nmake Release lu3.2

Outline of talk:

- nmake overview
- Where to find information
- nmake and year 2000
- nmake release lu3.2 new features
- Bug fixes and enhancements
- Features for future releases

This talk may be found at:
http://www.stc.lucent.com/nmake/vgraph/lu3.2
(Lucent intranet) and also at
http://www.bell-labs.com/project/nmake/vgraph/lu3.2 .

Contact address: nmake@Lucent.COM

nmake Overview

nmake is a software build tool used to manage the construction of software products from program source. nmake is:

- Mature, robust
- Widely used
- Scalable to large projects
- Feature rich
- Flexible/configurable
- Available on variety of platforms
  - Currently UNIX® only

nmake Features

Important features include:

- Viewpathing
- Parallel and distributed execution
- State retention
- Automatic implicit dependency generation
- High level build specification language
- Built-in knowledge of build procedures for many platforms, languages and compilers
- Customization on project/sub-project basis

Where to Find Information about nmake

http://www.stc.lucent.com/nmake or http://www.bell-labs.com/project/nmake
Especially:

- How to get technical support
- Ordering information
- Training, tutorial
- Release notes for lu3.2
- FAQ for up-to-date technical support information (includes compiler issues such as templates and viewpathing)
- nmake customer support newsletters
- nmake User's Guide and Reference Manual; updated for lu3.2 (downloadable)
- Downloads and availability by platform
- Y2K information

nmake and Year 2000

Using a non-compliant version of nmake could result in the inability to use nmake to build your software products.

- Only the following versions of nmake are compiled and certified for Year 2000:
  1. *nmake lu3.2* (all versions)
  2. Year 2000 version of *nmake 3.1.2* (binary version string dated 07/01/1998 or later)
- Year 2000 versions of nmake may be downloaded from http://www.stc.lucent.com/nmake/download.html or http://www.bell-labs.com/project/nmake/download.html
- See nmake year 2000 FAQ for complete nmake Year 2000 statement

nmake lu3.2 New Features

nmake lu3.2 (released May, 1999) provides many bug fixes, enhancements, and several new features:

- Output serialization for concurrent jobs
- Atom dependency reporting tool
- Local probe file support
- Support of additional instrumentation tools
- New **`recurse_begin_message`** and **`recurse_end_message`**
- New **`.ACTIONWRAP`** special atom
- ~50 bug fixes and feature enhancements

All changes are backward compatible with 3.x Makefiles.

Output Serialization for Concurrent Jobs

- nmake supports concurrent job execution through the **`-jn`** option
- nmake supports distributed execution via **`COSHELL=coshell`** (where **`coshell`** is the network shell server)

⇒ **Problem:** Output lines appear in the order written, so output of commands with multi-line output is intermixed.

The serialization feature introduces a new make log file (default **`makelog`**) in which:

- Output from each shell action block is contiguous
- Output from each recursive make is (recursively) contiguous
- Output from *nmake* itself is appropriately positioned
- Real-time output continues to appear on **`stdout`**

Output Serialization - Operation

The nmake output serialization feature:

1. Tags each line as it is written with its action block
2. Tags each action block with its parent block
3. Reassembles lines into action blocks back on the nmake host
4. Hierarchically composes action blocks along with sibling blocks

2 new utility executables introduced--**`taglines`** and **`logfilter`**.

To get serialized output, use **`nmakelog`** instead of **`nmake`**:

```
nmakelog -j6 -f mymakefile.mk
```

Output Serialization Example

Makefile:

```
CC = CC
m :: m.c a.c b.c c.c -l++
```

Command: `nmakelog -j6`

Real-Time Output (stdout)

```
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c m.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c a.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c b.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c c.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
CC m.c:
CC a.c:
CC c.c:
CC b.c:
cc -Xs -xs -xildoff -c -O m.c
cc -Xs -xs -xildoff -c -O a.c
CC[ptcomp]: waiting to lock repository ./ptr
cc -Xs -xs -xildoff -c -O c.c
cc -Xs -xs -xildoff -c -O b.c
+ CC -O -I-D983D2B92.0.3CC -I. -I- -I. -o m
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
cc -Xs -xs -xildoff -Wl,-L/apps/tools/sparc
Instantiating List...
Instantiating List...
Instantiating lnnk_ATTLC...
Instantiating lnnk_ATTLC...
```

Serialized Log (makelog)

```
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c m.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
CC m.c:
cc -Xs -xs -xildoff -c -O m.c
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c a.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
CC a.c:
cc -Xs -xs -xildoff -c -O a.c
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c c.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
CC c.c:
cc -Xs -xs -xildoff -c -O c.c
+ CC -O -I-D983D2B92.0.3CC -I. -I- -c b.c
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
CC b.c:
CC[ptcomp]: waiting to lock repository ./ptr
cc -Xs -xs -xildoff -c -O b.c
+ CC -O -I-D983D2B92.0.3CC -I. -I- -I. -o m
+ cppC=/tools/nmake/sparc5/lu3.2/lib/cpp
cc -Xs -xs -xildoff -Wl,-L/apps/tools/sparc
Instantiating List...
Instantiating List...
Instantiating lnnk_ATTLC...
Instantiating lnnk_ATTLC...
```

Atom Dependency Reporting Tool

Dependencies in nmake may originate in several ways:

1. Explicitly declared
2. Set up by assertion operators
3. Induced through meta-rules
4. Discovered during scanning
5. Asserted during **`.MAKE`** rule execution
6. Asserted by nmake engine

Except for (1), these dependencies are not obtainable by simply reviewing Makefile source.

The Atom Dependency Reporting tool obtains this information from MAM (Make Abstract Machine) files.

MAM Translator Commands

These commands translate MAM output to specified formats:

mamdep	Translate to simple textual format
mamdag	Translate to **dag** format
mamdot	Translate to **dot** format
mamdt	Translate to **dt** format

The following options are accepted by all MAM translators:

-a"shell_pattern"	Ignore atoms matching specified shell pattern, along with the direct descendants of each matched atom
-t"shell_pattern"	Ignore atoms with attributes matching the given shell pattern, along with the direct descendants of each matched atom.

Example:

```
mamdep -a"*.m[os]" -t"implicit*" < mamoutput
```

Dependency Generation Example

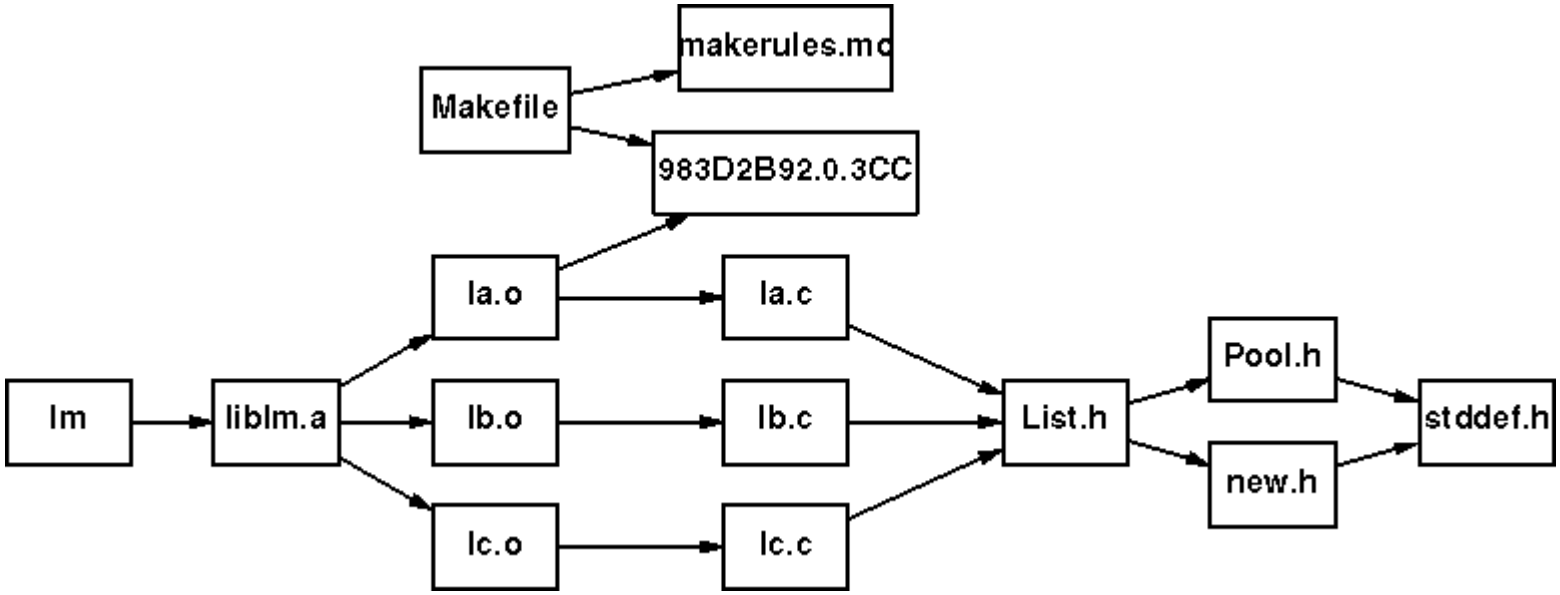
Makefile:

```
CC = CC
lm :LIBRARY: la.c lb.c lc.c
```

Commands:

```
nmake -x -Mdynamic:mamfile::/ lm
mamdot <mamfile >mam.dot
dot -Tps mam.dot >mam.ps
```

Result*:

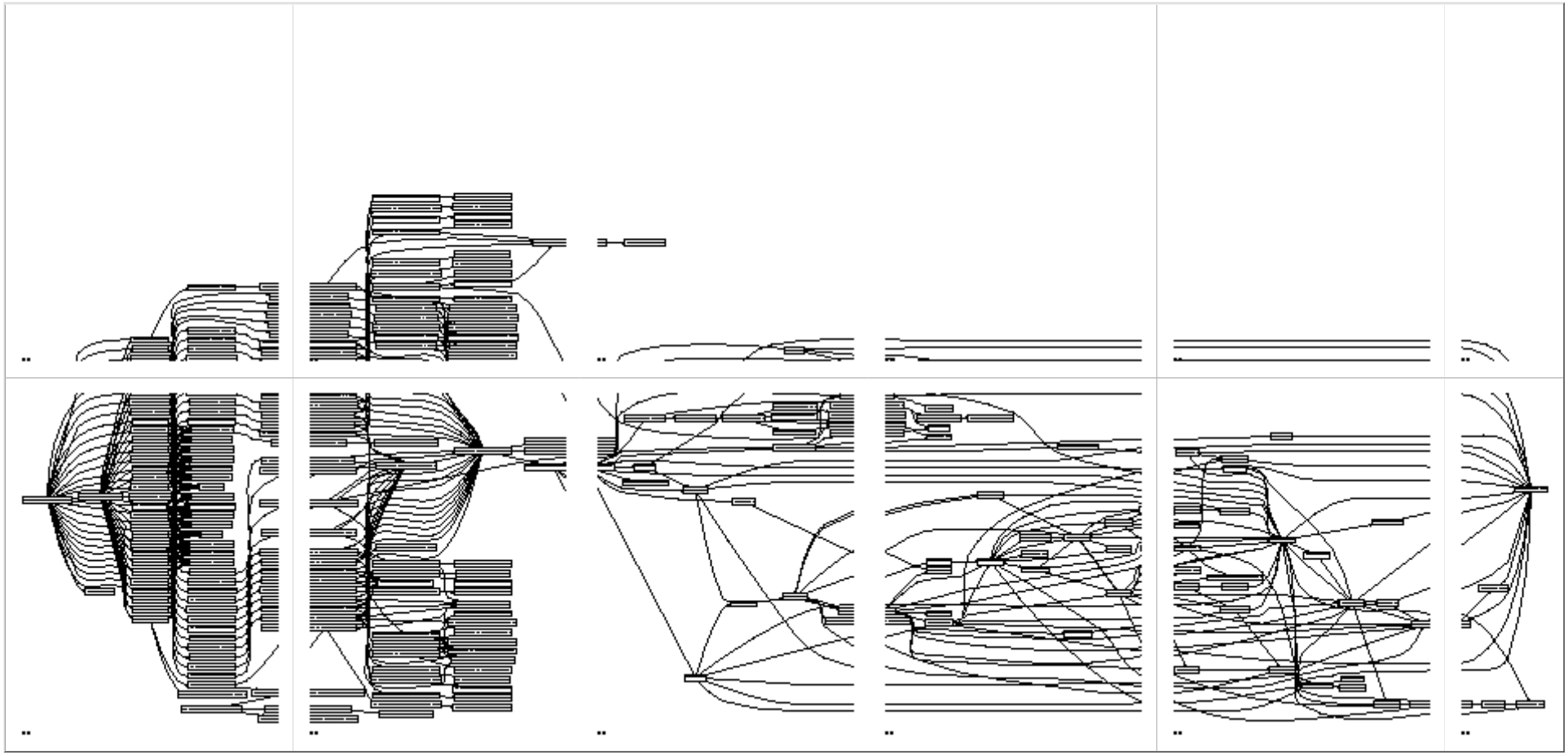


*simplified for presentation purposes

Realistically Sized Dependency Example

Commands used:

```
nmake -x -Mdynamic:mamfile::/ install
mamdot <mamfile >mam.dot
dot -Tps mam.dot >mam.ps
```



Local Probe File Support

Previously, probe information was located under the nmake install root.

The new local probe file feature:

- Allows probe information files to be placed anywhere
- Removes restrictions in customizing probe files
- Applies to both probe generation shells and probe information files

New base rule variable **`localprobe`** determines location of probe files base path. Relative offset is unchanged (e.g. **`lib/probe/C/make|pp`**).

Value of `localprobe`	**Effect**
`vpath`	Node relative to current **`VPATH`**
non-null	Node within **`PROBEPATH`**
null	Locate under nmake install root

Support of Instrumentation Tools

nmake now supports the following packages:

insight	quantify	purecov	purify
sentinel	insure	codewizard	

As before, instrumentation is controlled with the **`instrument`** base rule variable. For example:

`instrument=sentinel`	Build targets with sentinel package. Searches **`/usr/add-on`**, **`/usr/addon`**, or **`/usr/local`** for tool.
`instrument=/tools/sentinel`	Specify tool location.

New Messages for Recursive Makes

- New base rule variables control output for recursive makes

  `recurse_begin_message` **`recurse_end_message`**

- Example

Top-level makefile:

```
targ1 :MAKE: dir_one dir_two
```

Makefiles in dir_one and dir_two:

```
default:
    : $(<) $(PWD)
targ1:
    : $(<) $(PWD)
targ2:
    : $(<) $(PWD)
```

Run nmake:

```
$ nmake recurse targ1 \
recurse_begin_message="Entering into Directory" \
recurse_end_message="Leaving Directory"
```

Output:

```
Entering into Directory dir_one:
+ : targ1 /home/dma/dir_one
Leaving Directory dir_one
Entering into Directory dir_two:
+ : targ1 /home/dma/dir_two
Leaving Directory dir_two
```

New .ACTIONWRAP Special Atom

`.ACTIONWRAP` special atom allows easy modification of all rules in an nmake execution.

- Use to specify action to be expanded in place of original action
  - nmake behavior unchanged if **`.ACTIONWRAP`** not defined
- Expansion takes place in original rule context
- All automatic variables refer to original rule values
  - Allows testing of target attributes
- No new syntax introduced
  - Pre-existing **`$(@)`** variable expands to action for original rule

.ACTIONWRAP Example

Contents of Makefile:

```
if modify_rules
.ACTIONWRAP :
      : "starting $(<:A=.COMMAND:?command ??)target $(<) at `silent date`"
      $(@)
      : "target $(<) completed at `silent date`"
end
CC = CC
a :: a.c
```

Run nmake:

```
$ nmake
+ CC -O --prelink_copy_if_nonlocal -I- -c a.c
+ CC -O --prelink_copy_if_nonlocal -o a a.o

$ nmake clobber
+ ignore rm -f a.o a Makefile.mo Makefile.ms

$ nmake modify_rules=1
+ : starting target a.o at Tue Jun  8 13:34:03 EDT 1999
+ CC -O --prelink_copy_if_nonlocal -I- -c a.c
+ : target a.o completed at Tue Jun  8 13:34:04 EDT 1999
+ : starting command target a at Tue Jun  8 13:34:04 EDT 1999
+ CC -O --prelink_copy_if_nonlocal -o a a.o
+ : target a completed at Tue Jun  8 13:34:06 EDT 1999
```

Bug Fixes and Enhancements

nmake lu3.2 contains ~50 bug fixes and enhancements. See nmake lu3.2 release notes for a complete list.

- **`cc-`** now supports +*opt* and multiple options
- **`:LINK:`** now supports dynamic source file generation
- **`:MAKE:`** now allows targets to be recursively built
- **`:MAKE:`** supports prerequisites of form **`dir/file.mk`**
- **`print`**, **`:F`** now accept **`%[eEfFgG]`** conversion

Improved support for several compilers:

- HP aCC now supports **`-I-`**
- Full support for template archive libraries using the new Lucent C++ 5.0 `--one_instantiation_per_object` feature, improved 4.1 support
- Increased support for native preprocessors supporting **`-I-`**
- Support for **`.cpp`** and **`.cxx`** files in the base rules

Possibilities for Future Releases

Product features

- Probe enhancements
- coshell/genlocal enhancements
- Java support
- Improved C++ support
- IDL support
- Serializer enhancements

Support improvements

- Defined patch scheme
- Improved FAQ area