Introduction to cg_xml

November 2018

Agenda

- Marketing
- How to Run the Scripts
- Popular Scripts
- Build Your Own

Categorize Memory Usage

```
C:\dir>ofd6x -x app.out | sectti
Reading from stdin ...

*******************************************************
REPORT FOR FILE: app.out
*******************************************************
                  Name : Size (dec)   Size (hex)   Type    Load Addr
---------------------- : ----------   ----------   ----    ----------
                 .clk :          12   0x0000000c   UDATA   0x0000f0b4
             .hwi_vec :         512   0x00000200   CODE    0x00000000
                 .swi :         220   0x000000dc   UDATA   0x0000ecd0
                 .idl :          32   0x00000020   UDATA   0x0000dee0
                 .bss :         896   0x00000380   UDATA   0x0000d800
                 .far :        2124   0x0000084c   UDATA   0x0000c340
<snip ...>

-------------------------------------------------------------
Totals by section type
-------------------------------------------------------------
  Uninitialized Data :        7604   0x00001db4
    Initialized Data :        9478   0x00002506
                Code :       48800   0x0000bea0
```

See Library Memory Footprint

```
C:\dir>lib_footprint app_map.xml

algrf.l62
      CODE ----------------------------------------
                    .text:       2336   0x00000920
      DATA ----------------------------------------
                   .cinit:         44   0x0000002c
     UDATA ----------------------------------------
                     .far:         16   0x00000010
                          ==========   ==========
    LIB TOTAL            :       2396   0x0000095c

biosi.a62
      CODE ----------------------------------------
                    .bios:      20352   0x00004f80
                 .sysinit:        224   0x000000e0
                    .text:        512   0x00000200
      DATA ----------------------------------------
                   .cinit:       1400   0x00000578
                   .const:        110   0x0000006e
...
```

See Stack Usage

```
C:\dir> call_graph arm_hello.xml | more
Call Graph for arm_hello.out
************************************************************************
_c_int00 : wcs = 668
|  __args_main : wcs = 668
|  |  _main : wcs = 668
|  |  |  _printf : wcs = 664
|  |  |  |  __printfi : wcs = 632
|  |  |  |  |  __pproc_fflags : wcs = 0
|  |  |  |  |  __pproc_fwp : wcs = 36
|  |  |  |  |  |  _atoi : wcs = 4
|  |  |  |  |  |  _memset : wcs = 8
|  |  |  |  |  __pproc_str : wcs = 56
|  |  |  |  |  |  _free : wcs = 20
|  |  |  |  |  |  |  _minit : wcs = 8
|  |  |  |  |  |  |  _minsert : wcs = 8
|  |  |  |  |  |  |  _mremove : wcs = 0
|  |  |  |  |  |  _malloc : wcs = 20
|  |  |  |  |  |  |  _minit : wcs = 8
|  |  |  |  |  |  |  _minsert : wcs = 8
|  |  |  |  |  |  |  _mremove : wcs = 0
...
```

See Stack Usage of a Library

```
C:\dir>call_graph --stack_max rts6400.xml
_strftime : wcs = 1248
```

Compare Libraries

```
C:\dir>objdiff -v l1.lib l2.lib ofd6x
Processing l1.lib ...
Processing l2.lib ...
Comparing files ...

************************************************************
Filename: algrf_exit.o62
************************************************************
============================================================
Comparing Sections
============================================================
Raw data in section .text:exit is different
           Byte 0 is different
              l1.lib: 0x62
              l2.lib: 0x30

Files are different
```

How can you do all that cool stuff?

Agenda

- Marketing
- **How to Run the Scripts**
- Popular Scripts
- Build Your Own

Install cg_xml

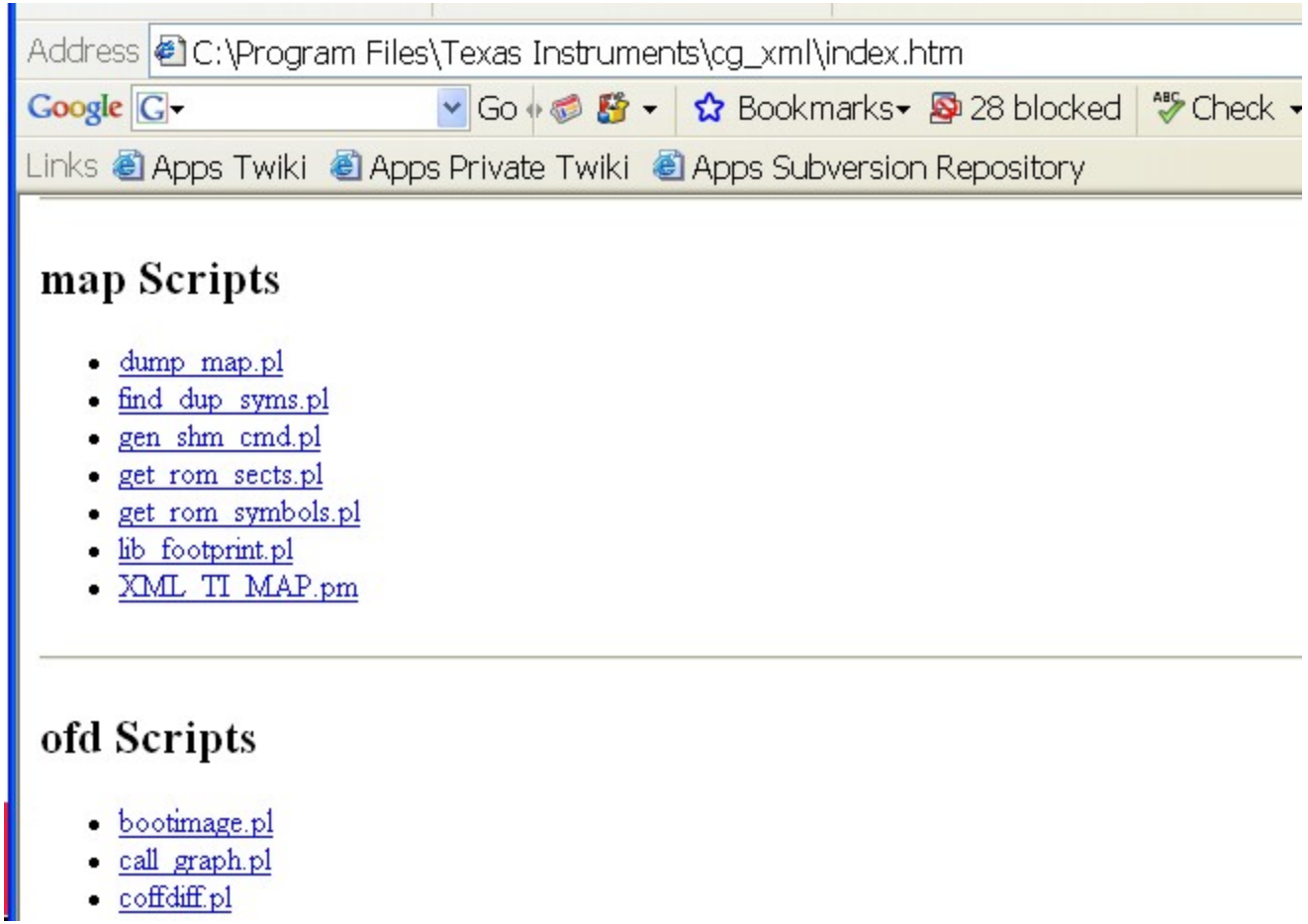
- Download Link
- Supports Windows, Linux, and Mac
- These slides are in the package

Execute Binaries

- Executable binaries are located in *install_root*/bin
  - *install_root*: where cg_xml is installed
- Execute like any other binary
  - Perl not required
  - Typically add *install_root*\bin to path
- System temporary directory is used
  - Files added the first time an executable runs
  - Subsequent runs use those files

HTML Documentation

- **Point browser to *install_root*/index.htm**



Support Details

- Works with all TI toolchains
  - ELF or COFF object file formats
- Can handle ELF code from other toolchains
  - ARM GCC has been tested
- Some scripts require Dwarf information
  - call_graph, global_types_gen, func_info
  - Only work with TI toolchains

A Bit of Background

- Developed as Perl scripts
  - They started small. Now they're pretty big.
- Thus, they are called "scripts" even when you might be running executables

Two Kinds of Scripts

- OFD
  - Input is XML from Object File Display Utility
  - Can process object, executable (.out), and library files
    - Some scripts do not process libraries
- MAP
  - Input is XML form of the linker map file

Run OFD Scripts

- Must use OFD -x option to create XML
- Can create separate XML file

```
C:\dir>ofd6x -x app.out > app.xml
C:\dir>sectti app.xml
...
```

- Or pipe it in
  - Slower on Windows systems

```
C:\dir> ofd6x -x app.out | sectti
Reading from stdin ...
…
```

- Some scripts also require -g

```
C:\dir>ofd6x -g -x app.out | call_graph
Reading from stdin ...
…
```

Cut Big XML Files Down to Size

- Sometimes the XML is huge
  - 100+ MB has been seen!
- Can make a script very slow
- Each script documents OFD options which reduce XML size
  - Specific to the script

```
OFD OPTIONS  (from sectti.pl documentation)
    Recent releases of OFD support options for filtering the XML
    output down to what is strictly of interest. The best options
    to use in combination with this script are:

      -x --xml_indent=0 --obj_display=none,sections,header
```

Run MAP Scripts

- Create XML version of the map file
- Linker option: `--xml_link_info=`*`file.xml`*
- Supply on command line

```
C:\dir>lib_footprint app_map.xml
...
```

- No need to pipe XML in
  - Linker XML always supplied in a file

Agenda

- Marketing
- How to Run the Scripts
- **Popular Scripts**
- Build Your Own

Popular OFD Scripts

- sectti.pl: Prints info on each section. Also totals by section type. Can output in .csv format for loading into Excel.
- call_graph.pl: Shows function call relationships and stack usage
- bootimage.pl: Creates boot image of .out file
- objdiff.pl: Compares two files or libraries

Popular MAP Scripts

- lib_footprint.pl: Finds all the libraries and reports their sizes
- gen_shm_cmd.pl: Automates sharing MEMORY definitions between processors on OMAP or similar systems

Questions?

Build Your Own

- Remaining slides first prepared for TI Developer's Conference in February 2005
- Updated to current state of the scripts
- These slides are for those interested in:
  - XML
  - How the scripts work
  - Executing directly from Perl
  - Extending a script
  - Writing your own script

Agenda: Build Your Own

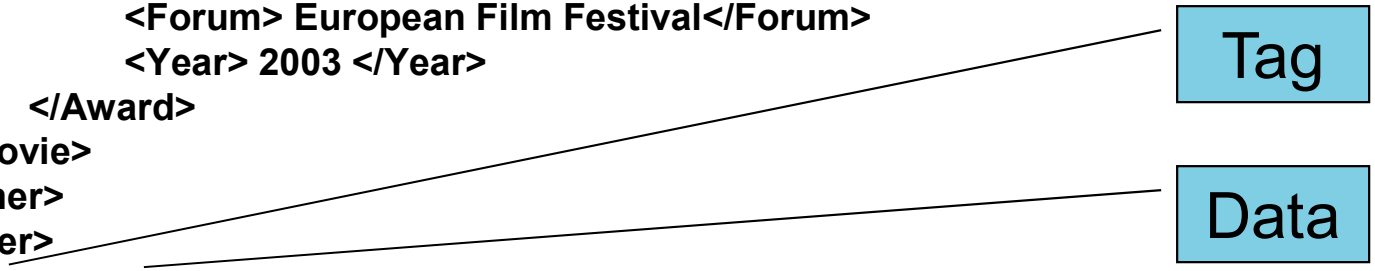
- Introduce XML
- XML Files Generated by TI Tools
  - Linker Map File
  - Object File Display Utility (OFD)
- Examples of How to Write Cool Utilities that Solve Interesting Problems
  - Memory Footprint of Libraries in Application
  - Size of Library Members
  - Creating a Boot Image
- Huge OFD XML Files
- References and Summary

Why XML?

- Standard: http://www.w3.org/XML (link)
  - XML: EXtensible Markup Language
- Simple
  - Text file
  - Well defined structure
- Separates information from its end use
  - Separates Content from Display (unlike HTML)
  - Content is easy to scan and repurpose
- Don't need to write custom XML parsers
  - VB: DOMDocument class (msxml.dll)
  - Perl: XML::Simple, XML::Twig, XML::DOM, etc.

XML Example

```
<?xml version="1.0"?>
<FavouriteMovies>
        <Customer>
         <Name> Jelena </Name>
                <Movie>
                        <Title> Dogville </Title>
                        <Director> Lars Von Trier </ Director >
                        <ProductionYear> 2003 </ProductionYear>
                        <Award >
                                <Forum> European Film Festival</Forum>
                                <Year> 2003 </Year>
                        </Award>
                </Movie>
        </Customer>
        <Customer>
                <Name> George </Name>
                <Movie>
                        <Title> Life of Brian </Title>
                        <Director> Terry Jones </ Director >
                        <ProductionYear> 1979 </ProductionYear>
                </Movie>
        </Customer>
</FavouriteMovies>
```



Agenda: Build Your Own

- Introduce XML
- XML Files Generated by TI Tools
  - **Linker Map File**
  - Object File Display Utility (OFD)
- Examples of How to Write Cool Utilities that Solve Interesting Problems
  - Memory Footprint of Libraries in Application
  - Size of Library Members
  - Creating a Boot Image
- Huge OFD XML Files
- References and Summary

Linker MAP file in XML

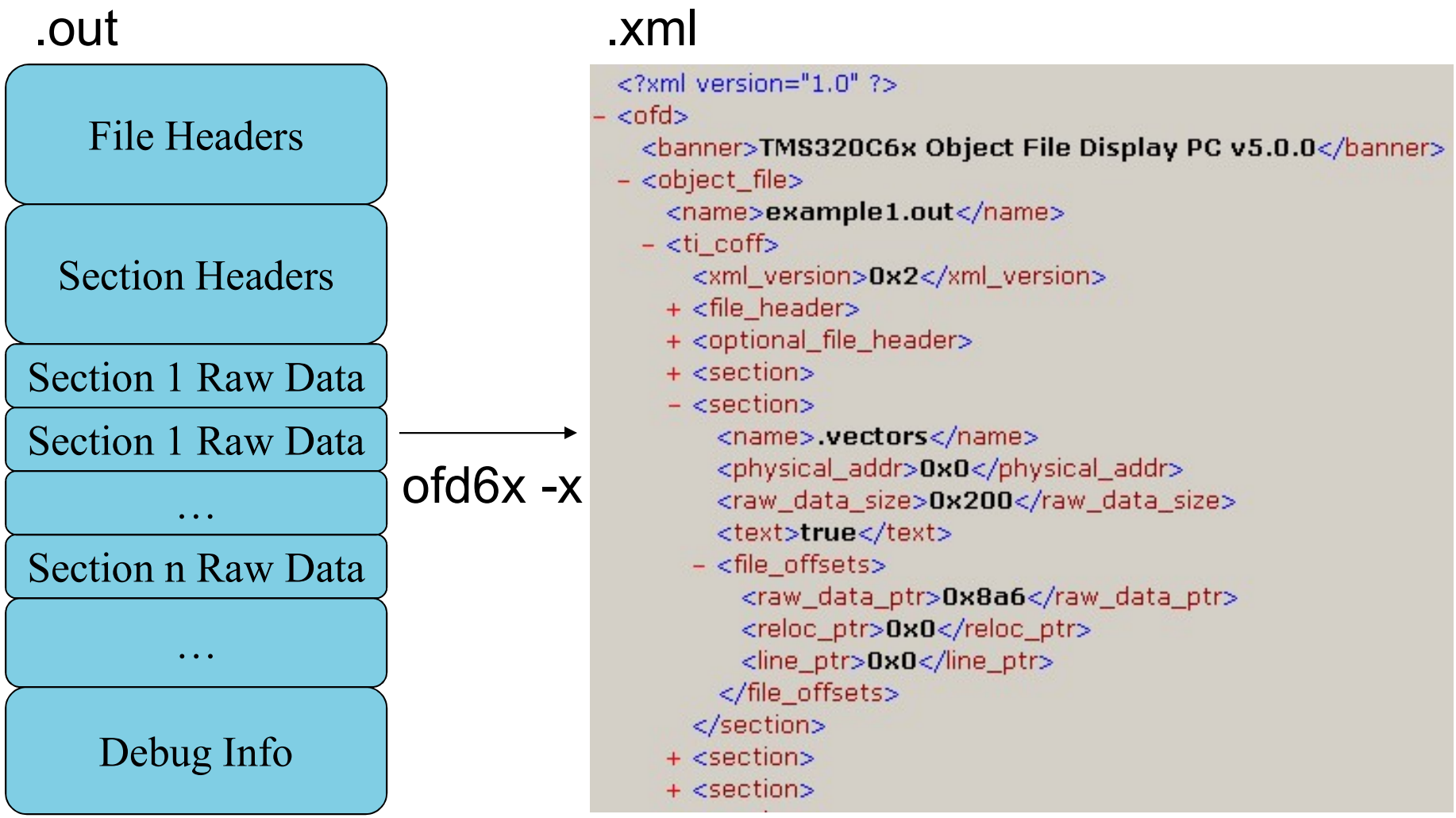
- **Option: *--xml_link_info=filename.xml***
- **Example viewed in Internet Explorer:**

```
<?xml version="1.0" ?>
- <link_info>
    <banner>TMS320C6x COFF Linker PC v5.0.0</banner>
    <copyright>Copyright (c) 1996-2004 Texas Instruments Incorporated</copyright>
    <link_time>0x41d97dee</link_time>
    <output_file>./Debug/app.out</output_file>
  + <entry_point>
  + <input_file_list>
  + <object_component_list>
  + <logical_group_list>
  - <placement_map>
    - <memory_area>
        <name>IRAM</name>
        <page_id>0x0</page_id>
        <origin>0x0</origin>
        <length>0x30000</length>
        <used_space>0x108e2</used_space>
        <unused_space>0x1f71e</unused_space>
        <attributes>RWXI</attributes>
      + <usage_details>
      </memory_area>
    + <memory_area>
    + <memory_area>
    </placement_map>
  + <symbol_table>
  </link_info>
```

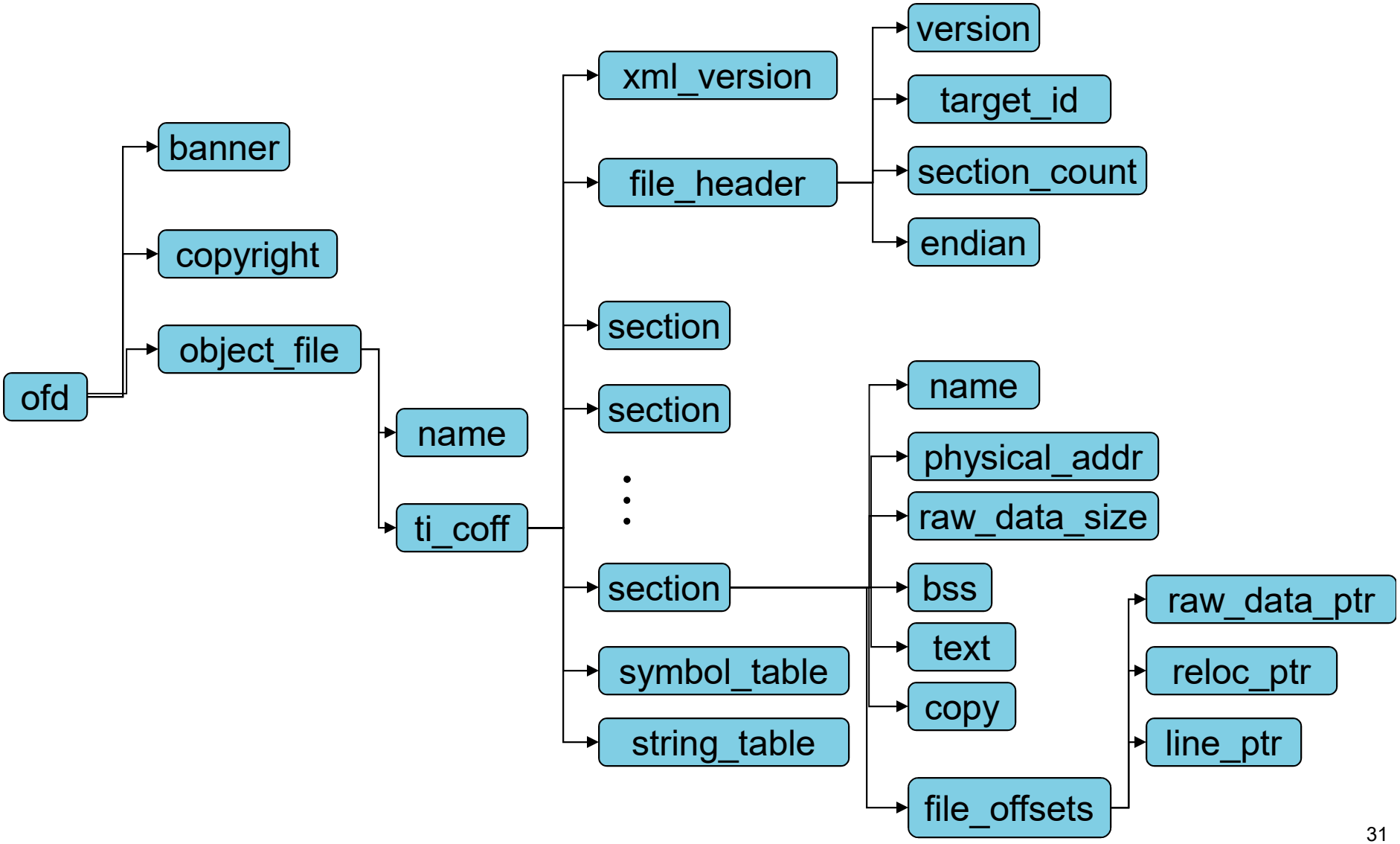
Agenda: Build Your Own

- Introduce XML
- XML Files Generated by TI Tools
  - Linker Map File
  - **Object File Display Utility (OFD)**
- Examples of How to Write Cool Utilities that Solve Interesting Problems
  - Memory Footprint of Libraries in Application
  - Size of Library Members
  - Creating a Boot Image
- Huge OFD XML Files
- References and Summary

Object File Display Utility



Object File Display Utility



Agenda: Build Your Own

- Introduce XML
- XML Files Generated by TI Tools
  - Linker Map File
  - Object File Display Utility (OFD)
- **Examples of How to Write Cool Utilities that Solve Interesting Problems**
  - **Memory Footprint of Libraries in Application**
  - Size of Library Members
  - Creating a Boot Image
- Huge OFD XML Files
- References and Summary

Problem: Memory Used by Libs

- Libraries in your application use how much memory?
- OFD info insufficient for this problem
  - Output section information does not include what parts come from libraries
  - Memory map info only indirect
- Can run OFD on libraries

```
ofd6x -x -o=george_code.xml george_code.lib      # C6000 specific
```

  - Determine size of every member → inaccurate
  - Only want members actually used in app
  - Inspect each library separately → inconvenient

Linker Map File Information

- Linker map file details relationships
  - Files or libraries → Input sections
  - Input sections → Output sections
  - Output sections→ Memory map
- Only includes library members actually used → accurate
- Info on all the libraries is in the map file
- Linker option: `--xml_link_info=file.xml`
  - Due to bug, usage in command file is different
  - `--xml_link_info file.xml /* no '=' */`

XML_TI_MAP.pm

- Perl module that wraps usage of XML::Simple
- Used by all map file XML Perl scripts
- Builds Perl data structure that is:
  - Consistent
  - As flat as possible
  - Easier to use

dump_map.pl

- Perl script that uses XML_TI_MAP.pm
- Reads in XML file and dumps out Perl data structure representation
- Use this script to see the data structure and understand how to traverse it

```
$VAR1 = {
  'banner' => 'TMS320C6x COFF Linker PC v5.0.0',
  'entry_point' => {
    'address' => '0x7b80',
    'name' => '_c_int00'
  },
  'input_file_list' => {
    'fl-1' => {
      'file' => 'rts6400.lib',
      'kind' => 'archive',
      'name' => 'boot.obj',
```

lib_footprint.pl Outline

- In Perl-ish pseudo-code

```
XML_TI_MAP::process_xml_file($xml_file);

foreach $osect (loop through output sections)
{
   $osect_type = classify_section($osect->{'name'});
   foreach $isect (loop through input sections)
   {
      if (came from a library)
      {
         $lib_name = $file_rec->{'file'};
         $lib_data{$lib_name}->{$osect_type} += oct($isect->{'size'});
      }
   }
}

# Print out totals computed in '%lib_data' ...
```

lib_footprint.pl Output

```
algrf.l62
      CODE ----------------------------------------
                    .text:        2336   0x00000920
      DATA ----------------------------------------
                   .cinit:          44   0x0000002c
     UDATA ----------------------------------------
                     .far:          16   0x00000010
                          ==========   ==========
    LIB TOTAL             :        2396   0x0000095c

biosi.a62
      CODE ----------------------------------------
                    .bios:       20352   0x00004f80
                 .sysinit:         224   0x000000e0
                    .text:         512   0x00000200
      DATA ----------------------------------------
                   .cinit:        1400   0x00000578
                   .const:         110   0x0000006e
     UDATA ----------------------------------------
                     .bss:        1096   0x00000448
                          ==========   ==========
    LIB TOTAL             :       23694   0x00005c8e

c6x1x_edma_mcbsp.l67
    …
```

Agenda: Build Your Own

- Introduce XML
- XML Files Generated by TI Tools
  - Linker Map File
  - Object File Display Utility (OFD)
- Examples of How to Write Cool Utilities that Solve Interesting Problems
  - Memory Footprint of Libraries in Application
  - **Size of Library Members**
  - Creating a Boot Image
- Huge OFD XML Files
- References and Summary

Problem: Size of Library Members

- How big is each member in a library?
  - Different from lib memory footprint in application
  - Especially important for library developers
- OFD is perfect for this task
  - No link map file from library members
  - Works on libs built with old codegen tools
- XML_TI_OFD.pm is Perl module for converting OFD XML to Perl data structures
- dump_ofd.pl is script for seeing the Perl data structures

sectti.pl Output

```
================================================================
REPORT FOR LIBRARY: rts6200.lib
================================================================

****************************************************
REPORT FOR FILE: abs.obj
****************************************************
                            Name : Size (dec)  Size (hex)  Type
---------------------------- : ----------  ----------  ----
                     .text:_abs :         32  0x00000020  CODE
                    .text:_labs :         32  0x00000020  CODE
                   .text:_llabs :         64  0x00000040  CODE
... snip ...

----------------------------------------------------------
Totals by section type
----------------------------------------------------------
  Uninitialized Data :       4454  0x00001166
    Initialized Data :       5018  0x0000139a
                Code :     166432  0x00028a20
```

Agenda: Build Your Own

- Introduce XML
- XML Files Generated by TI Tools
  - Linker Map File
  - Object File Display Utility (OFD)
- Examples of How to Write Cool Utilities that Solve Interesting Problems
  - Memory Footprint of Libraries in Application
  - Size of Library Members
  - **Creating a Boot Image**
- Huge OFD XML Files
- References and Summary

Creating a Boot Image: Problem

- Applicable when booting DSP via PCI or HPI
- DSP application (boot image) is part of host application (.c/.h)
- Have: .out file; Need: .c/.h file

```
const unsigned char _vectors[0x200] = {
0x2a, 0x60, 0x46, 0x00, 0x6a, 0x00, 0x00, 0x00, 0x62, 0x03, 0x00,
... }
const unsigned char _text[0x8c00] = {
0xf1, 0x18, 0xbc, 0x0f, 0xf4, 0xd4, 0x3d, 0x06, 0x45, 0x61, 0x7c,
... }
const unsigned char _cinit[0x35c] = {
0x30, 0x02, 0x00, 0x00, 0xd8, 0x9b, 0x00, 0x00, 0x00, 0x00, 0x00,
....}
…
```

Creating a Boot Image: Solution

- **Possible Solutions:**
  - **Write a custom COFF parser**
  - **Manipulate hex6x –a output**
    - **Appendix B of Application Note SPRA512 (link)**
- **Preferred Solution:**
  - **Use OFD's XML output**
  - **Perl script that uses functions in XML_TI_OFD.pm**
  - **bootimage.pl**

Creating a Boot Image: Perl Script

- Convert XML to Perl data structure

```
$xml_data =
  ofd_process_xml_file($xml_file);
```

- For each section

```
foreach $sect
  (ofd_each_section($file_data))
```

  - Find section length

```
$size  = oct($sect->{'raw_data_size'});
```

  - Find file pointer into .out file

```
$ptr   = oct($sect->{'file_offsets'}
      ->{'raw_data_ptr'});
```

  - Determine if it is initialized

```
if   ( (not defined $sect->{'bss'})   &&
       (not defined $sect->{'copy'}))
```

  - If yes, copy from .out file to .c/.h file

```
read(OBJFILE, $buff, $size);
print_c_struct($sectname, $size, $buff)
```

Result: .h/.c files

```
extern const unsigned char _vectors[0x200];
extern const unsigned char _const[0x138];
extern const unsigned char _text[0x8c00];
extern const unsigned char _cinit[0x35c];
```

```
/*********************************************************
** _vectors[0x200]: paddr = 0x00000000 vaddr = 0x00000000
*********************************************************/
const unsigned char _vectors[0x200] = {
0x2a, 0x60, 0x46, 0x00, 0x6a, 0x00, 0x00, 0x00, 0x62, 0x03, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
... }
/*********************************************************
** _text[0x8c00]: paddr = 0x00000200 vaddr = 0x80000000
*********************************************************/
const unsigned char _text[0x8c00] = {
0xf1, 0x18, 0xbc, 0x0f, 0xf4, 0xd4, 0x3d, 0x06, 0x45, 0x61, 0x7c, 0x05, 0xa0,
0x06, 0x10, 0x05, 0x64, 0x02, 0xa8, 0x03, 0x00, 0x00, 0x00, 0x00, 0xf6, 0x42,
0x60, 0x8d, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x60,
0xa5, 0x00, 0x00, 0xd8, 0x97, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
... }
/*********************************************************
** _cinit[0x35c]: paddr = 0x00009220 vaddr = 0x00009220
*********************************************************/
const unsigned char _cinit[0x35c] = {
0x30, 0x02, 0x00, 0x00, 0xd8, 0x9b, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
....}
```

Agenda: Build Your Own

- Introduce XML
- XML Files Generated by TI Tools
  - Linker Map File
  - Object File Display Utility (OFD)
- Examples of How to Write Cool Utilities that Solve Interesting Problems
  - Memory Footprint of Libraries in Application
  - Size of Library Members
  - Creating a Boot Image
- **Huge OFD XML Files**
- References and Summary

Huge OFD XML Files

- OFD may create huge XML files
  - Especially when using –g to see Dwarf info
  - `ofd6x -g -x rts6400.lib` creates 40 MB file!
  - Without `-g` file is 8 MB
  - Scripts that require –g info
    - call_graph.pl
    - global_types_gen.pl
    - func_info.pl
- Naive processing of such large files can take a long time, if it works at all

OFD XML Filtering Options

- OFD options can vastly reduce amount of XML output
- Use –h option to see option summary

OFD XML Filtering Options

- Optimal filtering options vary by script
- Details in perldoc for script …

```
C:\>perldoc sectti.pl
...snip...

OFD OPTIONS
    Recent releases of OFD support options for filtering the XML
    output down to what is strictly of interest. The best options to
    use in combination with this script are:

      -x --xml_indent=0 --obj_display=none,sections,header

    Filtering the XML in this way reduces the amount of data
    processed by this script, thus making it run faster.
...snip...

C:\>ofd6x -o=rts6200.xml -x --xml_indent=0
--obj_display=none,sections,header rts6200.lib
C:\>sectti.pl rts6200.xml
```

More XML Filtering in .pm

- Use these functions in XML_TI_OFD.pm to further filter XML before it is parsed
- ofd_filter_xml – Specify what to keep
- ofd_strip_xml – Specify what to delete
- ofd_filter_and_parse_xml
  - Filters and parses XML in a way that reduces overall script memory usage
- Using these functions requires good knowledge of ELF and DWARF structure
  - Work from existing examples

Agenda: Build Your Own

- Introduce XML
- XML Files Generated by TI Tools
  - Linker Map File
  - Object File Display Utility (OFD)
- Examples of How to Write Cool Utilities that Solve Interesting Problems
  - Memory Footprint of Libraries in Application
  - Size of Library Members
  - Creating a Boot Image
- Huge OFD XML Files
- **References and Summary**

References

- Perl Cookbook, 2nd edition, Chapter 22
  - Excellent summary
- http://www.w3schools.com (link)
  - Good tutorials on XML and other web technologies
- http://msdn.microsoft.com/XML (link)
  - For fans of VB

Summary

- Introduced XML
- XML files produced by TI tools
  - Object File Display Utility
  - Linker Map File
- What information is available in each
- Cool Utilities based on XML processing
  - Examples
  - Modules to build on
- So go write your own!

Questions?

Backup Slides

Other Approaches

- Many alternatives to Perl and XML::Simple
- Other Perl modules
  - More powerful
  - Steeper learning curve
  - XML::Twig, XML::DOM, XML::LibXML
- Visual Basic based on msxml.dll
- Java used in the first SAX parsers
  - SAX: Simple Access to XML
- Start with the language you know best
- Do not write your own XML parser!

What Version of Tools Built Lib?

- How to see the version of the compiler tools used to build a library …
- Use OFD with –g ➔ exposes debug info

```
% ofd6x –g rts6400.lib | grep DW_AT_producer                    # Unix
 DW_AT_producer TMS320C6x C/C++ Codegen Unix v5.0.0 Copyright …
 DW_AT_producer TMS320C6x C/C++ Codegen Unix v5.0.0 Copyright …
 …
```

- Use findstr on Windows

```
C:\> ofd6x –g rts6400.lib | findstr DW_AT_producer
```