DiVinE

0.8.0

Generated by Doxygen 1.5.6

Tue Jul 21 09:09:36 2009

Contents

Chapter 1

DiVinE Development Standards

1.1 Introduction

This tutorial contains standards, advices and methods of development of DiVinE

1.2 CVS

1.2.1 Usage

CVS is a system for team project development. It is able to keep a track of all versions of all source files in a project. For using our CVS, set following environmental variables:

```
CVSROOT="$USER@anna.fi.muni.cz:/mnt/sda1/CVSroot"
CVS_RSH=ssh
```

Then get the complete CVS tree of DiVinE using `cvs checkout divine` in directory, where you want to have it placed.

Basic usage of CVS:

- `cvs update -dPA` ... updates recursivly the current directory of CVS (-d = add new directories, -P = delete empty directories, -A = do not care of any tags, take really latest version).

  All files are added (letter U), changed accodring to latest changes in CVS (letter P), merged local and latest CVS changes without problems (letter M and warning about merging) or merged with conflicts (letter C). In case of conflicts it is necessary to edit the file to merge conflicting parts (highlighed with >>>>>>>) manually. Files having letter M by their left side during update are locally modified files not checked in CVS.

- `cvs add [file1] [file2] ...` ... adds files or directories to CVS

- `cvs commit [file1] [file2] ...` ... checks files in CVS (copies changes in local copies of files to CVS). Comment for change is needed.

Remember: It is always better to do `cvs update -dPA` before `cvs commit`

For more informations see `info cvs`, `man cvs` or one of many `tutorials on usage of CVS`.

1.2.2 Structure of CVS

Our CVS has a fixed structure, which should not be broken even through many people have rights to do it. Please contact project leaders before adding of new directories, if you are unsure where to put them.

- **bin** - directory where executables are placed after compilation (both binary and scripts) - it should be initially empty
- **doc** - documentation:
  - **web** - sources of DiVinE's web:
    - **content** - XML files containing the content of web pages
    - **layout** - XSL stylesheets converting XML to the final HTML form
  - **tutorials** - documentation describing various topics in DiVinE containing detailed descriptions, explanations, advices and methods.
  - **refs** - reference manuals for various parts of DiVinE. The most important is probably reference manual for DiVinE Library. These reference manuals are generated from source codes
- **dwi** - sources of DiVinE Web Interface
- **examples** - sources of example models and LTL properties possible to verify by DiVinE Tool Set
- **lib** - directory containg Makefile for compilation of DiVinE Library. Binaries of library are placed here
- **src** - directory contating source codes od DiVinE Library. Directly in this directory there are placed header files of library.
  - **common** - units containg classes and functions commonly used by other parts of DiVinE Library. It mostly contains technical stuff (making implementation easier or more effective)
  - **distributed** - classes regarding distributed computing
  - **por** - classes for partial order reduction of state space
  - **storage** - classes implementing management of storing of states
  - **syntax_analysis** - classes, functions and LALR grammar for reading in of DVE sources
  - **system** - classes implementing the representation of running system

- **support** - tools and settings for developers of DiVinE
  - **etc** - settings of utilities used only by developers (e. g. Doxygen)
  - **src** - sources of supporting utils
    - **debug** - scripts for easier debugging
    - **local** - scripts strongly dependent on the machine, where web pages and CVS of DiVinE are placed. They enable publishing of releases, documentation etc.
    - **makeutils** - scripts for creation of templates of Makefile.am files and auxiliary scripts and macros for Autoconf/Automake
    - **regression_tests** - tests of various parts of DiVinE. They should protect DiVinE from being published broken.
- **tool** - sources of tools built on DiVinE Library. Every single model checking algorithm or more complex utility should have its own directory. Directories containing more than 1 utility are:
  - **model_manipulation** - contain utilities processing sources of models
  - **utils** - unsorted utilities

1.3 Compilation of Sources

DiVinE uses compilation system based on `Automake`/`Autoconf`. In this section, the configuration of these tools for DiVinE is described and explained.

1.3.1 Beginning

If you have a copy of CVS repository (either using `cvs checkout` or `cvs copy`, you can compile it from a scratch using this sequence of commands:

```
./autogen
./configure
make
```

Note:

Script `configure` creates Makefiles from their Automake/Autoconf templates. It tries to detect configuration of system, paths to various programs needed for compilation etc. It has many useful command line arguments. Some of them are DiVinE-specific - namely:

```
--enable-checks
--enable-debug
--enable-non-cvs-compilation-flags
--disable-gui
```

Important: Defaultly sources are compiled without any optimization. To turn the optimization on, just run `configure` with argument `-enable-non-cvs-compilation-flags`

See `./configure -help` for the detailed description.

1.3.2 Basic Configuration Files

- `configure.ac` ... Main configuration file of Autoconf, it is also read by Automake to find out Makefiles to create
- `aclocal.m4` ... created by `aclocal` utility of Automake (using configure.ac) - contains macros supplied by Automake to extend macro set of Autoconf (in fact it is a little hack of Automake developers needing better functionality of Autoconf)
- `acinclude.m4` ... all included M4 macros (for Autoconf purposes) from directory `support/src/makeutils/`
- `Makefile.am` ... local configuration files for Automake. Needed everywhere you want to create Makefile. First, it is translated to `Makefile.in` by Automake and then `Makefile` is created by script `configure` (created by Autoconf)
- `Makefile.am.global` ... in DiVinE this file is included into every Makefile.am. It contains common definitions of variablesi (prefix of executables, flags for C++ compilation, paths to common data files, etc.)

1.3.3 Special Cases of Compilation

In this section some special cases of compilation, which are not fully integrated into Autoconf/Automake compilation system, are covered.

1.3.3.1 Translation of Syntax Analyzer

Directory `src/system/dve/syntax_analysis` contains files `dve_grammar.yy`, `dve_lexer.l` and `dve_keywords.gperf`. They have to be translated into corresponding C/C++ sources. It is not made using Autoconf/Automake, because a local Makefile is easier to create in this case and the translation to C/C++ is needed very rarely. Resulting C/C++ sources are stored in CVS and they are also a part of the distribution package.

Sources are translated using `divine/src/system/dve/syntax_analysis/Makefile`. This Makefile translates:

- `dve_lexer.l` to `dve_lexer.cc`
- `dve_keywords.gperf` to `dve_keywords.cc`
- `dve_grammar.yy` to `dve_grammar.cc`, `dve_expr_grammar.cc`, `dve_trans_grammar.cc` and `dve_proc_grammar.cc` ... LALR(1) syntax analyzers for systems, expressions, transitions and processes. Different outputs are caused by special substitutions applied to `dve_grammar.yy`.

1.3.3.2 Compilation of pml2s.jar

In next paragraphs *JAR* denotes directory `tool/model_manipulation/jar` and *SOURCES* stands for `tool/model_manipulation/nips_compiler`.

Directory *JAR* contains file `pml2s.jar`, which is compiled from Java sources in *SOURCES*`/src`. A Makefile used for compilation arises from `Makefile.am` in *JAR*.

File `pml2s.jar` is stored in CVS and its recompilation is needed, if Java sources in directory *SOURCES* change. The change of Java sources is not detected by the Makefile - recompilation has to be invoked manually by execution of `make` in directory *JAR*.

The Makefile changes the current directory to *SOURCES* and uses a Makefile in it to compile all Java sources. Then it builds `pml2s.jar` file using `jar c`.

Note:

In order to compile Java sources, libraries from *JAR* are always temporarily copied to temporary directory *SOURCES*`/lib`. They are deleted from there, if compilation finishes successfully.