

Managing Code Variants

Steven J Zeil

March 19, 2013

Contents

1	Problems	2
2	AUTOCONF	9
3	Dynamic Loading	14

1 Problems

Code Variations

- Environment management, Previously identified as common SCM problems:

Coping with change in

- hardware environment
- software environment

- Can lead to need for variant code to support different configurations

.....

The Sad Story of C/C++ Portability

- Both C and C++ existed as popular languages long before being standardized



- Widespread variations in the “system” headers
- Even after standardization, many common functions are not standardized
 - GUIs
 - multi-threading and distributed operations
 - network communications
- Even things covered by the standard aren't covered in enough detail

.....

C Portability Quiz

How would you declare an integer counter capable of holding non-negative values up to one million? Up to one billion?

- C90 requires `sizeof(short) ≤ sizeof(int) ≤ sizeof(long)`

Notice that's \leq , not $<$



A `texttchar` must hold a “natural” byte (minimum addressable unit) on the machine architecture.

- The C99 specification added `long long` and set minimum sizes as

<code>char</code>	8
<code>short</code>	16
<code>int</code>	16
<code>long</code>	32
<code>long long</code>	64

.....

C++ Portability Quiz

How would you declare an integer counter capable of holding non-negative values up to one million? Up to one billion?

- The C++ standard followed C90 (not 99!) until C++11
`sizeof(short) ≤ sizeof(int) ≤ sizeof(long)`



- C++11 (not yet implemented by most compilers) adds the C99 standards

.....

Coping With Variants in the C/C++ World

- Configuration headers used to define symbols describing selected variants, e.g.,

```
#ifndef CONFIG_STD
#define CONFIG_STD

//
// AlgAE Configuration file
//
// Currently recognizes g++, version 2.7.2 for Unix and 2.8.0
// MS Visual C++, version 5.0
//
```



```
// Define this if the compiler does not support reassignment of  
// buffers via the function rdbuf(streambuf&)  
#undef __bad_rdbuf__  
  
#ifdef __GNUG__  
    /* Compiler is gcc/g++ */  
  
#define MEM_INCL <mem.h>  
  
#define USING_STD  
#define STD  
#define USE_FORK
```



```
#ifdef __CYGWIN32__
    /* This is the GnuWin32 port for Windows 95/NT
#define USE_WINSOCK
#else
    /* This is some other port of g++, probably a Unix system. */
#endif

#elif defined(_MSC_VER)
/* compiler is Microsoft Visual C++ */

#define MEM_INCL <alloc.h>

#define USING_STD using namespace std;
#define STD std::

#define MEMDC
#define __bad_rdbuf__
```



```
#define USE_WINSOCK

#else

#pragma warning "Possible configuration error: Compiler is not r

#define MEM_INCL <mem.h>

#endif

#endif
```

- Code uses symbols defined in there
 - direct substitution, e.g.




```
#include MEM
```

```
loads <alloc.h> or <mem.h>
```

- or conditionally

```
#ifdef USE_WINSOCK  
#include <winsock2.h>  
#else  
#include <netinet/in.h>  
#include <sys/socket.h>  
#endif
```

.....

2 AUTOCONF

Compiling Software the Unix Way

If you've ever installed a Unix/Linux package from a source distribution, you've probably gotten used to the two-step process:



```
./configure  
make  
make install
```

- The configure script runs a series of tests on the compilation environment, e.g.,
 - operating system
 - compiler name
 - availability of selected libraries/header files
 - availability and/or behavior of selected functions
- Produces a Makefile and a configuration header config.h based upon the test results
- Source code may use conditional compilation based on the header to select appropriate code

.....



Generating The configure Script

A rough outline:

1. Create a `configure.ac`

```
AC_INIT(cppSpreadsheet, 1.0, zeil@cs.odu.edu)
AC_PREREQ([2.68])
AM_INIT_AUTOMAKE([1.16 foreign no-define])
AC_CONFIG_HEADERS([config.h])
AC_PROG_CXX
AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

.....

Generating The configure Script

2. Set up `config.h.in` (template for eventual `config.h` file)
3. Set up `Makefile.am`



```
AM_INIT_AUTOMAKE([1.10 no-define foreign])
```

```
bin_PROGRAMS = testssheet
```

```
testssheet_SOURCES=testssheet.cpp exprparser.cpp tokenizer.cpp exprf  
cellname.cpp numericnode.cpp stringnode.cpp cellrefnode.cpp neg  
absnode.cpp sqrtnode.cpp sumnode.cpp lessnode.cpp lesseqnode.cp  
greaternode.cpp greatereqnode.cpp equalnode.cpp notequalnode.cp  
subtractnode.cpp timesnode.cpp dividesnode.cpp ifnode.cpp \  
numvalue.cpp strvalue.cpp errvalue.cpp spreadsheet.cpp cell.cpp  
observable.cpp observerptrseq.cpp cellptrseq.cpp cellnameseq.cpp  
absnode.h          control.h          lessnode.h          ssi.h \  
binarynode.h       dividesnode.h       minusnode.h         ssview.h  
cell.h             elementseq.h        negatenode.h        streamtok  
celllistenerseq.h equalnode.h         notequalnode.h     stringnod  
cellname0.h        errvalue.h          numericnode.h       strvalue.  
cellname.h         expression.h        numvalue.h          subtractn  
cellnameseq.h     exprfactory.h      observable.h        sumnode.h  
cellptrseq.h       exprparser.h        observer.h          timesnode  
cellrange.h        greatereqnode.h    observerptrseq.h    unaryexpr
```



<code>cellrefnode.h</code>	<code>greaternode.h</code>	<code>plusnode.h</code>	<code>unarynode</code>
<code>clipboard.h</code>	<code>ifnode.h</code>	<code>spreadsheet.h</code>	<code>unittest.</code>
<code>constantnode.h</code>	<code>lesseqnode.h</code>	<code>sqrtnode.h</code>	<code>value.h</code>

.....

Generating The configure Script

4. `touch NEWS README AUTHORS ChangeLog`
or create real versions of these.
5. `run autoreconf -force -install`
 - Runs the sequence of programs: `aclocal autoconf autoheader automake`
 - Creates `config.h.in Makefile.in & configure`

.....



Alternatives

- imake for X code

.....

3 Dynamic Loading

autoconf is C/C++-centric

The configure approach relies heavily on conditional compilation features.

- Common in C++
- Only in Java via non-standard techniques

.....



Java: Abstraction

Java programs are more likely varied by altering entire classes at a time.

For example:

```
public abstract class OCRLauncher extends Thread {  
    /**  
     * Launch an OCR process to convert the input  
     * PDF into some kind of File of OCR output.  
     *  
     * @param inputPDFfile The PDF file to be converted to IDM (XML)  
     * @param outputFile The raw OCR output  
     * @return  
     */  
    public abstract boolean convertPDFtoOCR  
        (File inputPDFfile, File outputFile)  
        throws Exception;  
  
    /**  
     * Convert a file of OCR output into IDM  
     *  
     */  
}
```



```
* @param inputOCRfile  
*  
* @return XML (IDM) document  
*/  
public abstract Document convertOCRtoIDM  
    (File inputOCRfile) throws Exception;  
}
```

This class has distinct implementations for different OCR programs that might be installed on the running system.

.....

Configuration via Property Files

A property file, loaded at run time, specifies which class is actually desired:

```
input.OCRLauncherClass=edu.odu.cs.extract.input.OCRBatchLauncher  
input.OCRProgram=OCR  
input.OCRBatch=Batch  
input.ocr.in_dir=c:/Luratech/ocr_in  
input.ocr.out_dir=c:/Luratech/ocr_out
```



.....

Reflection: Dynamic Loading

And the desired class is loaded dynamically:

```
String OCRLauncherName
    = p.getProperty(Properties.Names.OCR_LAUNCH_CLASS);
Class<?> ocrLauncherClass
    = Class.forName(OCRLauncherName);
ocr = (OCRLauncher)ocrLauncherClass.newInstance();
idmDoc = ocr.convertOCRtoIDM(inputOCR);
```

.....

