Quick User Manual for the NIME Pre-Processor

12 February 2001
William M. Fawley - LBNL - (fawley@lbl.gov)

Introduction

NIME (**NI**fty **M**acro **E**xpander) is a relatively simple Tcl/TK script which accepts a standard ASCII format file as input and then, after scanning the file for various macro definitions, expands any macros in text to produce a final output file. The basic goal of NIME is to allow a user to generate multiple blocks of text from repeated use of a relatively few numbers of heavily commented macros. The idea for NIME sprang out of a Friday afternoon conversation with Bob Palmer of BNL with both of us bemoaning the difficulty of generating error-free ICOOL input decks and also noting that a large portion of the text lines which determine the ICOOL transport lattices are repeated over and over again. Moreover, ICOOL input decks permit little or no annotation via comments which makes then extremely difficult to analyze days or months after their original generation. Hence, we both felt that a simple utility which could expand text macros within a simple, heavily commented ASCII input files could be very useful. NIME, while "targeted" toward the generation and maintenance of ICOOL input decks, will actually work upon any ASCII file. Thus, NIME could prove useful, to give an example, for generating a series of input files for batch runs at NERSC in which only a few lines or variables might change from run to run.

Running NIME

The Tcl (Tool Command Language) scripting language and its GUI counterpart TK are freely available for all major platforms(e.g. UNIX, LINUX, WIndowa, MacOS) and may be obtained from the following Web URL: `http://www.scriptics.com/software/` . NIME itself is defined by the Tcl/TK script file `nime.tk` and, for non-Unix platforms, generally must run under the program `wish`, the TK shell interpreter. On Unix/Linux platforms, one can put a shell command calling `wish` in the first line of `nime.tk` to make it directly runnable (*i.e.* by simply typing **`nime.tk`**).

Here is a synopsis of the basic sequence of events one would normally follow to generate and use an ICOOL input deck: 1. In one's favorite editor, put together an ASCII NIME input file which would include the needed macro definitions, macro calls, and ICOOL input fields (see the following pages for further details) 2. In a terminal window, type: **`wish nime.tk`** . The main NIME window should then appear (see Fig. 1). 3. Via the "Read Input" button, read in the NIME input file. 4. Make additional changes in the text frame (if needed) and then, via the "Expand Macros" button, have NIME expand the macros at the appropriate places in text and place the results into a new text frame. 5. In the expanded macro text frame, make any desired changes and then hit the SAVE OUTPUT button to write the text to a resulting ICOOL input text file. 6. Via a soft link or an alias, make the resultant input deck synonymous with FOR001.DAT and run ICOOL in the normal fashion.

Detailed Explanation

Since NIME is a Tcl/TK script, one should make sure Tcl/TK is installed and that the directory containing the `wish` executable appears in one's "path" on Unix/LINUX system (or else one will get a "command not found" message when trying to execute `wish`). At present, nearly any version of Tcl/TK should be compatible with the nime.tk script; certainly any version at or beyond 8.0 should work just fine.

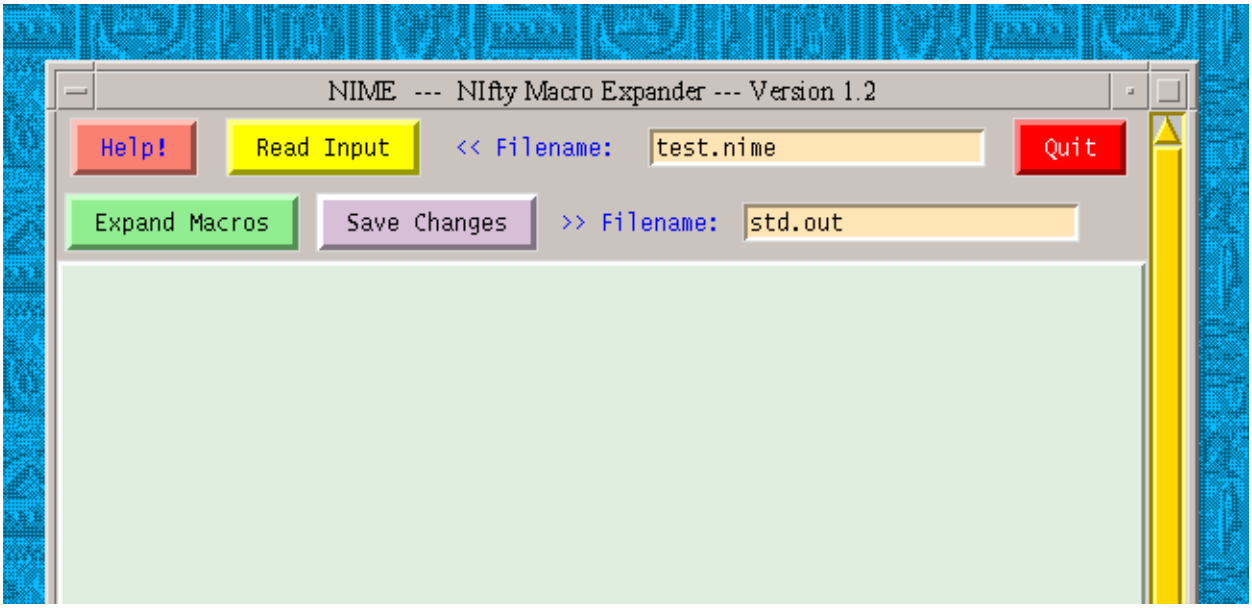

Fig. 1: Main NIME window before reading input file

NIME Input File

NIME seeks an ASCII input file (default name `test.nime`) as shown in the upper right text box (just to the left of the red QUIT button; see Fig. 1) of the main NIME window. This filename can be replaced by placing the cursor within the text box and then typing the appropriate keystrokes. Note that nearly all NIME text boxes and window are editable. Once one has changed the filename to that wanted, hitting the yellow "Read Input" button causes NIME to scan the file and places the contents into an editable text frame in the lower part of the main NIME window (see Fig. 2)

If one desires, one may then edit the interior text to make additional changes. Many of the standard EMACS key bindings are supported by TK text frames, such as CTRL-a, CTRL-e, *etc*. These changes propagate when the green "Expand Macros" button is hit. However, the changes are native to a NIME buffer only and will *not* be saved to disk until the purple "Save Changes" button is pressed upon which time the present state of the visible text will be written to a file whose name is given by the far right text box.

Input File Comments and Annotations

Comments are signaled in the NIME input file by an exclamation point (`!`) and may appear on isolated lines by themselves or on "shared" lines preceded by non-comment text. Comments may also appear within macro definitions and expansions. For example:

```
!  this is a comment on a line all by itself
\accel_macro  rf_freq=1.e9   !--- this is a GHZ RF cavity
CELL   !-- this cell contains apertures and thick foils
```

After NIME scans the input file, comments are highlighted in blue in the main NIME window (see Fig. 2). During macro expansion, once an exclamation point is encountered by the NIME parser, the remaining text on that line is ignored and is not echoed to the output expansion window.

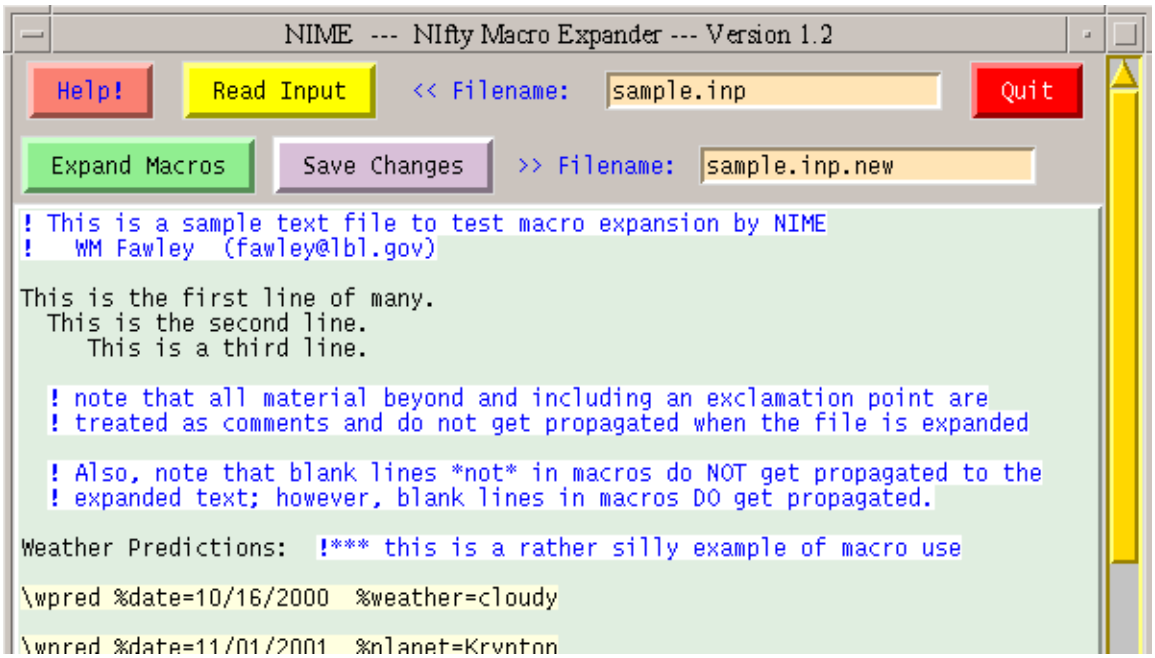

Fig. 2: Main NIME window after reading input file `sample.inp`.

Macro Definition Rules and Interior Variables

Rules concerning macro definitions are as follows. All macro definitions must begin on a new line that starts with **`\def`** immediately followed by a space and then the macro name, *e.g.* **`\def RF`** . The macro name is case-sensitive and should include only "normal" alphanumeric characters and not special symbols such as asterisks, dollars signs, ampersands, percent signs, pound signs, forward or backslashes, exclamation points, *etc.* These limitations are generally due to NIME's use of Tcl's pattern searching functions. There should be *no spaces* within a macro name; underscores, however, are permitted. Following the **`\def`** line should be lines containing the full text of the macro. FInally, a new line beginning with **`\enddef`** terminates the macro definition. See Fig. 3 for some examples.

Each macro definition may contain one or more "variables" whose individual names are preceded by a percent symbol (%). The individual characters of the variable name have the same restrictions as those of macro names (*i.e.* no special characters and no spaces within the variable name). A given variable may occur more than once within the same macro definition. A *default value* may be assigned to a variable (to be used when the macro is expanded) by *immediately* following the variable name by a parentheses-enclosed value, *e.g.* **`%freq(120.e6)`**. However, only *one* such default value is permitted for any particular variable in a given macro definition - the same

default value will be used for all instances of the variable within a macro expansion. The same variable name (and different default values) may appear in different individual macro definitions but effectively each instance is completely unique to each macro and unknown to other macros. In effect, the macro variables are similar to individual elements of self-contained (and completely private) C-structures or Fortran90 types. However, this name reuse is not advised and could become problematic in future NIME versions.

A macro definition may appear anywhere in the text body of the NIME input file. In particular, a macro may be used in the text before its definition actually appears. This is possible because NIME first scans the input file for macro definitions and then makes a second scan to expand the text, including any macro use. Macro definitions are highlighted in red in the main window while macro instances are backlighted in white (see Fig. 3).

```
\costf  %currency=dollars

\wpred  %weather=thunderstorms  %planet=Jupiter

!..begin macro definitions:

\def wpred   ! this is a comment on the def line of a macro

The predicted weather for %date(today) is %weather(rain) . ! macro body comment
The planet %planet(Mars) will rise on the following date.

\enddef


\def costf

Muon colliders may be costly beasts.
Cost could be minimized if built at %location(India) and paid for in %currency(P
ounds) .
%location is also thought to be quite a pretty location.
\enddef

\def RF
RF section %name
length = %length(1.0)   voltage = %voltage(0.0)
type(2)   %freq   0. 0. 0. 0. 0. 0. 0. 0.
1  0. 0.
\enddef

!..end macro definitions
```

Fig. 3 Sample input file macro definitions as they would appear in main NIME window

Macro Use

To actually use a macro (*i.e.* have it expanded in the text), one places the macro name preceded by a single backslash, *e.g.* **\RF** , at the beginning of a line (use anywhere else will probably lead to bizarre results!). If the macro definition included variables, one may override default values (if any) to these variables by including their name (preceded by a % symbol), an immediately following equal sign, and then the wanted value on the *same* line as the macro call. For example, if has a macro named **RF** as in Fig. 3, to call this macro and set the RF frequency to 360 MHz, one would write:

```
\RF  %freq=360.e6
```

If a later one needed 720 MHz RF, one would type:

```
\RF %freq=720.e6
```
Hence, it is most useful to use the "variable" feature of a macro for those quantities that often change from one wanted instance to the next. Within the final expansion of a given macro call, all expansions of a particular individual variable will result in the same value.

Comment-only and blank lines *not* contained within macro definitions are "eliminated" during expansion and are not echoed to the expanded text window (see Fig. 4). On one hand, this allows one to use blank lines liberally in the NIME input file to improve readability and is also compatible with ICOOL's requirements. On the other hand, if NIME is being used for another purpose and a blank line is needed in the final output, one can either add it manually by editing within the "expanded macros" window or by defining a special macro whose body consists of a single blank line which would be expanded unchanged for each call.

.

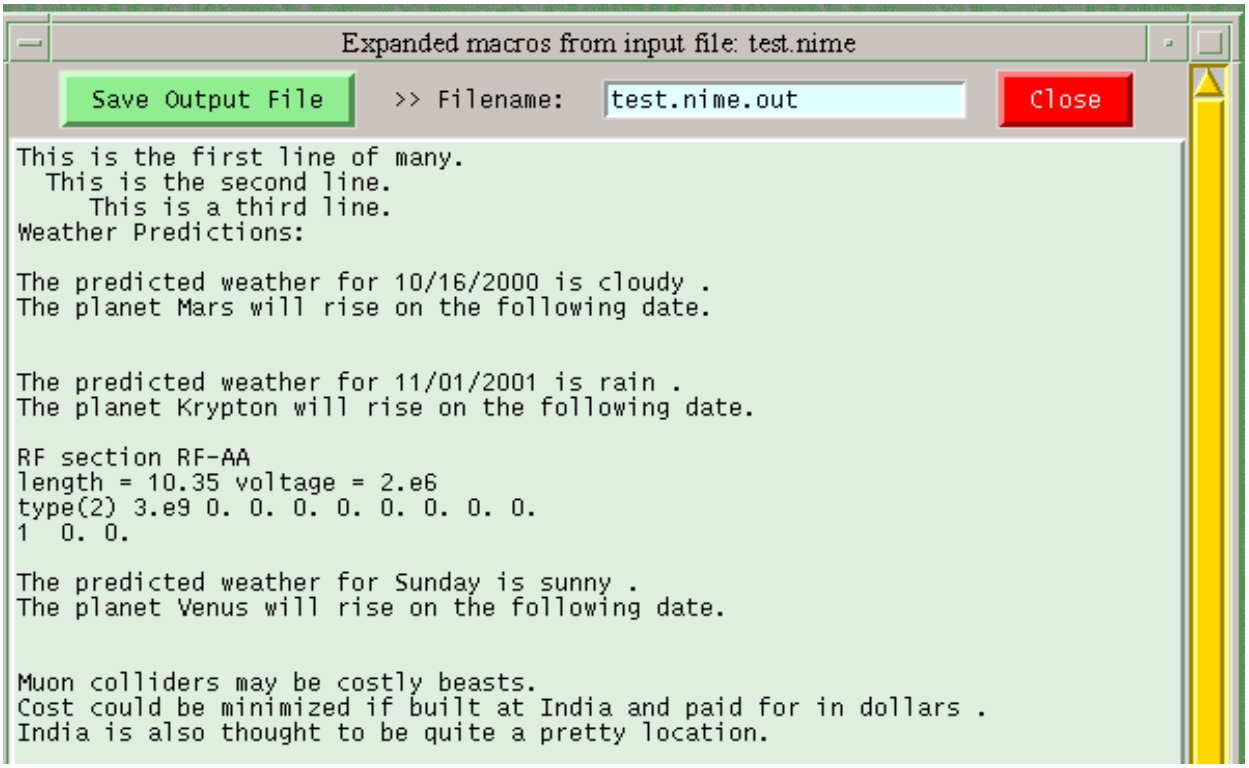

Fig. 4: "Expanded macros" window resulting from expansion of text in Fig. 2 window

Macro Expansion

Once one is satisfied with the interior text contents of the main window, the text may be expanded by hitting the green "Expand Macros" button to the left. At this point, NIME will parse the macro definitions, remove comments and blank lines, expand macro calls when encountered, and display the resultant text in a new "expanded macros" window (see Fig. 4). Lines similar to the following should appear in one's active terminal:

```
scanning macro definitions in file: test.nime
now expanding macro definitions in input file: test.nime
```

If NIME finds errors in the text to be expanded (usually due to mistyped macro names and/or macro variables), it outputs additional lines to the terminal flagging the error and its location in the main NIME window:

```
--> ERROR: line # 30
.......could not find macro definition: rf_frew
```

NIME's error detection facilities are primitive and there are probably quite a few ways to cause bizarre behavior and/or output. One particularly way is to have a line which is completely blank except for a non-printing character (such as CTRL-a). This line will expand to an apparently blank line (but the CTRL-a is there) which is problematic for ICOOL in particular.

The text in the expanded macro window is editable. Hence, one may make some quick and dirty changes if necessary (although it is probably better to do so in the main window and then save to a new NIME input text file using the purple "Save Changes" button). Once all appears OK, hitting the green "Save Output File" button will save the text to a file with the name appearing in the cyan text box to the right of the label >> *Filename:* . The default name is that of the NIME input file with a new suffix of ".out" but the user can change it to any system-permissible value. NIME does not check to see if the chosen name already exists on disk and will thus overwrite such a pre-existent file with absolutely no warning to the user. *Caveat emptor...*

Note that in Fig. 4 all of the macro definitions and comments of the original NIME input file (see Figs. 2 and 3) are gone and that the various macro calls with any associated variables have been expanded.

NIME Customization

Those with a nerdish streak can customize their own copy of NIME via `nime.tk`. In particular, the color scheme can be adapted to one personal taste. Lines that call TK widgets (such as text boxes) which contain the phrase `-background` *`color`* or `-foreground` *`color`* can be edited to substitute one's own colors. Similarly, window and button titles can also be easily modified. Unless one is reasonably adept at Tcl scripts, changing the parsing routines is probably quite ill-advised.