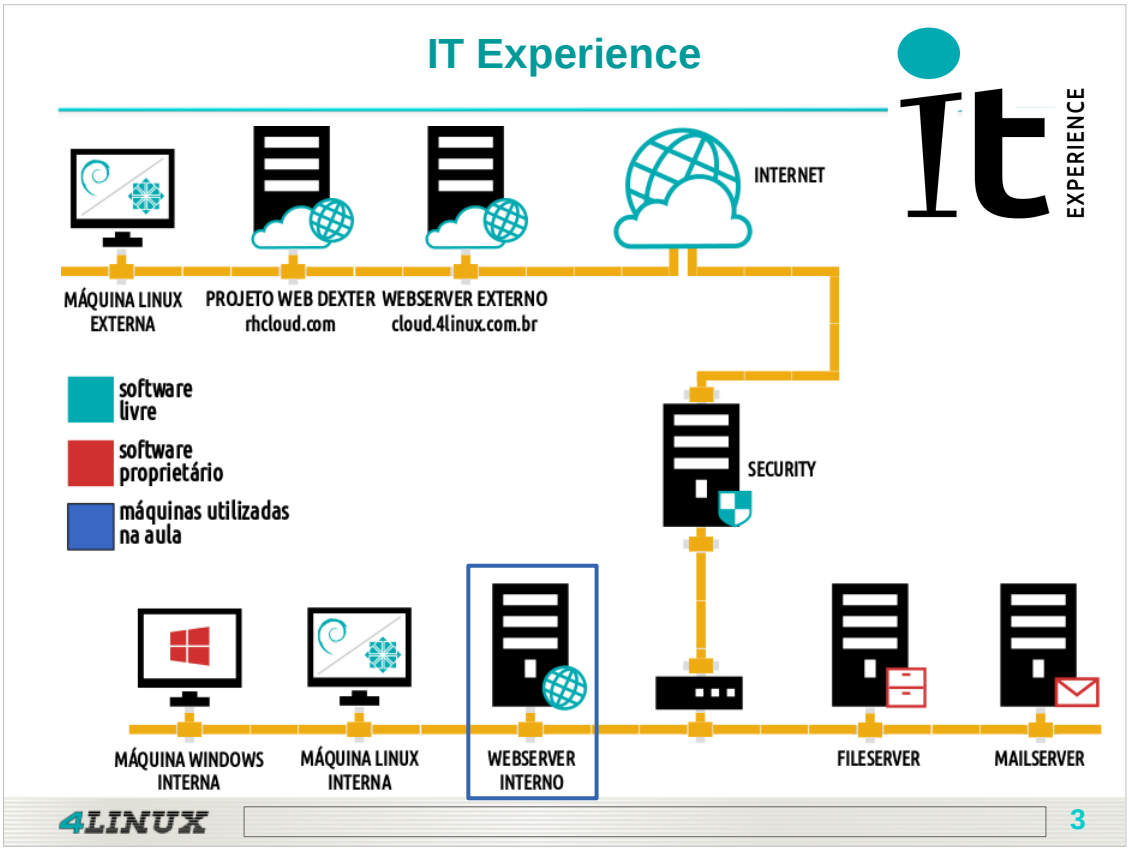
Curso 450

Linux Fundamentals in Cloud

Versão 2015_3.0

Gerenciando Processos

Quando executamos algum comando, script ou iniciamos algum programa, o kernel atribui a ele um número chamado PID e passa a gerenciar a quantidade de recursos que ele irá disponibilizar para essa atividade, cada uma dessas execuções é definida pelo sistema operacional como um processo.


IT Experience
IT EXPERIENCE
INTERNET
MÁQUINA LINUX EXTERNA
PROJETO WEB DEXTER rhcloud.com
WEBSERVER EXTERNO cloud.4linux.com.br
software livre
software proprietário
máquinas utilizadas na aula
SECURITY
MÁQUINA WINDOWS INTERNA
MÁQUINA LINUX INTERNA
WEBSERVER INTERNO
FILESERVER
MAILSERVER
4LINUX
3

Anotações:

Objetivos da Aula

Aula 07

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

Anotações:

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.

A forma como o kernel gerência os processos é bastante inteligente e podemos sempre visualizar o status do processo num determinado instante, para determinar se ele está sendo executado neste mesmo instante ou se ele está aguardando tempo de máquina para que seja executado.

Gerenciando Processos

Filtrar Informações sobre Processos:

Comandos:

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

Gerenciamento de Tarefas

Haverá sempre diversos processos rodando simultaneamente na máquina sendo que o kernel possui uma lista dos processos que necessitam de recursos, Já que não existe atualmente um sistema realmente multitarefa, capaz de realizar diversas atividades realmente ao mesmo tempo, o fica a cargo do kernel criar uma fila de processos e a percorrer disponibilizando recursos de máquina para cada um deles por um determinado período de tempo.

Quanto melhor essa distribuição for efetuada melhor será o desempenho do sistema como um todo e mais próximo de um sistema realmente multitarefas o sistema estará.

Filtrando Processos

Servidor: Webserver Interno

Filtrar Informações sobre 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.

Parâmetros do comando ps:

A → Mostra todos os processos não associados com um terminal;

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;

Outras informações disponibilizadas pelo comando ps:

% MEM → Porcentagem da memória usada ;

VSZ → Indica o tamanho virtual do processo ;

RSS → Resident Set Size, indica a quantidade de memória em uso (KB);

TTY → Indica o identificador do terminal do processo ;

STAT → Indica o estado atual do processo , sendo representado por uma letra conforme a descrição da próxima página.

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 não removido por seu processo pai.

Ao efetuar listagem de processos as letras acima podem ser combinadas ou acrescidas dos seguintes valores:

> → Processo rodando com prioridade maior que a padrão;

< → Processo rodando com prioridade menor que a padrão;

+ → Processo pai, ou seja, é um processo que possui processos filhos;

S → Processo tipo “session leader”, um processo que possui processos dependentes dele;

l → Processo que possui múltiplas threads;

L → Processo que possui páginas travadas na memória;

N → Processo que foi definido com uma prioridade diferente da padrão, tendo sido através de comandos executados pelo usuário.

Filtrando Processos

Servidor: Webserver Interno

Filtrando Informações sobre 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.

Hierarquia de Processos

Com relação a gerência de processos, é muito comum que processos criados pelo Kernel sejam partes de um outro processo já existente, Essa chamadas de kernel utilizada na criação de processos recém o nome de threads.

O comando pstree é uma ótima opção para visualizar esse cenário em forma de árvore, representando as dependências entre os processos.

Filtrando Processos

Servidor: Webserver Interno

Filtrando Informações sobre 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.

Anotações:

Filtrando Processos

Servidor: Webserver Interno

Filtrando Informações sobre Processos:

```
1# pidof crond

   1294 (PID do processo crond)
2# pidof cron
```

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.

Anotações:

Filtrando Processos

Servidor: Webserver Interno

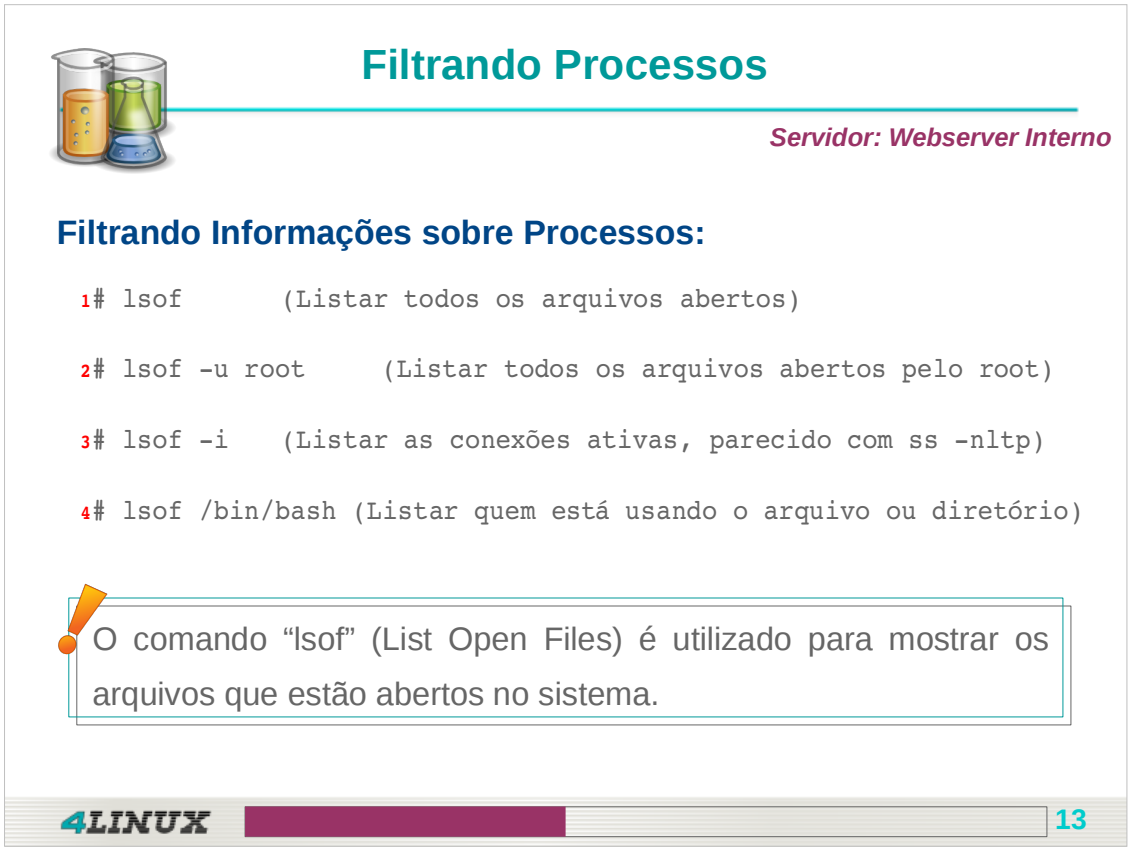
Filtrando Informações sobre Processos:

```
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 disponibilizado pelo Linux para monitorar o desempenho do sistema.

O “htop” é um avançado sistema interativo visualizador de processos.

Com o top podemos ver o horário atual, quanto tempo a máquina está ligada, quantos usuários estão logados, quantos processos estão em aberto, rodando, em espera e processos zumbi, mas sua principal característica é a organização dos processos em uma espécie de ranking baseado no consumo de recursos.



lsof

O comando lsof é um dos mais importantes comandos para quem administra sistemas Linux, principalmente na área de segurança. Este comando lista todos os arquivos abertos por todos os processos.

Sua listagem inclui também recursos que funcionam como arquivos (podem ser abertos, mapeados na memória, entre outros). Tal como bibliotecas, sockets e etc.

Alguns dos usos mais comuns do lsof incluem:

- Ver se algum processo está escutando uma porta na rede;
- Ver que processo está usando um certo arquivo (lsof);
- Ver que tipo de conexão esta sendo feita no sistema;
- Medir as memórias utilizadas pelos processos.

Gerenciando Processos

Enviar sinais aos Processos:

Comandos:

- kill -l;
- kill;
- killall;
- pkill.

E como fazemos para gerenciar os processos?

Apesar do kernel gerenciar os processos, nós podemos enviar sinais a esses processos requisitando que eles alterem seu comportamento. Utilizamos os comandos acima para enviar sinais de controle a determinados processos.

DICA:

Nem todos os sinais existentes possuem uma utilidade prática atualmente, sendo que, os mais importantes foram listados acima, caso queira uma listagem completa dos sinais possíveis você poderá obtê-la na seção STANDARD SIGNALS do **man 7 signal.**

Gerenciando Processos

Servidor: Webserver Interno

Enviando o sinal:

```
1# cd /root

2# vim arquivo.txt
 Linux is Open Source.
```

< Mantenha o vim ABERTO >

Acesse 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!

Anotações:

Gerenciando Processos

Servidor: Webserver Interno

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).

Anotações:

Gerenciando Processos

Servidor: Webserver Interno

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.

Anotações:

Gerenciando Processos

Servidor: Webserver Interno

Enviando o sinal:

```
1# pidof crond
2# 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.

Anotações:

Gerenciando Processos

Servidor: Webserver Interno

Enviando o sinal:

```
1# pgrep httpd
2# pidof httpd
  2095 2094 2093 2092 2091
3# pkill http
4# pidof httpd
```

Com o **pkill** conseguimos tratar um processo colocando apenas uma parte do nome. Diferente do killall, que precisa ser o nome exato do processo.

Anotações:

Laboratório Dexter

Servidor: Webserver Interno

Dropando conexão indesejada:

- Você está suspeitando que alguém está acessando o seu servidor de forma não autorizada, e quer matar o processo em que a conexão remota gerou.
- Faça um ssh da **Linux Interna** para a máquina **Webserver Interno**:

```
1# ssh root@192.168.200.20
```


Anotações:

Laboratório Dexter

Servidor: Webserver Interno

Dropando conexão indesejada:

- Primeiramente é necessário verificar em qual terminal este usuário está logado:

```
1# w
```

```
 15:42:47 up  1:18,  2 users,  load average: 0,00, 0,01, 0,04
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1                      14:24     6:13   0.42s  0.32s -bash
root     pts/0    scorpion.local   15:42     0.00s  0.03s  0.00s w
```

- Repare que há uma conexão no terminal /dev/pts/0:

```
2# fuser -k /dev/pts/0
```

Com o comando fuser também é possível matar processos.

Anotações:

Gerenciando Processos

Gerenciar Segundo e Primeiro Plano:

Comandos:

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

Gerenciando Segundo e Primeiro Plano

Em se tratando da execução de processos, é possível criar processos vinculados ao usuário porém com execução em background (Segundo Plano) este procedimento permite executar determinado processo e ao mesmo tempo manter o terminal livre.

Processos que trabalham de forma interativa, ou seja, ecoando informações no terminal e interagindo com o usuário não podem ser executados em background como por exemplo o ping, ao executar tais processos com a opção & que o colocaria em background, estes processos continuam a ecoar informações no terminal do usuário.

Gerenciando Processos

Servidor: Webserver Interno

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.

Anotações:

Gerenciando Processos

Servidor: Webserver Interno

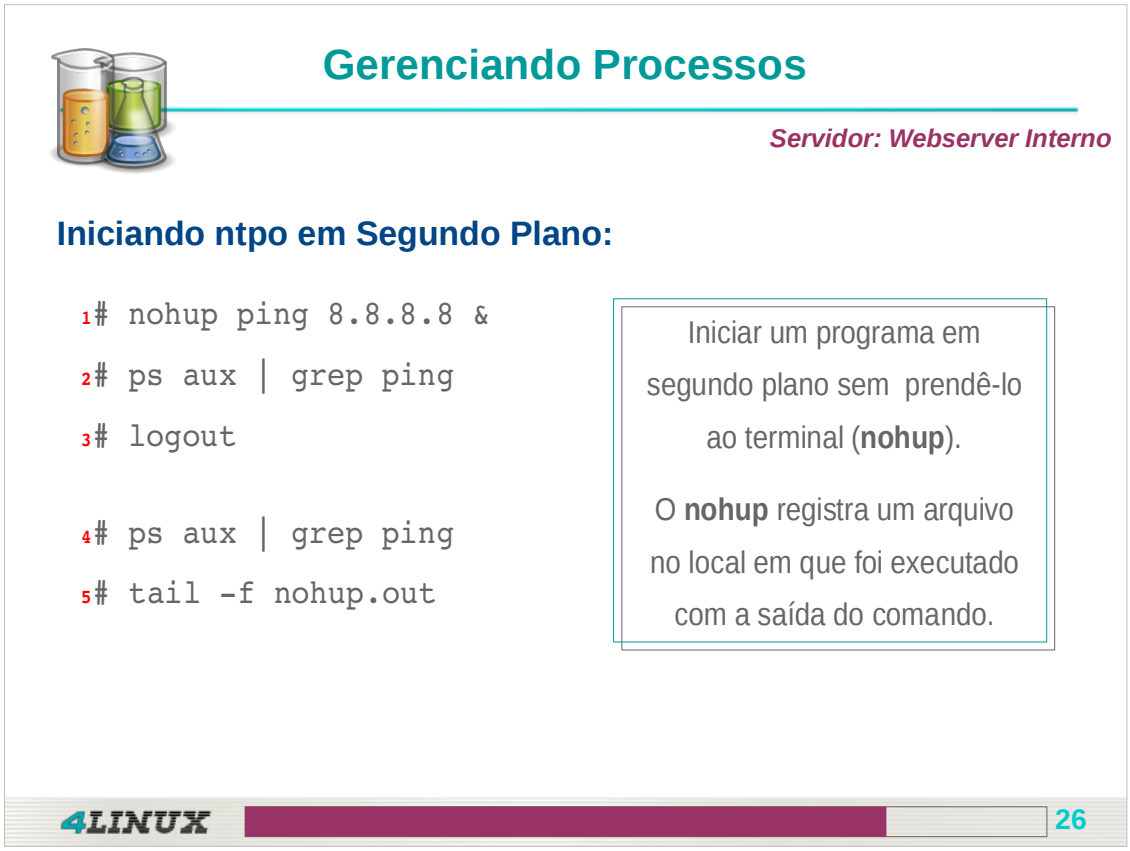
Programa em Segundo Plano:

```
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.

Anotações:



NOHUP

O comando nohup ignora os sinais de interrupção de conexão durante a execução do comando especificado. Assim, é possível ao comando continuar sua execução mesmo depois que o usuário se desconectar do sistema.

Se a saída padrão é uma tty, esta saída e o erro padrão são redirecionados para o arquivo “nohup.out” (primeira opção) ou para o arquivo “$HOME/nohup.out” (segunda opção).

O nohup não coloca o comando que ele executa em background. Isto deve ser feito explicitamente pelo usuário como no exemplo acima onde executamos um ping com o nohup.

Gerenciando Processos

Alterar Prioridade dos Processos

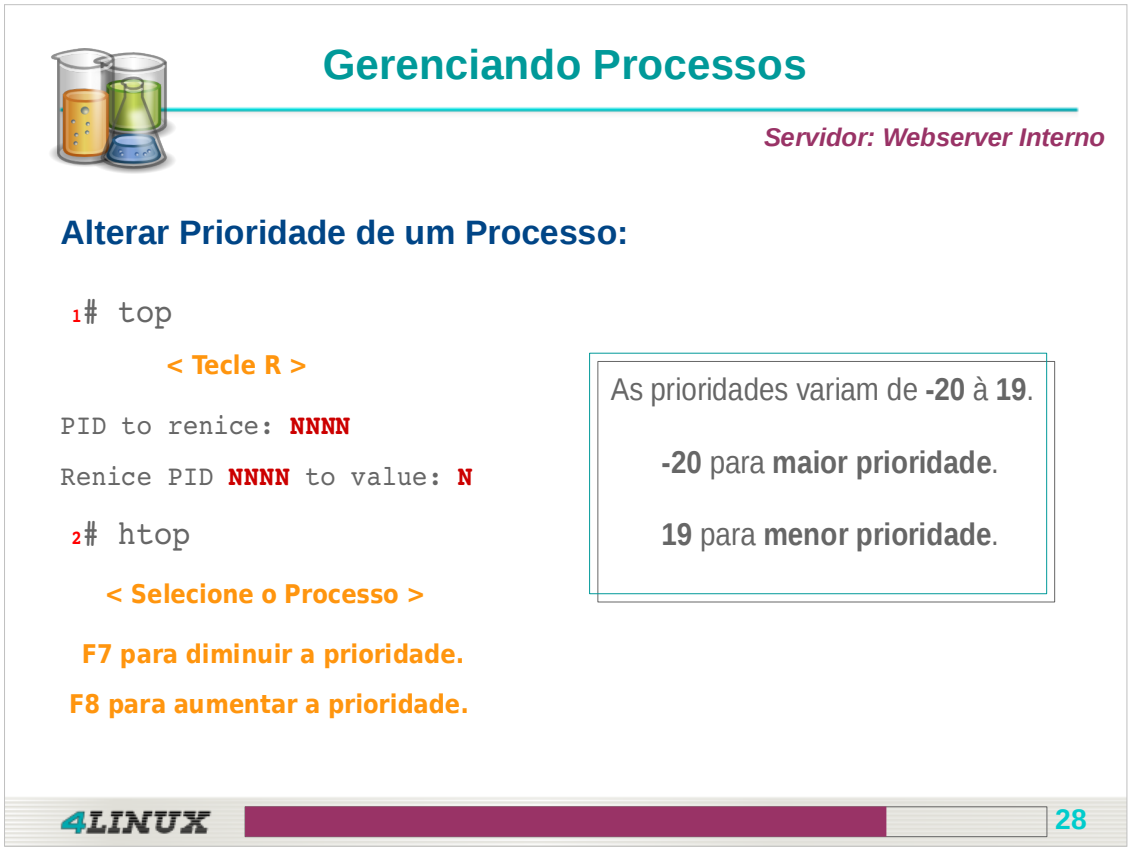
Comandos:

- renice;
- top;
- htop;
- nice;
- ps aux.

Prioridades de Execução

Sabemos que sistemas GNU/Linux criam uma lista de processos e executaM um a um de tempos em tempos, esta lista possui uma prioridade, o grau de importância para execução do processo.

Prioridades são definidas conforme o programa que está em execução, por exemplo um programa que depende de I/O tem maior prioridade, quem faz esse cálculo é o próprio sistema, quanto menor a prioridade de execução, maior será o uso do processador, Também podemos interagir com as prioridades de execução, através dos comandos **nice** e **renice.**



Prioridades de Execução

As prioridades podem variar do número **-20** que é a prioridade mais importante até o **+19** que representa a prioridade menos importante.

Por padrão todos os processos executados como **"root" tem a prioridade 0, já os processos executados com usuários comum tem a prioridade 10**, o usuário root pode mudar entre -20 e +19, enquanto que o usuário comum pode mudar esta prioridade entre 0 e +19.

DICA:

Perguntas relacionadas com os valores de prioridade acima são muito comuns em provas LPI.

Gerenciando Processos

Servidor: Webserver Interno

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**.

Anotações:

Pergunta LPI

Qual é o PID do processo init?

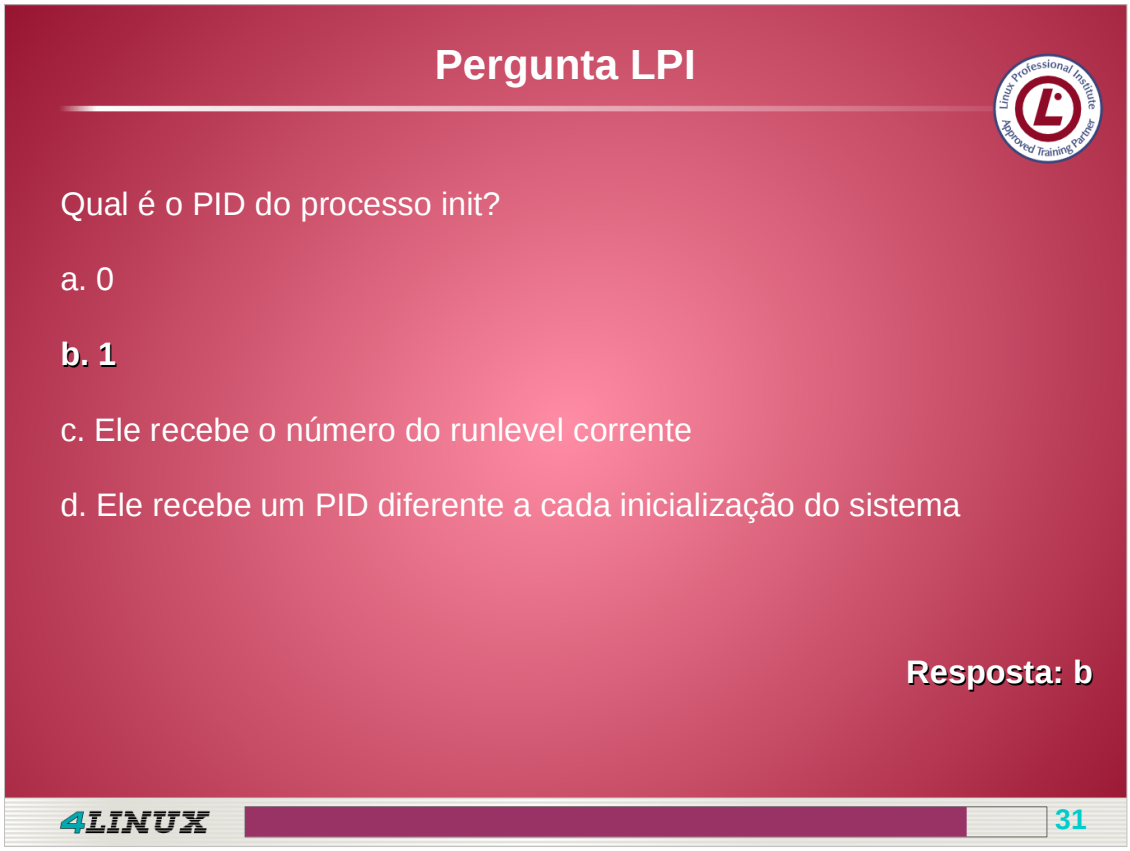
a. 0

b. 1

c. Ele recebe o número do runlevel corrente

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

Anotações:



Alternativa B: RESPOSTA CORRETA

O processo init é o principal processo para a operação do sistema Linux, e sendo o primeiro processo a ser executado sempre terá PID com valor igual a 1, Este processo é invocado logo após o carregamento do Kernel na memória e possui a função de controlar todos os outros processos que são executados no computador.

Pergunta LPI

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

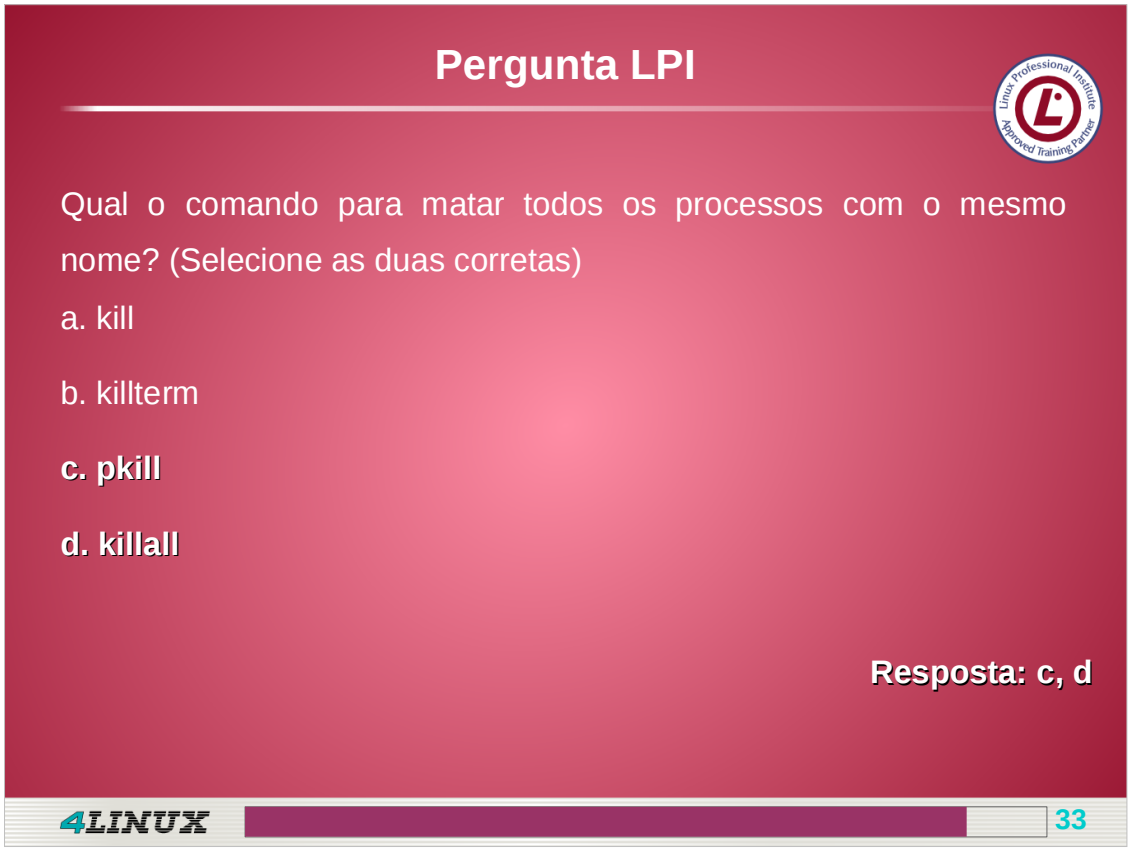
a. kill

b. killall

c. pkill

d. killterm

Anotações:



Alternativas C & D: RESPOSTA CORRETA!

O comando **pkill** permite enviar um sinal a um processo com base em uma busca por seu nome ou atributo, já o comando **killall** pode ser usado para matar todos os processos com determinado nome.

Próximos passos

Para que você tenha um melhor aproveitamento do curso, participes das seguintes atividades disponíveis no Netclass:

- Executar as tarefas do **Practice Lab**;
- Resolver o **Desafio OpenCloud Lab** e postar o resultado no Fórum Temático;
- Responder as questões do **Teste de Conhecimento** sobre o conteúdo visto em aula.

Mãos à obra!