Programación de tareas en Linux

Panorama general

Este artículo discute algunas herramientas usadas en un sistema Linux para programar tareas para que se ejecuten automáticamente en intervalos de tiempo específicos o en un momento dado en el futuro. Este manual no cubrirá estos comandos en profundidad; es sólo una breve introducción al uso de estos comandos. Consulte el CÓMO individual de cada comando para obtener una visión más detallada de todas las opciones y configuraciones relevantes.

Algunos dæmons de programación de tareas utilizados en Linux/UNIX son:

- `at` – programar tareas puntuales para el futuro
- `cron` – el programador periódico más utilizado
- `anacron` – cron anacrónico; un programador periódico que no depende de que el sistema se deje en funcionamiento las 24 horas del día, los 7 días de la semana.

Uso de at

El comando ***at*** permite al usuario ejecutar comandos o scripts a una hora (requerido) y fecha (opcional) específicas. Los comandos se pueden introducir a través de una entrada estándar, redirección o archivo.

```
darkstar:~% at
```

at interactivo

Usar el comando ***at*** con entrada estándar (teclado) es un poco más complicado que escribir una línea en el prompt. El comando utiliza una “sub-shell” interna para recopilar la información requerida. Una vez que se haya completado la entrada de la información del comando, Ctrl+D (EOT) significará que se ha completado la entrada. El indicador ***-m*** especifica que se enviará un mensaje de correo al usuario cuando finalice el trabajo, independientemente de si se ha creado alguna salida.

```
darkstar:~% at 12:01 -m
warning: commands will be executed using (in order) a) $SHELL b) login shell
c) /bin/sh
at> ./my_script.sh
at> <EOT>
job 4 at 2015-06-22 12:01
darkstar:~%
```

at dirigido por archivos

Los comandos también pueden estar contenidos dentro de un archivo y ser ejecutados por ***at***:

```
darkstar:~% at 12:32 -m -f /usr/local/bin/my_script.sh
warning: commands will be executed using (in order) a) $SHELL b) login shell
c) /bin/sh
job 8 at 2015-06-22 12:10
```

El indicador **-m** enviará un correo electrónico al usuario después de completar el comando; el indicador **-f** especifica que el comando leerá el trabajo desde un archivo, no desde una entrada estándar. Después de escribir el comando (y se muestre la advertencia correspondiente), se muestra el número de trabajo ***at*** [1)].

Programación interna de at

Los números de trabajo proporcionados después de escribir un comando, o cuando se lee un archivo, permiten al usuario saber qué trabajo interno se ejecutará en orden secuencial. Si un usuario desea borrar una tarea específica, todo lo que necesita saber es el número de trabajo interno. Para eliminar el trabajo, se utiliza el comando ***atrm*** (**at remove**):

```
darkstar:~%  at -l
7      2015-06-22 12:10 p tux
8      2015-06-22 12:15 p root
```

El comando ***atq*** (**at queue**) es el mismo que ***at -l***:

```
darkstar:~%  atq
7      2015-06-22 12:10 p tux
8      2015-06-22 12:15 p root
```

Para quitar el trabajo del usuario, utilice ***atrm*** con el número de trabajo:

```
darkstar:~%  atrm 7
```

Uso de cron

cron es un demonio que ejecuta tareas en segundo plano en momentos específicos. Por ejemplo, si desea automatizar las descargas de parches en un día específico (lunes), fecha (2 de julio) u hora (1300), ***cron*** le permitirá configurar esto de varias maneras. La flexibilidad inherente en ***cron*** puede permitir a los administradores y usuarios avanzados automatizar tareas repetitivas, como la creación de copias de seguridad y el mantenimiento del sistema.

cron se configura normalmente utilizando un archivo *crontab*. El siguiente comando abrirá el archivo *crontab* de su cuenta de usuario:

```
darkstar:~% crontab -e
```

Para editar el nivel de sistema *crontab*, primero inicie sesión en la cuenta root:

```
darkstar:~# crontab -e
```

Si su sistema tiene ***sudo*** instalado, ingrese:

```
darkstar:~% sudo crontab -e
```

La sintaxis del archivo *crontab* es:

```
 # * * * * *  Comando a ejecutar
 # | | | | |
 # | | | | |
 # | | | | └──── día de la semana (0 - 6) (Dom(0) /Lun (1)/Mar (2)/Mie
(3)/Jue (4)/Vie (5)/Sab (6))
 # | | | └─────────── mes (1 - 12)
 # | | └──────────────── día del mes (1 - 31)
 # | └─────────────────────── hora (0 - 23)
 # └──────────────────────────── minuto (0 - 59)
```

Usando un asterisco en cualquier ubicación de marcador de posición, coincidirá con cualquier valor. Por ejemplo, lo siguiente se ejecutará *example_script.sh* al mediodía (1200) todos los días durante los primeros tres meses del año:

```
#For more information see the manual pages of crontab(5) and cron(8)
#
# min hr day month weekday command
#
#

0 11 * 1-3 * /home/user/example_script.sh
```

Uso de anacron

anacron no se inastala en Slackware de manera predeterminada.[2)]

anacron es único respecto a ***cron*** en el sentido de que no espera que el sistema operativo se ejecute continuamente como un servidor 24×7. Si el tiempo de ejecución pasa mientras el sistema está apagado, ***anacron*** ejecuta el comando automáticamente cuando la máquina se enciende de nuevo. Lo contrario **no** es cierto para ***cron***: si el equipo está apagado a la hora de la ejecución programada, ***cron*** no ejecutará el trabajo. Otra diferencia clave entre ***anacron*** y ***cron*** es la “granularidad” cronológica mínima - ***anacron*** sólo puede ejecutar trabajos cada *día*, frente a la capacidad de ***cron*** de ejecutarlos al *minuto*/. Finalmente, ***anacron*** puede sólo ser usado por root, mientras que ***cron*** puede ser usado por root y usuarios normales.

Fuentes

- Originalmente escrito por vharishankar
- Contribuciones de mfillpot, tdrssb
- Ejemplo crontab modificado de en.wikipedia.org/wiki/cron
- Traducido por — *Pedro Herrero García* *2019/02/10 09:43 (UTC)*

howtos, task scheduling, needs attention, author vharishankar, author mfillpot

1)

A diferencia de un ID de proceso (PID) conocido por el sistema operativo

2)

Ver Slackbuilds.org para más información de ***anacron*** en Slackware