

UD 04_04 – GNU/Linux

Control de tareas en ejecución. Instalación de software. Montar y desmontar unidades. Cron.

Contenido

1	CREAR NUESTRO PROPIO COMANDO	2
1.1	EJERCICIOS.....	3
2	CONTROL DE TAREAS EN EJECUCIÓN	3
3	INSTALACIÓN DE SOFTWARE	8
3.1	El centro de software de Ubuntu	8
3.2	dpkg.....	8
3.3	Dependencias y repositorios.....	10
3.4	Gestores de paquetes	10
3.4.1	Introducción	10
3.4.2	apt.....	11
3.4.3	aptitude	12
3.5	sources.list.....	13
3.6	Si la actualización o descarga de software es lenta.....	15
3.1	EJERCICIOS.....	16
4	AMPLIACIÓN: MONTAR Y DESMONTAR UNIDADES EN GNU/LINUX	17
4.1	EJERCICIO.....	17
5	PROCESOS AUTOMÁTICOS: CRON	18
5.1	EJERCICIO.....	21

1 CREAR NUESTRO PROPIO COMANDO

Es posible que, tras la creación de un script, nos interese poder utilizarlo como si fuera uno de los comandos de nuestro sistema y que sea accesible simplemente con su nombre sin importar nuestra ubicación en el árbol de directorios.

Para crear nuestro propio comando debemos:

1. Escribir el código con nuestra herramienta utilizando el lenguaje elegido (por ejemplo C o un script para Bash).
2. Compilar el código fuente para generar el ejecutable. Si es C o C++ se puede usar el compilador gcc por ejemplo. Si se trata de un script para Bash, para hacerlo ejecutable modificamos los permisos con `chmod +x nombre_script`
3. Debemos ubicar el fichero ejecutable en una ruta incluida dentro de la variable de entorno `$PATH` (podemos visualizarlas con `echo $PATH`). Una vez hecho esto, simplemente escribiendo el nombre del ejecutable se ejecutará nuestro comando y no hará falta escribir su ruta absoluta.

EJEMPLO:

Vamos a crear el comando `hola` que saque por pantalla el texto "Hola, mundo" a partir de un script. No hará falta conocer la ruta absoluta del comando para ejecutarlo.

- Escribimos un script. Vamos a utilizar por ejemplo el editor nano (EL SIGNO DEL DOLAR AL PRINCIPIO DE ESTAS LÍNEAS SIMPLEMENTE ES EL PROMPT, NO LO TIENES QUE COPIAR):

```
$ nano hola
```

- Escribimos el contenido y guardamos:

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

```
echo "Hola, mundo"
```

- Modificamos los permisos del fichero creado:

```
$ chmod +x hola
```

- Prueba a ejecutar tu comando escribiendo simplemente:

```
$ hola
```

- Te habrá dado un error diciendo que no encuentra la orden.

- Visualizamos el contenido de la variable `$PATH`:

```
$ echo $PATH
```

- Veremos que no está la ruta donde se encuentra nuestro script.
- Visualiza el contenido del fichero `.profile` (recuerda que es uno de los fichero de inicialización de bash). Comprueba que incluye estas líneas:

```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

¿Qué quiere decir lo anterior? Que si dentro de tu directorio personal (la ruta se guarda en la variable de entorno \$HOME cuando inicias sesión) existe una carpeta llamada bin, esta se incluirá en el PATH. Vamos a hacer lo necesario para poder utilizar esa carpeta para nuestros scripts:

- Como seguramente esa carpeta no existe, la creamos:

```
$ mkdir ~/bin
```

- Ahora movemos nuestro script a la carpeta ~/bin. Por ejemplo, si nuestro script estaba en ~:

```
$ mv ~/hola ~/bin
```

- Recargamos .bashrc:

```
$ source ~/.bashrc
```

- Ya podemos ejecutar el comando simplemente escribiendo su nombre.

1.1 EJERCICIOS

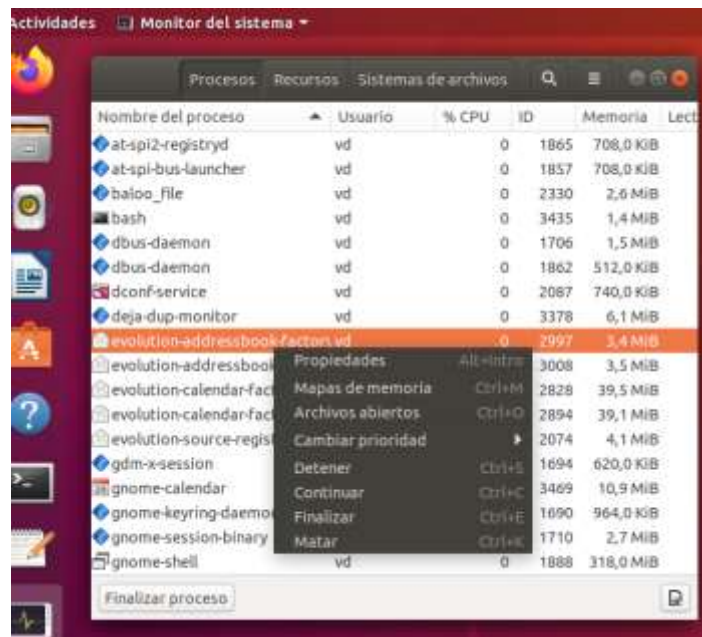
- Crea un comando con alguno de los scripts del tema anterior.
- Otra opción, en lugar de poner el script ~/bin o en uno de los directorios que por defecto están en la variable PATH, podría interesarnos añadir uno de nuestros directorios al PATH. Modifica el fichero .bashrc para modificar el PATH y añadir una carpeta tuya donde guardas los scripts. Prueba con: `export PATH=$PATH:~/tu_ruta`

2 CONTROL DE TAREAS EN EJECUCIÓN

Linux es responsable de distintas actividades relacionadas con la administración de procesos y tareas, entre las que se encuentra la creación de procesos, la terminación de procesos, la ejecución de procesos en primer y segundo plano, la suspensión de procesos y la conmutación de procesos de primer plano a segundo plano y viceversa.

Existen varias herramientas para ver los procesos en ejecución:

Desde el entorno gráfico, podemos lanzar el “Monitor del sistema”:



Desde ahí, con el botón derecho sobre un proceso, podremos elegir si, por ejemplo, detenemos, continuamos o finalizamos un proceso.

Desde la terminal, tenemos varios comandos posibles:

ps (process status)¹

Lista los procesos con su PID, datos de usuario, tiempo, identificador del proceso y línea de comandos usada.

```
$ ps
  PID TTY          TIME CMD
 6368 pts/0    00:00:00 bash
 7441 pts/0    00:00:00 ps
```

Sin opciones, ps sólo muestra los procesos lanzados desde el terminal actual y con el mismo EUID (*Effective User ID*) que el usuario que lo lanzó

Algunas opciones:

- -e o ax: muestra todos los procesos
- -u (o U o --user) *usuario*: muestra los procesos de un usuario
- u: salida en formato usuario
- j: salida en formato *job* (muestra PID, PPID, etc.)
- -f o l: salida en formato largo
- f: muestra un árbol con la jerarquía de procesos
- k (o --sort) *campo*: ordena la salida por algún campo (p.e. ps uxak rss)
- -o (o o o --format) *formato*: permite definir el formato de salida ps -o ruser,pid,comm=Comando

¹ http://persoal.citius.usc.es/tf.pena/ASR/Tema_3html/node2.html

Ejemplo:

```
$ps aux
```

Tras la ejecución, obtendrás varias columnas cuyo significado es:

- %CPU y %MEM: porcentajes de uso de CPU y memoria
- VSZ: memoria virtual del proceso, en KBytes
- RSS: tamaño de la memoria residente (*resident set size*) en KBytes
- STAT: estado del proceso (R ejecutándose (running), T detenido, N baja prioridad...)

ps tree

Muestra el árbol de procesos, similar a ps f

top

Da una lista de procesos actualizada a intervalos. ps da una versión estática de los procesos.

- en la cabecera nos muestra un resumen del estado del sistema:
 - hora actual, tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos
 - número total de tareas y resumen por estado
 - estado de ocupación de la CPU y la memoria
- por defecto, los procesos se muestran ordenados por porcentaje de uso de CPU (los más costosos arriba)
- pulsando h mientras se ejecuta top, obtenemos una lista de comandos interactivos
- para salir, q

Como usuario de GNU/Linux se puede solicitar el control de procesos y tareas empleando las órdenes del shell.

Cuando se escribe una orden y se pulsa <Intro>, el shell ejecuta la orden y retorna, mostrando el indicador del shell. Mientras se ejecuta la orden no tenemos acceso al shell y por tanto, no se puede ejecutar ninguna orden mientras no finalice la orden actual y retorne el shell. Cuando las órdenes se ejecutan de este modo, decimos que se ejecutan **en primer plano**. Más técnicamente, cuando se ejecuta una orden en primer plano, la orden toma el control del teclado y de la pantalla.

En ciertas ocasiones, es preciso ejecutar una orden de GNU/Linux (o cualquier programa) que requiera mucho tiempo para terminar; mientras se ejecuta la orden, necesitamos realizar otras tareas. Esto no se puede hacer si la orden se ejecuta en primer plano porque el shell no retorna mientras no finalice la orden. GNU/Linux nos permite ejecutar la orden de tal modo que (mientras se

ejecuta la orden) se recupera el indicador del shell y es posible seguir realizando otras tareas. Esta posibilidad se llama ejecutar la orden **en segundo plano**. Para ejecutar la orden en segundo plano, al final de la orden se pone un *ampersand* (&).

Si una tarea se encuentra en primer plano es posible suspender el proceso pulsando las teclas <ctrl-z>. Ese proceso suspendido se puede reanudar en primer o segundo plano según los siguientes comandos:

Sintaxis: `fg [%idtarea]`

Reanuda la ejecución de un proceso cuyo número de tarea es “idtarea” en primer plano, o llevar procesos de segundo plano a primer plano. Si no se indica ningún parámetro, se lleva a primer plano la tarea con “idtarea” más alta, es decir, la última tarea lanzada.

Sintaxis: `bg [%idtarea]`

Reanuda la ejecución de un proceso o tarea suspendida cuyo número de tarea se indique en “idtarea”. Si no se indica ningún parámetro, se lleva a segundo plano la tarea con “idtarea” más alta, es decir, la última tarea lanzada.

Para saber qué procesos tenemos ahora suspendidos o en segundo plano se utiliza el comando `jobs`. Si no se especifica un “idtarea” muestra un listado de todos los procesos.

Sintaxis: `jobs [opción] [%idtarea]`

Opción:

-l muestra también el PID de la tarea.

Ejemplos:

Nota: el comando `sleep` que se utiliza a continuación sirve para introducir un retardo de tiempo

```
$ sleep 25m ; date
<ctrl-z>
[1]+ Stopped sleep 25m
```

```
$ jobs
[1]+ Stopped sleep 25m
```

```
$ (sleep 25m ; cp /etc/passwd $HOME) &
[2] 13586
```

```
$ fg %2
$ jobs
[1]+ Stopped sleep 25m
```

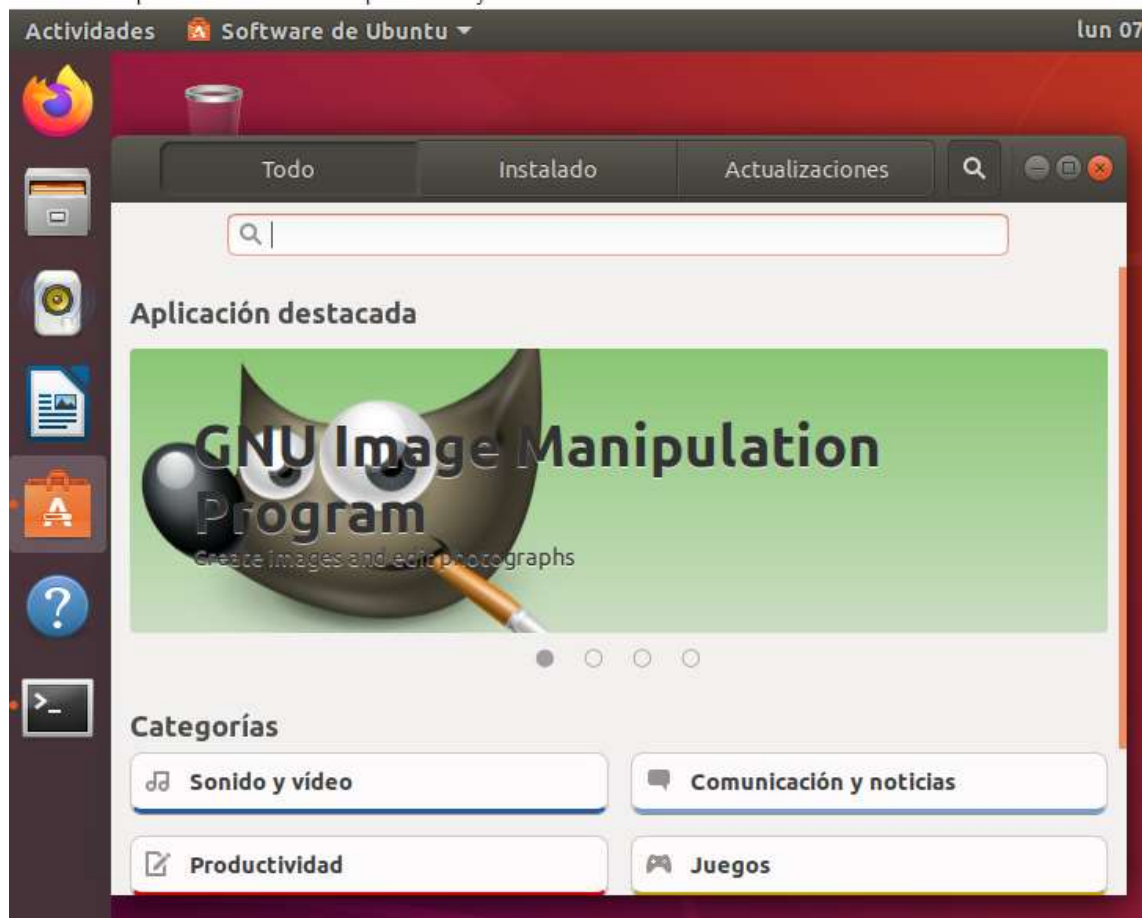
```
$ bg %1
[1] sleep 25m &
```

EJERCICIOS	Solución
<p>Hacer que dentro de 10 minutos se liste el contenido del directorio /etc.</p> <p>¿El proceso queda en primer plano o en segundo plano?</p>	<p><code>sleep 10m; ls -la /etc</code></p> <p>Queda en primer plano, con lo que la línea de comandos queda bloqueada a la espera de que el comando anterior termine.</p> <p><code>(sleep 10m; ls -la /etc) &</code> → si lo ejecutamos, quedaría en 2º plano.</p>
<p>Pasar el proceso anterior a primer plano si está en segundo plano o a segundo si está en primero.</p>	<p>Estando en primer plano:</p> <p><code>ctrl + z</code> → el proceso queda suspendido</p> <p>Necesitamos conocer el número de tarea, lo averiguamos con <code>jobs</code>.</p> <p><code>bg %2</code> (tras el % hay que poner el número de tarea. Sería el número que aparece entre corchetes al ejecutar <code>jobs</code>).</p>
<p>Hacer lo contrario al ejercicio anterior.</p>	<p>Si está en segundo plano y lo queremos pasar a 1º:</p> <p><code>fg %2</code></p>
<p>Con el proceso anterior en segundo plano, trata de suspenderlo.</p> <p>¿Tienes algún problema? En caso afirmativo, haz lo necesario para que el proceso quede suspendido.</p>	<p>Si está en 2º plano, no funciona pulsar <code>ctrl+z</code>.</p> <p>Para suspenderlo, podemos traerlo a primer plano con <code>fg</code> y luego suspenderlo con <code>ctrl+z</code>.</p>
<p>Muestra los procesos del shell actual.</p>	<p><code>ps</code></p>
<p>Muestra todos los procesos del sistema.</p>	<p><code>ps -ely</code></p> <p><code>ps aux</code></p>
<p>Elimina el proceso suspendido anterior (tras averiguar el PID con <code>jobs</code>, utiliza el comando <code>kill</code>).</p>	<p>Para ello necesitamos conocer el PID, con <code>jobs -l</code>, por ejemplo.</p> <p><code>kill -9 PID</code> (sustituir PID por el nº correspondiente)</p>

3 INSTALACIÓN DE SOFTWARE

3.1 El centro de software de Ubuntu

El entorno gráfico nos permite instalar software fácilmente en Ubuntu. Los programas se buscarán en los repositorios que se han definido (veremos qué son los repositorios más adelante en este documento).



Para instalar programas que no están en esos repositorios, o aún estando, para poder trabajar desde la línea de comandos o incluir acciones relacionadas con el mantenimiento del software en un script, por ejemplo, disponemos de otras herramientas que vemos a continuación.

3.2 dpkg

dpkg es el programa base para manejar paquetes .deb en el sistema, permite instalar o analizar el contenido de un paquete. Pero este programa sólo tiene una visión parcial: sabe lo que está instalado en el sistema y lo que se le pasa por línea de comandos, pero no sabe de otros paquetes disponibles, por lo que fallará si no se satisface una dependencia. En cambio, herramientas como apt (la veremos a continuación) crearán una lista de dependencias para instalar todo tan automáticamente como sea posible.

Para instalar un paquete .deb que hayamos descargado, bastará con ejecutar:

```
dpkg -i nombre_paquete.deb
```

Para eliminar un paquete, utilizaremos la opción -r seguida del nombre del paquete. Esta eliminación, sin embargo, no es completa: se mantendrán todos los archivos de configuración, scripts, archivos de registros (registros de sistema) y otros datos de usuarios que gestiona el paquete. Para eliminar completamente todo lo asociado con un paquete, utilizaremos la opción -P o -purge seguida del nombre del paquete.

```
dpkg -P nombre_paquete
```

En el capítulo 5 del libro *The Debian's Administrator Handbook* (<https://debian-handbook.info/get/now/>) , encontraréis más detalles acerca del manejo de dpkg.

EJERCICIO




Puedes probar el uso de dpkg instalando, por ejemplo, Dropbox. Aquí puedes encontrar los paquetes:

https://www.dropbox.com/es_ES/install-linux

Al instalar con dpkg, ¿tienes problemas de dependencias?

Observa que también tienes la opción de “compilar desde código fuente”, por si quieres hacer una instalación en una distribución de Linux distinta a las recogidas ahí:

Instala el paquete que corresponda si quieres utilizar Dropbox en tu escritorio Linux. Si tu distribución no aparece en la lista, elige "Compilar a partir de código fuente".

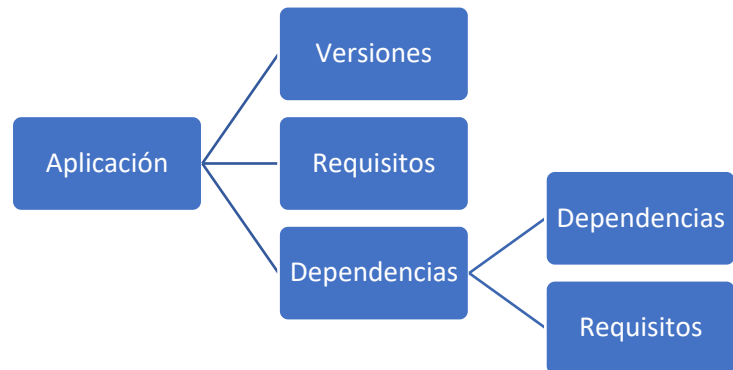
- ▶  Ubuntu 14.04 o superior (.deb) [64 bits](#) [32 bits](#)
- ▶  Fedora 21 o superior (.rpm) [64 bits](#) [32 bits](#)
- ▶  [Compilar desde código fuente](#)

Y en este artículo te muestran cómo hacer la instalación y algunas otras opciones de dpkg:

<https://www.cambiatealinux.com/dpkg-gesti%C3%B3n-de-paquetes-en-debian-ubuntu-y-derivados>

3.3 Dependencias y repositorios

Las aplicaciones que queramos instalar o mantener en nuestro sistema tendrán distintos requisitos (espacio en disco necesario, versión de la distribución de GNU/Linux que estemos utilizando) y distintas dependencias (otros programas o librerías necesarios para que la aplicación pueda funcionar). A su vez, las dependencias también tendrán requisitos y dependencias.



En GNU/Linux la solución a esto es el uso de los REPOSITARIOS. De forma simple, podríamos decir que un repositorio es el lugar donde se centraliza todo el software que existe para cada distribución de GNU/Linux, por lo que es necesario utilizar sólo los repositorios de la distribución con la que trabajemos. Con los repositorios, tendremos las versiones disponibles de las aplicaciones, su lista de dependencias y a su vez la lista de dependencias de estas dependencias, así como los requisitos.

Un repositorio puede estar por ejemplo en un dispositivo óptico como un DVD o en algún lugar en la red.

Hay distintos tipos de repositorios según las distribuciones. En Debian y derivados (como Ubuntu) se utilizan repositorios APT.

APT son las siglas de Advanced Packaging Tool) (herramienta avanzada de empaquetado)

3.4 Gestores de paquetes

3.4.1 Introducción

Además de los repositorios, cada distribución se caracteriza por su gestor de paquetes. Un gestor de paquetes es una herramienta que nos permite instalar paquetes, desinstalarlos, actualizarlos, resolver sus dependencias, etc., todo esto desde los repositorios que nombramos anteriormente. Cada gestor de paquetes se caracteriza por el formato de los paquetes que gestiona. Por ejemplo:

- **apt**: Advanced Packaging Tool, ó APT, es un sistema de gestión de paquetes creado por el proyecto Debian. Los paquetes tienen extensión `.deb`.
- **rpm** (RPM Package Manager, o RPM, originalmente llamado Red Hat Package Manager, pero se convirtió en acrónimo recursivo): Originalmente desarrollado por Red Hat para Red Hat Linux, en la actualidad muchas distribuciones GNU/Linux lo usan, dentro de las cuales las más destacadas son Fedora, Mandriva y openSUSE.

Para cada distribución de GNU/Linux que vayamos a utilizar tendremos que conocer su gestor de paquetes. Ejemplo de otros gestores de paquetes son:

- DNF en Fedora (antes YUM): <https://fedoraproject.org/wiki/DNF>
- zypper y rpm en openSUSE. En openSUSE además se dispone de una potente herramienta gráfica, YaST, que, entre otras muchas funciones, permite realizar la gestión de paquetes:



3.4.2 apt

Antes de hacer cualquier instalación o actualización es necesario realizar una actualización de la lista de paquetes que se encuentran en los repositorios. Para ello es necesario ejecutar como administrador:

```
# apt update
```

En versiones anteriores de Ubuntu, en lugar de `apt` el comando era `apt-get`. A partir de la versión 16.04 se pueden utilizar ambos comandos.

Una vez tengamos la lista de los últimos paquetes disponibles, podremos instalarlo mediante

```
# apt install nombre_del_paquete
```

Para actualizar todos los paquetes instalados en el sistema:

```
# apt upgrade
```

Para buscar paquetes en los repositorios:

```
# apt search nombre_del_paquete
```

(en este caso, en versiones anteriores estaba el comando `apt-cache`)

Para eliminar paquetes:

```
# apt remove nombre_del_paquete
```

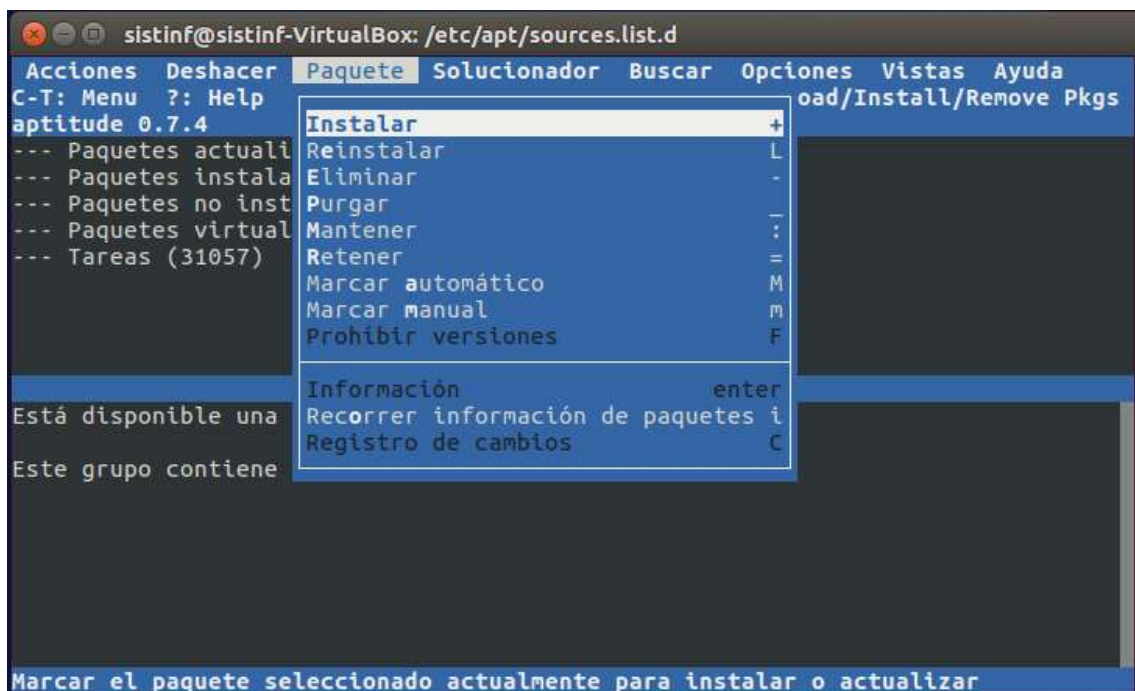
Para eliminar posibles restos de configuraciones, utilizar:

```
# apt purge nombre_del_paquete
```

3.4.3 aptitude

`aptitude` es una herramienta gráfica equivalente a `apt`. Para utilizarla, simplemente escribiremos en la línea de comandos:

`aptitude`



Podremos manejarla utilizando el ratón.

3.5 sources.list

Los repositorios se definen en el fichero `/etc/apt/sources.list` en distribuciones GNU/Linux derivadas de Debian GNU/Linux; ahí se listan las "fuentes" o "repositorios" disponibles de los paquetes de software candidatos a ser actualizados, instalados, eliminados, buscados, sujetos a comparación de versiones, etc.

(Prueba a listar el contenido del fichero anterior)

Los repositorios para aplicaciones que no están en el repositorio principal se pueden definir en el directorio `/etc/apt/sources.list.d/`. Aquí crearíamos un fichero de texto similar a `/etc/apt/sources.list`.

(Comprueba si tienes algún fichero en `/etc/apt/sources.list.d/`)

El **formato** para definir un repositorio dentro de los ficheros anteriores será el siguiente:

```
deb    ubicación    nombre_distribución    componentes
```

- `nombre_distribución` será el «nombre código» de la misma
- Los componentes (o secciones) que se desea activar (en un repositorio Debian típico: `main`, `restricted`, `universe`, `multiverse`) se explican más abajo.

Si definimos repositorios de código fuente:

```
deb-src    ubicación    nombre_distribución    componentes
```

En esta web puedes ver un generador de los repositorios de Ubuntu: <https://repogen.simplylinux.ch/> Recomendando probarlo para entender mejor las diferentes partes de la sintaxis expuesta arriba.

La herramienta `apt` administra el acceso a dichos paquetes, utilizando el fichero `sources.list`, para realizar las acciones previamente mencionadas.

Tras realizar alguna modificación en el `sources.list`, se debe ejecutar:

- Para actualizar lo realizado o la lista de paquetes de software:
`# apt update`
- Para actualizar la versión de uno o más paquetes:
`# apt upgrade`
- Para actualizar el sistema operativo completo:
`# apt dist-upgrade`

Ejemplos de configuración:

```
deb http://archive.ubuntu.com/ubuntu/ xenial main restricted universe multiverse
```

```
deb-src http://archive.ubuntu.com/ubuntu/ xenial main restricted universe  
multiverse
```

```
deb http://security.ubuntu.com/ubuntu/ xenial-security main restricted universe  
multiverse
```

```
deb-src http://security.ubuntu.com/ubuntu/ xenial-security main restricted  
universe multiverse
```

```
deb http://archive.ubuntu.com/ubuntu/ xenial-updates main restricted universe  
multiverse
```

```
deb-src http://archive.ubuntu.com/ubuntu/ xenial-updates main restricted  
universe multiverse
```

Los repositorios se clasifican en diferentes categorías:

- **Main:** contiene solamente los paquetes que cumplen los requisitos de la licencia de Ubuntu, y para los que hay soporte disponible por parte de su equipo. Éste está pensado para que incluya todo lo necesario para la mayoría de los sistemas GNU/Linux de uso general. Los paquetes de este componente poseen ayuda técnica garantizada y mejoras de seguridad oportunas.
- **Restricted:** contiene el software que está soportado por los desarrolladores de Ubuntu debido a su importancia, pero que no está disponible bajo ningún tipo de licencia libre para incluir en main. En este lugar se incluyen los paquetes tales como los controladores propietarios de algunas tarjetas gráficas, como, por ejemplo, los de Nvidia. El nivel de la ayuda es más limitado que para main, puesto que los desarrolladores pueden no tener acceso al código fuente.
- **Universe:** contiene una amplia gama de software libre, que no recibe apoyo por parte del equipo de Ubuntu. Esto permite que los usuarios instalen toda clase de programas en el sistema, pero los guarda en un lugar aparte de los paquetes soportados: main y restricted.
- **Multiverse:** paquetes sin soporte debido a que no cumplen los requisitos del software libre.
- **Commercial:** Contiene software comercial.

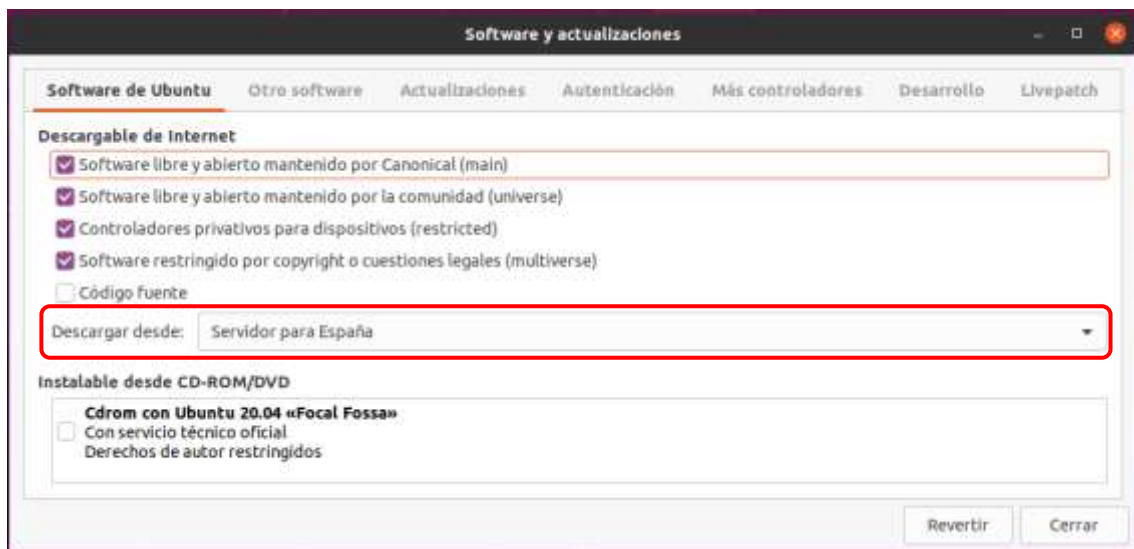
También es posible realizar modificaciones en los repositorios utilizando el comando `add-apt-repository`.

3.6 Si la actualización o descarga de software es lenta

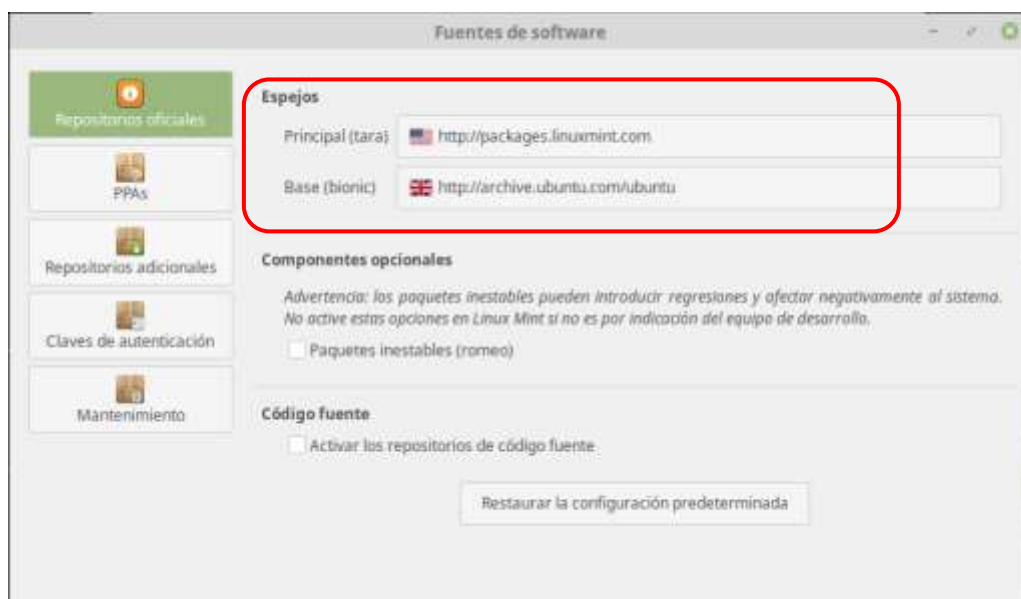
Si observamos que el proceso de hacer un update o una instalación es demasiado lento, puede ser que esto se solucione cambiando el servidor de descarga del repositorio que se está utilizando.

Para ver un listado de los servidores, probar cuál es más rápido y cambiarlo podemos buscar lo siguiente en Ubuntu y LinuxMint.

En Ubuntu 20.04, buscamos “Software y actualizaciones”



En LinuxMint 19, si buscamos “Fuentes de software”:



En ambos casos, en los desplegables anteriores aparecerán una serie de sitios y podremos seleccionar de entre los más rápidos.

3.1 EJERCICIOS

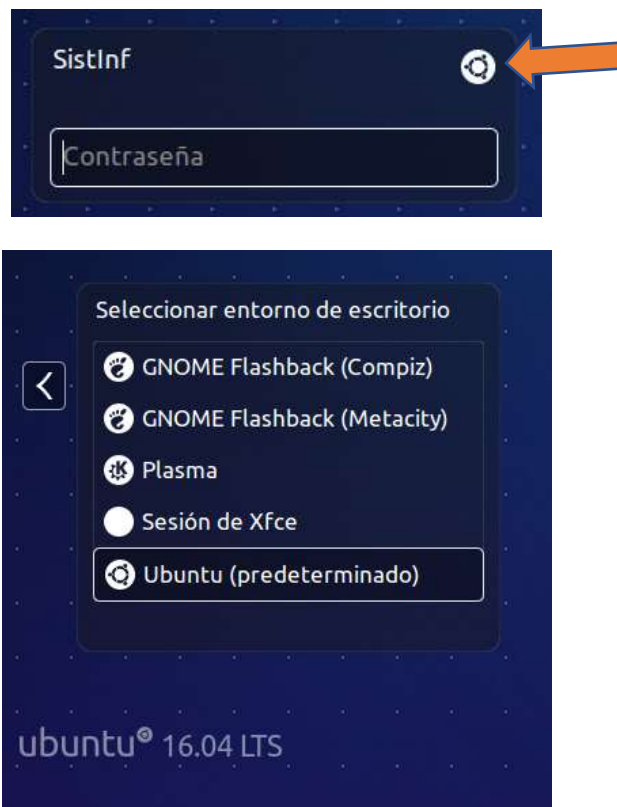
EJERCICIO 1

En este enlace <https://repogen.simplylinux.ch/> puedes generar la información que iría en el fichero sources.list en función de los repositorios seleccionados. Observa cómo varía la información contenida en cada una de las líneas en función de la selección que hagas.

EJERCICIO 2

Instala otro escritorio para tu Linux, prueba con xfce. Para ello, deberás, en primer lugar, actualizar la lista de paquetes de los repositorios. A continuación, puedes realizar una búsqueda para comprobar que el paquete está en los repositorios que tenemos listados (puede ser que el paquete no se llame exactamente xfce, puede incluir número de versión. Entre todo lo que te aparece, distingue el que debes elegir para instalar). Luego instala el paquete.

Para iniciar sesión con el nuevo escritorio, debes elegirlo antes de introducir el usuario y la contraseña. Para ello, por ejemplo en Ubuntu, pulsa el icono con el logo de Ubuntu que te aparecerá al lado del nombre del usuario.



4 AMPLIACIÓN: MONTAR Y DESMONTAR UNIDADES EN GNU/LINUX

En el siguiente artículo se explica cómo ver las unidades de almacenamiento en nuestro equipo (se habla de discos y particiones, pero puede tratarse, por ejemplo, de unidades USB) y cómo montarlas (es decir, hacer que estén accesibles) y desmontarlas manualmente. Estas acciones se hacen de forma automática, pero pueden darse situaciones en que sea necesario realizarlas de forma manual.

Para practicar lo que se indica, conecta un nuevo disco duro a tu máquina virtual y crea varias particiones en él.

<https://www.solvetic.com/tutoriales/article/3607-comandos-montar-desmontar-particiones-linux/>

A partir de ese artículo también puedes aprender cómo montar unidades ext4 en Windows y cómo clonar discos desde línea de comandos en Linux.

4.1 EJERCICIO

Prueba los comandos que se enseñan en el artículo adaptándolos a la situación que tienes en tu máquina virtual. Realiza capturas de pantalla de las acciones que realices.

5 PROCESOS AUTOMÁTICOS: CRON

El `cron` permite programar tareas, de forma que se ejecutarán en segundo plano los trabajos que se programen de manera automatizada. La ejecución se realizará a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el archivo `/etc/crontab`.

Cron sería el "equivalente" a Tareas Programadas o Programador de Tareas de Windows.

Cada línea del fichero `/etc/crontab` representa un trabajo, y está compuesto de una expresión propia de cron, seguida por un comando para ejecutarse.

EJERCICIO

Lista el contenido del fichero `/etc/crontab` y comprueba que entiendes sus partes siguiendo las notas que hay a continuación.

La siguiente imagen muestra un ejemplo de su contenido:

```
sistinf@sistinf-VirtualBox:~$ 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 )
#
```

La variable `SHELL` contiene el valor del shell por defecto que ejecutará la tarea, la variable `PATH` contiene las rutas en las que cron buscará el comando a ejecutar. A continuación, hay una línea (comentada) que contiene la siguiente información:

```
#      m      h      dom    mon    dow    user  command
```

Se trata de las cabeceras de los campos de cada una de las líneas que hay a debajo. Estas cabeceras significan:

m	minuto (0-59)
h	hora (0-23)
dom	día del mes (day of month) (1-31)
mon	mes (month) (1-12)
dow	día de la semana (day of week). 1=lunes, 2=martes,... 6=sábado y 0=domingo (para países que empiezan la semana en domingo) 7=domingo (para países que empiezan la semana en lunes)
user	usuario
command	comando a ejecutar

Vemos que los cinco primeros valores de cada línea sirven para indicar el momento de ejecución del comando.

Cada usuario puede tener su propio fichero crontab, que se guarda dentro de `/var/spool/cron/crontabs/<usuario>`. Para editarlo y realizar modificaciones en él, ejecutaremos:

```
crontab -e -u usuario
```

A continuación, tenemos un ejemplo de contenido de este fichero. Observa que en este caso no hay un campo referido al usuario.

```
# Ejecución de una copia de seguridad semanal del directorio  
# /home a las 5 a.m. cada semana:
```

```
0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
```

```
#m h dom mon dow command
```

Otros aspectos sobre la sintaxis del fichero `/etc/crontab`:

- Todo lo que se encuentre después del carácter almohadilla, #, no será ejecutado por cron, será un comentario.
- Para especificar todos los valores posibles de una variable se utiliza un asterisco (*).
- Para expresar un rango, separar por (-)
- Valor separado por comas (,) indica un valor válido para cualquiera de los valores indicados.
- rango/número indica que se debe saltar ese número en el rango

Ejemplos:

- 1,2,5,9
- 0-4,8-12
- 0-23/2 → 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22
- */2 → Por ejemplo, cada dos horas

Ejemplos de líneas del fichero crontab

30 10 * * * 1 /usr/bin/who >> /home/quien.tex

Ejecuta la orden who todos los lunes a las 10:30 y guarda la salida en el archivo quien.tex

0,30 * * * 1 /usr/bin/who >> /home/quien.tex

Ejecuta la orden who todos los lunes cada media hora y guarda la salida en el archivo quien.tex

0,15,30,45 * * * * /usr/bin/who >> /home/quien.tex

o

***/15 * * * * /usr/bin/who >> /home/quien.tex**

Si queremos que se ejecute cada 15 minutos

30 21 * * * cd /media/sda7/dexter/distributions/isos;wget http://hola.com/loquesea.html

Cómo pasarle más de un comando al cron y de paso cómo puede programarse una descarga

30 21 * * 6 /sbin/shutdown -h now

Este otro es para programar el apagado del equipo. En este caso todos los sábados a las 21.30

La sintaxis del comando crontab es la siguiente:

`crontab [-u user] archivo`

`crontab [-u user] [-l | -r | -e]`

`crontab archivo` reemplaza el archivo de configuración actual por el archivo especificado

`crontab -u usuario` accede al fichero de un usuario. Debe ir con alguno de los parámetros siguientes -e, -l o -r.

`crontab -e` edita el fichero de configuración del usuario

`crontab -l` lista el fichero de configuración del usuario

`crontab -r` borra el fichero de configuración del usuario.

En estos tres últimos casos, si no se indica usuario, se tratará del usuario que tiene iniciada sesión.

Encontramos entradas del cron en `/etc/cron.d` y otras en `/etc/cron.{daily,hourly,monthly,weekly}`.

Si el fichero `/etc/cron.allow` existe, el usuario debe aparecer listado en él (un usuario en cada línea) para poder utilizar el comando `crontab`. Si el fichero `/etc/cron.allow` no existe, pero sí existe `/etc/cron.deny`, entonces el usuario NO debe aparecer listado en este fichero si se quiere que pueda utilizar el comando. Si los dos ficheros existen, el que se considera es `/etc/cron.allow`.

Cada vez que se ejecuta el `crontab`, se envía un mensaje al usuario que aparece en la variable de entorno `MAILTO`, si está habilitada, indicándole la tarea realizada.

5.1 EJERCICIO

Crea una carpeta en tu directorio home con el nombre papelera. Copia o crea (por ejemplo con `touch`) archivos dentro de esta carpeta, da igual su contenido.

Para poder ver de forma rápida el funcionamiento del cron, programa una tarea de forma que cada minuto se borre el contenido de la carpeta `/home/<tu_usuario>/papelera`.

¿Cómo harías para que se ejecutara la misma tarea todos los lunes a las 8.00h?

El comando `anacron` complementa a cron. Este programa comprueba si alguna tarea programada en el cron no ha sido realizada debido a que el sistema haya estado apagado. En ese caso, `anacron` lo detecta y realiza las acciones adecuadas. Puedes encontrar más información sobre este comando y su fichero asociado en el siguiente enlace: <https://geekytheory.com/programacion-de-tareas-asincronas-en-linux-con-anacron>