Extending ExT_EX

Simon Pepping

EuroT_EX 2001

24 September 2001

(ExTeX or NTS?)

Part I

A small extension of ExTeX: the pluggable file locator architecture

Reason:

Decouple NTS from TeX distribution

coupling established at run time

by a file locator implementation

Third parties (distribution makers)

write an implementation:

a subclass of `FileLocatorSpi`.

Function to be implemented:

```
InputStream openForReading
  (String name, String format,
  boolean mustExist)
```

User selects implementation
on the command line or in configuration
file:

```
java -Dnts.filelocatorconfig=\
  '<file locator implementation class> \
  <constructor arguments>'
```

(Alternative configuration

and invocation)

Implementations

1. `kpsewhich`

kpathsea is a C library,

use Java Native Interface

C functions:

```
void kpseInitialize(String progpath,
 String progname)
String kpseGetpath(String filename,
 String type, boolean mustExist)
String kpseInitializeGetpath(
 String progpath, String progname,
 String filename, String type,
 boolean mustExist)
```

Force a fake program name and path
on kpathsea (glibc)
(`program_invocation_name`
and
`program_invocation_short_name`):

```
progpath before:
  /usr/local/IBMJava2-13/jre/
  bin/exe/java
progname before: java
progpath after:
  /usr/local/share/TeXLive/
  bin/i386-linux/ntsfilelocator.cfg
progname after: latex
```

Use file locator configuration file
as a pseudo kpathsea program:

```
java -Dnts.fmt=latex \
 -Dnts.filelocatorconfig=\
 /usr/share/TeX/bin/ntsfilelocator.cfg \
 Nts latex-file
```

Here the config file

contains the following line:

```
nts.filelocator=<package>.kpsewhich
```

(Alternative configuration

and invocation)

Example of a run with kpsewhich

Implementations

2. TeX file server and client

Inspiration:

Java has good networking facilities

Simple proof-of-concept implementation

Protocol:

`Open TeX session` (handshake)

`Open TeX file`

TeX file server implementation

uses the same kpsewhich over JNI

implementation as above

Shortcoming:

kpathsea was written

for a single run.

It cannot reset the program name:

If the first run is for $\LaTeX$,

all runs are for $\LaTeX$.

Server is started as:

```
java ntssp.tex.TeXFileServer 1745 \
  /usr/local/share/TeXLive/bin/\
  i386-linux/ntsfilelocator.cfg
(port and pseudo kpathsea program)
```

TeX file server client

first searches the current directory

for local files,

then requests file from server.

Usage:

```
java -Dnts.filelocatorclass=\
 '<package>.TeXFSclient \
 hostname:port' \
 Nts tex-file
```

(Alternative configuration

and invocation)

Example of a run

with TeXFileServer and TeXFSClient

More robust implementation:
XML over HTTP, or SOAP,
perhaps in a web services framework.
More versatile communication protocol,
with path reporting and error reporting

Part II

A large extension of ExTeX:

Rendering XSL

Formatting Objects

A FOT (Formatting Object Tree)

is an abstract description

of the intended page layout

Implementing Formatting Objects:

- implementing start of FO,

- implementing end of FO,

- taking into account properties

  and position within ancestor FOs

Example of `jadetex` file and style

Example of `fotex.xmt`

Why not directly

from XSLT processor

to ExTeX?

Example of FOP (`program-fop.pdf`)

ExT$_E$X without macros:

No L^{A}T$_E$X, no NFSS,

no graphics packages,

default output routine,

no or default . . . .

No typesetting engine,

just a loose set of classes

Will we ever see it,

or will we leave the arena

to other implementations?

Part III

Other extensions of ExT$_E$X

Existing functionality,
now implemented in macro packages,
may be implemented natively in ExTeX,
available for all users

- Large packages:

  NFSS, `graphicx/s`, Babel, `ifthen`

- LaTeX's interface for documentclasses:

  `\@startsection`, counter commands

- LaTeX's interface for packages:

  `\newcommand`, `\usepackage`,

  option commands

This was:

Extending ExT$_E$X