Prof. Mileti, P.

Programando un front-end con Gambas

Gambas es un excelente lenguaje y permite, entre tantas otras cosas, reutilizar aplicaciones de consola. A lo largo de este artículo veremos como programar un front-end para la aplicación de consola ImageMagick.

Un front-end no es mas que una interfaz gráfica de usuario cuya finalidad es recolectar los datos necesarios para activar ciertos comandos de consola. Se podría decir que un usuario de un programa del tipo front-end es un usuario que ejecuta comandos de consola sin escribirlos o sin ser consciente de ello.

Todos sabemos que en el mundo GNU/Linux existen potentes aplicaciones de consola, pero el desconocimiento sobre como utilizarlas hace que muchos usuarios las desechen, buscando alternativas gráficas a ellas. Entonces, les propongo en este artículo, tomar una aplicación de consola y dotarla de interfaz gráfica para facilitar su uso.

Sobre ImageMagick

Según Wikipedia ImageMagick es una aplicación que sirve para crear, editar y componer imágenes, puede leer, convertir y guardar imágenes en una gran variedad de formatos. Si consultamos la web oficial de ImageMagick veremos que promocionan su software como una aplicación que típicamente es utilizada desde la linea de comandos. Para instalar ImageMagick en distribuciones del tipo Debian no hay mas que abrir la consola y escribir:

```
sudo apt-get install imagemagick
```

La propuesta de este apunte

Supongamos ahora que tengo una imagen del tipo bitmap (vamos a llamarla unaimagen.bmp) y la quiero convertir a formato PNG (con el nombre unaimagen.png). Una posible solución, si tengo ImageMagick instalado, es ejecutar desde consola:

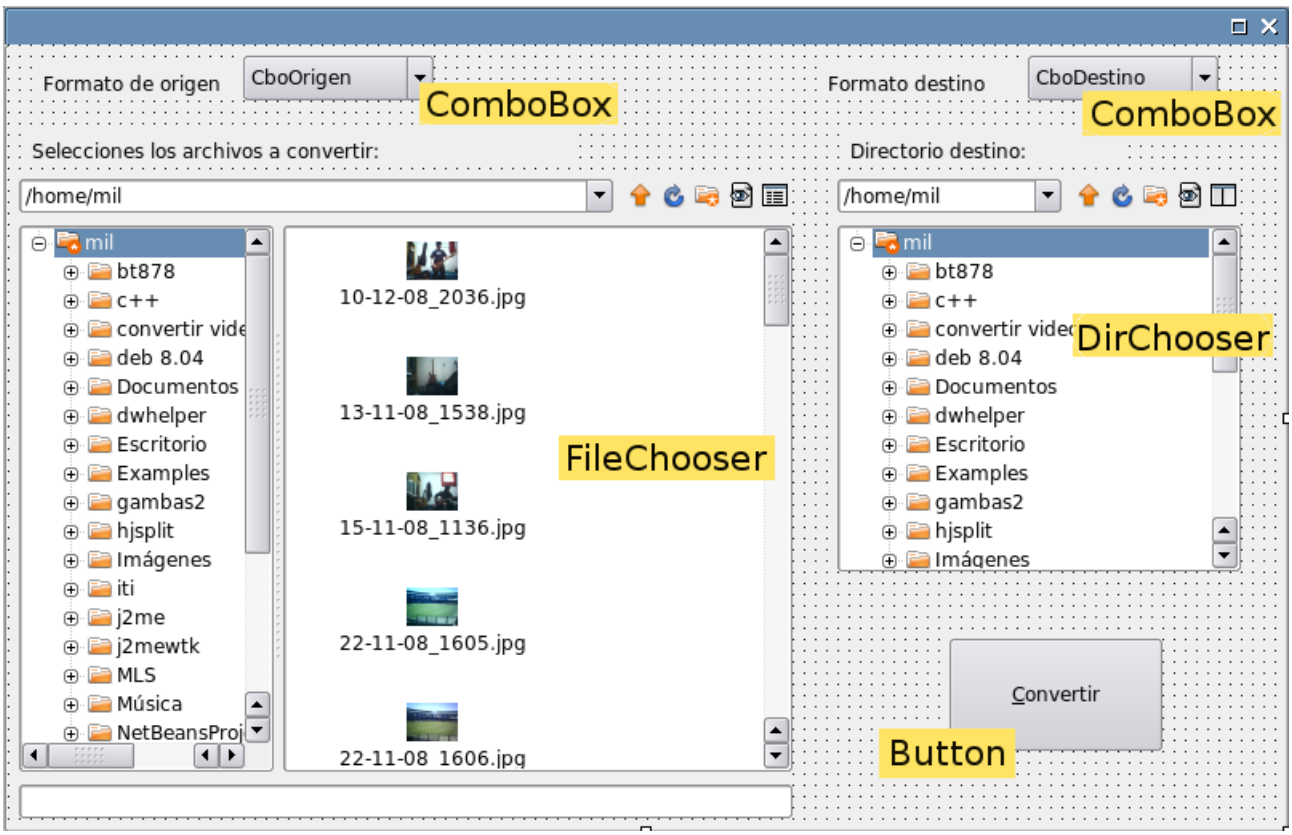
```
convert unaimagen.bmp unaimagen.png
```

Ahora, que pasa si tengo en un directorio 200 fotos en formato BMP y quiero elegir, mientras las voy viendo en pantalla, cuáles convertir...

Pensaron ya como hacerlo? Tal vez coincidan conmigo, creamos un front-end en gambas y listo.

Creando el front-end para convertir imágenes.

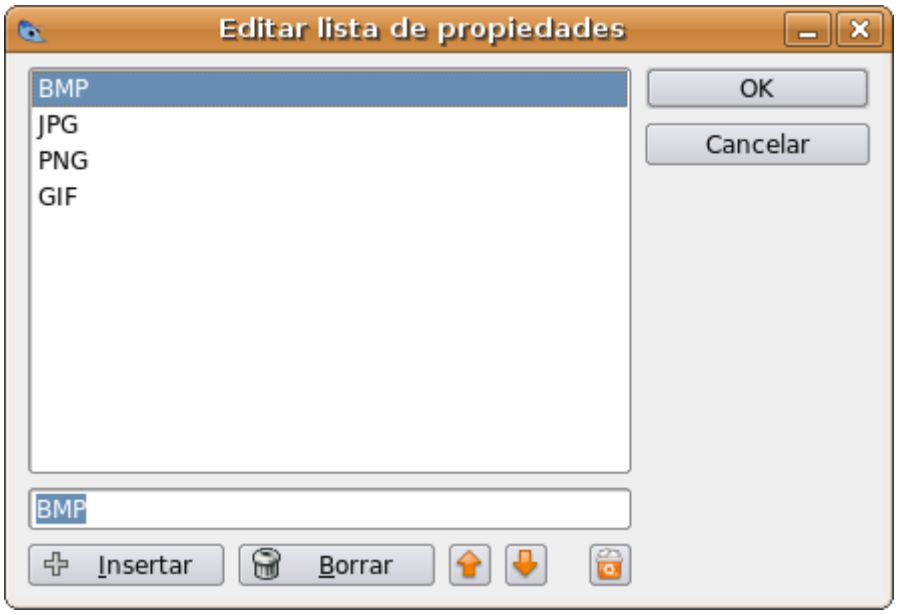
Iniciamos Gambas y creamos un nuevo proyecto del tipo aplicación gráfica. Una vez creado el proyecto, les propongo que diseñen el siguiente formulario:



Se trata de un formulario compuesto por dos controles ComboBox, un control FileChooser, un DirChooser, un Button y el resto son controles Label.

Los controles FileChooser y DirChooser se encuentran en la caja de herramientas bajo la solapa Dialog. El resto de los controles se hallan en la solapa Form.

Ubicamos primero los ComboBox, a uno lo llamaremos (mediante la propiedad Name) CboOrigen y al otro CboDestino. La propiedad List del ComboBox contiene los elementos de su lista desplegable. Ambos Combos tendrán los mismos elementos. En este caso, los elementos serán los formatos de imagen que nuestro front-end aceptará. Propongo entonces cargar los siguientes elementos en la propiedad List:



Otra propiedad que debemos modificar en los ComboBox es ReadOnly. Si ReadOnly es True el usuario solo podrá seleccionar un elemento de la lista desplegable. Si ReadOnly es False el usuario podrá escribir lo que desee en él. En este caso no es aconsejable que el usuario pueda escribir en ellos, así que le asignamos a ReadOnly el valor True.

Llego el momento de agregar los controles FileChooser y DirChooser. El FileChooser lo vamos a usar para que el usuario pueda visualizar y seleccionar los archivos que desea convertir. El DirChooser va a permitir seleccionar el directorio en donde se guardaran las nuevas imágenes resultantes de la conversión. Dejamos el mismo nombre por defecto para cada uno por ser lo suficientemente descriptivos a la hora de leer el código.

Por ultimo agregamos un control Button al que llamaremos BtnConvertir y por medio de la propiedad Text colocamos en su interior la leyenda Convertir.

La interfaz gráfica de usuario ya está terminada, solo resta darle vida mediante código.

Codificando el front-end

En Gambas. al iniciar una aplicación se dispara el evento Form_Open, es aquí donde vamos a definir los diferentes filtros del control FileChooser. Los filtros nos van a asegurar que sólo se muestren archivos que sean del tipo BMP, JPG, PNG o GIF. Estos filtros serán aplicados cuando seleccionemos el tipo de imagen en el control CboOrigen. Entonces, si en CboOrigen tenemos cuatro elementos, en la propiedad Filter del FileChooser debemos tener cuatro filtros. La propiedad Filter espera recibir un array de string. Cada filtro se crea pasando dos strings, el primer string es el fltro propiamente dicho y el segundo es la leyenda del fltro. Si queremos tener dos filtros tendremos que pasar cuatro strings, si queremos tres filtros pasaremos seis strings, y así se incrementan los strings de a dos por cada filtro que se agregue. Por eso, en nuestro caso, asignamos en Filter cuatro pares de string separados cada uno ellos por comas. Vean como queda el evento Form_Open:

```
PUBLIC SUB Form_Open()
  FileChooser1.Filter = ["*.bmp", "Imagenes BMP",
   "*.jpg;*.jpeg", "Imagenes JPG",
   "*.png", "Imagenes PNG",
   "*.gif", "Imagenes GIF"]
   FileChooser1.Multi = TRUE
   CboOrigen.Index = 0
   CboDestino.Index = 1
END
```

En el código anterior también indicamos que en el FileChooser esta permitido seleccionar múltiples archivos (propiedad Multi) y que la opción predeterminada de CboOrigen es el primer elemento de la lista (posición cero, BMP), mientras que la opción predeterminada para CboDestino, será el segundo elemento (posición uno, JPG).

Ya tenemos perfectamente inicializada la aplicación, ahora pensemos a qué eventos debe responder nuestro programa. Son solo dos lo eventos involucrados. Cuando el usuario seleccione el tipo de imagen que desea convertir (por medio de CboOrigen), debemos activar el filtro correspondiente en el FileChooser y cuando se pulse el botón Convertir se debe iniciar el proceso de conversión. Veamos como se programa el primer evento:

```
PUBLIC SUB CboOrigen_Click()
  FileChooser1.FilterIndex = CboOrigen.Index
END
```

Simplemente indicamos qué elemento del array cargado en Filter deseamos activar. Justamente la posición de cada filtro se corresponde con la posición de cada elemento del ComboBox. A eso apunta el código anterior.

Llegó el momento de programar el segundo evento, el más importante, el que desencadena la conversión de las imágenes seleccionadas. Los archivos que fueron seleccionados en el FileChooser están contenidos en un array de string que podemos consultar mediante la propiedad SelectedPaths. Ahora, qué pasa si el usuario pulsa Convertir y no ha seleccionado ningún archivo. En este caso deberíamos validar que existan archivos seleccionados para convertir, lo podemos hacer consultando la cantidad de elementos de la propiedad SelectedPaths mediante Count. Por otro lado, debemos evitar que el formato de origen sea el mismo al formato destino, esto lo haremos verificando que la propiedad index de ambos combos no sean iguales. Lo dicho anteriormente está programado en los dos primeros IF del siguiente código, en donde al detectar alguna de la posibilidades planteadas, avisamos al usuario y salimos del evento.

```
PUBLIC SUB Button1_Click()
  DIM archivo AS String
  IF FileChooser1.SelectedPaths.Count = 0 THEN
    Message.Error("Debe seleccionar los archivos a convertir.")
    RETURN
  END IF
  IF CboOrigen.Index = CboDestino.Index THEN
    Message.Error("El formato destino debe ser diferente al de origen.")
    RETURN
  END IF
  ME.mouse = 150  ' puntero wait
  Button1.Text = "Procesando..."
  WAIT 0.1  ' para redibujar el formulario
  FOR EACH archivo IN fileChooser1.SelectedPaths
    Convertir(archivo)
  NEXT
  ME.mouse = 2  'volvemos al puntero por defecto
  Button1.Text = "&Convertir"
  Message.Info("Operación finalizada.")
END
```

Con Me.mouse cambiamos el puntero del mouse para que el usuario sepa que la aplicación está trabajando y debe aguardar. Luego se cambia la leyenda del botón y se redibuja el formulario con la llamada a WAIT.

Lo genial del código anterior esta acá:

```
  FOR EACH archivo IN fileChooser1.SelectedPaths
    Convertir(archivo)
  NEXT
```

La variable archivo es del tipo String, y a lo largo del FOR irá tomando el valor de cada uno de los elementos de SelectedPaths. Sería algo así como decir: para cada archivo que está seleccionado en el FileChooser1, llamar a un procedimiento que lo convierta. El procedimiento Convertir va a recibir como parámetro un string que contendrá la ruta absoluta mas el nombre del archivo que se va a convertir.

La conversión la vamos a realizar dentro de un procedimiento llamado Convertir, lanzando un comando de consola mediante la sentencia EXEC. Recuerdan que el comando era:

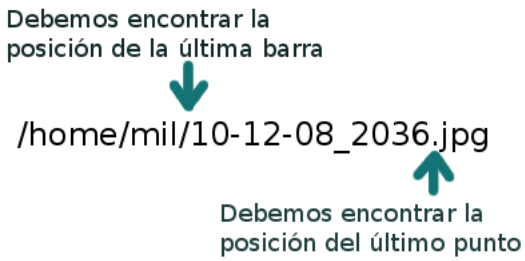
convert unaimagen.bmp unaimagen.png

El primer valor para el comando convert ya lo tenemos, será el parámetro que le pasemos al procedimiento Convertir. Nos faltaría el segundo, el nuevo nombre que tendrá. Y aquí vamos a tener que programar un poquito.

A partir del directorio destino seleccionado, el nombre del archivo que estamos convirtiendo y el nuevo formato que tendrá, debemos crear el segundo parámetro para el comando convert. Tomemos el siguiente ejemplo para que vean la dificultad que se presenta:

```
PUBLIC SUB Convertir(archivo AS String)
  DIM nuevoNombre AS String
  nuevoNombre = DirChooser1.SelectedPath & "/" & archivo & "." & LCase(CboDestino.Text)
  EXEC ["convert", archivo, nuevoNombre] WAIT
END
```

Esto es lo que planteaba anteriormente, es simple y claro, pero tropezamos con que el parámetro archivo contiene además la ruta del archivo origen, por ejemplo si selecciono una imagen de mi directorio home, la variable archivo valdrá "/home/mil/10-12-08_2036.jpg" y para que el código anterior funcione debería valer "10-12-08_2036", es decir, debería contener solo el nombre del archivo. Surgido este problema, vamos a trabajar con ciertas funciones con nos permita extraer de la cadena anterior el nombre del archivo. Veamos como proceder para solucionar el inconveniente:

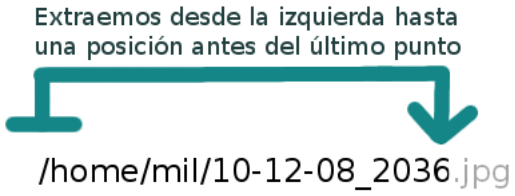


Para ello Gambas dispone de una función llamada RInStr que nos devuelve la posición de la primer ocurrencia empezando desde la derecha, la sintaxis es:

```
Posición = RInStr ( Cadena AS String , Subcadena AS String [ , Inicio AS Integer ] )
```

Con el siguiente código ubicamos la posición de la última barra y del último punto:

```
posicionUltimoPunto = RInStr(archivo, ".")
posicionUltimaBarra = RInStr(archivo, "/")
```



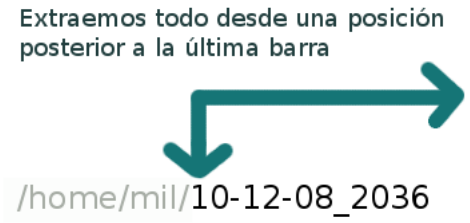
La función Left$ vendrá a nuestra ayuda, nos devuelve los primeros x caracteres desde la izquierda, la sintaxis es:

```
Resultado = Left$ ( Cadena AS String [ , Longitud AS Integer ] )
```

Truncamos entonces hasta un lugar antes del punto:

```
nuevoNombre = Left$(archivo, posicionUltimoPunto - 1)
```

Finalmente



Para lograr esto nos valdremos de la funcion Mid$ que extrae x cantidad de caracteres a partir de una posición determinada, su sintaxis:

```
Resultado = Mid$ ( Cadena AS String , Inicio AS String [ , Longitud AS Integer ] )
```

Extraemos todo lo que encuentra a continuación de la ultima barra:

```
nuevoNombre = Mid$(nuevoNombre, posicionUltimaBarra + 1)
```

Ya logramos extraer el nombre del archivo, sin la extensión y su ruta. Podemos agregar la ruta de destino y la extensión con la siguiente linea:

```
nuevoNombre = DirChooser1.SelectedPath & "/" & nuevoNombre & "." & LCase(CboDestino.Text)
```

Ahora sí, en la variable nuevoNombre tenemos conformado el nombre de destino para el comando convert, el cual lanzamos a través de EXEC:

```
EXEC ["convert", archivo, nuevoNombre] WAIT
```

El WAIT lo utilizamos para que la ejecución no continúe hasta que no finalice la ejecución del comando de consola.

Pasando en limpio la expuesto anteriormente el procedimiento completo es el siguiente:

```
PUBLIC SUB Convertir(archivo AS String)
  DIM posicionUltimoPunto AS Integer
  DIM posicionUltimaBarra AS Integer
  DIM nuevoNombre AS String
  posicionUltimoPunto = RInStr(archivo, ".")
  posicionUltimaBarra = RInStr(archivo, "/")
  nuevoNombre = Left$(archivo, posicionUltimoPunto - 1)
  nuevoNombre = Mid$(nuevoNombre, posicionUltimaBarra + 1)
  nuevoNombre = DirChooser1.SelectedPath & "/" & nuevoNombre & "." & LCase(CboDestino.Text)
  EXEC ["convert", archivo, nuevoNombre] WAIT
END
```

De esta manera damos por finalizado este front-end. Si están conforme con su funcionamiento pueden profundizar agregando nuevas funcionalidades, por ejemplo, se podría indicar también el tamaño en píxel para

la imagen final, con el fin de redimensionarla. También, siguiendo el mismo criterio de este artículo, pueden crear un fron-end, para convertir archivos de video mediante ffmpeg.

Para terminar les dejo dos links: el primero hacia el código fuente del fron-end y el segundo hacia el paquete Deb para instalar el front-end junto con ImageMagick.

http://pablomileti.googlepages.com/milimageconverter_0.0.10.orig.tar.gz

http://pablomileti.googlepages.com/milimageconverter_0.0.10-1_all.deb

Pablo Mileti
pablomileti@gmail.com