1 A small application for the Zauru

The Zaurus comes with some PIM tools, one of which is a small appointment manager. The data of this manager can be synchronized with a PC: on Linux or Windows, the qtopia desktop can be used to synchronize the contents of the Zaurus calendar with the calendar on the PC. In an earlier issue of Toolbox, an algorithm was presented to calculate when it's Easter day. Below, a small program is presented which inserts easter day in the calendar of the Zaurus (or the Qtopia desktop)

The program is developed on Linux, but could have been written on Windows just as easily. The only thing that changes is the location of the configuration file. The datebook application (calendar) stores its information in a file called datebook.xml. The information is stored in XML. To add the easter days to the calendar, it is necessary to add some elements in the XML tree. FPC is shipped with units that implement the DOM specifications of the W3 consortium, so this can be done quite easily.

Let's have a look at the format of the datebook data file. It looks something like this:

```
<?xml version="1.0"?>
<events>
  <event uid="1079034434" categories="1058558753"
         description="Something new" location="Here"
         timezone="None" start="1078999200" end="1079002800"
         type="AllDay"/>
  <event uid="1079034435" categories="1058558752"
         description="Meeting" location="Home"
         timezone="None" start="1079002800" end="1079006400"
         type="Timed"/>
</events>
```

The original entries have been splitted over several lines for printing purposes. As can be seen, the format is quite simple. All datebook entries are stored in `event` elements below the `events` element in the XML tree.

Each event has a `uid` attribute, which is needed for synchronization purposes. For this article, we'll ignore the contents of this attribute. What follows is a `category` attribute, which is an encoding of the various categories that have been defined by the user. Then follows the `description` attribute, which contains the text that was entered for the event. Next are another 2 optional attributes: the `location` of the event and the `timezone` in which the event occurs. After this, three more interesting attributes occur: the `start` and `end` time, plus the type of event. A 'Timed' event occurs on a specific time during the day, the 'AllDay' event is specific to the day.

This provides enough information to be able to insert an entry for easter day in the calendar. All that must be done now is to read the XML file, insert one or more `event` elements, and write the file again. Obviously the type of event will be 'Allday', and the start and end time can be calculated: it's a Unix timestamp: the number of elapsed seconds since January 1, 1970. The start time will be set to 00:00, while the end time will be set to 23:59. As description of the event 'Easter Day' will be used.

The program code is really quite simple:

```
begin
  ProcessCommandLine;
  ReadXMLFile(Doc,FileName);
  Try
    AddEasterDays(Doc,StartYear,EndYear);
```

```
    WriteXMLFile(Doc,FileName);
  Finally
    Doc.Free;
  end;
end.
```

First, the command-line is processed. The program takes 3 command-line arguments. A start year, optionally an end year, and a filename. If no end year is given, only the easter day for the start year will be inserted. The program contains defaults for the location of the location of the datebook.xml file.

Then the XML file is read. The `ReadXMLFile` call is part of the xmlread unit that comes with FPC. It reads an XML file and stores the XML tree in a `TDOMDocument` class instance, as described by the DOM specifications. The TDOMDocument class (and all other DOM elements) are part of the Dom unit.

After this, the easter days are added to the DOM tree, and the file is written to disk, using the `WriteXMLFile` call, which can be found in the `xmlwrite` unit.

The AddEasterdays call is a simple loop:

```
Procedure AddEasterDays(Doc : TXMLDocument;
                        StartYear,EndYear : Integer);

Var
  N : TDomNode;
  Y : Integer;
  E : TDomElement;

begin
  With Doc do
    begin
    N:=FindNode('events');
    If (N=Nil) or (Not (N is TDomElement)) then
      begin
      Writeln('Could not locate events in datebook file.');
      Halt(1);
      end;
    E:=TDomElement(N);
    For Y:=StartYear to EndYear do
      AddEasterEvent(E,Y);
    end;
end;
```

It starts by looking for the 'events' node in the DOM tree. The standard 'FindNode' call does this. If the node was found, it runs a simple loop from start till end year, and passes the year and the event node to the `AddEasterEvent` call, which actually adds the entry:

```
Procedure AddEasterEvent(Events: TDomElement; Year: Integer);

Var
  D : TDateTime;
  E : TDomElement;

begin
  D:=EasterDay(Year);
```

```
  E:=Events.OwnerDocument.CreateElement('event');
  E['description']:='Easter day';
  E['timezone']:='None';
  E['start']:=IntToStr(EpochTime(D+EncodeTime(12,0,0,0)));
  E['end']:=IntToStr(EpochTime(D+EncodeTime(23,59,0,0)));
  E['type']:='AllDay';
  Events.AppendChild(E);
end;
```

First the exact easter day is calculated using the `EasterDay` function, described in an earlier issue of Toolbox. Then a new element is created. An element cannot be created just like that, it must be requested from the `TDOMDocument` class, which is available through the `OwnerDocument` property of the `Events` instance. The attributes of an XML element are declared as the default property of the `TDomElement` node, so they can be easily set. As can be seen from the code above, setting the attributes to their desired values. The EpochTime function calculates the Unix epoch time starting from a `TDateTime` date/time indication.

That is all there is to it. All that remains to be done is to compile the program:

```
home: >fpc -Parm -XParm-linux- addeaster.pp
```

The `-Parm` parameter tells FPC that it should compile for the arm processor (it will choose the correct arm cross-compiler) and the `-XParm-linux-` tells the compiler that the arm cross-compilation utilities (such as the `as` assembler, and `ld` linker) are installed with the `arm-linux-` prefix, i.e. they're called `arm-linux-as` and `arm-linux-ld`.

Remains to transfer the program to the Zaurus (using ftp or scp) and run it on the Zaurus:

```
# uname -a
Linux localhost 2.4.6-rmk1-np2-embedix #1 2002Çŕ 7ůî30Æü(šÐ) 08żþ59Êň46ÉÃ
JST armv4l unknown
# ./addeaster 2004 2005
# cd /home/root/Applications/datebook
# cat datebook.xml
<?xml version="1.0"?>
<events>
  <event description="Easter day" timezone="None"
         start="1081684800" end="1081688400" type="AllDay"/>
  <event description="Easter day" timezone="None"
         start="1111924800" end="1111928400" type="AllDay"/>
</events>
```

And the easter day entries should appear in the calendar of the Zaurus. There is a small caveat: if the datebook application is still running, it must be stopped and restarted. Even if it is not visible on the screen, it may be that it is still running in the background (the Zaurus does this to be able to start applications faster). Use the process manager of the Zaurus to kill it, and then simply start it.