Gerenciamento de processos

Gerenciamento de processos

Introdução

Processo é definido como porções de programas ou programa inteiro em funcionamento na memória do computador.

É o processo que utiliza os recursos do computador - processador, memória para a realização das tarefas para as quais a máquina é destinada.

Falando de uma maneira mais clara, tudo que estiver em execução na máquina é um processo, ou seja, o VI aberto é um processo, o shell quando você loga é um processo. Tudo é encarado como um processo.

Quando um processo termina a execução do programa, o sistema destrói o processo e os recursos alocados são devolvidos para que sejam aproveitados por um outro processo.

OBS: Para manipularmos processos, precisamos estar logados como superusuário (**root**).

Gerenciamento de processos

Composição de um processo

Um processo tem características como:

- Proprietário do processo;
- Estado do processo (em espera, em execução, etc);
- Prioridade de execução;
- Recursos de memória.

Para gerenciar processos de maneira eficiente precisamos contar com as informações acima e com outras de igual importância. Um dos meios usados para isso é atribuir a cada processo um **PID**.

O PID (**Process Identifier**) é um número de identificação que o sistema dá a cada processo. Para cada novo processo, um novo número deve ser atribuído, ou seja, dois processos não podem ter o mesmo número de PID.

Gerenciamento de processos

Um processo que cria outro processo é chamado de **processo pai** e o processo criado é chamado de **processo filho**. No Linux, um processo pai cria um processo filho fazendo cópia de si mesmo.

Com exceção do **PID**, estes processos (pai e filho) são idênticos. Para executar um novo programa, o processo criado copia o processo pai e então inicia sua execução. **Um processo pai pode ter vários processos filho e um processo filho tem um único processo pai.**

Um processo pai pode suspender sua própria execução até que um ou mais de seus filhos termine sua execução. Por exemplo, o **shell** normalmente executa um programa criando um processo filho e espera até que o processo filho termine sua execução para liberar o **prompt** para o usuário para a execução do próximo comando.

Gerenciamento de processos

Atributos de um processo

- **PID** (identificação do processo)
- **PPID** (identificação do processo pai)
- **UID** (identificação do usuário que criou o processo)
- **GID** (identificação do grupo ao qual pertence o processo)

Para gerenciar processos de maneira eficiente precisamos contar com as informações acima e com outras de igual importância. Um dos meios usados para isso é atribuir a cada processo um **PID**.

O PID (**Process Identifier**) é um número de identificação que o sistema dá a cada processo. Para cada novo processo, um novo número deve ser atribuído, ou seja, e dois processos não podem ter o mesmo número de PID.

Gerenciamento de processos

Estado de processos

Um processo pode passar por vários estados. Quando um processo é criado, isso não significa que ele será imediatamente executado. Alguns processos podem ser temporariamente parados para que o processador possa executar um processo com maior prioridade.

O Linux trabalha com quatro tipos de estados:

Executável (running): o processo pode ser executado imediatamente;

Dormente (waiting): o processo precisa aguardar alguma coisa para ser executado. Só depois dessa "coisa" acontecer é que ele passa para o estado executável;

Zumbi (zombie): o processo é considerado "morto", mas, por alguma razão, ainda existe;

Parado (stopped): o processo está "congelado", ou seja, não pode ser executado.

Gerenciamento de processos

Classificação de processos quanto a execução

Foreground (primeiro plano): são inicializados no terminal de comandos, podem interagir com os usuários e exibem sua execução na tela. A desvantagem desse tipo de processo é que ele prende o prompt e isso impede a execução de novos processos nesse terminal.

Background (segundo plano): são inicializados no terminal de comandos, NÃO podem interagir com os usuários e NÃO exibem sua execução na tela. A vantagem desse tipo de processo é que ele NÃO prende o prompt e isso NÃO impede a execução de um novo processo nesse terminal.

Interativo: são inicializados a partir de um terminal do usuário e são controlados por ele, usamos os comandos **CTRL + Z**, **CTRL + X**, **fg**, **bg** e **jobs** para trabalhar com esse tipo de processo.

Lote (batch): são processo controlados pelos comandos at, batch e cron.

Gerenciamento de processos

Classificação de processos quanto a execução

Daemons: é um programa que funciona em background (segundo plano) esperando que um outro processo solicite seu serviço. Os processos do Daemon fornecem freqüentemente serviços de rede, tais como o email, web etc. Eles não precisam de nenhuma entrada e normalmente não produzem nenhuma saída.

Muitos daemons do Linux têm os nomes que terminam em um "**d**", como o **httpd**, **ftpd** etc. O oposto de um daemon é um processo que funciona em primeiro plano.

Daemons mais populares:

atd - roda trabalhos agendados pelo comando **at;**

httpd - servidor web **Apache**;

postfix - agente para transporte de correios substituto do **sendmail**;

smbd - o daemon **SAMBA** (ou smb).

Gerenciamento de processos

Classificação de processos quanto a execução

Daemons: é um programa que funciona em background (segundo plano) esperando que um outro processo solicite seu serviço. Os processos do Daemon fornecem freqüentemente serviços de rede, tais como o email, web etc. Eles não precisam de nenhuma entrada e normalmente não produzem nenhuma saída.

Muitos daemons do Linux têm os nomes que terminam em um "**d**", como o **httpd**, **ftpd** etc. O oposto de um daemon é um processo que funciona em primeiro plano.

Daemons mais populares:

atd - roda trabalhos agendados pelo comando **at;**

httpd - servidor web **Apache**;

postfix - agente para transporte de correios substituto do **sendmail**;

smbd - o daemon **SAMBA** (ou smb).

Gerenciamento de processos

ps

O comando **ps** exibe todos os processos sendo executados no servidor.

Exemplos:

 # ps -e => exibe todos os processos em execução com seus respectivos PID.

 # ps -t tty[n] => exibe todos os processos em execução no referido terminal.

 # ps -u login_name **=>** exibe todos os processos em execução pelo referido usuário.

 # ps -f => exibe informação completa dos processos rodando.

Gerenciamento de processos

ps aux

É opção mais usada para verificar processos.

a - mostra todos os processos existentes;

u - exibe o nome do usuário que iniciou determinado processo e a hora em que isso ocorreu;

x - exibe os processos que não estão associados a terminais;

Exemplo**:**

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.2	1492	476	?	S	14:18	0:00	init [2]
root	2	0.0	0.0	0	0	?	S	14:18	0:00	[keventd]
h4ck3r	141	0.1	13.0	334	292	?	Ss	15:04	0:36	gimp
root	151	0.0	0.7	302	1696	tty2	Ss+	15:34	0:00	-bash
root	164	0.0	0.2	1664	672	?	Ss	20:34	0:00	dhclient

Gerenciamento de processos

ps

Campo	Conteúdo
USER	Nome do usuário do proprietário do processo.
PID	ID (identificador) do processo.
%CPU	Porcentagem de recursos de CPU que este processo usa.
%MEM	Porcentagem de memória real que este processo usa.
VSZ	Tamanho virtual do processo.
RSS	Resident Set Size (número de páginas na memória)
TTY	ID de terminal (? - Não depende de terminal)
STAT	Estado do processo. (R, S, Z, D, T)
START	Horário em que o processo foi iniciado
TIME	Tempo de CPU que o processo consumiu
COMMAND	Nome do comando e argumentos

Gerenciamento de processos

STAT - indica o estado atual do processo, sendo representado por uma letra:

R - executável;

D - em espera no disco;

S - Suspenso;

T - interrompido;

Z - Zumbi.

W - processo paginado em disco;

< - processo com prioridade maior que o convencional;

N - processo com prioridade menor que o convencional;

L - processo com alguns recursos bloqueados no kernel.

Gerenciamento de processos

pstree

O comando **pstree** é usado para visualizar a árvore de processos.

```
# pstree
init-+-alarmd
|-apache---5*[apache]
|-atd
|-cron
|-6*[getty]
|-i2oevtd
|-inetd
```

Gerenciamento de processos

top

O comando **top** é um monitor do sistema que mostra a atividade do processador em tempo real. Exibindo uma lista das tarefas no sistema que usam com mais intensidade a **CPU** e fornecendo uma interface interativa para manipulação dos processos.

Execute o **top** em um terminal separado, assim você poderá monitorar o seu sistema enquanto trabalha normalmente.

 # top

Opções do Top

 # top d [tempo] – Atualiza a tela após o tempo determinado

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

Gerenciamento de processos

top

É possível manipular recursos do comando top através das teclas do teclado. Por exemplo, para atualizar imediatamente o resultado exibido, basta pressionar a **tecla de espaço**. Se pressionar a tecla **q**, o top é finalizado. Pressione a tecla **h** enquanto estiver utilizando o top para ver a lista completa de opções e teclas de atalho.

jobs

O comando **jobs** lista os processos em execução pelo shell que estão em **background**. Veja a sintaxe:

```
# jobs -l
[2]- 1245 Running asmixer &
[3]+ 1333 Running openoffice &
```

Gerenciamento de processos

jobs

O parâmetro **-l** faz com que o comando **jobs** liste também o **PID** de cada processo. Em sua saída o comando **jobs** retorna, além do número do serviço, o estado do processo, a linha digitada no comando e, opcionalmente, o PID.

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.

Exemplos:

```
# jobs -l
 [2]-1245 Running asmixer &
# fg 2
 asmixer
```

Gerenciamento de processos

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, devemos primeiro interromper a execução do comando com **CTRL+Z**; será mostrado o número da tarefa interrompida.

Use este número com o bg para iniciar a execução em segundo plano.

Exemplos:

 # fg 2

asmixer

Interropendo: CTRL+Z => [2]+ Stopped asmixer)

 # bg 2

Gerenciamento de processos

fuser: mostra qual processo faz uso de um determinado arquivo ou diretório.

Sua sintaxe é:

 # fuser -opção caminho (do arquivo ou diretório)

Opções:

-k - finaliza o processo que utiliza o arquivo/diretório em questão;

-i - deve ser usada em conjunto com a opção **k** e serve para perguntar se a finalização do processo deve ser feita;

-u - mostra o proprietário do processo;

-v - o resultado é mostrado em um padrão de exibição semelhante ao comando **ps**.

Gerenciamento de processos

fuser

Por exemplo, para ver quem está acessando o cd-rom, digite:

 # fuser -u /mnt/cdrom

Como a resposta, você tem os números PID de cada processo utilizado e o nome do usuário que está usando.

Este comando é útil quando você quer desmontar uma unidade, mas o sistema o impede porque alguém está usando naquele momento.

Gerenciamento de processos

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.

Com o **nohup** você NÃO fica preso ao terminal, assim poderá colocar qualquer processo em execução e fechar o terminal ou shell.

Exemplos:

 # nohup [_comando a ser executado_]

 # nohup cp -a /etc /tmp

OBS: As mensagens de saída do “nohup” são direcionadas para o arquivo **nohup.out** do diretório corrente.

Gerenciamento de processos

nice

Configura a prioridade da execução de um comando ou programa.

Exemplo:

 # nice -n [número]

A prioridade de execução de um Programa/comando pode ser ajustada de **-20** (a mais alta) até **19** (a mais baixa). A regra é inversamente proporcional, quanto menor for o número maior sua prioridade.

Exemplo:

 # nice -n -19 find / -name apropos

Gerenciamento de processos

renice

Configura a prioridade da execução de um comando ou programa que já está em execução. O **renice** segue o mesmo intervalo de prioridades que o **nice**, mas precisamos passar o **PID** do processo que será priorizado.

Exemplo:

#renice -20 -p PID

Para obter o PID do processo, use os comandos já visto anteriormente.

Gerenciamento de processos

kill

Esse comando permite cancelar processo em execução, desde que você tenha permissão para isso, ou seja, que você seja superusuário ou dono do processo.

Exemplo:

 # kill <SIGNAL> PID

 # kill -9 385

Onde :

-9 – É a opção do kill que força o processo ser fechado independente de qualquer coisa.

385 – É o número do processo obtido pelo comando **ps -aux**, que você deseja cancelar

Gerenciamento de processos

killall

Ele é idêntico ao **kill**, mas ele finaliza processo através do nome, ou seja, última coluna do comando ps.

 # killall <signal> [_processo_]

Exemplo:
 # killall -9 inetd

Gerenciamento de processos

Sinais de processos

Os sinais são meios usados para que os processos possam se comunicar e para que o sistema possa interferir em seu funcionamento.

STOP - esse sinal tem a função de interromper a execução de um processo e só reativá-lo após o recebimento do sinal **CONT**;

CONT - esse sinal tem a função de instruir a execução de um processo após este ter sido interrompido;

SEGV - esse sinal informa erros de endereços de memória;

TERM (**15**)- esse sinal tem a função de terminar completamente o processo, ou seja, este deixa de existir após a finalização;

ILL - esse sinal informa erros de instrução ilegal, por exemplo, quando ocorre divisão por zero;

KILL (**9**)- esse sinal tem a função de "**destruir**" um processo e é usado em momentos de criticidade.

Gerenciamento de processos

Sinais de processos

Alguns sinais com variam conforme a arquitetura de hardware.

 # man 7 signal

O **kill** também é um comando que o usuário pode usar para enviar qualquer sinal, mas se ele for usado de maneira isolada por padrão executa o sinal **TERM**.

A sintaxe para a utilização do comando **kill** é a seguinte:

 # kill -SINAL PID

Como exemplo, vamos supor que você deseja interromper temporariamente a execução do processo de PID **5550**. Para isso, pode-se usar o seguinte comando:

 # kill -STOP 5550

Para que o processo 5550 volte a ser executado, basta usar o comando:

 # kill -CONT 5550

Bibliografia

Linux – Guia do Administrador do Sistema
Autor: Rubem E. Pereira
Editora: Novatec

Manual Completo do Linux (Guia do Administrador)
Autor: Evi Nemeth, Garth Snyder, Trent R. Hein
Editora: Pearson Books

Guia Foca GNU/Linux
http://focalinux.cipsga.org.br/