Overview of Developing and Porting for OSR 6.0.0 and UW 7.1.4

Ron Record
rr@sco.com

John Wolfe
jlw@sco.com

SCO

Agenda

- Overview of SCO provided Development Systems
- Suggested open source tools
- Building open source applications
  - Getting Project Source
  - Configuration Issues
  - GCC-isms
  - Operating System Variance
  - C++ Issues

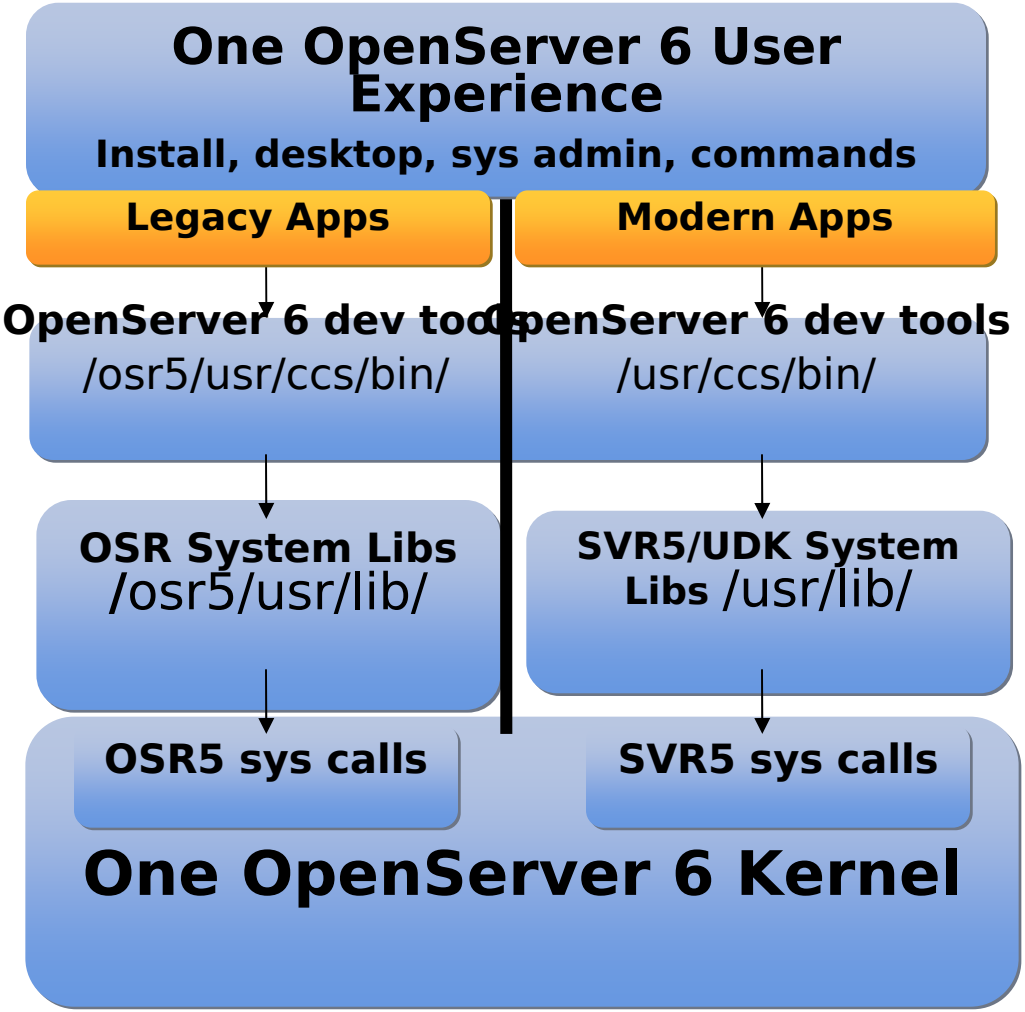
Review

OpenServer 6.0.0 Development System

SCO

How OpenServer 6 is Structured

OSR ABI



SVR5/ UDK ABI

Application Binary Interface – What is that?

- What an app looks like at the binary level
  - content and layout of information that it presents to system loaders and linkers (object file format)
- How different modules of an app communicate
  - function call conventions
  - size and layout of basic data types
  - size and layout of compound data types - structures, unions, bit-fields
- How an app communicates with the OS
  - pathnames, sys call numbers, errno’s, ioctl's
  - size and layout of basic and aggregate system data types

OpenServer 6 – SVR5 ABI - default

- OpenServer 6 Devsys for SVR5 (UDK) ABI
  - OpenServer 6 Devsys using **-K udk** option
    - or - **/usr/bin/cc** which defaults to **–K udk**
  - use for single certification on UnixWare 7 and OpenServer 6
  - use for modernizing existing OSR5 apps
  - use for device driver development (IHVs)
  - used to relink the OpenServer 6.0.x SVR5 kernel
  - provides access to “NEW” features
    - threads and LFS (> 2 Gbyte files)

OpenServer 6 – OSR ABI

- OpenServer 6 Devsys for OSR ABI
  - OpenServer 6 Devsys using **-K osr** option
    - or **/osr5/usr/bin/cc** which defaults to **–K osr**
    - set PATH environment variable with **/osr5/usr/bin** before **/bin**, **/usr/bin** or **/usr/ccs/bin**
  - use for binary compatibility to legacy OSR5 apps
  - provides more modern C and C++ compilers
    - Standards Conformance (almost) C and C++
    - same level of code generation and optimization as in the SVR5/UDK compilers
    - 64-bit “long long”
  - NOT available - threads or large files (> 2 Gbytes)

Mixing OSR and UDK ABI Object Files

- No safe way to link OSR5 ABI and SVR5/UDK ABI relocatable or shared objects (.o/.a/.so)
  - no way to intercept different system data types
  - no way to intercept different bit-field layouts and function calling conventions
  - no way to intercept system calls from objects
- Linker will reject mixture of objects, by default
- Force link mode provided - “I know what I’m doing”
  - but you probably don’t
  - not recommended

Building Legacy OpenServer 5 Apps

- When is OSR ABI needed?
  - when linking with existing OSR5 .o/.a/.so objects
- Use OSR ABI compilers
  - same as UDK but with **-Kosr** for OSR ABI
  - modern, reliable, standard, optimizing
  - 64-bit “long long” integer available
  - LFS, threads and EFT *not* available
  - can accept OSR5 COFF objects as input to linker
    - but cannot *generate* COFF
  - can link with existing OSR5 C .o/.a/.so objects
    - but *cannot* link with existing OSR5 C++ objects
  - use **CC -Xo** to compile very old OSR5 C++ sources

Want new features but need compatibility with old OSR5 library?

- If your own, recompile
- If from another ISV, get vendor to provide new, SVR5 ABI-built libraries
- If neither is possible ...
  - make app into 2 processes
  - one process calls old lib
    - compile **-Kosr**
  - one process uses new features
    - compile **-Kudk**
  - use socket, pipe, IPC, etc. to communicate between processes

Guidance on modernizing existing apps: Threads

- Must modify to use threads
  - pthreads API more standard than SVR4/UI threads
  - use **-Kudk** to recompile application
  - use **-Kpthread** when compiling threaded code
    - fixes some things like global errno automatically
- Existing OSR5 source may not be thread-safe!
  - may use non-reentrant functions such as **strtok()**
    - use **<name>_r()** replacements when available
  - may store application data globally
  - may return pointers to static data
  - must study your code

Guidance on modernizing existing apps: Large files

- Go forward with Large File Summit (LFS) APIs
  - use **-Kudk** to recompile application
  - create files up to one terabyte in size
  - can use size-specific interfaces
    - **fopen64**, **lseek64**, etc.
  - or, can use regular **fopen**, **lseek**, etc.
    - **cc -D_FILE_OFFSET_BITS=64**
    - **off_t**, etc. become 64 bits
  - must use vxfs filesystem and create filesystem with **largefiles** flag
    - **mkfs** or **fsadm_vxfs** to turn on/off
  - **ulimit** must be set to **unlimited**

Guidance on modernizing existing apps: Fundamental system types

- Be careful with expanded fundamental system types (EFT)
  - Size change between OSR5 and OSR6 in UDK mode:
    - mode_t, dev_t, uid_t, gid_t, nlink_t, pid_t, ino_t, sigset_t
    - typically size goes from 16 bits to 32 bits
    - system or app struct's containing them also change size
      - e.g., struct stat contains both dev_t and ino_t
    - dev_t also changes how major, minor numbers packed
    - all consequences of SVR5 infusion into OpenServer 6 kernel
  - Change should be transparent unless your code has assumptions about size

Guidance on modernizing existing apps: C++

- Existing OSR5 DevSys C++ compiler is old!
  - AT&T Cfront-based, c. 1992, buggy
  - predates 1998 ISO language/library standards
  - large-scale changes in language since then
- If your sources were developed with it ...
  - expect they will *not* compile cleanly now
  - source fix-ups are usually straight forward
    - you're doing your code a favor!
  - for bad cases try the CC -Xo option
  - old library classes will all still be there

Guidance on modernizing existing apps: C++ ABI issues

- C++ ABIs are unique for each compiler
  - Exception handling implementation
  - Class object layout
    - Virtual function table pointer position
    - Base class sub-object order
  - Virtual function call mechanism
    - Virtual function table format
    - Use of “thunks”
  - Name mangling conventions
- Cannot mix C++ compiler objects
  - SCO (USLC) C++ ≠ Cfront C++ ≠ GNU g++

Features of the OpenServer 6 Development System

- C Compilation System
- C++ Compilation System
- C/C++ Debugger
- memtool
- fur
- Except where noted, features apply to Dev Sys used for both SVR5/UDK and OSR ABIs and to UDK on UW7
- A major upgrade compared to existing (and outdated) OSR5 Development System product!!

The OpenServer 6 -K mode switch

- Compilers
  - **/usr/ccs/bin/cc** defaults to **-Kudk**
  - **/osr5/usr/ccs/bin/cc** defaults to **-Kosr**
  - “cross-ABI” compiles are allowed
    - **/usr/ccs/bin/cc -Kosr** ...
    - **/osr5/usr/ccs/bin/cc -Kudk** ...
  - ditto CC for C++ compiles – as & ld also
  - Use cc or CC to do linking – links against correct ABI startup routines.
- Other Dev Sys commands
  - have **-K osr | udk** option if necessary (e.g. lint)
  - don’t have option if irrelevant (e.g. lex and yacc)

OpenServer 6 C Compiler

- Robust compiler, excellent IA-32 code generation
- Standards-conforming libraries and headers
- Profiled versions of libraries
- prof, lprof in both ABIs
- fprof [SVR5/UDK ABI only]
- Standard set of command line tools, fully internationalized
- Conformance checking (-Xc) is against C 90 standard
- Support for Java native methods [SVR5/UDK ABI only]

- Almost all of C 99 - ISO/IEC 9899:1999
  - `inline`, `restrict`, variable argument macro functions, & 60 other features
  - Only things missing:
    - variable-length arrays
    - complex and imaginary numbers
    - minor variances in snprintf(3S)
  - [some new C99 library functions and headers may be SVR5/UDK ABI only]
  - Option –Xb will disable `inline` and `restrict`

OpenServer 6 C++ Compiler

- Accurate, robust implementation
- Almost all of the C++ standard - ISO/IEC 14882:1998
  - except rarely-used: export keyword, placement delete, function-try-blocks, two-phase template name binding, multi-byte characters in source code, partial specialization of a class member template

- Complete C++ Std Library
  - STL, iostreams, string, locale, numerics, etc.
  - fast and thread-safe
- Excellent IA-32 code generation
- Exception Handling - high
- performance
- Device driver support
- Thread safety [SVR5/UDK ABI only]
- Support for Java native

Basic - Suggested - Optional

Open Source Tools

SCO

Basic Open Source Tools

- Starter set
  - gmake
  - autoconf (2.13 and 2.59)
  - automake - synched with autoconf
  - GNU m4
- Probably will need (at sometime)
  - bison
  - gawk
  - flex

Highly Suggested Open Source Tools

- Depending on personal preferences, project build or change submission requirements ...
  - GNU diff
  - GNU patch
  - CVS – Concurrent Version System
  - GNU tar

Optional Open Source Tools

- GNU binutils (gas and ld)
  - OSR6 assembler
    - does not have Willamette SIMD instructions
    - Minor differences in SIMD mnemonics
- GNU GCC
  - SIMD instructions are in GCC “asm” statements
  - Avoid g++ especially for graphics
    - C++ ABI issues
- RPM
  - Use rpm2cpio to extract and examine spec files

Acquiring Open Source Tools

- OpenServer 5.0.7 GNU Development Tools
  - After **chsysinfo osr5**
    - Install GNU m4, bison, flex, diff, patch, awk, make, CVS and configuration creation tools
  - DO NOT INSTALL !!!!
    - GCC – not dual ABI aware
      - - Generates OSR5 ABI code
      - Looks in /usr/include for OSR 5 system headers
      - Looks in /usr/lib & /usr/ccs/lib for link libraries
    - GDB
      - - Not SVR5 kernel aware

Acquiring Open Source Tools (continued)

- UDK 7.1.4 OSTools set
  - Install individual packages – not the set

chsysinfo uw7

**pkgadd -d <mnt-pt> GNUm4 GNUautomk **
**GNUautocf GNUmake GNUawk GNUbison **
Osflex

  - GCC 2.95.3 and GDB are configured for SVR5
    - SVR5 /usr/gnu/lib/libstdc++.so.2.10.0

Acquiring Open Source Tools (continued)

- Additional tools or runtime required to build a project
  - Check for availability on Skunkware
  - May be part of project source
    - Part of the normal build sequence
    - May need to be built as a first step
  - May move to the front of your project list
  - May be optional interface(s)
    - Defer / omit now
    - Build later and rebuild complete project.

Building Open Source Applications

Getting Project Source

SCO

Getting the Source - From Where ?

- SCO FTP site
  - ftp://ftp.sco.com/pub/openserver6/600/opensrc
  - ftp://ftp.sco.com/pub/unixware7/714/opensrc
  - ftp://ftp.sco.com/pub/openserver5/507/opensrc/source
- SCO Skunkware
  - http://www.sco.com/skunkware
  - ftp://ftp2.sco.com/pub/skunkware/src/
  - ftp://ftp2.sco.com/pub/skunkware/osr6/src/patches/
  - ftp://ftp2.sco.com/pub/skunkware/uw7/src/patches/

Getting the Source – From Where (continued)

- Freshmeat web site
  - http://freshmeat.net
- FileWatcher web site
  - http://filewatcher.org
- Free Software Foundation FTP
  - ftp://ftp.gnu.org/gnu
- SUSE Source RPMs FTP site
  - ftp://ftp.suse.com/pub/suse/i386/update/<version>/rpm/src/
- SourceForge web site
  - http://sourceforge.net

Source Formats

- Varying Source release formats – choice of project maintainers
  - **tar** or **cpio** file archives
    - Often compressed – GNU **gzip** or **bzip2**
  - **zip archive files**
  - **cvs or svn repository on project hosted site**
  - **Linux source RPMs**
    - **Good source for recent patches**
    - **Spec file can provide configuration guidance**
  - **Start with patches from the last release**

Managing Source & Build Changes

- Important to track ALL changes
  - Avoid reinventing the wheel
  - Probably need most, if not all, changes in next release
  - Help others in the SCO community to customize to their needs
  - Ultimately to contribute source, build and config. changes back to the open source community
  - Others can reproduce problems and provide solutions or workarounds

Source Changes (continued)

- Preserve the original source file
  - Do not over-write previously saved originals

```
mv  [file]  [file].orig      # preserve orig file date
cp  [file].orig  [file]      # modified – today's date
chmod  uw+w  [file]
```

  - Create empty "original" for every "new" file

```
touch  [file].orig
```

Source Changes (continued)

- Use context or unified **diff** to capture changes

```
cd [TOP_OF_SRC_TREE]

for i in `find . –name ‘*.orig’`; do
  echo $i
  diff [-c|-u] $i ${i%.orig}
done > [project]_cumulative_patch.[date]
```

- Context or unified **diff** not applicable to non-text files
  - .jar, compressed data, binaries, .jpeg, .pdf, etc.
  - Copy/replace entirely

Source Changes - Build Afresh

- Some open-source projects are configurable for separate source and object directories
  - Makefile design/implementation
  - Blow away the object directory and make again
- Reconstruct project source
  - Unwind source into “clean” directory

    cd [TOP_OF_SRC_TREE]

    gzcat [compressed_tar_archive] | tar -xf -
  - Reapply cumulative patches

    **patch -b –p0 < [project]+cumulative_patch.[date] **

    2>&1 | tee log.patch

Source Changes - Using Previous Patches

- Prev. release patches may not apply cleanly
  - Source code changes in area of your patch
  - Some changes bought-back into project source
  - Project source restructure
- Unapplied patches written to [file].rej
  - Review rejections – rework as needed

```
find  .  -name  '*.rej'
```

Building Open Source Applications

Configuration Issues

SCO

config.guess

- 2001 submitted UW7 changes to FSF to standardize SVR5 triplet
  - Handled OpenUNIX 8
    - **i?86:*:5:[78]***
  - You may need to update for OSR 6.0.0
    - **i?86:*:5:[678]***
  - Produces triplet
    - **i?86-unknown-sysv5<OS name><version>**

Configure Script's Triplet Override

- preset HOST / TARGET / BUILD
- SVR5 ABI
  - i586-sco-sysv5
- OSR5 ABI
  - i586-sco-sco3.2v5.0.7
  - add -Kosr to CFLAGS, CXXFLAGS, LDFLAGS
    - or set **PATH** for OSR5 ABI preference

Configure Scripts

- Override default use of gcc, if installed
  - **CC="cc"**
  - **CPP="$CC –E"**
  - **CXX="CC"**
  - **RANLIB=true**
- Use cc or CC to do the linking
- Avoid use of compilation or linking options that specify default header or library paths
  - Avoid **-I/usr/include –I/usr/include/sys**
  - Avoid **-L/usr lib -L/usr/ccs/lib**

Absence of config.guess

- configure and configure.in use uname
  - SCO_SV typically configures for OpenServer 5
    - Correct if using OSR5 ABI
    - Unable to handle LFS files
- Resolution – recognize SCO_SV and release 5 as OpenServer 6.0.0 and force selection of SVR5
  - Hand edit the configure script
- or
  - Modify autoconf/aclocal.m4
    - Rerun autoconf to regenerate an updated configure

scoutils

ftp://ftp2.sco.com/pub/skunkware/osr6/vols/scoutils-1.3Sc-VOLS.cpio

- Shell script frontends
  - Configure & build open-source
    - **/usr/bin/Configure, /usr/bin/Build & /usr/bin/Prep**
  - Project source at:
    - ***/usr/src/sco/*<category>/<project>-<version>.tar.bz2**
  - Project patch at:
    - ***/usr/src/sco/patches/*<project>-<version>-osr6.patch**
    - ***cd /usr/src/sco/*<category>**
    - **Build <project>**

Scoutils (continued)

- Build
  - Extracts source
  - Applies patch
  - Run
    - **<project>-<version>/Configure-OSR6**, if it exists
    - **/usr/bin/Configure**, otherwise
  - Then run
    - **<project>-<version>/Build-OSR6**, if it exists
    - GNU make , otherwise

Building Open Source Applications

GCC-isms

SCO

Need information about gcc extensions?

- Check the gcc information provided in earlier ports
  - OSTools – UW 7.1.4
  - GNUTool Chain – OSR 5.0.7
  - **/usr/gnu/bin/info gcc**
    - Select “C extensions”

GCC-isms: VarArg Macro Functions

- GCC provided early VarArg Macro Functions

```
#define eprintf(format, args...) \
        fprintf(stderr, format, ##args)
```

- Supported ISO/IEC 9899 Standard feature

```
#define eprintf(format, ...) \
    fprintf(stderr, format, __VA_ARGS__)
```

- Condition the change

```
#ifdef __USLC__
    ...ISO format
#else
    ...GNU format
#endif
```

GCC-isms: return <void expression>

- GCC accepts:

```
void bar() { return;}

void foo() {
        return bar();
}
```

- To be ISO compliant, change to:

```
void foo() {
        bar();
        return;
}
```

GCC-isms: inline C functions

- GCC supported “inline” C functions
  - Treat function as statement expression at point of call
- ISO/IEC 9899 added “inline” C funtions
  - Supported on OSR 6.0.0 and UW 7.1.4
  - Designed to work with C++ “inline” in common headers
  - Requires 1 and only 1 external definition generated
    - If in module source file, probably not an issue
    - Potential PROBLEM if in a header file
      - Suppress “inline” keyword during configuration

        CC=“cc –Xb”

GCC-isms: statement expressions

- Compound statement in parentheses
  - Probably encountered in **#define**

```
#define maxint(a,b) \
  ({int _a = (a), _b = (b); _a > _b ? _a : _b; })
```

- If in a header file, conditionally replace with C static function

```
static int maxint(a,b) {
   return (a > b ? a : b);
}
```

GCC-isms: __attribute__

- Functions – specify side-effects
- Variables – packed, aligned, section, weak
- Types – packed, aligned
- Format – in declarations or definitions
  - **__attribute__((<attr_name>[(<arg>)]))**
- Change needed:
  - Conditionally remove attribute modifier
  - Use, as appropriate:
    - **#pragma pack(<n>)**
    - **#pragma weak <id1> [= <id2>]**

GCC-isms: Enhanced Asms

- Feature is generally “unique” to each compiler
- Used for:
  - Better or specialized optimization/performance
  - Access to hardware registers/instructions not typically utilized by the C/C++ code generator
- With exception to Willamette SIMD instr.
  - Recode to SCO Enhanced ASM Function
    - Prototyped as function Called as a function
    - Follow i386 calling convention
      - Preserve user and stack registers – edi, esi, ebx, ebp, esp
      - Return values in eax (edx) or fp0

GCC-isms: Enhanced Asms (continued)

- OSR 6.0.0 Documentation
  - Software Development
    - Programming in C and C++
      - Enhanced ASM facility

asm [type] identifier ([param-list]) {
[storage-mode-spec-line
asm-body] +
}

storage-mode-spec-line:
% [storage-mode [identifier [, identifier]*];]+

GCC-isms: Enhanced Asms (continued)

- **Enable optimization of function calling ASM function by:**

  #pragma partial_optimization <identifier>

- If and only if:
  - Followed calling and register conventions
  - Register **%ebp** has not been modified
  - Register **%esp** not modified with **movl**
  - No branch into or out of ASM function
  - auto or param only modified if address of variable is passed to ASM function
  - Auto or param accessed if passed by name or address to ASM function

GCC-isms: no equivalents (at present)

- Extended ASMs with Willamette SIMD
  - Use GCC or separate assembly source compiled with GNU assembler
- Variable Length Arrays
  - Local can be recoded using alloca() at function entry

GCC-isms: Command Line

- Delete -Wall and other GCC -W arguments from configure/configure.in/Makefile.in
- Replace -shared with -G
- Replace -Wl,-soname with -Wl,-h
- Replace -fpic with -Kpic
- Check any -f arguments

Building Open Source Applications

Operating System Variance

SCO

Operating System Variance

```
#if defined(__USLC__)
#define __FUNCTION__   __func__
#endif

#if defined(__USLC__)
    #include <heimdal/roken/ifaddrs.h>
#else
    #include <ifaddrs.h>
#endif
```

And add -lroken to LIBS

OS Variance (continued)

- Sometimes need to add -lgetopt -lsocket -lnsl ... to LIBS
- Use scoutils libsym script to find location of unresolved symbols
- Use scoutils findinc script to find location of structures etc in header files. For example, <sys/sockio.h> contains #defines for SIO... whereas these may be defined elsewhere on Linux (e.g. <linux/sockios.h>)
- When linking with -lpcap add -lresmgr

Operating System Variance

Track down differences in names for type declarations and structure entry names.

```
#ifdef __USLC__
#ifndef __s32
#define __s32 int32_t
#endif // __s32
#ifndef __u32
#define __u32 u_int32_t
#endif // __u32
#endif // __USLC__
```

Building Open Source Applications

C++ Issues

SCO

Template Instantiation

- Different behavior GNU g++ and SCO C++
  - Can present problems in compilation or linking
- GNU g++
  - Instantiates all possibly needed templates in each object file
    - Separately named .text sections
  - GNU collect2/ld eliminates “duplicates” when linking

Template Instantiation (continued)

- SCO (USLC) – “implicit instantiation”
  - C++ compiler determines where/when templates are instantiated
    - At “link” time - When collected into **.so**, **a.out** or **.a**
    - Use CC command to do the linking
  - Implementation
    - Template declaration in xxxx.h
    - Template definition in xxxx.c – same directory as xxxx.h
    - Auxiliary files – created by compiler
      - source.ti & source.ii (where .o created)
        - Info to recompile
        - Templates visible and to be instantiated in that .o
    - C++ compiler implicitly includes xxxx.c for needed template in xxxx.h

Template Instantiation (continued)

- Non “implicit” source construction
  - Declaration and definition in header file
    - Similarly named **.c** file visible
      - Probably related in functionality since same name
      - Contains non-template class/function definitions
    - If .c file is implicitly included in multiple .o
      - Multiply-defined errors at link time
  - Header also contains non-template class/function definitions
    - Multiple definitions if headers used by more than single .o
  - Solution: use preprocessor defines to control visibility of non-template definitions

Template Instantiation (continued)

- C++ templates & archives
  - Object file is now disassociated from .ti & .ii
    - Cannot recompile to get “needed” instantiation
    - ERROR: - undefined template function later in the build
  - “needed” templates must be resolved prior to adding to archive

```
CC  –Tprelink_objects  $(OBJS)
ar  <options> <archive_file> $(OBJS)
```

Friend Name Injection Change

- Slight scoping change in the 1998 C++ Standard
  - Previously “friend” name was injected in the enclosing scope
    - If file scope, became friend to everyone
  - Pre-GCC 3.x code may run into this
    - Most has probably been updated over the last 4 years

C++ variadic macro definitions

- Variadic macro definitions of the form:

```
#define PRT(buf, format, ...) \
  snprintf(buf, sizeof(buf), format, __VA_ARGS__)
```

- **Not currently part of ANSI/ISO C++ Standard**
- **To accept this extension, use the C++ compilation option**

```
-Wf,--variadic_macro
```

Guidance / Assistance

OpenServer 6 Support Resources

- Porting Guide:
  - http://www.sco.com/support/docs/openserver/600/porting/osr6portingTOC.html
- Upgrade Guide:
  - http://www.sco.com/support/docs/openserver/600/upgrade/index.html
- Online Documentation and Late News
  - http://www.sco.com/support/docs/openserver/

OpenServer 6 Support Resources

- Support Download Page for OpenServer 6:
  - http://www.sco.com/support/update/download/product.php?pfid=12&prid=20
- Tricks on getting OpenServer 5, UnixWare, SCO Unix and SCO Xenix applications running on SCO OpenServer 6 – Forum 2006
  - http://www.sco.com/2006forum/breakouts/breakout/140_Boland_J_tips-tricks.ppt

OpenServer 6 Support Resources

- SCO “Legend” Mailing List: Public
  - Legend-subscribe@list.sco.com
  - legend@sco.com
- Porting/Migration Alias:
  - osr5to6@sco.com
- Knowledge base:
  - http://wdb1.sco.com/kb/search