AdaDep User Guide

Last edited: 17 October 2016

This is the AdaDep User Guide. It describes how to install and use AdaDep.

Table of Contents

1 Introduction **2**

2 Installing AdaDep **3**

3 Principle of operation **4**

4 Using AdaDep **5**

5 AdaDep and ASIS **7**

6 Troubleshooting **8**

6.1 Message: Inconsistent tree file for ..., please remove .adt file 8

6.2 AdaDep halts with an error 8

7 Future of AdaDep **9**

1 Introduction

AdaDep is a tool for analysing dependencies. When an Ada unit references a package in a **with** clause, it means that something from this package is used in the unit. AdaDep will tell you which elements from the package are really used by the unit. Since it is based on semantic analysis, it does not depend on how you use the elements (i.e. either using full named notation, or thanks to a use clause). It is another example of the nice things that can be made using ASIS.

Please send bug reports, remarks, notes, good ideas, etc. to J-P. Rosen .

2 Installing AdaDep

This version uses ASIS-for-GNAT. As a consequence, you must have GNAT installed before you install AdaDep (even if you downloaded an executable version). If you are using another compiler that provides an ASIS interface, you may recompile AdaDep with this compiler. GNAT will not be necessary any more (but your compiler will, of course). See AdaDep and ASIS.

If you downloaded an executable distribution of AdaDep, please note that it has been compiled with GNAT GPL2016, and will not work with any other version of GNAT. If you have an older (or an ACT-supported) version of GNAT, either upgrade your GNAT to GPL2016 (recommended for older version!), or recompile AdaDep from the source distribution.

If you downloaded the source version of AdaDep, you must make sure that ASIS is installed on your system; then simply issue the following comaand:

```
gnat make -P build.gpr
```

If this fails (presumably because ASIS is not installed in a standard location), you can also gnatmake the file `adadep.adb`, including appropriate options to access the ASIS library.

AdaDep needs no setup, no special rights. Simply copy the executables to any convenient directory on your path; a good place, for example, is in the `bin` directory of your GNAT installation.

This version has been tested under Windows up to version 10 and Linux.

3 Principle of operation

AdaDep reads one or more ada units, and produces an output file that lists, for each "withed" package, which elements are actually used. Note that only elements of withed units are listed. A unit that has been renamed, or a locally declared instance of a library generic are not reported on the grounds that those are local to the unit under analysis.

If several units are analyzed, the listing maybe separated for the dependencies of each unit, or alternatively be merged (i.e. there is one listing showing which element is used by any of the units). Given the following Ada units:

```
package Pack is
   I : Integer;

   package Internal is
      V : Float;
   end Internal;
end Pack;
```

```
with Pack, Text_IO;
use Pack, Text_IO;
procedure Sample is
begin
   I := 1;
   Internal.V := 3.0;
   Put_Line (Integer'Image (I + Integer(Internal.V)));
end Sample;
```

A typical output of AdaDep will look like this:

```
SAMPLE (body) =>
=> from ADA.TEXT_IO
   PUT_LINE - A_Procedure_Declaration * 1
=> from PACK
   I - A_Variable_Declaration * 2
=> from PACK.INTERNAL
   V - A_Variable_Declaration * 2
=> from STANDARD
   INTEGER - An_Ordinary_Type_Declaration * 2
```

For each package, AdaDep writes a line for each used element, a short description of what it is (ASIS users will recognize it as the Declaration_Kind), and the number of times it is referred to.

Note that the name given in the listing is always the "true" name of the element; in the case of renamings, it may be different from the name that appears in the program text.

4 Using AdaDep

AdaDep is started by a command line of the form:

```
adadep [-bdhmrsw] [-o <output-file>] [-p <project-file>]
       <unit-name>{+|-<unit-name>} | @<list-file> {<element-name>}
       [-- <ASIS-options>]
```

Options:

As usual, options can be grouped or provided separately (i.e. "-b -s" is the same as "-bs"). Options can appear at any place on the command line.

-b process body of unit

-d debug

-h help. Print usage message and exit

-m merge. Merge output of the various units

-o specify the name of the output file. See below.

-p use source directory indications from provided project file. See below.

-r recursive. Process the unit and (recursively) all user units it depends on (including parent units if the unit is a child or a subunit). Predefined Ada units and units belonging to the compiler's run-time library are never processed.

-s process specification of unit

-v verbose mode: print extra messages, including the name of each unit as it is being processed.

-w overwrite an existing output file

If the -h option is given, AdaDep just prints a brief usage summary and exits. If neither -s or -b is given, -sb is assumed (i.e. both the specification and body are processed).

If no "-o <output-file>" option is given, the result is simply written to the standard output. Otherwise, AdaDep will output the result of processing each unit to the given file. If the file already exists, the output is appended to it (unless there is a -w option, in which case it is overridden); this makes it possible to run AdaDep separately on several units, and keep the result in one file.

<unit-name> is the name of the ada unit to analyze; note that it is a unit name, not a file name: case is not significant, and there should be no extension! Of course, child units are allowed following normal Ada naming rules: "Parent.Child". Alternatively, you can give an "@" followed by the name of a file instead of a unit name. This file should contain a list of unit names, one on each line. All units whose names are given in the file will be processed. If a name in the file starts with "; it will also be treated as an indirect file (i.e. the same process will be invoked recursively).

Normally, the output shows usage of entities on a unit-by-unit basis (i.e. you have a separate section of what is being used for each unit). If you specify the -m option, all sections are merged, i.e. you have the union of all used units, and the count of usage is the total count for all units.

If one or more <element-name> is specified, output will be limited to those elements (including their sub-elements). For example, if you specify "Ada.Text_IO", then only Ada.Text_IO and elements declared within Ada.Text_IO will be listed. Note that you must specify elements as full names: if you want to include only references to the variable V declared in the (library) package Pack, you must give it as "pack.V". Of course, since we are talking Ada elements here, case is irrelevant.

Everything that appears after a "--" will be treated as an ASIS option, as described in the ASIS user manual. Normally, you don't have to care about this, except for the issue discussed below.

If the units that you are processing reference other units whose source is not in the same directory, AdaDep needs to know how to access these units (as GNAT would). You can do this in two ways (not exclusive):

1. You can specify a project file (".gpr") with the "-p" option. AdaDep will automatically consider all the directories mentionned in all "source_dirs" specified in the project file or one of the projects it depends on (directly or indirectly).

   Alternatively, an old emacs project file (the file with a ".adp" extension used by the Ada mode of Emacs and older versions of AdaSubst) can also be specified with the "-p" option. AdaSubst will consider all the directories mentioned in "src_dir" lines from the project file.
2. you can include one or several "-I" options to reference other directories where sources can be found. The syntax is the same as the "-I" option for Gnat. Note that since "-I" is actually an ASIS option, it must appear after a "--".

5 AdaDep and ASIS

AdaDep has been tested only with ASIS-for-gnat; however, the only (known) dependency is for the implementation-dependents parameters used to open an ASIS context. These parameters are defined as strings in the package Implementation_Options (file implementation_options.ads), so all you have to do if you want to port AdaDep to another implementation is adjust these definitions. If you ever do so, please keep us informed by sending a note to rosen@adalog.fr.

6 Troubleshooting

6.1 Message: Inconsistent tree file for ..., please remove .adt file

It is a consequence of the way ASIS works that tree files (with a .adt extension) are created as part of the process. Currently, these files are not erased by AdaDep after execution. This speeds up execution if you run AdaDep several times. However, if you change your source files, the trees are no more consistent with the sources. Simply remove the .adt files and rerun AdaDep; new tree files will be created.

6.2 AdaDep halts with an error

There are many subtile special cases when analysing Ada code, so it may happen that you find a context that we didn't think about. If you encounter a bug while using AdaDep (generally an unhandled exception), AdaDep will output a number of information. Please rerun it with the -d option (debug) and save the output to a file, then send this file together with the file you are processing to rosen@adalog.fr. We will do our best to fix it. Of course, you are welcome to try and fix it yourself; we made every attempt to make the source of AdaDep quite readable, but be aware that you will need a good understanding of ASIS and Ada syntax to understand how it works.

7 Future of AdaDep

The structure of this programs allows it to be extended very easily. The notion of dependency extends far beyond "what do I use from this package". We intend to extend the functionnalities as we feel the need... or as users make suggestions!

Send good ideas, bug reports, suggestions of improvements to the program or to the documentation, etc. to J-P. Rosen.

Commercial support is available for this product; please refer to the file `support.txt` in the `doc` directory. For more information, or if you need any support or assistance for your Ada projects, please visit our Web site or send us a mail.