

Una aplicación Perl a medida para redes WiFi

GESTOR GÜIFI

El soporte de redes inalámbricas está incluido en la mayoría de las distribuciones de Linux desde hace tiempo. Apenas instalada cualquier distribución en un nuevo ordenador, nos muestra el adaptador wifi activo y listo para ser configurado. Es exactamente aquí donde comienza la polémica.

POR JULIAN COCCIA



Mi problema (y el de quien viaje con su portátil con frecuencia) está en la cantidad de pasos que implica configurar una nueva red WI-FI. Hace unos días vi en un aeropuerto cómo un hombre luchaba con su XP tratando de conectarse al punto de acceso. Después de perder unos cuantos minutos presionando el botón derecho y abriendo ventanas, logró su cometido. Fue mi primer encuentro (por suerte desde lejos) con el gestor WI-FI de XP, el cual me pareció patético. “¿Será que esta gente nunca se mueve en la misma red?”, me pregunta-



ba a mí mismo.

“¿Y qué es lo que busco?”, os preguntaré. Pues lo que busco es simplemente saber qué redes me rodean con un simple clic. Y quiero además poder conectarme a cualquiera de ellas con un segundo clic. ¿Seré tan complicado? Pues estuve buscando y probando muchas aplicaciones de gestión WI-FI para Linux, pero ninguna supo satisfacer mi necesidad. Las herramientas de consola proporcionadas por wireless-tools son muy poderosas y es muy fácil obtener, por ejemplo, una lista de puntos de acceso disponibles con iwlist (ver Figura 1). Pero esta herramienta está lejos del alcance del usuario principiante.

Listado 1: Ventana Principal

```
01 my $win = Gtk2::Window->new;
02 $win->set_title ('Gestor Güifi');
03 $win->set_border_width (6);
04 $win->set_default_size (600, 500);
05 $win->signal_connect (delete_event => sub { Gtk2->main_quit; });
06 $win->show_all;
07 Gtk2->main;
```

```
root@LinEspa: /root
root@LinEspa:~# iwlist eth0 scan
eth0 Scan completed :
Cell 01 - Address: 80:00:00:00:00:00 (E)
ESSID:"MyAccessPoint"
Protocol:IEEE 802.11b
Mode:Master
Channel:15
Encryption key:off
Bit Rate:11 Mb/s
Extra: Rates (Hb/s): 1 2 5.5 11
Quality:55/100 Signal level:-80 dBm
Extra: Last beacon: 37ms ago
Cell 02 - Address: 80:00:00:00:00:00 (E)
ESSID:"MyAccessPoint"
Protocol:IEEE 802.11b
Mode:Master
Channel:11
Encryption key:off
Bit Rate:11 Mb/s
Extra: Rates (Hb/s): 1 2 5.5 11
Quality:31/100 Signal level:-91 dBm
Extra: Last beacon: 17ms ago
root@LinEspa:~#
```

Figura 1: El comando iwlist es muy poderoso, pero requiere conocimientos avanzados.

Si yo fuese el hombre del “XP” en el aeropuerto, me resignaría (la resignación es práctica común para quien usa ese sistema operativo) a seguir perdiendo tiempo cada vez que preciso conectarme. Pero de ninguna manera me podría

resignar a perder el tiempo contando con la libertad de opciones que Linux me ofrece. Wireless-tools hace todo lo que necesito y es sólo cuestión de automatizarlo mediante una interfaz gráfica. Existen muchas herramientas para tal fin, así que se me ocurrió probar libgtk2-perl ya que el aspecto de GTK2 me resulta atractivo.

GTK2 (Gimp Toolkit Library versión 2.x) es una librería multiplataforma para

Listado 2: Lista de Puntos de Acceso

```
01 my $box1 = Gtk2::VBox->new
(0,3);
02 $win->add ($box1);
03
04 my $scwin =
Gtk2::ScrolledWindow->new;
05 $box1->pack_start ($scwin, 1,
1, 0);
06 $scwin->set_policy (qw/automatic
automatic/);
07
08 my $slist =
Gtk2::SimpleList->new (
09 'Nro' => 'text',
10 'Nombre' => 'text',
11 'Cifrado' => 'text',
12 'Canal' => 'text',
13 'Señal' => 'text',
14 'Calidad' => 'text',
15 'Protocolo'=> 'text',
16 'Dirección'=> 'text',
17 );
18 $scwin->add ($slist);
19
20 my $box2 = Gtk2::HBox->new (0,
6);
21 $box1->pack_start($box2, 0, 1,
0);
```

Listado 3: Creación de Botones

```
01 my $btn;
02 foreach
(['Buscar'],['Conectar'],['Des
conectar'])
03 {
04 $btn = Gtk2::Button->new
($->[0]);
05 $btn->signal_connect (clicked
=> \&boton_presionado,
$->[0]);
06 $box2->pack_start($btn, 0, 1,
0);
07 }
08
09 $btn =
Gtk2::Button->new_from_stock
('gtk-quit');
10 $btn->signal_connect (clicked
=> sub { Gtk2->main_quit; });
11 $box2->pack_end($btn, 0, 1,
0);
```

Listado 4: Función "boton_presionado"

```

01 sub boton_presionado          15 `ifconfig $iface down`;
02 {                             16 }
03 my ($button, $op) = @_;       17 elsif( $op eq 'Conectar' )
04 my $indice;                   18 {
05 my $iface;                     19 $iface = guifi_interface();
06 if( $op eq 'Buscar' )         20 $indice=join(", ",
07 {                               $slist->get_selected_indices);
08 guifi_scan;                   21 my
09 }                               $ap=$slist->(data)[$indice][1];
10 elsif( $op eq 'Desconectar' ) 22 `iwconfig $iface essid
11 {                               $ap`;
12 $iface = guifi_interface();    23 }
13 `iwconfig $iface essid        24 1;
    off/any`;                     25 }
14 `killall -9 dhclient`;

```

la creación de interfaces gráficas. Y libgtk2-perl es la interfaz para Perl de dicha librería. Como lo que quiero es eje-

cutar y leer la salida de comandos de consola, me pareció que ésta sería una solución rápida a mi problema.

Listado 5: Función "guifi_scan"

```

01 sub guifi_scan {              25 if ($linea =~ /^.*Cell\
02 my ($cell, $mac, $essid,      (.*) \- \ Address: (.*)/) {
    $linea, $key, $channel, $qua- 26 $cell = $1;
    lity, $signal, $protocol);    27 $mac = $2;
03 my $iface = guifi_interfa-   28 }
    ce();                          29 if ($linea =~
04                                /^.*ESSID:"(.*)"/) {
05 @{$slist->(data)} = ();         30 $essid = $1;
06 $command="iwlist $iface      31 }
    scan";                          32 if ($linea =~ /^.*on\
07                                key\:(.*)/) {
08 $cell="";                       33 $key = $1;
09 $essid="";                       34 }
10 foreach (`$command`) {        35 if ($linea =~
11 $linea=$_;                       /^.*Protocol:(.*)/) {
12 if ($linea =~ /Cell\ / &&      36 $protocol = $1;
    $essid) {                          37 }
13 push(@{$slist->(data)},        38 if ($linea =~
    [$cell, $essid, $key, $chan-     /^.*Quality:(.*).*Signal\
    nel, $signal, $quality, $pro-    level\:(.*)\ N/) {
    tocol, $mac]);                   39 $quality = $1;
14 if (!$cfg->{"$mac"}) {         40 $signal = $2;
    $cfg->set("$mac.essid", $essid)   41 }
    ; }                               42 if ($linea =~
15 $cell="";                       /^.*Channel:(.*)/) {
16 $mac="";                          43 $channel = $1;
17 $essid="";                         44 }
18 $key="";                            45 if ($linea =~ /^.*Bit\
19 $channel="";                       Rate:(.*)/) {
20 $quality="";                       46 $quality = $1;
21 $signal="";                         47 }
22 $protocol="";                       48 }
23 $numredes++;                       49 }
24 }

```

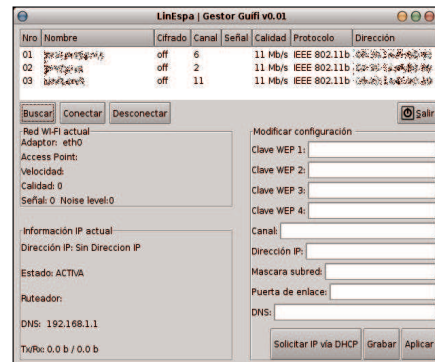


Figura 2: La interfaz nos permite acceder y leer la salida de iwlist de forma sencilla.

Después de unos minutos escribiendo con mi editor de texto pude contar con mi herramienta básica de gestión WI-FI, a quien bauticé "Gestor Güifi". A continuación vamos a ver el código en detalle.

Instalación

Para los impacientes, el Gestor Güifi está disponible en [1].

Se requieren los paquetes libgtk2-perl y wireless-tools que, por lo general, se incluyen en cualquier distribución actual de Linux. De lo contrario, se pueden instalar fácilmente en Debian (o derivados) ejecutando:

```
apt-get install libgtk2-perl wireless-tools
```

Para más información sobre GTK2, el sitio oficial está disponible en [2].

El código

El primer paso es declarar la ubicación del perl y el uso de Gtk2:

```
#!/usr/bin/perl -w
```

Listado 6: Función "guifi_interface"

```

01 sub guifi_interface {
02 my $iface;
03 foreach (`cat /proc/net/wire-
    less`) {
04 if (/\/ (.*)/) {
05 $iface=$1;
06 }
07 }
08 $adaptador->set_text($iface);
09 return $iface;
10 }

```

Listado 7: Cuadro de Configuración

```

01 my $redes =
    Gtk2::HBox->new(1,6);
02 $box1->pack_start($redes, 0,
    1, 0);
03
04 my $frame3 =
    Gtk2::Frame->new("Modificar
    configuración ");
05 $redes->pack_start($frame3, 0,
    0, 10);
06
07 $vbox2 = Gtk2::VBox->new(1);
08 $frame3->add($vbox2);
09
10 my $manual_ip =
    input_data($vbox2,"Dirección
    IP:");
11 my $manual_subnet =
    input_data($vbox2,"Mascara
    subred:");
12 my $manual_gw =
    input_data($vbox2,"Puerta de
    enlace:");
13 my $manual_dns =
    input_data($vbox2,"DNS:");
14
15 $hbox = Gtk2::HBox->new(0, 3);
16 $vbox2->pack_start($hbox, 0,1,
    0);
17 $btn = Gtk2::Button->new
    ('Aplicar');
18 $btn->signal_connect (clicked
    => \&apply_config);
19 $hbox->pack_end($btn, 0, 1,
    0);
20
21 $btn = Gtk2::Button->new
    ('Solicitar IP vía DHCP');
22 $btn->signal_connect (clicked
    => \&dhcp_client);
23 $hbox->pack_end($btn, 0, 1,
    0);

```

```

use Gtk2 -init;
use Gtk2::SimpleList;

```

A continuación, creamos la ventana principal (ver Listado 1).

Seguidamente creamos una tabla que contendrá la lista de puntos de acceso disponibles (ver Listado 2). La tabla se sitúa antes de la instrucción `$win->show_all`, que debe permanecer siempre al pie.

Lo siguiente es crear algunos botones que nos permitirán buscar redes, conectarnos a una red, desconectarnos, y abandonar el aplicativo (ver Listado 3).

Los botones se ven sin duda muy bonitos, pero salvo el botón para cerrar la aplicación, los demás aún no funcionan ya que nos está faltando la función `boton_presionado` (ver Listado 4). Esta función se llama al presionar cualquiera de los tres botones.

Al presionar el botón "Buscar", se llama a la función `guifi_scan` (ver Listado 5) que se detalla a continuación. Esta función ejecuta el comando `iwlist scan` y completa la tabla con los valores obtenidos.

Como podemos ver en el Listado 5, simplemente se invoca `iwlist scan` y se procesa la salida, identificando los valores que nos interesan mediante la utilización de expresiones regulares.

Ya tienes a quien escribir...

La revista que te escucha:
LINUX MAGAZINE

info@linux-magazine.es
subs@linux-magazine.es
anuncios@linux-magazine.es
atrasados@linux-magazine.es
preguntas@linux-magazine.es
correo@linux-magazine.es
eventos@linux-magazine.es
dvd@linux-magazine.es
director@linux-magazine.es
boletin-sub@linux-magazine.es
encuesta@linux-magazine.es
boletin@linux-magazine.es

www.linux-magazine.es

Listado 8: Función "input_data"

```

01 sub input_data {
02   my ($donde, $texto) = @_;
03   my $tmpbox =
04     Gtk2::HBox->new(0, 3);
05
06   $tmpbox->pack_start($tmpbox,
07     0,1, 0);
08
09   $tmpbox->pack_start(Gtk2::Labe
10     l->new($texto),0,1, 0);
11
12   my $tmpvar =
13     Gtk2::Entry->new;
14   $tmpbox->pack_start ($tmpvar,
15     TRUE, TRUE, 0);
16   return $tmpvar;
17 }

```

El adaptador de red se identifica automáticamente mediante la función *guifi_interface* (ver Listado 6). Esta función obtiene el nombre del adaptador WI-FI en */proc/net/wireless*:

Llegados a este punto, con tan pocas líneas de código, ya contamos con una interfaz gráfica que nos muestra las redes disponibles y nos permite conectarnos o desconectarnos con un simple clic. Esto nos ahorra valiosos minutos cuando nos movemos de un sitio a otro, evitándonos usar la consola de texto.

Pero hay más. Lo siguiente sería contar con una ventana que nos permita configurar la dirección IP o solicitar una dirección automáticamente vía DHCP. Para esto creamos un marco que contendrá todos los valores que deseamos configurar, junto con un botón para aplicar los cambios y otro para solicitar una dirección automática.

Para simplificar el código, creamos una función llamada *input_data* (ver Listado 8) para los campos de entrada de datos.

Por último, necesitamos dos funciones más. La primera se llamará *apply_config* y se encargará de aplicar los parámetros de red especificados. La segunda función se llamará *dhcp_client* y justamente invocará el cliente de DHCP para obtener una dirección IP automática (ver Listado 9).

Hemos así explicado el concepto básico del Gestor Güifi. La versión actual incluye más funciones, como un monitor de estado del adaptador de red, configuración WEP y la memorización de los parámetros de los puntos de acceso mediante un fichero de

configuración. Esto último nos permite guardar datos predeterminados para cada punto de acceso y de esta manera memori-

Listado 9: Activando cambios

```

01 sub dhcp_client {
02   my $iface = guifi_interfa-
03     ce();
04   `dhclient $iface`;
05 }
06
07 sub apply_config {
08   my $iface = guifi_interfa-
09     ce();
10   if ($manual_ip->get_text) {
11     $command = "ifconfig $iface
12       ".$manual_ip->get_text." net-
13       mask
14       ".$manual_subnet->get_text;
15     `"$command`";
16
17     $command = "route del -net
18       default";
19     `"$command`";
20
21     $command = "route add -net
22       default gw
23       ".$manual_gw->get_text;
24     `"$command`";
25
26     $command = "echo nameserver
27       ".$manual_dns->get_text." >
28       /etc/resolv.conf";
29     `"$command`";
30   }
31 }

```



zar direcciones de IP, claves WEP u otros parámetros para cada punto de acceso.

Si bien el Gestor Güifi es un guiión muy simple, hay muchos usuarios que ya lo han adoptado e inclusive forma parte de la última versión de LinEspa [3].

Es necesario advertir que este guiión necesita ser ejecutado como root (o mediante sudo) para contar con los permisos necesarios para realizar cambios en la configuración de red. Como toda tarea a ser ejecutada como root, es necesario tomar las debidas medidas de seguridad para evitar comprometer el sistema. ■

RECURSOS

- [1] Código del programa explicado en este artículo: <http://julian.coccia.com/guifi>
- [2] Sitio oficial de GTK2: <http://www.gtk.org/>
- [3] LinEspa (Linux Español): <http://www.linespa.com/>

EL AUTOR

Julian Coccia es el fundador de la Asociación Linux Español y colabora con la comunidad del software libre mediante diversos proyectos, entre ellos el foro de ayuda www.linuxespanol.com y la distribución LinEspa www.linespa.com.

