Porting OpenOffice.org to AMD64 Architecture

Jan Holesovsky
SUSE Labs

<kendy@suse.cz>

October 23, 2005

Overview

- Brief introduction
- What had to (and has to) be done
  - making it compile
  - implementing bridges
  - debugging
- How to get it up-stream
- The present and the future
  - where are we now
  - good developer habits
  - demo

The AMD64 Architecture

x86 and x86-64

- What is AMD64?
  - long mode
- Why to port OOo there?
  - dependencies in the distribution, etc.

- What is the difference for the programmer?
  - longs and pointers are 64bit
  - more registers
  - different calling conventions

Making it Compile and Run

Compilation

- We are talking about more than 5 million lines of code!
- And some of the code is really, really old
  - we have to create the infrastructure
  - define the suitable fixed-size types (sal_Int64, sal_IntPtr)
  - use them on the failing places

Usual Bugs

- Function is declared with `sal_Int32`, but defined with `long`
- Function is declared with `void foo( sal_Int32 * pInt)`, but used `long nInt; foo( &nInt );`

-
  ```
  class A { virtual sal_Int32 bar(); };
  class B : public A
          { virtual long bar(); };
  ```
- `void *pPtr; sal_Int32 = (sal_Int32) pPtr;`
- `sal_Int32 nInt = LONG_MAX;`

UNO (Universal Network Objects)

- OOo component architecture
- Allows interoperability between different programming languages, different object models, etc.
- Bindings to C++, Java, Python
- The interoperability is solved thanks to 'bridges'

C++ <-> UNO Bridge

- One for every supported C++ compiler and architecture
  - AMD64 ABI
- C++ -> UNO
  - we have to create table of virtual methods
  - only trampolines there – so that we can have one function handling all
  - return values have to be converted back
- UNO -> C++
  - we have to fill the registers and stack before the call
  - perform the call
  - convert the return values

Debugging - What Can One Expect

- Fortunately nothing like “I opened the file, but saw just every second letter.” ;-)
- But there were/are tricky ones
  - unusable menus thanks to wrong type when calling an X call for the screen size
  - able to open MS documents, but not OpenDocument!
- And of course a lot of crashes

Getting the Patches Up-stream

Child Workspace (CWS)

- CVS branch + additional info in EIS (Environment Information System)
- ooo64bit01
  - already integrated
  - basic support for AMD64, like types, building infrastructure
  - 1st implementation of the UNO<->C++ bridge, but unusable :-(
- ooo64bit02
  - opened more than a year ago
  - too many fixes => it cannot get through QA in a reasonable amount of time
  - breaks 32bit

Getting It There

- Avoid ooo64bit02 for the new patches!
  - committing the fixes to 'normal' CWSes whenever possible
  - letting it in ooo-build
- Split ooo64bit02 into smaller CWSes
  - each solves one particular problem => easier to do QA on this
  - time-consuming (split, commit to CWS, do a QA build, fix potential bugs, etc.)
  - but the chance that the fix finds its way to OOo increases

The Present and the Future

With All the Patches

- You can run the 64bit OOo, write, do basic operations
- Read MS documents (but not OpenDocument)
- The bridges testsuite is still failing
- Crashes often
- In short: not stable enough to run this presentation with it :-(

Good Developer Habits

- Think! :-)
  - sizeof(long) != sizeof(sal_Int32), sizeof(void *) != sizeof(sal_Int32)
  - avoid the bugs described at the beginning of the presentation
- long/ULONG is not evil
  - you can use it, but **consistently**
  - avoid it in the I/O
    - fixed size types
    - correct endianity (network byte order)
- casting pointers to an integer type is evil
  - but probably inevitable in some cases
  - sal_IntPtr – introduced in 'intptr' CWS

Demo & Questions...

Thanks for your attention!

More info:

- http://artax.karlin.mff.cuni.cz/~kendy/blog
- http://blog.janik.cz
- http://www.go-oo.org

Novell®