Including Literate Programs in Forth

17 December 2011

1a ⟨*origin file* 1a⟩≡ (7b)

```
literate-included.lyx
```

1b ⟨*version* 1b⟩≡ (7b)

```
1.1
```

1c ⟨*copyright* 1c⟩≡ (7b)

```
\ Copyright (c) 2010-2011, Krishna Myneni
\ The software given here may be used for any purpose,
\   provided the copyright notice, above, is preserved.
```

1 Introduction

Literate programming is an approach to organizing and writing programs, large or small[1]. This approach was devised and first implemented by Donald E. Knuth. One of the most useful features of literate programming is that it allows the author to strongly couple the description of a computational procedure with the code most closely associated with carrying out the procedure. Two immediate benefits of writing or maintaining a computer program in this way are: 1) the code for the given task may be examined by a human reader, for formal correctness, without having to search within a larger procedure to identify the appropriate code, and 2) the close coupling of code with its description clues the programmer to choose appropriate names for data and procedures. The focus of literate programming on describing and writing small fragments of code, called *code chunks*, mirrors the approach of developing programs in Forth, in which a set of short words are defined to accomplish specific tasks[2].

Literate programs are more work to write, because the software design, documentation, and programming activities are integrated. This forces the author(s) to document more clearly, logically, and completely, the set of tasks to be accomplished by a program. Another apparent drawback is that a more complex chain of tools is needed to develop and write programs, and, at the end, the actual computer program must be separated from the integrated document and assembled in the correct sequence for compilation or interpretation. However, in this author's experience, the one-time burden of tools installation and configuration, plus the added work necessary to do literate programming is more than made up by the resulting improvement in the quality of code, reduction in number of bugs, and reduction in time for comprehending, analyzing, and maintaining the code at a later date.

In this literate Forth program, we provide code for allowing convenient import of literate Forth programs into a Forth system. That is, we enable the Forth system to treat documents containing literate programs as source files in themselves. Procedures will be given for generating Forth source files from the native plain text formats of two literate programming tools: LyX [3] and Noweb [4]. The progam, `literate-included.fs`, given in 3.5, provides two words for transparently including both LyX format and Noweb format files from the Forth environment:

```
nw-included  ( caddr1 u1 caddr2 u2 -- ... )  \ include a noweb file
lyx-included ( caddr1 u1 caddr2 u2 -- ... )  \ include a lyx file
```

2 Literate Programming Tools

2.1 Noweb

Noweb is a collection of related literate programming tools developed by Norman Ramsey[4]. We will use the `notangle` tool for extracting Forth code from a

literate Noweb document. A Noweb file is simply a text file containing named code chunks and documentation chunks. The code chunks may appear in any order necessitated by the logical description and documentation of the code, and they will be assembled using their name references to produce the final program. In a Noweb file, typically having an extension, `.nw`, a code chunk starts with the name declaration, and is followed by the code,

```
<<name of chunk>>=
actual code...
```

while the start of a documentation chunk is indicated with the `@` symbol on a line by itself,

```
   @
   In order to switch to a new task, the system must save
the stack pointer ...
```

Inside the Noweb file, the actual source output file may be denoted by giving the file name as the name of the chunk, followed by code and/or named code chunks to be inserted at the given location in the output file, e.g.

```
<<tasker.fs>>=
<<task creation>>
<<task switching>>
<<...>>
```

The `notangle` program is used to extract the `tasker.fs` file from the .nw file in which it is contained. Assuming the noweb file is called `multi-tasking.nw`, we extract the `tasker.fs` file as follows,

```
$ notangle -Rtasker.fs multi-tasking.nw > tasker.fs
```

The above command will produce the Forth source file, `tasker.fs`, which may then be included from a Forth environment in the usual way. More than one Forth source file may be provided within a single noweb file. Furthermore, source files for different languages/compilers may be provided in a single file.

3 ⟨*nw to fs* 3⟩≡ (6b)

```
\ Execute a shell command to extract a Forth source file from a Noweb file
( anw u1  afs u2 ) 2>r
s" notangle -R%f2.fs  %f1.nw  >  %f2.fs"
s" %f2" 2r@  replace  s" %f1" 2rot replace  s" %f2" 2r@  replace
shell 2r> rot ( afs u2 n )
```

2.2 LyX

LyX is a document processing environment for use in editing and generating LaTeX documents[3]. LyX can be used to embed Noweb code chunks within a document, and thereby allow generation of a Noweb file, via the LaTeX style sheet, `noweb.sty`. Thus, a LyX document can provide the intelligent, *content-based* typesetting capabilities of LaTeX, while allowing piecewise construction of a program throughout the document. Such a literate source file may also be created using a plain text editor and LaTeX with Noweb alone[5]. However, the LyX environment provides several advantages for writing documents containing literate programs. These include a graphical view and interface for editing the document – should your document contain mathematical notation, math symbols, equations, images, or other graphical content, they will be visible within the editor, without generating the typeset output. In addition, LyX provides version control, via `rcs` [6], and change tracking features which can be useful for managing software development.

A LyX file is also a plain text file, and LyX can export its native format to a noweb file. In addition to exporting to noweb from the GUI, the same can be done via the command line:

```
$ lyx -e literate multi-tasking.lyx
```

4a ⟨*lyx to nw* 4a⟩≡ (7a)

```
( afname u )
s" lyx -e literate " 2swap strcat s" .lyx" strcat shell ( n )
```

Now, we can see from our previous discussion of Noweb files how to extract a Forth source file (or any other source file) from a `.lyx` format file, using two commands from the shell. The Forth code for extracting a Forth source file from either a noweb file (`.nw`) or a LyX file (`.lyx`) is given in section 3.

3 Forth Code

3.1 String Utilities

For execution of shell commands, it is necessary to build up command strings by concatenating commands and filenames. We use the lightweight Forth-94 compatible dynamic strings library, `strings.fs` [7], which provides the LMI-compatible dynamic string words, `STRPCK` and `STRCAT` [8].

4b ⟨*include strings library* 4b⟩≡ (5b)

```
[undefined] strcat [IF]  s" strings.fs" included  [THEN]
```

Also useful is the word, `REPLACE`, to replace an existing pattern within a string by the replacement pattern, and return the new string. In the following definition of `REPLACE`, only the first occurrence of the pattern string will be replaced in the search string – successive calls with the same pattern and replacement pattern may be used to replace multiple occurrences.

5a ⟨*string replace* 5a⟩≡ (5b)

```
[undefined] 4dup   [IF] : 4dup  2over 2over ; [THEN]
[undefined] 4drop  [IF] : 4drop 2drop 2drop ; [THEN]

\ Search the string, $str, for the pattern, $pat. If found,
\ replace $pat with $rep, and return the new string, $new.

: replace ( $str $pat $rep - $new )
    2>r        ( $str $pat ) ( r: $rep )
    4dup search
    if                 ( $str $pat $sub )   ( r: $rep )
      2rot 2over       ( $pat $sub $str $sub ) ( r: $rep )
      drop nip over -  ( $pat $sub $left ) ( r: $rep )
      2r> strcat       ( $pat $sub $left+$rep )
      2>r 2swap nip /string ( $right ) ( r: $left+$rep )
      2r> 2swap strcat ( $left+$rep+$right )
    else               ( $str $pat $sub )  ( r: $rep )
      4drop 2r> 2drop  ( $str )
    then ;
```

5b ⟨*string utilities* 5b⟩≡ (7b)

⟨*include strings library* 4b⟩
⟨*string replace* 5a⟩

3.2 Shell Command

Since there is no standardized way to pass commands to the shell in Forth-94, various Forth systems typically provide the word, `SYSTEM`. However, the semantics of `SYSTEM` are not identical among Forth sytems which provide this word. In addition to executing a shell command, the exit code is also needed to recognize when errors have occured. We therefore define the word `SHELL`, having consistent semantics across several Forth systems. `SHELL` passes a command string to the host environment for execution, and returns the exit code. We will assume an exit code of zero indicates no error, and any non-zero value indicates abnormal execution of the command.

5c ⟨*gforth shell* 5c⟩≡ (6a)

```
: shell ( caddr u - retcode )  system $? ;
```

6a ⟨*shell command* 6a⟩≡ (7b)

```
\ auto selection for those systems which identify themselves

[DEFINED] gforth   [IF] ⟨gforth shell 5c⟩     [THEN]
[DEFINED] bigforth [IF]
  also dos  : shell  strpck system ;  previous [THEN]
[DEFINED] vfxforth [IF]
  Extern: sys-command int system ( char * ); ( cmd - r )
  : shell  strpck 1+ sys-command ; [THEN]

\ manual selection for other systems

[UNDEFINED] shell [IF]
  1 [IF] ⟨gforth shell 5c⟩                [THEN]  \ gforth (older version)
  0 [IF] : shell  strpck  system         [THEN]  \ kforth
  0 [IF] : shell  system  RETURNCODE @ ; [THEN]  \ iForth
  0 [IF] : shell  system ;               [THEN]  \ pfe
[THEN]
```

For manual system selection, the default is set to Gforth.

3.3 Including a Noweb File

The procedure of extracting a Forth source file from a Noweb file, and subsequently including the Forth source file is provided by the word, `nw-included`.

6b ⟨*noweb included* 6b⟩≡ (7b)

```
\ Extract the Forth source (.fs) from a Noweb (.nw) file.
\ Return the full .fs filename.

: untangle ( anw u1 afs u2 - afs2 u3 )
    strpck count 2>r  strpck count 2r>
        ⟨nw to fs 3⟩ abort" Unable to extract Forth source file!"
    s" .fs" strcat ;

\ INCLUDED for a Noweb file
: nw-included ( anw u1 asrc u2 - ... )  untangle included ;
```

The arguments to `nw-included` are two strings, the Noweb filename and the Forth source file name. We assume that the files have the extensions, `.nw` and `.fs`, respectively, and therefore the filename arguments to `nw-included` do not include the extensions. File extensions are appended automatically. As an example, to extract and include the Forth file, `tasker.fs` , from the Noweb file, `multi-tasking.nw`,

```
s'' multi-tasking'' s'' tasker'' nw-included
```

3.4 Including a LyX File

The work of extracting the Forth source from a Noweb file is performed by the word given in the previous section. Therefore, we only need to convert a LyX file to a Noweb file, and execute `nw-included` to provide the word, `lyx-included`.

7a ⟨*lyx included* 7a⟩≡ (7b)

```
: lyx>nw ( alyx u1 - anw u1 retcode )
        strpck count
        2dup s" .nw" strcat DELETE-FILE drop
        2dup ⟨lyx to nw 4a⟩ ;

\ INCLUDED for a LyX (.lyx) file.
: lyx-included ( alyx u1 asrc u2 - )
    strpck count 2>r
    lyx>nw  abort" Unable to convert a lyx file to a noweb file!"
    2r> nw-included
;
```

Similar to `nw-included`, the arguments to `lyx-included` are the two strings containing the LyX filename and the Forth source filename, respectively and without extensions. Therefore, if we wish to extract and include the Forth file, `tasker.fs`, from the LyX file, `multi-tasking.lyx`,

```
s'' multi-tasking'' s'' tasker'' lyx-included
```

3.5 literate-included.fs

The program, `literate-included.fs`, is assembled below.

7b ⟨*literate-included.fs* 7b⟩≡

```
\ This file is generated using LyX and Noweb - Do Not Edit!
\ Please make modifications to the original file, ⟨origin file 1a⟩
\ Version ⟨version 1b⟩
⟨copyright 1c⟩
⟨string utilities 5b⟩
⟨shell command 6a⟩
⟨noweb included 6b⟩
⟨lyx included 7a⟩
```

4 System Requirements

4.1 Forth System

Our Forth code for including literate programs directly from the Forth environment assumes a standard Forth-94 system with two augmentations:

1. the Forth system provides a word, typically called `SYSTEM` , for executing system commands via the shell, and a method to obtain the exit code of the executing process.

2. strings may be defined with `S''` in the system's interpreter.

The only assumption made about the persistence of `S''` strings defined within the interpreter is that two such strings may be sequentially defined, without overwriting the same memory. If the intended Forth system cannot support these extended features of `S''` strings, it should not prove too difficult to modify the code given here to make it work within the limitations of the Forth system's string capabilities.

4.2 Noweb Version

The program `literate-included.fs`, and its predecessor, `lyx-included.fs`, has been used with Noweb versions 2.11a-2 and 2.11b.

4.3 LyX Version

We have developed literate programs using Noweb under LyX versions 1.5 and 2.0. LyX does not come pre-configured to work with Noweb, and some manual installation is necessary for the two systems to work together. Please see the Appendix for instructions on configuring LyX to work with Noweb.

5 Revision History

2011-02-16 km; Created `lyx-included.fs` based on an example given by Bernd Paysan on `comp.lang.forth`, February 16, 2011. See message id, `<ijipt8$28n$1@news.m-online.net>`.

2011-02-17 km; Make use of `strings.fs` for string handling; remove existing `.nw` file; require two arguments to `lyx-include`: the lyx filename and the Forth source filename.

2011-02-18 km; ported to kForth; replace use of `OPEN-PIPE` with generic `SHELL` command; used `INCLUDED` instead of `INCLUDE-FILE`; no requirement for persistence of `S"` strings.

2011-12-15 km; Created literate version of program `lyx-included.fs`; renamed to `literate-included.fs`; removed use of string variables; added References; updated LyX configuration procedure.

2011-12-17 km; Further factored definitions of `nw-include` and `lyx-included`; added string `REPLACE` word; used `DELETE-FILE` instead of shell command for portability; added definitions of `SHELL` for bigForth and VFX Forth (untested); added ref. to Peter Knaggs' paper; now at ver 1.1.

References

[1] Wikipedia article on *literate programming*: http://en.wikipedia.org/wiki/Literate_programming

[2] P. J. Knaggs, Literate Programming in Forth, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.4869 (1995).

[3] *LyX – The Document Processor*, http://www.lyx.org.

[4] N. Ramsey, *Noweb – A Simple Extensible Tool for Literate Programming*, http://www.cs.tufts.edu/∼nr/noweb/.

[5] N. Ramsey, *A One-Page Guide to Using noweb with LaTeX*, http://www.cs.tufts.edu/∼nr/noweb/onepage.ps.

[6] *Official RCS Homepage*, http://www.cs.purdue.edu/homes/trinkle/RCS/

[7] K. Myneni, *Gforth Version of kForth String Utility Words*, ftp://ccreweb.org/software/gforth/strings.fs.

[8] These and other dynamic string words were provided in Laboratory Microsystems, Inc.'s 32-bit UR/Forth for MS-DOS. Compatible definitions are given in `[7]`.

Appendix: Configuring LyX to Work with Noweb

We assume that LyX and LaTeX are already installed on the user's system. The following procedure to configure LyX to work with Noweb was pieced together from various sources on the web.

1. Install the Noweb package. If it is not pre-packaged for your OS or distribution, it may be built from source, found on the Noweb home page[4].

2. The LaTeX style sheet, `noweb.sty` , should be placed in the appropriate directory for use by LyX and LaTeX. On a Linux system, this directory is typically, `/usr/local/share/texmf/latex/` . The style sheet is included in the Noweb package, and may be automatically installed in the correct directory.

3. Run `texhash` as superuser, so that LaTeX knows about the `noweb.sty` stylesheet.

4. Start LyX. Using the menus, perform *Tools* > *Reconfigure*.

5. Exit and restart LyX.

6. If you wish to use version control under LyX, install the `rcs` package [6] on your system. Version control functions for a document (registration, check out, etc.) may be performed from the menu *File* > *Version Control*.

You are now ready to do literate programming in Forth, or in any other language, using LyX and Noweb!