ESCOLA LINUX

TREINAMENTOS

50 Comandos do Crontab para administradores de sistemas Linux

Já se deparou com a necessidade de fazer uma tarefa de modo programado? Não seria bom ter um programa capaz de fazer determinada tarefa em determinado dia/horario?
O Cron faz isso por você e nesse interessante material a Escola Linux preparou 50 comandos explicados para você facilitar sua vida com o Cron e o crontab, aproveite e bons estudos.

Comandos básicos do Crontab

Compreender os comandos básicos do crontab ajudará você a dominar a ferramenta a longo prazo. Abaixo, discutimos alguns comandos fundamentais e cruciais que aumentarão sua produtividade como administrador de sistemas Linux para um nível totalmente novo. Experimente-os agora mesmo para obter experiência em primeira mão.

1. Edite o Crontab

```
$ crontab -e
```

O comando acima é usado para chamar seu crontab padrão. Agora você pode editar este arquivo e inserir seus próprios trabalhos para executar em um determinado momento. Por padrão, este crontab deve conter alguns comentários que ajudarão você a entender sua funcionalidade em mais detalhes.

2. Lista Crontab

```
$ crontab -l
```

Você pode usar este comando para listar o crontab atual em uso. Este comando apenas despeja o conteúdo dentro do arquivo crontab na saída padrão. Pode ser útil ao inspecionar crontabs.

3. Edite o Crontab por usuário

```
$ crontab -u User -e
```

Digamos que seu sistema tenha um usuário chamado Usuário e você deseja editar a configuração crontab desse usuário. O comando acima demonstra como fazer isso. O nome de usuário é passado pela opção -u. Este é um comando útil para administradores de sistemas que precisam verificar outros usuários de um sistema

4. Listar Crontab por usuário

```
$ crontab -u User -l
```

A mesma estrutura de editar usuário é seguida ao listar o crontab de outro usuário. Tudo o que você precisa fazer é substituir a opção -e pela opção -l, como no exemplo.

5. Verifica o arquivo Crontab

```
$ sudo ls -l
/var/spool/cron/crontabs
```

Muitas vezes, você pode precisar verificar se um usuário específico possui ou não crontabs próprios.
Ele deve imprimir todos os crontabs disponíveis de cada usuário em seu sistema na saída padrão.

6. Exclua seu Crontab

```
$ crontab -r
```

Se você deseja finalizar todas as tarefas pré-agendadas, a exclusão do crontab é uma opção - o comando abaixo mostra como isso pode ser feito para o usuário conectado no momento.

7. Exclua o Crontab do usuário

```
$ crontab -u User -r
```

Para excluir a crontab do usuário, utilize esse comando.

8. Limite o acesso do Crontab a usuários específicos

```
$ ls /etc/cron.d/ |
grep ".allow"
```

Os administradores de sistemas podem querer limitar o acesso ao crontab apenas a alguns usuários especificados. Para fazer isso, precisamos do arquivo cron.allow. Execute o comando abaixo para verificar se ele existe no seu sistema ou não.
Se esse arquivo existir, você poderá editá-lo como root e especificar quem pode acessar os arquivos crontab no seu sistema.

9. Negar acesso do Crontab a usuários específicos

```
# vim /etc/cron.d/
cron.deny
```

Liste os nomes de usuário para os quais você deseja negar o acesso ao crontab. Esses usuários não poderão mais listar ou editar crontabs no seu sistema.

Comandos diários do Crontab para iniciantes

Agora, mostraremos alguns comandos crontab usados regularmente que tornarão a computação muito mais divertida para você. Esses comandos são genéricos para que você possa editá-los rapidamente com base em suas necessidades. Sugerimos que você tome cuidado, caso contrário, poderá fazer algo que acabará se arrependendo mais tarde.

10. Crie um backup de todas as contas de usuário às 5 da manhã diariamente

```
0 5 * * * tar -zcf
/var/backups/home.tgz
/home/
```

Coloque a entrada acima no seu arquivo crontab usando o comando **crontab -e** e o cron agora criará um backup de todas as contas de usuário em seu sistema às 5:00 em ponto todos os dias. O 0 no início indica o primeiro minuto e o 5 indica a hora 5 da manhã.

11. Crie um backup de todas as contas de usuário às 5 da manhã por semana

```
0 5 * * 1 tar -zcf
/var/backups/home.tgz
/home/
```

O comando crontab acima criará o arquivo de backup toda semana, e não todos os dias. Observe aqui como o valor da última vez foi substituído por 1 em vez de *.

12. Programe o Cron para executar um trabalho duas vezes por dia

```
0 5,17 * * *
/scripts/script.sh
```

O comando crontab acima fará com que o cron execute o script.sh executável às 5:00 e 17:00 diariamente. Observe como a vírgula foi usada para indicar vários valores de hora. Você pode adicionar mais valores usando uma lista separada por vírgula para executar a tarefa mais de duas vezes.

13. Programe Cron para executar um trabalho às 2 da manhã diariamente

```
0 2 * * * /bin/sh backup.sh
```

Se você adicionar a entrada acima no seu arquivo crontab, o cron executará o script backup.sh às 2 da manhã todos os dias.

14. Programe Cron para executar um trabalho às 3:15 da manhã diariamente

```
15 3 * * * /bin/sh script.sh
```

A entrada crontab acima executará o script bash chamado 'script.sh' às 3:15 da manhã todos os dias. Esse tipo de crontabs será benéfico ao agendar tarefas que precisam ser executadas todos os dias.

15. Programe Cron para executar um trabalho às 20h toda semana

```
0 20 * * 1 /bin/sh script.sh
```

O comando crontab acima fará com que o cron execute o arquivo script.sh às 20h toda semana. O valor da hora precisa ser especificado no formato de 24 horas para especificar valores de pm dentro de suas crontabs.

16. Programe Cron para executar um trabalho às 20h de segunda-feira

```
0 20 * * Mon /bin/sh script.sh
```

O comando crontab acima chama o cronjob às 20h toda segunda.

17. Programe um Trabalho Cron às 20h na segunda e no sábado

```
0 20 * * Mon,Sat /bin/sh script.sh
```

A entrada acima em seu crontab fará com que o cron execute o arquivo script.sh às 20h todas as segundas e sábados.

18. Programe um trabalho Cron para ser executado a cada minuto

```
* * * * * /scripts/script.sh
```

Você pode inclusive fazer com que um comando seja executado a cada minuto.
os asteriscos no campo de tempo dos seus crontabs significam que o arquivo script.sh será executado a cada minuto.

19. Programe um trabalho Cron para ser executado a cada 10 minutos

```
*/10 * * * * /scripts/script.sh
```

O comando acima fará com que o cron execute o script.sh executável a cada 10 minutos. O operador / é usado para atingir esses valores de etapa dentro do seu crontab.
Todos os asteriscos no campo de tempo dos seus crontabs significam que o arquivo script.sh será executado a cada minuto. Você não deve tentar isso em servidores; caso contrário, você pode interromper o sistema muito rapidamente.

20. Programe um trabalho Cron para ser executado a cada 15 minutos no domingo e segunda-feira

```
*/15 * * * Sun,Mon
/scripts/script.sh
```

Este comando diz ao cron para executar o trabalho especificado a cada 15 minutos durante domingo e segunda-feira.

21. Programe um trabalho Cron para ser executado nos meses especificados

```
* * * jan,may,aug *
/script/script.sh
```

O crontab acima fará com que o cron execute o script fornecido a cada minuto em janeiro, maio e agosto. Assim como nas semanas, o mesmo comando pode ser gravado usando valores numéricos apenas como mostrado abaixo.

22. Programe um trabalho Cron para ser executado em 15 de janeiro às 20:00

```
0 20 15 1 *
/script/script.sh
```

O verdadeiro poder do crontab é que ele permite que os administradores de sistemas definam períodos de tempo muito robustos. O comando acima executará o arquivo script.sh todos os 15 de janeiro às 20:00 em ponto.

23. Programe um trabalho Cron para ser executado a cada segundo mês

```
0 0 15 */2 *
/script/script.sh
```

A entrada crontab acima informa ao cron para chamar o arquivo script.sh a cada 15 dias do mês, a cada segundo mês do ano.

24. Programe um trabalho Cron para ser executado no primeiro domingo de cada mês

```
0 2 * * sun [ $(date
+%d) -le 07 ] &&
/script/script.sh
```

Será executado no primeiro domingo de cada mês usando os valores de período crontab. Podemos aproveitar a seção condicional da parte do comando para conseguir isso.

25. Programe um trabalho Cron para ser executado a cada três horas

```
0 */3 * * *
/script/script.sh
```

A entrada do crontab acima chama o trabalho do cron a cada intervalo de três horas.

26. Programe um trabalho Cron para executar duas vezes todos os sábados e segundas-feiras

```
0 8,20 * * 6,1
/scripts/script.sh
```

A entrada acima fará com que o cron execute um trabalho duas vezes todos os sábados e segundas-feiras.

27. Programe um trabalho Cron para ser executado a cada 30 segundos

```
* * * * *
/scripts/script.sh
* * * * * sleep 30;
/scripts/script.sh
```

Não é possível especificar um trabalho cron para ser executado a cada 30 segundos ou mais usando o parâmetro de campo de tempo do crontab. No entanto, ainda podemos fazer isso usando as entradas acima.

28. Agende vários trabalhos em uma única entrada do Crontab

```
0 8 * * *
/scripts/script.sh;
/scripts/scrit2.sh
```

Especifique mais de um trabalho em uma única entrada. Utilize o delimitador (;). Esse comando do crontab chama dois scripts às 8 da manhã todos os dias.

29. Programar trabalhos anuais do Cron

```
@yearly
/scripts/script.sh
```

O Crontab permite aos usuários agendar tarefas cron anuais. Ele executa esses trabalhos no primeiro minuto de cada ano.

30. Agende trabalhos mensais de Cron

```
@monthly
/scripts/system-
upgrade.sh
```

Também é possível especificar tarefas cron mensais e semanais usando formulários curtos.

31. Agende trabalhos semanais de Cron

```
@weekly
/scripts/system-
cleanup.sh
```

O Crontab permite que os usuários especifiquem trabalhos semanais facilmente usando o identificador @weekly.

32. Agende trabalhos diários do Cron

```
@daily
/scripts/script.sh
```

Cron também permite que os usuários usem o formato abreviado @daily para especificar tarefas cron diárias. Eles são úteis para a manutenção diária do seu sistema.

33. Agendar trabalhos de hora em hora

```
@hourly
/scripts/script.sh
```

O identificador @hourly pode ser usado para especificar tarefas cron que precisam ser executadas a cada hora.

34. Agendar um trabalho Cron na reinicialização do sistema

```
@reboot
/scripts/script.sh
```

O crontab permite que os administradores especifiquem tarefas cron que precisam ser executadas na reinicialização do sistema. Esses trabalhos podem variar da alteração de variáveis de caminho ao carregamento automático de arquivos de configuração personalizados.

35. Enviar resultados Cron para uma conta de email especificada

```
# crontab -l
MAIL=bob@admin.com
0 2 * * *
/script/backup.sh
```

Por padrão, o cron envia os relatórios dos trabalhos agendados agendados para o email do usuário que agendou o trabalho. Você pode redirecionar isso alterando o valor da variável de email, conforme mostrado no exemplo. Após a execução da tarefa cron backup.sh, o cron enviará os relatórios para o endereço de email bob@admin.com.

Executando comandos do Crontab como Root

No Linux, muitas tarefas requerem privilégios adicionais, como o sudo. No entanto, para executar comandos sudo a partir do crontab de um usuário padrão, os usuários precisam armazenar sua senha em um arquivo de texto sem formatação em algum lugar do sistema. Não é uma boa prática, e esses comandos devem ser agendados a partir do crontab do usuário root. O crontab do usuário root consiste em mais uma entrada entre o campo de hora e a seção de comando. É usado para especificar o usuário para quem executar os trabalhos.

36. Limpar todas as tentativas do Faillog às 01:00 Todos os dias

```
0 1 * * * root echo " "
> /var/log/faillog
```

O comando acima limpará todas as tentativas de login com falha no seu sistema à 1:00 da manhã todos os dias.

37. Salve todos os logs do sistema às 2 da manhã a cada 10 dias

```
0 2 */10 * * echo " ">
/var/log/syslog
```

O arquivo de log do sistema fornece informações úteis sobre nossa máquina Linux e é crucial para muitos administradores de sistemas. Adicione o comando crontab abaixo no seu crontab para salvar todos os logs do sistema às 2 da manhã a cada dez dias.

38. Verifique e baixe novos pacotes do sistema

```
0 12 1 * * root apt-get update
```

A entrada do comando crontab acima verifica novos pacotes de sistema às 12h, todos os primeiros dias do mês.

39. Programe atualizações do sistema como trabalhos Cron

```
0 12 1 * * root apt-get -y upgrade
```

A flag -y é necessária; caso contrário, o processo ficará parado aguardando sua aceitação manual.

40. Atualize a lista de pacotes e atualize o sistema

```
0 12 1 * * root apt-get update && apt-get -y upgrade
```

Você pode combinar os dois comandos acima para atualizar seus pacotes e atualizá-los para versões mais recentes usando a entrada crontab mencionada acima.

41. Agendar um trabalho Cron para remover dependências desnecessárias

```
0 1 1 * * root apt-get -y autoremove
```

Você pode usar o cron para automatizar esse processo para você. O crontab detecta e remove automaticamente todas as dependências que não são mais necessárias pelo seu sistema.

42. Programe um trabalho Cron para limpar o repositório local

```
0 2 1 * * root apt-get clean
```

Você pode usar o cron para limpar automaticamente o repositório local de arquivos de pacotes recuperados. Tudo o que você precisa fazer é adicionar o comando acima no seu crontab.

43. Agendar um trabalho Cron para limpeza de caches

```
0 3 1 * * root sync; echo 3 > /proc/sys/vm/drop_caches
```

Os caches são usados para fornecer acesso rápido aos serviços. Mas, eles podem ficar enormes e precisam de manutenção periodicamente. Essa entrada do crontab mostra como agendar um trabalho cron para limpar os caches PageCache, dentries e inodes.

44. Instale o Crontab personalizado para o seu usuário

```
$ crontab -a filename
```

Além do crontab padrão, os usuários podem instalar seu arquivo crontab personalizado. Este comando instala o documento "nome do arquivo" como seu crontab. Em muitos sistemas, o sinalizador -a não é necessário.

45. Faça backup de todos os trabalhos Cron no arquivo de texto sem formatação

```
$ crontab -l > cron-backup.txt
```

Geralmente, os administradores de sistema desejam armazenar um backup de suas entradas do crontab para referência futura. Isto pode ser feito de várias maneiras. O comando abaixo mostra como manter um backup de todos os trabalhos cron em um arquivo de texto sem formatação chamado cron-backup.txt.

46. Restaurar trabalhos Cron do arquivo de backup

```
$ crontab cron-backup.txt
```

Caso você tenha excluído o crontab atual, você pode restaurá-lo usando o arquivo de backup criado usando o comando acima. A sintaxe deste comando é mostrada abaixo.

47. Alterar o endereço de email para relatórios Cron

```
MAILTO=root@example.com
```

Você pode alterar o endereço de e-mail no qual o cron envia nossos relatórios específicos da tarefa adicionando a variável MAILTO dentro do seu crontab. Isso é demonstrado abaixo.

48. Alterar o valor da variável de caminho

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
```

Você pode alterar ou adicionar o valor das variáveis do caminho diretamente de dentro do seu crontab usando a variável PATH. Isso é mostrado abaixo.

49. Verifique o manual do Crontab

```
$ man crontab
```

O comando acima imprime a página de manual do comando crontab. Se você quiser aprender o crontab em detalhes ou procurar uma solução rápida para algum problema, verifique a página de manual definitivamente deve ser sua primeira prioridade.

50. Verifique o Manual do Cron

```
$ man cron
```

A página de manual do cron fornece informações específicas do cron. É uma ferramenta útil para usuários que desejam dominar cron com eficiência. Você pode consultar a página de manual do cron simplesmente usando o comando abaixo.