ICU Update

Markus Scherer (Google)
Peter Edberg (Apple)
Stuart Gill (Google)

Agenda

- Isn't Unicode enough?
- Why ICU?
- Where is ICU?
- What's new in ICU 4.6 & 4.8?
- What's next for ICU?

Isn't Unicode enough?

- The nature of Unicode
- Internationalization, Localization & Locales

The Nature of Unicode

- Handles all modern world languages
- Efficient and effective processing
- Lossless data exchange
- Enables single-binary global software

But...

- 1,400 pages + Annexes + additional standards
- Nearly 110,000 characters
- Major update every 3 years, minor update about once a year
- 80+ character properties, many multi-valued
- Affects many processes: display, line-break, regular expressions...

Internationalization, Localization & Locales

Requirements vary widely across languages & countries

- Sorting
- Text searching
- Bidirectional text processing and complex text layout
- Date/time/number/currency formatting
- Codepage conversion
- …and so on

Performance is key

- It might be easy to do the right thing
- It is hard to do it fast

Why ICU?

- ICU features
- ICU works everywhere
- ICU is kept up to date

ICU Features

- Unicode text handling
- Charset conversions (175+)
- Charset detection
- Collation & Searching
- Locales from CLDR (530+)
- Resource Bundles
- Calendar & Time zones
- Complex-text layout engine
- Unicode Regular Expressions
- Breaks: word, line, ...
- Formatting
  - Date & time
  - Durations
  - Messages
  - Numbers & currencies
  - Plurals
- Transforms
  - Normalization
  - Casing
  - Transliterations

ICU Works Everywhere

Mature, widely used set of C/C++ and Java libraries

- Basis for Java 1.1 internationalization, but goes far beyond Java 1.1

Very portable – identical results on all platforms/programming languages

- C/C++ (ICU4C): 30+ platforms/compilers
- Java (ICU4J): Oracle and IBM JRE

Full threading model

Customizable & Modular

Open source (since 1999) – but non-restrictive

- Governed by a Project Management Committee
- Contributions from many parties (IBM, Google, Apple, Yahoo, ...)

ICU Is Kept Up To Date

- 1..2 major ICU releases per year
- Each ICU release supports the latest
  - Unicode version
  - CLDR version
  - Time zone database update
- TZ DB updates for past ICU versions
- Maintenance releases for important bugs

Where is ICU?

- ICU in IBM, Google and Apple products
- Other ICU Users

ICU in IBM Products

- All 5 major software brands
- IBM operating systems
- Products

  Ascential Software, Cognos, PSD Print Architecture, DB2, COBOL, Host Access Client, InfoPrint Manager, Informix GLS, iSeries, Language Analysis Systems, Lotus Notes, Lotus Extended Search, Lotus Workplace, WebSphere Message Broker, NUMA-Q, OTI, OmniFind, Pervasive Computing WECMS, Rational Business Developer and Rational Application Developer, SS&S Websphere Banking Solutions, Tivoli Presentation Services, Tivoli Identity Manager, WBI Adapter/ Connect/Modeler and Monitor/ Solution Technology Development/WBI-Financial TePI, Websphere Application Server/ Studio Workload Simulator/Transcoding Publisher, XML Parser.

ICU in Google Products

- Web Search
- Chrome
- Android
- Adwords
- Google Finance
- Google Maps
- Blogger
- Google Analytics
- Google Gears
- Google Groups
- Others...

ICU in Apple Products

- Mac OS X, including applications
- iOS (iPhone, iPad, iPod touch, Apple TV)
- Windows applications and related support
  - Safari
  - iTunes
  - Apple Mobile Device Support
- Others...

Other ICU Users

ABAS Software, Adobe, Amazon (Kindle), Amdocs, Apache (Harmony, Lucene, Solr, PDFBox, Tika, Xlan, Xerces, ...), Appian, Argonne National Laboratory, Avaya, BAE Systems Geospatial eXploitation Products, BEA, BluePhoenix Solutions, BMC Software, Boost, BroadJump, Business Objects, caris, CERN, Debian Linux, Dell, Eclipse, eBay, EMC Corporation, ESRI, Free BSD, Gentoo Linux, GroundWork Open Source, GTK+, Harman/Becker Automotive Systems GmbH, HP, Hyperion, Inktomi, Innodata Isogen, Informatica, Intel, Interlogics, IONA, IXOS, Jikes, Library of Congress, Mathworks, Mozilla, Netezza, OpenOffice, Lawson Software, Leica Geosystems GIS & Mapping LLC, Mandrake Linux, OCLC, Progress Software, Python, QNX, Rogue Wave, SAP, SIL, SPSS, Software AG, Sun Microsystems (Solaris, Java), SuSE, Sybase, Symantec, Teradata (NCR), Trend Micro, Virage, webMethods, Wine, WMS Gaming, XyEnterprise, Yahoo!, and many others.

What's new in ICU 4.6 & 4.8?

- Unicode 6.0 & CLDR 1.9/2.0
- Java 7 Locale support
- Collation optimizations
- Collation script reordering
- AlphabeticIndex
- UTS #46 (updated IDNA)
- Regular Expression improvements
- MessageFormat improvements
- Other formatting improvements
- C++ RTTI

Locale Data (1.9)

- CLDR 1.9 in ICU 4.6
  - Collation: CJK based on Unihan, reordered DUCET, search collators
  - Improved transliteration, etc.
  - Punctuation exemplar characters

Locale Data (2.0)

- CLDR 2.0 in ICU 4.8
  - 20% more data overall, most of the increase in top 55 languages
  - Explicit parent locales, etc.
  - Relative times, with plural support

Java 7 Locale Support

- Java 7 Locale supports BCP 47 & Unicode locales
  - Full internet language tags
  - -u- Unicode locale extensions
  - With major contribution by ICU team members
- ICU ULocale adds parallel APIs
- ULocale on Java 7 calls new APIs
  - Fallback code for Java 5 & 6

Collation Optimizations

- Collation weights
  - Unicode 6.0 would have overflowed
  - Redistributed: Shorter weights for common scripts, room for CJK
- Default order
  - Separate ranges for spaces/punctuation/symbols/digits
- Smaller data
  - More compact rules, import

Collation Script Reordering

- Allows for entire script blocks to be reordered
  - Scripts: Han, Greek, Latin, ...
  - Special Groups: Symbols, digits, ...
- Available via API & custom data
  - CLDR/ICU data not yet changed

Reordering Examples

- Sort Greek before all other scripts
  - `[reorder Grek]`
- Sort Romaji/Kana/Kanji first before other scripts
  - `[reorder Latn Kana Hani]`
- Sort digits after all scripts
  - `[reorder Zzzz digit]`

AlphabeticIndex

- Index: A B C ... 一 丁 万 ...
  - Sorts strings into index buckets
- Matches collation for the language
- Chinese: Pinyin, stroke, radical/stroke

UTS #46

- Revamped IDNA API supports UTS #46
- Compatible upgrade from IDNA2003 to IDNA2008
  - Unicode standard mapping
    - Based on NFKC_Casefold
  - Optional transitional mappings (ß ς J NJ)
    - fußball.de → fussball.de
- Improved ICU performance

Regular Expressions (C)

- UText API finalized
  - Supports 64-bit offsets & indexes
- Find-progress callback
- Set start position independently of match region

MessageFormat

- Reimplemented Message/Choice/Plural/SelectFormat
  - Common parser, consistent behavior
    - Single pass parses full message
  - ASCII apostrophe mostly just text
- Extended Plurals
  - Explicit-value variants
    - =0 (“no file”), =1 (“one file”), …
  - List+: “A, B and 3 others” (offset:2)

ASCII Apostrophe

- Java: `"I don''t see {0,choice,0#O''''Brien in {1}}"`
  - Double or quad apostrophe in nested message depending on other contents
- ICU 4.8: `"I don't see {0,select,other{O'Brien in {1}}}"`
  - ' only starts quoting literal text if
  - preceding { or } or # (in plural) or | (in choice)
  - '' still works for single ' output

Extended Plurals

- "{num_people,plural, offset:1
  =0{No one went}
  =1{{person1} went}
  =2{{person1} and {person2} went}
  other{{person1} and # others went}}."
-
- "#" = remainder = num_people - offset

Formatting Improvements

- Alternate number symbols
  - Based on numbering system
  - For date formatting, etc.
- Improved parsing of numbers and dates/times
  - More consistent between C and J
  - Adds lenient number parsing for C

C++ RTTI

- Compiler RTTI used & required
  - dynamic_cast<pointer>
  - typeid(*this) == typeid(that)
- No new “poor man's RTTI”
  - Did not support is-subtype-of
  - No getDynamicClassID() in new class hierarchies

What's next for ICU?

- Version Numbers
- Release Schedule
- ICU 49 Release Summary

Version Numbers

- … 4.4 → 4.6 → 4.8 → 49 → 50 → 51 …
- Users confused by old scheme
- Combine first two fields into one
- Pre-release milestones used “odd minor” versions, now xx.0
- CLDR: … 1.9 → 2.0 → 21 → 22 …

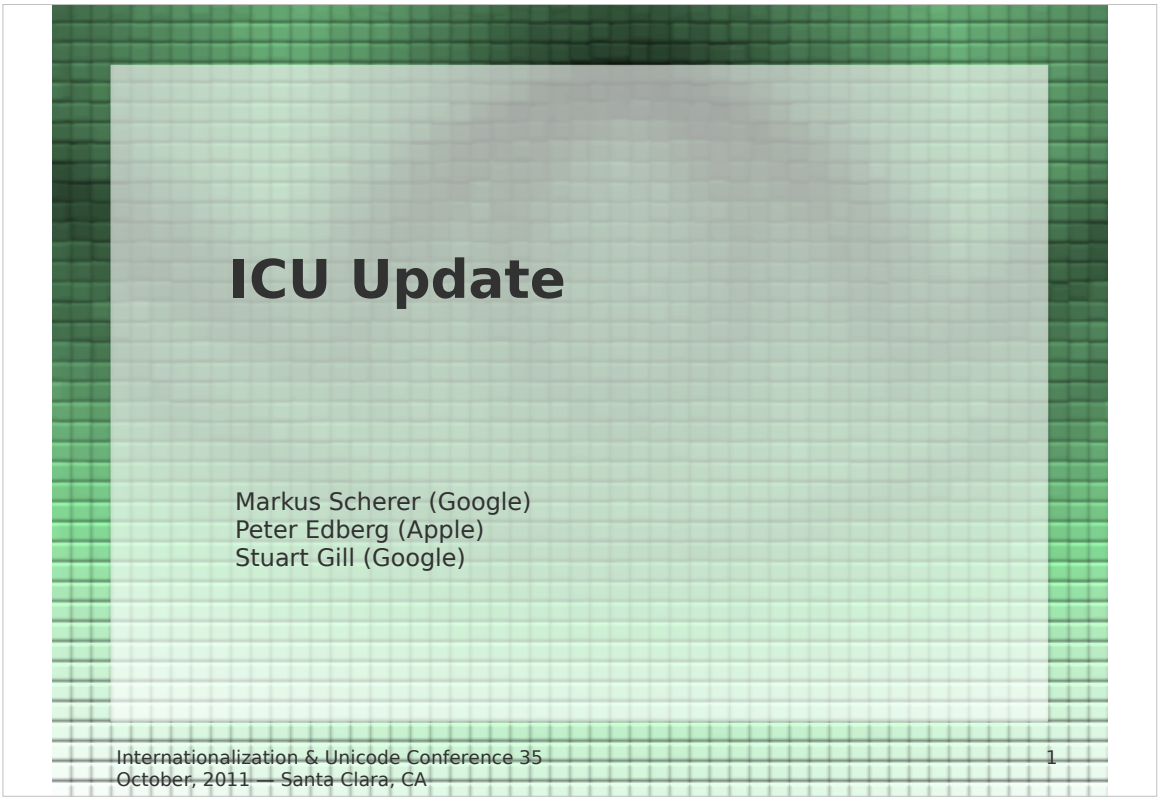
ICU 49 (Preliminary)

- ETA: End of March 2012
- Unicode 6.1 & CLDR 21
- C++: Simpler build & cross-compiling
  - Fixed platform.h (not autoconf'ed)
  - Improved U_HIDE_DRAFT_API
  - Requires namespace support
  - Internal dependencies cleaned up

References

ICU Main Site: http://icu-project.org

- Download ICU Releases
- User Guide
- Demonstrations
- Technical FAQ
- Bug Report
- Mailing Lists (design & support)



ICU (International Components for Unicode) is an open source development project sponsored, supported, and used by many organizations. It is dedicated to providing robust, full-featured, commercial quality, freely available Unicode-based technologies.

Comprehensive support for the Unicode Standard is the basis for multilingual, single-binary software. ICU uses the most current versions of the standard, and provides full support for supplementary characters.

As computing environments become more heterogeneous, software portability becomes more important. ICU lets you produce the same results across all the various platforms you support. It offers great flexibility to extend and customize the supplied system services.

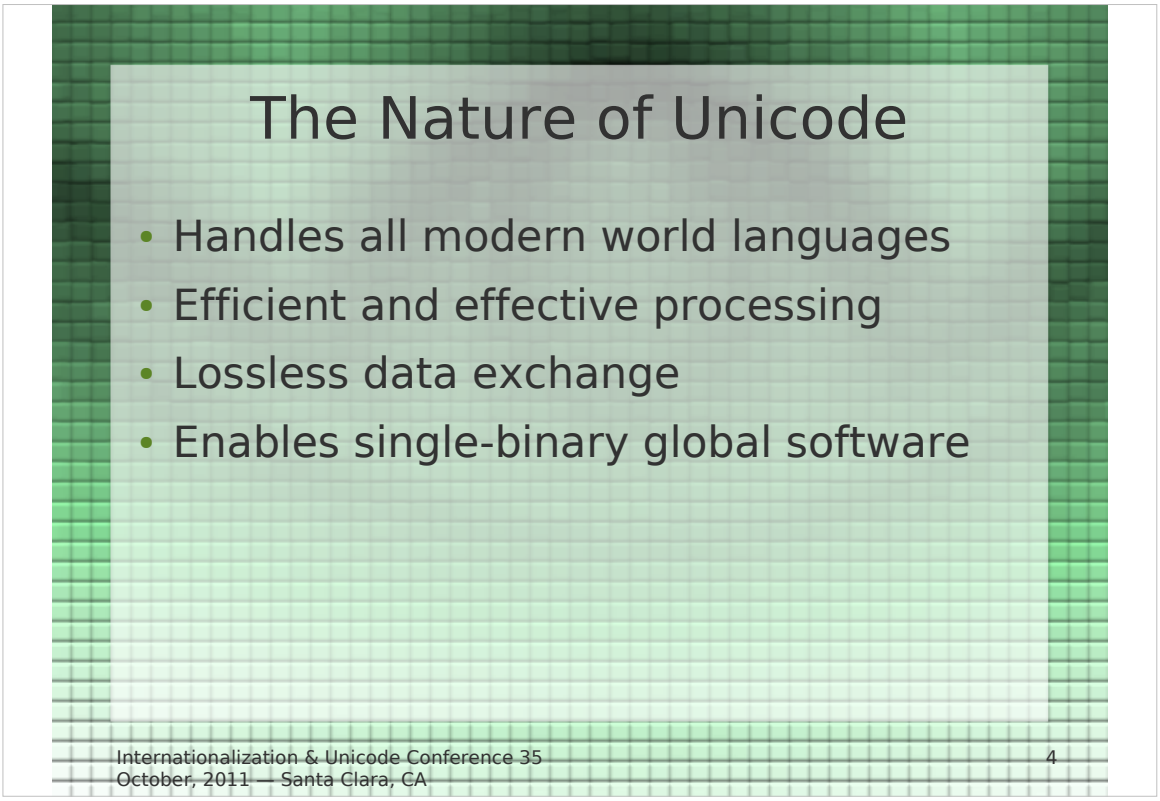
For more information, see the ICU website: http://icu-project.org

Agenda

- Isn't Unicode enough?
- Why ICU?
- Where is ICU?
- What's new in ICU 4.6 & 4.8?
- What's next for ICU?

Isn't Unicode enough?

- The nature of Unicode
- Internationalization, Localization & Locales



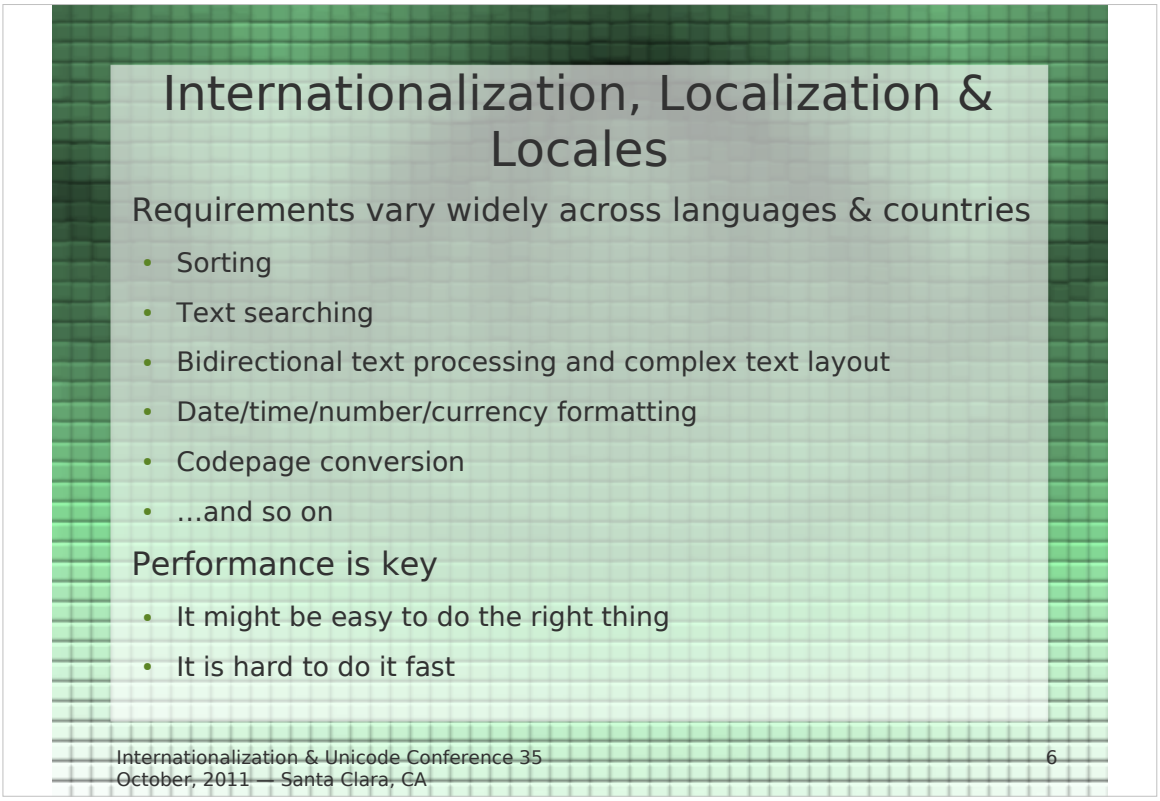
Unicode (and the parallel ISO 10646 standard) defines the character set necessary for efficiently processing text in any language and for maintaining text data integrity. In addition to global character coverage, the Unicode standard is unique among character set standards because it also defines data and algorithms for efficient and consistent text processing. This simplifies high-level processing and ensures that all conformant software produces the same results. The widespread adoption of Unicode over the last decade made text data truly portable and formed a cornerstone of the Internet.

Unicode enables lossless exchange of multilingual data between different types of computing systems, as well as single-binary installations of software which can handle text in all languages.

As a result, the Unicode Standard is complex and voluminous and not trivial to implement. ICU supports all Unicode characters, is regularly updated to the latest Unicode version, and implements and provides most of the properties and algorithms.

But...

- 1,400 pages + Annexes + additional standards
- Nearly 110,000 characters
- Major update every 3 years, minor update about once a year
- 80+ character properties, many multi-valued
- Affects many processes: display, line-break, regular expressions...



The design and architecture of software that can work with multiple languages and cultural specializations is called internationalization. It involves taking into account a variety of attributes for many areas of text handling and data input and output. The most common attributes are the written language and the country or region for which data is processed or presented. Standard codes for these attributes, and sometimes others, are often combined into "locale identifiers". Depending on the context, the term "locale" refers either to such locale identifiers or to the relevant collection of associated data and behaviors.

In addition to these familiar attributes, others are also important and cannot be reliably inferred. For example, currency codes and codepages need to be identified reliably for correct results.

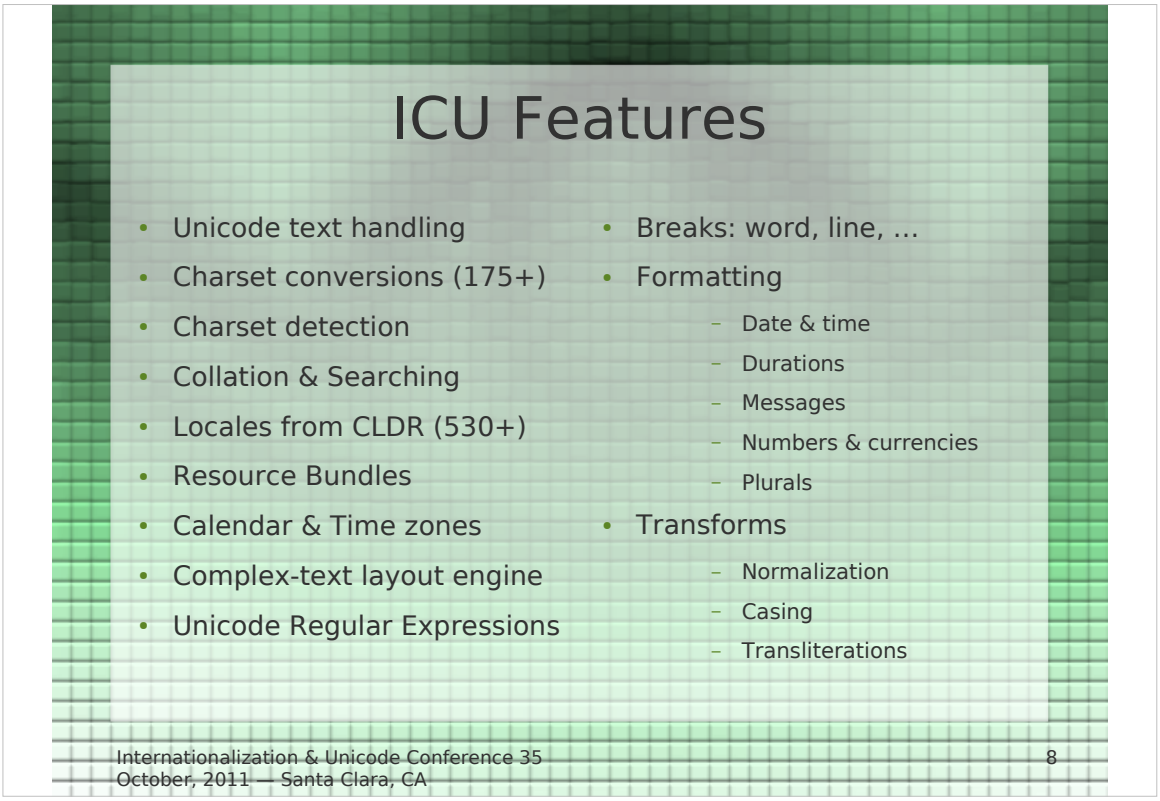
Localization provides internationalized software with locale-specific User Interface elements (text, images, layout) and sometimes functionality for regional business rules or similar.

The term globalization is sometimes used as a synonym for internationalization. We use it more narrowly, for software which can be compiled and installed once and handles text in all languages at the same time, as opposed to requiring recompilation for each locale. Such software needs to use Unicode for text processing.

It is often relatively easy to satisfy the requirements from one language or culture, or a small number of closely related ones. However, with the diversity of requirements from many languages and cultures on many processes, and the desire for high performance in many cases, the implementation of these processes can become rather complex. The ICU libraries provide "shrink-wrapped", reusable, tested implementations that were designed with performance in mind.

Why ICU?

- ICU features
- ICU works everywhere
- ICU is kept up to date

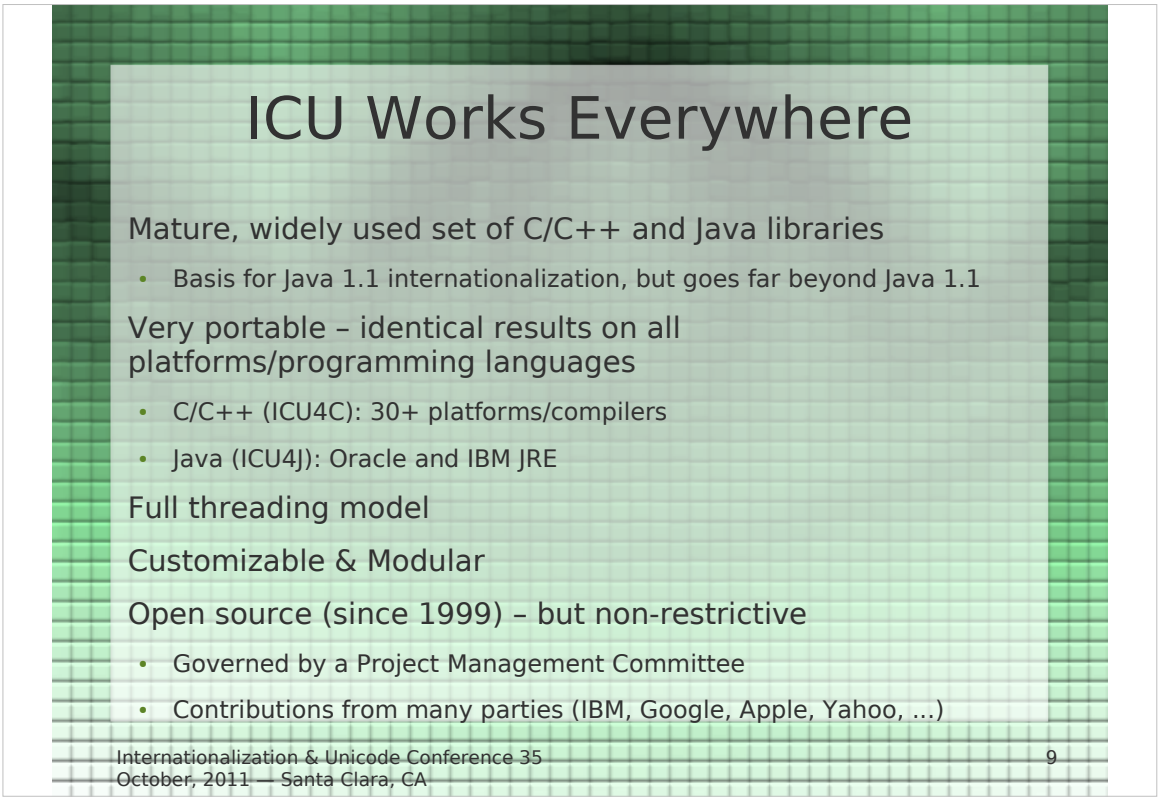


In addition to basic Unicode standard conformance, both the ICU Java library ("ICU4J") and the C/C++ libraries ("ICU4C") also provide a full set of internationalization features listed above.

Notes on C/C++ vs. Java

ICU C/C++ and Java APIs do differ slightly due to the differences of programming languages. Sometimes the feature development in ICU4C leapfrogs ICU4J or vice versa by 1-2 releases.

Since ICU is open source and closely tracks the Unicode Standard, ICU can support changes and additions to the Unicode Standard much more quickly than Java. Java support for Unicode is tied to major releases of the JDK, and can lag the Unicode Standard by a year or more.



International Components for Unicode (ICU) is a mature set of widely used C/C++ and Java libraries. They are portable to many environments and platforms. There are 2 sub-projects of ICU. There is ICU4C, which is written in C and C++. There is ICU4J which is written in Java.

Mature: Started in 1996, contributions to JDK 1.1 (1997), open source project since 1999.

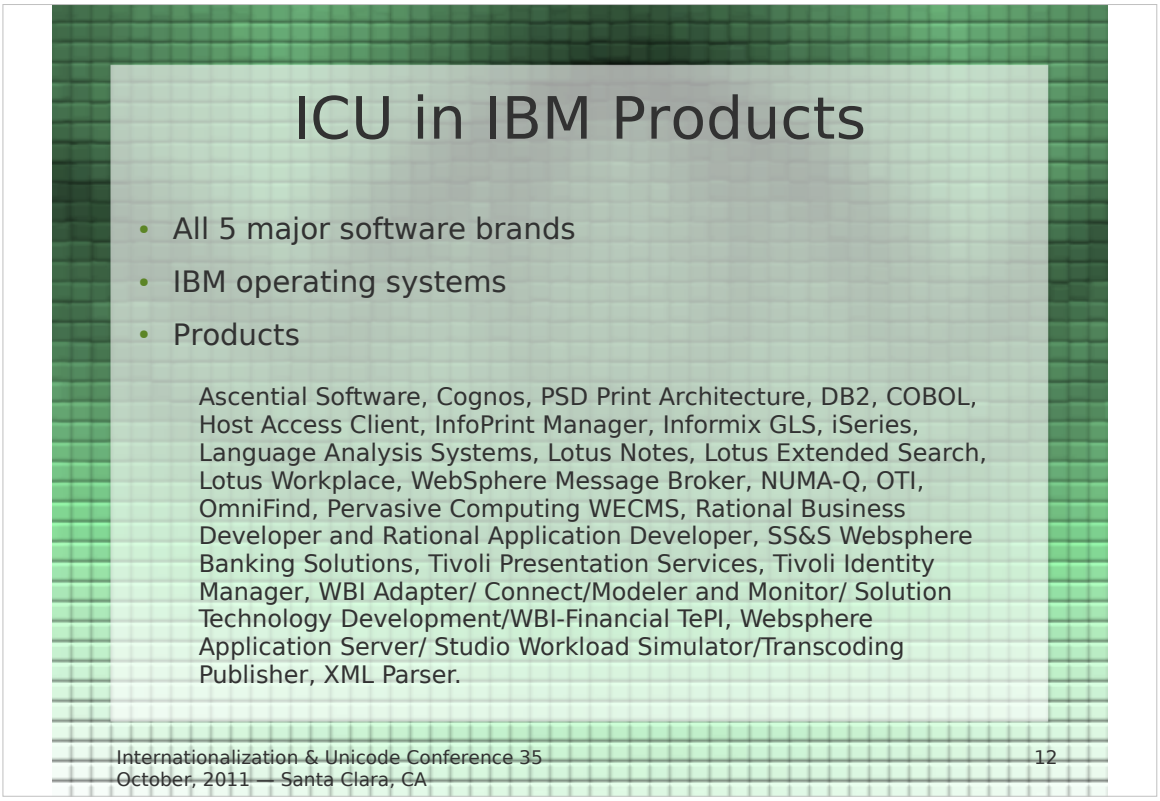
ICU is distributed under the X license. The license allows ICU to be incorporated into a wide variety of software projects using the GPL license, while also allowing ICU to be incorporated into non-open source products. You can read the license on ICU's web site for details.

ICU Is Kept Up To Date

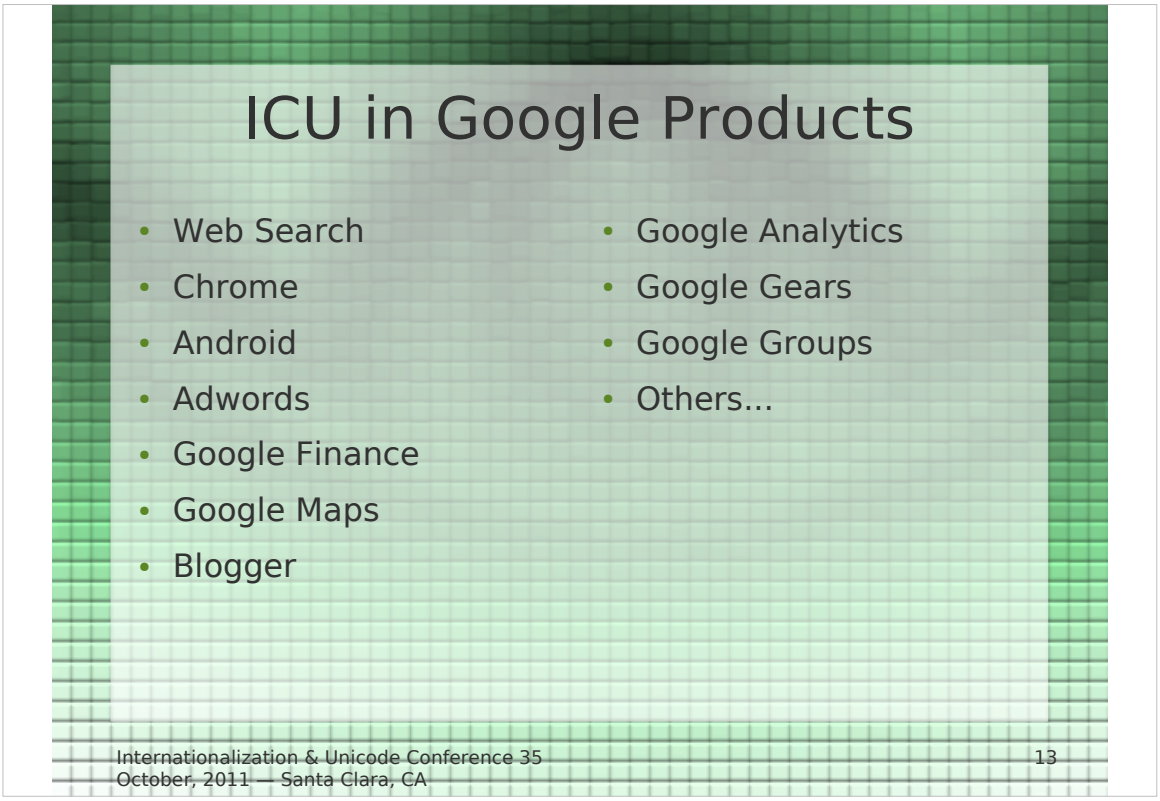
- 1..2 major ICU releases per year
- Each ICU release supports the latest
  - Unicode version
  - CLDR version
  - Time zone database update
- TZ DB updates for past ICU versions
- Maintenance releases for important bugs

Where is ICU?

- ICU in IBM, Google and Apple products
- Other ICU Users



ICU is used throughout IBM. It is also used by many other companies and organizations. Many of these companies and organizations also participate in improving ICU.



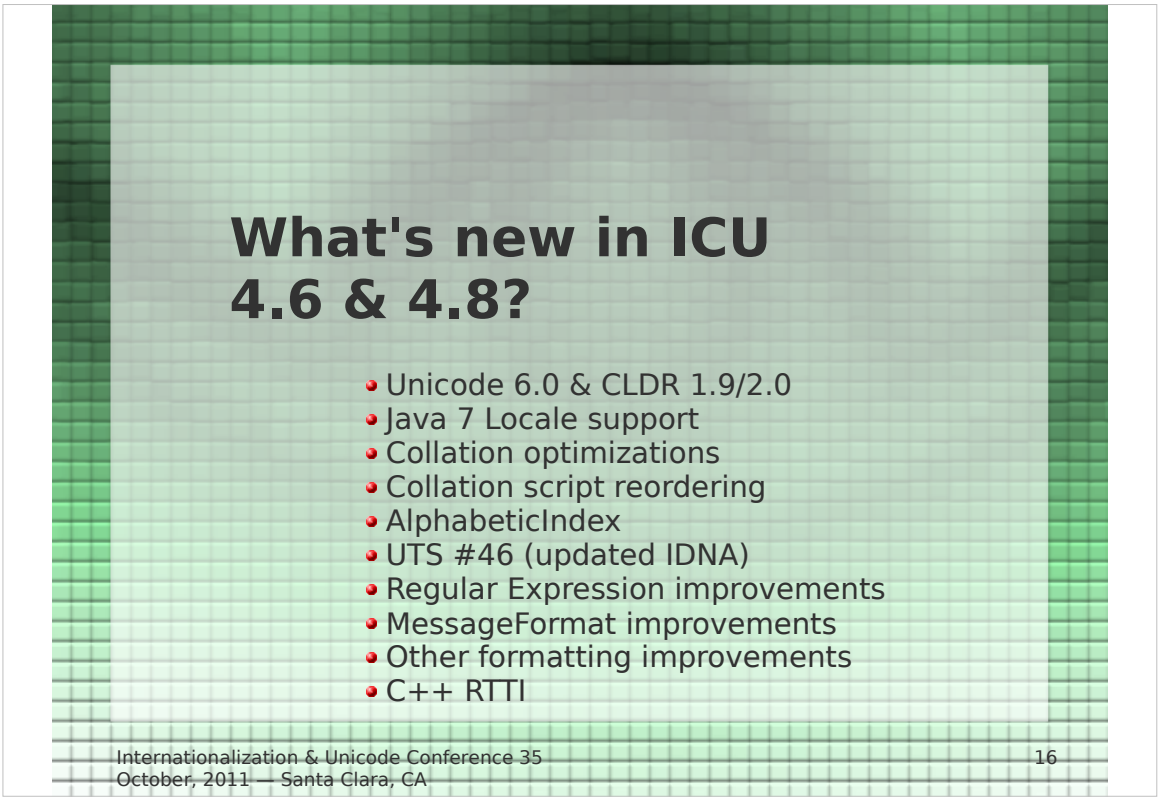
ICU is part of the Android platform and used in the implementation of some of the SDK libraries. It is the basis of Chrome internationalization and used widely in cloud-based Google products.

ICU in Apple Products

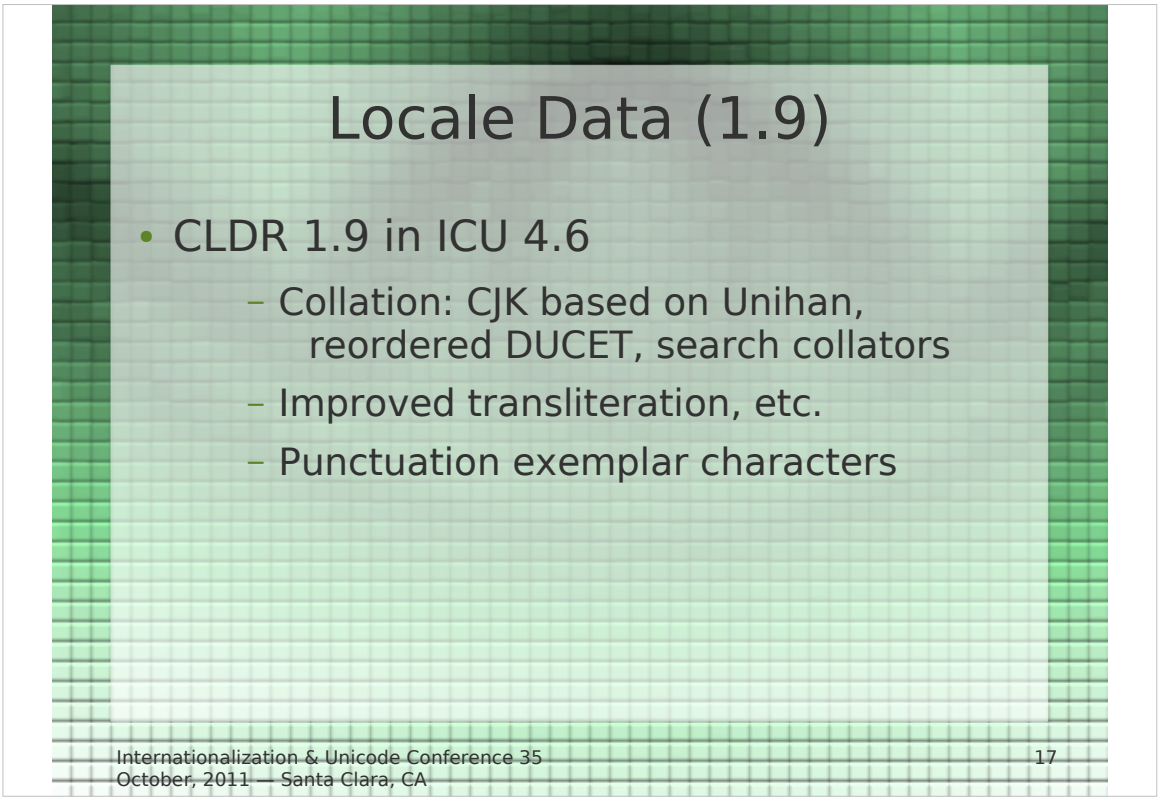
- Mac OS X, including applications
- iOS (iPhone, iPad, iPod touch, Apple TV)
- Windows applications and related support
  - Safari
  - iTunes
  - Apple Mobile Device Support
- Others...

Other ICU Users

ABAS Software, Adobe, Amazon (Kindle), Amdocs, Apache (Harmony, Lucene, Solr, PDFBox, Tika, Xlan, Xerces, ...), Appian, Argonne National Laboratory, Avaya, BAE Systems Geospatial eXploitation Products, BEA, BluePhoenix Solutions, BMC Software, Boost, BroadJump, Business Objects, caris, CERN, Debian Linux, Dell, Eclipse, eBay, EMC Corporation, ESRI, Free BSD, Gentoo Linux, GroundWork Open Source, GTK+, Harman/Becker Automotive Systems GmbH, HP, Hyperion, Inktomi, Innodata Isogen, Informatica, Intel, Interlogics, IONA, IXOS, Jikes, Library of Congress, Mathworks, Mozilla, Netezza, OpenOffice, Lawson Software, Leica Geosystems GIS & Mapping LLC, Mandrake Linux, OCLC, Progress Software, Python, QNX, Rogue Wave, SAP, SIL, SPSS, Software AG, Sun Microsystems (Solaris, Java), SuSE, Sybase, Symantec, Teradata (NCR), Trend Micro, Virage, webMethods, Wine, WMS Gaming, XyEnterprise, Yahoo!, and many others.

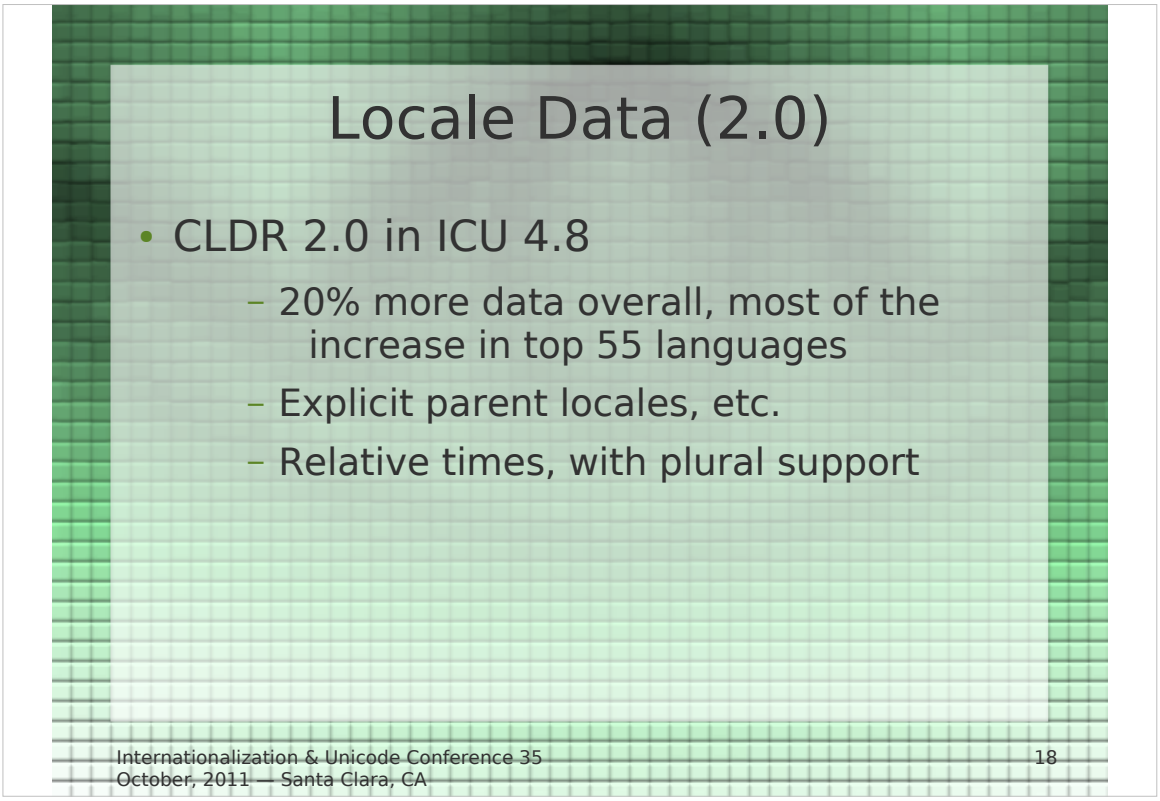


Within the last year, the ICU team has released two versions. ICU 4.6 upgraded to Unicode 6.0, and ICU 4.8 and CLDR 2.0 increased locale data significantly.

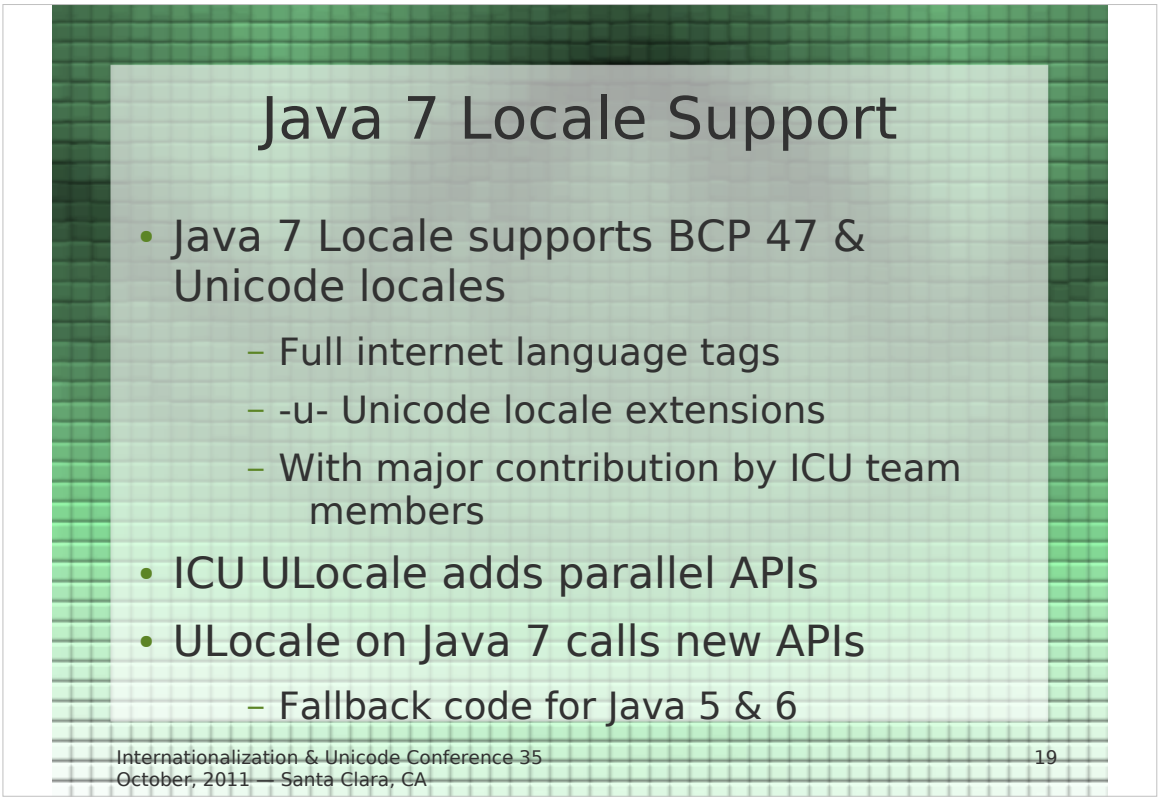


Every ICU release implements the current Unicode version, collation and CLDR locale data.
CJK collation was redone, based on updated Unihan data.
The default collation order was changed (see later slide).
Special tailorings for string search were added.
Pinyin transliteration data based on Unihan data.
Transliteration data for more languages.
A new exemplar character set type provides the punctuation characters customarily used with each language.

Details: http://cldr.unicode.org/index/downloads

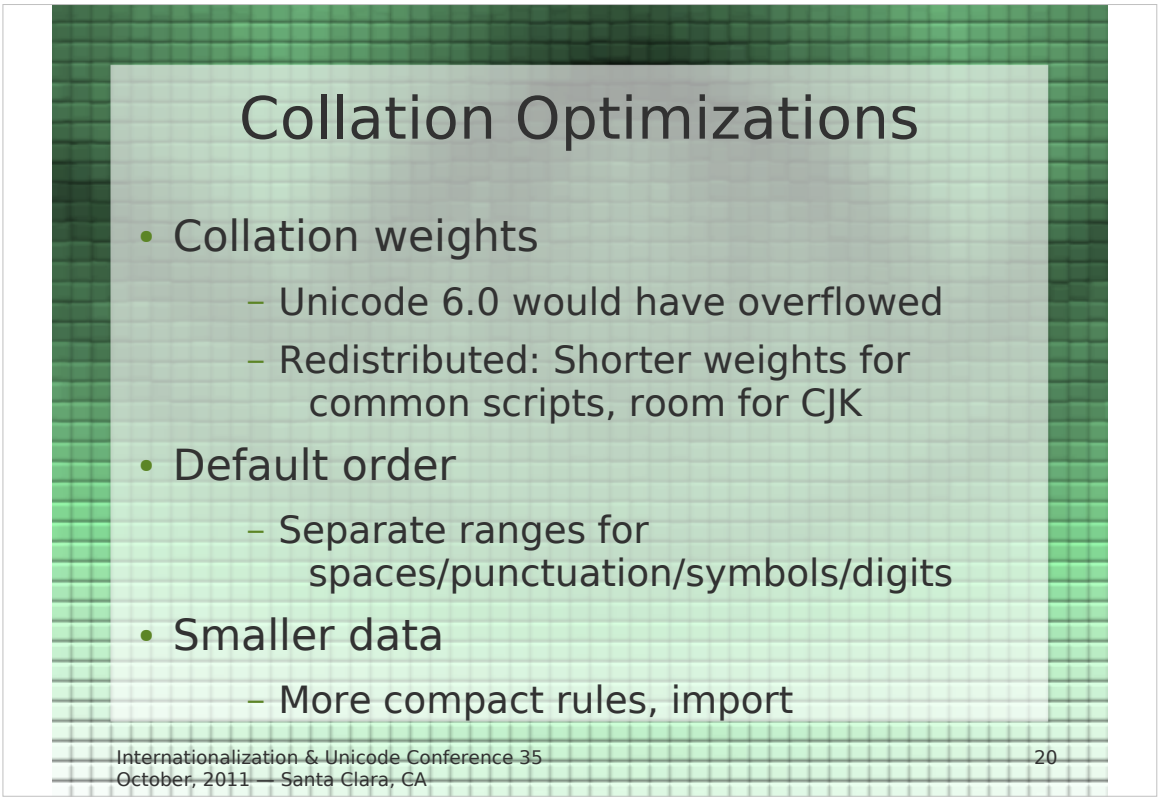


Data for top 55 languages increased by >45%.
Explicit parent locales where truncation parents are suboptimal: zh_Hant→root, sr_Latn→root, es_AR->es-419, pt_AO→pt_PT, …
Data for relative times such as “3 days ago”



Java 7 Locale: forLanguageTag(), toLanguageTag(), and methods for -u- Unicode locale extensions (http://openjdk.java.net/projects/jdk7/features/)

ICU ULocale has parallel API; conversion to/from Java Locale calls new Locale API if available (via reflection)



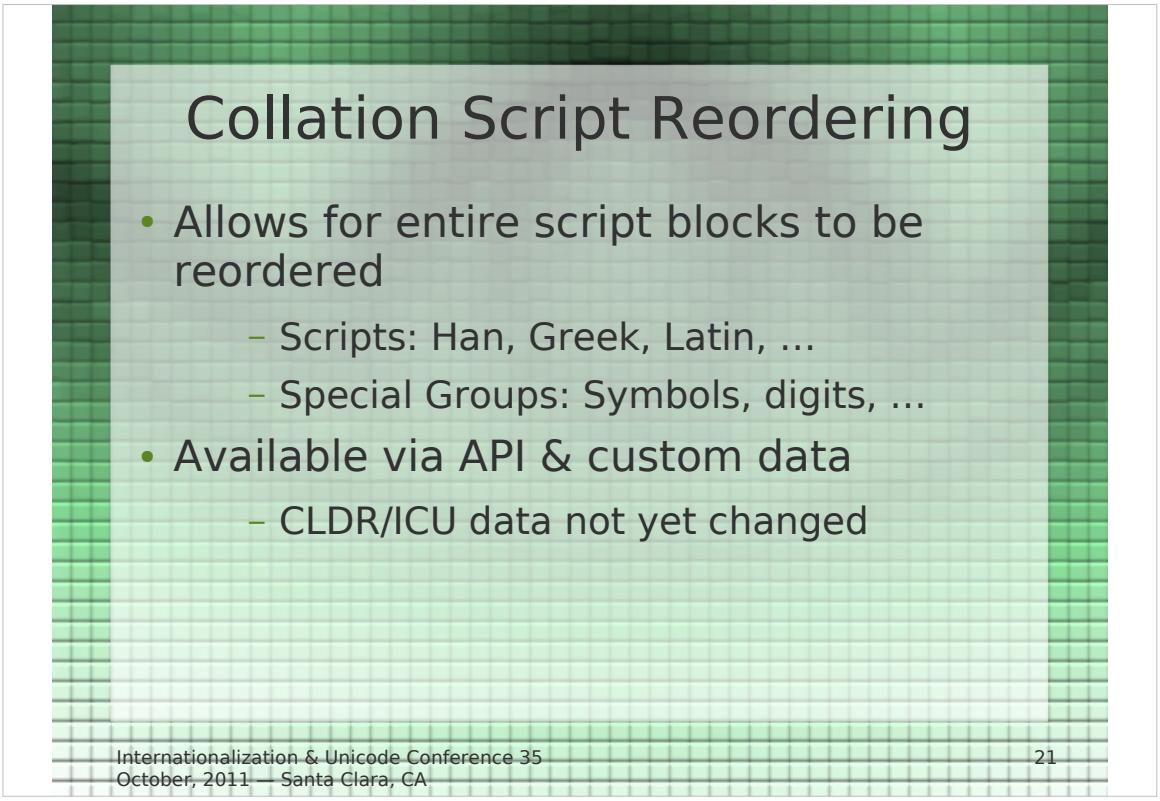
Repertoire outgrew available space for 2-byte primary weights. New weights use 3-byte primaries for rare/ancient scripts, re-expand large range for CJK tailorings. Lead bytes optimized for script reordering.

Default order:

- Disjoint categories (spaces, punctuation, symbols, currency symbols, numbers, letters).
- Symbols no longer “variable” by default.
- Special characters for minimum (U+FFFE) and maximum (U+FFFF) weights.

Additional rule syntax for ranges of consecutive code points and sharing sub-rules.

Java: The Collator is now Freezable, which makes it thread-safe.



New API and syntax for reordering scripts (Latin, Greek, Han, …) and sub-categories of “common” characters (symbols, digits, …)

Allows sorting native script first for each language

Allows sorting digits after letters (as in German standard) etc.

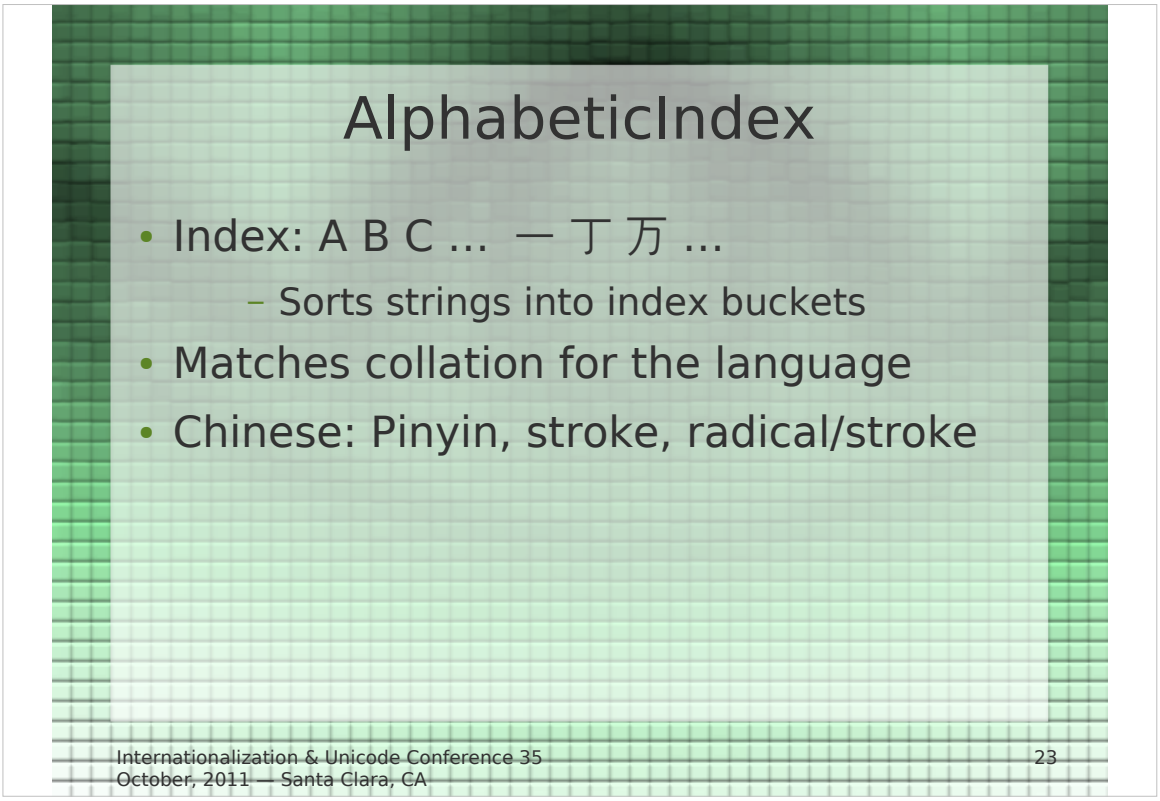
Able to mix “before” and “after” rules as well as specific sorting of specific scripts

All scripts sharing a leading collation byte move together — e.g. Deseret, Shavian, Yii, Bopomofo

Default collation data does not yet include such reorderings

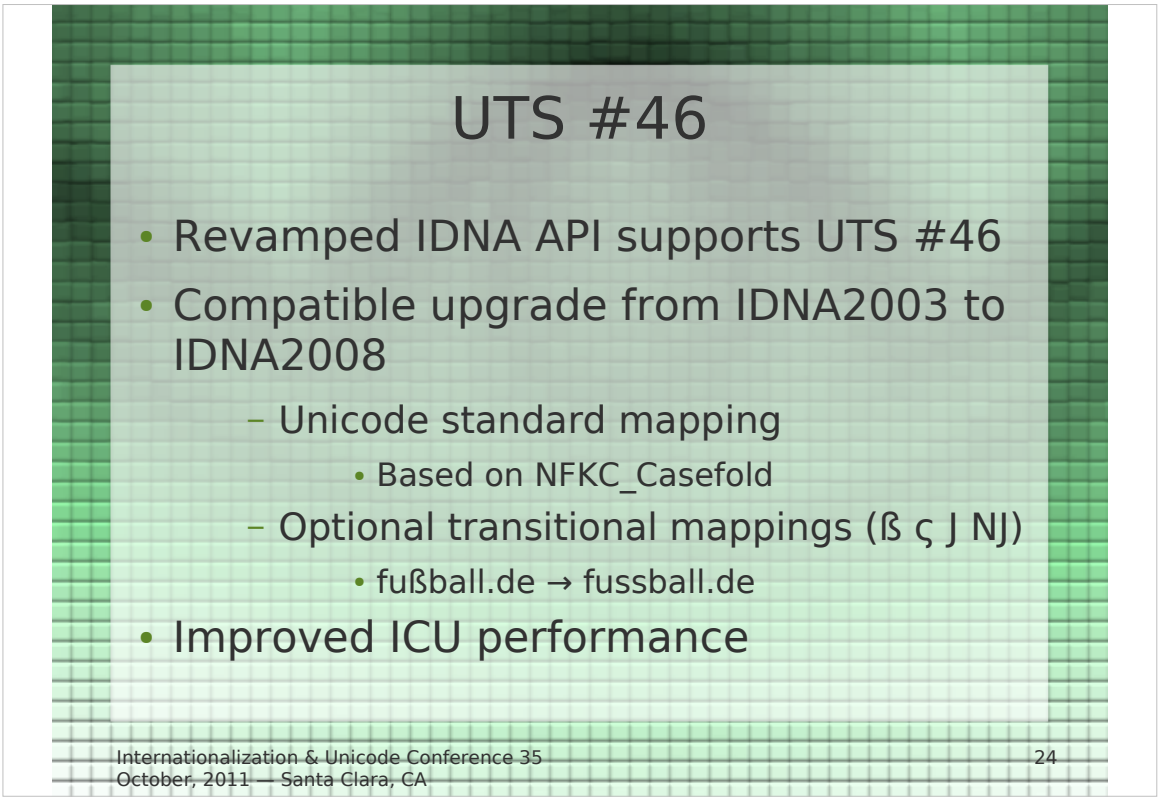
Reordering Examples

- Sort Greek before all other scripts
  - `[reorder Grek]`
- Sort Romaji/Kana/Kanji first before other scripts
  - `[reorder Latn Kana Hani]`
- Sort digits after all scripts
  - `[reorder Zzzz digit]`



The AlphabeticIndex provides index characters for lists of names, and sorts name strings into the resulting index “buckets”. Index characters and sort order are based on the language's collation tailoring.

For Chinese, special CLDR data supports custom index characters for the Pinyin, stroke and radical/stroke sort orders.

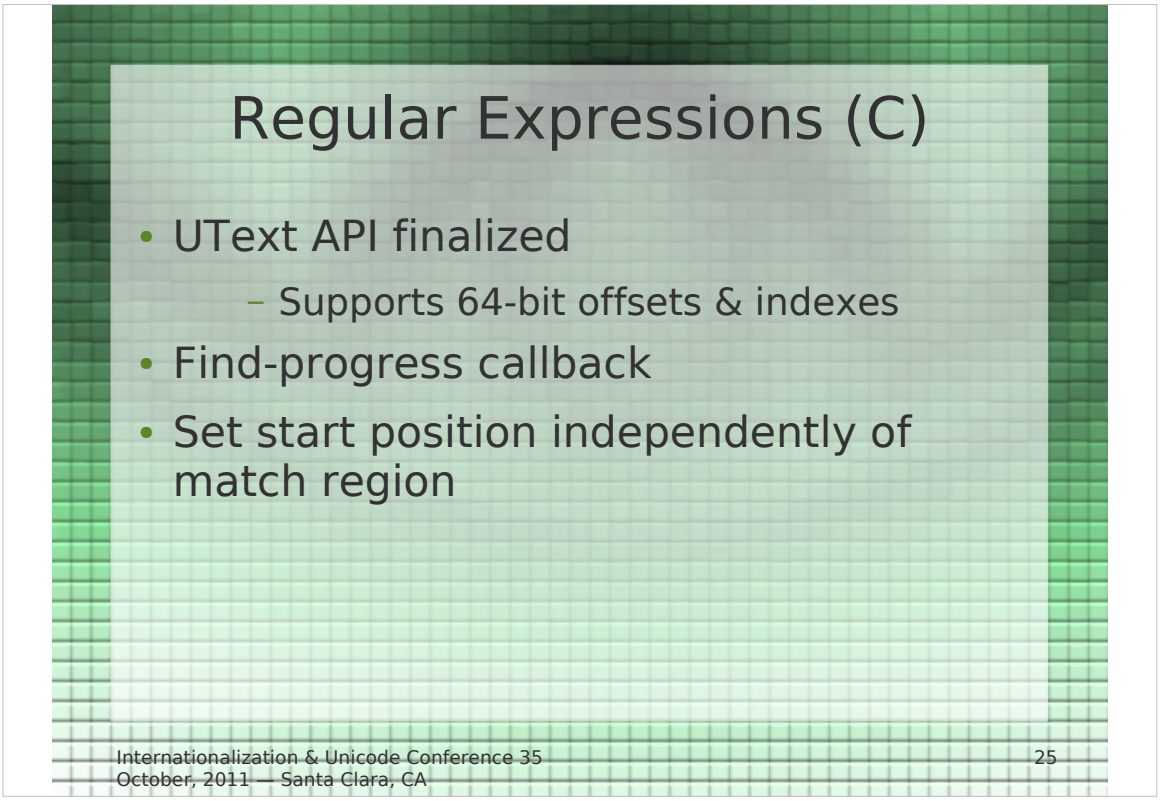


IDNA was upgraded from Unicode 3.2 to Unicode 6.0 and later, but the spec makes incompatible changes and drops the standard mapping table.

UTS #46 bridges IDNA2003 and IDNA2008 by providing a standard mapping table and adding optional “transitional” mappings for ß, ç, ZWJ and ZWNJ.

The new ICU API uses a factory to resolve options early and returns a set of errors. In C++, there is special API and a fastpath for UTF-8.

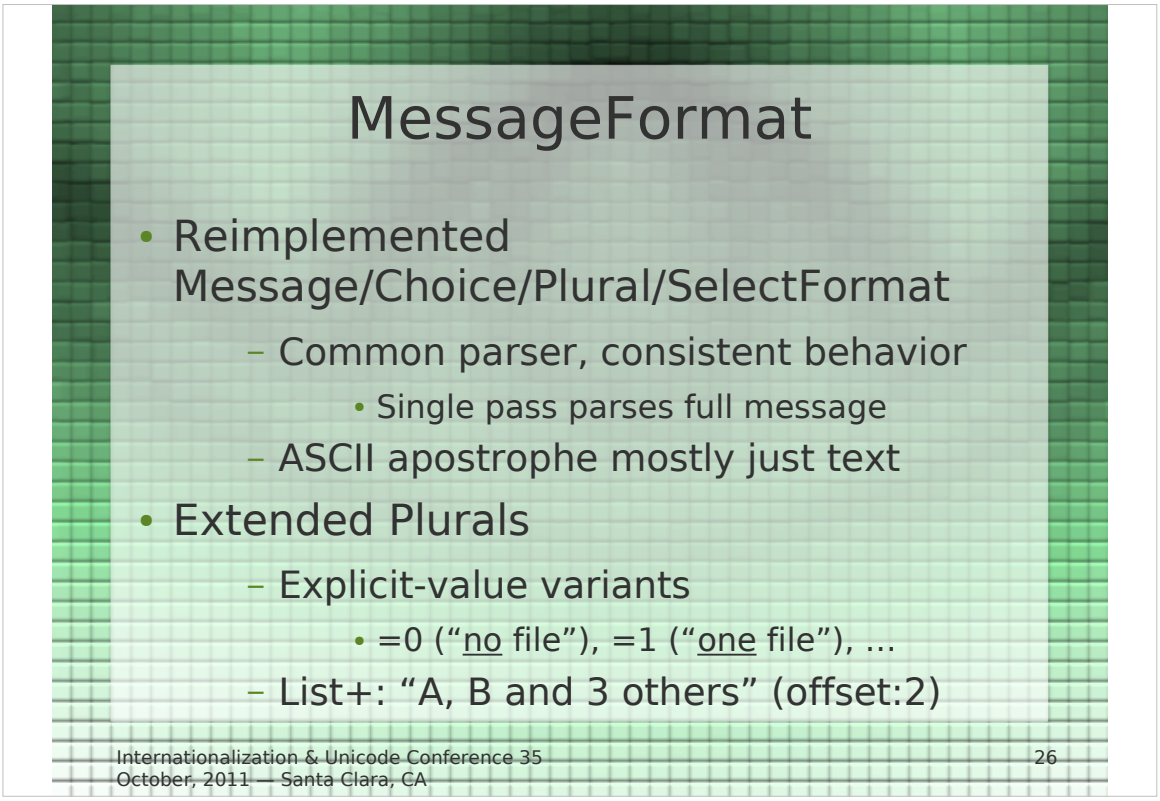
The UTS #46 implementation is built around a custom normalization table (Normalizer2), reducing the number of passes over the strings.



In addition to UnicodeString (C++) and UChar* (C), the regex API now also supports the UText interface. This interface supports discontiguous text storage (such as in OS X/iOS Cocoa's NSString) as well as 64-bit lengths and offsets, so there are also some new parallel API that use/return int64_t instead of int32_t.

The find-progress callback is invoked after each return from a match attempt, giving clients the opportunity to terminate a long-running find operation.

New API allows setting the start position without resetting region limits, so the start need not be at the beginning of a region. This is very useful when resuming a long-running search after it has been interrupted, e.g., in response to a find-progress callback.



ICU 4.8 reimplements MessageFormat and its complex-type arguments using a new, common parser which parses the entire message at once, not just the top level. The new parser is also public API (class MessagePattern).

ASCII apostrophes are now mostly just text (“O'Brien”) except when quoting a small set of syntax characters. Nested messages now consistently behave the same as the top level. Other details of the syntax have also been regularized, widened and/or documented.

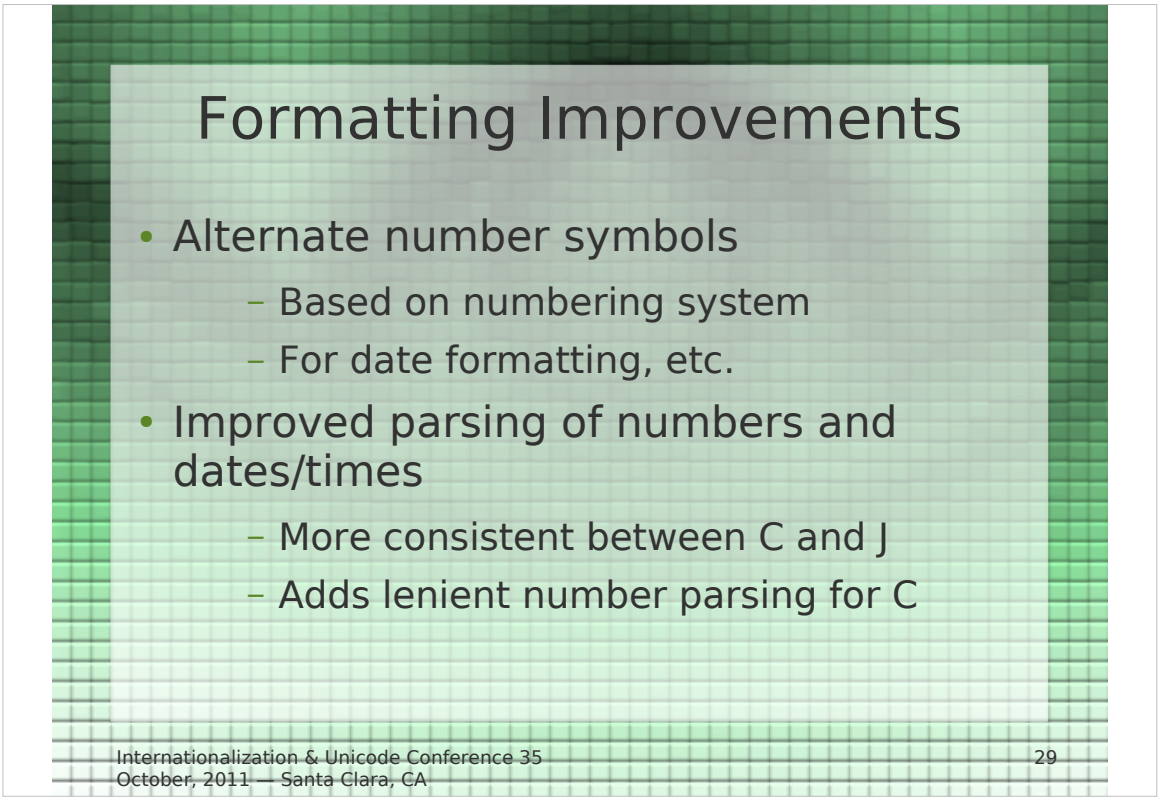
Plural arguments support explicit-value variants in addition to PluralRules keywords, for more natural messages, especially for 0 and 1. They also support a small number of directly mentioned items in addition to “and x others”.

ASCII Apostrophe

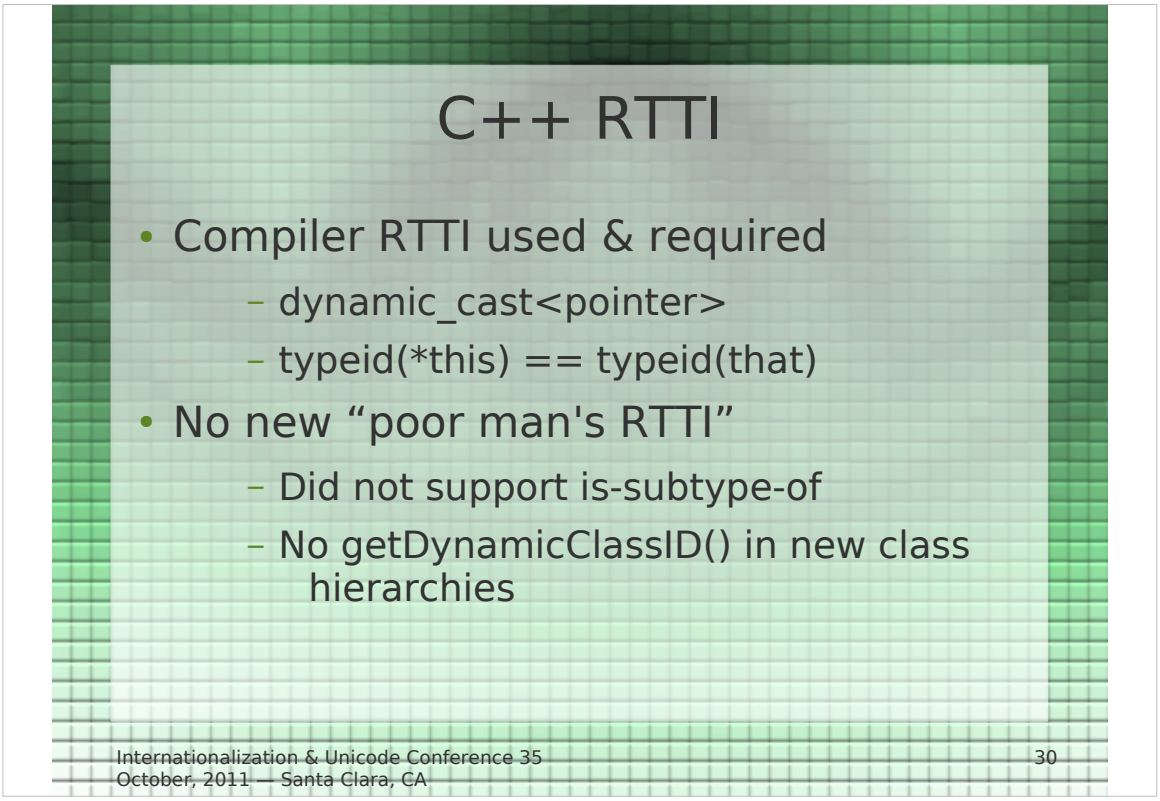
- Java: `"I don''t see {0,choice,0#O''''Brien in {1}}"`
  - Double or quad apostrophe in nested message depending on other contents
- ICU 4.8: `"I don't see {0,select,other{O'Brien in {1}}}"`
  - ' only starts quoting literal text if
  - preceding { or } or # (in plural) or | (in choice)
  - '' still works for single ' output

Extended Plurals

- "{num_people,plural, offset:1
  =0{No one went}
  =1{{person1} went}
  =2{{person1} and {person2} went}
  other{{person1} and # others went}}."
-
- "#" = remainder = num_people - offset



When switching the numbering system, for example between Latn (ASCII digits) and Arab (Arabic-Indic digits), some of the symbols need to change as well, especially the decimal and grouping (thousands) separator.



ICU4C was started when few C++ compilers supported RTTI reliably. ICU instead used a “poor man's RTTI” API. It only supports is-same-type-as, not is-subtype-of. (All known subtypes had to be tested, which was fragile.)

Modern compilers all support RTTI.

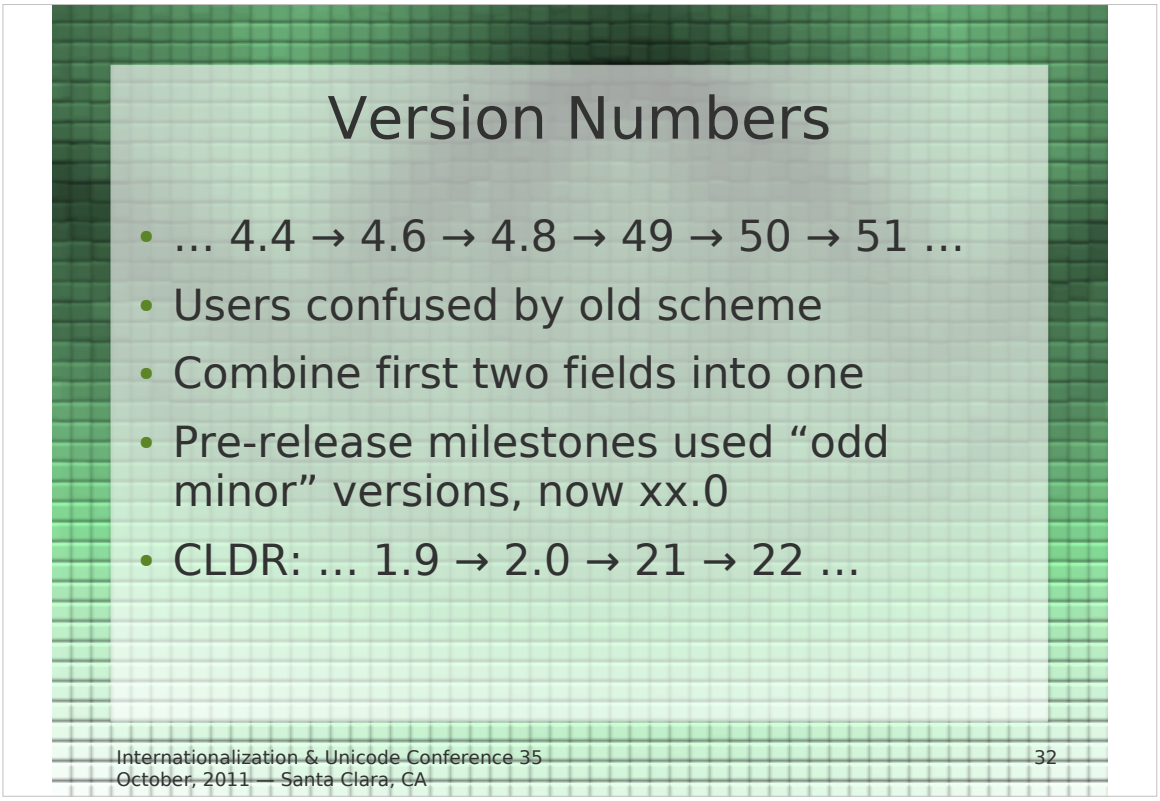
ICU4C 4.6 requires and uses compiler RTTI. It uses dynamic_cast on pointers, not references (which would throw exceptions). It uses equality of typeid in polymorphic operator== (but does not otherwise use the type_info class).

Existing “poor man's RTTI” remains supported. In particular, new classes in existing hierarchies continue to be added with “poor man's RTTI”.

New class hierarchies (e.g., Normalizer2) are added without “poor man's RTTI”.

What's next for ICU?

- Version Numbers
- Release Schedule
- ICU 49 Release Summary

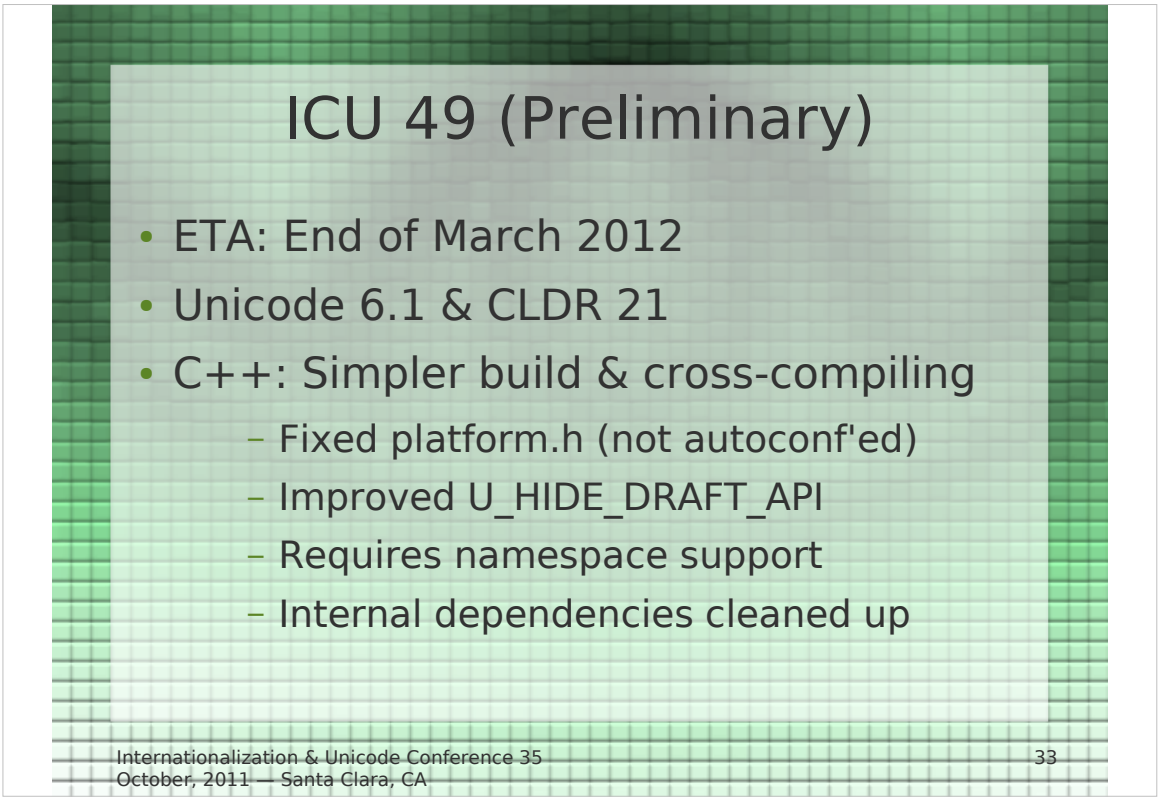


From ICU 1.6 (2000) to 4.8 (2011), the first two version fields together identified an ICU release. Versions with even second-field numbers were used for supported releases, odd numbers for development releases or milestones. Library names contained the two concatenated fields (“libicui18n.so.48”).

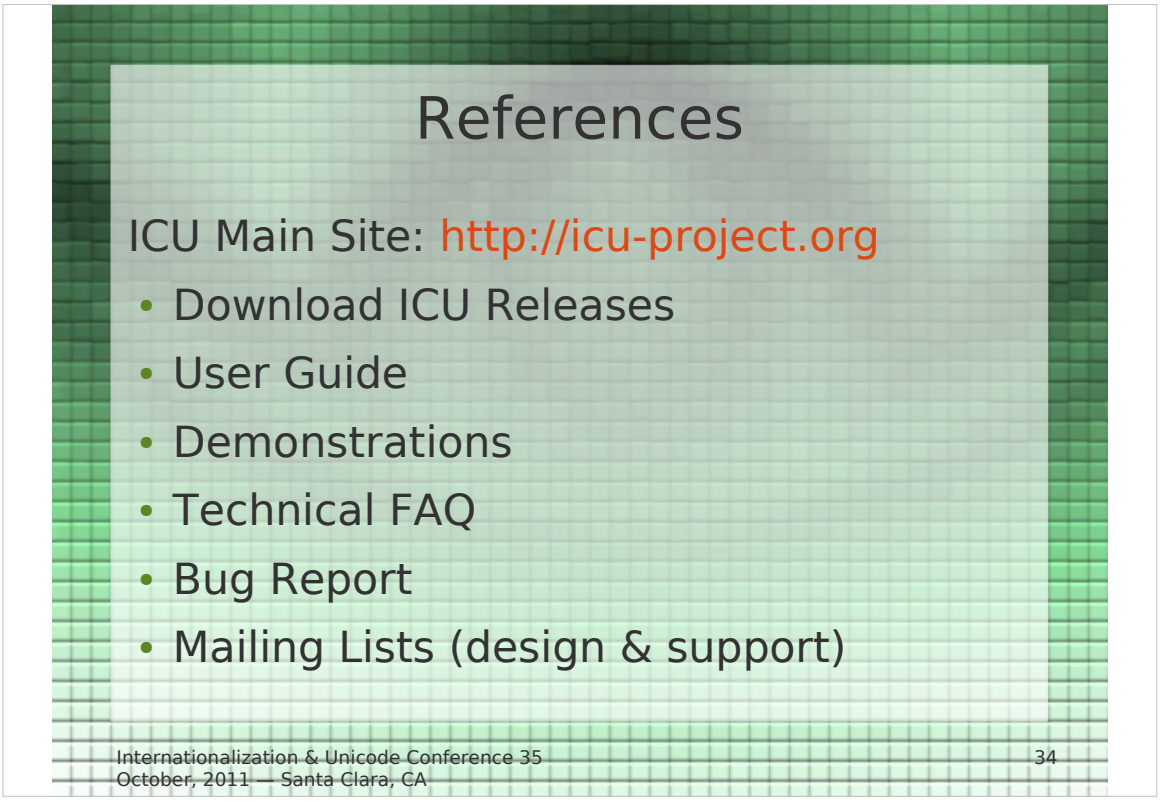
FAQ: “Can I use 4.7 in production?” – “I will wait until the next major x.0 release”

Change: ICU will use only the first field to identify releases. After 4.8 comes 49. Library names will use the first field (“libicui18n.so.49”). Pre-release milestones will be vv.0.m (e.g., 49.0.1), full release vv.1 (49.1), maintenance updates vv.u (e.g., 49.2).

CLDR is changing as well.



The next major ICU release is scheduled for 2012q1. It will support the upcoming Unicode and CLDR releases. We are working on simplifying the C++ build, and on a number of other features.



If you would like more information about ICU, you can go to our main site on http://icu-project.org There you will find links to download ICU, the ICU User Guide, the technical FAQ, where to get support, demonstrations of how ICU works, and many other topics related to ICUYou can also find more information about Unicode at the unicode.org site.