Ten steps to begin using the ELM on a unix file system

This works under Solaris 2.5-7 using Sun Forte Developer 6 C compiler (and earlier Sun products). The code adheres to ANSI C standards, and should work under K&R C standards (its original standard). To our knowledge, there are no significant changes necessary to create and use the executable and associated utility scripts under linux or other unix-based OS's, nor is it a problem to use gnu compilers etc (we've only recently discontinued using gcc). Regarding non-solaris systems, there are several linked libraries that may be specific to Solaris (see note below). The executable is not provided in the file archives in anticipation that the user should build the executable so that the model is native to their architecture.

1) **Create a directory** with the name "SME" (Spatial Modeling Environment) anywhere in the user's file system. Beware that in creating the ELM project, two new directories will be created one level below the SME directory. Upon completed the described installation, the fundamentals of your new directory structure will be as listed below, along with a number of subdirectories within all of these:

```
./include/          {see makefile for include needs}
./lib/                    {see makefile for lib needs}
./SME/SMDriver/     {ELM source codes}
./SME/scripts/            {ELM scripts}
./SME/Projects/     {ELM input data and output location}
```

2) **Set up your unix environment** with the following (put in .cshrc or equivalent if desired):

```
%prompt%>setenv SMD_HOME /YourRootDir/YourDir/SME
```

3) **Change directory** to the newly created "SME" ($SMD_HOME). Place the two archive files and the "ELMcreate" script in this directory. Thus, the following files must be in this directory:

```
./SME/ELM.src.tar.gz        {source C codes & scripts}
./SME/ELM.data.calib.tar.gz     {input data}
./SME/ELMcreate             {executable script}
```

4) **Run the ELMcreate script** *while still in the shell* that has SMD_HOME defined. ELMcreate will extract all code and data necessary for building and running the ELM (using dir structure shown above – the program exits if you have pre-existing ./include, ./lib, or ./sfwmm).

```
%prompt%>ELMcreate
```

5) **Edit the makefile** to suit your compiler and to provide the location of the lib and include directories:

```
path to makefile: ./SME/SMDriver/Sources/Driver_Sources/
%prompt%>vi Driver.make.Serial.Sun
```

{Within vi, a) Modify the "LIBCOMM" and "INCCOMM" path variables to point to the lib and include dirs just created (or use your own equivalents). b) Modify the "C" variable to match your C compiler name. c) Optional: set any optimization switches (CCOptions) as appropriate to your compiler}

6) **Further set up your unix environment** with the following (put in .cshrc or equivalent if desired):

```
%prompt%>set path = ( {$SMD_HOME}/scripts $path )
```

For executing scripts and running the model, you must remain in the shell that has SMD_HOME (and script path) defined.

7) **Compile and link** the ELM to make an executable with the "build" script.

```
%prompt%>build ELM
```
{we're aware of a variety of warnings from Generic_Driver.c and UnitMod.c – ignore them}

8) **Prepare to make a simple test run** of the ELM by checking the runtime parameters and output requests via the "Check" script.

```
%prompt%>Check ELM
```
{sets runtime and output parms}
{at the prompt for RunTime parms, answer "y" to invoke vi to edit the Driver.parm file. 1) Set N_iter to a low (ca. 99) value; 2) set the output path to /your_SMD_HOME_path/Projects/ }
{at the prompt for the output data, answer "n" to just stick with existing output requests}

9) **Make a simple test run** of the ELM from the plain-vanilla "go" script.

```
%prompt%>go ELM
```
{simple run, w/o file manipulations}

10) **Make a couple of basic checks** on some text output files, assuming the model terminated normally . An editor that does NOT wrap lines is highly recommended, as files such as the budgets are many-columns wide.

```
%prompt%>cd $SMD_HOME/Projects/ELM/Output/
%prompt%>xemacs
```
{FileOpen ./Debug/Driver1.out. You should not find any string with "ERROR" in it}

```
%prompt%>xemacs
```
{FileOpen ./Budget/budg_Wcm1. You should find cumulative SumErr values on the order of 0 to a few microns of depth for most basins}

```
Use any spatial analysis tool (such as a GIS) to view
the multiple binary (1 byte, unsigned integer/char)
surface water depth files in
$SMD_HOME/Projects/ELM/Output/Animation1/
```