Tema 6

Automatización de tareas

Administración de Sistemas Operativos

Índice

[Esquema 3](#_Toc170114879)

[Material de estudio 4](#_Toc170114880)

[1.1. Introducción y objetivos 4](#_Toc170114881)

[1.2. Introducción a la automatización de tareas en Linux 5](#_Toc170114882)

[1.3. Planificación y automatización de tareas 5](#_Toc170114883)

[1.4. Planificación de tareas diferidas en Linux 7](#_Toc170114884)

[1.5. Planificación de tareas periódicas en Linux 9](#_Toc170114885)

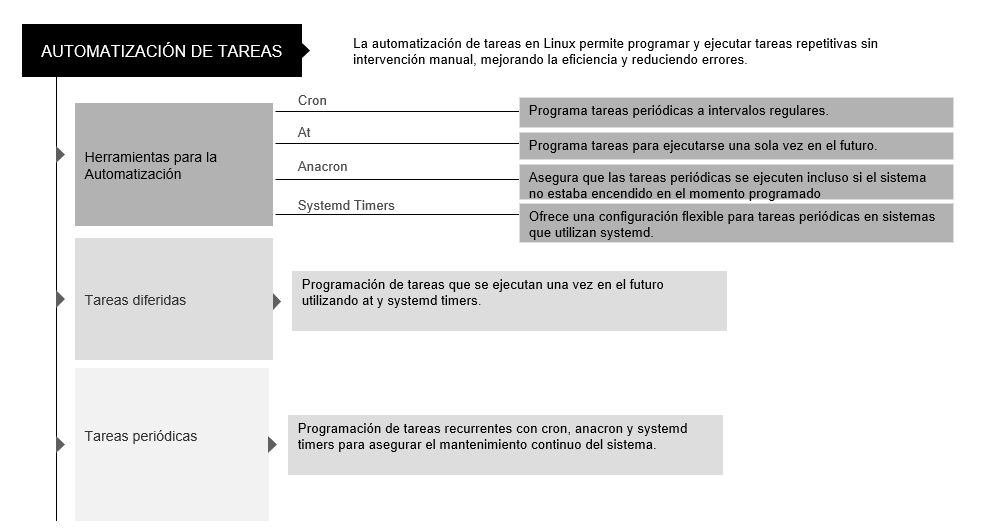
[1.6. Referencias bibliográficas 12](#_Toc170114886)

[A fondo 13](#_Toc170114887)

[Entrenamientos 14](#_Toc170114888)

[Test 18](#_Toc170114889)

Esquema



Material de estudio

6.1. Introducción y objetivos

La automatización de tareas en Linux es un aspecto fundamental para la administración eficiente de sistemas. Permite a los administradores de sistemas y usuarios programar y ejecutar tareas repetitivas sin intervención manual, ahorrando tiempo y reduciendo el riesgo de errores humanos. En un entorno Linux, la automatización de tareas se puede lograr mediante diversas herramientas y técnicas, como la planificación de tareas con cron y at, el uso de scripts de shell, y la implementación de tareas periódicas para el mantenimiento del sistema.

Este tema explora cómo configurar y utilizar estas herramientas para automatizar una amplia gama de tareas en un sistema GNU/Linux. Desde la creación de scripts simples para automatizar comandos hasta la programación de tareas complejas que se ejecutan periódicamente, los estudiantes aprenderán a aprovechar al máximo las capacidades de automatización de Linux.

Los objetivos que se pretenden conseguir al final de este tema son:

* **Comprender la importancia de la automatización de tareas:** Aprender cómo la automatización puede mejorar la eficiencia y confiabilidad de las operaciones del sistema.
* **Identificar las tareas que pueden ser automatizadas:**  Explorar ejemplos de tareas comunes que se benefician de la automatización, como copias de seguridad, actualizaciones de software y mantenimiento de logs.
* **Uso de cron y at**: Aprender a configurar y gestionar cron jobs para la ejecución periódica de tareas. Se cubrirá la sintaxis de crontab y cómo programar tareas diarias, semanales y mensuales. Así mismo, se enseñará cómo utilizar el comando at para programar la ejecución de tareas una sola vez en el futuro.
* **Configurar tareas periódicas utilizando cron y anacron**: Aprender a programar tareas de mantenimiento que se ejecuten automáticamente en intervalos regulares.

6.2. Introducción a la automatización de tareas en Linux

La automatización de tareas en Linux es un aspecto fundamental para la administración eficiente y efectiva de sistemas. Consiste en el uso de herramientas y scripts que permiten programar y ejecutar tareas repetitivas sin intervención manual. La automatización no solo ahorra tiempo y esfuerzo, sino que también minimiza el riesgo de errores humanos, mejora la eficiencia operativa y garantiza que las tareas críticas se realicen de manera oportuna. En un entorno Linux, la automatización es esencial para el mantenimiento y la administración proactiva de servidores y sistemas.

6.3. Planificación y automatización de tareas

En el ecosistema GNU/Linux, existen varias herramientas robustas que permiten la planificación y automatización de tareas. Entre las más comunes y útiles se encuentran cron, at, anacron, y systemd timers. Estas herramientas facilitan la programación de tareas de mantenimiento, monitoreo y otras funciones esenciales del sistema.

Cron

Cron es una de las herramientas más utilizadas para la planificación y automatización de tareas en Linux. Permite programar la ejecución de comandos o scripts en intervalos regulares como minutos, horas, días, semanas o meses. Los trabajos de cron se definen en archivos llamados crontabs. Cada usuario tiene su propio crontab, además del crontab del sistema.

En este archivo de configuración llamado crontab (abreviatura de "cron table") es donde se especifican las tareas a realizar y los tiempos en los que deben ejecutarse. Cada usuario en el sistema puede tener su propio crontab y también hay un crontab del sistema que puede ser gestionado por el administrador.

La sintaxis básica de una entrada en un crontab es:

\* \* \* \* \* comando

Los cinco asteriscos representan:

* Minuto (0-59)
* Hora (0-23)
* Día del mes (1-31)
* Mes (1-12)
* Día de la semana (0-7, donde 0 y 7 representan el domingo)

Por ejemplo, la siguiente entrada en un crontab ejecuta un comando a las 2:30 AM todos los días:

30 2 \* \* \* /ruta/a/tu/comando

Para editar el crontab del usuario actual, se utiliza el comando:

crontab -e

Para listar las tareas programadas en el crontab del usuario actual:

crontanb -l

Para eliminar el crontab del usuario actual:

crontab -r

Y para editar el crontab de otro usuario (requiere permisos de superusuario):

sudo crontab -u nombre\_usuario -e

Cron también soporta algunos atajos que simplifican la programación de tareas comunes:

* @reboot: Ejecuta el comando una vez al iniciar el sistema.
* @yearly o @annually: Ejecuta el comando una vez al año (0 0 1 1 \*).
* @monthly: Ejecuta el comando una vez al mes (0 0 1 \* \*).
* @weekly: Ejecuta el comando una vez a la semana (0 0 \* \* 0).
* @daily o @midnight: Ejecuta el comando una vez al día (0 0 \* \* \*).
* @hourly: Ejecuta el comando una vez a la hora (0 \* \* \* \*).

Por ejemplo, para ejecutar un script al iniciar el sistema, se puede usar:

@reboot /ruta/a/tu/script.sh

6.4. Planificación de tareas diferidas en Linux

Las tareas diferidas son aquellas que se programan para ejecutarse en un momento específico en el futuro, en lugar de repetirse a intervalos regulares. Herramientas como at son ideales para gestionar este tipo de tareas.

at

at permite la programación de tareas que deben ejecutarse una sola vez en el futuro. A diferencia de cron, at se utiliza para tareas que deben ejecutarse solo una vez. Para programar un comando con at, se usa la siguiente sintaxis:

echo “comando” | at time

Donde "time" puede ser una hora específica como "14:00" o expresiones relativas como "now + 2 hours". Por ejemplo, para ejecutar un script mañana a las 2 PM, se puede usar:

echo “/ruta/a/tu/script.sh” | at 2pm tomorrow

Ver y administrar tareas programadas

El **comando atq** muestra la cola de trabajos de at, indicando el número de trabajo, la fecha y hora programadas, y el nombre del usuario.

atq

La salida puede ser algo como esto:



Figura 1. Resultado de ejecución del comando atq. Fuente: elaboración propia.

Para eliminar una tarea programada, se usa el **comando atrm** seguido del número de trabajo:

atrm 1

Archivos de configuración

at y atq dependen de los permisos y configuraciones especificados en los archivos /etc/at.allow y /etc/at.deny:

* /etc/at.allow: Si existe, solo los usuarios listados en este archivo pueden usar at.
* /etc/at.deny: Si at.allow no existe, los usuarios listados en at.deny no pueden usar at. Si ninguno de estos archivos existe, solo los superusuarios pueden usar at.

6.5. Planificación de tareas periódicas en Linux

Las tareas periódicas son aquellas que se ejecutan a intervalos regulares (diarios, semanales, mensuales, etc.) para realizar tareas de mantenimiento, monitoreo, copias de seguridad y otras funciones necesarias en un sistema Linux. La automatización de estas tareas asegura que el sistema se mantenga en óptimas condiciones sin necesidad de intervención manual constante.

En Linux, las herramientas más comunes para programar tareas periódicas son cron, anacron, y systemd timers.

Anacron para tareas periódicas

**Anacron** complementa a cron al asegurar que las tareas periódicas se ejecuten incluso si el sistema no estaba encendido en el momento programado. Esto es especialmente útil para laptops y otros dispositivos que no están siempre en funcionamiento.

Anacron utiliza un archivo de configuración ubicado en /etc/anacrontab. La sintaxis de una entrada típica en un archivo de Anacron es:

periodo retraso identificador commando

* periodo: Frecuencia con la que debe ejecutarse la tarea (en días).
* retraso: Retraso en minutos después del arranque del sistema.
* identificador: Un nombre único para la tarea.
* comando: El comando o script a ejecutar.

Para ejecutar un script diariamente, se puede configurar el archivo /etc/anacrontab como sigue:



Figura 2. Configuración del archivo anacrontab. Fuente: elaboración propia.

Esto ejecutará el script backup.sh una vez al día, con un retraso de 5 minutos después del arranque del sistema.

Para ejecutar un script semanalmente, se puede usar:



Figura 3. Otra configuración del archivo anacrontab. Fuente: elaboración propia.

Esto ejecutará el script clean.sh una vez a la semana, con un retraso de 10 minutos después del arranque del sistema.

Systemd Timers para tareas periódicas

Systemd timers son una alternativa moderna a cron y anacron, utilizados en sistemas que emplean systemd como su sistema de init. Systemd timers ofrecen una configuración más flexible y detallada para tareas periódicas.

Para configurar una tarea periódica con systemd timers, se deben crear dos archivos: una unidad de servicio y una unidad de temporizador.

**Crear un Archivo de Unidad de Servicio:**

Este archivo define la tarea a realizar. Por ejemplo, para ejecutar un script diario, se puede crear un archivo /etc/systemd/system/mi-servicio-diario.service:

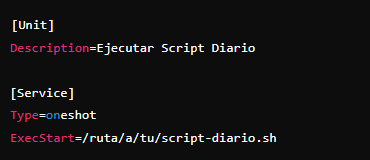


Figura 4. Configuración del archivo anacrontab. Fuente: elaboración propia.

**Crear un Archivo de Unidad de Temporizador:**

Este archivo define cuándo debe ejecutarse la tarea. Se puede crear un archivo /etc/systemd/system/mi-servicio-diario.timer:

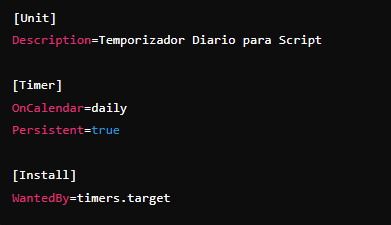


Figura 5. Configuración del archivo anacrontab. Fuente: elaboración propia.

**Habilitar y activar el temporizador:**

Para habilitar y activar el temporizador, se utilizan los siguientes comandos:

sudo systemctl enable mi-servicio-diario.timer

sudo systemctl start mi-servicio-diario. timer

Comparación entre Cron, Anacron y Systemd Timers

* **Cron:** Ideal para tareas recurrentes en sistemas que están siempre encendidos. No asegura la ejecución de tareas si el sistema no está encendido en el momento programado.
* **Anacron:** Complementa a cron para asegurar la ejecución de tareas periódicas en sistemas que no están siempre encendidos. Es útil para laptops y desktops.
* **Systemd Timers:** Ofrecen una configuración más flexible y detallada. Son ideales para sistemas modernos que utilizan systemd, y combinan lo mejor de cron y anacron con capacidades adicionales.

6.6. Referencias bibliográficas

Caballero, A. J., Administración de Sistemas Operativos. Ra-Ma, 2017.

García Carrasco, J. (2016). Administración de Sistemas Operativos. Editorial Síntesis.

Pons, N., Linux. Principios básicos de uso del sistema. Ediciones ENI, 2023.

Rohaut, S., Linux. Dominar la administración del sistema. Ediciones ENI, 2021.

A fondo

Cómo usar Crontab en Ubuntu para automatizar tareas.

El Rincón del Hacker. (2022). *Cómo usar Crontab en Ubuntu para automatizar tareas.* [[Vídeo]. Youtube.](https://www.redalyc.org/articulo.oa?id=337630234004) https://www.youtube.com/watch?v=d2Q0NiyVO5M&t=265s

En este vídeo se muestra el uso de la herramienta Crontab para automatizar la ejecución de comandos durante un determinado periodo de tiempo.

Understand Systemd Timer Units and how to create them

theurbanpenguin. (2023). Understand Systemd Timer Units and how to create them [Vídeo]. Youtube. https://www.youtube.com/watch?v=c20saf4q4pw

En este vídeo se muestra como enumerar unidades de temporizador usando systemctl list-timers y a crear nuevas unidades de temporizador con systemctl edit

.

Entrenamientos

Entrenamiento 1

* Programar una Tarea Diaria con Cron: Aprender a programar una tarea diaria utilizando cron.
* Paso a paso:
  + Abre una terminal y crea un script simple que imprima la fecha y hora actuales. Guarda el script como date.sh.
  + Abre el crontab del usuario actual para editarlo.
  + Añade una línea para ejecutar el script date.sh todos los días a las 8 AM.
  + Guarda y cierra el editor. Verifica si la tarea está programada correctamente listando el crontab
* Solución

echo "#!/bin/bash" > /ruta/a/tu/date.sh

echo "date" >> /ruta/a/tu/date.sh

chmod +x /ruta/a/tu/date.sh

crontab -e

0 8 \* \* \* /ruta/a/tu/date.sh

crontab -l

Entrenamiento 2

* Programar una Tarea con At. Aprende a programar una tarea que se ejecute una sola vez en el futuro utilizando at.
* Paso a paso:
  + Crea un script simple que escriba "Hola, Mundo!" en un archivo de texto. Guarda el script como hello.sh.
  + Programa la ejecución del script hello.sh para que se ejecute en 2 horas.
  + Verifica que la tarea esté programada.
* Solución

echo "#!/bin/bash" > /ruta/a/tu/hello.sh

echo 'echo "Hola, Mundo!" > /ruta/a/tu/hello.txt' >> /ruta/a/tu/hello.sh

chmod +x /ruta/a/tu/hello.sh

echo "/ruta/a/tu/hello.sh" | at now + 2 hours

atq

Entrenamiento 3

* Crear una Tarea Periódica con Systemd Timers. Aprende a programar una tarea periódica utilizando systemd timers.
* Paso a paso:
  + Crea un script simple que limpie los archivos temporales en un directorio. Guarda el script como clean\_temp.sh.
  + Crea un archivo de unidad de servicio en /etc/systemd/system/clean\_temp.service.
  + Crea un archivo de unidad de temporizador en /etc/systemd/system/clean\_temp.timer.
  + Habilita y activa el temporizador.
* Solución

echo "#!/bin/bash" > /ruta/a/tu/clean\_temp.sh

echo "rm -rf /ruta/a/temp/\*" >> /ruta/a/tu/clean\_temp.sh

chmod +x /ruta/a/tu/clean\_temp.sh

[Unit]

Description=Limpiar Archivos Temporales

[Service]

Type=oneshot

ExecStart=/ruta/a/tu/clean\_temp.sh

[Unit]

Description=Temporizador para Limpiar Archivos Temporales

[Timer]

OnCalendar=daily

Persistent=true

[Install]

WantedBy=timers.target

sudo systemctl enable clean\_temp.timer

sudo systemctl start clean\_temp.timer

systemctl status clean\_temp.timer

Entrenamiento 4

* Programar una Tarea Semanal con Anacron. Aprende a programar una tarea semanal utilizando anacron.
* Paso a paso:
  + Crea un script simple que realice una copia de seguridad de un directorio. Guarda el script como backup.sh.
  + Edita el archivo /etc/anacrontab para añadir una línea que programe la tarea semanal.
  + Añade la siguiente línea: 7 10 backup.weekly /ruta/a/tu/backup.sh. Puedes verificar que la tarea se haya programado correctamente revisando el archivo /etc/anacrontab
* Solución

echo "#!/bin/bash" > /ruta/a/tu/backup.sh

echo "tar -czf /ruta/a/backup.tgz /ruta/a/tu/directorio" >> /ruta/a/tu/backup.sh

chmod +x /ruta/a/tu/backup.sh

sudo nano /etc/anacrontab

7 10 backup.weekly /ruta/a/tu/backup.sh

Entrenamiento 5

* Listar y Eliminar Tareas Programadas con At. Aprender a listar y eliminar tareas programadas con at.
* Paso a paso:
  + Programa una tarea simple que imprima "Tarea programada con at" en un archivo de texto en 1 hora.
  + Lista las tareas programadas con at para el usuario actual.
  + Elimina la tarea programada.
* Solución

echo 'echo "Tarea programada con at" > /ruta/a/tu/at\_task.txt' | at now + 1 hour

atq

atrm [job\_number]

Test

1. ¿Cuál es la principal función de cron en Linux?

A) Ejecutar tareas una sola vez en el futuro

\_B) Ejecutar tareas periódicas a intervalos regulares

C) Monitorear el uso de recursos del sistema

D) Configurar la red del sistema

cron se utiliza para programar la ejecución de comandos o scripts en intervalos regulares como minutos, horas, días, semanas o meses.

1. ¿Cuál es la utilidad de at en Linux?

A) Ejecutar tareas periódicas

\_B) Ejecutar tareas una sola vez en el futuro

C) Monitorear procesos

D) Editar archivos de configuración del sistema

at permite programar la ejecución de tareas una sola vez en el futuro en un momento específico.

1. ¿Cuál de los siguientes comandos se utiliza para editar el crontab del usuario actual?

A) crontab -l

B) crontab -r

\_C) crontab -e

D) crontab -d

crontab -e abre el editor de crontab del usuario actual para agregar o modificar tareas programadas

1. ¿Qué archivo se utiliza para definir tareas periódicas con anacron?

A) /etc/crontab

\_B) /etc/anacrontab

C) /etc/at.allow

D) /etc/sysctl.conf

/etc/anacrontab es el archivo de configuración utilizado por anacron para definir tareas periódicas.

1. ¿Qué herramienta en Linux se utiliza para asegurar que las tareas periódicas se ejecuten incluso si el sistema no estaba encendido en el momento programado?

A) cron

B) at

\_C) anacron

D) systemd timers

anacron asegura que las tareas periódicas se ejecuten una vez que el sistema se encienda, incluso si no estaba encendido en el momento programado.

1. ¿Cuál es la sintaxis correcta para programar una tarea con cron que se ejecute todos los días a la medianoche?

\_A) 0 0 \* \* \* /ruta/a/tu/comando

B) \* \* 0 0 \* /ruta/a/tu/comando

C) 0 \* \* \* \* /ruta/a/tu/comando

D) \* 0 \* \* \* /ruta/a/tu/comando

0 0 \* \* \* /ruta/a/tu/comando programa la tarea para que se ejecute a la medianoche todos los días.

1. ¿Qué comando se utiliza para listar las tareas programadas en el crontab del usuario actual?

\_A) crontab -l

B) crontab -e

C) crontab -r

D) crontab -q

crontab -l lista las tareas programadas en el crontab del usuario actual.

1. ¿Cuál de las siguientes herramientas se puede utilizar para programar tareas periódicas en sistemas que emplean systemd como su sistema de init?

A) cron

B) at

C) anacron

\_D) systemd timers

systemd timers son utilizados en sistemas que emplean systemd para programar tareas periódicas de manera flexible y detallada.

1. ¿Cómo se elimina una tarea programada con at?

\_A) atrm [job\_number]

B) at -d [job\_number]

C) at -e [job\_number]

D) atr -r [job\_number]

atrm [job\_number] elimina una tarea programada con at identificada por su número de trabajo.

1. ¿Qué archivo se edita para hacer que los cambios en los parámetros del kernel sean permanentes?

A) /etc/fstab

\_B) /etc/sysctl.conf

C) /etc/crontab

D) /etc/rc.local

/etc/sysctl.conf se utiliza para definir parámetros del kernel que deben aplicarse permanentemente al arrancar el sistema.

1. ¿Qué comando se utiliza para programar una tarea con at que se ejecute en 2 horas?

\_A) echo "/ruta/a/tu/comando" | at now + 2 hours

B) echo "/ruta/a/tu/comando" | at 2 hours

C) echo "/ruta/a/tu/comando" | at 2h

D) echo "/ruta/a/tu/comando" | at 120m

echo "/ruta/a/tu/comando" | at now + 2 hours programa la tarea para que se ejecute en 2 horas a partir del momento actual.

1. ¿Cuál es el comando para iniciar un servicio administrado por systemd?

\_A) systemctl start

B) sysctl start

C) service start

D) systemd start

systemctl start [nombre\_del\_servicio] se utiliza para iniciar un servicio administrado por systemd.

1. ¿Cuál es la sintaxis correcta para programar una tarea con cron que se ejecute todos los lunes a las 3 AM?

\_A) 0 3 \* \* 1 /ruta/a/tu/comando

B) 0 3 \* 1 \* /ruta/a/tu/comando

C) 0 3 1 \* \* /ruta/a/tu/comando

D) 0 3 \* \* Mon /ruta/a/tu/comando

0 3 \* \* 1 /ruta/a/tu/comando programa la tarea para que se ejecute todos los lunes a las 3 AM.

1. ¿Qué comando se utiliza para listar los trabajos programados con at?

A) at -l

B) at -q

\_C) atq

D) atl

atq lista los trabajos programados con at para el usuario actual.

1. ¿Cuál de las siguientes opciones permite ejecutar un comando automáticamente al iniciar el sistema con cron?

\_A) @reboot

B) @startup

C) @boot

D) @init

@reboot se utiliza en cron para programar la ejecución de un comando automáticamente al iniciar el sistema.