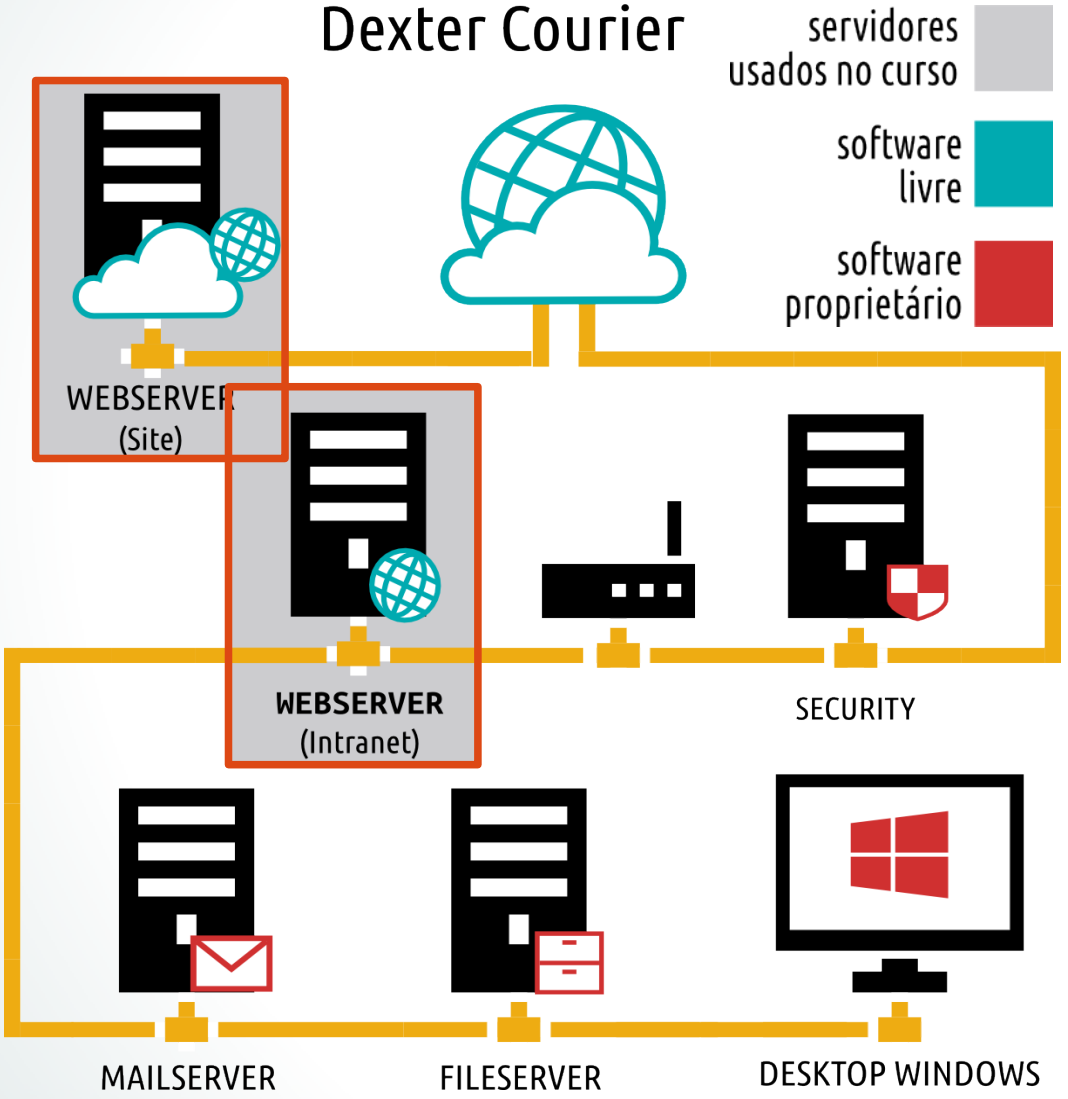

4LINUX
FREE SOFTWARE SOLUTIONS

Só na 4Linux você aprende MUITO MAIS!

Gerenciando Processos

IT Experience



Nesta Aula:

- Usaremos os Servidores da Dexter:
- WebServerInterno

Objetivos da Aula

- Conhecer os estados dos processos;
- Gerenciar processos:
  - Filtrar informações;
  - Enviar sinais;
  - Gerenciar segundo e primeiro plano;
  - Alterar prioridade.

Gerenciando Processos

Características:

- Um processo é um programa em execução;
- Todo processo possui um PID único;
- É possível obter informações dos processos através do /proc;
- Todo sistema operacional trabalha com processos, mesmo o usuário não tendo acesso a estes;
- O pai de todo processo é o init, com PID 1.

Gerenciando Processos

Estados dos processos:

D → Processo morto (usually IO);

R → Running (na fila de processos);

S → Dormindo Interruptamente (aguardando um evento terminar);

T → Parado, por um sinal de controle;

Z → Zombie, terminado mas removido por seu processo pai.

Gerenciando Processos

Filtrar informações sobre processos

Comandos:

- ps aux
- pstree
- pgrep
- pidof
- top
- htop
- lsof

Filtrando Processos

- Filtrando informações de processos:

```
1# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2   2900  1424 ?        Ss   11:36   0:02 /sbin/init
                    ..........................................
```

USER → Usuário que iniciou o processo (dono).
PID → Número único do processo.
%CPU → Utilização da CPU em porcentagem.
START → A hora em que o processo foi iniciado. Caso a hora seja do dia anterior, é representado pelo dia e mês.
COMMAND → O comando executado e todos seus argumentos.

Filtrando Processos

- Filtrando informações de processos:

```
1# pstree
init─┬─atd
     ├─auditd───{auditd}
.....................................
2# pstree -p -a
```

O comando **pstree** mostra a estrutura de processos em execução no sistema em forma de árvore. O parâmetro "**-p**" mostra o PID do processo e o "**-a**" para o caminho completo.

Filtrando Processos

- Filtrando informações de processos:

```
1# pgrep crond
   1294 (PID do processo crond)
2# pgrep cro
   1294
```

Utilitário **pgrep** usa expressões regulares semelhante ao comando grep buscando por parte do nome do processo, retornando apenas seu **PID**.

Filtrando Processos

- Filtrando informações de processos:

```
1# pidof crond

   1294 (PID do processo crond)

2# pidof cron
```

Humm... Algo Errado!

Para utilizar o comando **pidof** deve ser passado o nome **exato** do daemon, diferente do pgrep que pode ser passado apenas uma parte do nome.

Filtrando Processos

- Filtrando informações de processos:

```
1# top
2# top -d 1        (tempo de atualização)
3# top -n 3        (Quantidade de vezes que será atualizado)
4# htop
```

O comando "**top**" é um utilitário disponibilisado pelo Linux para monitorar o desempenho do sistema.
O "**htop**" é um avançado sistema interativo visualizador de processos.

Filtrando Processos

- Filtrando informações de processos:

```
1# lsof      (Listar todos os arquivos abertos)
2# lsof -u root      (Listar todos os arquivos abertos pelo root)
3# lsof -i     (Listar as conexões ativas, parecido com netstat)
4# lsof /bin/bash  (Listar quem está usando o arquivo ou diretório)
```

O comando “lsof” (**L**ist **O**pen **F**iles) é utilizado para mostrar os arquivos que estão abertos no sistema.

Gerenciando Processos

Enviar sinais aos processos

Comandos:

- kill -l
- kill
- killall
- pkill

Gerenciando Processos

- Listando os sinais:

```
1# 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
                ........................................
```

SIGHUP (1): Reinicia o processo.
SIGINT (2) :Interrompe o processo.
SIGQUIT (3): Fecha o processo.
SIGKILL (9): Finaliza o processo. Imediatamente (Mata o processo).
SIGTERM (15): Solicita ao processo que termine.

Gerenciando Processos

- Enviando o sinal:

```
1# cd /root

2# vim arquivo.txt

Linux is Open Source.
```

< Mantenha o vim ABERTO >

- Acesso outro terminal (tty2):

< No virtualbox, tecle CTRL (direito) + F2 >

Vamos criar um cenário no qual o usuário root estará editando um arquivo e em outro terminal iremos encerrar o Editor de Texto VIM!

Gerenciando Processos

- Enviando o sinal:

```
1# ps aux | grep vim
root 10554  0.0  0.6  11056  3424 tty1     S+   14:49   0:00 vim primeiro.txt
2# kill 10554
```

< Volte para o tty1 >

```
Vim: Caught deadly signal TERM
Vim: preserving files...
Vim: Finished.
```

O sinal padrão que é enviado para um determinado processo é o sinal **15 (TERM)**.

Gerenciando Processos

- Enviando o sinal:

```
1# pidof vim
   10562    (PID do processo VIM)
2# kill -9 10562
```

ou

```
3# kill -KILL 10562
```

Faça o mesmo procedimento realizado anteriormente, abra um arquivo com VIM no tty1 (Deixe o VIM Aberto) e logue no tty2.

O comando **kill** faz o tratamento apenas pelo **PID**.

Atenção! Muito cuidado ao enviar o sinal **(9) KILL**, este sinal encerra o proceso bruscamente. Se fosse o processo do mysql, pode ocorrer de corromper a base.

Gerenciando Processos

- Enviando o sinal:

```
1# crond ; crond ; crond
2# pidof crond
  10758 10756 1294 1297
3# kill $(pidof crond)
```

ou

```
3# killall crond;
4# pidof crond
```

Para enviar sinal para todos os processos do **crond**, podemos utilizar o comando **killall**.

O **killall** trata processos apenas com o nome exato.

crond → Daemon do crontab.

Gerenciando Processos

- Enviando o sinal:

```
1# crond ; crond ; crond
2# pidof crond
  10758 10756 1294 1297
3# pkill cron
4# pidof crond
```

Com o **pkill** conseguimos tratar um processo colocando apenas uma parte do nome.

Diferente do killall que precisa ser o nome exato do processo.

Gerenciando Processos

Gerenciar segundo e primeiro plano

Comandos:

- jobs
- CTRL + Z e &
- fg
- bg
- nohup

Gerenciando Processos

- Programa em Segundo Plano:

```
1# jobs
2# vim primeiro.txt
   Teste 1
    < Tecle: CTRL + Z >
3# jobs
   [1]+  Parado    vim primeiro.txt
4# fg 1
```

Execute **jobs** para listar os programas em segundo plano do terminal atual.

CTRL + Z joga o programa para segundo plano.

Para voltar ao programa **fg N**, onde N é a identificação da job.

Gerenciando Processos

- Programa em Segundo Plano:

```
1# vim segundo.txt &
2# jobs
[1]-  Parado      vim primeiro.txt
[2]+  Parado      vim segundo.txt
3# kill -9 %2 (Encerrar Job)
4# jobs
5# logout
```

Iniciar um programa em segundo plano com "**&**" no final.

As jobs são encerradas ao terminar a seção.

Gerenciando Processos

- Iniciando ntpo em segundo plano:

```
1# nohup ping 8.8.8.8 &
2# ps aux | grep ping
3# logout
```

< Logue novamente >

```
4# ps aux | grep ping
5# tail -f nohup.out
```

Iniciar um programa em segundo plano sem prender ele ao terminal (**nohup**).

O **nohup** registra um arquivo no local em que foi executado com a saída do comando.

Gerenciando Processos

Alterar prioridade dos processos

Comandos:

- renice
- top
- htop
- nice
- ps aux

Gerenciando Processos

- Alterar prioridade de um processo:

```
1# top
```

< Tecle R >

```
PID to renice: NNNN

Renice PID NNNN to value: N
```

```
2# htop
```

< Selecione o Processo >

F7 para diminuir a prioridade.

F8 para aumentar a prioridade.

As prioridades variam de **-20** à **19**.

-20 para **maior prioridade**.

19 para **menor prioridade**.

Gerenciando Processos

- Iniciar com a prioridade alterada:

```
1# killall cron
2# nice --20 crond
3# htop
```

< F4 Filter e busque por cron >

```
4# pgrep cron
4693
5# renice -2 4693
```

Utilizando o **nice** para iniciar um programa já com prioridade alterada.

Para alterar a prioridade de um programa em execução utiliza-se o **renice**.

A prioridade **padrão é 0**.

Pergunta LPI

Qual é o PID do processo init ?

A. 0

B. 1

C. Ele recebe o número do runlevel corrente

D. Ele recebe um PID diferente a cada inicialização do sistema

Pergunta LPI

Qual é o PID do processo init ?

A. 0

B. 1

C. Ele recebe o número do runlevel corrente

D. Ele recebe um PID diferente a cada inicialização do sistema

Resposta: B

Pergunta LPI

Qual o comando para matar todos os processos com o mesmo nome? (Selecione as duas corretas)

A. kill

B. killall

C. pkill

D. killterm

Pergunta LPI

Qual o comando para matar todos os processos com o mesmo nome? (Selecione as duas corretas)

A. kill

B. killall

C. pkill

D. killterm

Resposta: B e C


4LINUX
FREE SOFTWARE SOLUTIONS