CFGS ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS

Ud14.- Programación de tareas.

Índice

1.- Introducción.

Puede ocurrir que en algunas ocasiones nos interese ejecutar un determinado comando a una hora determinada.

También resulta muy interesante el poder ejecutar de forma periódica una tarea.

¿Se podría hacer una copia de seguridad todos los viernes a las 4 de la tarde?

2.- Programar una tarea a una hora determinada

2.1.- El comando at.

Este comando nos permite lanzar una acción a una determinada hora.

$ at [opciones] TIME $at TIME MMDDAA

TIME → HH:MM Es la hora en la que debe realizarse la tarea.
MMDDAA → Mes Día Año

Cuando ejecutamos ese comando nos devuelve el número de trabajo programado y el momento en el que se llevaría a cabo la ejecución.
Para terminar pulsamos ctrl+d

Opciones:

-f file → Lee los comandos desde un archivo

2.- Programar una tarea a una hora determinada

2.1.- El comando at.

Ejercicio :

1.- Visualiza la fecha y hora del sistema. Actualizarla. #date - -set="año-mes-dia hh:mm"

2.- Programar una tarea para las HH:MM que haga un listado de nuestro directorio de trabajo y guarde el resultado en listado.txt.

3.- Programa una tarea para las HH:MM que muestre un mensaje a todos los usuarios dando la bienvenida al sistema.

echo "Bienvenidos al sistema"|wall

2.- Programar una tarea a una hora determinada

2.1.- El comando at.

Ejercicio :

```
usuario@debian-pc200:~$ date
lun oct  1 06:24:27 CEST 2012
usuario@debian-pc200:~$ at 06:30
warning: commands will be executed using /bin/sh
at> ls >listado.txt
at> <EOT>
job 1 at Mon Oct  1 06:30:00 2012
usuario@debian-pc200:~$ at 06:31
warning: commands will be executed using /bin/sh
at> echo "Bienvenidos al sistema"|wall
at> <EOT>
job 2 at Mon Oct  1 06:31:00 2012
usuario@debian-pc200:~$
```

2.- Programar una tarea a una hora determinada

2.1.- El comando at.

Ejercicio :

1.- Dentro de nuestro directorio de trabajo, crea dos archivos llamados, ejercicio1 y ejercicio2. Crea un subdirectorio llamado seguridad.

2.- Realiza un pequeño script (copia, sin extensión) en tu directorio de trabajo, que copie los archivos del tipo ejercicio* al directorio seguridad.

3.- Darle al script copia los permisos de ejecución: $chmod a+x copia

4.- Comprobar que el script funciona.

5.- Programar una tarea para las HH:MM que ejecute el script copia.

2.- Programar una tarea a una hora determinada

2.1.- El comando at.

Ejercicio : Interpreta los siguientes formatos de hora y fecha.

```
# at 20:15

# at 10:00 06/08/10

#at 14:30 jul 25

# at now +3 minutes

# at now +4 days

# at 4pm +2 days

# at 10am tomorrow
```

Ejercicio : Manda un Mensaje de felicitación el día MMDDAA a las HHMM.

2.- Programar una tarea a una hora determinada

2.2.- El comando atq

Nos muestra un listado de las tareas programadas.

Si se ejecuta como root, se verán todas las tareas, si no, las propias de cada usuario.

Con el comando atq vemos el identificador asignado a las tareas pendientes y con el comando: $ at -c nº_trabajo, vemos el contenido de la tarea.

2.- Programar una tarea a una hora determinada

2.2.- El comando atq

Ejercicio : usuario, heidi y root

1.- Como usuario lanza una tarea para dentro de 10 minutos que mande un mensaje: “Hola ya estoy aquí”.

2.- Como heidi lanza una tarea para dentro de 20 minutos que mande un mensaje: “Esta tarde toca fútbito”.

3.- Entra cómo root, y visualiza todas las tareas pendientes.

4.- Podrías averiguar a quién corresponde cada una.

5.- Obtén información sobre una de las tareas.

2.- Programar una tarea a una hora determinada

2.3.- El comando atrm

Sirve para cancelar tareas programadas.

$ atrm Nº_tarea

Ejercicio :

1.- Visualiza todas las tareas pendientes del sistema.

2.- Cancélalas.

3.- Visualiza si queda alguna pendiente.

2.- Programar una tarea a una hora determinada

2.4.- Ficheros de configuración.

```
heidi@servidor200:~$ at now +1minute
You do not have permission to use at.
heidi@servidor200:~$ _
```

/etc/at.deny → Aparece el listado de los usuarios que no pueden realizar una programación de tareas:

Ejercicio :

1.- Añadir a la lista negra :-) el usuario heidi.

2.- Intentad realizar una tarea programada con el usuario heidi.

2.- Programar una tarea a una hora determinada

Resumen:

Ahora es el momento de realizar un resumen de lo que hemos visto.

3.- Programar una tarea periódicamente.

3.1.- El demonio cron.

Es el demonio (proceso que se ejecuta en segundo plano) que se encarga de comprobar si hay tareas pendientes para posteriormente ejecutarlas.

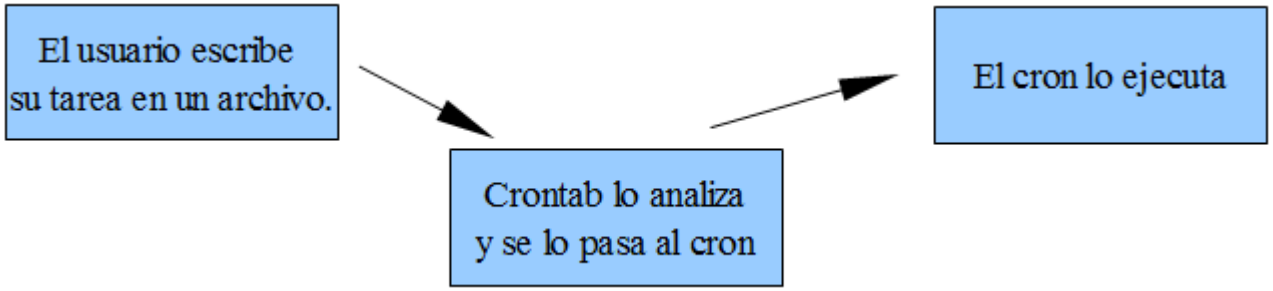
Ejemplo:

Cuando alguien escribe en la plataforma Moodle del Dpto, al cabo de un cierto tiempo, le llega a todos los usuarios. El cron examina si hay alguna tarea pendiente y la ejecuta.

3.- Programar una tarea periódicamente.

3.2.- Programación de tareas.

Las tareas no se pasan directamente a cron, sino que se incluyen en un archivo con un determinado formato y son establecidas mediante el comando crontab.



3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

Podemos abrir un archivo nuevo o bien abrir uno que está preparado, modificarlo y grabarlo con otro nombre.

IMPORTANTE:

Si no se guarda bajo otro nombre, sino que se deja el que viene por defecto, automáticamente se le está pasando al cron y se ejecuta.

TRABAJAREMOS: Vamos a editar el que está preparado ,lo modificaremos y lo guardamos bajo otro nombre.

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

$ crontab –e

Esta orden lanza nuestro editor favorito y abre un fichero donde se almacenan todos los trabajos que se lanzan periódicamente. Este fichero tiene un formato específico .

```
  GNU nano 2.2.4      Fichero: /tmp/crontab.5GT2mV/crontab

# 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
```

Ejercicio:

Lee las indicaciones de este fichero

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

Cambiar el editor de texto: Sistema

#update-alternatives - -config editor

Elegimos el editor de texto.

Cambiar el editor de texto: para nuestro usuario temporalmente

Export EDITOR=/usr/bin/nano

Cambiar el editor de texto: para nuestro usuario permanentemente.

Introducir la línea anterior en .bashrc

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

Bastará con poner un valor debajo de cada uno de los campos.

Campo	Posible valores
m (minutos)	[0-59]
h (hora)	[0-23]
dom (día del mes)	[1-31]
mon (mes)	[1-12]
dow (día de la semana)	[0-6] 0 Domingo 6 Sábado
Comando	El comando o comandos a ejecutar.

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

También se pueden especificar los siguientes ’sinónimos’:

@reboot: cada vez que el sistema arranque.

@yearly: equivale a 0 0 1 1 * (00:00 1Ene)

@monthly: equivale a 0 0 1 * *

@weekly: equivale a 0 0 * * 0

@daily: equivale a 0 0 * * *

@hourly: equivale a 0 * * * *

En vez de escribir los primeros 5 campos, escribimos alguna de las palabras claves y a continuación escribimos el comando a ejecutar.

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

Símbolos especiales:

Símbolos especiales	Significado
*	Indica cualquier valor. Ejemplo: En el campo del mes ponemos * La tarea se realizará cualquier mes.
*/t	Períodos de tiempo. Ejemplo: En el campo mes ponemos */3 La tarea se realizará cada tres meses
,	Separador de una lista de valores.
-	Rango de valores
#	Comentario

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

Ejercicio : Interpreta las siguientes líneas:

En cada línea se expresan condiciones AND; es decir, se deben cumplir las 5 condiciones para que se ejecute el comando.

m	h	dom	mon	dow	Comando	Interpretación
0	0	*	*	*	shutdown -h now	
0	7	*	*	*	/home/heidi/copia.sh	
0	7	1	*	*	/home/heidi/copia.sh	
30	21	*	*	5	/home/heidi/copia.sh	

m	h	dom	mon	dow	Comando	Interpretación
0,15,30,45	*	*	*	*	date>>fecha	
*/15	*	*	*	*	date>>fecha	

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

Ejercicio : (usuario)

Diseñar una tarea que se ejecute cada minuto y nos incluya la fecha y hora actual en un archivo llamado Todos_Los_Minutos.txt.

La tarea programada la vamos a guardar dentro de nuestro directorio de trabajo con el nombre de Mis_Tareas.

3.- Programar una tarea periódicamente.

3.3.- Formato del archivo.

Ejercicio :

1.- Abrimos el archivo crontab: $ crontab -e

2.- Añadimos la línea adecuada:

```
m     h     dom     mon     dow     Comando
*/1   *     *       *       *       date>>Todos_Los_Minutos
```

3.- Guarda el archivo en tu directorio de trabajo con el nombre de: Mis_Tareas.

3.- Programar una tarea periódicamente.

3.4.- Ejecutar la tarea programada

En el apartado anterior hemos diseñado la tarea a realizar, pero no la hemos activado. Para ello utilizaremos el comando crontab.

$Crontab [opciones] archivo

$crontab archivo: programamos la tarea.

$crontab -l : Visualiza las tareas programadas.

$crontab -r: Elimina las tareas programadas.

3.- Programar una tarea periódicamente.

3.4.- Ejecutar la tarea programada

Root:

Las tareas periódicas se guardan en: /var/spool/cron/crontabs . Un archivo por usuario.

#crontab –l –u usuario → Ver las tareas periódicas del usuario usuario

#crontab –r –u usuario →Borra las tareas periódicas del usuario usuario.

3.- Programar una tarea periódicamente.

3.4.- Ejecutar la tarea programada

Ejercicio:

0.- # echo " " >/var/log/syslog → No es recomendable. ¿Sabes para qué lo hacemos?

1.- Activamos la tarea programada: $crontab mis_tareas

2.- Visualizar las tareas programadas:$crontab -l

3.- Esperar un par de minutos. Ve visualizando el fichero todos_los_minutos.

4.- Visualiza el fichero /var/log/syslog (¿Localizas la tarea lanzada?)

5.- Cancelar las tareas programadas: $crontab –r

6.- Visualiza las tareas programadas:$crontab -l

3.- Programar una tarea periódicamente.

3.4.- Ejecutar la tarea programada

Nota Importante:

Cada vez que ejecutemos el comando crontab, las tareas que estuviesen programadas previamente se verán sustituidas por las que contenga el nuevo archivo; por ello, si deseamos seguir conservando las anteriores, lo que tenemos que hacer es añadir la nueva tarea al archivo que contenga las que ya están programadas y ejecutar de nuevo el comando crontab con dicho archivo.

3.- Programar una tarea periódicamente.

3.4.- Ejecutar la tarea programada

Ejercicio:

1.- Crea en tu directorio de trabajo un archivo llamado ejercicio1.

2.- Realiza un script llamado copia que realice una copia del archivo ejercicio1, a un archivo llamado ejercicio1-Fecha_actual en el mismo directorio de trabajo.

```
#!/bin/bash
cp ejercicio1 ejercicio1-$(date +%d-%m-%Y)
```

3.- Darle los permisos de ejecución al script. Comprueba si funciona: $./copia

3.- Programar una tarea periódicamente.

3.4.- Ejecutar la tarea programada

Ejercicio:

4.- Programar una tarea (Mis_Tareas) para que este script se ejecute cada día a las HHMM

```
# m h  dom mon dow   command
11 20 * * * /home/usuario/copia
```

5.- Comprueba si funciona.

6.- Comprueba /var/log/syslog.

7.- Cancela la tarea. $crontab -r

4.- Programar tareas como root.

El administrador o el root podría programar sus tareas de la misma forma que un usuario normal, pero ya existe un archivo predefinido para ello.

/etc/crontab.

Vamos a visualizarlo:

```
root@debian-pc200:/home/usuario# cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

4.- Programar tareas como root.

Se añade una utilidad: run-parts --report directorio

Esta utilidad se encarga de ejecutar todos los scripts (procesos) que se encuentren almacenados en el directorio indicado.

Por ejemplo:

17 * * * * root cd / && run-parts --report /etc/cron.hourly

En el minuto 17, se va a acceder a raíz y se va ejecutar todos los procesos que hay dentro del directorio /etc/cron.hourly.

4.- Programar tareas como root.

```
usuario@debian1xx:/etc$ ls cron
cron.d/       cron.hourly/  crontab
cron.daily/   cron.monthly/ cron.weekly/
usuario@debian1xx:/etc$ ls cron
cron.d/       cron.hourly/  crontab
cron.daily/   cron.monthly/ cron.weekly/
usuario@debian1xx:/etc$ ls cron.hourly/
usuario@debian1xx:/etc$ ls cron.daily/
apt-compat  bsdmainutils  dpkg  exim4-base  logrotate  man-db  passwd
usuario@debian1xx:/etc$ ls cron.monthly/
```

```
-rwxr-xr-x 1 root root 1474 sep 13  2017 apt-compat
```

Si el administrador quiere programar una tarea que se ejecute cada hora, bastará con copiar el script al directorio /etc/cron.hourly.

NOTA IMPORTANTE: Los scripts dentro de los directorios, no pueden tener extensión y deben tener permisos de ejecución (755).

5.- Anacron

Introducción:

1.- ¿Qué pasará si diseñamos una tarea para una hora determinada y a esa hora está apagado el ordenador?

Usuario → at HH:MM
Echo “Hola a todos”|Wall

2.- ¿Qué pasará si programamos una tarea periódica y a esa hora está apagado el ordenador?

Usuario → */2 * * * * echo “Soy el pensado”
Crontab Mis_Tareas.

5.- Anacron

Introducción:

1.- ¿Qué pasará si diseñamos una tarea para una hora determinada y a esa hora está apagado el ordenador?

Usuario → at HH:MM
Echo “Hola a todos”|Wall

→ Sol: No se ejecutará

2.- ¿Qué pasará si programamos una tarea periódica y a esa hora está apagado el ordenador?

Usuario → */2 * * * * echo “Soy el pensado”

→ Sol: Cuando se arranque, se ejecutará la próxima vez que le toque.

5.- Anacron

5.1.-¿Qué es el anacron?

Si la tarea es importante, y no puede esperar hasta la próxima vez que le toque (Por ejemplo, la próxima semana) ¿Qué podemos hacer?

Este problema se soluciona con anacron, que nos permite ejecutar las tareas periódicas pendientes al iniciar el sistema.

Con el anacron:

podemos indicar cada cuánto tiempo se realizan ciertas tareas y se encarga de llevarlas a cabo.

5.- Anacron

5.1.-¿Qué es el anacron?

¿Sustituye anacron a cron?

No. Con anacron no se pueden programar tareas en intervalos menores a días, mientras que con cron se pueden planificar tareas a ser ejecutadas en horas o minutos. Por otro lado, anacron no ejecuta tareas en tiempo específicos como cron hace.

Es una herramienta complementaria, no sustituye al cron.

¿Cuándo se ejecuta anacron?

A diferencia de cron, anacron no es un demonio, es decir, no está corriendo todo el tiempo. De hecho solo corre a través de scripts de inicio del sistema o a través de tareas programadas de cron.

5.- Anacron

5.1.-¿Qué es el anacron?

Conclusiones:

Las tareas del anacron se examinan y en su caso se ejecutan:

1.- Cuando el ordenador se reinicia.

2.- Y Todos los días, a las 7:30. Porque existe un demonio cron que lo hace:

/etc/cron.d/anacron

```
root@debian2xx:/etc/cron.d# cat anacron
# /etc/cron.d/anacron: crontab entries for the anacron package

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

30 7    * * *   root    test -x /etc/init.d/anacron && /usr/sbin/invoke-rc.d ana
cron start >/dev/null
root@debian2xx:/etc/cron.d# _
```

5.- Anacron

5.1.-¿Qué es el anacron?

Conclusiones:

Por lo tanto si tenemos una tarea en el anacron que se ejecuta diariamente, se va a ejecutar aunque el ordenador nunca se apague.

5.- Anacron

Ejercicio:

1.- Comprueba si está instalado anacron.

2.- Si no lo está, instálalo.

5.- Anacron

5.2.- Configuración de las tareas.

El fichero de configuración es: /etc/anacrontab

```
root@debian-pc200:/etc/cron.daily# cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# These replace cron's entries
1       5       cron.daily      nice run-parts --report /etc/cron.daily
7       10      cron.weekly     nice run-parts --report /etc/cron.weekly
@monthly        15      cron.monthly nice run-parts --report /etc/cron.monthly
```

5.- Anacron

5.2.- Configuración de las tareas.

El fichero de configuración es: /etc/anacrontab

Veamos lo que significa cada campo.

Período	Retraso	Identificador	Comando
frecuencia en días con la que se ejecuta el comando.	Tiempo de retraso en minutos.	Descripción de la tarea. Debe ser único.	Comando que debe ejecutar.

Aquí podemos especificar tareas que se ejecutan cada día, cada semana o cada mes, pero no se puede especificar el instante exacto.

Tomemos como ejemplo la segunda línea.

```
7       10      cron.weekly     nice run-parts --report /etc/cron.weekly
```

Esta indica que la próxima vez que arranque anacron se revisará si han pasado 7 días desde la última ejecución. De ser así se volverán a ejecutar los scripts del directorio /etc/cron.weekly una vez que hayan pasado 10 minutos después de la ejecución de anacron.

Cada 7 días se ejecuta esos scripts.

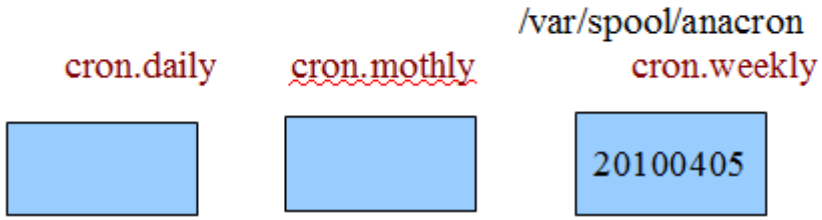
5.- Anacron

5.2.- Configuración de las tareas.

El fichero de configuración es: /etc/anacrontab

¿Cómo sabe anacron que se ha ejecutado la tarea?

Anacron creará un archivo en el directorio /var/spool/anacron para cada una de las tareas con su identificador. En cada uno de los ficheros se guarda la fecha de la última ejecución de la tarea por parte de anacron.



Esto indica que el día 05/04/2010 se ejecutaron los scripts del directorio /etc/cron.weekly, que es el comando programado en la segunda línea del archivo /etc/anacrontab de nuestro ejemplo.

Si se observa que lleva 8 días o más desde el último lanzamiento, lanzará dicha tarea.

5.2.- Configuración de las tareas.

Ejercicio: Tarea sólo en el /etc/anacrontab

1.- Diseña una tarea a la que llamaremos cron.prueba, que muestre el mensaje “Buenos días.”

2.- Esta tarea se ejecutará cada día a los tres minutos de encenderse el ordenador.

Deberás configurar sólo /etc/anacrontab

5.- Anacron

5.2.- Configuración de las tareas.

Ejercicio: Tarea sólo en el /etc/anacrontab

```
# These replace cron's entries
1       5       cron.daily      run-parts --report /etc/cron.daily
7       10      cron.weekly     run-parts --report /etc/cron.weekly
@monthly        15      cron.monthly    run-parts --report /etc/cron.monthly
1       3       cron.prueba     echo "Buenos días"|wall
```

Vemos que en /var/spool/anacron no se ha creado ninguna entrada, dado que nunca se ha ejecutado.

```
root@debian2xx:/var/spool/anacron# ls
cron.daily  cron.monthly  cron.weekly
root@debian2xx:/var/spool/anacron# _
```

5.2.- Configuración de las tareas.

Ejercicio: Tarea sólo en el /etc/anacrontab

Apagamos el ordenador y esperamos tres minutos

Observamos:
1.- Que se ha creado la entrada en su interior, pero no contiene ningún dato. Lo que significa que nunca se ha lanzado.

```
root@debian2xx:/var/spool/anacron# ls
cron.daily  cron.monthly  cron.prueba  cron.weekly
root@debian2xx:/var/spool/anacron# cat cron.prueba
root@debian2xx:/var/spool/anacron# _
```

2.- Esperamos tres minutos.

```
Broadcast message from root@debian2xx (somewhere) (Tue Mar 28 07:27:02 2017):

Buenos d\303\255as
```

5.2.- Configuración de las tareas.

Ejercicio: Tarea sólo en el /etc/anacrontab

Observamos:

3.- Ahora sí está rellena la entrada.

```
root@debian2xx:/var/spool/anacron# cat cron.prueba
20170328
```

5.2.- Configuración de las tareas.

Ejercicio:

1.- Diseña una tarea a la que llamaremos cron.saludos, que muestre diariamente a las HH:MM el mensaje “Qué tengas un buen día”.

2.- Si el ordenador está apagado a esa hora, se debe ejecutar la tarea la próxima vez que se inicie, transcurrido tres minutos.

Deberás configurar /etc/crontab y /etc/anacrontab

5.- Anacron

5.2.- Configuración de las tareas.

Ejercicio:

/etc/crontab

```
#Una nueva tarea
17 13    * * *    root echo "Buenos días" |wall
```

/etc/anacrontab

```
#Mis tareas
0 3 saludos echo "Buenos días" |wall
```

5.- Anacron

5.2.- Configuración de las tareas.

Ejercicio:

3.- Apagamos el ordenador para que no se realice la tarea.

4.- Lo volvemos a encender y transcurrido 3 minutos se debe lanzar la tarea.

```
usuario@servidor200:~$ w
 13:25:03 up 6 min,  1 user,  load average: 0,00, 0,04, 0,05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
usuario  tty1                     13:19    4.00s  0.29s  0.01s w
usuario@servidor200:~$ _
```

```
root@servidor200:/home/usuario# cat /var/log/syslog_
```

```
Mar  5 13:23:08 servidor200 anacron[378]: Normal exit (1 job run)
```

5.- Anacron

Resumen:

Ahora es el momento de realizar un resumen de lo que hemos visto.

5.- Anacron

Resumen:

CRON	ANACRON
Es un demonio (Daemon)	No es un demonio (Daemon)
Nos permite ejecutar tareas cada minuto	Solo permite ejecutar tareas diarias
Ideal para servidores (24X7)	Apropiado para equipos de escritorio o portátiles
Puede ser usado por usuarios root o usuarios normales	Solo puede ser usado por usuarios root
No ejecuta una tarea con el equipo apagado	Si el equipo está apagado ejecutara la tarea la próxima vez que este este activo

Sugerencias/mejoras del tema

Sugerencias /mejoras del tema

Referencias

- Los logotipos del Dpto de informática han sido diseñados por Manuel Guareño.
- Algunas de las imágenes proceden de Internet y pueden tener copyright.