

## Práctica 4.2. Programación Tareas Linux con cron y anacron



Índice

Ejercicio 1 ..... 3

Ejercicio 2..... 7

## Ejercicio 1

1. Planificación con el crontab de vuestro usuario (no root)
  - a) ¿Qué muestra `$apt-cache show cron`?

Esta orden muestra información resumida sobre el paquete cron que incluye:

- Descripción del paquete.
- Versión disponible.
- Dependencias necesarias.
- Tamaño instalado.
- Mantenedor del paquete.
- Prioridad y sección a la que pertenece.

Muestro información en mi sistema.

```
mauro@aso: ~$ apt-cache show cron
Package: cron
Architecture: amd64
Version: 3.0pl1-137ubuntu3
Multi-Arch: foreign
Priority: standard
Section: admin
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Javier Fernández-Sanguino Peña <jfs@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 255
Provides: cron-daemon
Pre-Depends: init-system-helpers (>= 1.54~)
Depends: libc6 (>= 2.34), libpam0g (>= 0.99-7.1), libselinux1 (>= 3.1-), debianutils (>= 1.7), sensible-utils, adduser, lsb-base (>= 3.0-10), libpam-runtime (>= 1.0.1-11)
Suggests: anacron (>= 2.0-1), logrotate, checksecurity, default-nta | mail-transport-agent
Filename: pool/main/c/cron/cron_3.0pl1-137ubuntu3_amd64.deb
Size: 73668
MD5sum: 9fe3b3b22ab0e99d2a7de51d6fa9
SHA1: e4b7daa572f5a40b88f21085f8e0b2f44db594f
SHA256: 5ddel799e4086cf8833b82855fe755d292c2ed1855364355daff4364ce62beab
SHA512: bcf115c3f1b07d9a5d05e89591bef61330476094bc9cc86c4ddca823a4f1d2baf674240be6923828d64895d11579b4211b32996f73aa46613449f46ab97b0
Homepage: https://ftp.lsc.org/lsc/cron/
Description-es: Demonto programador de procesos
El demonio cron es un proceso en segundo plano que ejecuta programas
particulares en momentos particulares (por ejemplo, cada minuto, día,
semana o mes), como se especifica en un crontab. De manera predeterminada,
los usuarios también pueden crear crontabs por ellos mismos, así los
procesos se ejecutan en su nombre.
.
La salida de los programas normalmente se envía al administrador del
sistema (o al usuario en cuestión); probablemente también debería instalar
un sistema de correo para poder recibir estos mensajes.
.
Este paquete cron no proporciona ninguna tarea de mantenimiento del
sistema. Las tareas básicas de mantenimiento periódico las proporcionan
otros paquetes, como «checksecurity».
Description-md5: 7384e614068d48b9ac2335cb05added3
Task: standard
mauro@aso: ~$
```

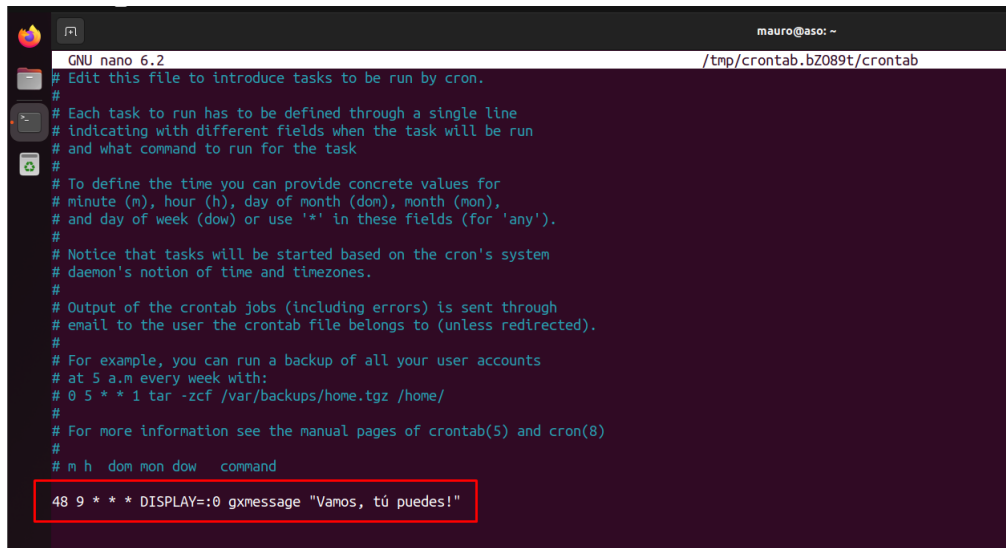
Ilustración 1. Salida de la orden `apt-cache show cron`

- b) Editar con la orden adecuada el archivo crontab de vuestro usuario para que a las 12:00 os muestre un mensaje: "¡Vamos, tú puedes!" y que aparezca en vuestro escritorio dándonos ánimos.

*Nota: Se puede usar "wall" o gxmessage (con entorno gráfico aparece el mensaje en una ventana flotante, pero hay que usar gxmessage junto con export DISPLAY)*

*Ayuda: ver ANEXO*

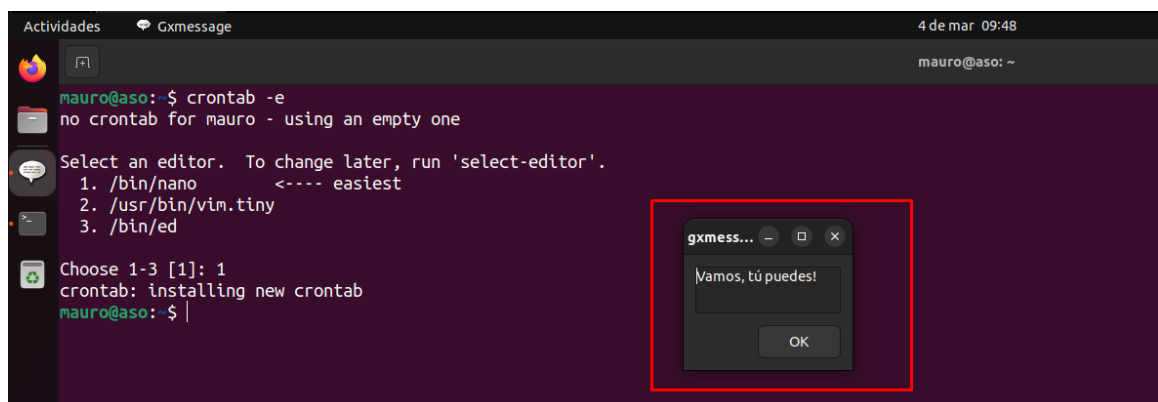
*Nota: si tu máquina no tiene entorno gráfico cambiar lo anterior para que se cree un archivo buenosdias\_%fecha.txt en el que se escriba "buenos días "y la hora que es*



```
GNU nano 6.2 /tmp/crontab.bZ089t/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
48 9 * * * DISPLAY=:0 gxmessage "Vamos, tú puedes!"
```

Ilustración 2. Programación de la tarea

Vemos que funciona perfectamente en el momento que se le ha indicado dentro de la sintaxis de la programación.



```
Actividades Gxmessage 4 de mar 09:48
mauro@aso: ~
mauro@aso:~$ crontab -e
no crontab for mauro - using an empty one

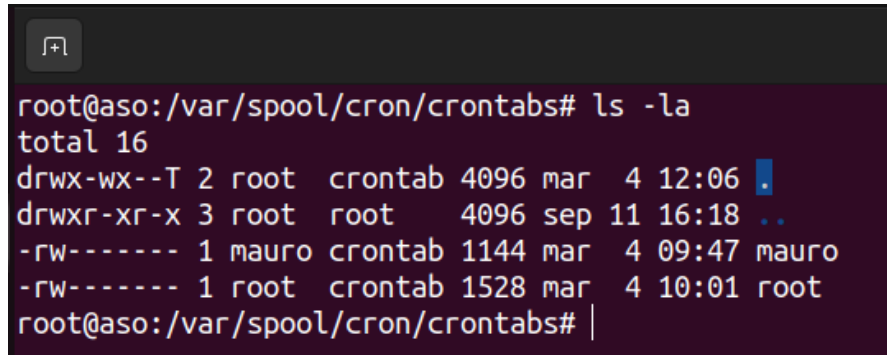
Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.tiny
 3. /bin/ed

Choose 1-3 [1]: 1
crontab: installing new crontab
mauro@aso:~$
```

Ilustración 3. Ejecución de la tarea programada

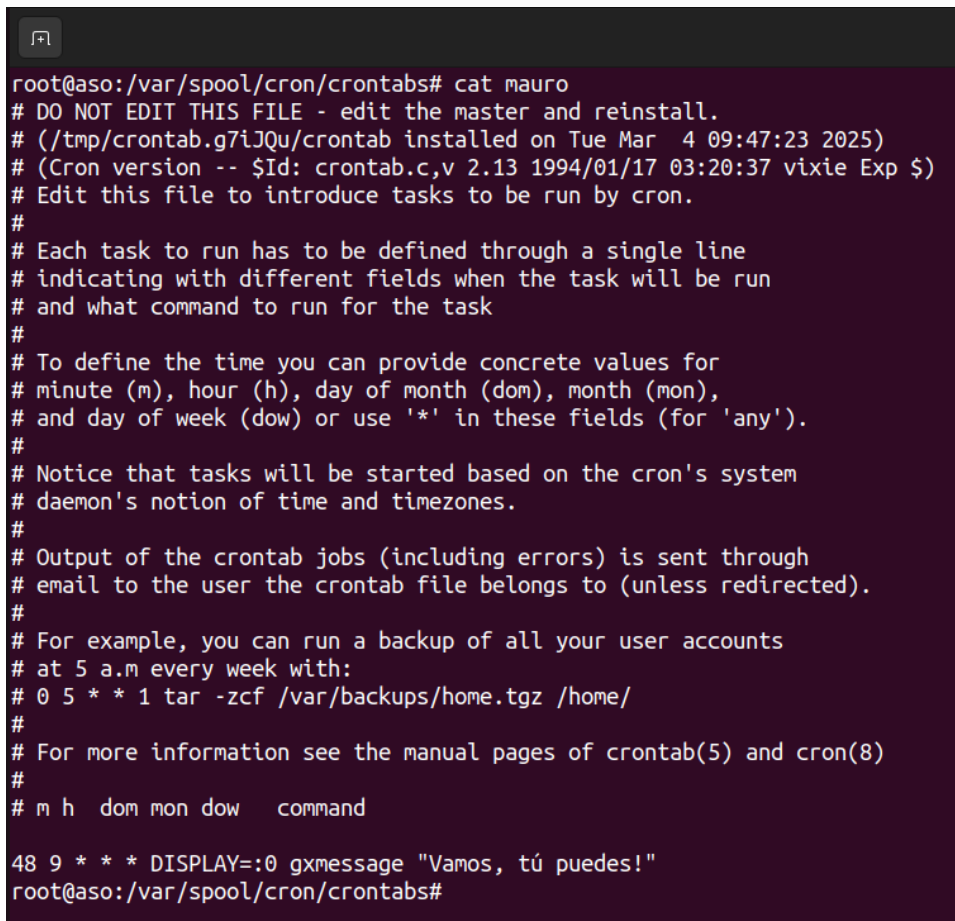
- c) ¿Cuál es la ruta donde está alojado dicho fichero crontab del usuario? Ve a dicha ruta y comprueba que efectivamente la fecha de modificación del fichero corresponde con tus modificaciones.

**La ruta del archivo crontab de un usuario del sistema se encuentra en el directorio /var/spool/cron/crontabs. Dentro de este, obviamente se encuentran los archivos de configuración personales de cada uno de los usuarios del sistema. Aquí muestro información de cada uno de ellos.**



```
root@aso:/var/spool/cron/crontabs# ls -la
total 16
drwx-wx--T 2 root  crontab 4096 mar  4 12:06 .
drwxr-xr-x 3 root  root    4096 sep 11 16:18 ..
-rw----- 1 mauro crontab 1144 mar  4 09:47 mauro
-rw----- 1 root  crontab 1528 mar  4 10:01 root
root@aso:/var/spool/cron/crontabs# |
```

*Ilustración 4. Ruta donde se encuentran los crontab personales*



```
root@aso:/var/spool/cron/crontabs# cat mauro
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.g7iJQu/crontab installed on Tue Mar  4 09:47:23 2025)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
48 9 * * * DISPLAY=:0 gxmessage "Vamos, tú puedes!"
root@aso:/var/spool/cron/crontabs#
```

*Ilustración 5. Contenido del archivo de configuración del usuario mauro*

```
root@aso:/var/spool/cron/crontabs# cat root
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.paEaXx/crontab installed on Tue Mar  4 10:01:06 2025)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
# m h dom mon dow   command
# Limpiar los archivos temporales del sistema mensualmente
0 0 1 * * rm -rf /tmp/* /var/tmp/*
# Verificar el estado del servicio "apache2" cada hora
0 * * * systemctl is-active --quiet apache2 || systemctl start apache2
# Escribir en un log si el servicio web esta levantado
0 * * * systemctl is-active --quiet apache2 || { systemctl start apache2 && echo "$(date) - apache2 was down, started now" >> /var/log/apache2_check.log; }
root@aso:/var/spool/cron/crontabs#
```

Ilustración 6. Contenido del archivo de configuración del usuario root

#### d) Verificaciones en los logs del sistema

Después de programar y probar tu tarea en cron, revisa los logs del sistema para asegurarte de que se está ejecutando correctamente:

Mostraré como se podría verificar mediante las dos utilidades, tanto syslog como journalctl para el caso de que se trabaje con systemd.

- En distribuciones basadas en **syslog** (Debian/Ubuntu), puedes consultar **/var/log/syslog** o **/var/log/cron** (si existe). Por ejemplo, ejecuta (verás líneas que indiquen la ejecución de tus tareas programadas.):

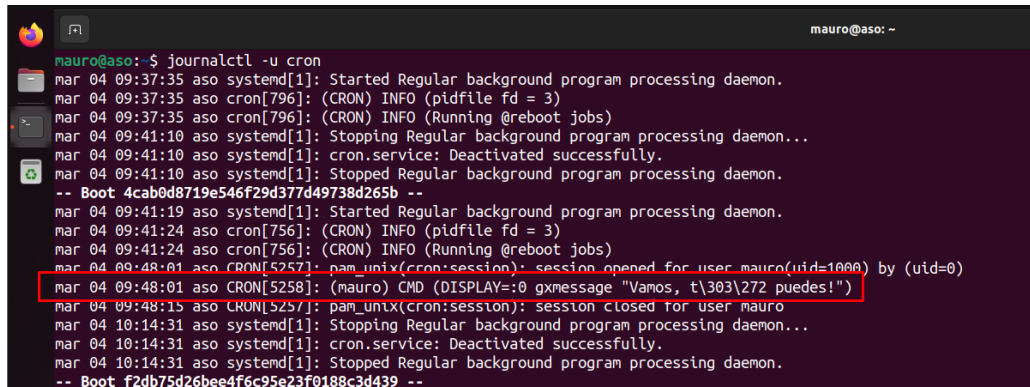
**Sudo grep cron /var/log/syslog**

```
mauro@aso:~$ tail -f /var/log/syslog
Mar  4 09:46:20 aso systemd[1]: Started Time & Date Service.
Mar  4 09:46:50 aso systemd[1]: systemd-timedated.service: Deactivated successfully.
Mar  4 09:47:16 aso crontab[5226]: (mauro) BEGIN EDIT (mauro)
Mar  4 09:47:23 aso crontab[5226]: (mauro) REPLACE (mauro)
Mar  4 09:47:23 aso crontab[5226]: (mauro) END EDIT (mauro)
Mar  4 09:48:01 aso CRON[5258]: (mauro) CMD (DISPLAY=:0 gxmessage "Vamos, t\303\272 puedes!")
Mar  4 09:48:19 aso crontab[5269]: (mauro) BEGIN EDIT (mauro)
Mar  4 09:48:30 aso crontab[5269]: (mauro) END EDIT (mauro)
Mar  4 09:49:39 aso crontab[5274]: (mauro) BEGIN EDIT (mauro)
Mar  4 09:49:52 aso crontab[5274]: (mauro) END EDIT (mauro)
Mar  4 09:51:19 aso anacron[750]: Job `cron.weekly' started
Mar  4 09:51:19 aso anacron[5303]: Updated timestamp for job `cron.weekly' to 2025-03-04
Mar  4 09:51:19 aso anacron[750]: Job `cron.weekly' terminated
```

Ilustración 7. Salida de la orden

- En distribuciones con **systemd**, también puedes usar para filtrar mensajes relacionados con el servicio de cron:

*Journalctl -u cron*



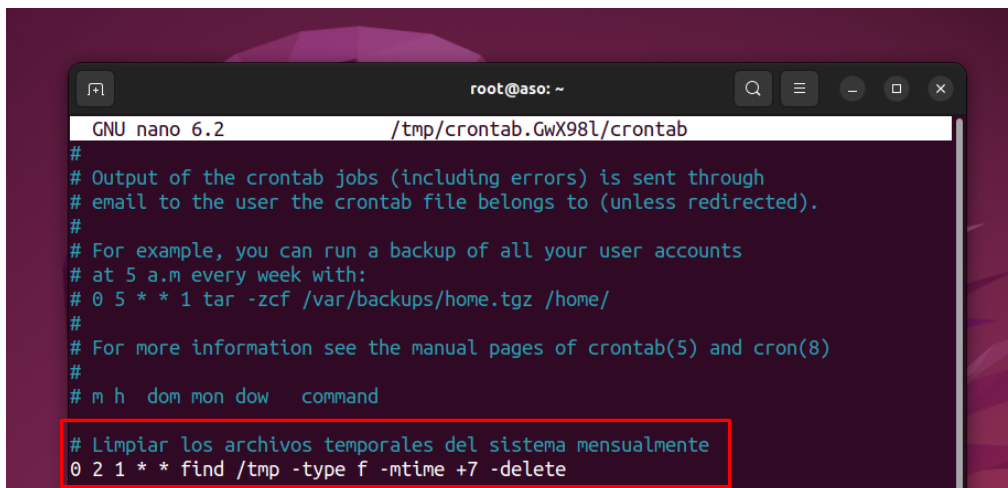
```
mauro@aso:~$ journalctl -u cron
mar 04 09:37:35 aso systemd[1]: Started Regular background program processing daemon.
mar 04 09:37:35 aso cron[796]: (CRON) INFO (pidfile fd = 3)
mar 04 09:37:35 aso cron[796]: (CRON) INFO (Running @reboot jobs)
mar 04 09:41:10 aso systemd[1]: Stopping Regular background program processing daemon...
mar 04 09:41:10 aso systemd[1]: cron.service: Deactivated successfully.
mar 04 09:41:10 aso systemd[1]: Stopped Regular background program processing daemon.
-- Boot 4cab0d8719e546f29d377d49738d265b --
mar 04 09:41:19 aso systemd[1]: Started Regular background program processing daemon.
mar 04 09:41:24 aso cron[756]: (CRON) INFO (pidfile fd = 3)
mar 04 09:41:24 aso cron[756]: (CRON) INFO (Running @reboot jobs)
mar 04 09:48:01 aso CRON[5257]: pam_unix(cron:session): session opened for user mauro(uid=1000) by (uid=0)
mar 04 09:48:01 aso CRON[5258]: (mauro) CMD (DISPLAY=:0 gmessage "Vamos, t\303\272 puedes!")
mar 04 09:48:15 aso CRON[5257]: pam_unix(cron:session): session closed for user mauro
mar 04 10:14:31 aso systemd[1]: Stopping Regular background program processing daemon...
mar 04 10:14:31 aso systemd[1]: cron.service: Deactivated successfully.
mar 04 10:14:31 aso systemd[1]: Stopped Regular background program processing daemon.
-- Boot f2db75d26bee4f6c95e23f0188c3d439 --
```

*Ilustración 8. Registro de los logs visualizados con journalctl*

Con esto podrás verificar en qué momento se lanzó tu tarea, si se produjo algún error de ejecución o si se completó con éxito.

## Ejercicio 2

- Editar el fichero *crontab* del usuario *root* (sesión usuario *root* o con "sudo crontab -e" para:
  - Que se limpien los archivos temporales del sistema manualmente.
  - Que se cheque cada hora si el servicio "apache2" está instalado y si está levantado. Si no lo está, que se levante.
  - Mejora el anterior escribiendo en su propio archivo de lo de su propio log.
  - Repite el chequeo de apache2 pero en una máquina remota. Usa conexión ssh con autenticación transparente.



```
GNU nano 6.2 /tmp/crontab.GwX98l/crontab
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
# Limpiar los archivos temporales del sistema mensualmente
0 2 1 * * find /tmp -type f -mtime +7 -delete
```

*Ilustración 9. Tarea programada en el usuario root para eliminar archivos temporales del sistema de manera mensual*

```
root@aso: ~
GNU nano 6.2 /tmp/crontab.GwX98l/crontab
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
# m h dom mon dow   command
# Limpiar los archivos temporales del sistema mensualmente
0 2 1 * * find /tmp -type f -mtime +7 -delete
# Verificar el estado del servicio "apache2" cada hora, en caso de que no se encuentre en el sistema instalado, se instala
0 * * * * systemctl list-units --full --all | grep -q apache2 || apt install -y apache2 && systemctl start apache2
```

Ilustración 10. Tarea programada en el usuario root para comprobar el estado del servicio web apache2 y en el caso de que no se encuentre instalado en el sistema lo instala al mismo tiempo

```
root@aso: ~
GNU nano 6.2 /tmp/crontab.GwX98l/crontab
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
# m h dom mon dow   command
# Limpiar los archivos temporales del sistema mensualmente
0 2 1 * * find /tmp -type f -mtime +7 -delete
# Verificar el estado del servicio "apache2" cada hora, en caso de que no se encuentre en el sistema instalado, se instala
0 * * * * systemctl list-units --full --all | grep -q apache2 || apt install -y apache2 && systemctl start apache2
# Escribir en un log si el servicio web esta levantado
0 * * * * (systemctl is-active --quiet apache2 || (echo "$(date) - Apache2 no estaba en ejecución. Reiniciando..." >> /var/log/apache_check.log && systemctl restart apache2)) 2>> /var/log/apache_check_error.log
```

Ilustración 11. Tarea programada en el usuario root para crear un log dentro del cual se indique que el servicio web se encuentra funcionando

Para el caso del apartado d, en primer lugar he tenido que generar desde mi propia máquina cliente un par de claves tanto pública como ssh empleando la sintaxis necesaria de la utilidad ssh-keygen para de esta manera, indicar al mismo tiempo sus opciones de seguridad de esta.

```
root@aso:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:uTaE9A0q1sLypJNBTVwCRRG3wIZPRACdetyucduiL5Y root@aso
The key's randomart image is:
+---[RSA 4096]-----+
| .o+#+@+o          |
| .+ *+ .           |
| o.=o o            |
| . o0o= +          |
| .B.B S .          |
| ..=o. .           |
|   = o+            |
|   E o...          |
| ..+..             |
+---[SHA256]-----+
root@aso:~#
```

Ilustración 12. Creación de la clave privada y pública desde mi propia máquina



**Seguidamente, la he enviado al contenedor web mediante la utilidad ssh-copy-id.**

```
root@aso:~# ssh-copy-id -i /root/.ssh/id_rsa.pub root@localhost
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:/yXtcMoXxEyhRWrH6CXtljEufyflQEWJn/wkCYQSSy8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@localhost's password: |
```

*Ilustración 13. Envío de la clave privada al contenedor*

**Verifico que dentro del contenedor se encuentra almacenado dentro de las claves autorizadas de este, la de mi máquina.**

```
root@1c870dad9dc9:~# cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQC1E/S+xSo2aGaIBeTwwAb8XpDSSKBDM6gFrHDVMeDIoN2sV4gyMK+Za8CpN4ZvX6Gx6k40NerUGXIqYFTscK1e3xU96XyDjNwv
Rbt1kDSGVRfFe-7bYtnFA1LM377p9mR9tAhTMqJxR/LLgPCpVgd1aWxvGAdEzWJLYXAEs3MbW4U3R5fvztNf04T8IuP+wAYOnORuFL6J7Mtp0hfhYovp45b+nLojazYC1RHoXfvk
p91kU8zj6kf0myE0CXktvJ6q/Orab1WOFU1WRBSeY2kH5jMeLZC6S86AjXSCgk8AQLbfCZofZwoZhJhEI+5xhVbmRJHg16XvQJuuXgVqr/Ns6iGHPpovB4X3ASK9sG/TCFdmWvkHd
SZGMpUCM0KV1/k049CRtBBWAZZcvft0HMDSP44M0F+HAU+jqYtPeGA106Ls3v7V4AIUquai0NB0d0mIT0qjGCF1Rn+VhVnboFa6ChyE4v1NA0FB1vENBjmyziJjFcpxVWMN43ys
QJeaA+V3+ryJgv9qkkP9J85joIGPA1wu/Hq8syitLhLbIV6Dua6LQA7h0UTjRsawcRmtRwybeKpeXlIRC0+Kxv8yBF3q90YF/R79qxLkpnQ== root@aso
root@1c870dad9dc9:~#
```

*Ilustración 14. Claves autorizadas SSH del contenedor*

**Posteriormente, he procedido con la creación de un script que me automatiza el conocimiento del estado del servicio web, sabiendo que en el caso de que dicho servicio se encuentre parado se inicio dentro de este, empleando la sintaxis correspondiente adaptado a su gestor de servicios a nivel de sistema.**

```
root@aso:~/script# cat check_apache2_docker.sh
#!/bin/bash

REMOTE_USER="root"
REMOTE_HOST="localhost"
SSH_PORT=9222
APACHE_PORT=8080
APACHE_SERVICE="apache2"

# Verificar si Apache está respondiendo en el puerto 8080
STATUS=$(curl -s --head http://$REMOTE_HOST:$APACHE_PORT | head -n 1)

# Comprobar si la respuesta contiene "200 OK"
if echo "$STATUS" | grep -q "200 OK"; then
    echo "✅ Apache está funcionando correctamente en el contenedor."
else
    echo "❌ Apache no está respondiendo en el puerto $APACHE_PORT. Intentando iniciar el servicio..."

    # Intentar iniciar el servicio apache2 dentro del contenedor
    ssh -p $SSH_PORT ${REMOTE_USER}@${REMOTE_HOST} "service $APACHE_SERVICE start"

    # Verificar si el servicio fue iniciado correctamente en el contenedor
    STATUS_AFTER_START=$(curl -s --head http://$REMOTE_HOST:$APACHE_PORT | head -n 1)

    if echo "$STATUS_AFTER_START" | grep -q "200 OK"; then
        echo "✅ Apache2 ha sido iniciado correctamente."
    else
        echo "❌ No se pudo iniciar Apache2."
    fi
fi
root@aso:~/script#
```

*Ilustración 15. Contenido del script*

**Junto con su programación dentro del crontab de mi usuario root.**

```
# Compruebo el estado del servicio web del contenedor mediante un script
*/5 * * * * /root/check_remote_apache2.sh >> /var/log/check_remote_apache2.log 2>&1
```

*Ilustración 16. Programación de la tarea dentro del crontab de mi usuario root*

Por tanto, cada vez que pasen 5 minutos en el sistema, se imprimirá por pantalla dentro de su correspondiente log el estado en el cual se encuentra actualmente el servicio web del contenedor.

```
root@aso:~/script# tail -f /var/log/check_apache2_docker.log
/bin/sh: 1: /ruta/al/script/check_apache2_docker.sh: not found
/bin/sh: 1: /root/scripts/check_apache2_docker.sh: not found
✓ Apache está funcionando correctamente en el contenedor.
```

*Ilustración 17. Log de la tarea programada en mi sistema*

3. Investiga sobre */etc/cron.daily*, */etc/cron.hourly*, etc

a) ¿Qué tienen estos directorios? ¿Para qué se usan? ¿Quién/dónde se pueden usar?

Los directorios */etc/cron.daily*, */etc/cron.hourly*, */etc/cron.weekly*, y */etc/cron.monthly* se utilizan para almacenar scripts que deben ejecutarse de manera periódica en función de la frecuencia especificada en su nombre. Son parte del sistema de gestión de tareas programadas en Linux, pero en lugar de utilizar un archivo crontab tradicional, se emplean estos directorios para simplificar la programación de tareas comunes y automáticas.

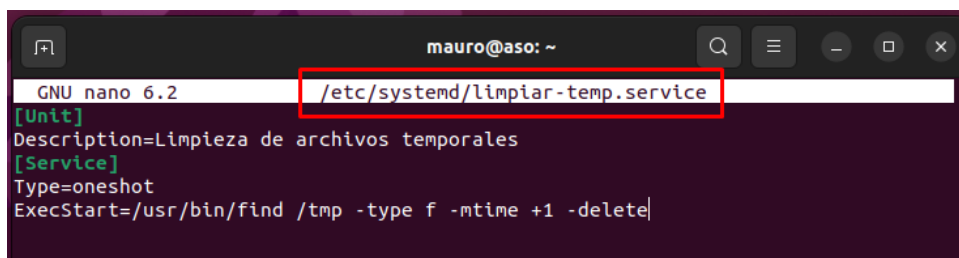
**Contenido:**

- */etc/cron.daily*: Contiene scripts que se ejecutan diariamente.
- */etc/cron.hourly*: Contiene scripts que se ejecutan cada hora.
- */etc/cron.weekly*: Contiene scripts que se ejecutan una vez por semana.
- */etc/cron.monthly*: Contiene scripts que se ejecutan una vez al mes.

Estos directorios son utilizados por los administradores del sistema (root) para organizar y programar tareas rutinarias, como limpieza de archivos temporales, actualizaciones de seguridad, análisis de logs, respaldos, entre otros.

b) Después de investigar úsalo para hacer la pregunta 2.a y de este modo no tener que usar el crontab del root.

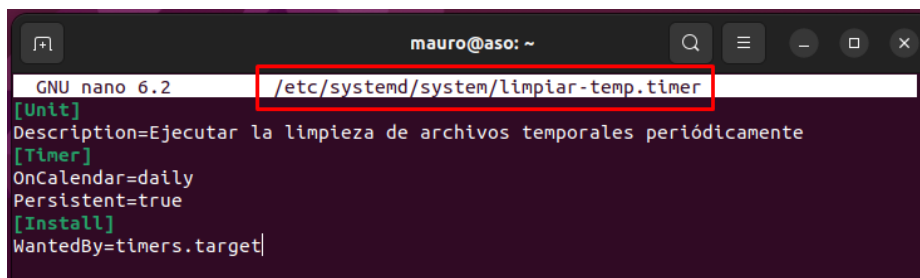
**En primer lugar, he tenido que crear un servicio systemd mediante un archivo de servicio ubicado dentro de su correspondiente ruta, indicando el siguiente contenido.**



```
mauro@aso: ~  
GNU nano 6.2 /etc/systemd/limpiar-temp.service  
[Unit]  
Description=Limpeza de archivos temporales  
[Service]  
Type=oneshot  
ExecStart=/usr/bin/find /tmp -type f -mtime +1 -delete
```

Ilustración 18. Contenido del archivo de servicio

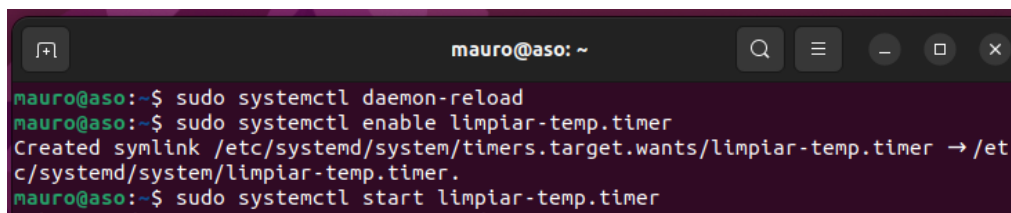
Acto seguido, se ha tenido que crear su respectivo temporizador, para poder indicar la frecuencia mediante la cual se quiere llevar a cabo dicha acción.



```
mauro@aso: ~  
GNU nano 6.2 /etc/systemd/system/limpiar-temp.timer  
[Unit]  
Description=Ejecutar la limpieza de archivos temporales periódicamente  
[Timer]  
OnCalendar=daily  
Persistent=true  
[Install]  
WantedBy=timers.target
```

Ilustración 19. Contenido del temporizador

Por último, se tiene que obviamente reiniciar cada uno de los demonios del sistema junto con la habilitación y arranque del servicio que se ha creado.



```
mauro@aso:~$ sudo systemctl daemon-reload  
mauro@aso:~$ sudo systemctl enable limpiar-temp.timer  
Created symlink /etc/systemd/system/timers.target.wants/limpiar-temp.timer → /etc/systemd/system/limpiar-temp.timer.  
mauro@aso:~$ sudo systemctl start limpiar-temp.timer
```

Ilustración 20. Recarga de los demonios del sistema, Habilitación y arranque del servicio creado

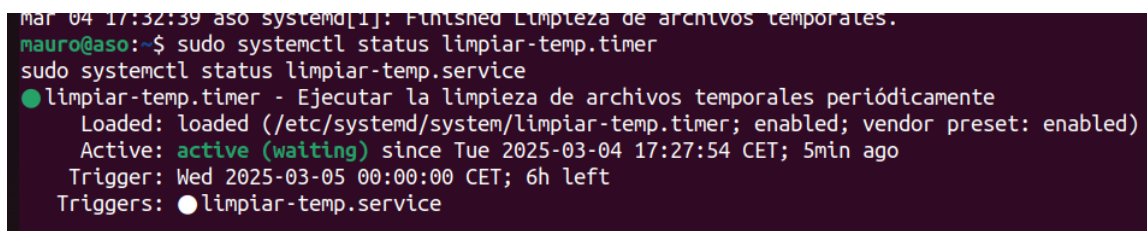
Verifico que su correspondiente temporizador esta levantado.



```
mauro@aso:~$ systemctl list-timers --all | grep limpiar  
Wed 2025-03-05 00:00:00 CET 6h left n/a n/a limpiar-temp.timer limpiar-temp.service  
mauro@aso:~$
```

Ilustración 21. Estado del levantador

Junto con el estado del servicio.



```
Mar 04 17:32:39 aso systemd[1]: Finished Limpieza de archivos temporales.  
mauro@aso:~$ sudo systemctl status limpiar-temp.timer  
sudo systemctl status limpiar-temp.service  
● limpiar-temp.timer - Ejecutar la limpieza de archivos temporales periódicamente  
Loaded: loaded (/etc/systemd/system/limpiar-temp.timer; enabled; vendor preset: enabled)  
Active: active (waiting) since Tue 2025-03-04 17:27:54 CET; 5min ago  
Trigger: Wed 2025-03-05 00:00:00 CET; 6h left  
Triggers: ● limpiar-temp.service
```

Ilustración 22. Estado del servicio