Squashing the beast into a 60MB cage

Tor Lillqvist <tml@collabora.com>

tml, #libreoffice-dev, irc.freenode.net

Background: One single executable in an iOS app. No own shared libs

Repeat: all non-system code has to be in one executable

App Store rules: “iOS App binary files can be as large as 2 GB” “but”

“the executable file cannot exceed 60 MB”

LibreOffice from COLLABORA

We have one test iOS app: TiledLibreOffice

(which is a simple viewer for Writer docs)

At first, the TiledLibreOffice executable was ~90MB

(optimised build, no debug information or symbols in the file)

A third had to go without loss of functionality

Obviously there is a lot of code that gets linked in but never will get called at run-time

But we don't want to sprinkle ugly ifdefs all over the place if we don't have to

Only in as few key places as possible

Largest code reduction: ICU data

LibreOffice from COLLABORA

(“Internationalisation Components for Unicode”)

Normally, ICU data is present as constant data in code segment

When building ICU one has the option to use a data file instead

This data file needs to be memory-mapped in and passed to a single ICU call

Saving from ICU data: 23MB. Still lots to go

Locale data tables

Desktop LibreOffice includes data for all locales we know of

… but no need to do that in an iOS app

Introduce --with-locales configure-time option

Restricts what locales have data compiled in

Even better would be to use data files instead of constant data in code

… but that can be complicated

Our Japanese and Chinese “dictionaries” are large

LibreOffice from COLLABORA

Luckily simply structured, so can use memory-mapped data files instead

Use generated data files instead of generated code for OOXML custom shape presets

Split UNO components into smaller ones by refactoring factory methods

LibreOffice from COLLABORA

More aggressive ifdeffing-out of code irrelevant on mobile platforms

(for instance: to bypass code for desktop-style help, a11y features or extensions)

Charset/encoding conversion tables in sal: Optionally bin obscure ones

Tell compiler to optimise harder: -Oz

Unfortunately, somewhat fragile, compiler bugs?

Link-time optimisation? Not feasible: Linker grew to 40 GB in one hour before I lost patience

Non-issue: Unreferenced functions. Linker is smart, we use -dead_strip

Note: Don't make assumptions based on Linux experience

Apple's object file format, executable file format, and toolchain are different

How to find stuff to get rid of?

Inspect the linker map, workdir/TiledLibreOffice.map

Use the bin/ios-mapfile-statistics script

Oh, and after the squashing spree, the size of TiledLibreOffice was 43MB

Thanks to CloudOn for funding this work

FIN

Collabora

- Collabora Ltd.
  - Leading Open Source Consultancy
  - 8 years of experience. 90+ People.
- Collabora Productivity Ltd.
  - Dedicated to Enterprise LibreOffice
  - Provides Level-3 support (code issues) to all Novell / SUSE LibreOffice clients
  - Architects of Microsoft OpenXML filters