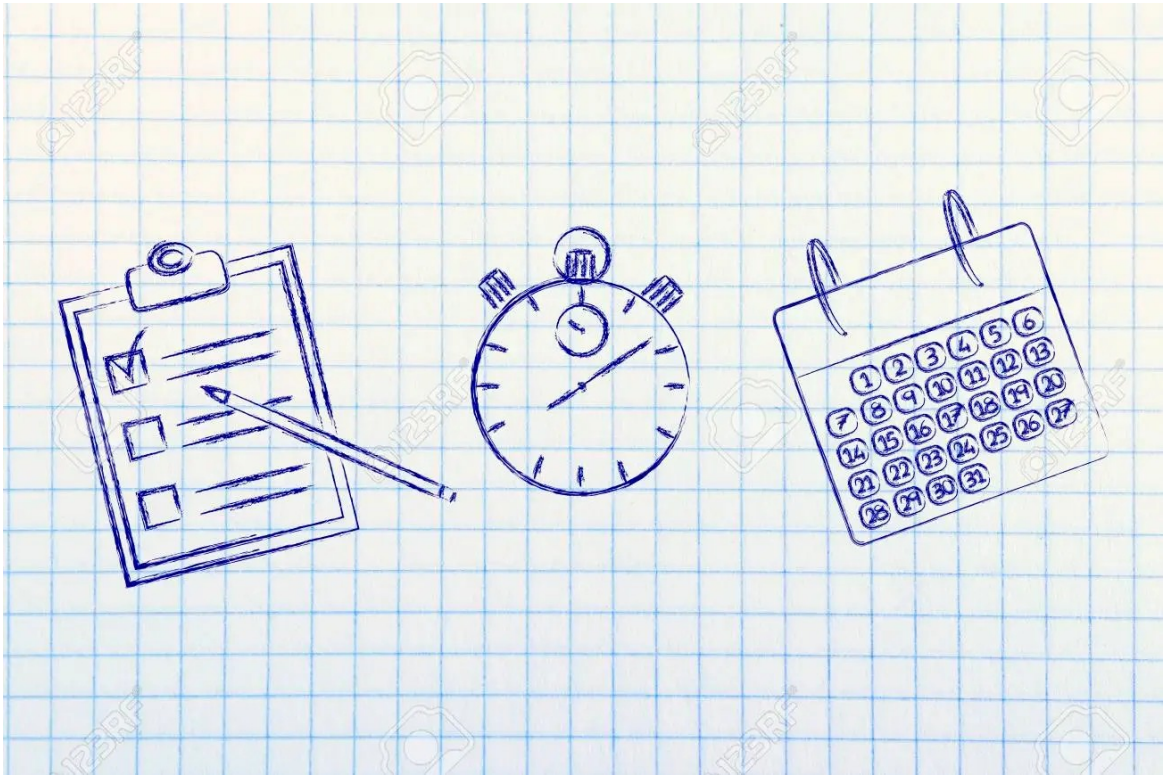
IES Gonzalo Nazareno



- Programación de tareas -

-IMPLANTACIÓN DE SISTEMAS OPERATIVOS-

01/1/2022 ***Jose Antonio Canalo Gonzalez***

Indice

INTRODUCCIÓN

Programación de tareas

Crea un manual que cubra los siguientes comandos relacionados con la **programación de tareas**:

watch, at.

Deberás explicar las opciones más comúnmente utilizadas, así como añadir ejemplos de utilización de los diferentes comandos.

WATCH

Con este comando podemos ejecutar un programa u otro comando cada X segundos que le pongamos. Así programamos la ejecución repetitiva de una cierta tarea. Esto sirve de gran ayuda para ciertas consultas periódicas o para algunas labores de mantenimiento etc. Es un programa con miles de combinaciones diferente.

Y ahora nos metemos en materia con su ejecución

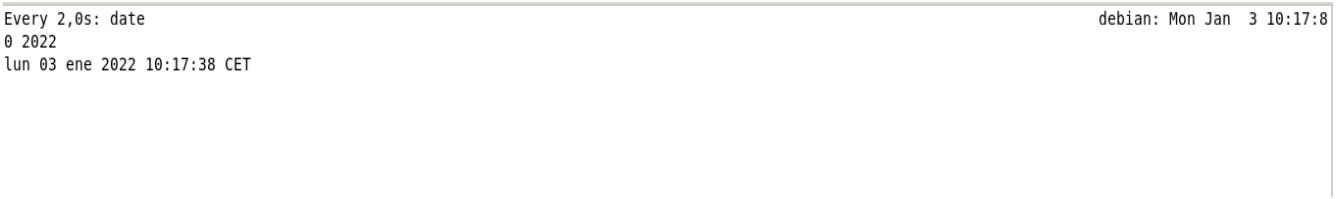
Ejemplos útiles del comando watch

La suerte del comando **Watch** es que no usa sintaxis muy complicadas para su ejecución.

Sin argumentos

watch date

Si ejecutamos el comando de esta manera nos muestra en la parte izquierda el comando ejecutado y el su periodo de intervalo



watch -n 5 date

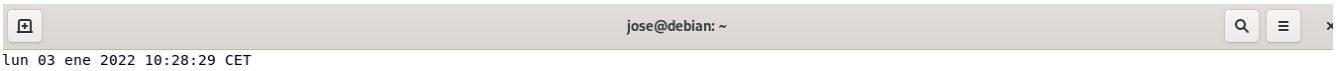
Ahora Vamos a especificar el período del intervalo para la actualización del comando **watch** muy fácilmente utilizando la **opción -n**. El intervalo de tiempo debe ser establecido en segundos. En este caso le daremos 5 segundos

Eliminación de título y encabezados

El comando watch muestra información en pantalla como el nombre del comando que se está ejecutando, el intervalo y la hora actual. Todo se sitúa en la parte superior de la pantalla. Si queremos evitarla, podremos utilizar la **opción -t** para desactivar esta información.

```
watch -t date
```

Como podíamos ver el comando **watch** muestra información en pantalla como el nombre del comando que se está ejecutando, el intervalo y la hora actual. Todo se sitúa en la parte superior de la pantalla. Si queremos evitarla, podremos utilizar la **opción -t** para desactivar esta información.

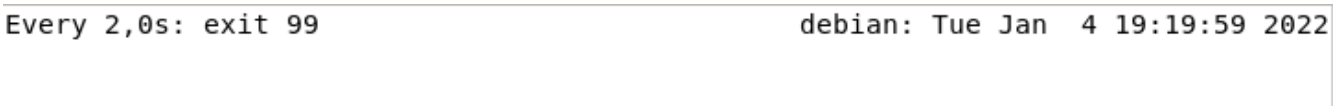


Salir de Watch en caso de error

También podemos especificar una vigilancia para salir siempre que haya un error producido por el comando que se está ejecutando. Simplemente tendremos que utilizar la **opción -e**.

```
watch -e exit 99
```

Al ejecutar este comando, vemos un mensaje que indica que el comando tiene un estado de salida distinto de cero. Hay que tener en cuenta que los comandos que se ejecutan sin ningún error, salen con un código de estado cero.



Salir si se producen cambios en la salida del comando

La **opción -g** sale de **watch** siempre que se produzca un cambio en la salida del comando.

```
watch -g date
```

Este comando se ejecutará durante **dos segundos** y, tan pronto como se actualice la salida, **watch** se cerrará.

(Después de números intentos no fui capaz de capturar porque a los dos segundos se quita)

Notificar en caso de error

La **opción -b de watch** emite un pitido cada vez que el comando sale con un código de estado distinto de cero. Como ya se ha comentado, un código de estado distinto de cero suele indicar un error o que se ha fallado en la ejecución del comando.

```
watch -b exit 99
```

Interpretar códigos de colores y secuencias de estilos

Podremos habilitar la interpretación de los códigos de color ANSI y las secuencias de estilo para **watch** usando la **opción -c**. Por defecto, **watch** no interpreta los colores en su salida.

```
1 watch -c echo "$(tput setaf 2) para la practica de rafa"
```

Este comando muestra la cadena codificada en verde '*Ejemplo para la practica de rafa*'. Si eliminamos la **opción -c** y volvemos a ejecutar el comando, veremos que la cadena no contiene ningún color esta vez.

Supervisar cambios en el contenido del directorio

El siguiente ejemplo ilustra cómo podemos usar la utilidad **watch** para monitorizar los directorios del sistema de archivos en busca de cambios del contenido.

```
jose@debian:~/comparadores$ ls -l
total 16
-rw-r--r-- 1 jose jose 44 dic 15 09:37 texto1
-rw-r--r-- 1 jose jose 45 dic 15 09:13 texto2
-rw-r--r-- 1 jose jose  8 dic 16 10:35 texto3
-rw-r--r-- 1 jose jose  8 dic 16 10:35 texto4
jose@debian:~/comparadores$
```

watch -d ls -l

```
Every 2,0s: ls -l                                                                                   debian: Tue Jan  4 19:30:21 2022

total 16
-rw-r--r-- 1 jose jose 44 dic 15 09:37 texto1
-rw-r--r-- 1 jose jose 45 dic 15 09:13 texto2
-rw-r--r-- 1 jose jose  8 dic 16 10:35 texto3
-rw-r--r-- 1 jose jose  8 dic 16 10:35 texto4
```

Este comando imprimirá la lista del directorio y resaltará los cambios en el contenido

Monitorizar la temperatura de la CPU usando watch

También podemos controlar la temperatura. Podremos utilizar la utilidad **watch** junto con sensors para controlar la temperatura del equipo.

watch -n 60 sensors

```
Every 60,0s: sensors                                                                                debian: Tue Jan  4 19:31:01 2022

asus-isa-0000
Adapter: ISA adapter
cpu_fan:        0 RPM
temp1:        +32.0°C

acpitz-acpi-0
Adapter: ACPI interface
temp1:        +32.0°C  (crit = +108.0°C)
temp2:        +27.8°C  (crit = +105.0°C)
temp3:        +29.8°C  (crit = +105.0°C)

coretemp-isa-0000
Adapter: ISA adapter
Package id 0:  +33.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:        +33.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:        +33.0°C  (high = +105.0°C, crit = +105.0°C)

pch_wildcat_point-virtual-0
Adapter: Virtual device
temp1:        +33.5°C

BAT0-acpi-0
Adapter: ACPI interface
in0:           7.50 V
```

Este comando verificará la temperatura del equipo por minuto.

El comando **watch** es el comando soñado por todos es un comando fácil de usar y casi de infinitas posibilidades, que es prácticamente imposible mostrar aquí así que para eso dejo la ayuda y el manual

Mostrar la página de ayuda y manual

No dudes en **consultar la ayuda del comando watch** si quieres información rápida para una opción específica.

```
watch -h
```

```
jose@debian:~/comparadores$ watch -h

Usage:
 watch [options] command

Options:
  -b, --beep             beep if command has a non-zero exit
  -c, --color            interpret ANSI color and style sequences
  -d, --differences[=<permanent>]
                         highlight changes between updates
  -e, --errexit          exit if command has a non-zero exit
  -g, --chgexit          exit when output from command changes
  -n, --interval <secs>  seconds to wait between updates
  -p, --precise          attempt run command in precise intervals
  -t, --no-title         turn off header
  -w, --no-wrap          turn off line wrapping
  -x, --exec             pass command to exec instead of "sh -c"

 -h, --help     display this help and exit
 -v, --version  output version information and exit

For more details see watch(1).
```

También vamos a poder **consultar la página del manual** para obtener información detallada sobre una determinada opción.

man watch

```
WATCH(1)                                                User Commands                                                WATCH(1)

NAME
       watch - execute a program periodically, showing output fullscreen

SYNOPSIS
       watch [options] command

DESCRIPTION
       watch runs command repeatedly, displaying its output and errors (the first screenfull).  This allows you to watch the program output change
       over time.  By default, command is run every 2 seconds and watch will run until interrupted.

OPTIONS
       -d, --differences[=permanent]
              Highlight the differences between successive updates. If the optional permanent argument is  specified  then  watch  will  show  all
              changes since the first iteration.

       -n, --interval seconds
              Specify  update  interval.   The command will not allow quicker than 0.1 second interval, in which the smaller values are converted.
              Both '.' and ',' work for any locales. The WATCH_INTERVAL environment can be used to persistently set a non-default  interval  (fol-
              lowing the same rules and formatting).

       -p, --precise
              Make  watch attempt to run command every --interval seconds.  Try it with ntptime (if present) and notice how the fractional seconds
              stays (nearly) the same, as opposed to normal mode where they continuously increase.

       -t, --no-title
              Turn off the header showing the interval, command, and current time at the top of the display, as well as the following blank line.

       -b, --beep
              Beep if command has a non-zero exit.

       -e, --errexit
              Freeze updates on command error, and exit after a key press.

       -g, --chgexit
              Exit when the output of command changes.
 Manual page watch(1) line 1 (press h for help or q to quit)
```

AT

El comando ***at***, es útil para apagar el sistema a una hora especifica, realizar una copia de seguridad única, enviar un correo electrónico como recordatorio a la hora especificada, entre algunas de sus funciones mas destacadas.

Ejemplos útiles del comando AT

Para empezar podemos decir que AT tiene tres comandos principales que son

- **at**: ejecuta comandos a la hora especificada.
- **atq**: enumera los trabajos pendientes de los usuarios.
- **atrm**: borra trabajos por su número de trabajo.

las fechas y horas pueden ser explícitas o relativas. Por ejemplo, supongamos que deseas que se ejecute un comando dentro de un minuto. **at** sabe lo que significa “ahora”, por lo que puedes usar now y agregarle un minuto, así:

at now + 1 minute

```
1 at now + 1 minute
```

at imprime un mensaje y una solicitud **at,** y espera a que escribas los comandos que deseas programar. Primero, sin embargo, considera el mensaje, como se muestra a continuación:

```
jose@debian:~$ at now + 1 minute
warning: commands will be executed using /bin/sh
at>
```

Le dices que **at** inicie una instancia de **la shell sh y ejecutará** los comandos dentro de esta. Tus comandos no se ejecutarán en la **shell Bash**, que es compatible con la **shell sh** pero tiene un conjunto de características más completo.

Si tus comandos o scripts intentan usar una función o facilidad que **Bash** proporciona, pero **sh** no lo hace, fallarán.

Es fácil probar si tus comandos o scripts se ejecutarán en **sh**. Usa el comando **sh** para iniciar una **shell**

```
jose@debian:~$ sh
$
```

la terminal cambia a un signo de dólar ($), y ahora puedes ejecutar tus comandos y verificar que funcionen correctamente.

Para volver a la **shell Bash**, escribe el comando **exit:**

Programación de tareas en un corto y largo espacio

1. Programar una tarea a las cinco de la tarde

    at 05:00 PM

2. Programar una tarea a las tres de la mañana

    at 03:00 AM

3. Programar una tarea a las siete de la mañana del próximo domingo

    at 07:00 AM Sun

4. Programar la tarea a las 11:00 AM del próximo 14 de abril

    at 11:00 AM April 14

5. Programar una tarea a las 10:00 AM del próximo 22 de junio de 2021

    at 10:00 AM 6**/**22**/**2021
    at 10:00 AM 6.22.2021

6. Programar una tarea a las 11 de la mañana el mismo día del próximo mes

    at 11:00 AM next month

7. Programar una tarea para mañana a las 11 AM

    at 11:00 AM tomorrow

8. Programar una tarea de aquí a una hora

    at now + 1 hour

9. Programar una tarea de aquí a 30 minutos

    at now + 30 minutes

10. Programar una tarea de aquí a una y dos semanas

    at now + 1 week

at now + 2 weeks

11. Programar una tarea de aquí a uno y dos años

at now + 1 year
at now + 2 years

12. Programar una tarea a media noche

at midnight

13. **Programar una tarea todos los dias**

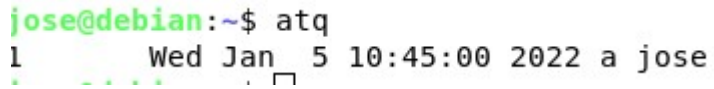
at everyday

13. **Programar una para la hora del te**

at teatime

Observando la cola de trabajo con AT

Puede escribir el comando **atq** para ver la cola de trabajos programados



Para cada comando en la cola, atq muestra la siguiente información:

- **ID Job**
- **Cita programada**
- **Hora programada**
- **Hacer cola en el trabajo**. Las colas están etiquetadas como “a”, “b”, etc. Las tareas normales que programa at son la cola “a”, mientras que las tareas que programa **batch** (cubiertas más adelante en este artículo) entran en la cola “b”.
- **La persona que programó el trabajo.**

Eliminacion de tareas con AT

Cómo ver una vista detallada de *trabajos*

Como mencionamos anteriormente, puedes programar trabajos en el futuro. A veces, puedes olvidar lo que va a hacer un trabajo. El comando **atq** te muestra los trabajos en la cola, pero no lo que van a hacer. Si deseas ver una vista detallada de un trabajo, puedes usar la opción **–c (cat).**

Primero, usaremos **atq** para encontrar el número de trabajo:

atq

```
atq
```

Ahora, usaremos el trabajo número 13 con la opción –c:

at -c 13

```
at -c 13
```

(Este ejemplo tuve que coger el que venia de internet porque no me salia)

```
dave@howtogeek:~$ at -c 13
#!/bin/sh
# atrun uid=1000 gid=1000
# mail dave 0
umask 22
CLUTTER_IM_MODULE=xim; export CLUTTER_IM_MODULE
LS_COLORS=rs=0:di=01\;34:ln=01\;36:mh=00:pi=40\;33:so=01\;35:do=01\;35
:bd=40\;33\;01:cd=40\;33\;01:or=40\;31\;01:mi=00:su=37\;41:sg=30\;43:c
a=30\;41:tw=30\;42:ow=34\;42:st=37\;44:ex=01\;32:\*.tar=01\;31:\*.tgz=
01\;31:\*.arc=01\;31:\*.arj=01\;31:\*.taz=01\;31:\*.lha=01\;31:\*.lz4=
01\;31:\*.lzh=01\;31:\*.lzma=01\;31:\*.tlz=01\;31:\*.txz=01\;31:\*.tzo
=01\;31:\*.t7z=01\;31:\*.zip=01\;31:\*.z=01\;31:\*.Z=01\;31:\*.dz=01\;
31:\*.gz=01\;31:\*.lrz=01\;31:\*.lz=01\;31:\*.lzo=01\;31:\*.xz=01\;31:
\*.zst=01\;31:\*.tzst=01\;31:\*.bz2=01\;31:\*.bz=01\;31:\*.tbz=01\;31:
\*.tbz2=01\;31:\*.tz=01\;31:\*.deb=01\;31:\*.rpm=01\;31:\*.jar=01\;31:
\*.war=01\;31:\*.ear=01\;31:\*.sar=01\;31:\*.rar=01\;31:\*.alz=01\;31:
\*.ace=01\;31:\*.zoo=01\;31:\*.cpio=01\;31:\*.7z=01\;31:\*.rz=01\;31:\
*.cab=01\;31:\*.wim=01\;31:\*.swm=01\;31:\*.dwm=01\;31:\*.esd=01\;31:\
*.jpg=01\;35:\*.jpeg=01\;35:\*.mjpg=01\;35:\*.mjpeg=01\;35:\*.gif=01\;
35:\*.bmp=01\;35:\*.pbm=01\;35:\*.pgm=01\;35:\*.ppm=01\;35:\*.tga=01\;
35:\*.xbm=01\;35:\*.xpm=01\;35:\*.tif=01\;35:\*.tiff=01\;35:\*.png=01\
```

Figura 1: Ejemplo cogido de la pagina de https://www.blog.binaria.uno/2020/01/27/como-usar-at-y-batch-en-linux-para-programar-comandos/

Aquí hay un desglose de la información que recibimos sobre el trabajo:

- **Primera línea:** esto nos dice que los comandos se ejecutarán bajo la shell sh.
- **Segunda línea:** vemos que los comandos se ejecutarán con un ID de usuario y de grupo de 1000. Estos son los valores para la persona que ejecutó el comando at.
- **Tercera línea:** la persona que recibe los correos electrónicos que at envía.
- **Cuarta línea:** la máscara de usuario es 22. Esta es la máscara utilizada para establecer los permisos predeterminados para cualquier archivo creado en esta sesión sh. La máscara se resta de 666, lo que nos da 644 (el equivalente octal de rw-r–r–).
- **Datos restantes:** la mayoría son variables de entorno.
- **Resultados de una prueba**. Una prueba verifica para asegurarse de que se pueda acceder al directorio de ejecución. Si no puede, se genera un error y se abandona la ejecución del trabajo.
- **Los comandos a ejecutar.** Estos se enumeran y se muestra el contenido de los scripts programados. Ten en cuenta que, aunque el script en nuestro ejemplo anterior fue escrito para ejecutarse bajo Bash, todavía se ejecutará en una shell sh.

Crontab

Son archivos de texto, se trata de un archivo con un contenido especial y específicamente diseñado para que sea leído correctamente por Cron(es un demonio que se ejecuta desde el mismo instante en el que arranca el sistema operativo. Cron se encargará de comprobar si existe alguna tarea (job) para ser ejecutada, de acuerdo a la hora configurada en el propio sistema operativo. Es muy importante que la hora esté bien configurada, y también la zona horaria, de lo contrario, las ejecuciones que realice Cron no se corresponderán con nuestras configuraciones. y proceder con la ejecución que nosotros hayamos programado. Crontab posee una lista con todos los scripts a ejecutar, generalmente cada usuario del sistema posee su propio fichero Crontab, de esta forma, cada usuario podría programar sus propias tareas repetitivas independientemente, sin necesidad de que siempre tengamos que acudir al usuario administrador. De esta forma, cualquier usuario (incluyendo los administradores) podrán programar tareas repetitivas para realizar diferentes ejecuciones.

Para generar el archivo propio, cada usuario deberá hacer uso del comando «crontab»

Funcionamiento de Crontab

Para garantizar que nuestro sistema operativo esté configurado correctamente, es muy importante que obtenga la hora automáticamente de los servidores NTP (Network Time Protocol) que existen. Para comprobar que tenemos la hora correcta en nuestro sistema operativo, en qué zona horaria estamos y si el reloj del sistema está correctamente sincronizado con los servidores NTP, podemos ejecutar la siguiente orden

jose@debian:~$ timedatectl

```
jose@debian:~$ timedatectl
               Local time: vie 2022-01-07 10:30:42 CET
           Universal time: vie 2022-01-07 09:30:42 UTC
                 RTC time: vie 2022-01-07 10:31:11
                Time zone: Europe/Madrid (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: yes
```

Tal y como se puede ver, todas las horas cuadran perfectamente y el tiempo es el correcto, esto es muy importante, de lo contrario, los «**Cron»** se ejecutarán en horas donde no deberían ejecutarse. En el caso de que no tengamos la zona horaria bien configurada, podremos configurarla de forma correcta con la siguiente orden (en mi caso porque estoy dentro de la peninsula iberica)

jose@debian:~$ timedatectl set-timezone Europe/Madrid

Agregar tareas a Crontab

Partiendo de que podemos ejecutar tareas, en primer lugar, vamos a configurar un script muy sencillo que llamaremos consulta.sh, pondremos en nuestro «Escritorio» o en «Documentos» este script, podremos ejecutar directamente el editor de texto «nano» en la ubicación donde queramos ponerlo:

```
jose@debian:~/comparadores$ nano consulta.sh
```

El contenido del script sería el siguiente

```
#!/bin/bash
```

Una vez que hayamos programado el script, que básicamente es un programa que lista los archivos y directorios del directorio actual, y lo exporta a un archivo de texto, tenemos que darle permisos de ejecución para poder ejecutarlo correctamente por parte de Cron, de lo contrario no tendrá permisos de ejecución:

```
jose@debian:~/comparadores$ chmod ugo+x consulta.sh
jose@debian:~/comparadores$
```

Ha llegado el momento de editar el fichero que posee las tareas. Para ello nos vamos a ayudar del comando crontab -e. Nos encontramos la siguiente estructura:

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

5 asteriscos y el comando a ejecutar. Cada uno de los 5 asteriscos significa:

- m: minuto
- h: hora
- dom: día de la semana
- mon: mes
- dow: día del mes

Lo valores que pueden adoptar cada una de estas variables se encuentran en la imagen. Por ejemplo, en el caso del último, podemos escribir el dia eso si hay que recordar que hay que hacerlo en ingles. Aunque en la imagen anterior no aparezca, es necesario indicar entre el comando y el último asterisco (el día) el propietario del archivo.

Para que quede todo claro, vamos a utilizar algunos ejemplos:

```
15 10 * * * /home/jose/pruebas/consulta.sh
```

Ejecutará el script consultas.sh a las 10:15 a.m. todos los días

aqui pongo una web con muchos mas ejemplos

https://crontab.tech/examples

Por último y no menos importante:

Administracion de trabajos en cron

crontab archivo

Remplaza el existente archivo crontab con un archivo definido por el usuario

crontab -e

Editar el archivo crontab del usuario, cada linea nueva sera una nueva tarea de crontab.

crontab -l

Lista todas las tareas de crontab del usuario

crontab -d

Borra el crontab del usuario

crontab -c dir

Define el directoriod e crontab del usuario (este debe tener permisos de escritura y ejecucion del usuario)

crontab -u usuario

CONCLUSIÓN

La verdad que me resulto una practica muy amena sobre todo el punto de **watch** y **cronbat,** la parte de **AT** no le acabe de coger el punto.

BIOGRAFÍA O FUENTES UTILIZADAS

https://ubunlog.com/comando-watch-algunas-formas-de-uso/
https://www.linuxadictos.com/ejecutar-un-comando-linux-cada-cierto-tiempo-con-watch.html
https://www.youtube.com/watch?v=bzsNsEdF4f0
https://www.ochobitshacenunbyte.com/2019/07/01/comando-at-programacion-de-tareas-unicas-en-linux/
https://www.youtube.com/watch?v=DFWFGQcqZGk
https://crontab.tech/examples
https://blog.desdelinux.net/cron-crontab-explicados/?utm_source=destacado-inside