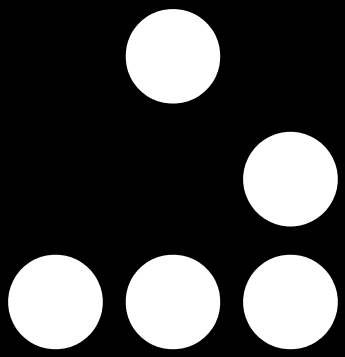


CHANGE

SUPLEMENTO ESPECIAL

HACKERS & DEVELOPERS MAGAZINE

RESPONSABLE EDITORIAL
Eugenia Bahit

HACKER TEAM:
Eugenia Bahit
María José Montes Díaz
Milagros Infante Montero

HACKERS & DEVELOPERS MAGAZINE "CHANGE"
FECHA DE PUBLICACIÓN: 31 DE AGOSTO DE 2013

CREADO CON **LIBREOFFICE WRITER**

CHANGE

SUPLEMENTO ESPECIAL
HACKERS & DEVELOPERS MAGAZINE

NUESTRO COMPROMISO CON QUIENES APUESTAN AL VERDADERO CAMBIO

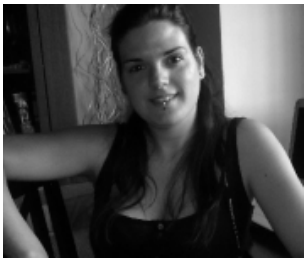
PAPERS DE • EUGENIA BAHIT • MARÍA JOSÉ MONTES DÍAZ • MILAGROS INFANTE MONTERO

PAPERS

Instalando Slackware

Slackware es una distribución basada en la filosofía KISS y además, es la distribución GNU/Linux más antigua en vigencia. Veamos cómo es su instalación.

Escrito por: **María José Montes Díaz** (Docente & Programadora)



Estudiante de Grado Ingeniería en Tecnología de la información. Técnico en informática de gestión. Monitora FPO. Docente de programación Python y Scratch para niños de 6-12 años. Activista del software libre y cultura libre.

Webs:
Blog: http://archninfa.blogspot.com.es/

Redes sociales:
Twitter: **@MMontesDiaz**

El creador de esta distribución fue Patrick Volkerding[1]. Su intención es producir la distribución GNU/Linux con el objetivo principal de la estabilidad y simplicidad de uso. Actualmente, es la distribución más antigua en vigencia, nació en abril de 1993, y es la más "UNIX-like". El equipo de desarrollo de Slackware considera la simplicidad y la estabilidad de suma importancia, y como resultado, esta distribución se ha convertido en una de las más populares disponibles, estable y amigable.

Esta distribución cuenta con el núcleo Linux 3.2.29 y Glibc 2.15. Podemos descargar la imagen ISO a través de BitTorrent[2] o mirrors[3]. Archlinux está inspirada, entre otras, en esta distribución.

Comenzamos la instalación

Nada más arrancar, nos aparecerá una pantalla en la cual poder ingresar algún parámetro extra necesario para arrancar la instalación. No es necesario ingresar nada y puede continuarse con la instalación pulsando Enter.

Debemos seleccionar el núcleo con el que arrancaremos. La opción seleccionada por defecto debería ir bien.
Una vez arrancado el sistema, nos preguntará qué mapa de teclado queremos. Por defecto está en inglés, así que, pulsamos 1 y buscamos **"qwerty/es.map"** (o el que aplique en cada caso).

Una vez seleccionado el mapa de caracteres, nos aparecerá otra ventana en la que comprobar el teclado. Una vez realizada la prueba, bastará introducir 1 en una nueva línea y pulsar Enter.

Una vez hecho esto, ya podremos iniciar sesión. El usuario es "root" y no tiene contraseña. Lo primero que debemos hacer es particionar el disco. Para ello disponemos de `cfdisk` y `fdisk` para sistemas con MBR y, `cgdisk` y `gdisk` para sistemas GPT. Para una instalación completa, con una partición raíz de 15 GB sería suficiente. Para la partición swap, si nuestro equipo tiene menos de 1 GB, podríamos darle ese tamaño; si es de 2 GB o más, incluso podríamos omitirla. De todas formas, siendo ésta el 70% de la física,

1 http://wikipedia.org/wiki/Patrick_Volkerding
2 http://slackware.com/getslack/torrents.php
3 http://mirrors.slackware.com/

sería suficiente para poder realizar la suspensión a disco del equipo.

SETUP

Tras crear las particiones que alojarán Slackware, ejecutamos setup en la línea de comandos para empezar a configurar el sistema. Es un sistema de menús que nos permitirá la instalación de paquetes. Dado que ya hemos seleccionado el teclado, pasaremos a la siguiente opción.

ADDSWAP

Es donde se establecen las particiones Swap, para ello escogemos esta opción y pulsamos Enter. Aparecerá la partición o particiones que tiene el sistema de archivos Swap, elegimos y pulsamos ok. Antes de formatear, el instalador preguntará si queremos que busque las partes dañadas del disco, si no tenemos sospechas de tener daños, elegimos no. Cuando termine de formatearse el disco, pulsamos la tecla Enter.

TARGET

En está sección es donde especificaremos el resto de particiones y estableceremos los puntos de montaje del sistema de archivos. Tendremos la posibilidad de formatear cada una de las particiones. Para la partición que alojará el sistema, es decir, el punto de montaje /, formatearemos la partición. Disponemos de varios sistemas de archivos para formatear. Una buena opción es EXT4.

SOURCE

En esta sección elegiremos el medio fuente desde el que instalar. En este caso, optaremos por la primera opción. Luego, bastará pulsar Enter y OK en la opción auto.

SELECT

Ahora toca seleccionar los paquetes que queremos instalar. La serie A es necesaria para obtener un sistema funcional básico. Las demás opciones son opcionales. Para elegir, basta pulsar Espacio sobre la opción. Una vez terminada la selección, pulsamos OK.

INSTALL

En este apartado podemos escoger entre siete métodos de instalación diferentes. La opción más rápida y recomendada es full. Por supuesto, es la opción que más espacio consume.
Una vez acabada la instalación, tendremos la opción de crear un USB autorrancable.

CONFIGURE

LILO (Linux Loader).- Es un gestor de arranque que permite elegir entre sistemas operativos Linux y otras plataformas. Para hacer una instalación automática, escogemos Simple. Lo siguiente es elegir la resolución de la pantalla, elegimos una opción (que puede ser `Standard`) y pulsamos Enter.

Si no queremos agregar ningún parámetro al kernel, dejamos el siguiente cuadro vacío y pulsamos Enter.

Cuando nos pregunte si queremos usar la consola UTF-8 pulsamos "No".

Lo siguiente es seleccionar dónde queremos instalar Lilo. Elegimos MBR, pulsamos OK.

Toca elegir los drivers para el ratón, seleccionamos `imps2`. A continuación se nos preguntará si queremos que el ratón se pueda usar

mientras reinicia, elegimos `yes`. Se nos pregunta si queremos configurar la red, elegimos `yes`.

Ahora ecribimos el HOSTNAME y pulsamos Enter (en mi caso, por ejemplo, `ninfa-Slack`).
Si queremos unir nuestro equipo a un dominio, escribimos el nombre, sino escribimos (.) y pulsamos Enter

Una vez hecho esto, elegimos cómo queremos nuestra IP, dinámica o estática. En este caso elegimos DHCP. Luego nos pedirá el hostname DHCP, si no tenemos ninguno, pulsamos Enter:

Aparece un menú con los servicios que pueden ser iniciados durante el arranque. De entrada, yo recomendaría utilizar estos: rc.cups, rc.fuse, rc.inetd, rc.messagebus, rc.samba, rc.ntpd, rc.syslog, rc.sshd, al final pulsamos Enter.

Nos pregunta si queremos personalizar fuentes adicionales, en mi caso No y pulsamos Enter. Luego, para seleccionar la zona horaria, escogemos la opción No y especificamos la ciudad.

¿Qué entorno gráfico queremos? Esa es la selección que hay que hacer en este punto. Una buena opción es KDE, así que correspondería elegir xinitrc.kde

Ya estamos casi terminando, pero nos falta un punto muy importante: la contraseña del usuario root, por tanto, cuando nos pregunte si queremos añadir una contraseña al root, escogeremos yes. Ya ha terminado la instalación, salimos de la utilidad y reiniciamos el equipo con reboot.

Habilitando la interfaz gráfica.

Una vez reiniciado el sistema, activamos la interfaz gráfica. Para ello editaremos el archivo /etc/inittab y, en la opción que aparece id:3:initdefault, la cambiamos por id:4:initdefault.

```
# nano /etc/inittab
```

Guardamos con Ctrl+O , Enter, y salimos con Ctrl + X, Enter. Reiniciamos con: `reboot`

Referencias:

http://www.slackware.com/

http://docs.slackware.com/

Las mejores prácticas de UX & UI para aplicaciones Android™

Al momento de desarrollar una aplicación, debemos estar al tanto de que ésta cumpla las expectativas de los usuarios de Android™ (tanto para la interfaz como para la navegación en un dispositivo) ya que, aunque no lo parezca, la aplicación se verá afectada enormemente por ellas.

Escrito por: **Milagros Alessandra Infante Montero** (Est. Ing. Informática)

Estudiante de Ingeniería Informática. Miembro de la comunidad de software libre **Lumenhack**. Miembro del equipo de traducción al español de **GNOME**. Apasionada por el desarrollo de software, tecnología y gadgets. Defensora de tecnologías basadas en software libre y de código abierto.

Webs:
Blog: www.milale.net

Redes sociales:
Twitter / Identi.ca: **@milale**

Cada vez que desarrollamos aplicaciones, podemos no darnos cuenta de un detalle que puede marcar la diferencia, a si durante el proceso de desarrollo hemos utilizado buenas prácticas tanto para obtener la mejor experiencia de usuario como para la mejor interfaz de usuario[4].

Ya en la edición 8 de Hackers & Developers Magazine[5], Fabio Durán nos hablaba de cómo la usabilidad se convierte en el primer paso al éxito y dándonos algunas recomendaciones para mejorar nuestro software. A continuación veremos algunas recomendaciones que nos darán la posibilidad de asegurar que la aplicación que desarrollemos satisfaga las expectativas de un usuario de Android™.

Diseño de la navegación

Cuando nos encontramos desarrollando y diseñando nuestras aplicaciones resultará fundamental determinar qué es lo que los usuarios verán y harán con la app que les entreguemos como resultado final, luego deberemos prestar cuidado en el diseño de las interacciones para lograr la mejor experiencia de navegación del usuario.

"El diseño es el pensamiento hecho visual" - Saul Bass

- **Relaciones entre pantallas:**
  El modelo de información de las aplicaciones se puede expresar mediante diagramas, sobretodo para tener una idea clara de las cosas con las que el usuario interactuará, se puede usar para ello diagramas Entidad-Relación. Después, debemos hacer

4 http://web.braintive.com/10-reglas-heuristicas-de-usabilidad-de-jakob-nielsen/
5 http://hdmagazine.org - "Horse", edición 8.

una lista en la que se indique como los usuarios van a interactuar con todo el contenido de la aplicación: la pantalla de inicio, la lista de categorias y la lista de las ímagenes como de cada uno de los elementos. Con todo ello, debemos ver a las relaciones entre pantallas pero gráficamente, usando algún mapa con flechas que permita ver que hará la aplicación y a donde se dirige de pantalla en pantalla.

Si queremos permitir que los usuarios suban nuevas historias o fotos, podemor añadir pantallas adicionales al diagrama.

A la interfaz de usuario se pueden añadir botones que lleven a otras secciones, listas verticales que representes colecciones e información detallada (poder maximizar fotografías, detalle de las vistas, etc).

- **Pantallas táctiles:**
  Las aplicaciones deben adaptarse a diferentes tamaños de pantalla debido a los diferentes dispositivos que tienen este sistema operativo, por ejemplo dispositivos desde 3" hasta las tablets que tienen 10" e incluso a televisores de 42".

Para cumplir con los estándares al diseñar para televisores es bueno revisar el Google TV:
https://developers.google.com/tv/

Las pantallas que poseen 3 o 4 pulgadas generalmente solo tienen espacio para poder presentar el contenido de manera vertical, siempre cumpliendo la jerarquía de la información: que va de categorías a lista de objetos a detalles de objeto. Al contrario de las pantallas de tablets y televisores que al contar con mucho más espacio muestran sus paneles ordenados de izquierda a derecha, se debe mostrar muchos paneles ya que se debe evitar los espacios en blanco, existen muchas estrategias usadas para el diseño en de paneles múltiples[6].

- **Navegación descendente, lateral, temporal y ancestral:**
  La navegación jerárquica anterior es una de las maneras, otra es la descendente y una navegación lateral permitiendo a los usuarios ir a través de las pantallas, mediante tablas, listas, pestañas y otros patrones.
  El patrón de navegación lateral es el paso de página a página horizontalmente, son como pestañas que agrupan cada una mucho contenido generalmente listas, pero hay que tener cuidado si dentro de estas vistas se encuentra algún elemento que requiera navegación lateral como los mapas, ya que de todas maneras la usabilidad se verá afectada.

Para saber más sobre deslizamiento de vistas es importante revisar este link:
http://developer.Android™.com/design/patterns/swipe-views.html

La navegación temporal se realiza mediante el botón "Atrás", esto se encuentra dentro de las convenciones de Android™, ya que todo usuario siempre buscará ello para volver a la vista anterior donde se encontraba.
La navegación ancestral antes del Android™ 3.0 era dado por "Inicio", llevando de esta manera al usuario a la página principal de la jerarquía. Pero Android™ 3.0 introdujo a "Arriba", similar a un "Atrás" que lleva al usuario a la vista anterior donde se encontraba.

- **Wireframing:**
  Es el paso en el que debemos diseñar las pantallas, pensar y ser creativos al momento de hacer que todos los elementos de la interfaz de usuario permitan navegar por toda la aplicación de la mejor manera.

La mejor manera de empezar, es dibujar a mano en una hoja de papel y de esta manera uno se podrá dar cuenta de los problemas para ver que patrones colocar o quitar según la necesidad

6 http://developer.Android™.com/design/patterns/multi-pane-layouts.html

Después de ello, se debe seleccionar una herramienta de su preferencia para hacer wireframes y de esta manera servirá como punto de partida para el diseño visual de la aplicación. Finalmente, después de haber diseñado las pantallas para toda la interacción dentro de la aplicación, ya se puede prestar atención a mejorar los pequeños detalles individualmente por cada pantalla.

Notificaciones al usuario

Algo que se encontrará fuera de la interfaz de usuario permanente son las notificaciones que se realizan al usuario cuando un evento nuevo a ocurrido y requiera atención, estas podrían llevar al usuario a otro servicio o permitirle elegir si desean o no responder.

Para implementarlas, Android™ cuenta con una clase en la librería de soporte, *NotificationCompat.Builder* para así poder brindar soporte a las notificaciones. Con este objeto se puede especificar el contenido de la interfaz de usuario y las acciones, el objeto *Builder* permitirá incluir un ícono pequeño, un título y texto detallado.

```
NotificationCompat.Builder mBuilder =
    new NotificationCompat.Builder(this)
    .setSmallIcon(R.drawable.notification_icon)
    .setContentFile("Notificación importante")
    .setContentText("Esta notificación es de prueba");
```

Pueden añadirse acciones a las notificaciones como también indicar que comportamiento queremos que tenga.

Funcionalidad de búsqueda

Las características de búsqueda brindan a las aplicaciones una manera fácil de brindar una experiencia de búsqueda consistente al usuario. Para esto Android™ nos permite usar el widget *SearchView*[7] como un elemento en la barra que permitirá la búsqueda.

> *"Uno de los errores más grandes que puede tener una aplicación es que su buscador sea malo"*

Accesibilidad

Los usuarios de Android™ tienen distintas habilidades que requieren que interactuen con sus dispositivos de diferentes maneras, incluyendo a personas que puedan tener limitaciones visuales, físicas o relacionadas a la edad que quizás no puedan usar una pantalla táctil del todo o incluso que no puedan escuchar alertas o información audible. Android™ provee características de accesibilidad y servicios para ayudar a los usuarios a navegar por sus dispositivos de manera más sencilla, por ejemplo navegación de texto a voz, navegación gestual, navegación direccional con el pad, entre muchos más; definitivamente es una función que los desarrolladores deben aprovechar al máximo.

> *"Cuando algo es accesible, es mucho más cómodo para todos"*

Android™ nos da la oportunidad de entrar a su maravilloso mundo y disfrutar de un happy hacking y que más si a ello le añadimos poder lograrlo usando las mejores prácticas para que nuestras aplicaciones cuenten con una genial interfaz y la mejor experiencia de usuario.

7 http://developer.Android™.com/reference/Android™/widget/SearchView.html

Snippets en Bash para agilización de scripts de Shell

Los snippets son pequeñas fracciones de código reutilizables, que con un simple copiar y pegar solucionan un problema. Esta es una pequeña recopilación de snippets en Bash, que a diario debo utilizar para agilizar mis scripts de Shell. Tomé aquellos que utilizo con mayor frecuencia.

Escrito por: **Eugenia Bahit** (GLAMP Hacker & eXtreme Programmer)

Eugenia es **Arquitecta de Software**, docente e instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y **eXtreme Programming**. Miembro de la **Free Software Foundation**, **The Linux Foundation** y **Debian Hackers**. Creadora de python-printr, Europio Engine y colaboradora de Vim. Fundadora y Responsable Editorial de **Hackers & Developers Magazine**.

Webs:
Cursos de programación: www.cursosdeprogramacionadistancia.com
Web personal: www.eugeniabahit.com

Redes sociales:
Twitter / Identi.ca: **@eugeniabahit**

Bash es uno de los lenguajes para Shell Scripting más comunes y por lo tanto, más utilizados no solo por programadores, sino por sobretodo, por administradores de sistemas. Frecuentemente, los scripts de Shell suelen utilizarse para agilizar y por qué no, automatizar, tareas de administración del Sistema Operativo. Muchas veces, estos scripts son utilizados por el mismo administrador del sistema que los ha programado y otras tantas, los mismos están destinados a los usuarios del sistema.

Esta breve recopilación de Snippets, está destinada a cualquier tipo de scripts, ya sea para uso propio o de terceros. Vale aclarar que cada Snippet se encuentra liberado bajo licencia GNU GPL 3.0.

> *Los Snippets son pequeñas fracciones de código fuente reutilizable, que con solo copiar y pegar o a lo sumo, copiar, pegar y "customizar", pueden ser utilizados en cualquier programa.*

Snippet #1: Conocer cantidad de usuarios en el sistema

Objetivo posible: ejecución de una tarea programada que necesita correr sólo si no hay usuarios operando sobre el sistema.

```
script_a_ejecutar="./miscript.sh"

cantidad_usuarios=`who | wc -l`

if [ $cantidad_usuarios -eq 0 ]; then
   exec "$script_a_ejecutar"
fi
```

Snippet #2: Mantener un script en ejecución continua

Objetivo posible: demonio o proceso en segundo plano.

```
script_a_ejecutar="./miscript.sh"

while true; do
    exec "$script_a_ejecutar"
```

Snippet #3: Impedir ejecución de script si el usuario no es root

Objetivo posible: ejecución de cualquier tipo de tarea que requiera permisos de súper usuario.

```
if [ "$USER" != "root" ]; then
    echo "Permiso denegado."
    exit 0
fi

# ...
```

Snippet #4: Guardar y recuperar preferencias del usuario

Objetivo posible: archivo de configuración de aplicación propia.

```
app_name="myapp"
directorio_app="$HOME/.$app_name"
archivo_configuracion="$directorio_app/.config"


# Snippet 4.1:
# Crea un directorio de aplicación con el archivo de configuración si no existe
if [ ! -d "$myapp" ]; then
    mkdir $directorio_app
    touch $archivo_configuracion
fi


# Snippet 4.2:
# Solicita preferencias al usuario (a modo de ejemplo)
function pedir_preferencias() {
    echo -n "¿Color favorito? (r: rojo / v: verde) "
    read color_favorito

    if [ "$color_favorito" != "r" ] && [ "$color_favorito" != "v" ]; then
          echo "Opción inválida"
```

```
            pedir_preferencias
    fi
}


# Snippet 4.3:
# Guarda las preferencias del usuario
function guardar_preferencias() {
    pedir_preferencias
    echo "COLOR_FAVORITO = $color_favorito" > $archivo_configuracion
    echo "OTRA_VARIABLE = otro valor" >> $archivo_configuracion
}


# Snippet 4.4:
# Recupera las preferencias del usuario
function recuperar_preferencias() {
    cat $archivo_configuracion | while read varname asignation varvalue; do
           case "$varname" in
                  COLOR_FAVORITO) echo "El color favorito es $varvalue";;
                  OTRA_VARIABLE) echo "El valor de otra variable es $varvalue";;
           esac
    done
}
```

Snipet #5: Parsear argumentos

Objetivo posible: script con un menú de opciones en el cual, según la opción elegida por el usuario, sea la acción que el script realice.

```
echo -n "Elegir opción (a/b/c): "
read OPCION

case $OPCION in
    a) echo "se eligió la opción A";;
    b) echo "se eligió la opción B";;
    c) echo "se eligió la opción C";;
    *) echo "OPCIÓN INCORRECTA";;
esac
```

Snippet #6: Capturar el valor de retorno no numérico de una función

Objetivo posible: se necesita obtener un determinado valor no numérico el cual es establecido de forma dinámica por el llamado a una función y se lo necesita almacenar con un nombre de variable distinto al del definido por la función implementada en la llamada de retorno.

```
function foo() {
    # ...
    variable="Valor de retorno para el argumento $1"
    # en bash, el valor de retorno se imprime
    echo $variable
}


function bar() {
```

```
    argumento="Lorem Ipsum"
    nueva_variable=$(foo $argumento)
}
```

Snippet #7: Convertir una cadena de texto a mayúsculas

(incluyendo caracteres acentuados, diéresis y eñes)

```
#     Uso: strin2upper mi cadena de texto
# Retorna: MI CADENA DE TEXTO
string2upper() {
   echo $* | tr 'a-z|á|é|í|ó|ú|ñ|ü' 'A-Z|Á|É|Í|Ó|Ú|Ñ|Ü'
}
```

Snippet #8: Convertir una cadena de texto a minúsculas

(la inversa del anterior)

```
#     Uso: strin2lower MI CADENA DE TEXTO
# Retorna: mi cadena de texto
string2lower() {
   echo $* | tr 'A-Z|Á|É|Í|Ó|Ú|Ñ|Ü' 'a-z|á|é|í|ó|ú|ñ|ü'
}
```

Snippet #9: Verificar argumentos pasados al script

Objetivo posible: un script que pueda fallar si no le son pasados una determinada cantidad de argumentos.

```
if [ $# -ne 1 ] ; then
    echo "Uso ./script.sh <argumento1> <argumento2>..."
    exit 0
fi
```

Snippet #10: Leer un archivo y asignar números de línea

Objetivo posible: un script cuyo objetivo sea trabajar con código fuente o manejar diferencias entre diversos archivos.

```
function read_file() {
    archivo=$1
    contenido_con_numeros_de_linea=`awk '{print NR, " ", $0}' $archivo`
}
```