



# **Criar, monitorar e matar processos**

## Sumário

### Capítulo 1

Criar, monitorar e matar processos.....	3
1.1. Mãos a obra.....	4

### Capítulo 2

Gerenciando .....	13
2.1. Objetivos.....	13
2.1. Troubleshooting.....	13

## Índice de tabelas

## Índice de Figuras

# Capítulo 1

## Criar, monitorar e matar processos

- Executar trabalhos no primeiro plano e fundo;
- Monitor de processos ativos;
- O envio de sinais aos processos;
- bg;
- fg;
- jobs;
- kill;
- nohup;
- ps;
- top;
- free;
- uptime;

•killall;

## 1.1. Mãos a obra

*Quando você executa um programa no Linux sendo modo gráfico ou texto (no terminal), o sistema cria números que identificam o programa que está sendo executado. Cada programa que esta em execução recebe do kernel um um PID (Process ID).*



*Mas para que serve isso?*

*O kernel precisa identificar o processo para executá-lo no processador.*

*A administração de servidores são 99% em modo texto, isso significa que posso usar as minhas 6 ttys. Mas se por questões de segurança eu desabilitei 5 ttys e fiquei apenas com a tty1.*



*Para desativar e ativar o uso das ttys edite o arquivo /etc/inittab.*

*Um exemplo com uso apenas de uma tty1, é fazer uma compactação demorada, deixando seu único terminal ocupado. Para resolver isso é possível rodar programas colocando processos em segundo plano*



*Mas o que é colocar o processo em segundo plano?*

*Colocar o processo em segundo plano significa que o kernel irá executar esse processo sem a interação do usuário. Existe varias maneiras de se fazer isso. Vamos a prática:*

Rodar um comando colocando o sinal "&" após o final do comando):



```
# top &
```

Ao executar o top em segundo plano você recebe logo abaixo um numero de

ordem dos programas em segundo plano, e o PID do programa em execução:

```
server:~# top &  
[1] 3265
```

Uma outra forma é executar um comando e usar a combinação no teclado CTRL + Z para colocar o programa em segundo plano:



```
# vim /etc/fstab  
CTRL + Z
```

Para listar os programas parados ou rodando em segundo plano é usado o comando jobs. Veja o exemplo:

```
server:~# jobs -l  
[1]+ 2430 Parado          vim /etc/fstab
```

*Se você fizer logout fechando a sessão do usuário logado, por exemplo o root, os programas que estão rodando em segundo plano, serão fechados.*



Como resolver isso?

*Você pode usar alguns comando, como o nohup, disown e screen para o que os programas continuem rodando mesmo após o logout. Vamos a prática.*

*No exemplo vamos usar o comando wget usado para fazer downloads via terminal. Com a opção -c o wget continua o download caso seja interrompido:*

```
server:~# nohup wget -c http://ubuntu.c3sl.ufpr.br/releases/lucid/ubuntu-10.04-desktop-i386.iso &  
[1] 2032  
server:~# nohup: a ignorar a entrada e a acrescentar a saída a 'nohup.out'
```

*Para usar o nohup a sintaxe é:*

*nohup comando &*

*É criado um arquivo nohup.out no mesmo diretório onde foi usado comando, contendo as saídas stdout e stderr do comando.*

```
server:~# tail -f nohup.out
117100K ..... 16% 72,7K 2h53m
117150K ..... 16% 150K 2h53m
117200K ..... 16% 39,3K 2h53m
117250K ..... 16% 27,8K 2h53m
117300K ..... 16% 77,1K 2h53m
117350K ..... 16% 28,2K 2h53m
117400K ..... 16% 30,5K 2h53m
117450K ..... 16% 83,1K 2h53m
117500K ..... 16% 25,9K 2h53m
117550K ..... 16% 28,6K 2h53m
117600K ..... 16% 87,0K 2h53m
117650K .....
```

Vamos fazer logout, que envia a todos os processos o sinal (SIGHUP) que os termina. Para evitar esta situação usamos o comando nohup, que é wrapper usado para iniciar um programa imune ao SIGHUP. Veja o exemplo:

```
server:~# logout
Debian GNU/Linux 5.0 server tty1
server login: root
Password:
Last login: Mon Jun 14 20:13:24 BRT 2010 on tty1
server:~# ps aux | grep wget
root      2032  0.5  0.4  5380  1708 ?        S    19:45   0:09 wget -c http://ubuntu.c3sl.ufpr.br/releases/lucid/ubuntu-10.04-
desktop-i386.iso
root      2163  0.0  0.1   3140    768 tty1    S+   20:15   0:00 grep wget
```

O logout foi feito, e ao fazer login no sistema foi usado o comando ps aux, onde é possível ver que o wget ainda esta rodando.

### Monitoração de processos

Em nosso exemplo o comando wget continua rodando e fazendo download de um arquivo iso. Para listar os processos que estão em andamento no sistema, é usado o comando ps.

Opções mais utilizadas do comando ps:

*a* - Exibe todos os processos criados;

*l* - Exibe informações extensas como por exemplo a prioridade dos processos;

*x* - Exibe os processos que não são controlados pelo terminal;

*u* - Exibe o nome do usuário e a hora que o processo foi iniciado.

*Veja o exemplo:*

```
server:~# ps aux | grep wget
root    2521  0.6  0.4  5380  1712 ?        S    21:58   0:00 wget -c http://ubuntu.c3sl.ufpr.br/releases/lucid/ubuntu-10.04-
desktop-i386.iso
```

*O wget ganhou o PID 2521, que pode ser usado para gerenciar o seu processo. Para poder o wget é usado o comando kill + PID do processo.*



```
# kill 2521
```

*Vejamos se o wget ainda esta sendo executado com o comando ps aux*

```
server:~# kill 2521
server:~# ps aux | grep wget
root    2779  0.0  0.1  3140   768 tty1     S+   23:05   0:00 grep wget
```

*A monitoração de processos pode também ser feita usando o comando top, que monitora o sistema e mostra a atividade do processador em tempo real.*

*Algumas opções do top*

*No terminal fora do top:*

*- h - Lista os comandos que podem ser usados;*

*i - Ignora processos ociosos;*

*u - Mostra os processos de um usuário;*

*Exemplo:*



```
# top -u aluno
```

*-p - Monitora apenas um processo usando seu PID;*

*Exemplo:*



```
# top -p 2825
```

*-d - Atualiza a lista de processos usando intervalos de segundos.*

*Exemplo:*



```
# top -d 5
```

Dentro do top:

*N - Classifica os processos por número de PID;*

*P - Classifica os processos por uso de CPU;*

*A - Classifica os processos por período;*

*M - Classifica os processos por uso de memória;*

*T - Classifica os processos por tempo;*

*k - Mata um processo;*

*s - Especifica o tempo em segundos para a atualização da tela;*

*r - Aplica um renice no processo;*

*q - Sai do top.*



*Como trazer para o primeiro plano um programa que esta rodando em segundo plano?*

*Vamos lá, você já viu maneiras de enviar para o segundo plano usando o sinal & e as combinações CTRL + Z, e também listar com o comando jobs. Para trazer para primeiro plano é usado o comando fg + o numero do processo que esta em segundo plano. Vamos a prática:*

```
server:~# jobs -l  
[1]+ 2799 Parado vim /etc/fstab
```

*Veja no exemplo que o vim esta parado em segundo plano (foi enviando através do CTRL + Z), e ao lado esquerdo ele recebeu o numero [1], então para trazer para primeiro plano:*





# fg 1

Vamos agora a um outro exemplo, enviar um programa para segundo plano e permitir que este continue rodando. O comando bg é responsável por esta tarefa. Veja a sequência:

1 - Rode o programa e depois use CTRL + Z para enviar para segundo plano. Feito isso liste com o comando jobs -l.

```
server:~# jobs -l
[1]+ 2885 Parado          man ls
```

2 - Perceba que o programa esta parado sem saída para o terminal. Para continuar rodando em segundo plano use o comando bg + seu numero ao lado esquerdo [1]

```
server:~# bg 1
[1]+ man ls &
```

Veja o resultado com o comando jobs -l:

```
server:~# jobs -l
[1]+ 2885 Parado (saída tty)    man ls
```

Envio de sinais

Como foi visto em nosso exemplo o comando kill encerra um programa que está em execução, é possível enviar através do comando kill outros tipos de sinais, como pausar, finalizar (forçar), continuar (tirar da pausa), entre outros. Vamos a prática:

```
server:~# kill -l
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL
5) SIGTRAP     6) SIGABRT     7) SIGBUS      8) SIGFPE
9) SIGKILL     10) SIGUSR1    11) SIGSEGV    12) SIGUSR2
13) SIGPIPE    14) SIGALRM    15) SIGTERM    16) SIGSTKFLT
17) SIGCHLD    18) SIGCONT    19) SIGSTOP    20) SIGTSTP
21) SIGTTIN    22) SIGTTOU    23) SIGURG     24) SIGXCPU
25) SIGXFSZ    26) SIGVTALRM  27) SIGPROF    28) SIGWINCH
29) SIGIO      30) SIGPWR     31) SIGSYS     34) SIGRTMIN
35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3 38) SIGRTMIN+4
39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Veja no exemplo acima a lista de sinais que podem ser enviados aos processos. A lista foi exibida através do comando kill -l.

Você pode usar de duas maneiras o envio de sinais:

kill - Envia sinais através do PID.

Killall - Envia sinais através do nome.

sintaxe do comando kill:

kill + N° do sinal + PID



```
# kill -9 2885
```

sintaxe do comando killall:

kill + N° do sinal + nome do programa



```
# killall -9 vim
```



Como posso saber a função de cada numero e nomes de sinais, exibidos com o comando kill -l?

Simple, usando comando man signal. Veja o exemplo:



```
# man signal
```

Sinal	Valor	Ação	Comentário
SIGHUP	1	A	Travamento detectado no terminal controlador
SIGINT	2	A	Interrupção do teclado
SIGQUIT	3	C	Sinal de @Quit@ do teclado
SIGILL	4	C	Instrução ilegal
SIGABRT	6	C	Sinal abort derivado de abort(3)
SIGFPE	8	C	Exceção de ponto flutuante
SIGKILL	9	AEF	Sinal de kill
SIGSEGV	11	C	Referência inválida a memória
SIGPIPE	13	A	Broken pipe: escrita para um pipe sem um leitor.
SIGALRM	14	A	Sinal do timer de alarm(2)
SIGTERM	15	A	Sinal de terminação
SIGUSR1	30,10,16	A	Sinal definido pelo usuário 1
SIGUSR2	31,12,17	A	Sinal definido pelo usuário 2
SIGCHLD	20,17,18	B	Processo descendente parado ou terminado.
SIGCONT	19,18,25		Continuar se parado
SIGSTOP	17,19,23	DEF	Parar processo
SIGTSTP	18,20,24	D	Stop digitado no tty
SIGTTIN	21,21,26	D	Entrada via tty para processo no background
SIGTTOU	22,22,27	D	Saída via tty para processo no background

Exemplos de uso do kill:




```
# killall -15 firefox
```

Envia o sinal de terminar para o programa firefox.

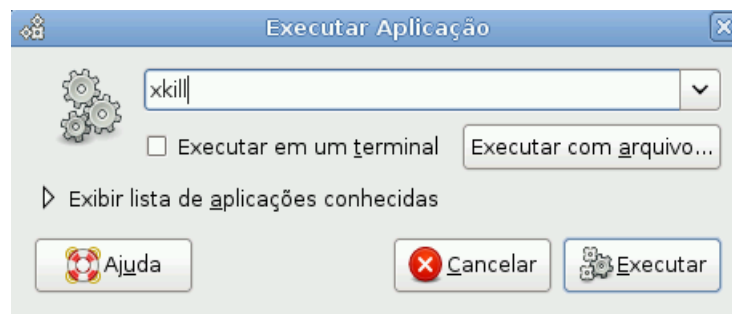


```
# kill -9 3839
```

Envia o sinal de finalizar (forçar) para o PID de numero 3839.

Para você encerrar um programa no modo gráfico, é usado o comando xkill. Por exemplo se você esta no gnome e um programa não fecha ao clicar no 

Use a combinação de teclas CTRL + F2 e na caixa digite xkill, use o Enter e aponte para a janela que você quer encerrar.



Mais comandos uteis

*free* - Exibe a quantidade de memória livre e utilizada no sistema. Vamos a prática:



```
# free -m
```

```
Mem:      total    used    free   shared  buffers   cached
-/+ buffers/cache:    28    348
Swap:      972       0    972
```

No nosso exemplo foi usado a opção -m com o comando free, para exibir os valores em MB. Para fazer o calculo de memória RAM + SWAP adicione a opção -t.



```
# free -m -t
```

```
Mem:      total    used    free   shared  buffers   cached
-/+ buffers/cache:    28    348
Swap:      972       0    972
Total:    1349    315   1034
```

*uptime* - Exibe quanto tempo o sistema esta em funcionamento. Vamos a prática:



```
# uptime
```

```
19:11:47 up 1:00, 2 users, load average: 0.51, 0.34, 0.29
```

O comando exibe algumas informações interessantes como por exemplo a hora atual, quanto tempo o servidor esta ligado, quantos usuário estão logados e as ultimas três colunas informações do load average.



*O que é load average?*

Um média de carga do sistema nos últimos 1, 5 e 15 minutos. É exibido um número médio de processos que estejam em um executável ou ininterrupto.

# Capítulo 2

## Gerenciando

### 2.1. Objetivos

- Troubleshooting: de Processos.

### 2.1. Troubleshooting



*Como eu faço para recuperar a mesma sessão quando ocorre uma perda de conexão?*

*Imagine a seguinte situação: Você está digitando um texto no vim conectado a um computador remoto, através de uma sessão SSH e perde sua conexão. Ao reconectar você perde sua sessão aberta com o vim, e assim terá que abrir o vim novamente.*

*Para resolver isso instale na maquina remota o programa screen:*

*No Debian:*



```
# aptitude install screen
```

*No RedHat*



```
# yum install screen
```

Vamos a prática:

1 - Acesse a maquina remota via ssh e instale o programa screen:

```
server:~# ssh 192.168.200.254
The authenticity of host '192.168.200.254 (192.168.200.254)' can't be established.
RSA key fingerprint is 41:34:12:6d:01:2f:54:91:ad:a0:0e:59:22:67:bc:1c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.200.254' (RSA) to the list of known hosts.
root@192.168.200.254's password:
Linux debian 2.6.26-2-686 #1 SMP Wed Aug 19 06:06:52 UTC 2009 i686
```

```
root@debian:~# aptitude install screen
```

2 - Digite screen para abrir uma sessão e depois edite um arquivo com o vim:



```
# screen (Enter)
# vim /etc/fstab
```

```
Screen version 4.00.03jw4 (FAU) 2-May-06

Copyright (c) 1993-2002 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program (see the file COPYING); if not,
write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Send bugreports, fixes, enhancements, t-shirts, money, beer & pizza to screen@uni-erlangen.de
```

3 - Para nosso exemplo ficar bem realista, na maquina remota que esta sendo acessada, use o comando `ps aux | grep ssh` para exibir o PID do processo do SSH.

4 - Use o comando `kill -9 PID` do processo do SSH para finalizar a conexão.

Veja a mensagem que é exibida ao perder a conexão SSH:

```
root@debian:~# Connection to 192.168.200.254 closed by remote host.
Connection to 192.168.200.254 closed.
```

5 - Como você abriu uma sessão com o screen é só reconectar ao servidor remoto e usar o comando `screen -ls` para listar e `screen -r` para recuperar. Vamos a prática:



```
# ssh 192.168.200.254  
# screen -ls
```

```
root@debian:~# screen -ls  
There are screens on:  
  4086.pts-1.debian      (15-06-2010 20:26:46)  (Detached)  
  3902.pts-1.debian      (15-06-2010 20:10:34)  (Detached)  
2 Sockets in /var/run/screen/S-root.
```

6 – Para recuperar a sessão use o comando `screen -r PID.terminal.hostname`



```
# screen -r 4086.pts-1.debian
```



*Pronto!!! Sua sessão foi recuperada.*



*Use o comando `exit` para sair da sessão criada pelo screen.*