ISBN 0-7714-2175-3

A C++ to XML translator

Yannis Chicha, Florence Defaix
and Stephen M. Watt

Technical Report # 536

Department of Computer Science
The University of Western Ontario
London, Canada
N6A 5B7

A C++to XML translator

Yannis Chicha, Florence Defaix & Stephen M. Watt
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Ontario, Canada N6A 5B7
{chicha,fdefaix,watt}@csd.uwo.ca

C++ 2 XML

Contents

Chapter 1

Introduction

Often one would like an abstract representation for a given C++ source code. But C++ is a complex language, not easy to parse.
XML (eXtended Markup Language) is a metalanguage that lets the user build his own Markup Language in order to represent any electronic document. It is very regular and easily parsed. It is also a subset of SGML (Standard Generalized Markup Language) a more sophisticated Markup Language).

C++2xml is a tool based on gcc that generates an XML representation from a C++ source file.In the current version the bodies of functions are not treated. This is usually what is wanted in working with header files.
The first idea was to write a C++ header parser from scratch. It has been very soon forgotten because C++ accepts almost anything to be compiled !
In a second thought, we looked for a public domain parser. We found several (eg Roskind's, Parr's, ...) but none of them was fully functional for recent definitions of C++.
Finally we chose another solution: to use GNU C++ compiler. Although this compiler isn't among the easiest to study and modify, it has a great advantage: it is complete and is used extensively. Currently, we work with Gcc V2.8.0.

Gnu Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include

anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Part I

User Guide

Chapter 2

Installation

- In order to install Cpp2XML Version 1.3 you need the gcc-2.8.0 source archive (gcc-2.8.0.tar.gz). You can also use an installed version of gcc-2.8.0 as cpp2Xml won't change the behaviour comportement of gcc when the new options are not set.
  Note: if you gave special flags to gcc during the install, we recommend that you use a new version or that you modify the cpp2xml Makefile to insert your flags.

  If you want to use a new version of gcc, decompress it now:
  ```
  tar -xvzf gcc-2.8.0.tar.gz
  ```
  or (if your tar doesn't handle -z)
  ```
  gunzip gcc-2.8.0.tar.gz | tar xvf -
  ```

- Then decompress in the same way cpp2xml-1.3.tar.gz

- Edit cpp2xml-1.3/Makefile and fill the three variables GCC_SOURCES_DIR and GCC_INSTALL_DIR and INSTALL_DIR.
  GCC_SOURCES_DIR is the directory where you just uncompressed gcc (or the source directory of your installed gcc v2.8)
  GCC_INSTALL_DIR is the directory where the executables and documentation will go. (the same as in –prefix for a previous installation of gcc). This could be a non-standard place, if you want to keep an older version of g++ as the default one.
  INSTALL_DIR is the directory where the script **cpp2xml** will be installed. The latter is a script that calls the modified version of gcc. Once again If you don't want the new g++ as your default, set INSTALL_DIR to a different location than GCC_INSTALL_DIR. Thus you can have cpp2xml in your PATH but not the new g++.

- Run the installation

  ```
  make
  ```

- Add INSTALL_DIR to your PATH Add GCC_INSTALL_DIR to your PATH **ONLY** if you want the new g++ as your default compiler.

  That's it!

  Note: If you want to remove the new functionalities, run `make restore_gcc`.

Chapter 3

Using cpp2xml

3.1 Full syntax

The full syntax for the generation of XML is the following:

```
$ cpp2xml  [-o outfile] [-split] [-stdDir standard.lst] [-treeIncludes] infile
```

-o outfile The XML generation will be done the specified file instead of taking the basename of the input file and appending .xml to it.

-split With this option, the (unique) generated XML file is completed with a new tag called "Change Of File". When the source contains "#include" directives - classes, variables and functions are defined in several files - it helps to know in which file was defined such or such item.

-stdDir standard.lst We added an option to allow the user to specify where standard header files are located in order to do a special treatment with them. In fact we print only global scope type declaration names, since that is what is wanted in one of our applications. The -stdDir option needs another parameter which is the name of a file containing the list of directory prefixes where standard header files are located.

-treeIncludes Generates a file named *infile.inc* which contains an XML representation of the hierarchy of header files included in *infile*.

-GenXML `cpp2xml` is an alias for `g++ -GenXML`.
Note: you don't need this option when using the command cpp2xml.
This is the main option which was added to gcc. If it is not set, gcc works as usual... When this option is set, gcc generate the XML represention and no other file.

3.2 Examples

1. **cpp2xml foo.C**
   generates foo.xml from headers in foo.C
2. **cpp2xml -o dummy.xml foo.C**
   generates dummy.xml from headers in foo.C
3. **cpp2xml -split -stdDir excludes.lst**
   generates in split mode, "#include" files listed in exclude.lst are handled in a different way (see 3.1)
4. **cpp2xml -treeIncludes foo.cpp**
   generates foo.C.inc containing file inclusions

3.3 Syntax excludes.lst

This file, used with the **-stdDir** option, contains a list of prefixes to handle as standard files.

For example, with the following content:

```
/usr/include
/usr/james/examples/includes
```

These files would be handled as standard:

```
/usr/include/stdio.h
/usr/include/sys/time.h
/usr/include_foo/dummy.h
/usr/james/examples/includes/example.h
```

But not the followings:

```
/usr/local/include/tk.h
/usr/james/examples/include/example2.h
```

3.4 A small output

Given the following C++ header (in dummy.H):

```
class Complex {
  int real_part;
  int im_part;
public:
  Complex(int i, int j);
  int re() { return real_part; }
  int im() { return im_part; }
  virtual void print();
};
```

And the file dummy.C:

```
#include "dummy.H"
```

The command `g++ -GenXML -split dummy.C` will produce dummy.xml whose content is the following:

```
<class_decl>
   <name uniq="Complex">Complex</name>
   <protection><public/></protection><concrete/>
   <parents></parents>
   <templates></templates>
   <var_decl>
      <name uniq="real_part">real_part</name>
      <simple_type><int/></simple_type>
      <initial></initial>
      <protection><private/></protection>
      <modifiers></modifiers>
   </var_decl>
   <var_decl>
      <name uniq="im_part">im_part</name>
      <simple_type><int/></simple_type>
      <initial></initial>
      <protection><private/></protection>
      <modifiers></modifiers>
   </var_decl>
   <function_decl>
      <name uniq="__7Complexii">Complex</name>
      <not_operator/>
      <templates></templates>
      <pointer_type>
        <record_type>
          <name uniq="7Complex">Complex</name>
        </record_type>
      </pointer_type>
      <arg-types>
      <param>
         <name uniq="parm0">parm0</name>
         <simple_type><int/></simple_type>
         <initial></initial>
         <modifiers></modifiers>
      </param>
      <param>
         <name uniq="parm1">parm1</name>
         <simple_type><int/></simple_type>
         <initial></initial>
         <modifiers></modifiers>
      </param>
      </arg-types>
      <protection><public/></protection>
      <modifiers><method/></modifiers>
   </function_decl>
   <function_decl>
      <name uniq="re__7Complex">re</name>
      <not_operator/>
      <templates></templates>
      <simple_type><int/></simple_type>
      <arg-types>
      </arg-types>
      <protection><public/></protection>
      <modifiers><method/></modifiers>
   </function_decl>
   <function_decl>
      <name uniq="im__7Complex">im</name>
      <not_operator/>
      <templates></templates>
      <simple_type><int/></simple_type>
      <arg-types>
      </arg-types>
      <protection><public/></protection>
      <modifiers><method/></modifiers>
   </function_decl>
   <function_decl>
      <name uniq="print__7Complex">print</name>
      <not_operator/>
      <templates></templates>
      <simple_type><void/></simple_type>
      <arg-types>
      </arg-types>
      <protection><public/></protection>
      <modifiers><virtual/><method/></modifiers>
   </function_decl>
</class_decl>
```

Chapter 4

Current limitations and further developments

Here is a list of the current limitations. They could be developed in the future depending of the needs...

- Only headers are handled (no body)
- Macros are not generated
- Function pointers are not detailed
- Init values are simplified

Chapter 5

XML representation

```
<!--

  cpp2xml.dtd     XML syntax used in by g++ -XML

-->

<!-- Entities -->
<!ENTITY % doctype       "cpp2xml"  -- C++ to XML -->

<!--   Declarations -->
<!ENTITY % vdecl              "(var_decl | field_decl)"                          >
<!ENTITY % classesdecl        "(class_decl | struct_decl | union_decl)"          >

<!ENTITY % declaration        "(%vdecl | type_decl | %classesdecl | enum_decl
                               | function_decl |change_of_file)"                 >

<!--   Types entities --                                                        >
<!ENTITY % shorttype          "(simple_type | array_type | record_type
                               | union_type | enumeral_type
                               | pointer_type | reference_type
                               | template_type_parm | template_record_type)"     >

<!ENTITY % inttype            "(int | unsigned_int | long_int | long_unsigned_int
                               | long_long_int | long_long_unsigned_int
                               | short_int | short_unsigned_int)"                >
<!ENTITY % chartype           "(char | unsigned_char | signed_char)"             >
<!ENTITY % returntype         "(method_type | function_type)"                    >


<!--   Misc --                                                                  >
<!ENTITY % classheader        "(name, protection, parents, templates)"          >
<!ENTITY % protections        "(public | protected | private)"                   >
<!ENTITY % operator_or_not    "(not_operator | operator)"                       >
<!ENTITY % action             "(enter | leave)"                                 >
<!ENTITY % abstraction        "(concrete | abstract)"                           >
```

```
<!-- Elements -->

<!ELEMENT %doctype              - -  (%declaration)*                                          >

<!--   High level elements --                                                                >
<!ELEMENT %vdecl                - -  (name, %shorttype, initial, protection, modifiers )     >
<!ELEMENT parm_decl             - -  (name, %shorttype, modifiers , initial )                >
<!ELEMENT type_decl             - -  (name, %shorttype? )                                    >
<!ELEMENT %classesdecl          - -  (%classheader, (%declaration)* )                        >
<!ELEMENT function_decl         - -  (name, %operator_or_not, templates, %returntype,
                                     arg-types, protection, modifiers )                      >
<!ELEMENT enum_decl             - -  (name,(item)*)                                          >
<!ELEMENT change_of_file        - -  (%action, cof_name)                                     >
<!--   Types --                                                                              >
<!ELEMENT simple_type           - -  (void | %inttype | %chartype | bool | float |double)    >
<!ELEMENT array_type            - -  (domain,%shorttype)                                     >
<!ELEMENT template_record_type  - -  (name, (templatesparams)+)                              >
<!ELEMENT pointer_type          - -  (%shorttype)                                            >
<!ELEMENT reference_type        - -  (%shorttype)                                            >
<!ELEMENT template_type         - -  (%shorttype)                                            >
<!ELEMENT enumeral_type         - -  (name)                                                  >
<!ELEMENT record_type           - -  (name)                                                  >
<!ELEMENT union_type            - -  (name)                                                  >
<!ELEMENT enumeral_type         - -  (name)                                                  >
<!ELEMENT template_type_parm    - -  (name)                                                  >


<!--   Class header --                                                                       >
<!ELEMENT parents               - -  (parent)*                                               >
<!ELEMENT parent                - -  (name, protection, modifiers)                           >
<!ELEMENT templates             - -  (templates)*                                            >
<!ELEMENT template              - -  (type_decl | parm_decl)                                 >

<!--   Constantes --                                                                         >
<!ELEMENT integer_cst            - -  CDATA                                                  >
<!ELEMENT char_cst               - -  CDATA                                                  >
<!ELEMENT float_cst              - -  CDATA                                                  >
<!ELEMENT string_cst             - -  CDATA                                                  >
<!ELEMENT enumeral_cst           - -  CDATA                                                  >

<!--   Misc --                                                                               >
<!ELEMENT name                  - -  CDATA                                                   >
<!ATTLIST name                        uniq  CDATA  #REQUIRED                                 >

<!ELEMENT cof_name              - -  CDATA                                                   >
<!ATTLIST cof_name                    old   CDATA  #REQUIRED                                 >

<!ELEMENT protection            - -  %protections                                            >
<!ELEMENT modifiers             - -  (abstract | virtual | readonly | static
                                      | unsigned | external)* >
<!ELEMENT %returntype           - -  (%shorttype)                                            >
<!ELEMENT arg-types             - -  ((param)*, ellipsis?)                                   >
<!ELEMENT param                 - -  (name, %shorttype, initial, modifiers)                  >
```

```
<!ELEMENT item                - -  (CDATA)                                  >
<!ELEMENT initial             - -  CDATA                                    >
<!ELEMENT domain              - -  CDATA                                    >
<!ELEMENT templates_params    - -  (template_parm | template_type)*         >
<!ELEMENT template_parm       - -   (integer_cst | enumeral_cst)            >



<!-- Short tags --                                                          >

<!--    modifiers --                                                        >
<!ELEMENT abstract              - O EMPTY                                   >
<!ELEMENT virtual               - O EMPTY                                   >
<!ELEMENT readonly              - O EMPTY                                   >
<!ELEMENT static                - O EMPTY                                   >
<!ELEMENT unsigned              - O EMPTY                                   >
<!ELEMENT method                - O EMPTY                                   >
<!ELEMENT function              - O EMPTY                                   >

<!--    simple types  --                                                    >
<!ELEMENT void                  - O EMPTY                                   >

<!ELEMENT int                   - O EMPTY                                   >
<!ELEMENT unsigned_int          - O EMPTY                                   >
<!ELEMENT long_int              - O EMPTY                                   >
<!ELEMENT long_unsigned_int     - O EMPTY                                   >
<!ELEMENT long_long_int         - O EMPTY                                   >
<!ELEMENT long_long_unsigned_int - O EMPTY                                  >
<!ELEMENT short_int             - O EMPTY                                   >
<!ELEMENT short_unsigned_int    - O EMPTY                                   >

<!ELEMENT char                  - O EMPTY                                   >
<!ELEMENT unsigned_char         - O EMPTY                                   >
<!ELEMENT signed_char           - O EMPTY                                   >

<!ELEMENT bool                  - O EMPTY                                   >
<!ELEMENT float                 - O EMPTY                                   >
<!ELEMENT double                - O EMPTY                                   >

<!--    Function misc -->
<!ELEMENT declaration           - O EMPTY                                   >
<!ELEMENT definition            - O EMPTY                                   >
<!ELEMENT operator              - O EMPTY                                   >
<!ELEMENT not_operator          - O EMPTY                                   >
<!ELEMENT ellipsis              - O EMPTY                                   >

<!--    Class misc -->
<!ELEMENT concrete              - O EMPTY                                   >
<!ELEMENT abstact               - O EMPTY                                   >

<!--   ChangeOfFile --                                                      >
<!ELEMENT enter                 - O EMPTY                                   >
<!ELEMENT leave                 - O EMPTY                                   >
```

Part II

Implementation Reference

Chapter 6

C++ to abstract representation

6.1 How it works

This application is an add-on to gcc and is intended to generate an internal abstract representation of a C++ source code.
gcc compiles source codes in 4 phases: preprocessing, parsing, compilation and linkage.
During the parsing phase, gcc stores information about the source code in abstract trees structures. For example a function header is a tree with sub-trees for name, return type, parameters, context of execution... In the same way, a parameter is a tree with name, type, init value,... We use these trees to generate one unique and clean tree containing all the needed information.

gcc provides a full set of macros to access fields in its nodes. The main function (cpp2tree) takes a gcc tree and builds a part of the global tree.
In some cases, we need several gcc trees to build one single declaration representation.
For example, if the node is a class definition, we should store:

- the class header : name, parents, templates
- the data members with their type, name and protection
- the methods with their name, parameters, protection
- any static functions with their name, parameters, protection
- any static variables with their name, types, init values and protection
- any inner-classes recursively with their fields, methods and inner-classes...

In fact inner-classes are given in separate gcc trees. So we need to re-integrate them in the correct level of imbrication.

The main difficulties are to:

- handle the correct amount of information at each level so that all the information is handled yet only once. For example if we handle a parameter of type A we don't need to know everything which is in A. Often nodes are chained in several ways, we have to choose the best one...
- find good entry points. A good entry point is a moment in the parsing when the trees are entirely built and not already destroyed! More, we have to find entry points which don't overlap as every declaration has to appear yet only once. So there are 4 entry points :

  - one for class declaration with their header, fields and methods
  - one for external variable, function and type declarations
  - one for external enumeral types
  - one for external function definitions (needed if there is no declaration)
- handle only what appears in the source files.
  - no internal functions used by gcc
  - no "this" parameters in methods

6.2 Data structures explanation

6.2.1 Our tree representation

At top level, this internal representation contains nodes each representing a C++ declaration or definition with their characteristics. We have several kinds of top-level nodes:

- function/methods
- class/struct/union
- variable/field
- enumerated type

Each of them has more or less information such as the name, the type, a list of items.
If we take a function declaration for example, the cell contains:

- its name
- whether it's a method or a function
- whether it's an operator or not
- the return type
- the list of parameters with their types and initial value

The type representation

The type of a declaration can be very simple (ex: int) or much more complex (ex: `foo<dummy> *`). That's why we represent any type with a list of cells each containing an atomic part of the type. Here is our structure to represent an atomic part of the type.

```
typedef struct {
  Type     t;            /* the kind of type ex: pointer type, record type, simple type....
  BaseType bt;           /* if t == simpleType the name of the type   ex: int, bool
                         /* if t == Template Type the kind of type    ex: instantiated class templat
  Identifier *id  ;      /* name and uniq name for some types
  int      array_domain; /* domain of arrays  ex: 10 in  a[10]
  int      qualif;       /* modifiers : static, const, volatile, ...
  List     *tt;          /* list of templates ex: T and n in <class T, int n>
} CppType;
```

Let's take an example. If we decompose the type **char[40]**, the tree representation will be:

```
     ---------------------          ------------------------
    /                       \      /                        \
   |  t            = ARRAY |     |  t            = SIMPLE   |
   |  bt           = /     |     |  bt           = CPP_CHAR |
==>|  id           = /     | ==> |  id           = "char"   |
   |  array_domain = 40    |     |  array_domain = /        |
   |  qualif       = /     |     |  qualif       = /        |
   |  tt           = /     |     |  tt           = /        |
    \_____________________/       \________________________/
```

6.2.2 Gcc Nodes representation

During the parsing pass, gcc stores information about the source code in abstract trees structures. A tree node can represent a data type, a variable, an expression or a statement. In 'C++ to XML', as we don't generate a representation of the bodies, we only need the first two kinds of nodes.

Information in nodes are accessed through a full set of macros. For example, TREE_TYPE(node) will return a node containing the type info of node. In the same way TREE_PRIVATE(node) will be set when node's protection is "private".

Even if the fields are never accessed directly but always via macros, it can be interesting to know the internal structure of the tree nodes.
The low-level type of a tree node is a union containing more or less infos depending of the node. In any case, there is a minimum of common infos contained in the *common* field (see below). Then there are more or less specific fields. Here is the low-level representation with the corresponding set of macros.

```
typedef union tree_node *tree;
union tree_node{
  struct tree_common common;
  struct tree_int_cst int_cst;
  struct tree_real_cst real_cst;
  struct tree_string string;
  struct tree_complex complex;
  struct tree_identifier identifier;
  struct tree_decl decl;
  struct tree_type type;
  struct tree_list list;
  struct tree_vec vec;
  struct tree_exp exp;
  struct tree_block block;
 };
struct tree_int_cst{
  char common[sizeof (struct tree_common)];
  struct rtx_def *rtl;              Not used in cpp2xml
  HOST_WIDE_INT int_cst_low;        TREE_INT_CST_LOW (node)
  HOST_WIDE_INT int_cst_high;       TREE_INT_CST_HIGH (node)
};
```

The tree_common structure

This is the common part to all nodes. Three of the fields are unavoidable: chain, type and code.
chain:
Often, nodes are chained together. First of all, lists (TREE_LIST) are chained.
Then declarations in the same scope are chained together.
At last types are chained.
type:
This is the data type in all nodes that are expressions.
In POINTER_TYPE nodes, this is the type that the pointer points to.
In ARRAY_TYPE nodes, this is the type of the elements.
code:
This is an enum code specifying the kind of the node (FUNCTION_DECL, INTEGER_TYPE, ...)

The following table presents all the fields in the tree_common structure. The second column shows which macro to use to access the fields, **node** beeing a **tree***.

Rem: Sometimes a low-level field is used for multiple purposes depending of the kind of node we are in. **That's why it is important to use access macros all the time.**

Fields Names	Accessor Macro	In which kind of Nodes
`union tree_node *chain;`	`TREE_CHAIN(node)`	all nodes
`union tree_node *type;`	`TREE_TYPE(node)`	all nodes
`enum tree_code code : 8;`	`TREE_CODE(node)`	all nodes
`unsigned side_effects_flag : 1;`	`TREE_SIDE_EFFECTS(node)`	all expressions
`unsigned constant_flag : 1;`	`TREE_CONSTANT(node)`	all expressions
`unsigned permanent_flag : 1;`	`TREE_PERMANENT(node)`	all nodes
`unsigned addressable_flag : 1;`	`TREE_ADDRESSABLE(node)`	`VAR_DECL, FUNCTION_DECL, CONSTRUCTOR, LABEL_DECL, ..._TYPE, IDENTIFIER_NODE`
`unsigned volatile_flag : 1;`	`TREE_THIS_VOLATILE(node)` `TYPE_VOLATILE(node)`	all expressions `..._TYPE`

`unsigned readonly_flag : 1;`	`TREE_READONLY(node)`
	
	`ITERATOR_BOUND_P(node)`
	`TYPE_READONLY(node)`
		`VAR_DECL, PARM_DECL, FIELD_DECL, ..._REF`
		`VAR_DECL if iterator (C)`
		`..._TYPE`

`unsigned unsigned_flag : 1;`	`TREE_UNSIGNED(node)`
	
	`DECL_BUILT_IN_NONANSI(node)`
	`TREE_PARMLIST(node)`
	`SAVE_EXPR_NOPLACEHOLDER(node)`
		`INTEGER_TYPE, ENUMERAL_TYPE, FIELD_DECL`
		`FUNCTION_DECL`
		`TREE_PARM_LIST`
		`SAVE_EXPR`

`unsigned asm_written_flag: 1;`	`TREE_ASM_WRITTEN(node)`	`VAR_DECL, FUNCTION_DECL, RECORD_TYPE, UNION_TYPE, QUAL_UNION_TYPE, BLOCK`
`unsigned used_flag : 1;`	`TREE_USED(node)`	`expressions,IDENTIFIER_NODE`
`unsigned raises_flag : 1;`	`TREE_RAISES(node)`	expressions

`unsigned static_flag : 1;`	`TREE_STATIC(node)`
	
	`TREE_VIA_VIRTUAL(node)`
	`TREE_CONSTANT_OVERFLOW(node)`
	`TREE_NO_UNUSED_WARNING(node)`
	
	`TREE_SYMBOL_REFERENCED(node)`
		`VAR_DECL, FUNCTION_DECL, CONSTRUCTOR`
		`TREE_LIST and TREE_VEC`
		`..._CST`
		`CONVERT_EXPR, NOP_EXPR, COMPOUND_EXPR`
		`IDENTIFIER_NODE`

`unsigned public_flag : 1;`	`TREE_PUBLIC(node)`
	`TREE_OVERFLOW(node)`
	`TREE_VIA_PUBLIC`
		`VAR_DECL and FUNCTION_DECL`
		`..._CST`
		`TREE_LIST and TREE_VEC`

`unsigned private_flag : 1;`	`TREE_VIA_PRIVATE(node)` `TREE_PRIVATE(node)`	`TREE_LIST or TREE_VEC` unspecified nodes
`unsigned protected_flag : 1;`	`TREE_VIA_PROTECTED(node)` `TREE_PROTECTED(node)`	`TREE_LIST` `BLOCK, unspecified nodes`

`unsigned lang_flag_0 : 1;`
`unsigned lang_flag_1 : 1;`
`unsigned lang_flag_2 : 1;`
`unsigned lang_flag_3 : 1;`
`unsigned lang_flag_4 : 1;`
`unsigned lang_flag_5 : 1;`
`unsigned lang_flag_6 : 1;`
	`TREE_LANG_FLAG_0(node)`
	`TREE_LANG_FLAG_1(node)`
	`TREE_LANG_FLAG_2(node)`
	`TREE_LANG_FLAG_3(node)`
	`TREE_LANG_FLAG_4(node)`
	`TREE_LANG_FLAG_5(node)`
	`TREE_LANG_FLAG_6(node)`	

Other useful macros

Here are some very useful macros.
It is not intended to be an exaustive list. (for a full listing, see tree.def and tree.h)

- TREE_CODE_CLASS(TREE_CODE (node))
  This is a one letter code telling in which category the node belongs. It can be one of the following:

```
"x" for an exceptional code (fits no category).
"t" for a type object code.
"b" for a lexical block.
"c" for codes for constants.
"d" for codes for declarations (also serving as variable refs).
"r" for codes for references to storage.
"<" for codes for comparison expressions.
"1" for codes for unary arithmetic expressions.
"2" for codes for binary arithmetic expressions.
"s" for codes for expressions with inherent side effects.
"e" for codes for other kinds of expressions.
```

- DECL_NAME(node)
  This is the name of the object as written by the user. It is an IDENTIFIER_NODE.

- DECL_ASSEMBLER_NAME(node)
  This is the unique name of the object as the assembler will see it. It is very useful for overloaded function as we go from C++ to Aldor via C. (C doesn't handle overloading).

Misc

Although I said to use macros all the time, there is one case when you have to use directly a data structure.
Rem: This is only because there is no standard macro doing it, but the macro could be created...
So in order to have a human readable version of a node's TREE_CODE, you have to use:
`tree_code_name[(int) TREE_CODE (node)]`.
It is exactly the string translation of the enum name (ex: "var_decl" for VAR_DECL).

6.2.3 Gcc fields used in the tree generation

This paragraph's purpose is to list which macro to use in order to access such or such characteristic of a C++ declaration.

```
class, struct and union
  -- class header
  protection      TREE_PURPOSE(node)  (access_public_node, access_protected_node
                                       or  access_private_node)
    uniq name     IDENTIFIER_POINTER(DECL_ASSEMBLER_NAME (TREE_VALUE (node))
    name          IDENTIFIER_POINTER(DECL_NAME (TREE_VALUE (node))
    kind          CLASSTYPE_DECLARED_CLASS(TREE_TYPE(class))  -> class
                  TREE_CODE(TREE_TYPE(class))                 -> union or struct
    parents       (only for class and structs)

 i = TYPE_BINFO_BASETYPES (node)      : TREE_VEC

                    p = TREE_VEC_ELT(1, 0 .. TREE_VEC_LENGTH(i) )  : TREE_VEC
                        parent              TREE_TYPE(p)           : RECORD_TYPE
                        protection          TREE_PUBLIC(p)         true => public
                        virtual inheritance TREE_VIA_VIRTUAL(p)    true => virtual

    templates     if CLASSTYPE_TEMPLATE_INFO(node)
                  i = TREE_VALUE (DECL_ARGUMENTS(CLASSTYPE_TI_TEMPLATE(node)))
                  TREE_VEC:
                     t = TREE_VEC_ELT(i, 0 .. TREE_VEC_LENGTH(i) )
                         => tree_decl or parm_decl
                          type_decl:
                                name            IDENTIFIER_POINTER(DECL_NAME(t))
                                uniq name       IDENTIFIER_POINTER(DECL_ASSEMBLER_NAME(t))
                                type            TREE_TYPE(t)
                                modifiers       /
                                initial         /
                           parm_decl:
                                name            IDENTIFIER_POINTER(DECL_NAME(t))
                                uniq name       IDENTIFIER_POINTER(DECL_ASSEMBLER_NAME(t))
                                simple_type     TREE_TYPE(t)
                                modifiers       const
                                initial         DECL_INITIAL(t) : INTEGER_CST


  -- data_members and methods by protection are chained  TREE_CHAIN(node) ,
  -- TREE_CHAIN(TREE_CHAIN(node)) ...
  protection      TREE_PURPOSE(node)
  list of decls   TREE_VALUE(node)
                  TREE_LIST:
                    TREE_VALUE : field_decl, function_decl ...
                    TREE_CHAIN : next in this protection


function
    uniq name                       IDENTIFIER_POINTER(DECL_ASSEMBLER_NAME (node)
    name                            IDENTIFIER_POINTER(DECL_NAME (node)
    constructor                     if DECL_CONSTRUCTOR_P (node)
    ~declaration (vs definition)    if DECL_EXTERNAL(node)      (no implementation here)
    operator                        by checking the name in optable defined in cplus-dem.c
```

```
templates                       i =. TREE_VALUE (DECL_ARGUMENTS (DECL_TI_TEMPLATE(node))
                                TREE_VEC:
                                t = TREE_VEC_ELT(i, 0 .. TREE_VEC_LENGTH(i) )
                                        => tree_decl or parm_decl
return type :                   ft = TREE_TYPE (node) => any type
parameters                      p = TYPE_ARG_TYPES(ft)
                                type            TREE_VALUE (p)
                                init value      TREE_PURPOSE (p)
                                next parameter p = TREE_CHAIN (p)
                                rem:  1 - additional 'int' first param when virtual inheritance
                                      2 - this parameter, first param for methods
                                      3 - additional 'void' last param if not ellipsis
                                rem:  parameters names are located in DECL_ARGUMENTS(node)

-------------------------------------------------------------------------------
var decl and field decl:
  name        IDENTIFIER_POINTER(DECL_NAME (node))
  uniq name   IDENTIFIER_POINTER(DECL_ASSEMBLER_NAME (node))
  type        TREE_TYPE(node)
  initial     DECL_INITIAL(node)
-------------------------------------------------------------------------------
enum: TREE_CODE == ENUMERAL TYPE
        name                    IDENTIFIER_POINTER(DECL_NAME(TYPE_NAME(node))))
        uniq name               IDENTIFIER_POINTER(DECL_ASSEMBLER_NAME (TYPE_NAME (node)))))
        domain of indices       min: TYPE_MIN_VALUE (node)
                                max: TYPE_MAX_VALUE (node)
        elements                TREE_LIST in  l := TYPE_VALUES(node)
                                   name   TREE_PURPOSE(l)
                                   value  TREE_VALUE(l)    (an INTEGER_CST)

-------------------------------------------------------------------------------
typedef` TYPE_DECL
        name                    IDENTIFIER_POINTER(DECL_NAME(node)))
        redefined type          TREE_TYPE(node)
-------------------------------------------------------------------------------
types:
simple types: INTEGER_TYPE , REAL_TYPE, COMPLEX_TYPE, BOOLEAN_TYPE, VOID_TYPE
        name            IDENTIFIER_POINTER (DECL_NAME(TYPE_NAME(node)))
        modifiers        (see \ref{fig_cpp2xml_1b} for more information). e.g: unsigned

pointer types and reference types:
        type pointed    TREE_TYPE

record_type
        name            IDENTIFIER_POINTER (DECL_NAME(TYPE_NAME(node)))
        modifiers       (see \ref{fig_cpp2xml_1b} for more information)
```

Chapter 7

Abstract tree to XML

7.1 Introduction

This part deals with the generation of an XML representation code from the tree representation.

7.2 XML Syntax used

An XML document contains a succession of elements delimited by **tags**. There are two kinds of **tags** :

- the short ones which are of the following form :

```
<tag_name/>
```

- the long ones which have starting and ending tags and possibly a content : some text or a new tagged element.

```
<tag_name>
   content
</tag_name>
```

or

```
<tag_name1>
   <tag_name2>
      value
   </tag_name2>
   <tag_name3/>
 </tag_name1>
```

So we used such a syntax to represent a C++ source code where "tag_name" represents a C++ entity (e.g var_decl for a variable declaration) and "value" any identifier or value which is not C++-specific.
A full listing of tags used can be found in 5

7.3 A small example

Here is a small function declaration :

```
float foo(int dummy1, Class1 dummy2);
```

And the corresponding XML code:

```
<function_decl>
    <name uniq="foo__FiG6Class1">foo</name>
    <declaration/>
    <function/>
    <function_type>
        <simple_type>
            <float/>
            <modifiers></modifiers>
        </simple_type>
        <modifiers></modifiers>
    </function_type>
    <arg-types>
        <param>
            <simple_type>
                <int/>
                <modifiers></modifiers>
            </simple_type>
            <initial></initial>
        </param>
        <param>
            <record_type>
                <name uniq="6Class1">Class1</name>
                <modifiers></modifiers>
            </record_type>
            <initial></initial>
        </param>
    </arg-types>
    <modifiers></modifiers>
</function_decl>
```

Chapter 8

Source code architecture

There are two kinds of files in the archive: gcc's files we have modified and new files written from scratch.

The first two section sections will describe briefly what is in which files. Then you will be explained how to integrate cpp2xml into a new release of gcc.

8.1 Description of the new files

They are stored in the frisco directory and sub-directories:

```
treegen/    This directory contains the sources for tree building (cpp2tree)
  treegen_main.c      Entry point and general functions
  treegen_decls.c     Build top-level nodes (variables, functions, classes)
  treegen_types.c     Build types
  treegen_skips.c     Function to skip unwanted internal gcc nodes
                      and unhandled functionalities

xmlgen/     This directory contains the sources for the XML generation (tree2xml)
  xmlgen_main.c       Entry point and general functions
  xmlgen_decls.c      Generate top-level nodes (variables, functions, classes)
  xmlgen_types.c      Generated types nodes (simple types, arrays, ...)

  xmlgen_tags.c       XML specific syntax


misc/      This directory contains miscellaneous source code ...
  XMLinput.c      ** misnamed **  handle string conversions in identifiers
  display.c       ** Debug purpose ** Quick representation of the internal tree
  files_tools.c   Handle the -StdDir option
  glob_def.c      Global definitions...
  list.c          A simple template list
  memory.c        memory managment for hi-level data structures
  simple_stack.c  A simple template stack
  utilities.c     misc functions (checks nodes, get some characteristics)
```

8.2 Brief description of gcc modified Files

These are the files from gcc-2.8.0 we have modified in order to make cpp2xml work.

```
Makefile.in         Used in Makefile generation
cccp.c              GNU preprocessor main source code
toplev.c            GNU parser main source code
gcc.c               GNU compiler main source code
cplus-dem.c         Contains the table of all C++ operators

cp/
  Make-lang.in      C++ specific, used in Makefile generation
  Makefile.in       C++ specific, used in Makefile generation
  lex.c             Entry point for change of files
  class.c           Entry point for classes generation
  decl.c            Entry point for function, variables and enums generation

  g++spec.c         Used in command line options (-GenXML, -split, -stdDir)
  lang-specs.h      Used in command line options C++ specific
  typeck2.c         Modified compiler bug message
```

8.3 How to integrate cpp2xml in new releases of gcc

To perform the integration of **cpp2xml** in gcc, you must first add the **frisco** directory under the source directory of gcc and then modify several gcc files.

In this section, we will describe the modifications made to gcc files so that they can easily be integrated into a new version of gcc even if some changes have been made in one of the files concerned.
A full "*diff result*" is provided in subsection 8.3.2 and in the source directory too.

8.3.1 Modifications to gcc

Cpp2xml adds three options to gcc: **-GenXML**, **-split** and **-stdDir** *standard.lst.* In addition, there is also **treeIncludes** but it is not needed for the XML generation.

To add new options, several steps are needed.

1. Modify the "**spec language**" for **C** and **C++**. This is done in **gcc.c** and **lang-specs.h**.
   - `-GenXML` works on a preprocessed file, during the parsing phase
   - `-treeIncludes` works during the preprocessing phase
   - When `-GenXML` is set, the compilation stops before writing assembler
   - `-split` and `-stdDir` are not valid without `-GenXML`
   - `-o` will generate the output xml representation in the specified file.
   - `-S` and `-GenXML` are incompatible.
2. Perform checks and actions with the new options
   - Assign one flag for each of the four options (three of them in **toplev.c** and one in **cccp.c**)
   - Add stdDir to the list of options which take one parameter (**gcc.c**)
   - Tell the compiler not to perform linkage when -GenXML is set (**gcc.c**)

- Perform any change needed to the fact that when GenXML is set, no assembler has to be generated(**toplev.c**)

3. Integrate new actions at the right time of the compilation

   Changes specific to the **-treeIncludes** option (all in **cccp.c**)

   - Initialisation must appear after the the processing of all switches and before the actual preprocessing phase.
   - The actual generation of line for an include file should be made as soon as the preprocessor knows there has been a change of file and whether it's entering or leaving a file.
   - Small changes are made so that even already included files have their name generated any time they appear in the hierarchy tree

4. Changes specific to the **-GenXML** option:

   - The entry point for a class representation should be done at the beginning of finish_struct (in **cp/class.c**
   - The entry point for global function and variable declarations should be done at the far end of cp_finish_decl (in **cp/decl.c**)
   - The entry point for global function definitions (with bodies) should be done at the beginning of finish_function (in **cp/decl.c**)
   - The entry point for enumerated types should be done at the far end of finish_enum(in **cp/decl.c**)
   - Generation of "change of file" nodes are done in **lex.c** as soon as a change of file occurs.

5. Miscellaneous

   - The bug report message should be changed in **cp/typeck2.c**
   - A small patch has been applied so that f(...) (ellipsis with no argument) doesn't crash the compiler. It is done in **cp/decl.c**

6. Changes to the Makefile

   gcc Makefile is build from several files. Three of them are to be changed: Makefile.in, cp/Makefile.in and cp/Make-lang.in.

   - Compile files added to gcc (the frisco directory) to object files.
   - Compile gcc using the new includes and object files

8.3.2 Differences, file by file with g++ 2.8.0

Makefile.in

```
> #   Adapted to generate XML, Florence Defaix. FRISCO 1998
> #   !! NOTE: This file is NOT the original file given with g++ !!
>
492a496,506
> ################## cpp2xml begin ################################
> # Florence Defaix  for C++2Aldor
>
> CPP2XML_DIR = frisco
> CPP2XML_INC = -I$(srcdir)/$(CPP2XML_DIR) \
>             -I$(srcdir)/$(CPP2XML_DIR)/treegen \
>             -I$(srcdir)/$(CPP2XML_DIR)/xmlgen \
>               -I$(srcdir)/$(CPP2XML_DIR)/misc
>
> ################## cpp2xml end ################################
>
497c511,512
< INCLUDES = -I. -I$(srcdir) -I$(srcdir)/config
---
> # cpp2xml + $(FRISCO_INC)
> INCLUDES = -I. -I$(srcdir) -I$(srcdir)/config $(CPP2XML_INC)
```

cccp.c

```
 5a6,8
>    Adapted to generate XML, Florence Defaix. FRISCO 1998
>    !! NOTE: This file is NOT the original file given with g++ !!
>
397a401,422
>
> /* cpp2xml begin */
>
> /* boolean : true if the file has already been included
>    so that we can add an empty changeOfFile */
> static int already_included = FALSE;
>
> static int depth_include = 0;
> static FILE *  includesFile;
> static int dump_hierarchy = 0;
>
> void XML_indent_to (file, column)
>      FILE *file;
>      int column;
> {
>   int i;
>   fprintf(file,"\n");
>   for (i = 0; i < column; i++)
>     fprintf(file," ");
> }
> /* cpp2xml end *
>
1208a1234,1237
>   /* cpp2xml begin */
>   char *includesFileName;
>   /* cpp2xml end */
>
1432a1462,1466
>       /* cpp2xml begin */
>       else if (!strcmp (argv[i], "-treeIncludes")) {
>         dump_hierarchy = 1;
>       }
>       /* cpp2xml end*/
1695a1730,1761
> /* cpp2xml begin */
>   if (dump_hierarchy){
>
>     /* construct name for the hierarchy file */
>     if (in_fname){
>       includesFileName = (char *) malloc((strlen(in_fname)+5) *sizeof(char));
>       strcpy(includesFileName, in_fname);
>       /* to do handle time.h and sys/time.h */
>     }else{
>       includesFileName = (char *) malloc(10 *sizeof(char));
>       strcpy(includesFileName, "stdin");
>     }
>     strcat(includesFileName, ".inc");
>
```

```
>     /* open hierarchy file */
>     includesFile = fopen(includesFileName,"w");
>     if (!includesFile){
>       fprintf(stderr,"couldn't create/open for writing %s!\n",
>             includesFileName);
>       exit(-1); /* EXIT */
>     }
>     free(includesFileName);
>
>     fprintf(includesFile,"<file>");
>     XML_indent_to(includesFile,depth_include + 4);
>     if (in_fname)
>       fprintf(includesFile,"<name>\"%s\"</name>",in_fname);
>     else
>       fprintf(includesFile,"<name>\"\"</name>");
>   }
>   /* cpp2xml end*/
>
2193a2260,2266
>   /* cpp2xml begin */
>   if (dump_hierarchy){
>     XML_indent_to(includesFile,depth_include);
>     fprintf(includesFile,"</file>");
>   }
>   /* cpp2xml end */
>
4835a4909
>   /* cpp2xml begin */
4838a4913,4919
>     already_included = FALSE;
>   }
>   else {
>     /* Rem: We go in the file anyway and will leave later. should not change
>        anything in the compiler without -GenXML */
>     already_included = TRUE;
>   }
4840,4862c4921,4942
<     fd = open (fname, O_RDONLY, 0);
<
<     if (fd < 0)
<       return fd;
<
<     if (!inc) {
<       /* FNAME was not in include_hashtab; insert a new entry.  */
<       inc = (struct include_file *) xmalloc (sizeof (struct include_file));
<       inc->next = head;
<       inc->fname = fname;
<       inc->control_macro = 0;
<       inc->deps_output = 0;
<       if (fstat (fd, &inc->st) != 0)
<       pfatal_with_name (fname);
<       *phead = inc;
<
<       /* Look for another file with the same inode and device.  */
<       if (lookup_ino_include (inc)
```

```
<         && inc->control_macro
<         && (!inc->control_macro[0] || lookup (inc->control_macro, -1, -1))) {
<       close (fd);
<       fd = -2;
<       }
---
>   fd = open (fname, O_RDONLY, 0);
>
>   if (fd < 0)
>     return fd;
>
>   if (!inc) {
>     /* FNAME was not in include_hashtab; insert a new entry.  */
>     inc = (struct include_file *) xmalloc (sizeof (struct include_file));
>     inc->next = head;
>     inc->fname = fname;
>     inc->control_macro = 0;
>     inc->deps_output = 0;
>     if (fstat (fd, &inc->st) != 0)
>       pfatal_with_name (fname);
>     *phead = inc;
>
>     /* Look for another file with the same inode and device.  */
>     if (lookup_ino_include (inc)
>       && inc->control_macro
>       && (!inc->control_macro[0] || lookup (inc->control_macro, -1, -1))) {
>       close (fd);
>       fd = -2;
4864,4873d4943
<
<     /* For -M, add this file to the dependencies.  */
<     if (! inc->deps_output  &&  (system_include_depth != 0) < print_deps) {
<       inc->deps_output = 1;
<       deps_output (fname, ' ');
<     }
<
<     /* Handle -H option.  */
<     if (print_include_names)
<       fprintf (stderr, "%*s%s\n", indepth, "", fname);
4874a4945,4956
>
>   /* For -M, add this file to the dependencies.  */
>   if (! inc->deps_output  &&  (system_include_depth != 0) < print_deps) {
>     inc->deps_output = 1;
>     deps_output (fname, ' ');
>
>   /* Handle -H option.  */
>   if (print_include_names)
>     fprintf (stderr, "%*s%s\n", indepth, "", fname);
>
>   }
>   /* cpp2xml end */
5048d5129
<   rescan (op, 0);
5050,5051c5131,5138
```

```
<   if (missing_newline)
<     fp->lineno--;
---
>   /* cpp2xml begin (CHANGE)*/
>   if (!already_included){
>     rescan (op, 0);
>
>     if (missing_newline)
>       fp->lineno--;
>   }
>   /* cpp2xml end */
7823a7911,7927
>
>     /* cpp2xml begin */
>     if (dump_hierarchy){
>       if (file_change == enter_file){
>       depth_include +=4;
>       XML_indent_to(includesFile,depth_include);
>       fprintf(includesFile,"<include>");
>       XML_indent_to(includesFile,depth_include + 4);
>       fprintf(includesFile,"<name>\"%s\"</name>",ip->nominal_fname);
>       }
>       else{
>       XML_indent_to(includesFile,depth_include);
>       fprintf(includesFile,"</include>");
>       depth_include -=4;
>       }
>     }
>     /* cpp2xml end */
7824a7929
```

cplus-dem.c

```
5c5,8
<
---
>
>     Adapted to generate XML, Florence Defaix. FRISCO 1998
>    !! NOTE: This file is NOT the original file given with g++ !!
>
112c115,118
< static const struct optable
---
> /* cpp2xml begin */
> /*   removed "static" to use optable from outside... */
> const struct optable
> /* cpp2xml end */
```

gcc.c

```
4,6d3
<    Adapted to generate XML, Florence Defaix. FRISCO 1998
<    !! NOTE: This file is NOT the original file given with g++ !!
546d542
< /* cpp2xml begin (CHANGE) */
553,555c549
<   || !strcmp (STR, "isystem") || !strcmp (STR, "specs")\
<   || !strcmp (STR, "stdDir"))
<      /* cpp2xml end */
---
>   || !strcmp (STR, "isystem") || !strcmp (STR, "specs"))
589,590d582
< /* cpp2xml begin (CHANGE)*/
< /* treeInclude GenXML split stdDir */
607d598
<         %{treeIncludes}\
629d619
<         %{treeIncludes}\
667d656
<         %{treeIncludes}\
682,684d670
<                         %{GenXML} %{stdDir*} %{split} \
<                         %{stdDir*:%{!GenXML:%eGNU C does not support -stdDir without using -GenXML}} \
<                         %{split:%{!GenXML:%eGNU C does not support -split without using -GenXML}} \
686,691c672,675
<                         %{GenXML:%{S:%eYou cannot use -GenXML and -S at the same time}} \
<                         %{S:%W{o*}%{!o*:-o %b.s}} \
<                         %{!S:{GenXML:%W{o*}%{!o*:-o %b.xml}}{!GenXML:-o %{|!pipe:%g.s}}} |\n\
<                     %{!GenXML:%{!S:as %a %Y\
<                         %{c:%W{o*}%{!o*:-o %w%b%O}}%{!c:-o %d%w%u%O}\
<                         %{!pipe:%g.s} %A\n }}}}}"},
---
>  %{S:%W{o*}%{!o*:-o %b.s}}%{!S:-o %{|!pipe:%g.s}} |\n\
>       %{!S:as %a %Y\
>       %{c:%W{o*}%{!o*:-o %w%b%O}}%{!c:-o %d%w%u%O}\
>       %{!pipe:%g.s} %A\n }}}}"},
732c716
<  %{!c:%{!M:%{!MM:%{!E:%{!GenXML:%{!S:ld %l %X %{o*} %{A} %{d} %{e*} %{m} %{N} %{n} \
---
>  %{!c:%{!M:%{!MM:%{!E:%{!S:ld %l %X %{o*} %{A} %{d} %{e*} %{m} %{N} %{n} \
739c723
<  \n }}}}}}}";
---
>  \n }}}}}}";
744c728
<  %{!c:%{!M:%{!MM:%{!E:%{!GenXML:%{!S:ld %l %X %{o*} %{A} %{d} %{e*} %{m} %{N} %{n} \
---
>  %{!c:%{!M:%{!MM:%{!E:%{!S:ld %l %X %{o*} %{A} %{d} %{e*} %{m} %{N} %{n} \
751c735
<  \n }}}}}}}";
>  \n }}}}}}";
753d736
< /* cpp2xml end */
```

toplev.c

```
3a4,6
>   Adapted to generate XML, Florence Defaix. FRISCO 1998
>    !! NOTE: This file is NOT the original file given with g++ !!
>
246a250,257
> /* cpp2xml begin */
> /* c++toXML Flags */
> int xml_gen = 0;
> int xml_split = 0;
> char *xml_dest;
> char *xml_stdDir;
> /* cpp2xml end */
>
267a279,283
> /* cpp2xml begin */
> /* Name for output file of xml generation  */
> char *xml_file_name;
> /* cpp2xml end */
>
880a897,899
> /* cpp2xml begin */
> FILE *xml_out_file;
> /* cpp2xml end */
1042a1062,1065
>   /* cpp2xml begin */
>   if (xml_out_file)
>     fflush (xml_out_file);
>   /* cpp2xml end */
2280c2303,2327
<   /* Open assembler code output file.  */
---
>   /* cpp2xml begin */
>   /* set xml output_file */
>   if (xml_gen){
>     if (! name_specified && xml_file_name == 0){
>       xml_out_file = stdout;
>     }
>     else
>       {
>  if (xml_file_name == 0){
>    int len = strlen (dump_base_name);
>    register char *xml_name = (char *) xmalloc (len + 8);
>    strcpy (xml_name, dump_base_name);
>    strip_off_ending (xml_name, len);
>
>    strcat(xml_name,".xml");
>    xml_file_name = (char *) xmalloc (strlen (xml_name) + 1);
>    strcpy (xml_file_name, xml_name);
>  }
>  xml_out_file = fopen (xml_file_name, "w");
>  if (xml_out_file == 0)
>    pfatal_with_name (xml_file_name);
```

```
>       }
> #ifdef IO_BUFFER_SIZE
>   setvbuf (xml_out_file, (char *) xmalloc (IO_BUFFER_SIZE), _IOFBF, IO_BUFFER_SIZE);
> #endif
2282,2283c2329,2337
<   if (! name_specified && asm_file_name == 0)
<     asm_out_file = stdout;
---
>   asm_out_file = fopen (asm_file_name, "w"); /* /dev/null */
>   }
>   else{
>   /* cpp2xml end */
>
>     /* Open assembler code output file.  */
>     if (! name_specified && asm_file_name == 0){
>       asm_out_file = stdout;
>     }
2290,2301c2344,2355
<       strcat (dumpname, ".s");
<       if (asm_file_name == 0)
< {
<   asm_file_name = (char *) xmalloc (strlen (dumpname) + 1);
<   strcpy (asm_file_name, dumpname);
< }
<       if (!strcmp (asm_file_name, "-"))
< asm_out_file = stdout;
<       else
< asm_out_file = fopen (asm_file_name, "w");
<       if (asm_out_file == 0)
< pfatal_with_name (asm_file_name);
---
> strcat (dumpname, ".s");
> if (asm_file_name == 0)
>   {
>     asm_file_name = (char *) xmalloc (strlen (dumpname) + 1);
>     strcpy (asm_file_name, dumpname);
>   }
> if (!strcmp (asm_file_name, "-"))
>   asm_out_file = stdout;
> else
>   asm_out_file = fopen (asm_file_name, "w");
> if (asm_out_file == 0)
>   pfatal_with_name (asm_file_name);
2303d2356
<
2308c2361,2363
<
---
>   /* cpp2xml begin */
>   }
>   /* cpp2xml end */
2797c2852,2854
<   if (ferror (asm_out_file) != 0 || fclose (asm_out_file) != 0)
---
>
```

```
>   /* begin cpp2xml (CHANGE) */
>   if (asm_out_file && (ferror (asm_out_file) != 0 || fclose (asm_out_file) != 0))
2798a2856,2858
>   if (xml_out_file && (ferror (xml_out_file) != 0 || fclose (xml_out_file) != 0))
>     fatal_io_error (xml_file_name);
>   /* end cpp2xml */
3965a4026,4034
>    /* cpp2xml begin */
>    /* cpp2xml options */
>    else if (!strcmp (str, "GenXML"))
>      xml_gen = 1;
>    else if (!strcmp (str, "split"))
>      xml_split = 1;
>    else if (!strcmp (str, "stdDir"))
>      xml_stdDir  = argv[++i];
>    /* cpp2xml end */
4209c4278,4286
<        asm_file_name = argv[++i];
---
>        /* cpp2xml BEGIN NEW */
>        if (xml_gen){
>  xml_file_name = argv[++i];
>  asm_file_name = xstrdup("/dev/null");
>        }
>        else{
>  asm_file_name = argv[++i];
>        }
>        /* cpp2xml END NEW*/
```

cp/Make-lang.in

```
5a6,8
> #   Adapted to generate XML, Florence Defaix. FRISCO 1998
> #   !! NOTE: This file is NOT the original file given with g++ !!
>
148a152,187
> ################## cpp2xml begin ################################
> # Florence Defaix  for C++2xml
> CPP2XML_DIR = frisco
> XML_TREE_H = $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_main.h
>
>
> CXX2XML_OBJS =  $(srcdir)/../cplus-dem.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_main.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_decls.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_types.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_skips.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_main.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_decls.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_types.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_tags.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/display.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/glob_def.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/list.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/memory.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/utilities.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/files_tools.o
>
>
> CPP2XML_INC = -I$(srcdir)/../$(CPP2XML_DIR) \
>       -I$(srcdir)/../$(CPP2XML_DIR)/treegen \
>       -I$(srcdir)/../$(CPP2XML_DIR)/xmlgen \
>       -I$(srcdir)/../$(CPP2XML_DIR)/misc
>
> # compile file in the source directory
> $(srcdir)/../$(CPP2XML_DIR)/%.o: $(srcdir)/../$(CPP2XML_DIR)/%.c
>  $(CC) -c -o $@ $(ALL_CFLAGS) $(ALL_CPPFLAGS) $(INCLUDES) $<
>
> $(srcdir)/../cplus-dem.o: $(srcdir)/../cplus-dem.c $(DEMANGLE_H)
>  $(CC) -c -o $@ $(ALL_CFLAGS) $(ALL_CPPFLAGS) $(INCLUDES) $<
> ################## cpp2xml end  ################################
>
153c192,195
< INCLUDES = -I. -I.. -I$(srcdir) -I$(srcdir)/.. -I$(srcdir)/../config
---
> # cpp2xml begin +$(CPP2XML_INC)
> INCLUDES = -I. -I.. -I$(srcdir) -I$(srcdir)/.. -I$(srcdir)/../config\
>  $(CPP2XML_INC)
> #cpp2xml end
176c218,221
< ../cc1plus: $(P) $(CXX_OBJS) $(OBJDEPS) $(LIBDEPS)
---
>
```

```
> # cpp2xml begin (CHANGE)
> # + $(CXX2XML_OBJS)
> ../cc1plus: $(P) $(CXX_OBJS)  $(CXX2XML_OBJS) $(OBJDEPS) $(LIBDEPS)
178c223,224
<        $(CXX_OBJS) $(OBJS) $(LIBS)
---
>        $(CXX_OBJS) $(CXX2XML_OBJS) $(OBJS) $(LIBS)
> # cpp2xml end
233c279,280
< decl.o : decl.c $(CONFIG_H) $(CXX_TREE_H) $(srcdir)/../flags.h \
---
> #cpp2xml begin + $(XML_TREE_H)
> decl.o : decl.c $(CONFIG_H) $(CXX_TREE_H) $(XML_TREE_H) $(srcdir)/../flags.h \
235a283
> #cpp2xml end
242c290,292
< class.o : class.c $(CONFIG_H) $(CXX_TREE_H) $(srcdir)/../flags.h
---
> #cpp2xml begin + $(XML_TREE_H)
> class.o : class.c $(CONFIG_H) $(CXX_TREE_H) $(srcdir)/../flags.h $(XML_TREE_H)
> #cpp2xml end
```

cp/Makefile.in

```
5a6,8
> #   Adapted to generate XML, Florence Defaix. FRISCO 1998
> #   !! NOTE: This file is NOT the original file given with g++ !!
>
148a152,188
>
> ################## cpp2xml begin ################################
> # Florence Defaix  for C++2xml
> CPP2XML_DIR = frisco
> XML_TREE_H = $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_main.h
>
>
> CXX2XML_OBJS =  $(srcdir)/../cplus-dem.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_main.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_decls.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_types.o \
>  $(srcdir)/../$(CPP2XML_DIR)/treegen/treegen_skips.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_main.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_decls.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_types.o \
>  $(srcdir)/../$(CPP2XML_DIR)/xmlgen/xmlgen_tags.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/display.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/glob_def.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/list.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/memory.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/utilities.o \
>  $(srcdir)/../$(CPP2XML_DIR)/misc/files_tools.o
>
>
> CPP2XML_INC = -I$(srcdir)/../$(CPP2XML_DIR) \
>             -I$(srcdir)/../$(CPP2XML_DIR)/treegen \
>             -I$(srcdir)/../$(CPP2XML_DIR)/xmlgen \
>               -I$(srcdir)/../$(CPP2XML_DIR)/misc
>
> # compile file in the source directory
> $(srcdir)/../$(CPP2XML_DIR)/%.o: $(srcdir)/../$(CPP2XML_DIR)/%.c
>       $(CC) -c -o $@ $(ALL_CFLAGS) $(ALL_CPPFLAGS) $(INCLUDES) $<
>
> $(srcdir)/../cplus-dem.o: $(srcdir)/../cplus-dem.c $(DEMANGLE_H)
>       $(CC) -c -o $@ $(ALL_CFLAGS) $(ALL_CPPFLAGS) $(INCLUDES) $<
> ################## cpp2xml end  ################################
>
153c193,196
< INCLUDES = -I. -I.. -I$(srcdir) -I$(srcdir)/.. -I$(srcdir)/../config
---
> # cpp2xml begin +$(CPP2XML_INC)
> INCLUDES = -I. -I.. -I$(srcdir) -I$(srcdir)/.. -I$(srcdir)/../config\
>       $(CPP2XML_INC)
> # cpp2xml end
176c219,222
< ../cc1plus: $(P) $(CXX_OBJS) $(OBJDEPS) $(LIBDEPS)
---
```

```
>
> # cpp2xml begin (CHANGE)
> # + $(CXX2XML_OBJS)
> ../cciplus: $(P) $(CXX_OBJS) $(CXX2XML_OBJS) $(OBJDEPS) $(LIBDEPS)
178c224,225
<             $(CXX_OBJS) $(OBJS) $(LIBS)
---
>             $(CXX_OBJS) $(CXX2XML_OBJS) $(OBJS) $(LIBS)
> # cpp2xml end
233c280,281
< decl.o : decl.c $(CONFIG_H) $(CXX_TREE_H) $(srcdir)/../flags.h \
---
> #cpp2xml begin + $(XML_TREE_H)
> decl.o : decl.c $(CONFIG_H) $(CXX_TREE_H) $(XML_TREE_H) $(srcdir)/../flags.h \
235a284
> #cpp2xml end
242c291,293
< class.o : class.c $(CONFIG_H) $(CXX_TREE_H) $(srcdir)/../flags.h
---
> #cpp2xml begin + $(XML_TREE_H)
> class.o : class.c $(CONFIG_H) $(CXX_TREE_H) $(srcdir)/../flags.h $(XML_TREE_H)
> #cpp2xml end
```

cp/class.c

```
6a7,9
>    Adapted to generate XML, Florence Defaix. FRISCO 1998
>    !! NOTE: This file is NOT the original file given with g++ !!
>
38a42,45
> /*cpp2xml begin */
> extern int xml_gen;
> /*cpp2xml end */
>
4330a4338,4346
>   /* cpp2xml begin */
>   /* ************************************************************************
>    * CPP2XML ENTRY POINT
>    * print everything about one class
>    * ************************************************************************ */
>   if (xml_gen && list_of_fieldlists)
>     cpp2tree(list_of_fieldlists);
>   /* cpp2xml end */
>
```

cp/decl.c

```
4a5,7
>    Adapted to generate XML, Florence Defaix. FRISCO 1998
>    !! NOTE: This file is NOT the original file given with g++ !!
65a69,72
> /* cpp2xml begin */
> extern int xml_gen;
> /* cpp2xml end */
>
438a446,447
> /* cpp2xml begin (CHANGE) */
> /* removed "static" to be used from outside */
439a449
> /* cpp2xml end */
6977a6988,6993
>   /* cpp2xml begin */
>   /* ENTRY POINT for definitions outside a class : function_def and variables */
>   if (xml_gen && !current_class_name)
>     cpp2tree(decl);
>   /* cpp2xml end */
10250c10266,10270
<       parmtype = TREE_VALUE (parmtypes);
---
>       /* cpp2xml begin (CHANGE)*/
>       /* patch so that f(...) doesn't crash ...*/
>       if (parmtypes != NULL_TREE)
>       parmtype = TREE_VALUE (parmtypes);
>       /* cpp2xml end */
10281,10282c10301,10306
<   else if (TREE_CODE (parmtype) == VOID_TYPE
<          || TREE_PURPOSE (parmtypes) != NULL_TREE)
---
>   /* cpp2xml begin (CHANGE) */
>   /* patch for f(...) */
>   else if ( parmtype && parmtypes &&
>           (TREE_CODE (parmtype) == VOID_TYPE
>            || TREE_PURPOSE (parmtypes) != NULL_TREE)
>           )
10283a10308
>   /* cpp2xml end */
11083a11109,11114
>   /* cpp2xml begin */
>   /* ENTRY POINT for Enum declarations */
>   if (xml_gen && !current_class_name)
>     cpp2tree(enumtype);
>   /* cpp2xml end */
>
11886a11918,11923
>   /* cpp2xml begin */
>   /* ENTRY POINT for external functions definition (with bodies) */
>   if (xml_gen && !current_class_name)
>     cpp2tree(fndecl);
>   /* cpp2xml end */
```

cp/g++spec.c

```
3a4,6
>     Adapted to generate XML, Florence Defaix. FRISCO 1998
>     !! NOTE: This file is NOT the original file given with g++ !!
>
152a156,159
>     /* cpp2xml begin */
>     else if (strncmp (argv[i], "-GenXML",7)  == 0)
>       library = 0;
>     /* cpp2xml end */
```

cp/lang-specs.h

```
3a4,6
>    Adapted to generate XML, Florence Defaix. FRISCO 1998
>    !! NOTE: This file is NOT the original file given with g++ !!
>
33a37
>  %{treeIncludes}\
41a46,48
>                              %{GenXML} %{stdDir*} %{split} \
>                              %{stdDir*:%{!GenXML:%eGNU C does not support -stdDir without using -GenXML }} \
>                              %{split:%{!GenXML:%eGNU C does not support -split without using -GenXML}} \
46,49c53,58
<      %{S:%W{o*}%{!o*:-o %b.s}}%{!S:-o %{|!pipe:%g.s}}|\n\
<               %{!S:as %a %Y\
<        %{c:%W{o*}%{!o*:-o %w%b%O}}%{!c:-o %d%w%u%O}\
<                        %{!pipe:%g.s} %A\n }}}}",
---
>                              %{GenXML:%{S:%eYou cannot use -GenXML and -S at the same time}} \
>                              %{S:%W{o*}%{!o*:-o %b.s}} \
>                              %{!S:%{GenXML:%W{o*}%{!o*:-o %b.xml}}%{!GenXML:-o %{|!pipe:%g.s}}} |\n\
>                         %{!GenXML:%{!S:as %a %Y\
>                              %{c:%W{o*}%{!o*:-o %w%b%O}}%{!c:-o %d%w%u%O}\
>                              %{!pipe:%g.s} %A\n }}}}}",
54a64,66
>                              %{GenXML} %{stdDir*} %{split} \
>                              %{stdDir*:%{!GenXML:%eGNU C does not support -stdDir without using -GenXML}} \
>                              %{split:%{!GenXML:%eGNU C does not support -split without using -GenXML}} \
57,60c69,74
<      %{S:%W{o*}%{!o*:-o %b.s}}%{!S:-o %{|!pipe:%g.s}} |\n\
<              %{!S:as %a %Y\
<      %{c:%W{o*}%{!o*:-o %w%b%O}}%{!c:-o %d%w%u%O}\
<      %{!pipe:%g.s} %A\n }}}}",
---
>                              %{GenXML:%{S:%eYou cannot use -GenXML and -S at the same time}} \
>                              %{S:%W{o*}%{!o*:-o %b.s}} \
>                              %{!S:%{GenXML:%W{o*}%{!o*:-o %b.xml}}%{!GenXML:-o %{|!pipe:%g.s}}} |\n\
>                         %{!GenXML:%{!S:as %a %Y\
>                              %{c:%W{o*}%{!o*:-o %w%b%O}}%{!c:-o %d%w%u%O}\
>                              %{!pipe:%g.s} %A\n }}}}}",
```

cp/lex.c

```
4a5,7
>   Adapted to generate XML, Florence Defaix. FRISCO 1998
>    !! NOTE: This file is NOT the original file given with g++ !!
>
72a76,82
> /* begin cpp2xml */
> extern int xml_gen;
> extern int xml_split;
> extern FILE *xml_out_file;
> char *oldFileName = NULL;
> /* end cpp2xml */
>
126c136,139
< static int c_header_level = 0;
---
> /* cpp2xml begin */
> /*  removed static */
> int c_header_level = 0;
> /* cpp2xml end */
403a417,421
>   /* cpp2xml begin */
>   if (xml_gen && xml_split)
>     buildChangeOfFile(TRUE,"",input_filename);
>   /* cpp2xml end */

>
412a431,441
>
>   /* cpp2xml begin */
>   if (xml_gen && xml_split){
>     buildChangeOfFile(FALSE,oldFileName,"");
>     free(oldFileName);
>     oldFileName = NULL;
>   }
>   if (xml_gen)
>     tree2xml(xml_out_file);
>   /* cpp2xml end */
>
2439a2469,2472
>       /* cpp2xml begin */
>       oldFileName = strdup(input_filename);
>       /* cpp2xml end */
>
2537a2571,2579
>             /* cpp2xml begin */
>             if (xml_gen && xml_split)
>               buildChangeOfFile(TRUE,oldFileName,input_filename);
>             /* initialized here so that olfFileName is correct when we leave
>                the main file */
>             free(oldFileName);
>             oldFileName = strdup(input_filename);
>             /* cpp2xml end */
```

```
>
2555a2598,2607
>             /* cpp2xml begin */
>             if (xml_gen && xml_split)
>               buildChangeOfFile(FALSE,oldFileName,input_filename);
>
>             /* initialized here so that olfFileName is correct when we leave
>                the main file */
>             free(oldFileName);
>             oldFileName = strdup(input_filename);
>             /* cpp2xml end */
>
```

cp/typeck2.c

```
43a47,50
> /* cpp2xml begin */
> extern int xml_gen;
> /* cpp2xml end */
>
325c332,339
<         ack ("Please submit a full bug report to 'bug-g++@prep.ai.mit.edu'.");
---
>         /* cpp2xml begin */
>         if (xml_gen){
>           ack ("C++ to XML : if your file compile without -GenXML, please submit a full bug report to 'fdefai
>           ack ("Otherwise please submit a full bug report to 'bug-g++@prep.ai.mit.edu'. ");
>         }
>         else
>           /* cpp2xml end */
>           ack ("Please submit a full bug report to 'bug-g++@prep.ai.mit.edu'.");
339c353,360
<   fatal ("Please submit a full bug report to 'bug-g++@prep.ai.mit.edu'.");
---
>   /* cpp2xml begin */
>   if (xml_gen){
>     ack ("C++ to XML : if your file compile without -GenXML, please submit a full bug report to 'fdefaix@csd.u
>     fatal ("Otherwise please submit a full bug report to 'bug-g++@prep.ai.mit.edu'. ");
>   }
>   else
>     /* cpp2xml end */
>     fatal ("Please submit a full bug report to 'bug-g++@prep.ai.mit.edu'.");
```