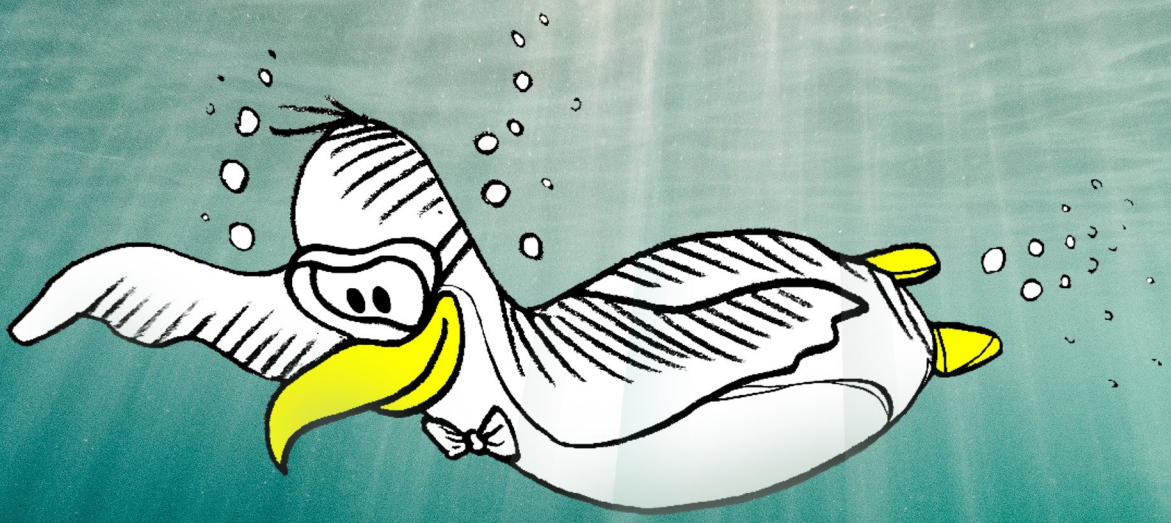
Visita nuestro sitio SoloLinux.es

MAGAZINE SOLO LINUX

Nº 34

Tu revista, la revista de tod@s

ENERO 2022



- Instalar **Xampp** en **Ubuntu 20.04** y otras distribuciones Linux
- Terminar o matar un proceso **zombi** en Linux
- **RKHunter:** análisis del sistema Linux en busca de **rootkits** y **malware**
- Curso de **Vim** by Victorhck
- Uso del gestor pacman en **Arch Linux** Comandos básicos
- Seguridad y hardening en Linux
- **Gestor de juegos en Linux:** Lutris más que un lanzador

MANUALES, SCRIPTS, SOFTWARE, HARDWARE, DISTROS LINUX, SEGURIDAD, REDES Y MUCHO MAS EN LA WEB...

Esta revista es de **distribución gratuita**, si lo consideras oportuno puedes ponerle precio.
Tu también puedes ayudar, contamos con la posibilidad de hacer donaciones para la REVISTA, de manera muy simple a través de **PAYPAL**

AYUDANOS A SEGUIR CRECIENDO

PayPal™

Donar a Revistalinux

SoloLinux

Dirección y maquetación

- **Adrián Almenar**

e-mail: adriansololinux@gmail.com

Redacción

En este numero especial los redactores han sido los siguientes:

- **Sergio G. B.** (Solo Linux. Seguimos compartiendo tus artículos, Gracias)
- **Erwin Andres Espitia** (Espacio Tecnológico)
- **Baltolkien** (Kde Blog)
- **Alexynior** (esgeeks)
- **Jaime Pons** (AprendoLinux)
- **Victorhck** (victorhckinthefreeworld)
- **Leillo1975** (Jugando en linux)
- **Lorenzo** (Atareao)
- **Pedro Crespo** (Latin Linux)
- **Andrea y Flavia** (CarreraLinux)

Diseño Portada

- **Karina Fernández**

Instagram: @karyfernandez.design

Publicidad

Quieres poner publicidad en la revista, ahora puedes hacerlo de forma muy simple, llegando a todo el mundo con esta revista digital de software libre y GNU/Linux en ESPAÑOL

CON SOLOLINUX MULTIPLICARAS TUS CLIENTES

Para mayor información escribe un e-mail: adriansololinux@gmail.com

Contacto

Para cualquier consulta sobre la revista, publicidad o colaboraciones escribir un email a: adriansololinux@gmail.com

Aviso – Nota del autor:

Los sitios **SoloLinux.es, RevistaLinux.es** y la **Revista SoloLinux**, no mantienen ningún tipo de relación contractual con los propietarios de otros blogs, autores de opiniones publicadas o anunciantes de la revista.

La revista SOLOLINUX esta realizada con **Libre Office Impress 7.2.1.2**

Bienvenido a la Revista SOLOLINUX

Hola a todos y bienvenidos a este nuevo numero de la Revista SOLOLINUX.

En este numero 34 ronda cierta incertidumbre en la forma de continuar con la revista. Esperemos que podamos continuar nuestras publicaciones sin cesar.

Este numero ha sido posible a la gran ayuda de diferentes comunidades de Linux, que nos han cedido varios artículos para poder homenajear el gran trabajo que realizo Sergio para la comunidad Linux en general.

Si quieres participar en la revista ponte en contacto con adriansololinux@gmail.com

Gracias a TOD@S
Adrián Almenar

Jugando En Linux

LATIN LINUX TU LUGAR EN EL UNIVERSO GNU

ATAREAO Con Linux

Victorhck in the free world omnia sunt communia!

APRENDOLINUX

Espacio Tecnológico

KDE Blog

ESGEEKS

Esta revista es de **distribución gratuita**, si lo consideras oportuno puedes ponerle precio.
Tu también puedes ayudar, contamos con la posibilidad de hacer donaciones para la REVISTA, de manera muy simple a través de **PAYPAL**

PayPal Donar a Revistalinux

AYUDANOS A SEGUIR CRECIENDO

Solo Linux

Espacio Tecnológico

Espacio Tecnológico

Esgeeks

ESGEEKS

Kde Blog

KDE Blog

Aprendo Linux

APRENDOLINUX

Victorhck in the freeworld

Victorhck in the free world
omnia sunt communia!

Jugando en Linux

Jugando En Linux

Atareao

ATAREAO
Con Linux

Latin Linux

LATIN LINUX
TU LUGAR EN EL UNIVERSO GNU

Carrera Linux

VANT

SOMOS LINUXEROS

CALIDAD HARDWARE, LIBERTAD SOFTWARE

La gama más completa de ordenadores con GNU/Linux

CORAZÓN LINUXERO

Siente la velocidad, estabilidad, seguridad, versatilidad y libertad que ofrece GNU/Linux y el software libre, y todo listo desde el momento en que enciendas tu ordenador VANT, porque queremos que entres de la forma más sencilla en esta gran comunidad

CALIDAD HARDWARE

Qué mejor forma de disfrutar del mejor software que acompañándolo de un hardware de calidad. El hardware de los ordenadore VANT está cuidadosamente seleccionado y testado para garantizar un funcionamiento óptimo con GNU/Linux

GARANTÍA ESPAÑOLA

2 años de garantía. Recogida a domicilio y envío, mano de obra y piezas. Todo está incluido. Atención telefónica para aclarar tus dudas o resolver cualquier problema.
La calidad hardware y software siempre acompañada de un soporte cercano y eficaz.

vantpc.es @vantpc @vantpc_es vant.pc t.me/vantpc

INSTITUTO LINUX

[UNIVERSIDAD TECNOLOGICA NACIONAL] FRD [FACULTAD REGIONAL DELTA]

Linux Professional Institute APPROVED TRAINING PARTNER

THE LINUX FOUNDATION AUTHORIZED TRAINING PARTNER

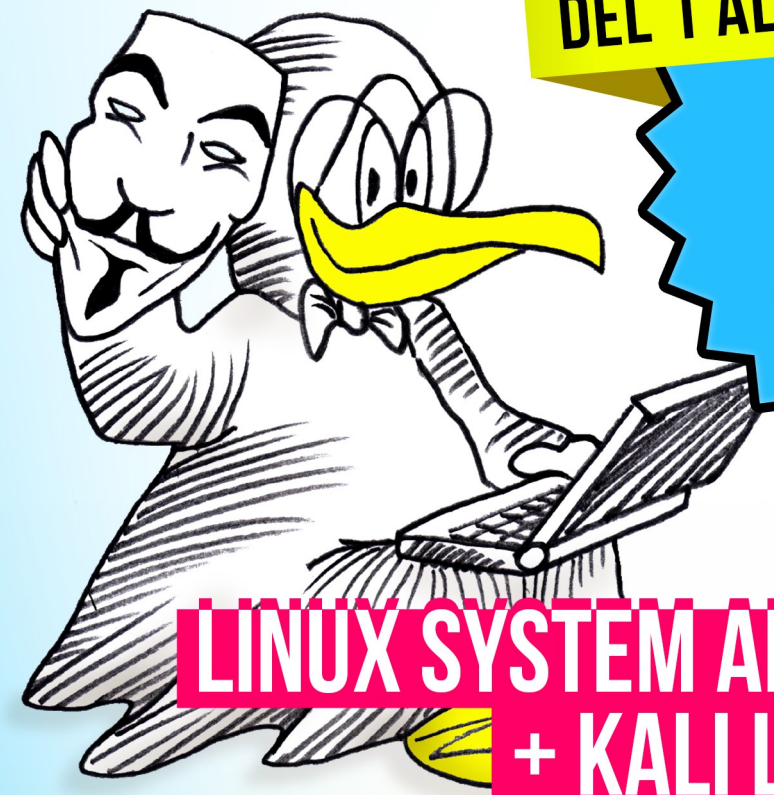
SUPER PROMO

DEL 1 AL 8 DE ENERO DE 2022

VALOR REAL
~~€ 990~~

1 PAGO DE
€ 147

CURSO PROXMOX DE REGALO PARA LAS PRIMERAS 6 VACANTES



LINUX SYSTEM ADMINISTRATOR + KALI LINUX

- 100 HORAS DE FORMACIÓN ON LINE -

CERTIFICACIÓN:
EXPERTO EN ADMINISTRACION Y SEGURIDAD EN REDES LINUX

+54 9 11 6969 9993

Uso del gestor pacman en Arch Linux – Comandos básicos

Arch Linux es una distribución que no instala aplicaciones por defecto como estás acostumbrado, te digo más... lo tienes que instalar todo tu mismo, incluso el entorno de escritorio. Parece complejo pero no lo es tanto, además de esta manera consigues un sistema perfecto adaptado a tus necesidades.

Arch viene con un administrador de paquetes llamado **Pacman**, que ofrece a los usuarios una forma simple de administrar paquetes. Los paquetes se pueden administrar desde repositorios oficiales o desde las compilaciones propias de un usuario con Pacman.

En este artículo, vemos los comandos básicos de este excelente gestor de paquetería linux, bajo **Arch Linux** y sus derivados. Lo que haremos es una pequeña explicación seguida del comando, así de fácil y rapido.

SoloLinux

Autor: Sergio G. B.

Fundador y editor de SoloLinux.
DEP Amigo.
Profesor de Unix en la universidad técnica de Odessa (Ucrania)
Una gran persona. Gracias por todo.

Gestor pacman en Arch Linux – Comandos básicos

FUNCIÓN	COMANDO BÁSICO
Actualizar el sistema y el software instalado.	pacman -Syu
Forzar la actualización de la base de datos y actualizar todo.	pacman -Syuu
Buscar un paquete con palabra clave.	pacman -Ss [paquete] # ejemplo pacman -Ss gimp
Buscar un paquete instalado.	pacman -Qs [paquete]
Instalar un paquete o varios paquetes.	pacman -S [paquete1] [paquete2]
Actualizar e instalar paquete.	pacman -Syu [paquete1] [paquete2]
Instalar paquete desde un repositorio local.	pacman -U /path/to/paquete.pkg.tar.xz
Instalar paquete desde un repositorio no oficial.	pacman -U https://sitioweb.com/repo/paquete.pkg.tar.xz
Listar paquetes instalados en Arch.	pacman -Ql
Descargar paquete sin instalarlo.	pacman -Sw paquete
Desinstalar un paquete.	pacman -R paquete
Desinstalar un paquete y sus dependencias.	pacman -Rsu paquete
Buscar archivos huérfanos.	pacman -Qdt
Borrar todo excepto el sistema base.	pacman -D --asdeps $(pacman -Qqe) pacman -D --asexplicit base linux linux-firmware pacman -Rns $(pacman -Qtdq)
Borrar cache de paquetes excepto los tres más recientes.	paccache -r
Borrar cache de paquetes que ya no están instalados.	pacman -Sc
Borrar toda la cache de paquetes.	pacman -Scc

Los comandos vistos son los más utilizados de **pacman en Arch linux**, realmente son muchos más.

Qué es el software Open Source

Antes de comenzar el artículo de hoy, debes conocer algunos conceptos que suelen pasar desapercibidos para la mayoría de usuarios.

Al escribir una aplicación o software, ese código no puede ser leído por nuestras máquinas. Lo que hacemos es, desarrollarlo íntegramente en texto plano con el lenguaje que nos interese. Posteriormente, este código se debe compilar de forma que se genere un código binario.

Ahora sí que nuestros sistemas pueden leer el software, en binario. Los binarios son archivos de texto que solo contienen ceros y unos, que dependiendo de la secuencia representan variadas instrucciones en el sistema. Por ejemplo, si pasamos a binario «**sololinux.es**», el resultado sería...

01110011 01101111 01101100 01101111 01101100 01101001 01101110 01110101 01111000 00101110 01100101 01110011

Qué es el software Open Source
Un poco de historia
Si nos remontamos a la década de los 80, el código fuente casi siempre se ofrecía forma gratuita. Lo normal era otorgar permisos para redistribuir y modificar, a tu antojo y libre albedrío. Si te lo paras a pensar, era algo lógico pues la distribución de aplicaciones se producía de mano en mano, por lo tanto pocos usuarios tenían acceso a ellas.

A principio de los 90, todo cambio. Se produjo la primera revolución de internet, el acceso a la red de redes ya era algo factible para la mayoría de usuarios. Los más viejos del lugar aun recordaran aquellos modems full duplex de 300 bits por segundo (una auténtica revolución, si señor). Si nos centramos en España, el servicio ofrecido era **Ibertex**, unos años más tarde paso a ser **Infovia** hasta que desapareció por la liberación del mercado.

Esto produjo una amplia distribución del software. Los desarrolladores no tardaron en reaccionar, se dieron cuenta de que su trabajo les podía reportar altos emolumentos. Las compañías de software comenzaron a nacer como setas, y con ellas las licencias privativas.

Cada licencia tenía sus propios términos legales, en ellos se indicaba lo que podías hacer con el software y lo que no. Muchos programas requerían firmar un **Acuerdo de licencia de usuario final**, lo que aún se conoce como **EULA**. El software **Open Source** parecía destinado a desaparecer.

El momento del cambio
Unos años antes, **Richard Stallman** imagino esta situación y creó lo que se conoce como la **Free Software Foundation** (**FSF**), que hoy en día continúa promoviendo las alternativas del software libre.

Stallman redactó las cuatro normas para que un software sea libre:

0- Debes tener la libertad de ejecutar el programa como quieras y para lo que quieras.
1- Se debe permitir el estudio del software y su modificación sin condiciones. El acceso al código fuente es obligatorio.
2- Eres libre de redistribuir copias del software.
3- Debes redistribuir copias del software que tu mismo has modificado.

Hoy en día, existen muchos tipos de licencia de **software Open Source**. La más conocida es la GPL (GNU Public License), que se lanzó en 1989.

¿Qué es el Open Source?
Durante años existió una cierta controversia con el «software libre», debido a los muchos tipos de licencia, algunos usuarios confundían el software libre (gratis), con libertad absoluta. Corría el año 1998, cuando para solventar estas dudas se estableció la Iniciativa Open Source.

La diferencia real entre el software libre y el código abierto, es sustancial. Pero es evidente que son la cara y la cruz de una misma moneda. El software libre es una filosofía que considera que crear software libre es una elección moral, el resto es inmoral. Por otro lado, el movimiento **Open Source** afirma que es solo un tecnicismo, ellos se centran en el código, en cómo mejorarlo y como colaborar para hacer crecer una aplicación.

Hoy en día, tenemos a nuestra disposición toneladas de aplicaciones y sistemas operativos que se pueden considerar **Open Source**. Firefox, WordPress, **LibreOffice**, la mayoría de **distribuciones linux**, etc.

¿Por qué debes usar software Open Source?
Vamos por partes, el software Open Source nos permite…

1) Acceder al código fuente, estudiarlo, modificarlo, redistribuirlo y publicar nuestras propias modificaciones.
2) El código fuente es abierto, por tanto es fácil verificar que no tiene puertas traseras o spyware integrado en el código.
3) Por lo general, siempre existe una comunidad detrás del software Open Source. Puedes integrarte en el equipo de desarrollo y colaborar con la comunidad.
4) Si eres usuario de alguna aplicación Open Source y no te gusta el rumbo que toma el software, puedes bifurcarlo y seguir tu propio camino.
5) Algunas herramientas Open Source pueden ser de pago. No es lo normal, pero existen.
6) Puedes ser parte activa de la comunidad.

¿Existe software Open Source de pago?
No todo el software de código abierto es gratuito. Dependiendo de la licencia elegida, puedes decidir aplicar una tarifa para poder descargar la herramienta. Aun siendo así, debes recibir el código fuente con derechos para redistribuirlo y modificarlo.
Lo explicamos… el usuario que descargue el software tiene el derecho a cederlo de forma gratuita a otros usuarios. Esta fórmula no es práctica, por eso muchos desarrolladores prefieren obtener beneficios a cambio de prestación de servicios, soporte, complementos o características adicionales.

¿Por qué se desarrolla software Open Source?
Los motivos son variados, seguro que alguno te sorprende:

- Grandes empresas como Google, Facebook y muchas más, dependen del software para ejecutar correctamente su propia infraestructura. Lo que hacen es colaborar con la comunidad para lograr su objetivo y, una vez conseguido continuar su propia rama ya privativa. Un ejemplo claro lo tenemos con el **navegador web Chromium y Chrome**.
- Otros programadores disfrutan desarrollando software, es su pasión y su máxima alegría es compartir su trabajo. Es una buena forma de perfeccionar tu aplicación, ya que obtienes críticas constructivas, consejos y la colaboración desinteresada de otros usuarios.
- En la mayoría de los casos, es que existe un software privativo y patentado que cobra por su uso (a veces de manera desorbitada). Para permitir que este tipo de herramientas lleguen al máximo de usuarios posibles, un equipo colaborativo desarrolla un **sofware Open Souce** que tenga un manejo y funciones similares.

Nota final
Las **licencias Open Source** no solo tratan el software, también otras aplicaciones, incluso post de sitios web también se pueden acoger a estas licencias. Por ejemplo en **SoloLinux** nos acogemos a la licencia **Atribución-CompartirIgual 4.0 Internacional (CC BY-SA 4.0)**.

Los usuarios que no son desarrolladores también contribuyen de forma importante, sus ideas, su visión de futuro y más, puede ayudarnos a elaborar un desarrollo o sitio web realmente interesante.

Autor: Sergio G. B.

Fundador y editor de SoloLinux.
DEP Amigo.
Profesor de Unix en la universidad tecnica de Odessa (Ucrania)
Una gran persona. Gracias por todo.

Instalar Xampp en Ubuntu 20.04 y otras distribuciones Linux

Antes de comenzar con la explicación, quiero aclarar que estas instrucciones son válidas para cualquier **distribución Linux**, no únicamente para **Ubuntu 20.04** y derivados.

Si quieres desarrollar aplicaciones o crear un sitio web, necesitas un entorno de pruebas inaccesible desde el exterior (en local). Este entorno debe incluir como mínimo todo lo que nos aporta **LAMP**; Este paquete es fácil de usar y administrar, pero si aún lo quieres más sencillo, tienes XAMPP.

XAMPP está diseñado para crear entornos de prueba puros y, su objetivo es instalar en un único paquete nuestro servidor local en cuestión de segundos. Es **Open Source**, ademas funciona en la mayoría de distribuciones Linux

Apache, MySQL / MariaDB, PHP y Perl, además de otras herramientas como phpMyAdmin, hacen de Xampp la opción preferida por los usuarios de linux. También se incluye OpenSSL, un servidor de correo electrónico con POP, SMTP e IMAP. La herramienta de análisis web **Webalizer** viene incluida, pero puede dar problemas en algunos entornos.

Instalar Xampp en Ubuntu 20.04 y otras distribuciones Linux
Lo primero que necesitamos son los binarios ejecutables de XAMPP. Para ello, lo único que necesitas es visitar su **página oficial** y descargar la aplicación. En los siguientes enlaces tienes la última versión actual, todo depende del php que necesites.

- **7.3.33 / PHP 7.3.33 (64 bit)**
- **7.4.26 / PHP 7.4.26 (64 bit)**
- **8.0.13 / PHP 8.0.13 (64 bit)**

También puedes descargar tu versión preferida desde la terminal

```
# Xampp con php 7.3
wget -r https://www.apachefriends.org/xampp-files/7.3.33/xampp-linux-x64-7.3.33-0-installer.run
# Xampp con php 7.4
wget -r https://www.apachefriends.org/xampp-files/7.4.26/xampp-linux-x64-7.4.26-0-installer.run
# Xampp con php 8.0
wget -r https://www.apachefriends.org/xampp-files/8.0.13/xampp-linux-x64-8.0.13-0-installer.run
```

Una vez lo tengas descargado debes concederles permisos, si lo has descargado desde la web lo puedes localizar en tu carpeta de descargas, si fue a través de «wget», en...

/home/tu-usuario/www.apachefriends.org/xampp-files/8.0.13

Accedemos a su localización y desde la consola le damos permisos.

```
chmod 755 xampp-linux-*-installer.run
```

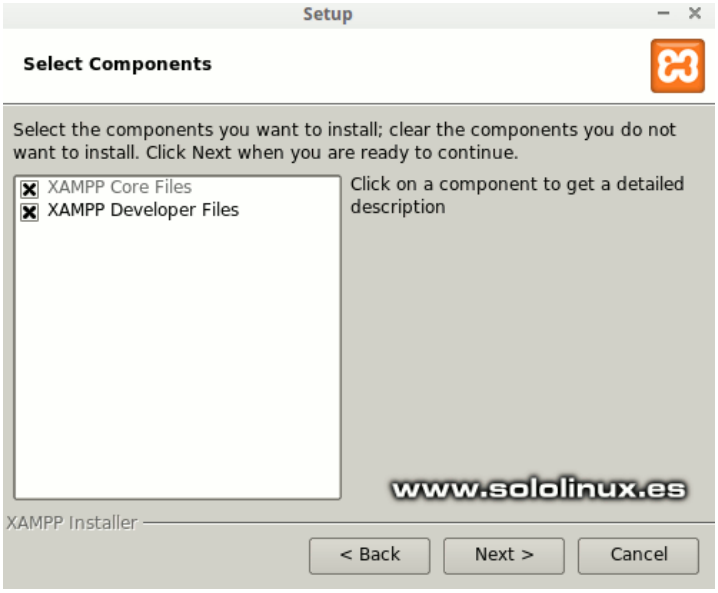
Lo instalamos.

```
sudo ./xampp-linux-*-installer.run
```

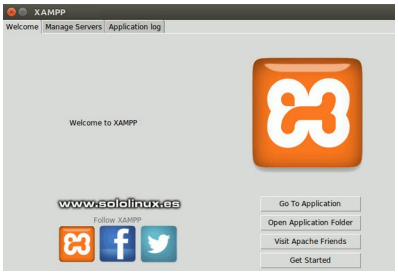
```
adrian@adrian-SoloLinux: ~/www.apachefriends.org/xampp-files/8.0.13
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
adrian@adrian-SoloLinux:~/www.apachefriends.org/xampp-files/8.0.13$ chmod 755 xampp-linux-*-installer.run
adrian@adrian-SoloLinux:~/www.apachefriends.org/xampp-files/8.0.13$ sudo ./xampp-linux-*-installer.run
```

Aparece una pantalla similar a la siguiente.

Comienza el instalador, tan solo tienes que seguir las indicaciones en pantalla.



Una vez termine la instalación, puedes iniciar tu servidor local XAMPP.

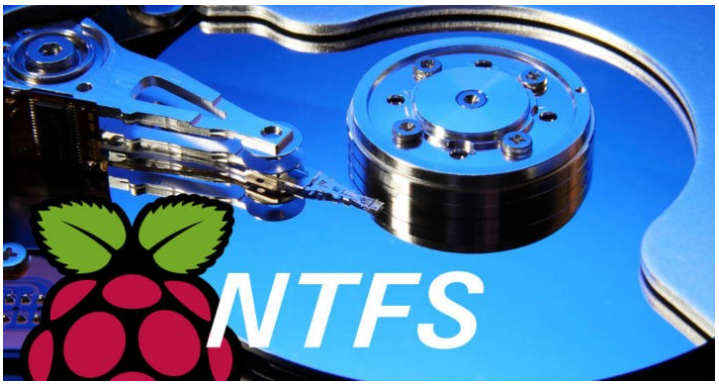


SoloLinux

Autor: Sergio G. B.
Actualizado: Adrián A. A.

Fundador y editor de SoloLinux.
DEP Amigo.
Profesor de Unix en la universidad tecnica de Odessa (Ucrania)
Una gran persona. Gracias por todo.

Configurar y conectar un disco duro externo en la Raspberry Pi 3 habilitada con NTFS



Que gran versatilidad la de la placa Raspberry Pi, las posibilidades de implementar una solución tecnológica con este dispositivo son casi infinitas. La imaginación es el límite. En esta ocasión les mostraré el procedimiento para configurar y conectar un disco duro externo en la Raspberry Pi 3, y habilitada con NTFS.

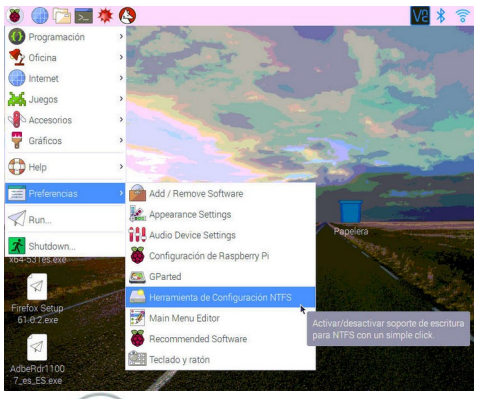
Si fuéramos a trabajar con un disco rígido netamente en GNU/Linux, bastaría con formatearlo con el sistema de fichero ext4, o similar. Pero como lo más seguro es que también interactúe en un ambiente Windows con restricciones, lo formatearémos con NTFS.

¿Por qué NTFS y no con un FAT32?... Por que con NTFS podemos almacenar archivos mayores a 4 GB, con FAT32 no, y eso nos quita versatilidad.

Instalar el soporte para NTFS en la Raspberry Pi 3
Bien, vamos a la línea de comandos para que instalemos el paquete ntfs-config, así:

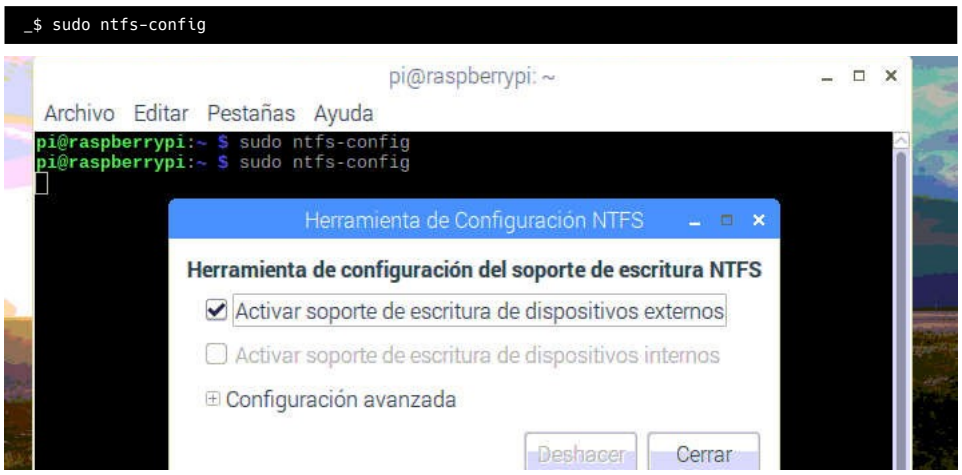
```
_$  sudo apt-get install ntfs-config
```

Después, vamos al **Menú -> Preferencias -> Herramienta de Configuración NTFS**, y activamos el soporte de escritura:



Espacio Tecnológico

También podemos abrir la herramienta con el siguiente comando:



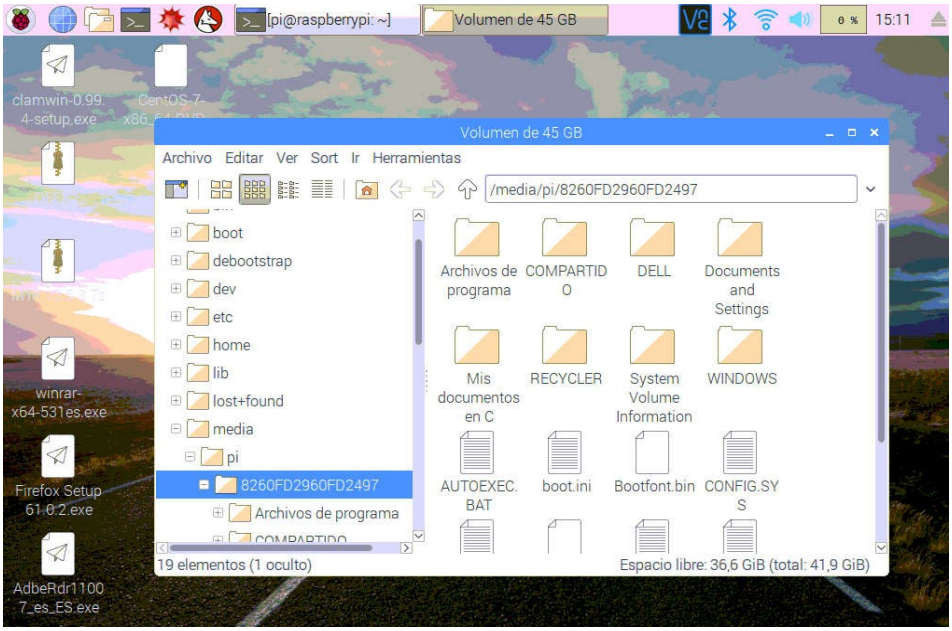
Instalación
En este caso, para configurar y conectar un disco duro externo en la Raspberry Pi debemos valernos de un convertidor **IDE/SATA to USB**.

Si el disco es **SATA**, fácil.

Si el disco es **IDE**, debemos configurarle sus jumpers para que trabaje como Master, y listo.

A la izquierda el convertidor IDE/SATA to USB.
A la derecha un disco duro antiguo tipo IDE de 80 GB.

Disco duro externo conectado a la Raspberry Pi 3

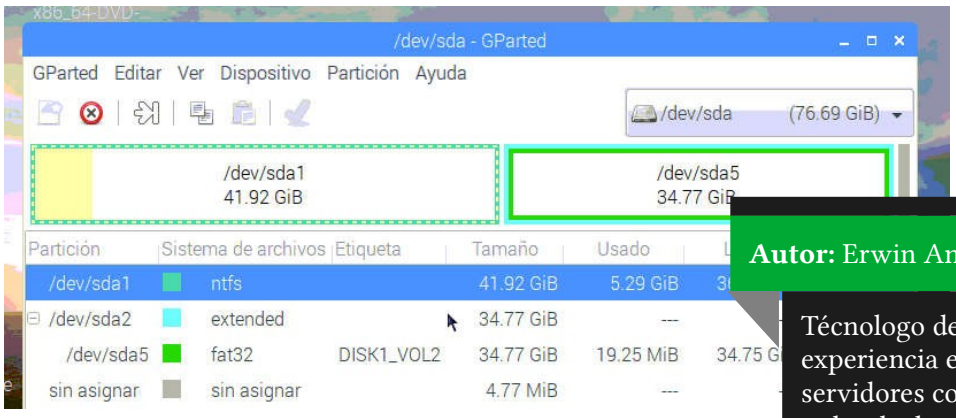


Ahora bien, como se trata de un disco duro antiguo con Microsoft Windows XP y queremos eliminarlo de ahí, podríamos borrar y ya, o usar **GParted** para formatear.

Instalamos GParted:

```
_$ sudo apt install gparted
```

Lo abrimos, seleccionamos el disco externo y formateamos (el disco debe estar desmontado):



Espacio Tecnológico

Autor: Erwin Andres Espitia

Técnologo del área de las TICs con 15 años de experiencia en el campo de gestión de servidores con Linux, bases de datos MySQL y redes de datos.
Web: **https://espaciotecnologico.co/**

Terminar o matar un proceso zombi en Linux

Los procesos en un sistema operativo son fundamentales para dotar de «vida» al hardware, o al sistema en general. En informática, un proceso es cualquier programa en ejecución, es decir, un conjunto de instrucciones en funcionamiento asociados a un estado y a una porción de recursos (memoria, CPU). Sin embargo, en Linux es muy rara la vez que un proceso finalizado continúe con su ciclo de «vida». A esta «rareza de la naturaleza» es lo que llaman un **proceso zombi** o **zombie**.

Un proceso defunct (difunto), o proceso zombi, en realidad es un proceso hijo cuyo proceso padre no lo finalizó. Básicamente el proceso hijo ha finalizado su ejecución, pero mantiene una entrada en la tabla de procesos. Simbólicamente hablando, el proceso hijo muere pero su «**alma**» sigue deámbulando en el sistema.

Verificación de un proceso zombi
Empleando el comando ps podemos verificar si nuestro sistema alberga algún proceso zombi. Pero antes, recordemos la identificación del estado de cada proceso que nos muestra **ps -Al**, en la columna Estado **(S)**.

- **S**– procesos suspendido (el proceso está esperando su turno para ejecutarse)
- **R**– proceso en corriendo
- **D**– proceso esperando a que finalice alguna operación de Entrada/Salida
- **Z**– proceso zombi

Tengamos en cuenta la columna PPID (id del proceso padre) y PID (id del proceso – proceso hijo), porque cuando un proceso pierde a su proceso padre, el proceso mayor init se convierte en su nuevo padre (PPID 1). Cuando el proceso es adoptado por init se le llama proceso huérfano. Sin embargo, cuando esto no sucede da nacimiento a un proceso zombi.

Realmente un proceso zombi no consume memoria ni procesador, pues se trata de una entrada «olvidada» en la tabla de procesos. En nuestro computador personal no tenemos mucho rollo para lidiar con este tema. Con reiniciar la máquina es suficiente. Sin embargo, en un entorno productivo quizá nos moleste ver esa entidad deámbulando por ahí, pues es evidencia de que un servicio o proceso no está funcionando del todo bien. En este caso debemos aplicar otras medidas más profesionales que veremos en seguida.

En busca de procesos zombis en Linux
El comando top nos puede hacer un recuento de posibles procesos zombis

```
Tareas: 196 total,   1 ejecutar,  195 hibernar,    0 detener,    0 zombie
%Cpu(s):  3,0 usuario,  8,9 sist,  0,0 adecuado, 87,4 inact,  0,0 en espera,  0,0 hardw int,  0,7 softw int,  0,0 robar tiempo
KiB Mem :  8174652 total,  3442196 free,  1766764 used,  2965692 buff/cache
KiB Swap:  8385532 total,  7894992 free,   490540 used.  5996424 avail Mem

  PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU %MEM     HORA+ ORDEN
25100 squid     20   0  905772 775016   4392 S  19,2  9,5 404:54.04 squid
23592 sistemas  20   0 49,921g 337544 112956 S   2,0  4,1   6:15.78 Web Content
  958 root      20   0  305024  69300  13696 S   1,0  0,8 626:00.97 Xorg
21428 root      20   0   36288   3540   2848 R   0,7  0,0   0:00.05 top
 1644 ntp       20   0  106616    384    176 S   0,3  0,0   5:36.95 ntpd
 1795 sistemas  20   0  677508  14464   6332 S   0,3  0,2  57:01.65 indicator-cpufr
 2212 sistemas  20   0  689264   4568   1220 S   0,3  0,1 147:45.14 mate-multiload-
23539 sistemas  20   0 2541928 327932 135616 S   0,3  4,0   4:52.32 firefox
```

El siguiente comando nos da información más detallada del proceso zombi.

```
ps -A -ostat,ppid,pid,cmd | grep -e '^[Zz]'
```

ó también

```
ps axo pid=,stat= | awk '$2~/^Z/ { print $1 }'
```

Eliminación de un proceso zombi en Linux

Si detectamos algún un proceso zombi en nuestro sistema, tenemos tres opciones para «matarlo» (kill). Son las siguientes:

- **Primera opción:**

```
kill -HUP `ps -A -ostat,ppid,pid,cmd | grep -e '^[Zz]' | awk '{print $2}'`
```

- **Segunda opción:**

```
ps -Ao state,pid | awk '$1=="2" {print $2}' | xargs kill -s SIGKILL
```

- **Tercera opción:**

```
ps -xaw -o state,ppid | grep Z | grep -v PID | awk '{ print $2 }' | xargs kill -9
```

- **Cuarta opción:**

```
kill -9 `ps xawo state=,pid=|sed -n 's/Z //p'`
```

- **Quinta opción:**

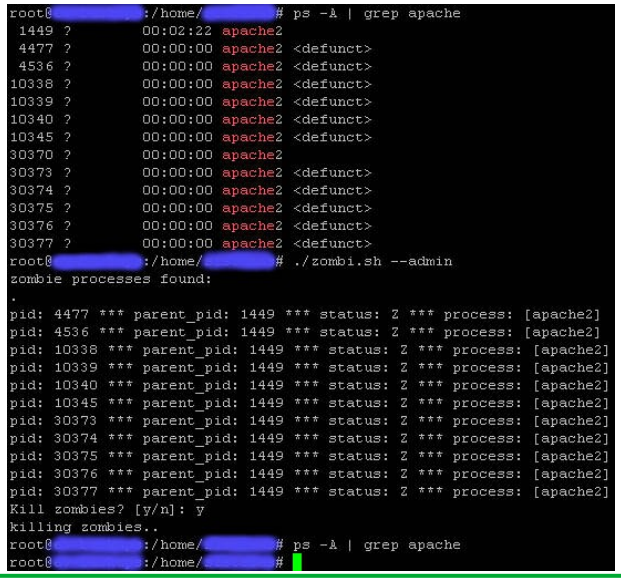
```
kill -9 `ps -xaw -o state -o ppid | grep Z | grep -v PID | awk '{print $2}'`
```

Si las cinco opciones anteriores no surten efecto, no se extrañe, ahí es donde el proceso zombi honra su nombre. Pero la última es la vencida:

- **Sexta opción:**

Descarga el siguiente archivo (link en la imagen), quítele la extensión «.txt», asígnele permisos de ejecución y corralo como usuario root con la opción –admin.

Por ejemplo, un día me sucedió con **Apache Web Server**. Sucedía que la página de la intranet corporativa no cargaba. Procedí a verificar el estado de los procesos y de inmediato el sistema me informa sobre 11 procesos zombis (defunct) de Apache. Acto seguido procedí a ejecutar el programa zombi.sh y solucionado:



Causas

Las causas más general de un proceso zombi es la mala programación del software de donde procede.


Espacio Tecnológico

Autor: Erwin Andres Espitia

Técnologo del área de las TICs con 15 años de experiencia en el campo de gestión de servidores con Linux, bases de datos MySQL y redes de datos.
Web: **https://espaciotecnologico.co/**

Instalar Whatsapp en Linux – Diciembre del 2021

Cada ciertos meses, **Gustavo González** nos sorprende con una nueva versión de **WhatsApp Desktop para Linux**.

WhatsDesk es un cliente no oficial de **Whatsapp**. Este proyecto solo inserta un **whatsapp web** en una **aplicación electron** y agrega una notificación de escritorio.

El creador solo ofrece versiones **.deb**, pero puedes instalarla en otras distros mediante **snap**.

Instalar Whatsapp en Linux – Diciembre del 2021
Para instalar **Whatsapp Desktop** en Linux (Debian, Ubuntu, Linux Mint y derivados), tan solo tienes que descargar tu versión preferida e instalarla haciendo **doble click sobre ella**.

También podemos instalarla desde la linea de comandos: abre una terminal en el directorio de descarga y entra el siguiente comando:

```
sudo dpkg -i whatsdesk_0.3.8_amd64.deb
```

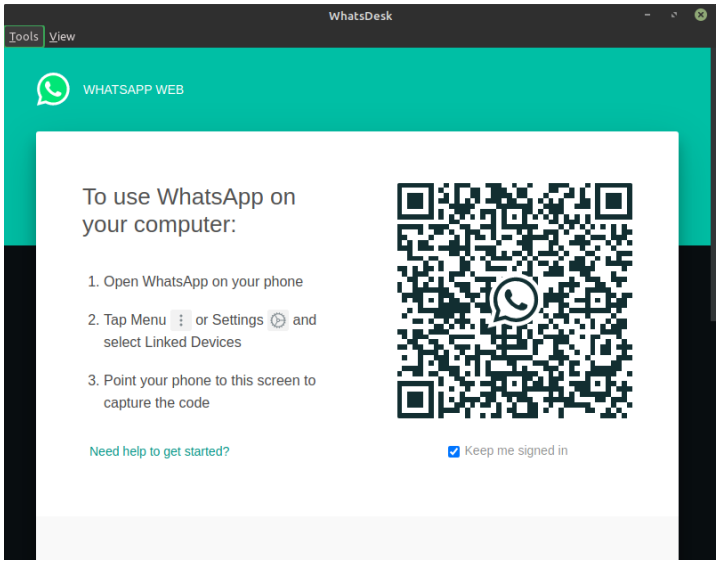
```
adrian@adrian-SoloLinux:~$ sudo dpkg -i whatsdesk_0.3.8_amd64.deb
Seleccionando el paquete whatsdesk previamente no seleccionado.
(Leyendo la base de datos ... 444724 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar whatsdesk_0.3.8_amd64.deb ...
Desempaquetando whatsdesk (0.3.8-72) ...
Configurando whatsdesk (0.3.8-72) ...
Procesando disparadores para gnome-menus (3.36.0-1ubuntu1) ...
Procesando disparadores para desktop-file-utils (0.24+linuxmint1) ...
Procesando disparadores para mime-support (3.64ubuntu1) ...
Procesando disparadores para hicolor-icon-theme (0.17-2) ...
```

En otras distribuciones Linux, puedes instalar la aplicación a través de **snap** con el siguiente comando:

```
sudo snap install whatsdesk
```

Iniciamos el programa, dando **click** en la aplicación desde el menu de aplicaciones de nuestra distribucion o si estas mas acostumbrado a la linea de comandos escribe el siguiente comando para iniciarla:

```
whatsdesk
```



Enlaces de descarga:

- **WhatsApp Desktop v.0.3.8 deb 64bits**
- **WhatsApp Desktop v.0.3.8 deb 32bits**

Descarga mediante comando:

```
64 Bit
wget -r https://zerkc.gitlab.io/whatsdesk/whatsdesk_0.3.8_amd64.deb
32 Bit
wget -r https://zerkc.gitlab.io/whatsdesk/whatsdesk_0.3.8_i386.deb
```

Autor: Adrián A. A.

Administrador y editor de la revista Solo Linux. Intentando acostumbrarme a esto de escribir :)

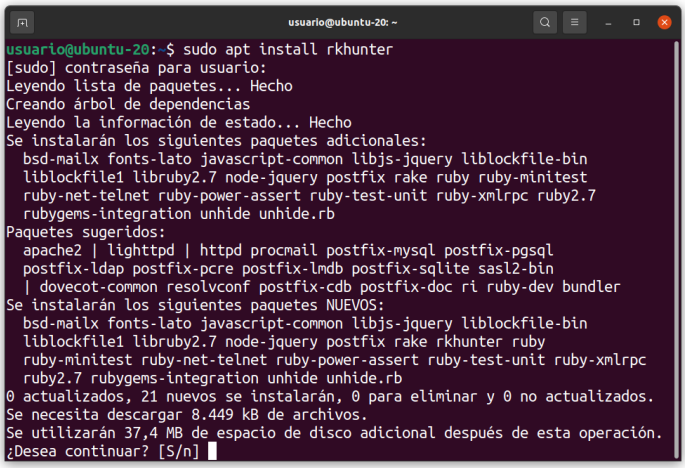
RKHunter: Análisis del sistema Linux en busca de rootkits y malware

Linux es un sistema operativo popular para muchos usuarios domésticos y empresariales. Para aquellos que tienen inquietudes acerca de la seguridad de su sistema Linux, **RKHUNTER** es una pequeña, pero potente herramienta que puede escanear el sistema para detectar cualquier virus y otro malware dañino.

rkhunter (Rootkit Hunter) buscará **rootkits**, **backdoors** y posibles **exploits** locales. Para ello, compara los hashes SHA-1 de los archivos importantes con los buenos conocidos en las bases de datos en línea, busca directorios por defecto (rootkits), permisos erróneos, archivos ocultos, entre otras cosas. **Rkhunter** destaca por estar incluido en sistemas operativos populares (Fedora, Debian, etc.)
Veamos cómo instalar y configurar esta utilidad para comprobar adecuadamente tu sistema.

INSTALACIÓN DE RKHUNTER EN EL SERVIDOR UBUNTU

Puedes instalar el software en Ubuntu con el comando



Si este método no te conviene, puedes construir el programa desde el código fuente:

```
cd /tmp
wget
http://downloads.sourceforge.net/project/rkhunter/rkhunter/1.4.6/rkhun
ter-1.4.6.tar.gz
tar -xvf rkhunter-1.4.6.tar.gz
cd rkhunter-1.4.6
sh ./installer.sh --layout default --install
```

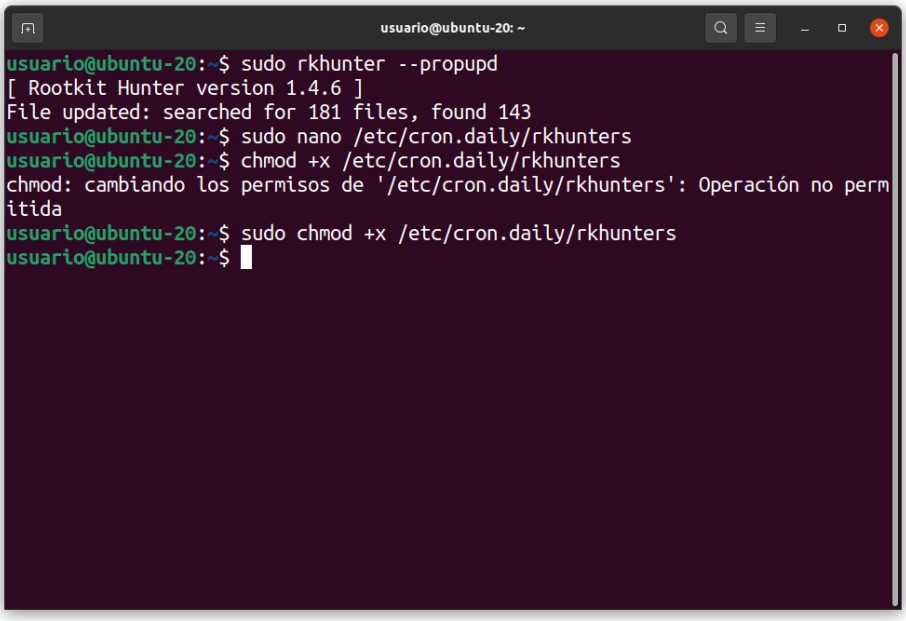
ACTUALIZACIÓN DE RKHUNTER

Antes de poder realizar la comprobación de virus en Linux, es necesario actualizar la base de datos de la utilidad. Para ello, ejecuta:

```
rkhunter –update
```

Ahora tenemos que recoger información sobre los archivos instalados en el sistema, para que el software pueda entender si alguien ha intentado modificar los archivos del sistema durante el siguiente análisis. Para ello, realiza la siguiente acción:

```
rkhunter –propupd
```



Es aconsejable actualizar regularmente, así que vamos a crear un **script** especial y ejecutarlo mediante cron todos los días. Para ello, crea un archivo de script en el directorio **/etc/cron.daily**:

```
sudo nano /etc/cron.daily/rkhunters

#!/bin/sh
(
/usr/bin/rkhunter --versioncheck
/usr/bin/rkhunter --update
/usr/bin/rkhunter -c --sk –rwo
) | mail -s 'rkhunter resultado escaneo'
hola@esgeeks.com
```

Aquí ejecutamos una comprobación de la versión, actualizamos las bases de datos y en la última línea que hemos programado para comprobar y enviar una notificación por correo electrónico. Para que funcione es necesario sustituir hola@esgeeks.com por tu dirección de correo electrónico.

Ahora sólo queda darle al programa los derechos de ejecución:

```
chmod +x /etc/cron.daily/rkhunters
```

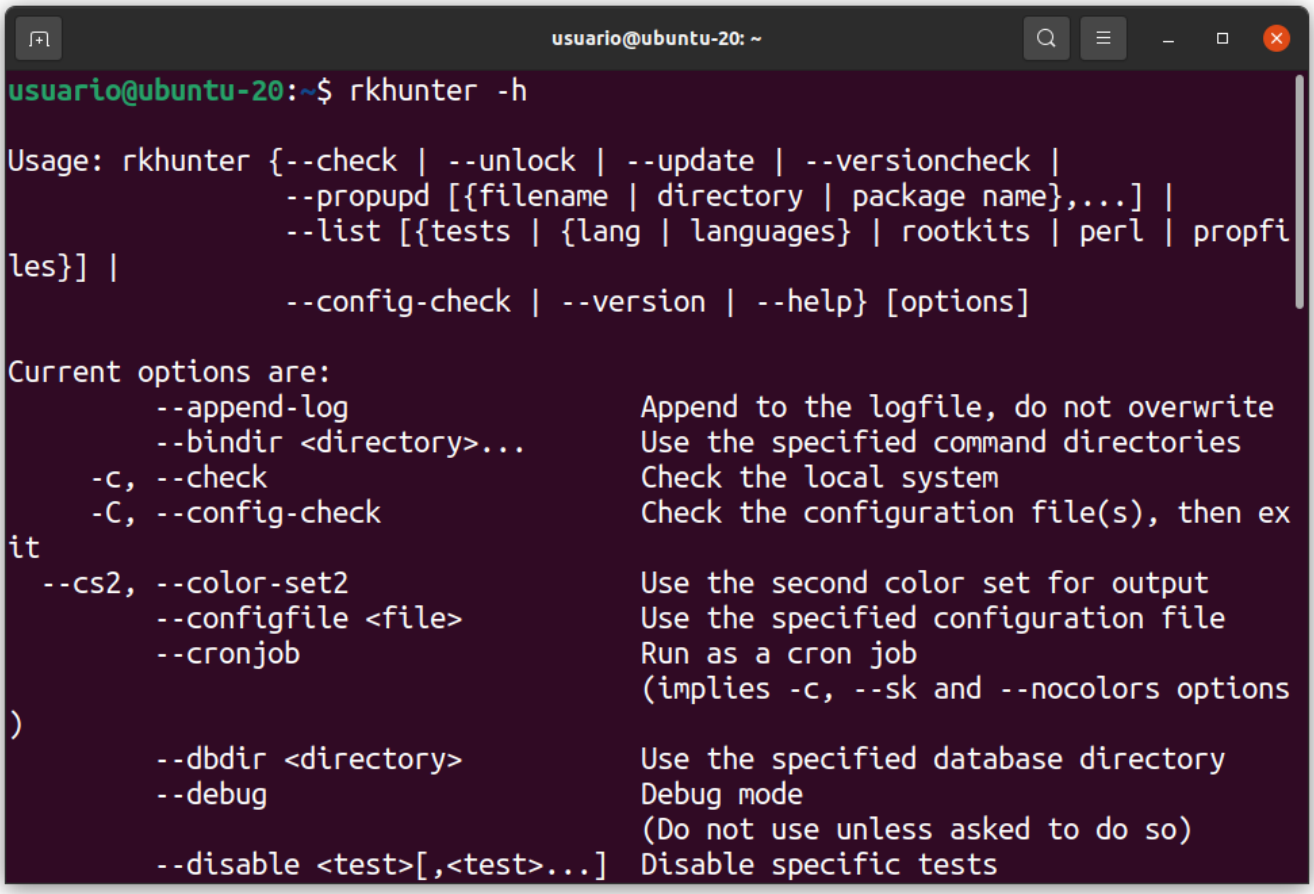
BUSCAR VIRUS EN LINUX RKHUNTER

En primer lugar, veamos las principales opciones que puedes necesitar:

- **--verbose-logging :** salida máxima detallada
- **--quiet :** salida mínima
- **-l, --logfile :** escribe el registro del programa en tu archivo
- **--cronjob** - es un modo no interactivo, que se utiliza para ejecutar mediante cron, de ahí su nombre.
- **--list** - te permite ver qué opciones soporta el programa, puedes pasar varios parámetros.
- **--unlock :** elimina el archivo de bloqueo de la base de datos, puede ser útil si la sesión anterior fue terminada incorrectamente.
- **--check :** comprobación del sistema
- **--update :** actualizar las bases de datos de rootkits
- **--versioncheck** - actualización del software
- **--propupd** - crear una base de datos de archivos

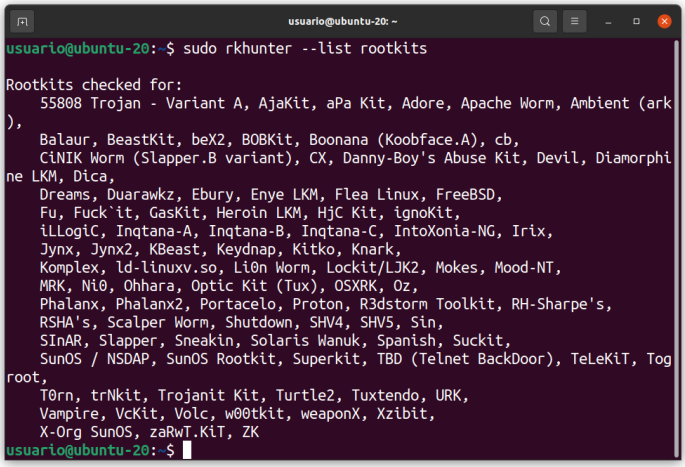
Para otras opciones, siempre puedes ejecutar el comando de ayuda:

```
sudo rkhunter -h
```



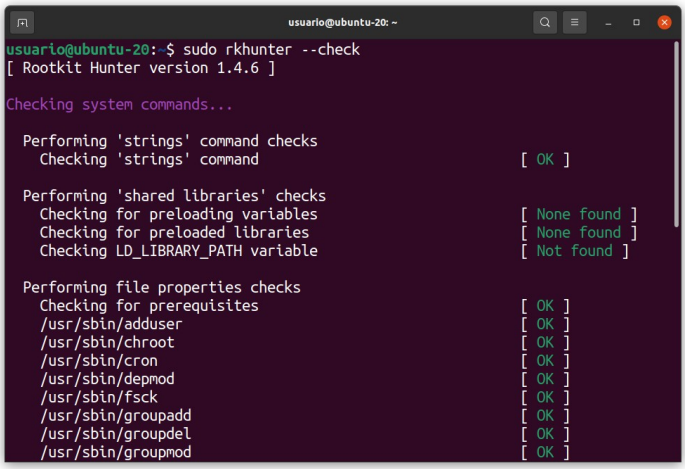
Por ejemplo, para ver todos los rootkits que el software puede encontrar, ejecuta:

```
sudo rkhunter --list rootkits
```



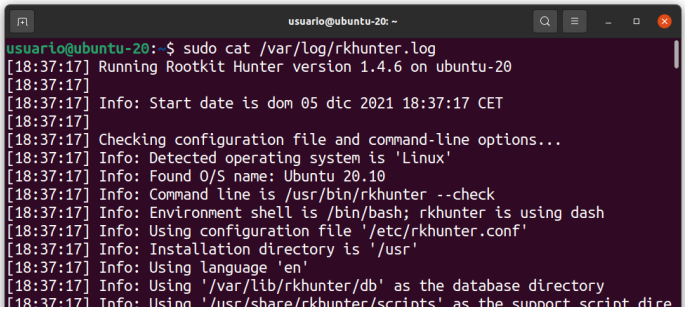
Para comprobar si hay virus en Linux, ejecute todo el sistema como usuario root:

```
sudo rkhunter –check
```



Además de mostrar información en la pantalla, el programa también creará un registro de pruebas. No prestes demasiada atención a la información que se muestra durante la prueba, está un poco truncada y se aclarará cuando mires el registro.

```
sudo cat /var/log/rkhunter.log
```



Desgraciadamente, sólo funciona en inglés, así que tendrás que entender un poco de inglés para comprender el estado de tu sistema.

Para ayudarte a entender lo que hace el software y cómo analizar tus resultados, veamos el registro de exploración.

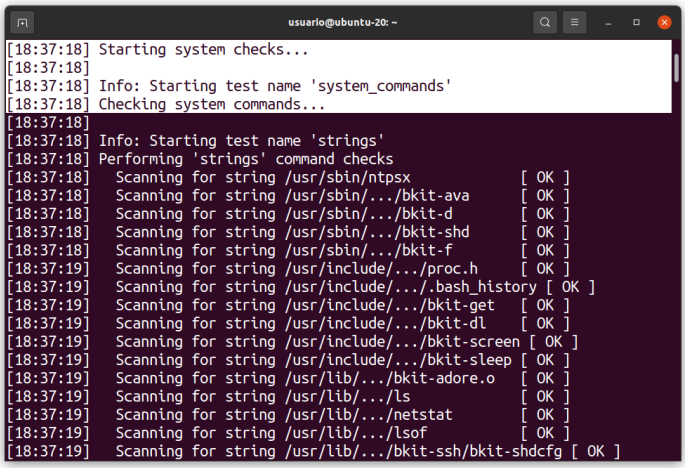
Primero inicializa y descarga los archivos de configuración, no hay nada interesante aquí:

```
usuario@ubuntu-20:~$ sudo cat /var/log/rkhunter.log
[18:37:17] Running Rootkit Hunter version 1.4.6 on ubuntu-20
[18:37:17] Info: Start date is dom 05 dic 2021 18:37:17 CET
[18:37:17] Checking configuration file and command-line options...
[18:37:17] Info: Detected operating system is 'Linux'
[18:37:17] Info: Found O/S name: Ubuntu 20.10
[18:37:17] Info: Command line is /usr/bin/rkhunter --check
```

Ten en cuenta que estamos viendo el registro de comprobación del sistema, los registros de actualización y de creación de la base de datos, que se encuentran arriba en el mismo archivo, no nos interesan.

La comprobación del sistema comienza con estas líneas:

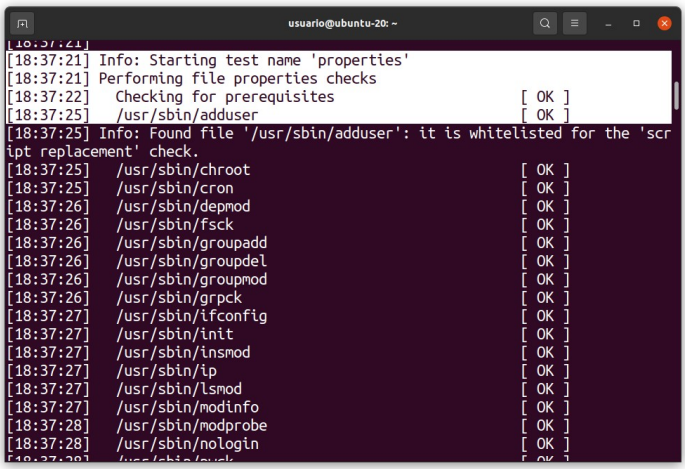
```
[18:37:18] Starting system checks...
[18:37:18]
[18:37:18] Info: Starting test name 'system_commands'
[18:37:18] Checking system commands...
```



El software escanea las herramientas del sistema y trata de detectar anomalías.

También compara la caché de la herramienta con la caché almacenada en la base de datos para ver si se ha modificado. Normalmente, si las herramientas están bien, el registro se llena de estas líneas:

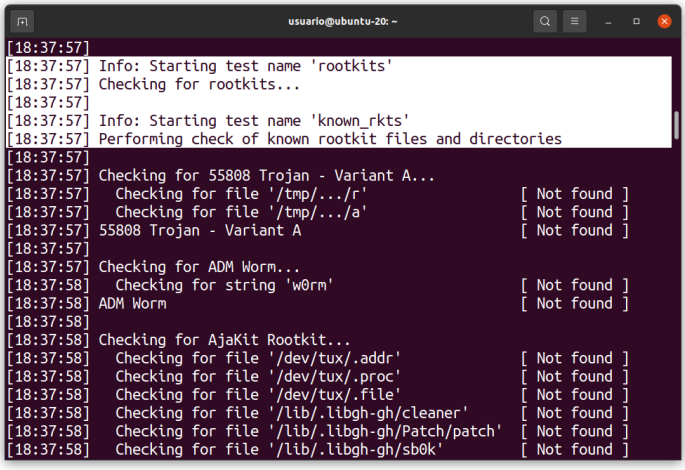
[18:37:21] Info: Starting test name 'properties'
[18:37:21] Performing file properties checks
[18:37:22] Checking for prerequisites [OK]
[18:37:25] /usr/sbin/adduser [OK]



Si se detecta un archivo sospechoso, el programa explica inmediatamente cuál es el problema.

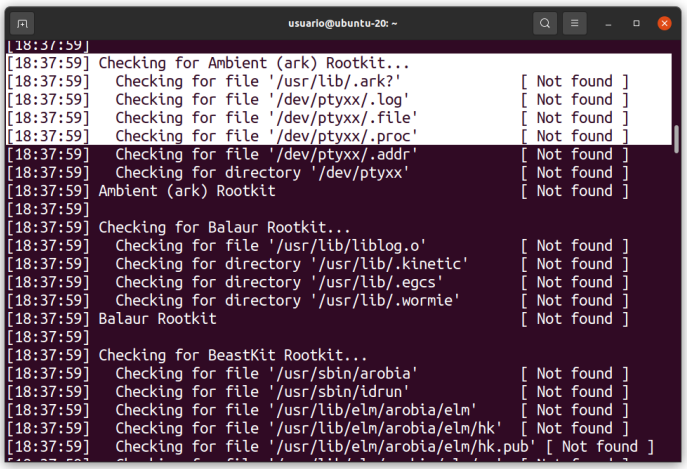
A continuación, analizará Linux en busca de virus y buscará rootkits conocidos:

[18:37:57] Info: Starting test name 'rootkits'
[18:37:57] Checking for rootkits...
[18:37:57]
[18:37:57] Info: Starting test name 'known_rkts'



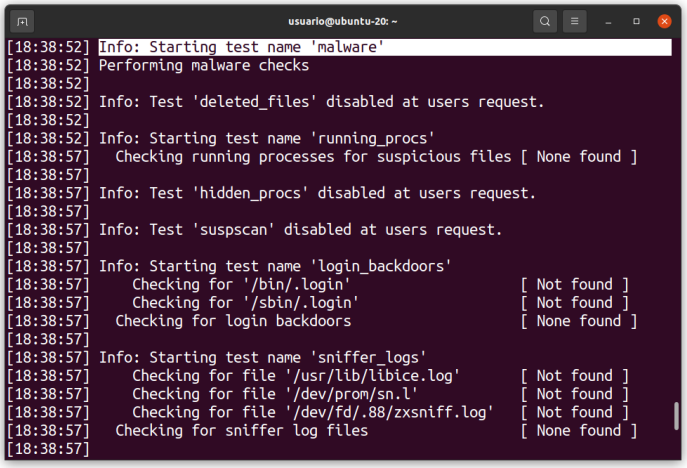
Normalmente, si se encuentra algo en esta sección, significa que hay un rootkit en él. El sistema tiene un rootkit y hay que hacer algo al respecto, pero normalmente vemos las líneas **Not found** (No encontrado):

[18:37:59] Checking for Ambient (ark) Rootkit...
[18:37:59] Checking for file '/usr/lib/.ark?' [Not found]
[18:37:59] Checking for file '/dev/ptyxx/.log' [Not found]



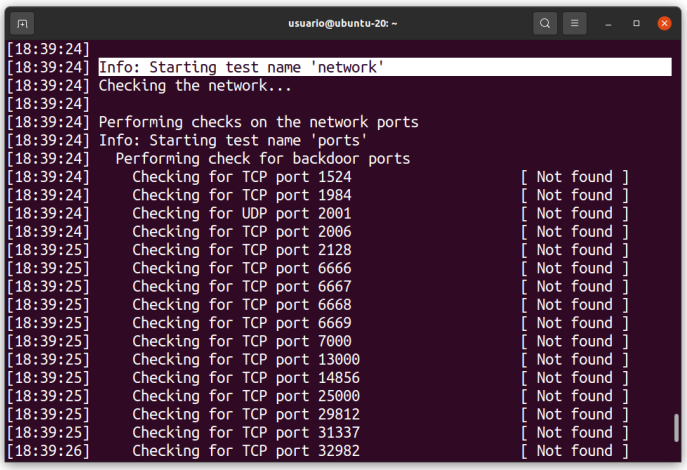
A continuación, se iniciará una búsqueda de malware:

[18:38:52] Info: Starting test name 'malware'
[18:38:52] Performing malware checks



Comprobará puertos peligrosos:

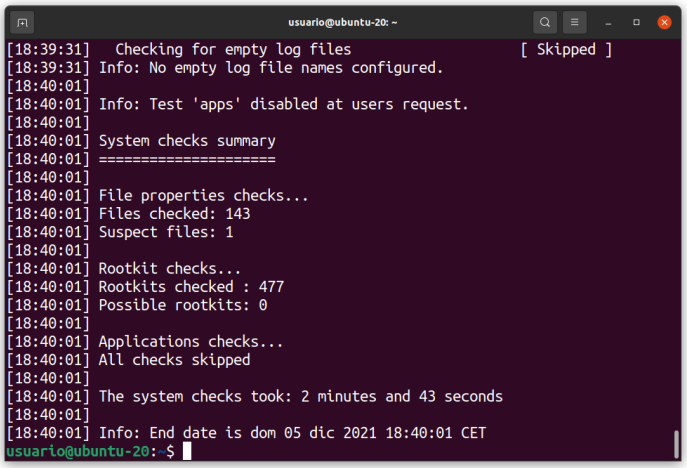
[18:39:24] Performing checks on the network ports
[18:39:24] Info: Starting test name 'ports'
[18:39:24] Performing check for backdoor ports
[18:39:24] Checking for TCP port 1524 [Not found]
[18:39:24] Checking for TCP port 1984 [Not found]
[18:39:24] Checking for UDP port 2001 [Not found]



Y no sólo ello. También verificará la configuración del sistema, apps, archivos de configuración, etc.

Dependiendo de tu configuración, es posible que detectes una advertencia. Por ejemplo, podrías estar utilizando la primera versión del protocolo SSH, y ello podría ser visto como ***muy inseguro***.
Finalmente, se mostrará un pequeño informe sobre los problemas encontrados:

```
[18:40:01] System checks summary
[18:40:01] ====================
[18:40:01]
[18:40:01] File properties checks...
[18:40:01] Files checked: 143
[18:40:01] Suspect files: 1
[18:40:01]
[18:40:01] Rootkit checks...
[18:40:01] Rootkits checked : 477
[18:40:01] Possible rootkits: 0
[18:40:01]
[18:40:01] Applications checks...
[18:40:01] All checks skipped
[18:40:01]
[18:40:01] The system checks took: 2 minutes and 43 seconds
```



Para la comodidad de ver el registro, no es necesario que lo veas en su totalidad, sino que selecciones solo las advertencias:

```
sudo cat /var/log/rkhunter.log | grep -A5 "\[ Warning \]"
```

El parámetro A5 significa mostrar cinco líneas más después de la línea con la ocurrencia detectada, por lo que definitivamente no nos perderemos nada.

Espero haber ilustrado por qué RKHunter es una gran opción para detectar rootkits y malware, y espero que empieces a usarlo para mantener tus sistemas Linux seguros.

ESGEEKS

Autor: Alexynior

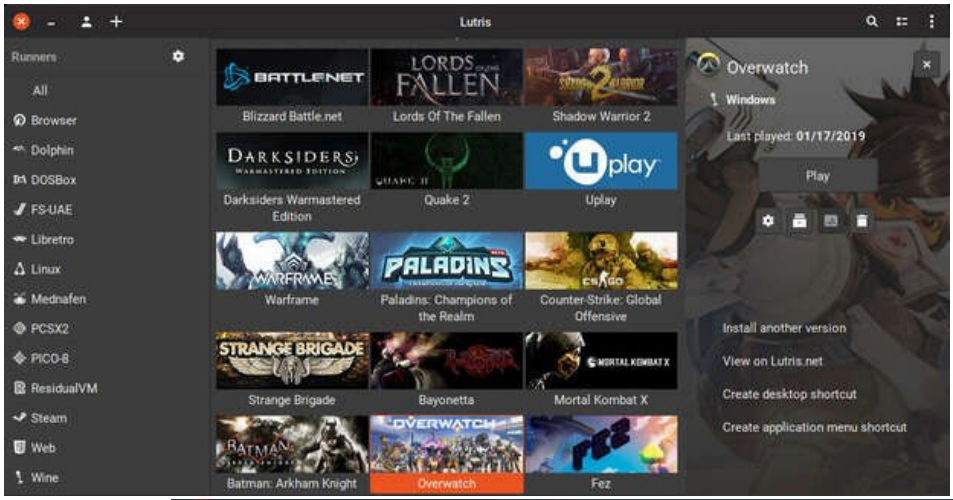
Redactor para Esgeeks.
Web: www.esgeeks.com

Gestor de juegos en Linux: Lutris más que un lanzador

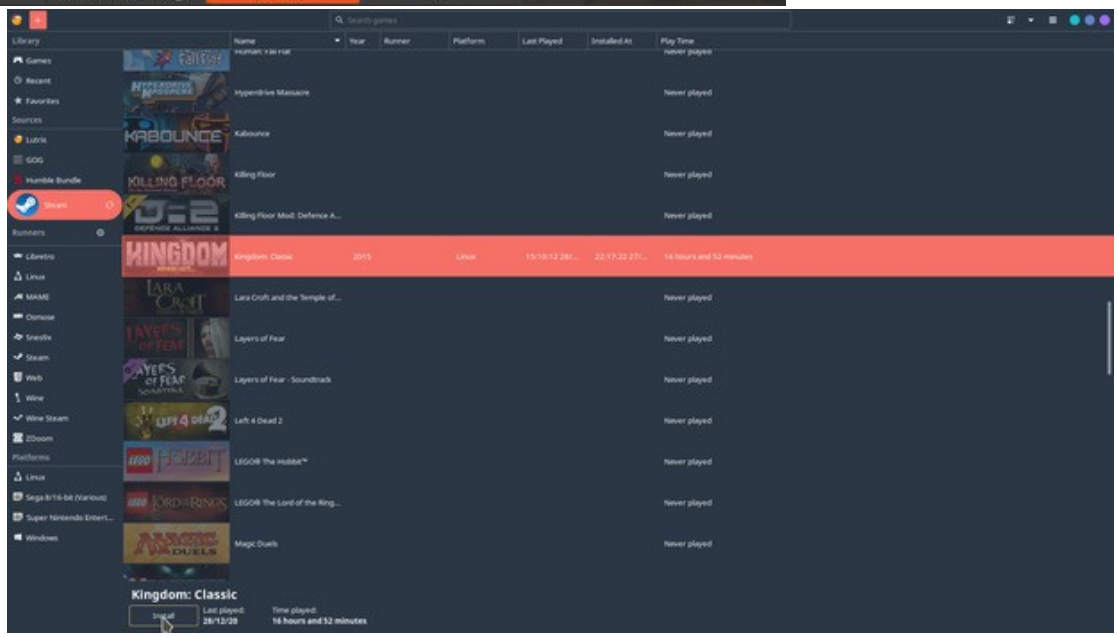
Atrás quedó el tiempo en que jugar en Linux era una quimera ya que era imposible hacerlos funcionar. Pero eso ha cambiado.

Dejando de lado los juegos libres y nativos para GNU/Linux (podéis ver un ejemplo de 31 de ellos activos en 2020 en esta entrada) y los disponibles vía emulación, las compañías de videojuegos y los emuladores especializados como Wine están haciendo que cada vez sea más fácil utilizar nuestra máquina libre en una plataforma de videojuegos.

Además, aparecen proyectos como el que nos ocupa en este articulo, **Lutris**, que no solo nos ayudan a instalar juegos sino que te ayuda a gestionarlos.

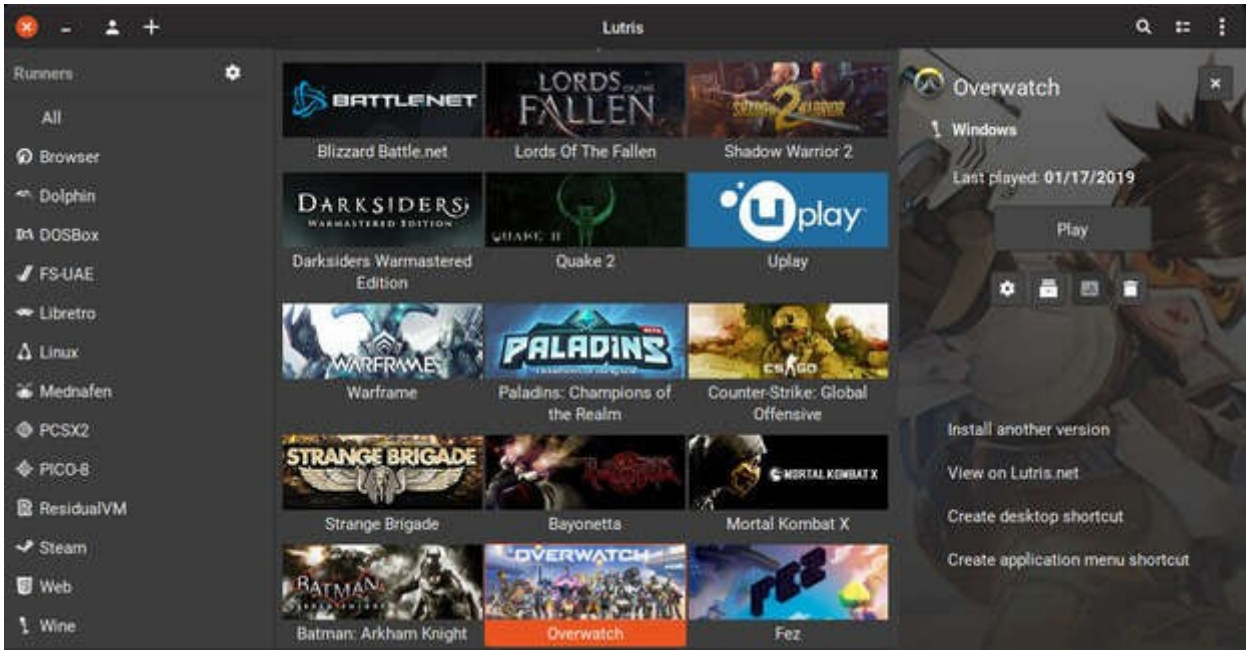


En pocas palabras, **Lutris es una aplicación que busca centralizar todos los juegos que tu equipo pueda ejecutar** (o al menos lo intentará) recopilando los juegos que tengas tiendas virtuales como **GOG, Humble Bumble o la todopoderosa Steam** y, además, gestionará los juegos que tengas instalados en tu equipo, tanto nativos como emulados.

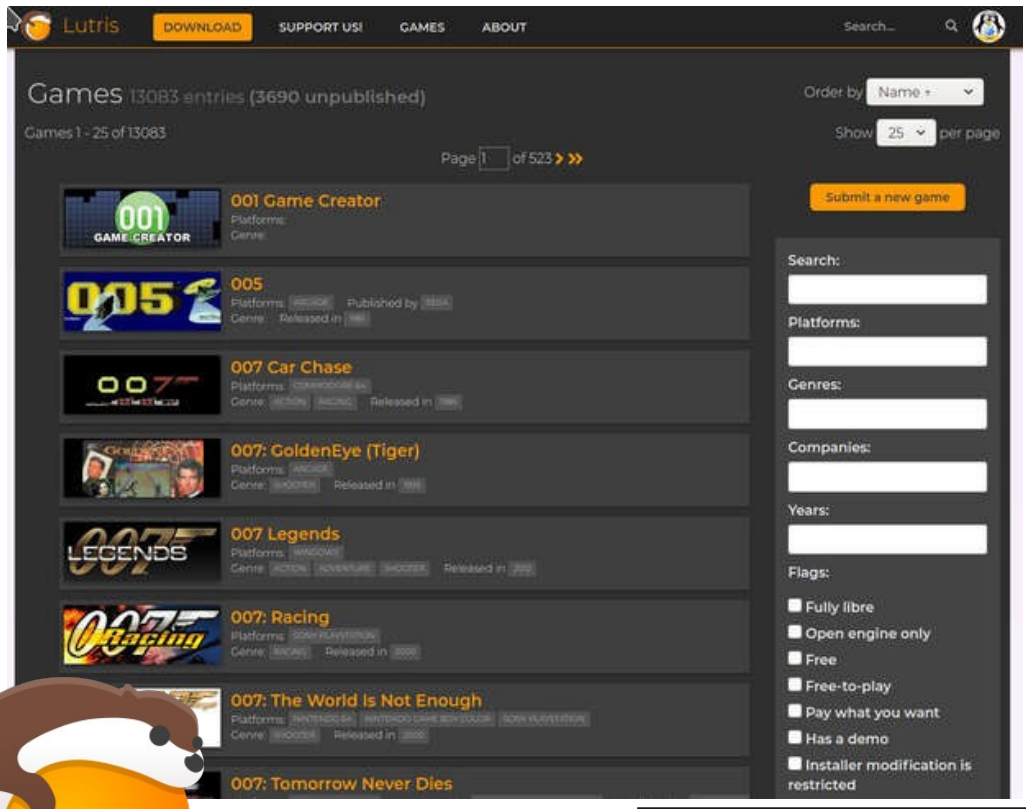


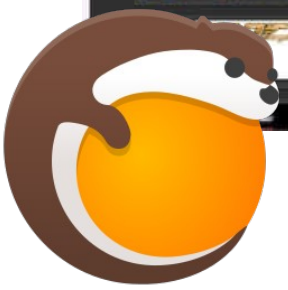
Para ello, tenemos las siguientes secciones:

- **Library,** que nos muestra los juegos instalados en nuestro sistema.
- **Sources,** que nos muestra los juegos que tenemos en nuestras tiendas online (Lutris, GOG, Humble Blumble y Steam)
- **Runners,** que nos muestra los diferentes motores que tenemos para lanzar juegos (Linux, MAME, Wine, Wine Steam, Snes, ZDoom, etc.)
- **Platforms,** que nos muestras las diferentes plataformas que tenemos ya activos y disponibles.



Quizás lo más interesante es navegar por la librería de Lutris y utilizar los scripts de la Comunidad que nos facilitan su instalación y optimización, con lo que nos hacen posible disfrutar de nuestra máquina linuxera como plataforma de juegos.





Autor: Baltolkien

Fundador y editor de KDE Blog. Profesor de ciencias en Secundaria, enamorado de su familia y del Software Libre.
Web: www.kdeblog.com

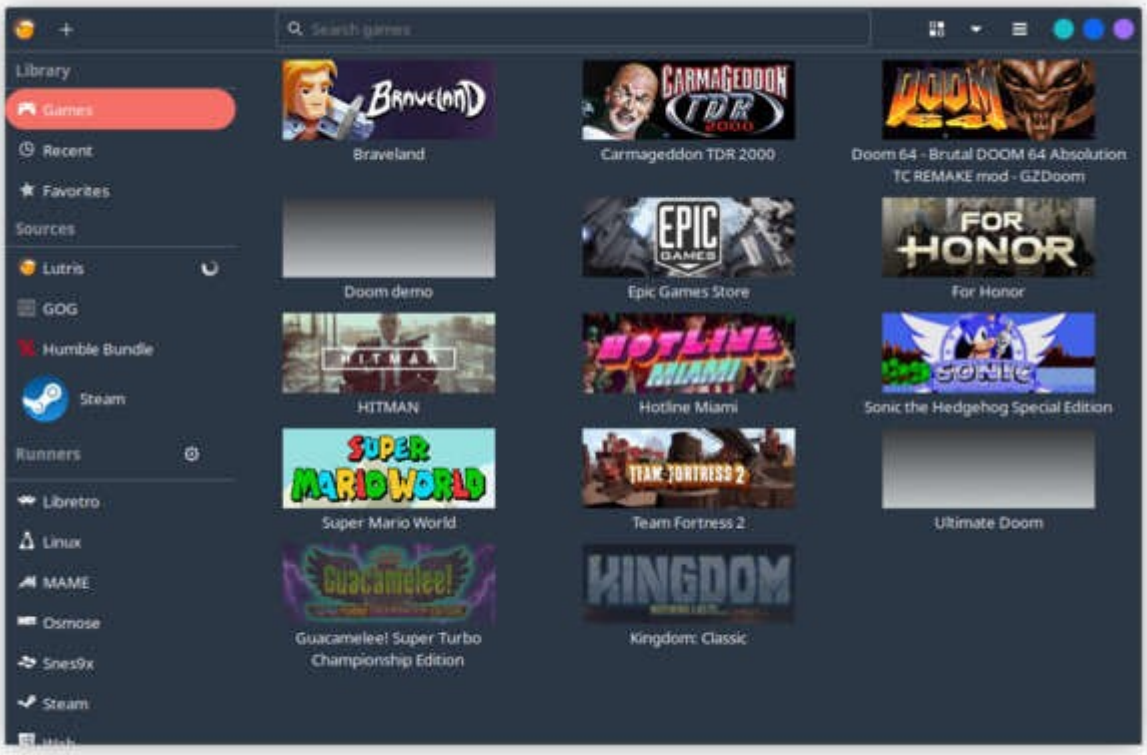
Cómo instalar Lutris, un gestor de juegos para Linux

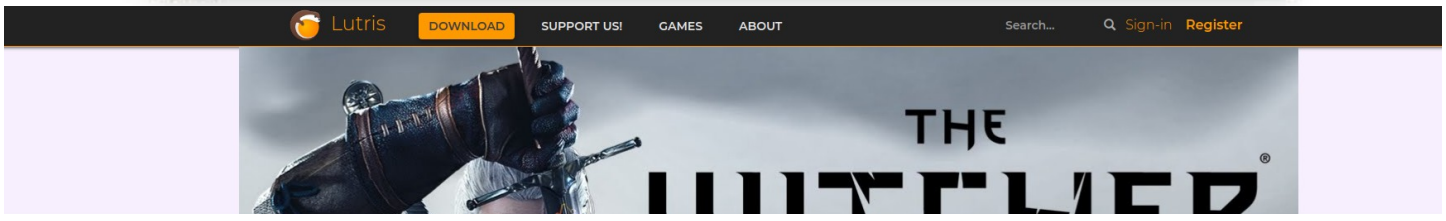
En primer lugar quiero recordar que es Lutris que no es más que una aplicación que busca centralizar todos los juegos que tu equipo pueda ejecutar recopilando los juegos que tengas tiendas virtuales como GOG, Humble Bumble o la todopoderosa Steam y, además, gestionará los juegos que tengas instalados en tu equipo, tanto nativos como emulados. Eso si, es una aplicación que se apoya en la creación de Scripts de instalación creados por la Comunidad, así que tiene la potencia de las personas detrás de ella.

Lutris es una aplicación que utiliza Python 3 y las librerías GTK, y su instalación es muy sencilla en casi todas las distribuciones pero creo que es bueno hacer una explicación para aquellos usuarios que todavía no tienen mucha experiencia en el mundo GNU/Linux.

Estamos ante un proyecto personal de **Mathieu Comandon** que necesita todo el apoyo que le podamos dar, así que os recomiendo visitar su **página de donaciones** si os gusta el proyecto y queréis financiarlo de algún modo (toda donación es bienvenida, por pequeña que sea).

Si nos vamos a la **página de descargas** de **Lutris** vemos que tenemos una explicación detallada para decenas de distribuciones, así que yo solo explicaré unas cuantas.





Ubuntu y derivadas
Empezamos con Ubuntu y derivadas (entre las que se encuentran KDE Neon o Kubuntu), Elementary y Linux Mint.

En este caso Lutris no está en los repositorios oficiales, así que seguiremos estos pasos:

Abrimos un terminal o una consola y escribir lo siguiente para añadir el repositorio a tu sistema, evidentemente te pedirá el password de superusuario (que se escribe pero no se ve):

```
sudo add-apt-repository ppa:lutris-team/lutris
```

A continuación vamos a escribir dos comandos, el primero actualiza tus repositorios y el segundo instala Lutris.

```
sudo apt update
sudo apt install lutris
```

Debian
Para los usuarios de Debian el proceso es similar. Abrir en una consola y escribir (creo que no debo ser más específicos ya que los usuarios de esta distribución no necesitan tantos detalles):

```
echo "deb http://download.opensuse.org/repositories/home:/strycore/Debian_10/ ./" | sudo tee /etc/apt/sources.list.d/lutris.list
wget -q https://download.opensuse.org/repositories/home:/strycore/Debian_10/Release.key -O- | sudo apt-key add -
apt update
apt install lutris
```

Otras distribuciones
En las siguientes distribuciones Solus, opeSUSE, Arch Linux, Manjaro, Fedora y Clear Linux, Lutris está en sus repositorios básicos, así que simplemente se debe ejecutar en consola la orden de instalación:

Solus:

```
sudo eopkg it lutris
```

openSUSE:

```
sudo zypper in lutris
```

Arch Linux y Manjaro:

```
sudo pacman -S lutris
```

Manjaro:

```
sudo dnf install lutris
```

Clear Linux:

```
sudo swupd bundle-add lutris
```

KDE Blog

Autor: Baltolkien

Fundador y editor de KDE Blog. Profesor de ciencias en Secundaria, enamorado de su familia y del Software Libre.
Web: www.kdeblog.com

Y, finalmente, en distribuciones como **Mageia, Gentroo, CentOS, Pop!_OS o Slackware** simplemente buscad en sus Centros de Software.

Y bien, con estas simples instrucciones ya lo tendréis, ahora simplemente falta configurar Lutris, pero eso ya es otra historia.

Seguridad y hardening en Linux

¿Qué es el hardening en Linux?
Muchos os preguntaréis si el hardening tiene que ver con endurecer el sistema (por la traducción directa del inglés) y efectivamente, es exactamente eso. Es el proceso de **reducción de vulnerabilidades** en el sistema y recoge una serie de **medidas de seguridad** que se deben cumplir con el objetivo de estar preparados ante un ataque informático.

Cada vez estamos más expuestos al exterior, cualquiera puede tener ya una página web o un servidor de ssh, y con ello ya puedes ser objeto de un ataque. Por este motivo, cuando eres tú el que debe administrar el equipo, deberás tomar en consideración el **seguir una serie de normas para asegurar y fortificar** lo máximo posible tu sistema.

Consejos esenciales para configurar la seguridad y el Linux Hardening
A continuación te voy a explicar algunas de las medidas más aconsejables para mejorar el hardening de tu equipo. Si consiguieses seguir todas ellas, podrías conseguir un equipo prácticamente invulnerable.. pero eso es casi imposible. Las tecnologías y los conocimientos de hacking malicioso cada vez son más elevados y cada día surgen nuevos problemas o brechas de seguridad que aprovechan para atacar.

Por eso es tan importante tener un conocimiento, al menos básico, de lo que es el hardening y el fortalecimiento de la seguridad de tu equipo.

Contraseñas seguras
Seguro que estarás pensando, otro con la misma tabarra de usar contraseñas seguras y no reutilizables... Pues si, efectivamente esta es una de las principales medidas de seguridad de un equipo informático o de una cuenta online. Y créeme que una contraseña segura, larga (mínimo 12 caracteres), con mayúsculas, minúsculas, números y signos de puntuación es muy difícil (no imposible) que sea descifrada.

En Linux hay varias formas de **obligar al usuario a que utilice una contraseña segura**, pero te voy a explicar una de ellas. En primer lugar deberás descargar e instalar el paquete *«libpam-pwquality»*.

```
root@aprendolinux:/home/jaime# apt install libpam-pwquality
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  cracklib-runtime libcrack2 libpwquality-common libpwquality1
Se instalarán los siguientes paquetes NUEVOS:
  cracklib-runtime libcrack2 libpam-pwquality libpwquality-common libpwquality1
0 actualizados, 5 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 244 kB de archivos.
Se utilizarán 977 kB de espacio de disco adicional después de esta operación.
```

Después puedes acceder al fichero de configuración en *«/etc/security/pwquality.conf»* y modificamos las entradas que se refieren al la longitud mínima de la contraseña y al tipo de contraseña. En este caso, el número 4 significa que deberá contener las 4 opciones mencionadas justo antes.

```
root@aprendolinux:/home/jaime# cat /etc/security/pwquality.conf
# Minimum acceptable size for the new password (plus one if
# credits are not disabled which is the default). (See pam_cracklib manual.)
# Cannot be set to lower value than 6.
minlen = 12
#
# The minimum number of required classes of characters for the new
# password (digits, uppercase, lowercase, others).
minclass = 4
....
```

Ahora puedes intentar crear un nuevo usuario en tu sistema, y te pedirá que cumplas con todos estos requisitos.

Revisar los permisos de ficheros y directorios por defecto
Hay una serie de permisos al **crear los ficheros y directorios que en Linux no se configuran por defecto correctamente**, y que deberíamos cuidar para que un posible atacante haga el menor daño en la máquina.

Por ejemplo, por defecto en Linux cuando creas un nuevo fichero se le asignan unos permisos con los que estás permitiendo que otro usuario del sistema pueda al menos leer ese fichero. Y cuando creas un directorio, pueden hacer un listado de los ficheros que contenga.

```
root@aprendolinux:/home/jaime/borrar# su - alex
alex@aprendolinux:~$ ls -las /home/jaime/
total 68
4 drwxr-xr-x 11 jaime jaime 4096 nov 16 23:56 .
4 drwxr-xr-x  4 root  root  4096 nov 16 23:58 ..
4 -rw-------  1 jaime jaime 3863 nov 16 23:58 .bash_history
4 -rw-r--r--  1 jaime jaime  220 mar 10  2021 .bash_logout
4 -rw-r--r--  1 jaime jaime 3526 mar 10  2021 .bashrc
4 drwxr-xr-x  3 jaime jaime 4096 nov 16 23:55 borrar
```

Para configurar correctamente estos permisos por defecto, habría que modificar el fichero *«/etc/bash.bashrc»* o *«/etc/profile»* y agregar la siguiente línea al final, para cambiar la máscara de los ficheros y directorios.

```
root@aprendolinux:/home/jaime/borrar# tail -n 1 /etc/bash.bashrc
umask 027
root@aprendolinux:/home/jaime/borrar# su - jaime
jaime@aprendolinux:~$ touch fichero
jaime@aprendolinux:~$ ls -l fichero
-rw-r----- 1 jaime jaime 0 nov 17 00:06 fichero
jaime@aprendolinux:~$ mkdir dir1
jaime@aprendolinux:~$ ls -ld dir1/
drwxr-x--- 2 jaime jaime 4096 nov 17 00:06 dir1/
```

Con esto hemos bloqueado completamente el acceso al resto de usuarios de la máquina a todos los ficheros que se creen en el futuro en nuestras cuentas de usuario.

Eliminar aplicaciones y software innecesario para mejorar la seguridad y hardening en Linux
Otra de las medidas aconsejables para mantener tu equipo lo más seguro posible es **desinstalar y eliminar toda aplicación que ya no estés usando o que no tenga sentido tener instalada**. Es decir, por ejemplo si tu equipo va a trabajar como un servidor Web o un almacén de backups.. ¿para qué necesita tener el servicio «cups» que sirve para la búsqueda y configuración de impresoras?

En Linux, si no tienes interfaz gráfico, puedes revisar los servicios que están instalados en el equipo (y que quizás no estás utilizando) con el siguiente comando:

```
root@aprendolinux:/home/jaime/borrar# service --status-all
 [ + ]  apparmor
 [ - ]  cgroupfs-mount
 [ - ]  console-setup.sh
 [ + ]  cron
 [ + ]  dbus
 [ + ]  docker
 [ - ]  hwclock.sh
 [ - ]  keyboard-setup.sh
 [ + ]  kmod
 [ + ]  networking
 [ + ]  procps
 [ + ]  rsyslog
 [ - ]  screen-cleanup
 [ + ]  ssh
 [ - ]  sudo
 [ + ]  udev
 [ - ]  x11-common
```

Configurar el acceso seguro por ssh a tu equipo
La mayoría de equipos Linux se pueden gestionar o al menor recibir accesos mediante el servicio SSH (Secure Shell), por eso es muy importante que lo mantengas lo más seguro posible. Este servicio está instalado por defecto en la mayoría de distribuciones, pero podemos instalar el servidor completo con el paquete *«openssh-server«*.

Para empezar la configuración de este servicio, que se encuentra en el directorio *«/etc/ssh/sshd_config«*, debe tener unos permisos restringidos al usuario *«root»*.

```
root@aprendolinux:/home/jaime# ls -l /etc/ssh/sshd_config
-rw-r--r-- 1 root root 3289 jul 28 15:33 /etc/ssh/sshd_config
root@aprendolinux:/home/jaime# chmod 600 /etc/ssh/sshd_config
root@aprendolinux:/home/jaime# ls -l /etc/ssh/sshd_config
-rw------- 1 root root 3289 jul 28 15:33 /etc/ssh/sshd_config
```

En este fichero hay varias entradas que es importante que estén habilitadas, como las siguientes:

- **UsePAM:** Debe estar a «yes» para que se habilite el uso del módulo de **autenticación PAM**.
- **ClientAliveInterval:** Establece un intervalo de tiempo de espera en segundos después del cual, si no se han recibido datos del cliente, se le enviará un mensaje para solicitar una respuesta. Debería establecerse entre 300 a 600 segundos (entre 5 y 10 minutos de inactividad).
- **ClientAliveCountMax:** Establece el número de mensajes que se pueden enviar a un cliente sin que se reciba ninguna interacción del cliente. Si se alcanza este umbral sshd desconectará al cliente y finalizará la sesión. Este valor debería ser menor que 3 para no permitir que una conexión se quede abierta por un tiempo excesivamente alto.
- **MaxAuthTries:** Especifica el número máximo de intentos de autenticación permitidos por conexión. Debería ser entre 3 y 6, para evitar los ataques por fuerza bruta.
- P**ermitEmptyPassword:** Establece si se permite el acceso al equipo con usuarios que tengan la contraseña vacia… obviamente esto no es nada seguro y habría que especificar a «no».

Configurar sudoers para el comando sudo

Ahora te voy a explicar como reforzar la seguridad y hardening de Linux configurando correctamente el super comando **«sudo«**.

Como bien sabes, el comando sudo es muy importante para la gestión y administración de tu equipo, pero a la vez es una herramienta peligrosa que se debe configurar correctamente. En caso de no hacerlo, podrías permitir a cualquier usuario ejecutar comandos como usuario «root» lo que podría ser desastroso.

El fichero que se debe cambiar para la configuración del comando sudo está en **«/etc/sudoers«**. De forma predeterminada, se debe usar el comando **«visudo»** para la edición del fichero de configuración.

En la parte inicial del fichero, hay que definir valores por defecto y estos son los valores que deberías tener en cuenta:

- **use_pty:** Obliga al usuario a tener una pseudo terminal abierta para lanzar comandos. Así no se podrían ejecutar comandos y dejarlos en segundo plano (lo que suele pasar cuando se infecta un equipo), ya que al terminar la sesión se cerrarán todas las terminales del usuario.
- **logfile:** Si se define un fichero, podemos conseguir que se almacenen todos los comandos que son ejecutados con sudo. Así, revisando frecuentemente este fichero, serás capaz de vez si hay ejecuciones no autorizadas (o al menos intentos).

```
Defaults  logfile="/var/log/sudo.log"
Defaults        use_pty
```

Cambiar permisos de usuario y de grupo en sudoers

Para permitir que un usuario especifico pueda ejecutar un comando con privilegios de root, utilizando el comando sudo, se debería poner una línea con este formato (Con NOPASSWD delante, podemos especificar que no pida contraseña):

```
usuario    <terminal>=(usuario:grupo)       Comando/s
jaime          ALL=(root:root)            NOPASSWD:/usr/bin/apt
```

Cuando queremos referirnos a un grupo, deberemos especificar delante del nombre del grupo el signo de porcentaje, es decir debería quedar de esta forma:

```
%grupo      <terminal>=(usuario:grupo)      Comando/s
%wheel            ALL=ALL                /usr/bin/mount
```

Instalar actualizaciones y parches

Una de las cosas buenas que tienen prácticamente todas las distribuciones de Linux es que tienen una comunidad muy activa, que siempre están buscando mejorar cada aplicación. Además, en caso de que se encuentre un error o un fallo de seguridad, los parches que lo solucionan no tardan mucho en publicarse.

Realizando una actualización periódica de tu sistema puedes estar seguro de que vas a estar protegido y a prueba de errores y fallos. Por lo que esta es una de las recomendaciones más sencillas, y a la vez más importantes que te puedo hacer.

Backups

Otra de las tareas que deberías realizar periódicamente en cualquier sistema que estés manejando es la de realizar copias de seguridad. Con esto, estás asegurando que todo el trabajo que realizas cada día, las ultimas configuraciones o actualizaciones de tus aplicaciones, o tus datos personales van a estar seguros.

Estos backups no es recomendable que los hagas en otra partición del mismo disco duro, ni en otro disco duro que tengas conectado internamente en el ordenador. Esto es debido a que si hubiera un problema de fallo electrico, o incluso un incendio o inundación, podrías disponer siempre de una copia de tus datos más importantes y así conservarlos.

Hay muchas herramientas que te facilitar el almacenaje de copias de seguridad en los diferentes sistemas operativos, por lo que te recomiendo que revises cual puede ser la mejor opción.

Si quieres conocer más sobre cómo funciona Linux y quieres saber cómo instalar tu propia distribución de Debian, puedes acceder de forma gratuita a mi **curso básico de Linux**

Autor: Jaime Pons

Ingeniero informático y con mas de 10 años de experiencia como administrador de sistemas.
Administrador de Aprendo Linux
Web: www.aprendolinux.com

Ansible: Un lenguaje universal

¿De dónde viene el nombre de Ansible?
Para empezar a hablar de esta tecnología, voy a comenzar explicando de dónde viene su nombre y qué tiene que ver con la tecnología de automatización Ansible.

El nombre lo tomó el creador de esta herramienta, **Michael DeHaan,** del sistema de comunicación instantáneo (más rápido que la velocidad de la luz) del hiperespacio. Este sistema fue inventado por Ursula K. Le Guin en su novela de 1966 "El mundo de Rocannon". El término (y el sistema de comunicación) fue también utilizado por Orson Scott Card en la conocida novela de ciencia ficción " El juego de Ender".

La idea de poner este nombre a un sistema de comunicación, es porque hace que los mensajes sean capaces de ser respondidos (**"answerable"**, palabra que fonéticamente es parecida a **Ansible**) en un tiempo relativamente razonable en el hiperespacio.

¿Qué es ansible?
Si consultamos la definición de su página web oficial, podemos decir que "Ansible es un **lenguaje universal**, que desentraña el misterio de cómo se realiza el trabajo", sin duda una definición demasiado etérea.

En Wikipedia encontramos otra definición que dice que Ansible se trata de "una plataforma de software libre para **configurar y administrar ordenadores**". A mi esta definición me gusta mucho más.
El autor de Ansible, como te comenté antes, es Michael DeHann y el repositorio lo puedes encontrar en " **https://github.com/ansible/ansible**". El proyecto tiene una gran comunidad activa detrás y sigue creciendo día a día.

Además combina instalación multi-nodo, es decir: permite desplegar configuraciones de servidores y servicios por lotes, ejecuciones de tareas ad hoc y administración de configuraciones. Adicionalmente, Ansible es categorizado como una herramienta de **orquestación o automatización.**

¿Qué es la orquestación?
Los sistemas de orquestación son sistemas que automatizan el despliegue, la gestión, el escalado, la interconexión y la disponibilidad de nuestras aplicaciones o servidores.
Desde mi experiencia os puedo decir que cuando tenemos que trabajar con multitud de dispositivos, no solo servidores, sino también dispositivos de red como switches o routers, o incluso trabajar con plataformas de la nube, como AWS o GCP, Ansible nos puede ayudar muchísimo, en nuestro día a día.
Con permiso de otros sistemas como Puppet o Chef, Ansible se ha convertido en la herramienta de automatización más popular. Está disponible para prácticamente todas las distribuciones de Linux y para MacOs. En Windows todavía no lo podemos usar de forma nativa, pero se puede usar con máquinas virtuales o con Windows **Subsystem for Linux (WSL**).

Ventajas de Ansible
¿En qué se diferencia de otras herramientas similares?
- No necesita agentes o aplicaciones extras instaladas en los nodos gestionados.
- Su instalación es muy sencilla.
- Compatibilidad con la mayoría de los elementos de nuestra infraestructura.
- Soporta la mayoría de las distribuciones.
- Configuraciones sencillas en lenguaje YAML.
- Flexibilidad (API, módulos, plugins) y portabilidad.
- Facilidad de uso.

Aunque no todo son ventajas, también tiene desventajas:
- Es menos potente que otros sistemas similares, en lo que respecta a la administración de configuraciones.
- No trabaja bien con gran cantidad de elementos para administrar (cuando hablamos de miles), ya que en estos casos, requiere configuraciones avanzadas para obtener un buen rendimiento.
- Al tener tantos módulos disponibles, estos no siempre están actualizados... pero gracias a su gran comunidad es raro encontrar componentes no actualizados.

Es completamente gratuita, ya que utiliza una licencia GNU GPL v3 y como ya os he dicho, también permite trabajar con los proveedores de la nube, como AWS, Azure o Google Cloud Platform.

Fue tan popular que Ansible fue adquirida por la compañía Red Hat en el año 2015

Arquitectura de Ansible
Como os dije previamente, en Ansible sólo existen dos tipos de servidores, no diré roles porque como veremos más adelante, los roles en ansible son para otras cosas:
- **Controlador**: La máquina desde la que comienza la orquestación, debe tener instalado Ansible, así como Python mínimo 2.7 y ssh-server.
- **Nodo gestionado**: Es manejado por el controlador a través de SSH, y su único requisito es tener instalado al menos Python 2.7.

Para la gestión de los nodos, Ansible despliega módulos mediante conexiones de SSH. Los módulos son guardados temporalmente en los nodos y se comunican con la máquina de control. Como las acciones de Ansible se ejecutan en la máquina remota, no consumen recursos locales en el controlador.

El diseño de Ansible incluye estas premisas:
- **"Mínimo por naturaleza".** Los sistemas de administración de infraestructuras TI no deben tener dependencias adicionales.
- **"Consistente".** Viene con una amplia biblioteca de componentes ya cargados, lo que te he nombrado antes como módulos.
- **"Seguro".** Ansible no instala agentes, que podrían ser vulnerables en los nodos.
- **"Alta confiabilidad"**. El modelo de **idempotencia** es aplicado para las instalaciones y configuraciones, para prevenir efectos secundarios en la ejecución repetitiva de scripts.
- **"Suave curva de aprendizaje"**. Los playbooks usan el lenguaje simple YAML.

Conceptos básicos de Ansible

Veamos un resumen de los principales conceptos a tener en cuenta.

- **Facts**: Información de los clientes o nodos gestionados que se obtienen en cada ejecución.
- **Inventario**: Incluye información, que puede ser estática o dinámica, de los nodos gestionados.
- **Módulos**: Son las bibliotecas que se utilizan para controlar elementos como ficheros, servicios, paquetes o comandos. Estos se copian al nodo gestionado para que ejecute la tarea indicada.
- **Nodo (gestionado)**: Objeto o dispositivo que se desea administrar, que puede ser un servidor, un router u otros elementos.
- **Tasks**: Las tareas a realizar en los clientes especificados en el Playbook.
- **Playbook**: Es el fichero donde se deben definir la lista de tareas que queremos realizar sobre un conjunto de hosts clientes.
- **Roles**: Es una agrupación de tareas, playbooks, ficheros y plantillas, que pueden ser reutilizados.

Autor: Jaime Pons

Ingeniero informático y con mas de 10 años de experiencia como administrador de sistemas.
Administrador de Aprendo Linux
Web: www.aprendolinux.com

Curso de Vim: ¿Cómo salir del editor Vim?

¿Estás atascado en el editor Vim y no sabes cómo salir? ¡No eres el único!

Si por algún motivo **se ha abierto un archivo en el editor Vim y quieres salir pero no encuentras cómo hacerlo,** ahora veremos la solución. Pero no te frustres no eres el único al que le pasa eso.

La **web Stack Overflow** es una de las mayores comunidades y con mayor reputación de desarrolladores donde buscan soluciones, ayuda sobre cómo resolver diferentes problemas a los que se enfrentan programadores de muchos ámbitos y lenguajes.

En esa comunidad donde se dan y encuentran respuestas a complejos problemas de programación, **ha habido una consulta que ha llegado al hito de ser vista por un millón de visitantes. ¿Qué pregunta es esa?**

Pues ni más ni menos que la cuestión de un usuario que preguntaba a la comunidad **¿cómo salir del editor Vim?** Una pregunta bastante trivial... o no tanto

Durante años esa pregunta ha sido una broma recurrente entre los más geeks de internet, con diversos memes, e incluso el mismísimo **San iGNUcio aka Richard Stallman recurre a bromas con Vi**.

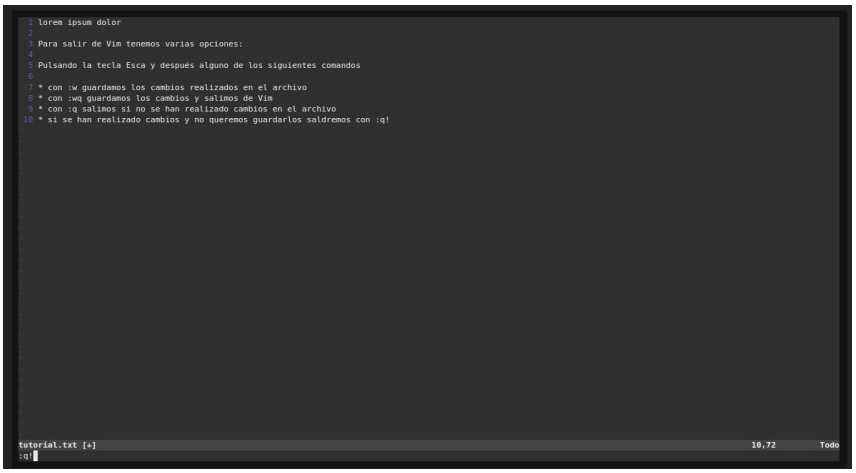
Tal como detallan en **Stack Overflow** la pregunta ha conseguido el millón de visitas, de diferentes países y visitada por usuarios con diferentes aspectos técnicos, así que algo debe de tener de santa el agua cuando la bendicen.

No entraremos en diatribas entre defensores de **Emacs**, de Vi o de cualquier otro editor de texto basado en la consola. Reconozcan esos **«Vim lovers»** que su editor preferido requiere de una cierta curva de aprendizaje y que es poco intuitivo para aquellos que recalan en él sin venir a cuento.

Para evitar esos dolores de cabeza que nos puede causar Vim a la hora de salir, es buena opción utilizar el editor nano o micro.

Pero no demoremos más la solución. **Aquí está la respuesta a cómo salir del editor Vim:**

- Pulsar la tecla **Escape**. El cursor se irá a la parte inferior (la zona de comandos)
- Pulsar la secuencia de teclas **:q**
- O también puedes pulsar **:q!** para salir del editor sin guardar los cambios
- Pulsar la tecla **Enter**



Recopilando y resumiendo los aportes de los comentarios. Primero pulsamos Esc para situarnos en el buffer inferior en modo normal, y:

- **:q** ⟶ sale del archivo, si no tenemos cambios sin guardar sale sin más.
- **:q!** ⟶ sale del archivo descartando los cambios no guardados.
- **:w** ⟶ guarda los cambios del archivo, pero no sale de Vim para seguir editando el archivo.
- **:wq** ⟶ guarda los cambios y sale de Vim.
- **:x** ⟶ igual que el anterior.
- **ZZ** ⟶ igual que **:x**
- **ZQ** ⟶ **i**gual que **:q!**

Con esa secuencia habremos salido del editor Vim. No desesperes ¿ves como no era para tanto?

De todas formas, te animo a que como yo, le des una oportunidad al editor Vim, tiene muchas cosas buenas que ofrecer si editas mucho texto o código. Tiene una pequeña curva de aprendizaje, para lo básico, pero aprendido eso, después es cuestión de usarlo y habituarse a sus múltiples opciones al alcance de los dedos.

Hace unos meses me auto-propuse aprender poco a poco a manejar Vim, y la verdad es que el reto ha sido todo un descubrimiento interesante.

También tienes una guía en español para aprender Vim desde lo básico hasta conceptos más complejos: **https://victorhck.gitbook.io/aprende-vim/**

Aprende Vim

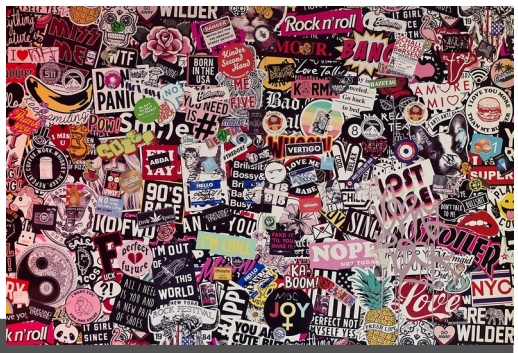
Planeta Libre

Victorhck in the free world
omnia sunt communia!

Autor: Victorhck

Administrador de su blog personal sobre openSUSE, GNU/Linux y software libre

Web: www.victorhckinthefreeworld.com



Curso de Vim: Mejora tu experiencia usando el editor Vim

Con estas sencillas configuraciones puedes hacer que el editor **Vim (Vi Mejorado)** sea más productivo y darle una segunda oportunidad.

En el anterior articulo gracias a **Victorhck** ya has podido leer sobre cómo salir del editor Vim. Pero supongamos que además de eso queremos darle una segunda oportunidad y utilizarlo como tu editor de texto principal, **con estas sencillas configuraciones puedes mejorar esa experiencia de uso**.

Más allá de lo anecdótico de esa rivalidad entre usuarios de editores de texto **Emacs y de Vi (o Vim)**, o de que **Stallman lo llame el editor del diablo**, primero he de puntualizar que no uso ninguno de forma intensiva. A mí me sirven cosas más simples y pequeñas como micro o nano.

Pero sí he de decir que en varias ocasiones me he encontrado a desarrolladores que me han dicho que usaban **Vim**. Así que como dice el refrán: **«algo tiene el agua cuando la bendicen»**.

Es decir, que si mucha gente lo usa será porque le resulta una herramienta interesante, quizás es hora de descubrir este editor más en profundidad. Si te animas a probarlo, quizás estas pequeñas configuraciones hagan que sea más sencillo ese cambio hacia Vim.

Este artículo es una traducción/adaptación de un artículo en inglés escrito por **Girish Managoli publicado en la web opensource.com bajo licencia CC-by-sa**, que me pareció interesante compartir por aquí. Empezamos...

Las configuraciones que vamos a ver a continuación deberás incluirlas en el archivo oculto de configuraciones de vim que hay en tu **/home llamado .vimrc**.

Si no existe deberás crearlo y editarlo e ir añadiendo de todas las opciones que veamos las que creas que son más interesantes.

Las configuraciones que veremos cubrirán aspectos como:

- **Los tabuladores y el sangrado de texto**
- **El formato y cómo mostraremos el texto**
- **La búsqueda de texto**
- **Cómo navegar por el texto**
- **Algunas otras variadas**

1.- Tabuladores y sangrado de texto

Para **alinear automáticamente** la sangría del texto de una línea en un archivo

```
set autoindent
```

El **sangrado inteligente** utiliza la sintáxis y el estilo del código para alinear el texto

```
set smartindent
```

Para establecer el **número de espacios para mostrar por cada pulsación del tabulador**:

```
set tabstop=4
```

Para establecer el **número de espacios para mostrar en una "shift operation" (como '>>' o '<<')**:

```
set shiftwidth=4
```

Si deseas **utilizar espacios en vez de tabulaciones**, esta opción inserta espacios cuando se pulsa la tecla tabulador.

```
set expandtab
```

2.- El formato y cómo mostraremos el texto
Para **mostrar los números de las líneas**:

```
set number
```

Para **«cortar» el texto cuando este exceda el ancho máximo de la línea**:

```
set textwidth=80
```

Para **«cortar» el texto pero basándose en el número de columnas desde el lado derecho**:

```
set wrapmargin=2
```

Para identificar y que **resalte la apertura y cierre de paréntesis o corchetes** cuando el cursor pase por encima de uno de estos:

```
set showmatch
```

3.- Búsqueda
Para **resaltar el término buscado** en un archivo:

```
set hlsearch
```

Para realizar **búsquedas incrementales** mientras estás escribiendo:

```
set incsearch
```

Para **realizar búsquedas ignorando si están en mayúsculas o minúsculas**:

```
set ignorecase
```

Para buscar sin tener en cuenta la configuración anterior **ignorecase** cuando ambos **ignorecasey smartcase** están activados y el patrón de búsqueda contiene mayúsculas:

```
set smartcase
```

Por ejemplo, imaginemos que el texto contiene este testo
test
Test

Activando ambos, tanto **ignorecase** como **smartcase**. Una búsqueda del término "test" encuentra y resalta ambas:
test
Test

Una búsqueda de "Test" encuentra y resalta sólo la segunda linea:
test
Test

4.- Cómo navegar por el texto
Para mejorar la experiencia visual , quizás puedes preferir que el cursor este en alguna parte del texto, quizás por el medio, en vez de en la primera línea. **La siguiente opción ubica el cursor en la línea 5.** Modifícalo a tu gusto.

```
set scrolloff=5
```

Para **mostrar de forma permanente la barra de estado en la parte inferior de la pantalla de vim,** donde se mostrará el nombre del archivo, los números de fila y columna, etc:

```
set laststatus=2
```

5.- Opciones variadas

Para **inhabilitar la creación automática de un archivo de respaldo.** Cuando esta opción está activada, vim crea un «backup» del archivo antes de la edición. Los archivos de backup llevan este símbolo (~)al final del nombre del archivo.

```
set nobackup
```

Para inhabilitar la creación de un archivo «swap«: Cuando esta opción está activada, vim crea un archivo «swap» que existe hasta que empiezas a editar el archivo.

El archivo «swap» es utilizado para recuperar el archivo si el equipo o el programa se quedan «colgados» o fallan Los archivos «swap» son ocultos y empiezan con un **.** y terminan con **.swp**.

```
set noswapfile
```

Supongamos que necesitas **editar múltiples archivos en la misma sesión de vim** y poder cambiar entre ellos. Una característica molesta es que no es evidente que el directorio de trabajo es el que se abrió en el primer archivo.

A menudo es útil cambiar automáticamente el directorio de trabajo al del archivo que se está editando. Para habilitar esta opción:

```
set autochdir
```

vim mantiene un historial de acciones para deshacer. De manera predeterminada este historial es activo sólo hasta que el archivo es cerrado.

vim incluye la opción de mantener ese historial incluso cuando el archvo es cerrado, lo que significa que puedes deshacer acciones incluso después de haber cerrado, guardado y vuelto a abrir el archivo.

El archivo es oculto y tiene la extensión **.un~**

```
set undofile
```

Para establecer sonidos de alerta (por ejemplo sonará una alerta si tratas de hacer «scroll» más allá de la última línea:

```
set errorbells
```

Si lo prefieres puedes **establecer alertas visuales:**

```
set visualbell
```

Estas dos últimas opciones particularmente no me gustan.

Es posible **activar un comando sólo para un archivo en concreto y no para toda la configuración global.** Para hacer esto, abre el archivo y escribe **:** seguido de **set** y el comando que deseas. Esa configuración será efectiva sólo para esa sesión de edición del archivo.

También puedes tener **información más detallada de un comando** si escribes dentro de vim

```
:help autoindent
```

Esto nos dará información y ayuda sobre **autoindent**, puedes hacer lo mismo con cualquier otro comando.

También tienes una guía en español para aprender Vim desde lo básico hasta conceptos más complejos: **https://victorhck.gitbook.io/aprende-vim/**

Victorhck in the free world
omnia sunt communia!

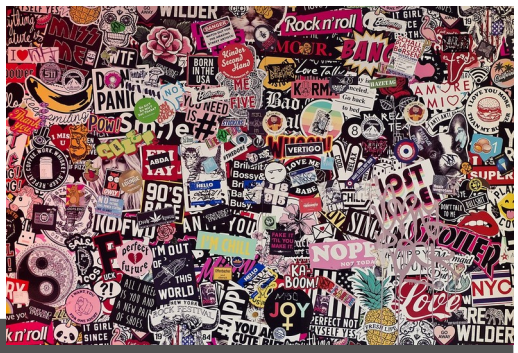
Aprende Vim

Planeta Libre

Autor: Victorhck

Administrador de su blog personal sobre openSUSE, GNU/Linux y software libre

Web: www.victorhckinthefreeworld.com



SPEED DREAMS, Un simulador libre de carreras de coches.

Existen muchos y muy buenos juegos de conducción dentro del mundo del Open Source y el Software Libre. Empezando por el más conocido, el divertidísimo **SuperTuxKart**, y pasando por **Stunt Rally**, **Vdrift**, **Yorg** o **Trigger Rally**, por enumerar algunos títulos. Pero si bien estos son buenos ejemplos de la vena más árcade del género, los jugadores más exigentes demandan juegos más centrados en la simulación y con unas físicas más completas. Juegos donde el disputar la primera posición no es un camino de rosas y las reacciones y sensaciones que se tienen al conducir estos coches virtuales se asemejen lo máximo posible a la realidad.

De esta premisa nació hace un par de décadas **TORCS**, el "papá" del juego que hoy nos ocupa. Debido a ciertas diferencias entre la dirección a tomar en este juego por parte de los desarrolladores originales, y algunos de los nuevos, finalmente el proyecto se escindió, naciendo en 2008 un nuevo Fork llamado en principio TORCS-NG (new generation) y cambiando finalmente al **nombre definitivo de Speed Dreams**. Bastantes años han pasado, y muchas lineas de código se han escrito desde entonces, convirtiendo a Speed Dreams en un **juego muy diferente y muchísimo más avanzado que su predecesor**, pero no vamos a perder más el tiempo describiendo sus diferencias, sinó que vamos a daros una visión general de este admirable proyecto.

Antes de continuar conviene aclarar que en Speed Dreams encontraremos un **juego completamente en 3D**, escrito principalmente en **C++** y publicado bajo la **GPL v2** y la **Free Art License**. El motor gráfico por defecto es **Plib**, aunque en las últimas versiones, se está introduciendo de forma experimental **OpenSceneGraph** (OSG), conocido también por ser el motor de otro gran simulador libre como es **FlightGear**. Este nuevo motor permite una **mayor calidad y un mejor rendimiento gráfico**.

Mayor calidad y mejor rendimiento son las caracteríticas de OpenSceneGraph

Nada más arrancarlo encontraremos un menú donde podremos además de comenzar a jugar, configurar según nuestro gusto o posibilidades aspectos como los gráficos, efectos de postprocesado, el sonido, la dificultad de las IAs, o el motor de físicas a utilizar, entre otras muchas cosas.

También podremos crear nuestros **perfiles de jugadores** donde además de configurar la dificultad de conducción, podremos personalizar nuestros controles, siendo admitidos dispositivos de todo tipo, como el **teclado, el ratón, joysticks, gamepads o incluso volantes**, poseyendo estos últimos un más que convincente sistema de **Force Feedback** que permite al jugador sentir la conducción del vehículo con mucho más detalle. Otra característica interesante a tener en cuenta en este menú de "**Players & Controls**" es la posibilidad de introducir las credenciales de **nuestra cuenta** en el "**Masterserver**"para registrar nuestros tiempos y datos de la partida con los diferentes coches y circuitos de Speed Dreams, pudiendo luego consultarlos y compararlos con los de otros jugadores a lo largo del mundo.

Pero vamos ya a la "chicha" de Speed Dreams, que son las carreras. Tan pronto como **pulsamos "Race" en el menú principal** veremos los diferentes modos en los que es posible disfrutar el juego. En primer lugar encontraremos "**Practice**", donde como su nombre indica podremos practicar nosotros solos en una pista y un vehículo de nuestra elección. También, como en el resto de modos del juego, podremos configurar el número de vueltas, tiempo o kilómetros que recorrer en esta prueba; y algo muy interesante **escoger el momento del día (o de la noche) en el que queremos correr, así como las condiciones climáticas**, permitiéndonos el juego rodar incluso bajo la lluvia. Esta última característica **permite también usar el tiempo real en la ubicación del circuito**, por lo que si esta diluviando en la vida real, por ejemplo en Sao Paulo (Brasil), en el juego también lo hará en el circuito homónimo si así lo hemos configurado.

Otro modo es "**Quick Race**", donde al igual que en el modo anterior configuraremos el circuito, nuestro coche, y las condiciones climáticas y resto de opciones, pero con la particularidad de que **en esta ocasión tendremos que añadir contrincantes a nuestra parrilla de salida**. Estos pueden ser tanto **humanos**, pudiendo **dividir la pantalla hasta en 6 jugadores locales;** como **contra IAs**, en este juego llamadas Robots. En cuanto a estos últimos, existen de diversos tipos, siendo unas más lentas pero regulares, u otras más rápidas pero que arriesgan más.

Mayor calidad y mejor rendimiento son las características de OpenSceneGraph

En el modo "**Single Event**" podremos realizar un **fin de semana completo** con una determinada categoría de coches en un circuito de nuestra elección. Será posible configurar las diferentes sesiones de Práctica, Cualificación, Calentamiento y Carrera; y por supuesto tendremos que disputarlas, siendo las de los robots simuladas, excepto la propia carrera.

"**Championship**" será el modo que nos permitirá correr en varios fines de semana, recibiendo puntos por cada uno de los disputados, hasta la finalización, donde el que más tenga se llevará el campeonato.

También podremos enrolarnos en el modo "**Career**" donde simularemos la vida de un piloto corriendo en diferentes categorías, coches y pistas. Sería lo más parecido al modo "Carrera Profesional" que podemos encontrar en otros juegos del género.

Por último dispondremos de un modo "**Online**" que nos permitirá crear y unirnos a partidas con otros jugadores en red. Conviene aclarar que está **modo experimental** y solo funciona más o menos bien en redes privadas (LAN). A través de internet presenta problemas de sincronización que aun no han sido corregidos y sigue en desarrollo.

En cuanto a las carreras en si, una vez pulsamos el botón "**Start**", cargaremos todo aquello que hemos configurado previamente y comenzaremos por fin a jugar presentándose ante nosotros un juego con unos **gráficos en 3D de buena factura** y diversa información mostrada en un **HUD completamente configurable** donde podremos consultar tiempos, posiciones en carrera, velocidad y revoluciones, estado del vehículo, nivel de combustible, fuerzas y estado y agarre de las ruedas, entre otras cosas. También podremos utilizar **multitud de cámaras**, desde las **subjetivas** en el cockpit o el morro del coche, a las de **tercera persona**, desde la trasera de nuestro vehículo. Pero no solo se limitará a eso, pues podremos ver nuestro coche (o el coche que seleccionemos) desde un helicóptero, en modo televisión, desde los laterales, al lado de las ruedas... También podremos **ralentizar o acelerar el tiempo**. Esto es especialmente util si realizamos carreras de bots y queremos ver el resultado rápido.

Speed Dreams dispone de multitud de variadas cámaras, como la de Televisión

Si no hemos configurado previamente los controles del juego podremos arrancar nuestro coche usando las flechas (arriba para acelerar y abajo para frenar). La dirección se controlará con el ratón. En un principio este método de control puede parecer muy difícil e incómodo, pero a medida que uno se acostubra puede llegar a ser altamente preciso, habiendo jugadores verdaderamente expertos con esta configuración. Como antes os comentamos podemos usar otros mandos más adecuados para controlar el coche, siendo obviamente un volante con Force Feedback el mejor, y como conseguiremos la experiencia más realista. A medida que vamos conduciendo en los diferentes circuitos, con cada uno de los coches que conforman el juego, apreciaremos las **grandes diferencias que podemos encontrar entre conducir coches** tan diferentes como como un "Monoposto 1" o un "Supercar".

Seremos conscientes también de la dificultad que entraña la conducción de los coches, ya que como hemos mencionado anteriormente, **este título está enmarcado en el campo de la simulación**, y pretende reflejar con la mayor veracidad posible las reacciones de un vehículo real en una carrera, con todo lo que ello conlleva. Por lo tanto no nos vamos a encontrar un juego donde vayamos a entrar en todas las curvas derrapando y sin aminorar, o donde podamos chocarnos sin consecuencias con nuestros competidores. Eso no quiere decir, para nada que sea un juego aburrido, aunque si **exige cierta paciencia al principio**. Al igual que otros títulos de simulación, lo divertido está en conseguir dominar el coche y sacar lo máximo de él en todo momento.

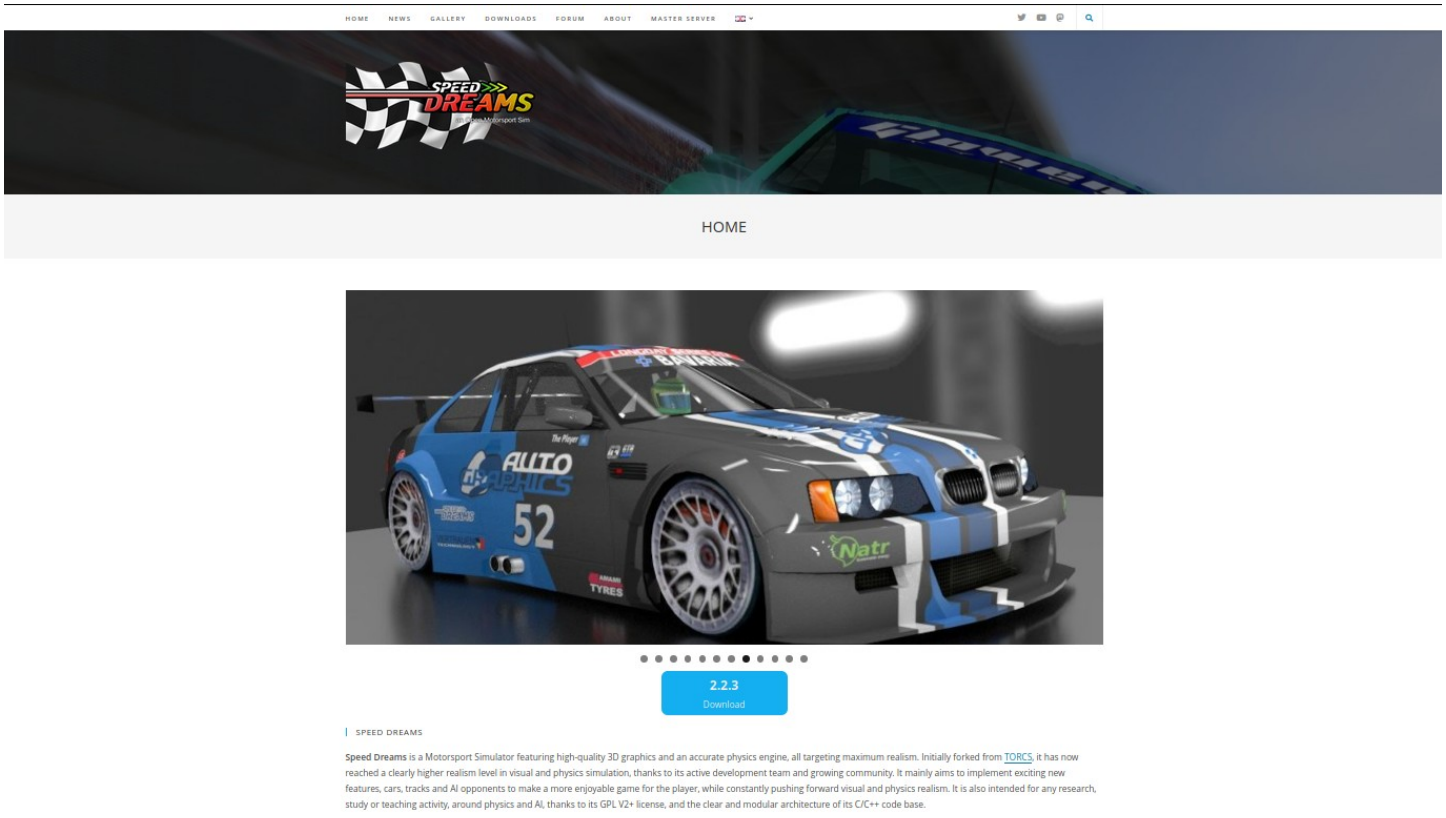
En cuanto a esto, existen **mutitud de factores que influyen en la condución** más allá de la velocidad o las características de la pista en la que rodemos. Por poner ejemplo, la aceleración, el agarre y temperatura de los neumáticos, la suspensión o incluso la meteorología, condicionarán nuestra conducción y tendremos que ser conscientes de ello. También **podremos mejorar y personalizar multiples aspectos de nuestros coches en la sección de "Setup" del garage**, donde ajustaremos los automoviles mecánicamente para obtener el rendimiento que mejor se adapte a nuestro estilo de conducción y al circuito en el que vayamos a rodar. Podremos por ejemplo regular el frenado, la suspensión, la transmisión, la aerodinámica...

Hablando de **los coches**, estos **están normalmente inspirados en modelos reales**, y están agrupados en multiples categorías, muy diferentes entre si, desde **superdeportivos** de calle (Supercars), **monoplazas** (Monoposto 1, Monoposto America, Monoposto 5), **Gran Turismo** (Long Day Series GT1, Long Day Series GT2), **Rally**, e incluso **históricos** (1936 Grand Prix, 1967 Grand Prix). Encontraremos también otras categorías aun en desarrollo, como por ejemplo Prototipos de Resistencia o Stockcars de los 70.

Speed Dreams dispone de multitud de variadas cámaras, como la de Televisión

También podremos rodar en **montones de circuitos**. **Muchos estarán basados en circuitos reales**, como por ejemplo Karwada (Suzuka), Corkscrew (Laguna Seca) o Salzfelden (Salzburgring); y otros serán completamente ficticios, como Espie, Garguree o Alicante. Estarán divididos en diversos tipos, como circuitos de "**Grand Prix**", "**Dirt**", "**Road**" o "**Speedways**". En ellos, además de los diferentes paisajes o elementos del escenario, encontraremos pistas rápidas, circuitos intrincados, urbanos, ovales, carreteras de montaña, pistas con multitud de cambios de rasante e incluso de tierra para las carreras de Rallycross. En todos ellos habrá una **zona de boxes** donde podremos reparar nuestro vehículo, cambiar neumáticos o rellenar combustible.

Esto sería a grandes rasgos todo lo que vais a encontrar en Speed Dreams, pero si quereis más información **os recomendamos que visiteis su nueva página web**, recientemente lanzada con su nuevo dominio **https://www.speed-dreams.net/**, donde además podreis estar al tanto de las últimas noticias del proyecto, consultar **interesantes enlaces** como los de su **Wiki** o sus redes sociales; o encontrar los **enlaces de descarga** del juego.



En su completa web encontrareis gran cantidad de información y por supuesto sus descargas

En cuanto a estos últimos, este **juego multiplataforma** puede ser instalado en nuestro sistema de diversas formas, siendo el más sencillo el formato **AppImage**. Por supuesto podremos **compilar el código fuente** si así lo deseamos, usar el **PPA de Xtradeb** para Ubuntu y derivadas, o instalar el Flatpak a través del **repositorio de Flathub.**

Ahora ya solo falta que te pongas al volante y disfrutes a lo grande de Speed Dreams.

Jugando En Linux

Autor: Leillo1975

Administrador de sistemas y redes de profesión PCero desde el 1992, Linuxero desde el 2005

Web: http://www.oblogdeleo.es/
https://www.jugandoenlinux.com/

Primeros pasos con Rust

¿Que es Rust?
Rust es un lenguaje de programación, relativamente, reciente, y que nació bajo el manto de la fundación Mozilla.
Se trata de un lenguaje que se caracteriza por su rapidez, fiabilidad y, sobre todo, por su **eficiente** gestión de la memoria.

Un poquito de historia
Cuando he indicado que se trataba de un lenguaje de programación relativamente reciente, me refería a que la primera versión estable, es decir, *Rust 1.0*, vio la luz en Mayo de 2015, aunque sus orígenes son anteriores.
La realidad es que este lenguaje nació, en 2006, como un proyecto personal de *Graydon Hoare*, un empleado *Mozilla* por aquel entonces. Sin embargo, *Mozilla* empezó a esponsorizarlo en 2009 y lo anunción en 2010. Ese mismo año 2010, nació el compilador implementado en Rust, y en 2011 consiguió compilarse a si mismo.
En Agosto de 2020, y como consecuencia del impacto de la pandemia, en la que actualmente nos encontramos, Mozilla despidió a 250 de sus 1000 empleados. La mayoría de estos eran trabajadores relacionados directamente on el equipo de Rust.
Esto llevó a la creación de la *Rust Foundation*, cuya primera misión fue hacerse con licencias, créditos, marca registrada, dominios, etc.
A principios de 2021, se anunció la creación de la *Rust Foundation* por parte de las cinco compañías fundadoras, entre las que por supuesto se encontraba Mozilla. Pocos meses después Google incorporó Rust a Android, como una alternativa a C/C++.
El 8 de diciembre de 2021, **Rust** se convirtió en un lenguaje oficial para el desarrollo de Linux.
Como ves, se trata de un lenguaje, relativamente nuevo, sobre todo comparados con otro, pero que sobre todo tiene un **muy rápido crecimiento** e integración en casi cualquier ámbito tecnológico.

¿Pero que tiene Rust?
Llegados a este punto, seguro que te estás preguntando, ¿pero que tiene Rust para que haya tanto movimiento en torno a el?¿Porque Rust se ha convertido en el lenguaje mas querido por los desarrolladores?
Quizá estés pensando que exagero cuando te digo que Rust es el lenguaje mas querido por los desarrolladores. Pues no exagero ni un ápice. En el *Stack Overflow Developer Survey* de 2021 salió elegido como *The most loved programing language*. No fue el único. Pero lo verdaderamente reseñable, es que ha **ganado esta distinción en los últimos seis años**, lo que dice mucho en favor de Rust.

Algunas características de este lenguaje de programación
Entre las características que hacen interesante Rust como lenguaje de programación, seguro que encontrarás, allá por donde consultes en Internet, al menos estas tres,

- **Rendimiento**. Se trata de un lenguaje **rápido** y **eficiente** con el uso de la memoria. No tiene ni *runtime* ni *colector de basura*. Esto le permite soportar servicios críticos e integrarse con otros lenguajes fácilmente. Cuando se dice que Rust es rápido, es que en ocasiones y en algunas tareas incluso supera a C y C++. Cierto es que en otras es algo mas lento, pero en promedio, se puede considerar del mismo orden, en cuanto a rapidez se refiere.
- **Fiabilidad**. Rust tiene un *basto* sistema de tipos y un modelo de gestión de la memoria `ownership` y `borriwing`, muy peculiar, pero que le permite garantizar la seguridad, tanto en la memoria como en hilos. Esto permite eliminar o al menos reducir en gran medida los errores en tiempo de compilación. Esto, implica que te tienes que ocupar de la gestión de memoria, pero utilizando esos conceptos. Al principio, es algo difícil de *entender*.
- **Productividad**. Una de las características que mas me han llamado la atención de este lenguaje es la documentación. No solo la documentación relativa al propio lenguaje y a su sintaxis, sino la información que arroja el compilador cuando se produce un error. Básicamente, no solo te dice donde se encuentra el error, sino que además, en muchas ocasiones, te dice hasta lo que tienes que hacer para corregirlo.

¿Donde puedes utilizar Rust?
Pues casi te diría que sucede lo mismo que con *Python*, pero a menor escala. Rust lo puedes utilizar prácticamente en cualquier ámbito. Y comentarte que a menor escala, porque actualmente tiene menos *expansión* de la que tiene Python, y que no existen tantos módulos y librerías. Pero, con el desarrollo que está teniendo, simplemente es cuestión de darle tiempo.

- **Para la línea de comandos**

Puedes encontrar decenas de aplicaciones implementadas con Rust para la terminal, para la línea de comandos o `CLI`. Durante estos últimos tiempos, he descubierto docenas de aplicaciones que o bien reemplazan a aplicaciones equivalentes, o bien, que te ofrecen nuevas características.
Aunque vayas a crear un sencillo script, Rust, es un lenguaje ideal, ya que sabes que tendrás una aplicación eficiente y rápida. Además, los usuarios de la aplicación no necesitarán ni `runtime` ni librerías instaladas, se compila a un único binario. Esto ayudará tremendamente a su distribución.
Pero es que la distribución es realmente sencilla, puedes utilizar `cargo.io` o si tienes tu repositorio hospedado en GitHub, puedes utilizar *Travis CI*.
En el caso de la línea de comandos, yo ya he hecho *mis pinitos*, alguna que otra aplicación sencilla. Lo cierto, es que existe una gran cantidad de librerías que te ayudarán a hacer aplicaciones para la terminal, relativamente potentes, y con esfuerzo, reducido.

- **Para Web**

Puedes desarrollar para la Web con WebAssembly, que permite ejecutar código prácticamente a la velocidad de código nativo. En este caso, Rust, te ofrece unas características interesantes como,
Un rendimiento predecible, porque, entre otras razones, carece de colector de basura.
El código que se produce es reducido, por la misma razón que lo indicado en el punto anterior, pero porque además elimina el *código muerto*.
Tiene un ecosistema de librerías que te ayudará a no empezar de cero.
En este caso, no te puedo aportar gran cosa, porque lo cierto es que hasta el momento todavía no he podido *hincarle el diente al WebAssembly*, pero no lo descarto en un futuro.

- **Servicios y redes**

Este es el próximo objetivo al que le tengo *echado el ojo*. Hasta el momento las APIs y otros servicios los he estado implementando en Python y PHP. Sin embargo, lo *poco* que he visto de Rust, me ha gustado **muchísimo**.
De hecho, tengo puesto *el punto de mira* en dos *frameworks*. Pero es sobre todo **Rocket** el que me está haciendo pensar en dar el salto. Espero poder *migrar* algunas de las APIs implementadas anteriormente en otros lenguajes a Rust, a lo largo de este próximo año.
De nuevo, las razones para recurrir a Rust para implementar este tipo de servicios, son,
Un bajo consumo de recursos, tanto en lo que se refiere a CPU como a memoria.
Se trata de una solución segura y fiable.
Rust te garantiza que no compartes el estado entre tareas, por lo que depende de ti la estrategia de *concurrencia* que quieras aplicar, porque sabes que *tienes el éxito garantizado*.

- **Dispositivos embebidos**

Como has podido ver hasta el momento, una de las características de Rust, es precisamente su **fiabilidad**. En este sentido, y especialmente, Rust, hace imposible que de forma accidental se pueda compartir estados entre hilos. De esta forma, puedes utilizar cualquier aproximación de *concurrencia* que quieras, porque Rust, te garantizará el correcto comportamiento.
Otra de las características que te ofrece Rust, y que es especialmente interesante en los sistemas embebidos es la *portabilidad*, en el sentido, de que solo tienes que escribir una librería o un *controlador* una vez, y la puedes utilizar en todo tipo de sistemas, desde microcontroladores hasta *SBC*.
Así por ejemplo, tienes un repositorio en GitHub, que te va a permitir desarrollar tu propio sistema operativo en Rust para la Raspberry.

Con las manos en la masa
Una vez ya has visto el potencial que **Rust** pone al alcance de tus manos, llega el momento de ponerse en marcha, y quitarte el gusanillo que tienes encima. Para esto, es necesario, por un lado un IDE para desarollar, y por otro lado instalar las herramientas necesarias.
Respecto al IDE, depende de tus costumbres y preferencias. Si no utilizas Vim/NeoVim, mi recomendación es que te decantes por IDEA o por Visual Studio Code. Cualquiera de estos IDE tiene complementos para trabajar con Rust y son perfectamente funcionales. A mi, en particula, me gusta mas IDEA que Visual Studio Code, pero como de costumbre, esto es simplemente cuestión de gustos.
Si eres de los que utilizan Vim o NeoVim, te recomendaría que utilizaras NeoVim, con algunos complementos esenciales para que te ayuden en la labor del desarrollo. Sobre todo, te serán de gran utilidad en los inicios.

Instalando Rust y Cargo
¿Que es Cargo?
Cargo es el **gestor de paquetes de Rust**. Es la herramienta que se encargará de descargar las dependencias que utilices en tu proyecto. Igualmente, es el que compilará tus paquetes, los preparará para que sean distribuibles y por último subirlos a Crates.io, que es donde se suben y registran todos los paquetes, para que de esta manera puedan ser fácilmente distribuibles y utilizables por todo el mundo.

Instalación
La forma mas sencilla, práctica y cómoda de instalar tanto Cargo como Rust es utilizando rustup. Para esto, en *Linux* y *MacOS*, tan solo tienes que ejecutar la siguiente instrucción en una terminal,

```
curl https://sh.rustup.rs -sSf | sh
```

En el caso de Windows, tienes que descargar un archivo que puedes encontrar en la página de Rust.

Primeros pasos con Cargo
Ahora que ya tienes instalado Cargo, ya estás en disposición de crear tu primer paquete. Y es que *Cargo*, es una **auténtica maravilla**. Cargo, prácticamente hace todo el trabajo, excepto la parte de *programar* que para eso, estás tu. Cargo, prepara toda la estructura del programa, tanto para el caso de que quieras desarrollar una aplicación, como para el caso de que quieras desarrollar una librería.
Vamos al lío, a por un *Hola mundo*. Para esto, el primer paso, es crear toda la estructura de archivos. Algo tan sencillo como ejecutar la siguiente instrucción en una terminal,

```
cargo new hola_mundo
```

Esto lo que genera es lo siguiente

```
.
├── Cargo.toml
└── src
    └── main.rs
```

El archivo Cargo.toml, contiene lo siguiente,

```
[package]
name = "hola_mundo"
version = "0.1.0"
authors = ["autor <autor@correo.es>"]
edition = "2018"

[dependencies]
```

Por supuesto, que esto lo puedes personalizar según tus necesidades para personalizarlo. Y en su caso, mas adelante, añadir todas las dependencias que necesites para que tu aplicación funcione correctamente.
Por otro lado, dentro de **src/main.rs**, encontrarás lo siguiente,

```
fn main(){
    println!("Hello, world");
}
```

Para personalizarlo, puedes cambiar el Hello, world por un *Hola Mundo!!*, o dejarlo así, esto ya depende de tí.
Ahora toca compilar. Para ello puedes o bien compilar primero, para lo que simplemente tienes que utilizar de nuevo a Cargo, con cargo build o directamente ejecutarlo. En este segundo caso ejecuta cargo run --.
Para el caso de la compilación, el archivo ejecutable lo tienes en target/debug/hola_mundo. De forma que si ejecutas target/debug/hola_mundo, tendrás el mismo resultado que el que has visto anteriormente.

Más información,

- Rust
- Wikipedia
- Rust for Linux
- The New Stack
- Stack Overflow Developer Survey
- Rocket
- Actix
- GitHub

Autor: Lorenzo

Administrador de Atareao. Donde ha dado rienda suelta a mis dos grandes aficiones, la difusión del Open Source y y el desarrollo de aplicaciones

Web: https://www.atareao.es/

¡Linux la última frontera de la libertad!

Se han vertido auténticos ríos de tinta estos últimos años hablando sobre cuáles son las ventajas de usar Linux frente a otras plataformas informáticas conocidas, hoy me centraré principalmente en la revolución social y cultural que supuso la creación y posterior desarrollo de Linux en todos estos años. ¡Habéis pensado en ello! Cerrarlo ojos y desconectar por unos instantes de **"la vorágine de la vida moderna"** y centraos en este pensamiento, ¿qué hubiese pasado en la sociedad actual tan capitalista y consumista, si esta última frontera de libertad que aporta Linux no hubiese existido y cuáles serian sus posibles consecuencias?

¡Devastador paisaje intuimos frente a nosotros!, ¿verdad? Un mundo sin la existencia de GNU/Linux y el Open Soucer, como a los más puristas les gusta llamarle, hubiese supuesto un abismo infranqueable al acceso a las tecnologías por la mayor parte de la población mundial. Como consecuencia habría originado una brecha cultural de resultados impredecibles en todos los lugares del globo terráqueo, independientemente de la nacionalidad, potenciada por la imposición del aspecto de la Obsolescencia Programada que las compañías privativas imponen a sus productos de consumo, viéndonos obligados de este modo la sustitución por falta de recursos de nuestros dispositivos en periodos cortos de 2 a 4 años.

Bonita palabreja esa la de las **"Obsolescencia Programada"** claro reflejo de la dictadura impositiva que las grandes compañías "específicamente dos" ¡que no voy a mencionar, pues no lo considero necesario, ya que a todos nos viene a la cabeza! Las cuales imponen a sus productos de manera inquisitorial para que el usuario medio se vea obligado a la sustitución de los mismos sin derecho a réplica. ¡Compras una vivienda que sabes que a los pocos años deberás desocupar tengas o no medios para hacerlo! No eres dueño del producto adquirido solamente un inquilino temporal y desechable del mismo.

En este panorama que a grandes pinceladas que he expuesto, y que se intuye devastador. Si reflexionamos un poco, percibiremos claramente a Linux y las aplicaciones del software libre como los paladines y defensores del necesario equilibrio entre lo justo, práctico e igualitario, enriquecedor de cultura y equivalente en oportunidades sociales.

Afortunadamente, Linux independientemente de todas sus otras, ventajas que son muchas a garantizado acceso a la cultura a generaciones durante todos estos años desde su creación allá por el año 1991 del siglo pasado. Supuso la reducción de la brecha tecnológica en nuestra sociedad y se ha convertido en claro garante y medio para poder acceder a los medios educativos y de formación existentes.

Disponemos pues de una grandísima herramienta, no solo de esparcimiento y ocio cuyas funciones cubren ampliamente todas nuestras necesidades en la intimidad de nuestros hogares, además, por otro lado, se convierte en un sistema muy necesario en el terreno laboral, científico y educacional, etc. **"El Paladín"** que nos garantiza el acceso a la formación y a las nuevas tecnologías de una manera democratizada, por encima de imposiciones inquisitoriales de las grandes compañías deshumanizadas.

En mis vivencias por las distintas plataformas de comunicación linuxeras especialmente la de Telegram, donde conviven un universo de comunidades Linux. Me he encontrado con grandes personas, activistas rebeldes de la lucha social, abanderados y hombres con conciencia de justicia social, que se dedican a recuperar ordenadores de todo tipo, de cualquier lado, muchas veces abandonados juntos a contenedores de basura. Me comentaban cómo sirviéndose de este botín, la riqueza que otros desdeñaron por desconocimiento mayoritariamente, a los cuales, consideraban inservibles y supuestamente desfasados, como los conocidos Atom con 1 GB RAM y unos escasos 270 de disco duro o menos, por poner un ejemplo. Procedían a "rescatar esos PC viejitos" para reutilizarlos y devolverles a la vida adaptandoles a las distribuciones GNU/Linux más ligeras, con sus aún más ligeros entornos de escritorios. Afortunadamente, existen muchas distribuciones apropiadas con este fin en este increíble universo lleno de posibilidades, aptas para cubrir todo tipo de necesidades.

Muchos de esos grandes hombres de corazón generoso e ideas claras, donan o prestan esos PC recuperados con Linux a algunas escuelas, centros educativos o particulares, niñ@s mayoritariamente, para que tengan acceso a las nuevas tecnologías y de este modo ir creando nuevas canteras entre sectores de la población, dando a conocer las grandezas de Linux, crean conciencia social y solidaria, sembrando al mismo tiempo el germen de una nueva manera de entender el mundo, más democrática y justa. Consiguiendo de este modo que sean estos nuevos jóvenes afortunados que comprueban con asombro y gratitud de como pueden disponer de un grandísimo soporte tecnológico con lo más básico, y se conviertan a su vez los impulsores de las bondades de Linux generando que sé reúnan, interactúen y colaboren entre ellos, ayudando a instalar esta grandiosa herramienta social igualitaria, que es Linux, entre sus conocidos y cercanos.

Sembrando el germen entre las nuevas generaciones de "La Resistencia Linuxera" como yo llamo a esta actitud grandiosa de rebeldía y contraposición al abuso capitalista del consumismo y la insidia de la "Obsolescencia Programada". Logrando que crezca su interés de manera igualitaria y libre por la formación en las nuevas tecnologías, para ir creciendo de este modo sin límites impuestos, asegurándonos que las nuevas generaciones futuras se aseguren un porvenir enriquecedor, que sean capaces de adquirir cada día capacidades más avanzadas, que disfruten de Linux y sus inmensas posibilidades y aprendan como sirviéndose de lo mínimo y pueden adquirir conocimientos como para poder montar un "Nas casero", servidor de impresión casero, Batocera para juegos retro, etc…. He, ir creciendo y desarrollando sus capacidades en un fructífero terreno sin límite y en completa libertad sin barreras de ningún tipo.

A nivel personal como usuario medio con la experiencia adquirida en todos estos años de mi amor por Linux. Puedo decir que su uso me ha demostrado que no me equivoque cuando hace décadas ya por causa de fuerza mayor mi W Xp quedo desfasado y no tenía recursos para sustituir un PC de pocos años. Veréis con mucha inexperiencia y resquemor queme mi primer DVD y me dispuse a instalar mi primera distribución Linux en un dual boot ¡Si en un dual boot!, por miedo absurdo a no poder hacer frente aquel desconocido mi primer Ubuntu, mi nuevo sistema operativo Linux.
He de confesar que me paso de todo por el camino, pero jamás me desilusione fue un proceso más o menos largo de **"desintoxicación del ventanas"** y su modo de trabajar. En Linux como en la vida hay que enfocar todo de otro modo distinto, diferente, de a poco y pausadamente fui adquiriendo nuevas habilidades en su manejo.

He de destacar que fue un viaje apasionante, con pocas decepciones y muchas satisfacciones, cuando comprendí que podía hacer lo mismo, más y mejor, en este nuevo medio, que en "el ventanas", eso si de un modo diferente, más eficaz y seguro. Una vez que gané confianza, libertad de uso y una gran adquisición de conocimiento que, por otro lado, he de señalar se mantienen en el tiempo. Eso precisamente es una de las cosas que más me gusta, la de mantener intacta la capacidad y ganas por aprender más y más, día a día, de a poco, sin prisas como se hacen todas las cosas buenas de la vida y sin imposiciones de ningún tipo, sirviéndome de un sistema operativo Unix que me es grato, agradable y poco o nada impositivo.

Ir desarrollando mis capacidades, ir creciendo y adaptarme a las mejoras tecnológicas las cuales se desarrollan y cambian de un modo voraz, tan veloz como la vida actual, pero sirviéndome de mi viejo compañeros de aventuras mi PC de más de 10 años que se mantiene hay impasible gracias a las bondades de GNU/Linux y el Open Soucer y lejos de las garras del desfase tecnológico impuesto por otros.

Bien hasta aquí este pequeño artículo que espero les hayan resultado agradable y les hiciese reflexionar un poco más sobre los aspectos geniales de Linux, algunos de los cuales generalmente solemos pasar por alto.

Nos seguiremos hablando en el Ciberespacio independientemente del costo del material utilizado como soporte. ¡Ahí nos vemos! **¡Animaros a usar Linux!**, y bien venidos una vez más **"A la resistencia Linuxera"**.

Autor: Pedro Crespo

Administrador de Latin Linux

Web: https://www.latinlinux.com/

Puntos ciegos: El mito de no saber.

Te acostumbraron durante años a que tenías que “saber“, y ese saber estaba ligado, posiblemente, a tu eterna capacitación, lecturas, y demás yerbas. Si no ibas por el camino convencional ibas a ser un burro. ¡Frase desafortunada si la hay! Los burros son animales extremadamente pacientes, colaboradores, miden sus tiempos, observan, caminan a su paso, son dueños de si mismos y de todas las cargas que le imponen. Deberíamos mirar más a nuestros amigos los burros que tienen la mala fama.

Hoy tu saber se arma desde lo que fuiste incorporando en tu vida no solo desde los libros y cientos de pruebas dadas aprobadas o reprobadas, sino de la experiencia que te dejó cada una de esas situaciones. Cerrá los ojos. Ponete a pensar cuanto sabés de toda la geometría marcada con el compás, cuantas veces usaste en los últimos dos años un transportador, o más aún si recordás o utilizás algún método taquigráfico de los años de María Castaña. Vale cualquier respuesta porque lo cierto es que uno recuerda lo que le resulta importante y tanto la ciencia como la realidad actual, ya no trabaja sobre conceptos duros, sino que cuando te eligen quieren notar y ver tu potencial, sí, que tan humano sos y como impactás en los demás.

Hoy derribar el no saber es **“madre llave”** para tus avances. Sacarte el molde te tener que saberlo todo para poder ser más. Ya no está a la moda . Hoy la posibilidad desde tu mayor ignorancia es la que te va a permitir utilizar tu creatividad y tu posibilidad de invención. ¿ Acaso no te enloquece cuando estás intentando ver que sucede en el momento en que ponés a prueba un código nuevo? Esa adrenalina de no saber cual será el resultado de tu acción. Así estás creando. Inventar, crear, darte ese espacio jamás conocido ni por vos ni por otros para descubrir cosas nuevas. Entonces a partir de hoy, tu NO SE es el inicio para el cursor cuando titila y no tenés ni la menor idea como seguir.

El no saber invita a que otros puedan colaborar en tu proceso, aportando ideas, opiniones, alternativas, herramientas que suman a tu trabajo profesional y personal. El no saber siempre estuvo asociado a ser menos, sin embargo hoy, es un enorme paso de valentía y coraje entrar en las profundidades de lo desconocido porque vos sabes lo que sabés, podrás saber lo que NO sabés, pero el desafío es que vayas por todo eso que está ahí afuera que no sabés ni siquiera que existe y que no sabés.

Suena trabalenguoso ? Jajajaja , interesante verdad? Atravesar tus propios limites y animarte a decirle al mundo que tus todos no saberes son los que abrazan tu futuro. Un futuro incierto, desconocido, despoblado de curiosos e interesados en meterse en los mil laberintos jamás explorados.

Cuando algo te interpele en tu vida y “no sepas” agarrate como una garrapata y empezá a investigar con preguntas y cuestionamientos que rompan cualquier esquema . Estarás evolucionando y creciendo , aprendiendo . Ahhh y si en el medio de toda esta nueva invención hay errores, mejor. Esas son las estaciones que te van a dejar re calcular para ver cual será el próximo paso que elijas dar.

Pero ... ¿ Como sabés que llegaré al resultado?
No sé.

Autoras: Flavia Rossi & Andrea Garcia

Founders Consultora Puntos Ciegos & Business for Sucess