

UTF-16 and C/C++ language

TANAKA Keishiro EBLE Markus



18th International Unicode Conference



• Why do we need UTF-16 in C/C++ ?

• How to support UTF-16 in C/C++

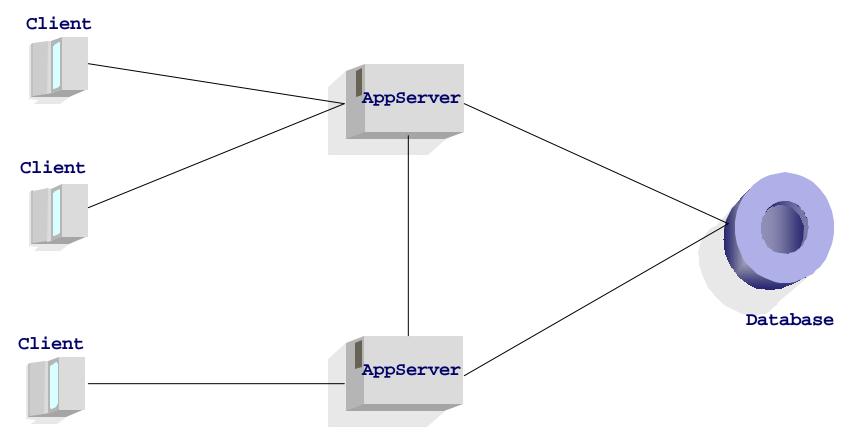
• Practical experiences

Question period





Classic Client-Server Architecture

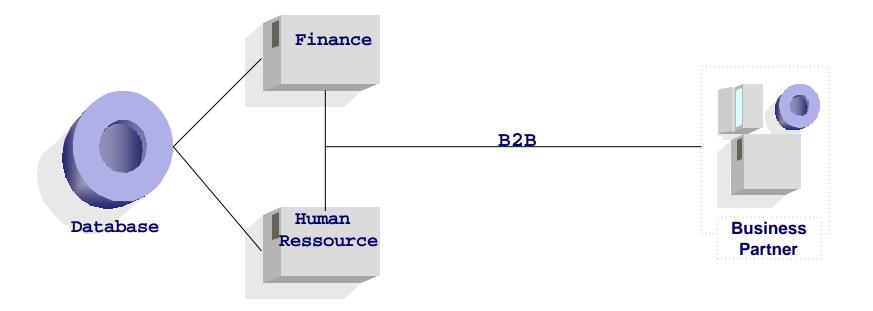


Each client wants to use his native language and script





Typical B2B Collaborative Systems



How to communicate between different companies ?





Ok, let's use Unicode.

But which encoding shall we use ?

Considering:

- Integration with the existing Unicode products
- Migration of existing non-Unicode products
- Performance and memory consumption





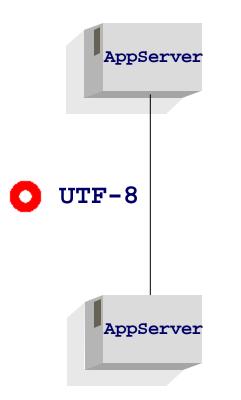
Inter-process communication
 Communication across the border of an address space
 Maybe within one machine or cross machines
 Data representation may differ

In-process communication
 Communication within one address space
 e.g. Function call into a shared library
 Data represention shareable





Inter-process communication



- No endian problems
- Minimum average data size
- Limited communication with non-Unicode systems possible





Which encoding is good for in-process communication in C/C++ programs ?

- In-process communication typically has much higher frequency than inter-process communication
- In-process communication has high performance requirements
- Time consuming data conversion should be avoided
- The same data representation should be shareable between several programming languages
- Encoding of text data should be defined exactly





Structure of SAP application server

User Interface	ABAP (UTF-?)
	JavaScript (UTF-16)
Business Logic	ABAP (UTF- ?)
Technical Base	SAP Kernel C/C++ (?)
	DB-Interface(UTF-8/UTF-16)

Why did SAP choose UTF-16?



18th International Unicode Conference



UTF-16 combines the worst of

UTF-8 and UTF-32?

- Use UTF-32 to get rid of multibyte handling
- Use UTF-8 if memory consumption is important





UTF-32 vs UTF-16

- In UTF-32 one 32-bit unit encodes one character No multibyte handling required But handling for combining characters needed
- UTF-32 requires about twice the memory size for text data
- UTF-32 would cause about 70% more memory consumption for the whole SAP system





UTF-8 vs UTF-16

- UTF-8 has the minimum average character size
- Size is language dependent
- For fixed-size buffer 50% larger size is required to cope with worst case
- UTF-8 causes higher complexity of algorithms because up to 3 bytes look ahead is required
- Error handling more tricky





Important Unicode products using UTF-16

- Java: The main programming language for the Internet
- Microsoft Windows: The most important frontend platform
- Databases: UTF-16 API available from all vendors
- IBM ICU: Platform and programming language independent internationalisation library



SAP

Text processing support in C/C++

- Character and string literals
- Character data type
- C standard library functions
- C++ standard library classes





String literal definition in C/C++

- "single byte string" implementation-defined encoding 8 bits width on any systems character can consist of multibyte character seq As encoding, ASCII is used broadly
- L"wide character string" implementation-defined value and size character consist of fixed length value There is no de facto standard encoding
- From the pragmatic point of view, fixed bit width on any systems (16bits preferred) and well-known and broadly used encoding character and string literals are required





u"Unicode string literal"

• Suggestion:

u^wUnicode string literal" e.g u"Hello, Unicode" Multibyte characters are converted to 16 bits width code value sequence by C/C++ compiler

 \u escape sequence supports all planes e.g u"A\u3230B" is encoded as 0x0041 0x3230 0x0042





Merits of u"Unicode string literal"

- Well defined encoding value
 On different systems, we can get same result.
- Less storage consumption than wchar_t
 By comparison 2 bytes u"literal" and 4bytes wchar_t, the half storage consumption.
- Locale independent behavior of application Application works without system dependent locale mechanism if you need.
- Similiar to other languages, especially to Java.
- Easy to modify the existing source program.





C standard library for UTF-16 example strchr

```
wchar_t * wcschr(
const wchar_t *wcs, wint_t c) {
  wchar_t wc = c;
  do {
    if (*wcs == wc)
       return (wchar_t *) wcs;
    } while (*wcs++ != L'\0'));
    return NULL;
}
```

```
utfl6_t* strchrU16(
const utfl6_t *ucs, uint_t c) {
    utfl6_t uc = c;
    do {
        if (*ucs == uc)
            return (utfl6_t *) ucs;
        } while (*ucs++ != u'\0');
        return NULL;
}
```





- C++ standard library for UTF-16
- char_traits define basic properties for character type
- ctype defines character properties
- numpunct, moneypunct, time_get, time_put define how to parse and print numerical, monetary and time values
- codecvt defines conversion between internal and external character format





- Modification Fujitsu C/C++ compiler supporting u^wUnicode string literal" About 1 month by 2 persons
- Extension of standard C library 2 students, 6 months
- Extension of standard C++ library About 3 months by 1 person





- SAP kernel (Unicode and non-Unicode) is developed with one single source tree.
- SAP supports the following systems
 - Unicode without wchar_t (UTF-16)
 - Unicode system with 16bits wchar_t (UTF-16)
 - non-Unicode system (locale dependent)





- Introducing UTF-16 in C/C++ improves
 - memory consumption and performance
 - platform independence
 - integration with other programming languages
- Workload for implementation is moderate
- We suggest to add u"Unicode string literal" to the C/C++ standard or accept it as defacto standard spec

