Guia Foca GNU/Linux
Capítulo 7 - Execução de programas

Este capítulo explica como executar programas no `GNU/Linux` e o uso das ferramentas de controle de execução dos programas.

7.1 Executando um comando/programa

Para executar um comando, é necessário que ele tenha permissões de execução (veja Tipos de Permissões de acesso, Seção 13.2 e ls, Seção 8.1) e que esteja no caminho de procura de arquivos (veja path, Seção 7.2).
No aviso de comando #(root) ou *$*(utilizador), digite o nome do comando e tecle 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 (será visto mais adiante neste capítulo).
Todo o programa recebe uma identificação de utilizador (UID) quando é executado o que determina quais serão suas permissões de acesso durante sua execução. O programa normalmente usa o UID do utilizador que o executou ou o utilizador configurado pelo bit de permissão de acesso SUID caso estiver definido. Existem também programas que são executados como root e modificam sua identificação de utilizador para algum que tenha menos privilégios no sistema (como o `Apache`, por exemplo). Para maiores detalhes veja Permissões de acesso a arquivos e diretórios, Capítulo 13.
Todo o programa executado no `GNU/Linux` roda sob o controle das permissões de acesso. Recomendo ver mais tarde o Permissões de acesso a arquivos e diretórios, Capítulo 13.
Exemplos de comandos: `ls`, `df`, `pwd`.

7.2 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 no diretório `/usr/local/bin` ele inicia a procura em`/usr/bin`, 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).

O caminho de diretórios vem configurado na instalação do Linux, mas pode ser alterado no arquivo `/etc/profile`. Caso deseje alterar o caminho para todos os utilizadores, este arquivo é o melhor lugar, pois ele é lido por todos os utilizadores no momento do login.
Caso um arquivo/comando não esteja localizado em nenhum dos diretórios do path, você deve executa-lo usando um `./` na frente do comando.
Se deseja alterar o `path` para um único utilizador, modifique o arquivo `.bash_profile` em seu diretório de utilizador (home).
OBSERVAÇÃO: Por motivos de segurança, não inclua o diretório atual `$PWD` no `path`.

7.3 Tipos de Execução de comandos/programas

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.

Para iniciar um programa em `primeiro plano`, basta digitar seu nome normalmente. Para iniciar um programa em `segundo plano`, acrescente o caracter`"&"` após o final do comando.

OBS: Mesmo que um utilizador execute um programa em segundo plano e saia do sistema, o programa continuará sendo executado até que seja concluído ou finalizado pelo utilizador que iniciou a execução (ou pelo utilizador root).

Exemplo: `find / -name boot.b &`
O comando será executado em segundo plano e deixará o sistema livre para outras tarefas. Após o comando `find` terminar, será mostrada uma mensagem.

7.4 Executando programas em seqüência

Os comandos podem ser executados em seqüência (um após o término do outro) se os separarmos com ";". Por exemplo: `echo primeiro;echo segundo;echo terceiro`

7.5 ps

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

`ps [opções]`

Onde:

opções
a

Mostra os processos criados por você e de outros utilizadores do sistema.

x

Mostra processos que não são controlados pelo terminal.

u

Mostra o nome de utilizador 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).

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.

--sort:*[coluna]*

Organiza a saída do comando `ps` de acordo com a coluna escolhida. Você pode usar as colunas `pid, utime, ppid, rss, size, user, priority`.
Pode ser especificada uma listagem em ordem inversa especificando `--sort:[-coluna]`. Para mais detalhes e outras opções, veja a página de manual.

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`. Para detalhes, veja | (pipe), Seção 14.5.
Ao contrário de outros comandos, o comando `ps` não precisa do hífen "-" para especificar os comandos. Isto porque ele não utiliza opções longas e não usa parâmetros.
Exemplos: `ps`, `ps ax|grep inetd`, `ps auxf`, `ps auxw`.

7.6 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:

-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.

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.

7.7 Controle de execução de processos

Abaixo algumas comandos e métodos úteis para o controle da execução de processos no `GNU/Linux`.

7.7.1 Interrompendo a execução de um processo

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, Seção 7.7.6 para interromper um processo sendo executado.

7.7.2 Parando momentaneamente a execução 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, Seção 7.7.4 ou bg, Seção 7.7.5.

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.

7.7.3 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 (veja Tipos de Execução de comandos/programas, Seção 7.3) 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, Seção 7.7.4 e bg, Seção 7.7.5. Um processo interrompido pode ser finalizado usando-se o comando `kill %[num]`, onde `[num]` é o número do processo obtido pelo comando `jobs`.

7.7.4 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 ativa-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`.

7.7.5 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* número do programa obtido com o pressionamento das teclas `CTRL+Z` ou através do comando `jobs`.

7.7.6 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, Seção 7.5. Também pode ser o número após o sinal de % obtido pelo comando`jobs` para matar uma tarefa interrompida. Veja jobs, Seção 7.7.3.

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 utilizador root para termina-lo ou destruí-lo. Você pode verificar se o processo foi finalizado através do comando`ps`. Os tipos de sinais aceitos pelo `GNU/Linux` são explicados em detalhes em Sinais do Sistema, Seção 7.7.9.
Exemplo: `kill 500`, `kill -9 500`, `kill %1`.

7.7.7 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.

Os tipos de sinais aceitos pelo `GNU/Linux` são explicados em detalhes na Sinais do Sistema, Seção 7.7.9.

Exemplo: `killall -HUP inetd`

7.7.8 killall5

Envia um sinal de finalização para todos os processos sendo executados.

`killall5 [`*`sinal`*`]`

7.7.9 Sinais do Sistema

Retirado da página de manual `signal`. O `GNU/Linux` suporta os sinais listados abaixo. Alguns números de sinais são dependentes de arquitetura.

Primeiro, os sinais descritos no *POSIX 1*:

```
Sinal Valor Ação Comentário

----------------------------------------------------------------------

HUP 1 A Travamento detectado no terminal de controle ou

finalização do processo controlado

INT 2 A Interrupção através do teclado

QUIT 3 C Sair através do teclado

ILL 4 C Instrução Ilegal

ABRT 6 C Sinal de abortar enviado pela função abort

FPE 8 C Exceção de ponto Flutuante

KILL 9 AEF Sinal de destruição do processo

SEGV 11 C Referência Inválida de memória
```

PIPE 13 A Pipe Quebrado: escreveu para o pipe sem leitores

ALRM 14 A Sinal do Temporizador da chamada do sistema alarm

TERM 15 A Sinal de Término

USR1 30,10,16 A Sinal definido pelo utilizador 1

USR2 31,12,17 A Sinal definido pelo utilizador 2

CHLD 20,17,18 B Processo filho parado ou terminado

CONT 19,18,25 Continuar a execução, se interrompido

STOP 17,19,23 DEF Interromper processo

TSTP 18,20,24 D Interromper digitação no terminal

TTIN 21,21,26 D Entrada do terminal para o processo em segundo plano

TTOU 22,22,27 D Saída do terminal para o processo em segundo plano

As letras da coluna Ação tem o seguinte significado:

- A - A ação padrão é terminar o processo.
- B - A ação padrão é ignorar o sinal.
- C - A ação padrão é terminar o processo e mostrar o core.
- D - A ação padrão é parar o processo.
- E - O sinal não pode ser pego.
- F - O sinal não pode ser ignorado.

Sinais não descritos no *POSIX 1* mas descritos na *SUSv2*:

Sinal Valor Ação Comentário

BUS 10,7,10 C Erro no Barramento (acesso incorreto da memória)

POLL A Evento executado em Pool (Sys V). Sinônimo de IO

PROF 27,27,29 A Tempo expirado do Profiling

SYS 12,-,12 C Argumento inválido para a rotina (SVID)

TRAP 5 C Captura do traço/ponto de interrupção

URG 16,23,21 B Condição Urgente no soquete (4.2 BSD)

VTALRM 26,26,28 A Alarme virtual do relógio (4.2 BSD)

```
XCPU 24,24,30 C Tempo limite da CPU excedido (4.2 BSD)

XFSZ 25,25,31 C Limite do tamanho de arquivo excedido (4.2 BSD)
```

(Para os casos SIGSYS, SIGXCPU, SIGXFSZ, e em algumas arquiteturas também o SIGGUS, a ação padrão do Linux para kernels 2.3.27 e superiores é A (terminar), enquanto SYSv2 descreve C (terminar e mostrar dump core).)
Seguem vários outros sinais:

```
Sinal Valor Ação Comentário

--------------------------------------------------------------------

IOT 6 C Traço IOT. Um sinônimo para ABRT

EMT 7,-,7

STKFLT -,16,- A Falha na pilha do processador

IO 23,29,22 A I/O agora possível (4.2 BSD)

CLD -,-,18 Um sinônimo para CHLD

PWR 29,30,19 A Falha de força (System V)

INFO 29,-,- Um sinônimo para SIGPWR

LOST -,-,- A Perda do bloqueio do arquivo

WINCH 28,28,20 B Sinal de redimensionamento da Janela (4.3 BSD, Sun)

UNUSED -,31,- A Sinal não usado (será SYS)
```

O "-" significa que o sinal não está presente. Onde três valores são listados, o primeiro é normalmente válido para o Alpha e Sparc, o do meio para i386, PowerPc e sh, o último para o Mips. O sinal 29 é SIGINFO/SIGPWR em um Alpha mas SIGLOST em um Sparc.

7.8 nohup

Executa um comando ignorando os sinais de interrupção. O comando poderá ser executado até mesmo em segundo plano caso seja feito o logout do sistema.

`nohup [`*`comando que será executado`*`]`
As mensagens de saída do `nohup` são direcionadas para o arquivo `$HOME/nohup.out`.
Exemplo: `nohup find / -uid 0 >/tmp/rootfiles.txt &`.

7.9 nice

Configura a prioridade da execução de um comando/programa.

`nice [opções] [comando/programa]`

Onde:

comando/programa

Comando/programa que terá sua prioridade ajustada.

opções
-n [numero]
Configura a prioridade que o programa será executado. Se um programa for executado com maior prioridade, ele usará mais recursos do sistema para seu processamento, caso tenha uma prioridade baixa, ele permitirá que outros programas tenham preferência. A prioridade de execução de um*programa/comando* pode ser ajustada de -19 (a mais alta) até 19 (a mais baixa).

Exemplo: `nice -n -19 find / -name apropos`.

7.10 fuser

Permite identificar e fechar os processos que estão utilizando arquivos e soquetes no sistema.

`fuser [opções] [nome]`

Onde:

nome

Especifica um nome de processo, diretório, arquivo, etc.

opções
-k
Finaliza os processos acessando o arquivo especificado. O sinal desejado deve ser especificado com a opção `-signal [num]`, ou o sinal -9 será enviado como padrão. Não é possível matar o próprio processo `fuser`.

-i
Pergunta antes de destruir um processo. Será ignorada caso a opção `-k` não seja especificada.

-l

Lista todos os nomes de sinais conhecidos.

-m [nome]

Especifica um arquivo em um sistema de arquivos montado ou dispositivo de bloco que está montado. Todos os processos acessando aquele sistema de arquivos serão listados. Diretórios são mostrados seguidos de uma `/`

-signal [número]

Usa o sinal especificado ao invés de -9 (SIGKILL) quando finalizar processos.

-u

Acrescenta o nome do dono de cada processo ao PID.

-v

Os processos são mostrados em um estilo idêntico ao `ps`.

7.11 tload

Representa de forma gráfica a carga do sistema.

`tload [`*`opções`*`]`

Onde:

opções
-s [número]

Mostra uma escala vertical com espaçamento especificado por [número]. É recomendável o uso de números entre 1 e 10 para melhor visualização da escala.

-d [número]

Especifica o intervalo entre atualizações, em segundos.

7.12 vmstat

Mostra estatísticas sobre o uso da memória virtual do sistema.

`vmstat [`*`intervalo`*`] [`*`contagem`*`]`

Onde:

intervalo

Número especificado em segundos entre atualizações.

contagem

Número de vezes que será mostrado.

Se não for especificado nenhum parâmetro, o `vmstat` mostra o status da memória virtual e volta imediatamente para a linha de comando. A descrição dos campos do `vmstat` são as seguintes:

Processos

r

Número de processos aguardando execução.

b

Número de processos em espera não interrompíveis.

w

Número de processos extraídos do arquivo de troca ou caso contrário em execução.

Memória

swpd

A quantidade de memória virtual usada em Kb.

free

Quantidade de memória livre em Kb.

buff

Quantidade de memória usada como buffer em Kb.

Memória Virtual

si

Quantidade de memória gravada para o disco Kb/s.

so

Quantidade de memória retirada do disco em Kb/s.

Entrada/Saída

bi

Blocos enviados para um dispositivo de bloco (medido em blocos por segundo).

bo

Blocos recebidos de um dispositivo de bloco (em blocos por segundo).

Sistema

in

Número de interrupções por segundo, incluindo o clock.

cs

Número de mudanças de contexto por segundo.

Porcentagem do total de tempo da CPU
us

Tempo do utilizador

sy

Tempo do sistema

id

Tempo ocioso

7.13 pidof

Retorna o PID do processo especificado

`pidof [opções] [nome]`

Onde:

nome

Nome do processo que seja obter o número PID

opções
-s

Retorna somente o primeiro PID encontrado.

-x

Retorna o PID do do shell que está executando o script

-o [PID]
Ignora o processo com aquele PID. O PID especial %PPID pode ser usado para nomear o processo pai do programa `pidof`, em outras palavras

OBS: O programa `pidof` é um link simbólico ao programa `killall5`. Cuidado ao executar o `killall5` as funções e opções são completamente diferentes dependendo da forma como é chamado na linha de comando! (veja killall5, Seção 7.7.8 para detalhes.)
Exemplo: `pidof -s init`

7.14 pstree

Mostra a estrutura de processos em execução no sistema em forma de árvore.

`pstree [opções] [pid]`

Onde:

pid

Número do processo que terá sua árvore listada. Se omitido, lista todos os processos.

opções
-a

Mostra opções passadas na linha de comando.

-c

Mostra toda a estrutura (inclusive sub-processos do processo pai).

-G

Usa caracteres gráficos no desenho da árvore de processos.

-h

Destaca o processo atual e seus antecessores.

-H [pid]

Destaca o processo especificado.

-l

Não faz quebra de linha

-n

Classifica pelo número PID ao invés do nome.

-p

Mostra o número PID entre parênteses após o nome do processo.

-u

Mostra também o dono do processo.

-U

Usa o conjunto de caracteres Unicode para o desenho da árvore.

7.15 Fechando um programa quando não se sabe como sair

Muitas vezes quando se esta iniciando no `GNU/Linux` você pode executar um programa e talvez não saber como fecha-lo. Este capítulo do guia pretende ajuda-lo a resolver este tipo de problema.

Isto pode também ocorrer com programadores que estão construindo seus programas e por algum motivo não implementam uma opção de saída, ou ela não funciona!

Em nosso exemplo vou supor que executamos um programa em desenvolvimento com o nome `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 finaliza-lo:

1. Normalmente todos os programas `UNIX` (o `GNU/Linux` também é um Sistema Operacional baseado no `UNIX`) podem ser interrompidos com o pressionamento das teclas `<CTRL>` e `<C>`. Tente isto primeiro para finalizar um programa. Isto provavelmente não vai funcionar se estiver usando um Editor de Texto (ele vai entender como um comando de menu). Isto normalmente funciona para comandos que são executados e terminados sem a intervenção do utilizador.

   Caso isto não der certo, vamos partir para a força! ;-)

2. Mude para um novo console (pressionando `<ALT>` e `<F2>`), e faça o *login* como utilizador **root**.
3. 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 aparecerem 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". Para maiores detalhes, veja o comandogrep, Seção 10.8.
4. Feche o processo usando o comando `kill` ***`PID`***, lembre-se de substituir PID pelo número encontrado pelo comando `ps ax` acima.
   O comando acima 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.
5. 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>`. Freqüentemente acontece isto com o comando `kill`, você finaliza um programa mas o aviso de comando não é mostrado até que se pressione `<ENTER>`.

6. Caso o programa ainda não foi finalizado, repita o comando `kill` usando a opção -9: `kill -9 PID`. Este comando envia um sinal de DESTRUIÇÃO do processo, fazendo ele terminar "na marra"!
   Uma última dica: todos os programas estáveis (todos que acompanham as boas distribuições `GNU/Linux`) tem sua opção de saída. Lembre-se que quando finaliza um processo todos os dados do programa em execução podem ser perdidos (principalmente se estiver em um editor de textos), mesmo usando o`kill` sem o parâmetro `-9`.
   Procure a opção de saída de um programa consultando o help on line, as páginas de manual, a documentação que acompanha o programa, info pages. Para detalhes de como encontrar a ajuda dos programas, veja o Como obter ajuda no sistema, Capítulo 31

7.16 Eliminando caracteres estranhos

As 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 (prompt) 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 bater `ENTER` e o aviso de comando voltará ao normal.