What's New in Alcatel-Lucent nmake

February 2011

Outline

Recent nmake new features and enhancements:

- Baserules / Utilities
- Engine
- General
- Java support
- Eclipse support
- Build reporting/analysis/diagnostics
- Where to Find More Information

Enhancements to Baserules / Utilities

- Enhancements to versioned shared libraries (nmake 11)
- OS Version dependent probe files (nmake 10)
- Improved -I compiler flag generation (alu3.9)
- Support for recent Solaris C/C++ compilers (nmake 10)
- Support for recent tools/platforms (lu3.8, alu3.9)
- New utility: astutil (nmake 11)

Enhancements to Engine

- W3C conformant time zone designator (nmake 11)
- New recurse message (alu3.9, nmake 10)
- Variable edit operator :P=VL (alu3.9)
- Option to suppress umask change (lu3.8)
- Long --name Style Options (nmake 12)
- Support for DOS-style text makefiles (nmake 10)

Capture of Triggered Target Explain Data

Provides enhanced build diagnostics by capturing the reason each target is triggered.

- Introduced in nmake 12
- Records reason target is triggered in state file
  - Gives access to explain info "after-the-fact“
- Reasons recorded in structured build log
- New variable edit operators to access data
  - :T=QE - query explain message
  - :T=QEU - query unformatted explain data
  - :F=%(explain)S - format raw explain data
- lib/make/explain.map contains mapping of explain codes

Use –lef options to list explain info from state file

```
$ nmake –lef Makefile.ms
/* Explain */
hello :
  hello.h [Feb 14 13:44:17 2011] has changed [Feb 14 13:44:05 2011]
hello.o :
  hello.h [Feb 14 13:44:17 2011] has changed [Feb 14 13:44:05 2011]
```

Java support enhancements

- jdk 1.4.2 class names (nmake 10 / jdeps alu2.2.4)
- Performance improvements (alu3.9)
- Remote java build directory (lu3.8)
- Support to filter #empty files (lu3.8)
- Example makefiles
  - The following makefile builds all java packages under com.

```
JAVAPACKAGEROOT = $(VROOT)/java
:JAVA: com
```

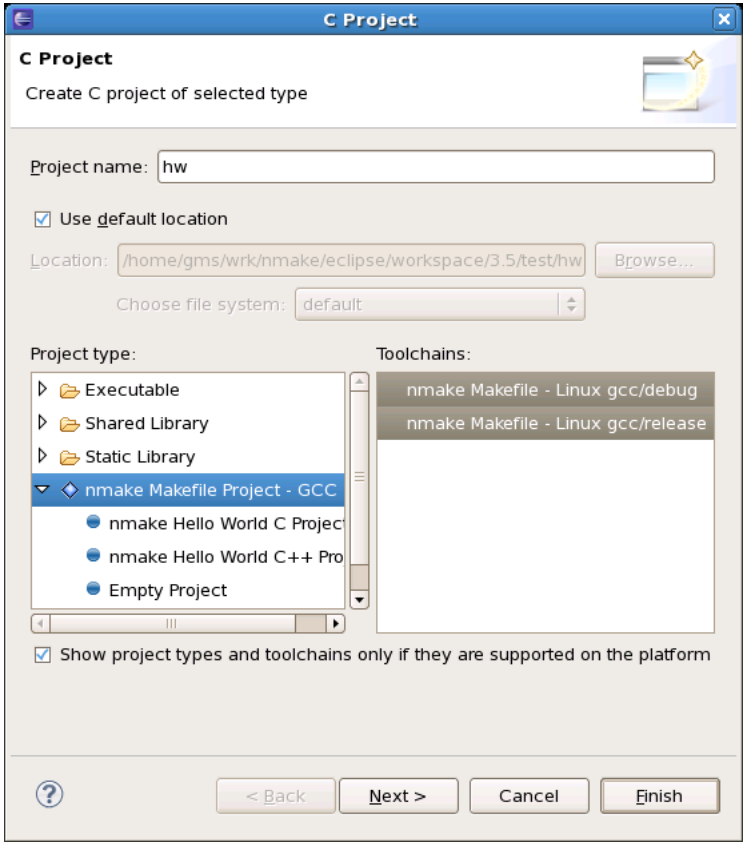
  - The following makefile creates a jar of all class files under com.

```
:ALL:
abc.jar :JAR: com/*.class
```

nmake Eclipse CDT Plugin

The nmake Eclipse plugin supports use of nmake in Eclipse CDT

- nmake/CDT tool chain definitions
  - Configure nmake as the C/C++ project build tool
  - Support for gcc/g++ on Solaris/Linux
- Fully functional multi-level C and C++ project templates
- “Make Targets View” allows build from lower levels in the hierarchy
- Eclipse correctly populates “problems view” using alternate GNU compatible directory recurse message format
- Plugin contains an integrated helpbook and is distributed using the nmake update site



Eclipse CDT project creation wizard showing nmake project types and tool chains.

See http://www.bell-labs.com/project/nmake/manual/eclipse/help_cdt_plugin/gettingstarted.html for more information.

Structured Build Log

Captures additional build data in an XML formatted build log. Captured data includes:

Per target data

- Job start and stop times
- Exit code
- Name of triggering rule
- Reason rule triggered

Per Makefile data

- Makefile start and stop times
- VPATH, PATH
- Makefile Name
- Execution host
- nmake version

Structural data

- Delimited output for each triggered target
- Hierarchical relationships of jobs and Makefiles

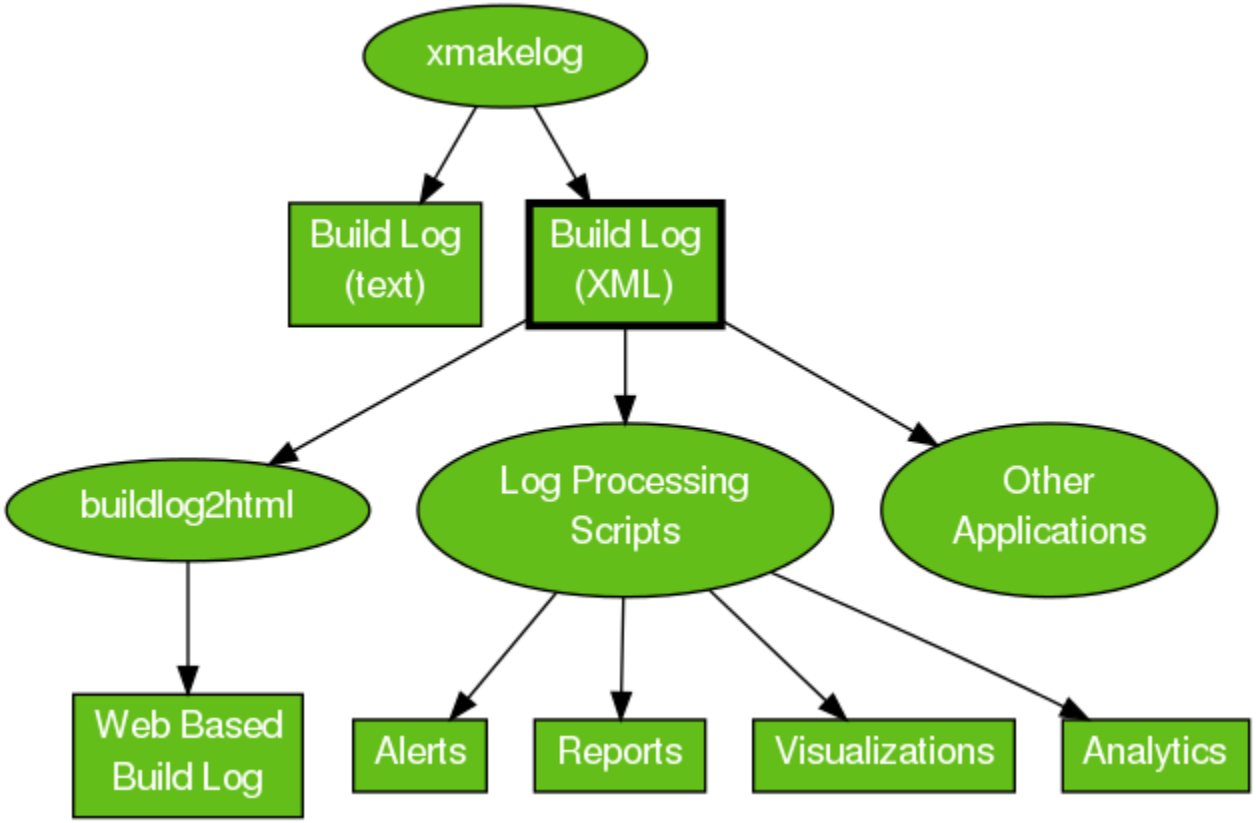
Build metadata (information describing the build)

- Build start and stop times
- Build arguments
- User defined (such as build-id)

Log is easily extended by user to capture any data accessible in rule and job shell execution contexts.

http://www.bell-labs.com/project/nmake/manual/buildlog.html

Build Log Data Flow



xmakelog command is drop-in replacement for “nmake”

Web Based Build Logs

Command `buildlog2html` transforms a structured build log into an interlinked collection of web pages providing overview and detailed views of a build.

- High level views show overall hierarchical structure of build
- Links provide drill-down to details for each triggered target
- Color coding is used to highlight failed targets
- Indices provide fast navigation to failed targets
- Multi-build indices are automatically generated
- http://www.bell-labs.com/project/nmake/manual/buildlogs/xyz/nightly-2010072

Script-Based Build Reporting / Analysis Tools

- Set of small stand-alone scripts analyzing structured build logs
- Currently available reports include: plain text error report, graphical job/Makefiles durations reports, hierarchical build reports including sunburst
- Results are output in plain text, HTML, and graphical formats
- High-level scripting allows quick prototyping
  - Useful data extraction possible in 4-6 line scripts
- Any modern scripting language can be used
  - Example scripts in several scripting languages are available
  - Good support for XML, JSON, date/time manipulation desirable
- Currently using Python for prototyping build processing scripts
- http://www.bell-labs.com/project/nmake/tools/reports.html

Tool Repository

- Web based collection of small standalone tools and scripts
- Allows to publicize tool availability and get tools out to users fast
- Facilitates quick experimentation and prototyping
- Encourages user customization and tool extension
- Encourages tool sharing
- Current contents include scripts for processing structured build logs
  - Demonstrate log processing techniques
  - Provide simple common infrastructure simplifying log processing
  - Includes script producing informative plain text build error report
- http://www.bell-labs.com/project/nmake/tools/

Where to Find More Information

- nmake home page:

  http://www.bell-labs.com/project/nmake/

- Link from nmake home page: Releases and Downloads
  - Release notes

  http://www.bell-labs.com/project/nmake/download.html

- Link from nmake home page: Documentation & Resources
  - Links to pages with more information on specific features:
    Eclipse support, Java support, Structured build logs, Tool Repository

  http://www.bell-labs.com/project/nmake/documentation.html

www.alcatel-lucent.com