



AT (automatización de tareas)

Explicación sobre el uso de `at` en Linux

El comando `at` en Linux se usa para programar la ejecución de tareas en un momento determinado. A diferencia de `cron`, que es usado para tareas recurrentes, `at` ejecuta un trabajo **una sola vez** en el futuro.

Principales Características

- Se pueden programar tareas con una fecha y hora específica o con un tiempo relativo.
- Las tareas pueden ingresarse desde un archivo o directamente desde la terminal.
- Se pueden listar (`atq`) o eliminar (`atrm`) tareas programadas.
- Se puede combinar con `batch` para ejecutar tareas cuando la carga del sistema sea baja.

Uso y Sintaxis del comando `at`

1. Programar un trabajo en una fecha y hora específica

```
at 11am May 20 -f lista_de_comandos.txt
```

Ejecutará los comandos en `lista_de_comandos.txt` el **20 de mayo a las 11:00 AM**.

2. Programar un trabajo con tiempo relativo

```
at now + 5 minutes
```

Ejecutará el comando dentro de **5 minutos**.

3. Ejecutar un comando dentro de 1 hora a partir de ahora

```
echo "touch /tmp/testfile" | at now + 1 hour
```

Crearé el archivo `/tmp/testfile` en **una hora**.

4. Ejecutar un conjunto de comandos desde la terminal

```
at now + 2 min
at> touch /tmp/lala.txt
at> touch /tmp/lala2.txt
at> <Ctrl+D> # Para finalizar la entrada
```

Se crearán los archivos `/tmp/lala.txt` y `/tmp/lala2.txt` en **dos minutos**.

5. Listar trabajos pendientes

```
atq
```

Muestra todos los trabajos en cola.

6. Eliminar un trabajo programado

```
atrm 4
```

Elimina el trabajo con ID `4`.

7. Ejecutar un trabajo solo cuando la carga del sistema sea baja

```
batch
at> updatedb
at> <Ctrl+D>
```

Se ejecutará el comando `updatedb` cuando la carga del sistema sea menor a **1.5**.

8. Ejecutar un trabajo en segundo plano y salir de la terminal

```
at -f myscript.sh now + 1 min
exit
```

El script `myscript.sh` se ejecutará en **1 minuto**, incluso si el usuario cierra la sesión.

Ejercicios Prácticos

Ejercicio 1: Crear un archivo en el futuro

Objetivo: Programar un comando que cree un archivo en `/tmp` dentro de 3 minutos.

💡 **Comando esperado:**

```
at now + 3 min
at> touch /tmp/fichero_prueba.txt
at> <Ctrl+D>
```

Verificación: Luego de 3 minutos, usa `ls /tmp/` para ver si el archivo fue creado.

Ejercicio 2: Enviar un mensaje al usuario en el futuro

Objetivo: Programar un mensaje para que aparezca en la terminal dentro de 5 minutos.

💡 **Comando esperado:**

```
echo "¡Recuerda tomar un descanso!" | at now + 5 minutes
```

Verificación: Espera 5 minutos y verifica que el mensaje apareció en la terminal.

Ejercicio 3: Listar trabajos pendientes y eliminarlos

1. Programa un trabajo que cree un archivo en 10 minutos

```
echo "touch /tmp/archivo_temporal.txt" | at now + 10 minutes
```

2. Lista los trabajos pendientes

```
atq
```

3. Elimina el trabajo antes de que se ejecute

Sustituye `<ID>` por el número de trabajo obtenido en `atq`.

```
atrm <ID>
```

Verificación: Usa `atq` después de eliminar el trabajo para confirmar que fue removido.

Ejercicio 4: Programar una actualización de paquetes cuando la carga sea baja

Objetivo: Usar `batch` para actualizar la lista de paquetes cuando el sistema no esté sobrecargado.

💡 **Comando esperado:**

```
batch
at> sudo apt update && sudo apt upgrade -y
at> <Ctrl+D>
```

Verificación: Usa `sudo apt list --upgradable` después de un tiempo para ver si la actualización se ejecutó.

Conclusión

- `at` es útil para tareas únicas programadas en el tiempo.
- Se pueden usar tiempos absolutos o relativos.
- Se puede combinar con `batch` para optimizar el rendimiento del sistema.
- `atq` y `atrm` ayudan a gestionar trabajos programados.

Para depurar y ver logs del comando `at`, puedes utilizar los siguientes métodos:

1. Revisar la salida del sistema (`journalctl` o `syslog`)

En sistemas basados en `systemd` (como Ubuntu y CentOS 7+):

```
journalctl -u atd --no-pager
```

```
# revisar los ultimos 10 minutos
sudo journalctl -u atd --since "10 minutes ago"
```

Esto muestra los registros del servicio `atd` (el demonio que ejecuta los trabajos de `at`).

En sistemas sin `systemd` (como CentOS 6, Debian antiguo):

```
cat /var/log/syslog | grep at
```

O en otros sistemas:

```
cat /var/log/messages | grep at
```

Esto filtra los eventos relacionados con `at` en los logs del sistema.

2. Ver la salida estándar y errores de un trabajo de `at`

Por defecto, `at` envía la salida estándar (`stdout`) y errores (`stderr`) por email al usuario. Para capturar estos logs en un archivo:

1. Redirigir la salida a un archivo al programar un trabajo

```
echo "ls /home > /tmp/at_output.log 2>&1" | at now + 1 minute
```

Esto guarda la salida en `/tmp/at_output.log`.

2. Verificar el contenido del log después de ejecutarse

```
cat /tmp/at_output.log
```

3. Revisar trabajos ejecutados recientemente

El sistema guarda los trabajos de `at` en `/var/spool/atjobs/`, pero se eliminan después de ejecutarse. Para ver los trabajos que aún no han sido ejecutados:

```
ls -l /var/spool/atjobs/
```

Si quieres ver los detalles de un trabajo en ejecución, puedes listar su contenido:

```
cat /var/spool/atjobs/<nombre_del_trabajo>
```

4. Revisar si el servicio **atd** está corriendo

Si los trabajos no se ejecutan, verifica que el demonio **atd** esté activo:

```
systemctl status atd
```

Si no está activo, puedes iniciarlo con:

```
sudo systemctl start atd
```

Y asegurarte de que inicie con el sistema:

```
sudo systemctl enable atd
```

5. Hacer pruebas interactivas

Si un trabajo no se ejecuta, intenta correrlo manualmente para ver errores:

```
sh -x /tmp/script_a_ejecutar.sh
```

Esto mostrará cada comando antes de ejecutarlo, ayudando a identificar fallos.


Resumen

 **journalctl -u atd** → Logs del servicio **atd**.

 **cat /var/log/syslog | grep at** → Eventos de **at** en logs del sistema.

 **ls -l /var/spool/atjobs/** → Ver trabajos programados.

 **Redirigir salida con** **> /tmp/logfile.log 2>&1** → Capturar logs de ejecución.

 **Verificar** **systemctl status atd** → Revisar si **atd** está corriendo.