Treinamento: FreeBSD – Introdução e Prática

Instrutor: Danilo Perillo Chiacchio

Nessa Aula Vamos Aprender:

- **Visualizar/Gerenciar os processos do sistema FreeBSD.**

Tópico 4: Administração do Sistema Operacional

Gerenciamento de Processos no Sistema

- É sempre válido ressaltar que o FreeBSD é um sistema multi-tarefa e que cada programa em execução é chamado de processo. Dessa forma, em um mesmo sistema FreeBSD podemos ter muitos, vários processos em execução;

- Cada processo é identificado unicamente no sistema por um número chamado de **Process ID (PID)**. Essa é a identificação “global” do processo no sistema e através dela podemos gerenciar o processo da maneira que precisarmos (alterar prioridade de execução, finalizar processo, verificar a sua execução, etc). Muito similar aos arquivos presentes no sistema operacional, cada processo possui um usuário e um grupo dono onde essas informações são utilizadas para determinar quais arquivos/diretórios e demais acessos o processo poderá possuir. Muitos processos possuem processos **“pai”** na qual são executados a partir desses processos. Por exemplo: o shell csh é um processo do sistema onde qualquer comando que é executado sobre o shell terá o mesmo como processo “pai” (parent).

Tópico 4: Administração do Sistema Operacional

Gerenciamento de Processos no Sistema

- **Nota:** A única exceção para isso é o processo “init”. Esse é o primeiro processo carregado no sistema e possui o PID igual a 1.

- Alguns programas não são projetados para serem executados com interação contínua do usuário. Um exemplo são servidores WEB e servidores de E-mail que, mesmo sem interação do usuário com frequência sempre ficam em execução. Esses tipos de programas são chamados de “daemons”. Uma convenção importante para descrever programas que se comportam como daemon e a letra “d” no final do nome do programa. Por exemplo:

  - BIND é um programa para servidor de nomes DNS e seu daemon é chamado de named;
  - Apache é um programa para servidor web e seu daemon é chamado de httpd.

- É importante ressaltar que isso não é uma regra e sim apenas uma convenção. Nem todos programas seguem essa convenção, como por exemplo o Sendmail.

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

- Para visualizar os processos em execução no sistema podemos utilizar dois comandos: **“ps” e “top”;**
- O **“ps”** basicamente apresenta uma lista “estática” dos processos em execução no sistema naquele momento;
- Já o **“top”** lista os processos em execução no sistema de forma “dinâmica”, atualizando a sua listagem a cada poucos segundos (2 segundos por padrão) de forma automática.
- Exemplo de execução do comando “ps”:

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

```
root@freebsd01:~ # ps
PID  TT  STAT      TIME COMMAND
626  v0  Ss    0:00.02 login [pam] (login)
643  v0  S     0:00.03 -csh (csh)
646  v0  R+    0:00.00 ps
627  v1  Is+   0:00.00 /usr/libexec/getty Pc ttyv1
628  v2  Is+   0:00.00 /usr/libexec/getty Pc ttyv2
629  v3  Is+   0:00.00 /usr/libexec/getty Pc ttyv3
630  v4  Is+   0:00.00 /usr/libexec/getty Pc ttyv4
631  v5  Is+   0:00.00 /usr/libexec/getty Pc ttyv5
632  v6  Is+   0:00.00 /usr/libexec/getty Pc ttyv6
633  v7  Is+   0:00.00 /usr/libexec/getty Pc ttyv7
root@freebsd01:~ #
```

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

- **Nota:** Por padrão, o “ps” somente mostra processos do usuário corrente, ou seja, nesse exemplo somente estão sendo mostrados processos do usuário root. Para listar processos de outros usuários juntamente, devemos utilizar a opção “-au”. Além disso, outras opções serão exibidas como USER, PID, % de consumo de CPU e Memória entre outras informações:

```
root@freebsd01:~ # ps -au
USER    PID %CPU %MEM   VSZ  RSS TT  STAT STARTED    TIME COMMAND
root    626  0.0  0.6 47756 2936 v0  Is    6:59PM 0:00.02 login [pam] (login)
root    643  0.0  0.7 23600 3668 v0  I+    7:02PM 0:00.05 -csh (csh)
root    627  0.0  0.4 14520 2084 v1  Is+   6:59PM 0:00.00 /usr/libexec/getty Pc ttyv1
root    628  0.0  0.4 14520 2084 v2  Is+   6:59PM 0:00.00 /usr/libexec/getty Pc ttyv2
root    629  0.0  0.4 14520 2084 v3  Is+   6:59PM 0:00.00 /usr/libexec/getty Pc ttyv3
root    630  0.0  0.4 14520 2084 v4  Is+   6:59PM 0:00.00 /usr/libexec/getty Pc ttyv4
root    631  0.0  0.4 14520 2084 v5  Is+   6:59PM 0:00.00 /usr/libexec/getty Pc ttyv5
root    632  0.0  0.4 14520 2084 v6  Is+   6:59PM 0:00.00 /usr/libexec/getty Pc ttyv6
root    633  0.0  0.4 14520 2084 v7  Is+   6:59PM 0:00.00 /usr/libexec/getty Pc ttyv7
suporte 667  0.0  0.6 17096 2768  0  Is+   7:04PM 0:00.01 -sh (sh)
root    677  0.0  0.7 23600 3576  1  Ss    7:04PM 0:00.03 -csh (csh)
root    716  0.0  0.5 18768 2312  1  R+    7:07PM 0:00.00 ps -au
```

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

- Conforme demonstra a figura, a saída do comando “ps” é estruturada em colunas onde cada coluna possui seu significado, como por exemplo:

- **USER =** Nome do usuário sob o qual o processo está em execução;

- **PID =** Process ID do processo. Trata-se de um número único no sistema que serve para identificar o processo de forma “global” no sistema. Não podem ter dois processos com o mesmo PID no sistema;

- **%CPU =** Porcentagem de CPU que o processo está consumindo do sistema;

- **%MEM =** Porcentagem de Memória RAM que o processo está consumindo do sistema;

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

- **TT =** Informação do canal tty que está executando o processo;
- **STAT =** Estado do processo;
- **STARTED =** Tempo na qual o processo foi iniciado no sistema;
- **TIME =** Porção de tempo que o processo está em execução na CPU do sistema (e não o tempo na qual o processo está em execução desde seu ínicio);
- **COMMAND =** Comando utilizado para iniciar/execução o processo/programa.

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

- A saída do comando **“top”** é similar, porém a principal diferença é que com “top” as informações sobre os processos são atualizadas de forma dinâmica e automática a cada 2 segundos por padrão;

- Por exemplo:

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

```
last pid:   923;  load averages:  0.14,  0.15,  0.15
23 processes:  1 running, 22 sleeping
CPU:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
Mem: 3368K Active, 25M Inact, 41M Wired, 16M Buf, 404M Free
Swap: 512M Total, 512M Free

  PID USERNAME    THR PRI NICE   SIZE    RES STATE    TIME    WCPU COMMAND
  536 root          2  20    0 22036K 13960K kqread   0:01   0.00% ntpd
  572 root          1  20    0 24156K  5848K select   0:00   0.00% sendmail
  674 root          1  20    0 86584K  7612K select   0:00   0.00% sshd
  643 root          1  20    0 23600K  3668K ttyin    0:00   0.00% csh
  677 root          1  29    0 23600K  3576K pause    0:00   0.00% csh
  579 root          1  20    0 16624K  2348K nanslp   0:00   0.00% cron
  376 root          1  20    0 14520K  2112K select   0:00   0.00% syslogd
  663 root          1  20    0 86584K  7568K select   0:00   0.00% sshd
  626 root          1  20    0 47756K  2936K wait     0:00   0.00% login
  666 suporte       1  20    0 86584K  7596K select   0:00   0.00% sshd
  923 root          1  20    0 21948K  3028K RUN      0:00   0.00% top
  667 suporte       1  21    0 17096K  2768K ttyin    0:00   0.00% sh
  302 root          1  20    0 13628K  4940K select   0:00   0.00% devd
  629 root          1  52    0 14520K  2084K ttyin    0:00   0.00% getty
  575 smmsp         1  20    0 24156K  5608K pause    0:00   0.00% sendmail
  627 root          1  52    0 14520K  2084K ttyin    0:00   0.00% getty
  630 root          1  52    0 14520K  2084K ttyin    0:00   0.00% getty
  569 root          1  20    0 61316K  7084K select   0:00   0.00% sshd
  631 root          1  52    0 14520K  2084K ttyin    0:00   0.00% getty
  632 root          1  52    0 14520K  2084K ttyin    0:00   0.00% getty
  628 root          1  52    0 14520K  2084K ttyin    0:00   0.00% getty
  132 root          1  52    0 12360K  1756K pause    0:00   0.00% adjkerntz
  633 root          1  52    0 14520K  2084K ttyin    0:00   0.00% getty
```

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

- Conforme podemos visualizar na figura anterior, a saída do comando “top” é divida em duas sessões: A primeira é uma espécie de cabeçalho contendo as 5 ou 6 primeiras linhas. Nessa primeira sessão temos várias informações importantes, como:
  - PID do último processo executado;
  - Load average do sistema (informação utilizada para mensurar o quão utilizado/ocupado está o sistema). Quando maior o load average maior é o consumo de recursos como CPU e Memória por exemplo;
  - Número total de processos;
  - Número de processos em execução; processos em sleeping; processos “mortos” (kill);
  - Informações de consumo de CPU, Memória RAM, Swap e outras informações.

Tópico 4: Administração do Sistema Operacional

Visualizando Processos em Execução

- A segunda sessão contém informações sobre os processos, onde as informações são separadas por colunas (similar a saída do comando “ps”). É importante ressaltar que, por padrão, o “top” mostra a quantidade de memória utilizada pelo processo. Essa informação é divida em duas colunas, onde:
  - **SIZE =** Tamanho total de espaço em memória que a aplicação/processo necessita para a sua operação;
  - **RES =** Resident Size é o espaço em memória sendo utilizado pela aplicação/processo no momento de sua execução.

Tópico 4: Administração do Sistema Operacional

Finalizando Processos em Execução

- Uma maneira que o sistema utiliza para se comunicar com os processos/daemons é através de sinais utilizando para isso o aplicativo “kill”. Existem vários tipos de sinais que podem ser utilizados juntamente com o comando “kill”. Cada sinal é representado por um valor númerico. Os sinais mais comuns e utilizados são:

1 HUP (hang up)
2 INT (interrupt)
3 QUIT (quit)
6 ABRT (abort)
9 KILL (non-catchable, non-ignorable kill)
14 ALRM (alarm clock)
15 TERM (software termination signal)

Tópico 4: Administração do Sistema Operacional

Finalizando Processos em Execução

- Um usuário somente consegue enviar sinais para processos na qual seja dono, com exceção do usuário root que consegue se comunicar com qualquer processo/daemon em execução no sistema. É importante ressaltar que cada processo/daemon pode se comportar de maneiras diferentes ao receber um sinal via comando “kill”. Por isso é importante consultar a documentação da aplicação para saber mais detalhes a respeito.

- Como exemplo, vamos enviar um sinal “HUP” para a sessão SSH do usuário “suporte”. O primeiro passo é identificar qual PID (Process ID) dessa sessão SSH:

```
root@freebsd01:~ # ps auxww | grep suporte
root    663   0.0  1.5 86584  7568  -  Is    7:04PM   0:00.03 sshd: suporte [priv] (sshd)
suporte 666   0.0  1.5 86584  7596  -  I     7:04PM   0:00.01 sshd: suporte@pts/0 (sshd)
suporte 667   0.0  0.6 17096  2768  0  Is+   7:04PM   0:00.01 -sh (sh)
root    993   0.0  0.1   392   312  1  R+    9:43PM   0:00.00 grep suporte
```

Tópico 4: Administração do Sistema Operacional

Finalizando Processos em Execução

- Conforme podemos visualizar na figura anterior, o PID da sessão SSH do usuário suporte é 666. Vamos enviar o sinal de HUP:

```
root@freebsd01:~ # kill -s HUP 666
root@freebsd01:~ #
root@freebsd01:~ #
root@freebsd01:~ # ps auxww | grep suporte
root@freebsd01:~ #
root@freebsd01:~ #
```

Observe que o comportamento foi desconectar/desligar a conexão SSH do usuário suporte. Para mais detalhes, consulte “man kill”.