Using PDF to distribute code

Joe Armstrong
SICS
joe@sics.se

12 - December 2001

PDF

The conventional way of distributing code plus documentation is to create some documentation and the code and squash it all together into an archive file, often a gzipped tar file or an rpm file. In order to read the documentation you have to unpack the archive file and search for the appropriate Readme file or for some PDF or HTML file which if you're lucky you might find carefully hidden in the distribution.

I usually start by reading the documentation - that way I can find out what the program is *supposed* to do, and decide if its worth looking at.

To do this I propose distributing software packages as single PDF files - and to pack the code *inside* the PDF files.

Then, if you just want to read the documentation you don't have to do anything - you just read it, but if you want the code you can unpack it from the documentation.

To do this I have written `pdf.erl` a simple program that can manipulate PDF files.

> This PDF file contains code which can be used to pack and unpack code from a PDF file

the module `pdf.erl` (hidden in this file) exports two functions:

- `pdf:pack(In, File, Out)` - this takes a PDF file `In` and an additional file `File` and produces `Out` a new PDF file.

  For example, the call
  `pdf:pack("pdfdoc.pdf", "pdf.erl", "erlpdf.pdf")` was used to produce the file you are reading.

- `pdf:unpack(File)` unpacks any file that has been packed into `File` - for example `pdf:unpack("erlpdf.pdf")` creates `pdf.erl` from this PDF file.

Exercise: extract `erlpdf.erl` *from this file*[1] *then run it on itself and verify that it works correctly.*

[1] I won't tell you how to do this :-)

Details

You will absolutly not want to read *Adobe portable document format version 1.4 ISBN 0-201-75839-3* available on the net at `http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html` which describes the PDF internals in more more detail than any human being could conceivably desire.

I have read it so now I can answer the question *how do you add a file to PDF?*

Answer: If you go to the end of a PDF file you'll find something like this:

```
trailer
<<
/Size 73
/Root 71 0 R
/Info 72 0 R
>>
startxref
27277
%%EOF
```

all these little numbers mean something.

Here's an annotated example of what you have to append to the end of the file to add a "hidden" file:

```
<<
/Size 73
/Root 71 0 R
/Info 72 0 R
>>
startxref
27277
%%EOF
73 0 obj <<
/Length 5
/Comment (erlangAddedFile)
>>
stream
abc
endstream
endobj
xref
73 1
0000028818 00000 n
trailer
<<
/Size 74
/Root 71 0 R
/Prev 27277
/Info 73 0 R
>>
```

```
27 startxref
28 28923
29 %%EOF
```

In the above:

- Line 9 introduces a new object (object number 73) - why 73? Line 2 said that the size of the old object table was 73, since the objects are numbered from 0 then the first free object is object number 73.
- Line 10 says the length of the new object is 5 bytes -. the object itself is a *stream* object (a stream object just contains any sequence of bytes and is delimited by the keywords `stream` and `endstream`). The stream object itself is in line 14 - why 5 bytes, it's the string `abc` with a couple of line feeds at either side.
- Lines 17 - 19 contain a new *cross reference table* containing the absolute addresses of any new objects. The table has one object (number 73) this can be seen from line 18 - the object starts at byte 28818 in the PDF file (This can be seen in line 19 - byte 28818 is the absolute address of the "`7`" character at the start of line 9.
- Line 20 introduces a new trailer. The size of the object table is 74 (line 22) one more that the previous size of the object table (line 2); the root is `71 0 R` (line 23 is just a copy of line 3). The `Prev` field (line 24) is the old value of `startxref`, that is, 27227 found in line 7. The `Info` field (line 25) points to the new object.
- Line 28 is a pointer to the new cross reference table and is the byte offset in the PDF file to the start of line 17
- line 29 the field must end with `%%EOF`.

Get *any* of this wrong and the acrobat reader will refuse to read your file.