permite fazer um programa rodando em primeiro plano ou parado, rodar em segundo plano. Para fazer um programa em primeiro plano rodar em segundo, é necessário primeiro interromper a execução do comando com `CTRL+Z`, será mostrado o número da tarefa interrompida, use este número com o comando `bg` para iniciar a execução do comando em segundo plano;

`bg [número]`

- Onde: `[número]` é o número do programa obtido com o pressionamento das teclas `CTRL+Z` ou através do comando `jobs`.



Comando `kill`

- Permite enviar um sinal a um comando/programa. Caso seja usado sem parâmetros, o `kill` enviará um sinal de término ao processo sendo executado;

```
kill [opções] [sinal] [número]
```

- Onde:

`número`

É o número de identificação do processo obtido com o comando `ps`. Também pode ser o número após o sinal de `%` obtido pelo comando `jobs` para matar uma tarefa interrompida;

`sinal`

Sinal que será enviado ao processo. Se omitido usa `-15` como padrão;

`opções`

`-9`

Envia um sinal de destruição ao processo ou programa. Ele é terminado imediatamente sem chances de salvar os dados ou apagar os arquivos temporários criados por ele.

- Você precisa ser o dono do processo ou o usuário `root` para terminá-lo ou destruí-lo. Você pode verificar se o processo foi finalizado através do comando `ps`.

- Exemplo:

```
kill 500
```

```
kill -9 500
```

```
kill %1.
```



Comando `killall`

- Permite finalizar processos através do nome.

```
killall [opções] [sinal] [processo]
```

- Onde:

`processo`

Nome do processo que deseja finalizar;

`sinal`

Sinal que será enviado ao processo (pode ser obtido usando a opção `-i`);

`opções`

`-i`

Pede confirmação sobre a finalização do processo;

`-l`

Lista o nome de todos os sinais conhecidos;

`-q`

Ignora a existência do processo;

`-v`

Retorna se o sinal foi enviado com sucesso ao processo;

`-w`

Finaliza a execução do `killall` somente após finalizar todos os processos.

- Exemplo:

```
killall -HUP inetd
```



Sinais do Sistema

- Os sinais de sistemas serão mostrados no arquivo a seguir:
 - Sinais do Sistema

Fechando um Programa a Força

- Em nosso exemplo vamos supor que executamos um programa em desenvolvimento com o nome DE "contagem", que conta o tempo em segundos a partir do momento que é executado, mas que o programador esqueceu de colocar uma opção de saída. Siga estas dicas para finalizá-lo:
 - Normalmente todos os programas UNIX podem ser interrompidos com o pressionamento das teclas `CTRL + C`. Tente isto! Mas provavelmente não vai funcionar se estiver usando um Editor de Texto (o programa entenderá como um comando de menu). Isto normalmente funciona para comandos que são executados e terminados sem a intervenção do usuário;
 - Caso a tentativa acima tenha falhado, então, vamos partir para a força! ;-)
 - Mude para um novo console (pressionando `ALT + F2`), e faça o login como usuário `root`;
 - Localize o `PID` (número de identificação do processo) usando o comando: `ps ax`. Aparecerão várias linhas cada uma com o número do processo na primeira coluna, e a linha de comando do programa na última coluna. Caso apareçam vários processos você pode usar `ps ax|grep contagem`, neste caso o `grep` fará uma filtragem da saída do comando `ps ax` mostrando somente as linhas que tem a palavra "contagem";
 - Feche o processo usando o comando `kill PID`, lembre-se de substituir `PID` pelo número encontrado pelo comando `ps ax` acima.



Fechando um Programa a Força

- O comando no *slide* anterior envia um sinal de término de execução para o processo (neste caso o programa “contagem”). O sinal de término mantém a chance do programa salvar seus dados ou apagar os arquivos temporários que criou e então ser finalizado, isto depende do programa;
 - Alterne para o console onde estava executando o programa “contagem” e verifique se ele ainda está em execução. Se ele estiver parado mas o aviso de comando não está disponível, pressione a tecla `ENTER`. Frequentemente acontece isto com o comando `kill`, você finaliza um programa mas o aviso de comando não é mostrado até que se pressione `ENTER`;
 - Caso o programa ainda não tenha sido finalizado, repita o comando `kill` usando a opção `-9`: `kill -9 PID`. Este comando envia um sinal de DESTRUIÇÃO do processo, fazendo ele terminar a força!
- Uma última dica: todos os programas estáveis tem sua opção de saída. Lembre-se de que quando se finaliza um processo, todos os dados do programa em execução podem ser perdidos, mesmo usando o `kill` sem a opção `-9`.



Eliminando Caracteres Estranhos

- Às vezes, quando um programa mal comportado é finalizado ou quando você visualiza um arquivo binário através do comando `cat`, é possível que o aviso de comando (`terminal` ou `console`) volte com caracteres estranhos;
- Para fazer tudo voltar ao normal, basta digitar `reset` e teclar `ENTER`. Não se preocupe! O comando `reset` não reiniciará seu computador (como o botão `reset` do seu computador faz), ele apenas fará tudo voltar ao normal;
- Note que enquanto você digitar `reset` aparecerão caracteres estranhos ao invés das letras. Não se preocupe! Basta digitar corretamente e teclar `ENTER` e o aviso de comando voltará ao normal.



Referências

- GUIA FOCA GNU/Linux. **Iniciante**. Disponível em:
<<http://www.guiafoca.org/cgs/guia/iniciante/ch-run.html>>. Acesso em: 07 ago. 2017.
- MANZANO, D. Z. **Processos de usuário em segundo plano e como manipulá-los**. Disponível em:
<<https://www.vivaolinux.com.br/dica/Processos-de-usuario-em-segundo-plano-e-como-manipula-los>>. Acesso em: 21 Set. 2017.