nolbox
versión v0.2

buscar GO

Usando portage correctamente

El sistema portage de gentoo es un recurso excelente cuando se usa apropiadamente; el uso incorrecto, puede desembocar en un sistema desordenado, con paquetes y ficheros descontrolados, y un fichero **world** incorrecto. Siguiendo esta guía el usuario tendrá un sistema limpio y ordenado.

emergiendo paquetes:
emerge foo

No se debe usar ACCEPT_KEYWORDS="~x86" para instalar paquetes **masked**, porque, entre otras cosas, instalará también las dependencias en su versión inestable.

El método adecuado es: Intentar instalar el paquete, si portage nos comunica que el ebuild es inestable, tomar la solución apropiada (abajo).
Si una vez resuelto, al instalar el paquete, alguna dependencia se queja, por estar también **masked** se repite la solución apropiada (abajo).
Si, esto puede parecer un dolor, frente a los rápidos **USE="blah" emerge foo** o **ACCEPT_KEYWORDS="~x86" emerge foo** pero es la forma que portage tiene de llevar el control de paquetes, incluso de los inestables, correctamente.

ATENCIÓN: *emerger* directamente un ebuild: **emerge /path/to/foo.ebuild** puede causar problemas y **NO** debe ser usado; en ocasiones este método de instalación, rechaza añadir el paquete al fichero world, además usando este método portage no nos notificará de las nuevas versiones del paquete aun cuando estas pudieran contener importantes correcciones de seguridad.

MANTENIENDO PAQUETES:

Algunas veces, cuando ejecutamos **emerge -u world** portage quiere retroceder en la versión de algún paquete (por ejemplo, si está **masked**)
La solución es añadir el nombre completo del paquete, seguido de **~x86** en el archivo **/etc/portage/package.keywords**.
Por ejemplo, para que portage reconozca que queremos tener la versión inestable de gaim en nuestro sistema:

echo net-im/gaim ~x86 >> /etc/portage/package.keywords

Si haces esto para todos los paquetes inestables, quedarán correctamente bajo el control de portage, entonces no tratará de volver a la última version estable.

También podemos determinar versiones espécificas, de forma que portage no actualizará ni retrocederá las versiones.

echo =app-misc/foo-version ~x86>> /etc/portage/package.keywords

La línea anterior, permite que una versión específica sea mantenida. Tambien podemos indicar Pero si queremos que además de la versión específica, las siguientes versiones a pesar de estar **masked**, también puedan ser instaladas bajo el control de portage.
Haciendo así posible, instalar un paquete **masked** y además actualizarlo a medida que salgan nuevas versiones **masked**:

echo ~app-misc/foo-versión ~x86>> /etc/portage/package.keywords
(donde **versión** es la versión sin -rX (ejemplo, en foo-1.2-r4 sobraría -r4).

NOTA: Es mucho más sencillo llevar un control de los paquetes, si la lista está ordenada alfabéticamente.
ATENCIÓN: Usar **emerge -U world** podría causar problemas con el sistema y **NO** debe usarse. Se explica y debate este tema en gentoo-forums.

No obstante, algunas veces portage sigue queriendo retroceder en la versión de un paquete, a pesar de aplicar la solución anterior. Generalmente hay una buena razón para esto, y es recomendable hacer caso a portage. Hay alguna excepción, con las que se puede usar otro método pero **SOLO** se debe usar para estas excepciones. Por ejemplo linux-headers:

echo sys-kernel/linux-headers -* >> /etc/portage/package.keywords

NOTA: linux-headers-2.4.x funciona perfectamente, aunque tengas un kernel 2.6.x instalado, es solo un ejemplo gráfico de como usar esa solución.

A veces, algunos paquetes, intentan volver al sistema, tras ser desinstalados, aun cuando no son necesarios. Los usuarios de x.org

habrán detectado a xfree intentando volver al sistema. para corregir esto:

echo x11-base/xfree >> /etc/portage/package.mask

Existen también paquetes que están **hard masked** un ejemplo es realone, está **hard masked** porque contiene un agugero de seguridad que puede comprometer la seguridad del sistema. Si por alguna razón, queremos instalar realone, o algun otro paquete **hard masked** hay una forma correcta de hacerlo:

echo media-video/realone >> /etc/portage/package.unmask

Si quieres que un paquete no incluya, o incluya algun valor USE (que no esté así indicado en /etc/make.conf) la forma adecuada de hacerlo, para que portage mantenga el control sobre los valores USE de ese paquete siempre, es algo como:

echo net-p2p/bittorrent -X >> /etc/portage/package.use

MANTENIEDO EL FICHERO WORLD

En raras ocasiones, los paquetes no son añadidos a world por alguna razón. Para intentar corregir esto:

regenworld

¿CÖMO FUNCIONA DEPCLEAN?

Bien, ahora que hemos aprendido como mantener el registro de nuestros programas (fichero **world**) y con ello el control sobre ellos. Vamos a ver, para que sirve y como funciona la herramienta depclean:

Cuando ejecutamos: **emerge depclean** depclean toma el fichero **world**, y calcula todas las dependencias de los elementos que están en este. Cualquier cosa que no esta en el fichero **world**, ni es dependencia (indirecta o directa) de algo que esta en el **world**, es eliminado.

Ejemplo de depclean
emerge gnome instalará dep1, dep2a, dep2b, dep2, dep3 , luego gnome, y añadira **gnome** al archivo **world**.
gtkapp depende de dep1 y dep3. **emerge gtkapp** instalará **gtkapp** y lo añadirá al archivo **world**.
emerge -C gnome desinstalará gnome, y lo eliminará del world, pero deja las 5 dependencias en el sistema.
emerge depclean leerá entonces el world, calculará las dependencias, y desde que gnome se ha ido, no hay paquetes que dependan de: dep2a,, dep2b, o dep2, estos serán eliminadas. Sin embargo dep1 y dep3 se mantendrán en el sistema, ya que son dependencia de **gtkapp**, que esta en el **world.**

NOTA: Imagino, que como yo hasta hace muy poco, pensaréis que depclean es un suicidio, que no funciona bien, pero todo lo contrario, funciona tan bien, como tu archivo **world** este mantenido. citando algunos *post* en los foros de gentoo sobre este tema:

```
Packages installed:   547
Packages in world:    155
Packages in system:   79
Unique package names: 547
Required packages:    563
Number to remove:     0
```

ACTUALIZACIÓN:

Cuando leí sobre este tema, en gentoo forums encontré un par de scripts, escritos por etcatmur (gracias por el trabajo ;-), Filesystem cruft script y dep script.
El primero, se utiliza para limpiar el sistema de ficheros, es decir, te devuelve una lista de ficheros, que no pertenecen a ningun paquete instalado, y controlado por **portage**.
El segundo, trata de minimizar el contenido del fichero world, eliminando entradas, que etcatmur considera **recursivas**. Para tenerlo claro, aquí hay un ejemplo de salida en pantalla:

```
# dep -w
      !!!REDUNDANT!!! app-admin/gkrellm depended on by:
              x11-plugins/gkrellflynn-0.6
              x11-plugins/gkrellmoon-0.6
```

```
        x11-plugins/gkrellmss-2.3
        app-admin/syslog-ng
!!!REDUNDANT!!! app-editors/gvim depended on by:
        app-text/vim-latex-1.5_rc1
        app-vim/info-1.7-r1
        app-editors/nano
        app-emulation/wine
        x11-themes/mplayer-skins
        x11-themes/nautilus-themes

145 packages in worldfile: 137 valid, 8 redundant; 70 packages in system.
563 packages installed: 12% in system, 25% in world, 63% deps.
```

Bien, personalmente, pienso que es mucho mejor, llevar ordenado el sistema, y el registro desde un principio, **usando portage correctamente** que tratar de ordenar algo desordenado. El principal inconveniente, es que, como veis, el script, decide eliminar la entrada **app-admin/gkrellm** por que, esta ya es dependencia, de otros paquetes que constan el el fichero **world**.

¿Cuál es el problema de esto? que si un dia decido desinstalar **gkrellflynn, gkrellmoon, gkrellmss** por lo que sea, e incluso decido cambiar de logger, y también desinstalo syslog-ng, entonces no quedará nada en el world, referente a gkrellm, y cuando ejecute **emerge depclean** trataría de desinstalarlo.

Esto no pasaría solo con gkrellm, así que desbarata todo el orden que intentamos conseguir con este artículo.
De todas formas, echad un ojo a este script también porque tiene otras funciones que resultan muy útiles, como calcular las dependencias inversas de un paquete.

Agradecimientos a NuclearFusi0n y GaMMa por las ideas originales, y por compartirlas con toda la comunidad ;-)