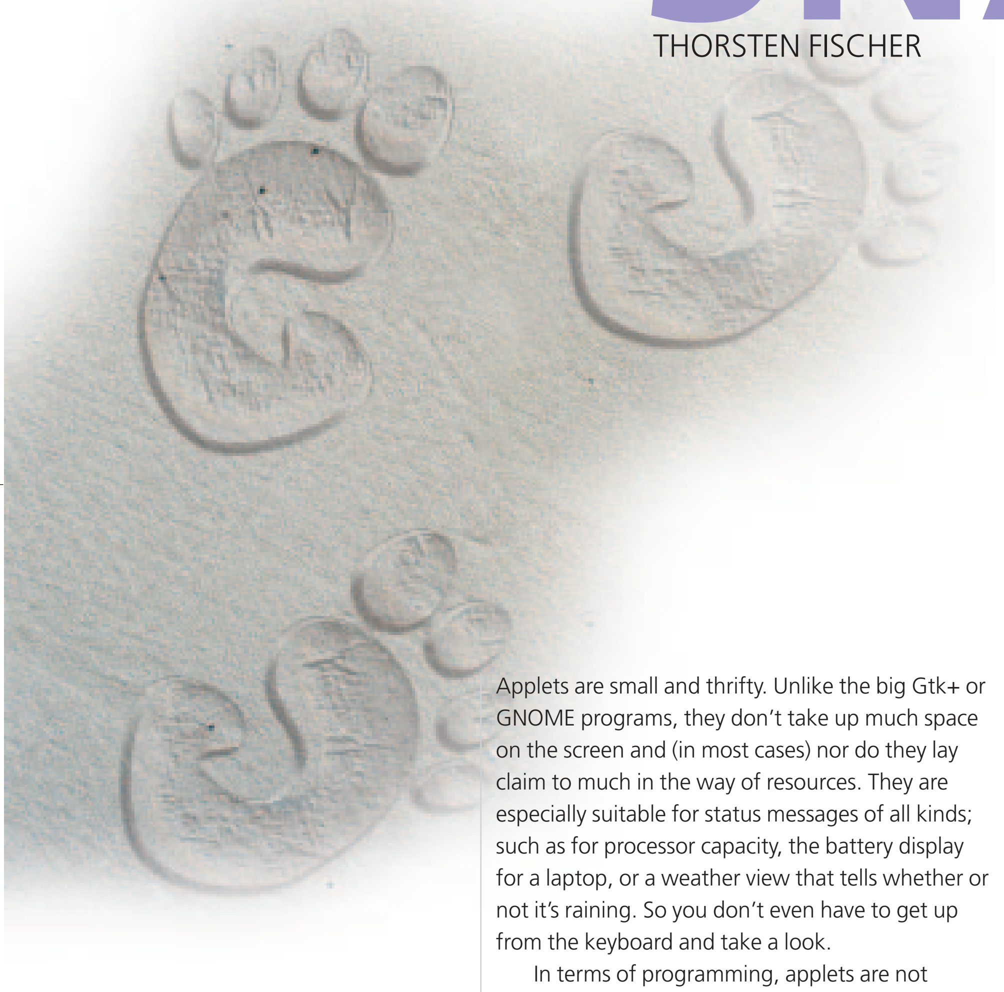
Applets

A LITTLE SNACK

THORSTEN FISCHER



You don't always have to use a fully-grown program. Lots of little tasks can also be handled by applets, which live in the panel and are just waiting to be called up.

Applets are small and thrifty. Unlike the big Gtk+ or GNOME programs, they don't take up much space on the screen and (in most cases) nor do they lay claim to much in the way of resources. They are especially suitable for status messages of all kinds; such as for processor capacity, the battery display for a laptop, or a weather view that tells whether or not it's raining. So you don't even have to get up from the keyboard and take a look.

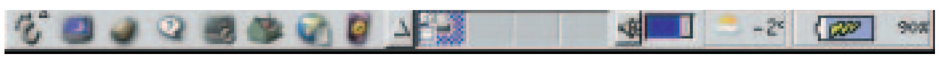
In terms of programming, applets are not substantially different from normal GNOME programs. In Listing 1 you can see a normal program for GNOME, showing only an empty window. It contains the usual steps: Define internationalisation, create application window, connect with the standard signals and then display. The definitions in lines 3 to 5 are created automatically and passed on during compilation, when you are working with a source text tree that you have created previously.

An empty applet behaves very similarly, and at this point (as a variation on the Hello world theme) I would like to call it the ´Bon appetit´ applet. And accordingly, what does it do first but display *Bon appetit*. This can be seen in Listing 2.

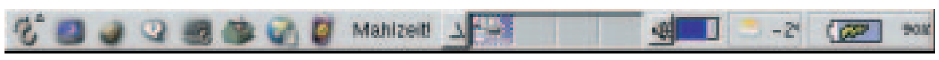
The defines in lines 4 to 6 have again the same origin as before. In line 2 the header file *applet-widget.h* still has to be included, to allow access to the library function for applets.

The first really new part can be seen in line 16: The program is not initialised by *gnome_init()*, but by a command from *applet_widget_init()*. The parameters of this function are the program name, the program version, the command line parameters *argc* and *argv* and the arguments for parsing the command line by the library *libpopt*. But this procedure is not going to be discussed just yet.

Applets are a particular type of widget, as can be seen from the command *applet_widget_ new()* in line 18. The applet must communicate with the panel, so as to be able to insert itself at the right spot. Occasionally this communication fails right at the start, so that no applet can be created. If so, then there is a void pointer in our example *applet*, which we catch in the lines 19 to 22 and then end the applet with an error message. In the following lines a label is merely created and inserted into the applet.



[top] Figure 1: The GNOME panel with weather applet.



[below] Figure 2: The same panel, now with 'Bon appetit' applet.

As the applet functions perfectly normally as a container-widget, in theory any other widget could be inserted. But for the moment it remains to be seen which widget would produce sense and which would not.

Applets require their own parameters during compilation. If our file is called *Bon appetit_applet.c*, then the following command is correct:

```
gcc Bon appetit applet.c `gnome-config libs➜
cflags gnome gnomeui applets` -o bon appetit➜
_applet
```

If the applet is compiled and started (which can also be done from the command line as with a normal program) then it appears as if drawn by a ghostly hand in the panel, as can be clearly seen from Figure 2.

Decoration

So much then for the first applet. If you click on it with the right mouse button, two menu points called *Move* and *Remove* will be seen, and their names speak for themselves. You can also see another menu point, *Panel*, behind which the familiar panel configuration is hiding. At this point you may wish to add your own menu points. This can be done without any trouble at all. With the function *applet_widget_register_stock_callback* new menu points can be generated and also immediately connected with the callback functions.

In our example, which is listed in full in Listing 3, there is a menu point *About*, which displays a corresponding box. There are also functions that remove the menu points again or make complete sub menus.

Notification

If you have put the code together and perhaps even installed it in its own code tree (with the aid of which compilation on any systems can be done very easily) then you would surely also like the applet to be put to good use. To do this it should appear in the menus seen by the user when clicking on the panel with the right mouse button and selecting *add applet*. For this, basically, a *.desktop* file is necessary, which should look roughly like Listing 4. This file must be copied into the directory

```
prefix/share/applets/category
```

where the prefix is the root directory of the GNOME installation (in SuSE for example */opt/gnome*) and the category corresponds to the type entry in the *.desktop* file, thus here *Application*.

More stuff

The requirements for an applet are of a somewhat different nature than those of an ordinary program with windows. Of course, I that know that in the sense of the X-server an applet also has a window; ultimately a window is nothing but an area in which it is possible to draw.

For example it should be able to react to vertical and horizontal orientations on its own, and also to changes in size, because after all, GNOME has seven different sizes for an applet, extending from 12 to 128 pixels in size. As soon as the size of the panel

Listing 1: Skeleton of a normal GNOME

```
#include gnome.h

#define PACKAGE "gnome-std"
#define VERSION "0.1.0"
#define GNOMELOCALEDIR "/opt/gnome/share/locale"

int main(int argc, char *argv [])
{
  GtkWidget *app;

  bindtextdomain (PACKAGE, GNOMELOCALEDIR);
  textdomain (PACKAGE);

  gnome init (PACKAGE, VERSION, argc, argv);

  app = gnome app new (PACKAGE, PACKAGE);
  gtk signal connect (GTK OBJECT (app), "delete event", gtk main quit, NULL);
  gtk signal connect (GTK OBJECT (app), "destroy event", gtk main quit, NULL);

  gtk widget show all (app);

  gtk main();

  return TRUE;
}
```

Listing 2: The first bon appetit applet

```
#include gnome.h
#include applet-widget.h

#define PACKAGE "Bon appetit applet"
#define VERSION "0.1.0"
#define GNOMELOCALEDIR "/opt/gnome/share/locale"

int main(int argc, char *argv [])
{
  GtkWidget *applet;
  GtkWidget *label;

  bindtextdomain (PACKAGE, GNOMELOCALEDIR);
  textdomain (PACKAGE);

  applet widget init (PACKAGE, VERSION, argc, argv, NULL, 0, NULL);

  applet = applet widget new (PACKAGE);
  if (!applet)
  {
    g error ("cannot create applet.\n");
  }

  label = gtk label new (" Bon appetit! ");
  applet widget add (APPLET WIDGET (applet), label);

  gtk widget show all (applet);

  gtk main();

  return TRUE;
}
```

Listing 3: A full applet

```
#include gnome.h
#include applet-widget.h

#define PACKAGE "Bon appetit-applet"
#define VERSION "0.1.0"
#define GNOMELOCALEDIR "/opt/gnome/share/locale"

void change_pixel_size_callback (appletWidget *applet, int pixels, gpointer data)
{
  printf ("The panel is %d pixels in size.\n", pixels);
}

void change_orient_callback (appletWidget *applet, PanelOrientType orient, gpointer data)
{
  switch (orient)
  {
    case ORIENT_UP:
      printf ("The panel is on the bottom.\n");
      break;
    case ORIENT_DOWN:
      printf ("The panel is at the top.\n");
      break;
    case ORIENT_LEFT:
      printf ("Right of the panel!\n");
      break;
    case ORIENT_RIGHT:
      printf ("Left of the panel!\n");
      break;
    default:
      printf ("No orientation !?!\n");
      break;
  }
}

void about_callback (appletWidget *applet, gpointer data)
{
  GtkWidget *dialog;
  const gchar *authors [] = {"Thorsten Fischer @derfrosch.de>", NULL};

  dialog = gnome_about_new (PACKAGE, VERSION, "(c) 2000 Thorsten Fischer", authors,
        "Sampleapplet for Linux-Magazine.\n Code subject to the GPL.", NULL);

  gtk_widget_show (dialog);

  return;
}

int main(int argc, char *argv [])
{
  GtkWidget *applet;
  GtkWidget *label;
  GtkWidget *frame;

  bindtextdomain (PACKAGE, GNOMELOCALEDIR);
  textdomain (PACKAGE);

  applet_widget_init (PACKAGE, NULL, argc, argv, NULL,0,NULL);

  applet = applet_widget_new (PACKAGE);
  if (!applet)
  {
    g_error ("Cannot create applet.\n");
  }
  gtk_signal_connect (GTK_OBJECT (applet), "change_orient",
                   GTK_SIGNAL_FUNC (change_orient_callback), NULL);
  gtk_signal_connect (GTK_OBJECT (applet), "change_pixel_size",
                   GTK_SIGNAL_FUNC (change_pixel_size_callback), NULL);
  applet_widget_register_stock_callback (APPLET_WIDGET (applet), "about", GNOME_STOCK_ME➲
NU_ABOUT,
                                          _("About"), about_callback, NULL);


  frame = gtk_frame_new (NULL);
  applet_widget_add (APPLET_WIDGET (applet), frame);

  label = gtk_label_new (" Bon appetit! ");
  gtk_container_add (GTK_CONTAINER (frame), label);

  gtk_widget_show_all (applet);

  gtk_main();

  return TRUE;
}
```

alters – and when the applet is first started – the signal *change_ pixel_size* is sent out, which can be caught by a corresponding callback.

In the example in Listing 3 this is done using the function *change_pixel_ size_callback*. Although as a programmer one should never work on the assumption of standard situations, comparisons with possible pixel sizes can be done using constants; they all bear the prefix *PIXEL_SIZE_* and are then called *ULTRA_TINY*, *TINY*, *SMALL*, *STANDARD*, *LARGE*, *HUGE* and *RIDICULOUS*, with a size of 12, 24, 36, 48, 64, 80 or 128 pixels respectively. No one can do anything with the last size, but it's nice to take a look at the anti-aliasing of the icons.

The representation of a widget may also depend on the orientation of the panel. It is therefore convenient that an applet always receives a signal from the panel, when it changes its orientation, and also of course immediately on first starting the applet. It is called *change_orient* and is caught in the example (Listing 3) with the callback function *change_ orient_callback*.

It may appear miraculous that the constants, which can exist as the result, produce the exact opposite of what they say at first glance. However, it is not actually the position of the panel that is being passed on, but the direction in which a menu would extend if it had to be created.

Listing 4: The desktop file for an applet

```
[Desktop Entry]
Name=Bon appetit applet
Comment=An applet which says Bon appetit
Exec=Bon appetit applet
Icon=
Terminal=0
Type=Application
```

It should also be noted that the connection with the signal should take place before additional content is added into the still-empty applet with *applet_widget_add()*. Otherwise this does away with the option of being able to respond accordingly to the orientation.

Happy Gnomes! ■

The author

Thorsten Fischer is a student of computer science and/or media constancy. He can be contacted at frosch@derfrosch.de.

Info

Source texts for the article: http://www.derfrosch.de/weichewaren/linux-magazin.html
Gnome-core applet writing HOWTO: http://cvs.gnome.org/lxr/source/gnome-core/panel/APPLET_WRITING
Gnome applet Tutorial: http://developer.gnome.org/doc/tutorials/applet/index.html

■

AD??