Software Release Notice
Acquired Software

Installation was performed on new computer -

1. Software Name: MCNP - A General Monte Carlo N-Particle Transport Code
   Software Version: 5
   Acquired from Radiation Safety Information Computational Center (RSICC), a Department of Energy Specialized Information Analysis Center (SIAC), Oak Ridge National Laboratory (ORNL) as software packages MCNP5 (code proper) and MCNP5DATA (data libraries for MCNP5)

2. Software Function: Los Alamos National Laboratory developed code to be used in radiation transport evaluation in arbitrary 3-dimensional configurations of materials. It is directly applicable to radiation shielding and criticality safety evaluations.

3. Summary of Actions:
❑ New Software X Update to Existing Software ❑ Software Retirement

4. Software Installation

4a. Computer Platform(s): Pentium III or later PC
4b. Operating System(s):Windows operating system (95/98/NT/2000/XP/ME)
4c. Programming Language(s): Fortran 90

4d. Installation Testing:
X Passed
Testing Performed On: Paladin machine
Description of Testing Performed: The installation test was conducted according to the code installation instructions provided on the installation CD. The test is automated and implemented by running runprob.bat (Run test suite command) file that initiates an automated testing sequence. The sequence includes execution of 42 test cases, comparison of results and generation of dif* files documenting encountered differences. Examination of all dif* revealed no differences between results provided by the software developer and results obtained by execution of MCNP 5 on Paladin machine. Input, output and dif* files are stored on Paladin machine in C:\programs\lanl\mcnp5\Installation directory. The test cases cover wide range of problem types and are relevant to the types used by CNWRA staff. The files involved in installation tests are on the attached CD.

4e. Archive Copy:
X Enclosed ❑ Not Available, Why:

Installation Performed by: IMS	Date: 01/2005

Remarks:

5. Software Assessment

Validation Status: NRC Endorsed [illegible] 2/11/05
X ~~Full Validation~~ ❑ Limited Validation Date of Validation: 02/11/2005
❑ Not Validated, Explain:

Software User: Oleg Povetko. Additional users may obtain this software under a free individual license as an NRC contractor via the Radiation Safety Information Computational Center (RSICC) at http://rsicc.ornl.gov	Date: 02/11/2005

Remarks:	
6. Approval	
Manager: NA	Date:
Remarks:	
7. QA Verification	
SRN Number: NA - MCNP5 has been released in 12/8/03	
[illegible]	Date: 2/11/05

Remarks: Validation is in accordance with CNWRA TOP-018, Revision 9, Section 4.2.2 for acquired software approved or endorsed by the U.S. NRC for use in regulatory reviews. NRC endorsement and approval of the MCNP code package is documented in a number of NRC Standard Review Plans and Regulatory Guides.

The reference documents here are NUREG-1536, "Standard Review Plan for Dry Cask Storage Systems", Spent Fuel Project Office, U.S. NRC, Washington, DC, January 1997. and NUREG-1567, "Standard Review Plan for Spent Fuel Dry Storage Facilities," Spent Fuel Project Office, U.S. NRC, Washington, DC, March 2000. NUREG - 1536 Sections 5.4 and 6.4 endorse use of MCNP in shielding and criticality analyses, respectively. NUREG-1567 Sections 7.5.4.1 and 8.4.4.1 endorse use of the MCNP for criticality and shielding analyses, respectively.
The attached list of verification and validation reports from the MCNP code website http://laws.lanl.gov/x5/MCNP/publication/mcnp_publications.html includes a number of Los Alamos National Laboratory documents available at this website in the areas of the code applicability, such as radiation transport, radiation shielding and criticality safety.

Attachments:
1. List of MCNP5 verification and validation documents available on MCNP5 website http://laws.lanl.gov/x5/MCNP/publication/mcnp_publications.html#verification
2. Readme file with installation instructions
2. CD-ROM containing
- File with a list of MCNP5 verification and validation documents MCNP5VerificationAndValidationReportsLANL.txt;
- Files involved in installation tests on Paladin machine
-Readme file with installation instructions

```
README file for MCNP5_RSICC_1.14
-------------------------------

Code name:    MCNP5

Version:      MCNP5_RSICC/1.14

LANL No.:     LA-CC-02-083

Date:         2003-05-05

Contact:      Forrest B. Brown
              MCNP Team Leader, X-5 Group, LANL
              <fbrown@lanl.gov>
              505-667-7581

Contents:
--------

  This distribution contains 3 files:

   1. README
      This file, ASCI text

   2. MCNP5_Install_Guide.pdf
        A PDF file with detailed instructions for installing
        & running MCNP5. This is a Appendix C of the MCNP manual.

   3. MCNP5_RSICC_1.14.tgz
      A gzip'd tar file containing
      * directory MCNP5/Source
          source files for building MCNP5 on all systems
      * directory MCNP5/Testing
          regression test set for verifying correct
          compilation/execution,
      * directory MCNP5/bin
          executabes for MCNP5 and MAKXSF for Linux & Windows,
          and an executable for the Visual Editor (VISED)
          on Windows,
      * directory MCNP5/Manual
          MCNP5_manual_VOL_I.pdf     (LA-UR-03-1987)
          MCNP5_manual_VOL_II.pdf    (LA-CP-03-0284)
          MCNP5_manual_VOL_III.pdf   (LA-CP-03-0245)
          VISED_manual.pdf

  Cross-section files required for MCNP5 usage are NOT
  included in the above tar file.

  ***********************************************************
  Note:  MCNP5 source coding (in file MCNP5_RSICC_1.14.tgz),
  and Volumes II & III of the MCNP5 manuals
  (MCNP5_Install_Guide.pdf, MCNP5_manual_VOL_II.pdf,
  MCNP5_manual_VOL_III.pdf) are EXPORT CONTROLLED.

  All relevant DOE/NNSA procedures must be followed
  in order to further distribute them.
  ***********************************************************


Short Guide to Building/Testing MCNP5:
-------------------------------------

  0. Follow directions in the Installation Guide
     for using the "install" script

  -- or --

  1. On a Unix system, or in a Cygwin shell on a
     Windows PC, un-tar the .tgz file:

        cd /home/fbrown
        tar xvfz MCNP5_RSICC_1.14.tgz
```

This will create a directory MCNP5,
with subdirectories Source, Testing, Manual, & bin

2. Change to the Source directory:

   ```
   cd MCNP5/Source
   ```

3. Ensure that GNU make is available on the system, and that "make" invokes the GNU version of make. (This may involve setting a link or adjusting the PATH variable.)

4. Build & test the code:

   ```
   make
   ```

   The code will be compiled, with the executable:
   MCNP5/Source/src/mcnp5
   (with a ".exe" extension on PCs) The default compilation is sequential. See the installation guide for details on selecting options for plotting &/or parallel operation.

   Then 42 test problems will be run, with a summary of diffs at the end.

5. See the PDF file MCNP5_Install_Guide.pdf for more details.

Comments on the Installation Test Problem Set:

1. We have generated reference results ("templates") for 3 systems:

a) Unix:	SGI Origin-2000, IRIX64, 1 processor
b) Linux:	Linux PC cluster with Absoft F90
c) Windows:	P-III, 1GHz, Windows-2000, using Cygwin environment, HP/Compaq CVF F90 compiler

2. If compiling on those specific systems, exact diffs are made between the OUTP & MCTAL files and the templates. All diffs should be zero.

3. On other systems or if running with any of MPI/PVM/OpenMP, then "fuzzy" diffs are taken to attempt to account for small, unimportant roundoff differences. Very small floating point numbers are converted to zero, and the last 2 digits are trimmed from floating point numbers. Integers & strings are not changed. Diffs are then. All MCTAL diffs should be zero (except for problem inp41, which depends on processor speed). Diffs for OUTP files should be zero or small (a few 100 or 1000 bytes or less).

4. See the NOTES at the end of the test results summary.

Last-minute Additions:

1. In addition to the computer systems mentioned in the "MCNP5_Install_Guide" file, MCNP5 has been recently ported to:
   a) Mac OS X (10.2.4), with the Developers Toolkit & Apple port of X11, using the Absoft Fortran-90 compiler (Version 8)
   b) Itanium-I, using the Intel Fortran-90 compiler (Version 7.1)
   c) Linux, using the Intel Fortran-90 compiler

2. On these new systems, all tests ran satisfactorily. Users are cautioned, however, that testing on these systems is still ongoing, and that there may be future changes required to either source coding or the build system. Be sure to check the MCNP5 website & the mcnp-forum mailing list for possible changes.

3. For Linux systems using the Portland Group Fortran-90 compiler: MCNP5 can be compiled correctly using PGF90 v4.0. MCNP5 will not compile & execute correctly using PGF90 v4.1.

APPENDIX C - INSTALLING AND RUNNING MCNP ON VARIOUS SYSTEMS

Some of Appendix C is adapted from project development research notes.[1] The first half of the appendix (Sections I through VIII) addresses the mechanics of configuring, building, and installing the MCNP executable program(s) on both UNIX and PC platforms. The second half (Sections IX through XII) is adapted from a previous version of the MCNP manual and addresses the following topics: MCNP testing, modification, verification, and cross-section file conversion. Section XIII is a list of references.

I. A NEW BUILD SYSTEM FOR MCNP FORTRAN 90 ON UNIX

A new build system has been developed for use with the Fortran 90 version of the MCNP code. This build procedure is based upon the utilization of GNU *make*[2] and the burst file format (i.e., separate files for each subprogram) of the MCNP code.[3] Features of the new build system include a custom configuration utility which can be utilized in several modes to tailor the installation and execution of the code. The new build system is described in this appendix.

The build system previously used for MCNP was based upon a large "ID file" and a combination of scripts and Fortran 77 code.[4] The modernization of the code to Fortran 90, together with the use of burst files rather than large ID files, presents the need to upgrade and modernize the build system. GNU *make* is widely used on many different UNIX platforms to build and execute code.[5,6] GNU *make* may be installed on a system with the name *gmake* or *make*, depending on choices of the system administrator. In this appendix, use of the name *make* is assumed to refer to *gmake.* A version of GNU *make* can also be obtained for Personal Computers (PCs).[7]

Make is a tool for automating the compilation of large amounts of source code. Proper use of *make* reduces time spent compiling programs and guarantees that programs are compiled with appropriate options and linked with appropriate versions of modules and libraries. The *make* facility uses a special *Makefile* to define and describe targets, dependencies, abbreviations and macros, directory searches, and rules to automate the build process. For descriptions of the *make* facility, see References 2, 10, and 11.

With the help of the *make* facility, building MCNP for a variety of hardware platforms becomes easier for the end user. The end user simply types a *make* command, optionally specifying the desired target names and configuration features. As a prelude to issuing the *make* command, an installation script queries users about the relevant characteristics of their environment, then assigns values to special variables that are used in the special *Makefile* files that appear throughout the hierarchical levels of the source distribution.

Using *Makefiles* the new MCNP system can build much faster than the old system for any given platform using a single processor.[12,13] A detailed overview of the build system, with specific focus on the different modes of operation, and a discussion of the testing performed to date on different computer platforms with the MCNP Version 5 (MCNP5) code are included in this appendix.

The work described herein is specific to UNIX systems or UNIX-like environments, such as the Cygwin system for Windows PCs.

II. NEW UNIX BUILD SYSTEM DESCRIPTION

In the MCNP5 distribution, the user will find the two directories, *Source* and *Testing,* under the current working directory. The user should change directories to the *Source* directory, where all functions of the build system can be invoked. Table C.1 summarizes the directory structure and the files included in the MCNP5 distribution. The build system can be invoked simply by typing *./install.* This *install* utility will set up the configuration and invoke *gmake* to build and test the code. If the *install* utility described in the following pages does not meet the user's needs, the build can also be controlled directly with *gmake* as described in Section V.

The following files are provided with the MCNP5 distribution:

Table C.1
Description of MCNP Distribution Directories and Files

Directory	Relevant Files	Description
Source	install	Configuration setup utility, invokes gmake
	Makefile	Top level Makefile, invokes build and test
	answer.${sys}	Answer file generated by running *install* utility
	install.log	Install log file generated by running *install* utility
Source/config	Makefile	Makefile for generating custom configurations
	${OS}.gcf	Operating system specific GNU configuration file where ${OS} represents an operating system name, e.g., SunOS, AIX, IRIX64, OSF1, Linux
	${OS}-modes.gcf	Modes file for building and testing many configurations in series
	${OS}_aux.gcf	Rules for expected Fortran compiler and operating system combination
	Unix_options.gcf	Configuration options that are common to UNIX platforms
	VC_info.gcf	A generated file containing Version Control information from the configuration management repository

MCNP5 Verification and Validation Reports available on MCNP website
http://laws.lanl.gov/x5/MCNP/publication/mcnp_publications.html#verification

Tim Goorley, Richard E. Prael, and Grady H. Hughes, "Verification of Stopping Powers for Proton Transport in MCNP5," Trans. Am. Nucl. Soc., 89, 456 (November 2003).
(LA-UR-03-4446)

Avneet Sood, Michael S. Reed, and R. Arthur Forster, "New Results for Pulse Height Tally Verification in MCNP5," Trans. Am. Nucl. Soc., 89, 456 (November 2003).
(LA-UR-03-4271)

Forrest B. Brown, Russell D. Mosteller, and Avneet Sood, "Verification of MCNP5," Proc. M&C 2003: A Century in Review, A Century Anew, Gatlinburg, Tennessee (April 2003).
(LA-UR-03-0011)

Russell D. Mosteller, "Testing MCNP5$ Version 4.18 with Criticality and Radiation-Shielding Validation Suites," Los Alamos National Laboratory (October 2002).
(LA-UR-02-6532)

F. B. Brown, R. C. Little, A. Sood, and D. K. Parsons, "MCNP Calculations for the OECD/NEA Source Convergence Benchmarks," Trans. Am. Nucl. Soc., 87, 150 (November 2003).
(LA-UR-02-3987)

Avneet Sood, R. Arthur Forster, and D. Kent Parsons, "Analytic Benchmark Test Set for Criticality Code Verification," Prog. Nucl. Energy, 42, pp. 55-106 (2003).
(LA-UR-01-3082)

Russell D. Mosteller, "Validation Suites for MCNP™," Proc. 12th Biennial Topl. Mtg. Radiation Protection and Shielding Div., Santa Fe, New Mexico (April 2002).
(LA-UR-02-0878)

Forrest B. Brown, Robert C. Little, Avneet Sood, D. Kent Parsons, and Todd A. Wareing, "MCNP Calculations for the OECD/NEA Source Convergence Benchmarks for Criticality Safety Analysis," Los Alamos National Laboratory (September 2001).
(LA-UR-01-5181)

Table C.1
Description of MCNP Distribution Directories and Files

Directory	Relevant Files	Description
	custom_${OS}.gcf	User-specific custom configuration file generated by running the *install* utility with appropriate options given for a specific operating system
Source/src	Makefile	Makefile for building MCNP executable program
	*.F90	MCNP Fortran source code files
	mc.c	MCNP C source code file
	FILE.list	A list of all MCNP source code files
	Depends	A dependencies file
Source/datasrc	Makefile	Makefile for building MAKXSF executable program (cross-section file)
	makxsf.F90	MAKXSF Fortran source code file
Source/dotcomm/include	dotcomm.h	Include file for building libdotcomm.a
Source/dotcomm/src	Makefile	Makefile used to build libdotcomm.a
	*.F90	Fortran source for libdotcomm.a
Source/dotcomm/src/ internals/mpi	Makefile	Makefile used to compile C source for libdotcomm.a
	*.c, *.h	C source for libdotcomm.a interface to MPI libraries
Source/dotcomm/src/ internals/pvm	Makefile	Makefile used to compile C source for libdotcomm.a
	*.c, *.h	C source for libdotcomm.a interface to PVM libraries
Testing/Regression	Makefile	Makefile for building the regression tests for MCNP
Testing/Regression/Inputs	testinp.tar	Archive of test set input files in tar format (do not untar)
	testlib1	Type 1 cross-section (XS) for regression testing
	testdir1	XSDIR for regression testing
	specs.1-2	Specification file for type 1 to 2 XS generation
	specs.2-1a	Specification file for type 2 to 1 XS generation (not used by build system, present in Eolus Razor Repository)

Table C.1
Description of MCNP Distribution Directories and Files

Directory	Relevant Files	Description
Testing/Regression/ Templates	testoutp.${OS}	Operating system specific expected output for test problems
	testmctl.${OS}	Operating system specific expected tally output for test problems
Testing/config	test_options.mk	Make macros and options for testing

III. THE UNIX INSTALL UTILITY

The *install* utility, which is the top level component of the build system, can be utilized in the different modes described in Table C.2.

Table C.2
Ways of Invoking the Install Utility

Mode of Operation	System	Code	Version
install	Operating system is detected via uname command in user's current login session	MCNP	5 or 5.mpi or 5.pvm
install <code>	Operating system is detected via uname command in user's current login session	User input	5
install <code> <version>	Operating system is detected via uname command in user's current login session	User input	User input
install <sys>	User input (allowable values for <sys> include sgi, alpha, sun, aix, linux)	MCNP	5 or 5.mpi or 5.pvm
install <sys> <code>	User input (allowable values for <sys> include sgi, alpha, sun, aix, linux)	User input	5 or 5.mpi or 5.pvm
install <sys> <code> <version>	User input (allowable values for <sys> include sgi, alpha, sun, aix, linux)	User input	User input

The install script stores the options chosen by the user in the answers.${sys} file. When the install script is executed, the answers.${sys} file is read to restore the user's options.

Note that the order of arguments on the install command line IS IMPORTANT and must conform to the specifications in Table C.2

The last three options depicted in Table C.2 are useful for generating answer files and custom configuration files for UNIX systems that are different from the user's current login system. For example, a user working at an IRIX64 system might generate configuration files for Sun (OS-SunOS, sys=sun). In this case, the build system would generate the GNU configuration to include file and then exit, allowing the user to examine and edit the configuration files.

Please note that the answer files and custom configuration files perform the same function by setting up a custom configuration file for future reuse. The rationale for this redundancy is to accommodate a wide spectrum of users with varying preferences. This new build system will serve those individuals who prefer to install the code in the more traditional MCNP mode of menu options and answer files, as well as those who prefer to directly control the building and testing of the code through the widely used *gmake* utility.

Since this system is *gmake* centric, it is important to ensure that the user's environment variable for PATH is set correctly. The existence of a correct path can be confirmed by a positive response to typing "*which gmake.*" NOTE: *make* is aliased to *gmake* on some systems.

IV. UNIX CONFIGURATION WITH INSTALL UTILITY

Before starting construction it is wise to check the values of the PATH and DATAPATH environment variables with the UNIX echo command (`echo $PATH, echo $DATAPATH`). All compilers (which f90, which cc) you intend to use must be locatable with the $PATH value. The PATH environment variable must include the current working directory (.) to assure that all items referenced in scripts and Makefiles can be found. The UNIX `whoami` command must be locatable with the $PATH value. Any cross-section directories and libraries you intend to use must be locatable with the $DATAPATH value. Compilers and cross-section files and libraries are referenced in the config/$(OS).gcf, config/$(OS)-modes.gcf, and config/$(OS)_aux.gcf files. Rules associated with file name extensions for the specific compiler used on each operating system platform appear in the config/$(OS)_aux.gcf files.

Table C.3 shows the current key default configuration options for different systems that are included in the current *install* utility.

Table C.3
Default Configuration Options for UNIX Install Utility

Platform	OS	CONFIG=options	Datapath
sgi	IRIX64	seq plot	/usr/projects/data/nuclear/mc/type1
alpha	OSF1	seq plot	/usr/projects/data/nuclear/mc/type1
ibm	AIX	seq plot	/usr/local/udata/mcnpxs
sun	SunOS	seq plot cheap	/usr/local/udata/mcnpxs
intel	Linux	seq plot cheap	/usr/local/udata/mcnpxs
Notes: The HP system has not been configured or tested yet. Cray and Vax are no longer supported.			

In the figures that follow (C-1 through C-6), items tagged with a (toggle) label alternate between the values on/off or the alternate sequentially through a set of values that is listed near the (toggle) label. Items that are tagged with a (menu) label present a sub-menu of choices. The sub-menu works in a similar way to the top level setup menu, i.e., items are tagged. Items labeled (entry) require the user to type in the desired value or path. The examples given in the figures are the result of running the install script on a Sun platform.

Figure C-1. Example Top Level Setup Menu for UNIX Install Script

```
*************************** MCNP SETUP MENU ***************************
****** Type item number and <Enter> to toggle or change an item *******
********** Or type letter and <Enter> to execute the action ***********
***********************************************************************

COMPUTER SYSTEM DESCRIPTION                          sun, SunOS

Configurable Numbered Items                    Current Value
-----------------------------------------------------------
1. 64-bit Consts & Vars (toggle)               off
   32 bit (off) is recommended for Sun, Linux and Windows
   64-bit (on) is recommended for all others

2. Plotting (menu driven)                      on
   Graphics Option                             XLIB
   Graphics Library Path                       /usr/openwin/lib
   Graphics Library Name                       libX11.a
   Graphics Include Path                       /usr/openwin/include

3. CROSS-SECTION DATAPATH (menu driven)        /usr/local/udata/mcnpxs

4. MULTIPROCESSING OPTIONS (menu driven)       seq

5. COMPILER OPTIONS (menu driven)
   FORTRAN 90 Compiler                         f90
   C Compiler                                  cc

6. Generate a Debuggable Version? (toggle)     no
```

```
7. Generate a Post-processed Version? (toggle)   no

8. Comparison with other source files? (menu)    no

9. Test Type 2 Cross Section Data? (toggle)      no

Permitted Letter Actions
------------------------
Z. Compile MCNP and run test problems
C. Only compile MCNP
T. Only run test problems
M. Only generate custom Makefiles
D. Run an automated script using default values and actions
X. Exit without doing anything

NOTE: 1 simultaneous gmake execution(s) will be allowed.
      Set the GNUJ environment variable to the desired number
      of parallel  executions before running this script.
```

Figure C-2. Example Plotting Sub-Menu (Item 2 from Top Level Menu)

```
**************************** GRAPHICS OPTIONS *************************

Configurable Numbered Items                   Current Value
--------------------------------------------------------------
1. Plotting (toggle)                          on

2. Graphics Option (toggle)                   XLIB
                          XLIB
                          LAHEY
                          QWIN

3. Graphics Library Path (entry)              /usr/openwin/lib

4. Graphics Library Name (entry)              libX11.a

5. Graphics Include Path (entry)              /usr/openwin/include

Confirmed:  Graphic Library Path /usr/openwin/lib is valid.
Confirmed:  Graphics Library Name libX11.a is valid.
Confirmed:  Graphics Include Path /usr/openwin/include is valid.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Warning: Changes to the Graphics Library Name or Graphics Library Path!
!         here will not actually change the Name or Path.  You must    !
!         make the changes by hand to PLOTLIBS in config/SunOS.gcf!    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Permitted Letter Actions
------------------------
X. Exit Graphics Options


***************** Type your choice and hit <Enter> *******************
**********************************************************************
```

Figure C-3. Example Cross-Section DATAPATH Sub-Menu (Item 3 from Top Level Menu)

```
**************************** Cross Section Data ***********************

Configurable Numbered Items                   Current Value
------------------------------------------------------------
1. Cross Section DataPath (entry)          /usr/local/udata/mcnpxs

Permitted Letter Actions
------------------------
X. Exit Cross Section Data Options


       Confirmed: Cross Section Datapath verified.

***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

Figure C-4. Example Multiprocessing Options Sub-Menu (Item 4 from Top Level Menu)

```
**************************** Multiprocessing OPTIONS *************************

Configurable Numbered Items                   Current Value
------------------------------------------------------------
   MULTIPROCESSING OPTIONS                    seq

1. Distributed Memory Model (toggle)          seq
      Sequential                  seq
      Distributed Memory  MPI     mpi
      Distributed Memory  PVM     pvm

2. Shared Memory Model (toggle)               none
      No Shared Memory            none
      Shared Memory       OpenMP  omp


Permitted Letter Actions
------------------------
X. Exit Multiprocessing Options


***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

Figure C-5. Example Compiler Options Sub-Menu (Item 5 from Top Level Menu)

```
**************************** COMPILER OPTIONS *************************

Configurable Numbered Items                   Current Value
------------------------------------------------------------
2. Fortran 90 Path (entry)                    f90

3. C Path (entry)                             cc

Permitted Letter Actions
------------------------
X. Exit Compiler Options

***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

Figure C-6. Example Comparison Sub-Menu (Item 8 from Top Level Menu)

```
******************** Source Comparison Options ************************

Configurable Numbered Items                     Current Value
------------------------------------------------------------
1. Comparison with other source files? (toggle)    no

Permitted Letter Actions
------------------------
X. Exit Source Comparison Options


***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

V. UNIX CONFIGURATION WITHOUT INSTALL UTILITY

Before starting construction, it is wise to check the values of the PATH and DATAPATH environment variables with the UNIX `echo` command (`echo $PATH, echo $DATAPATH`). All compilers (which f90, which cc) you intend to use must be locatable with the $PATH value. The PATH environment variable must include the current working directory (.) to assure that all items referenced in scripts and Makefiles can be found. The UNIX `whoami` command must be locatable with the $PATH value. Any cross-section directories and libraries you intend to use must be locatable with the $DATAPATH value. Compilers and cross-section files and libraries are referenced in the config/$(OS).gcf, config/$(OS)-modes.gcf, and config/$(OS)_aux.gcf files (if they exist for the platform). Rules associated with file name extensions for the compiler used on each platform also appear in these files.

We recommend you work in the Source directory of the distribution. Verify the results of the build with the tests located in the Testing directory. The Testing directory contains cross-section files and libraries used with the set of tests in the same directory. Use of cross-sections other than the ones included in the Testing subdirectory (testdir1, testlib1) requires that the environment variable DATAPATH be set to an appropriate cross-section directory and the xsdir=value parameter must be used to specify the cross-section file that is appropriate for the $DATAPATH value. Content of input files must match the contents of the XSDIR that is specified.

Build the executable by issuing the following command:

```
    gmake build
```

This will build a sequential version of MCNP according to the settings found in the $(OS).gcf (see Table C.5) and the corresponding Makefile that includes it. See the commands documented in Table C.6 for more complex examples that specify the CONFIG=values parameter.

The Source and Testing directories should be at the same directory level. From within the Source directory, run the set of tests in the Testing/Regression directory by issuing the following command:

```
gmake test
```

The behavior of the test target depends upon whether or not the environment variable TESTDATA is set. If TESTDATA is not set, only type 1 cross-sections will be used with the tests. If TESTDATA is set, then the tests will be run first with type 1 cross-section data then with type 2 cross-section data.

If you wish to run the tests with only type 1 cross-section data, use the command

```
gmake test1
```

If you wish to run the tests with only type 2 cross-section data, use the command

```
gmake test2
```

Although this runs a variety of differently contrived tests, it does not test the plotting utilities. To run a few simple tests of the plotting utilities, work from the Testing/Regression directory. It is easy to access the cross-section files and directories from there. Create a symbolic link in the Testing/ Regression directory that points to the mcnp executable (`ln -s ../../Source/src/mcnp5 mcnp`). Your path to the mcnp5 executable (the argument after the -s in the ln command) may vary, depending upon how you unpackage the source distribution.

Testing the 3 types of plotting will be done in 3 separate executions of mcnp. In these tests, the type 1 cross-section data is used.These tests assume that you have created the symbolic link, mcnp, as described above.

GEOMETRY PLOTTING [PLOT UTILITY]

Run the executable (mcnp) and specify the input file (i=inp01), the cross-section file (xsdir=testdir1), and options to process input and enable geometric plotting (ip) with the command shown below:

```
mcnp i=inp01 xsdir=testdir1 ip
```

Interact with the plot window that appears according to the instructions it displays.

To exit, click on the item in the plot window labeled *End.*

TALLY PLOTTING [MCPLOT UTILITY]

Run the executable (mcnp) and specify the input file (i=inp01), the output file (o=out01), the tally file (r=rtpe01), and the cross-section file (xsdir=testdir1) with the command shown below:

```
mcnp i=inp01 o=out01 r=rtpe01 xsdir=testdir1
```

Run the executable (mcnp) and specify the tally file where results of the prior run were collected (r=rtpe01), and enable tally plotting (z) with the command shown below:

```
mcnp r=rtpe01 z
```

When the mcplot> prompt shows up, specify the tally (tally 1) and a tally fluctuation chart (tfc=m) with the command shown below:

```
tally 1 tfc=m
```

To exit, give the exit command shown below:

```
e
```

CROSS-SECTION PLOTTING [MCPLOT UTILITY]

In the example that follows, a simple neutron problem is used and the default particle type in use is neutrons (par=n).

Run the executable (mcnp) and specify the input file (i=inp01), the cross section file (xsdir=testdir1), and the options to process input, process cross-section, and enable cross-section plotting (ixz) with the command shown below:

```
mcnp i=inp01 xsdir=testdir1 ixz
```

When the mcplot> prompt shows up, specify appropriate commands and values from the input file inp01 (e.g., material 2 (xs=m1) and reaction identifier 1 (mt=1)) with the command shown below:

```
xs=m1 mt=1
```

Try another plot at the mcplot> prompt (e.g., material 2 (xs=m2) and reaction identifier 2 (mt=2)) with the command shown below

```
xs=m2 mt=2
```

To exit, give the exit command as shown below:

```
e
```

VI. UNIX MODES OF OPERATION

The different modes of operation of the build system and the constituent Makefiles are described below and illustrated in Table C.6.

As mentioned earlier, the *install* utility defines and exports environment variables that are utilized in the Makefile system to build and test the code. In addition to invoking the execution of *gmake* from the *install* utility, it is also possible to control the execution of *gmake* from individual directories and Makefiles to perform specific functions. In particular, the *Source* Makefile controls the overall building and testing of the code, the *Source/src* Makefile controls the building of the MCNP executable, the *Source/datasrc* Makefile controls the building of the MAKXSF executable, the *Source/dotcomm/src* Makefile controls the building of the dotcomm library, and the *Testing/ Regression* Makefile controls the regression testing of the MCNP code.

A. Source Directory

This directory contains the top level Makefile, which, as stated, controls the main functions to build and test the code. These functions include the cleaning of all directories (removing files from previous builds and testing), conditional generation of a custom configuration file, conditional generation of a post-processed version of the MCNP code, building of the MCNP executable, building of the MAKXSF executable, and regression testing of the code. The conditional generation of a post-processed version of the code is set by the *install* utility.

A custom configuration file is only generated when the *install* utility is executed; otherwise, a custom configuration file, from a previous execution of *install,* is used to set the desired environment variables for a user-specific installation. The user is given the option to delete this custom configuration file when using the install utility. If a user desires to use the system defaults, then *gmake* may be invoked at this level without the presence of a custom configuration file. The defaults are included in the standard operating specific configuration file, ${OS}.gcf, which is also included as a Make include file. These defaults are the same as those invoked by the *install* utility, with the exception of the default configuration variable for the IBM AIX system.

In addition, there is a global export of environment variables from this top level Makefile to all the sub-makes that are subsequently called, but only when the *install* utility is executed. An option also exists to build the MCNP executable in parallel, and it is controlled through the environment variable GNUJ. If the user sets a value for GNUJ, then that value will be used to attempt to execute multiple make commands concurrently. If the user has not set a value for GNUJ, the default value of GNUJ is 1. Some installations are limited by single-user compiler licenses.

B. Source/config Directory

CUSTOM CONFIGURATION FILE

The configuration Makefile in this directory generates the custom configuration file based upon the variables that are exported from the *install* utility. As stated, this Makefile is only executed if the *install* utility is exercised; otherwise the custom configuration file is not created, but is included as a Make include file if present. The custom configuration file is called custom_${OS}.gcf, where ${OS} is the name of the operating system the user is logged into and gcf is a file suffix denoting "GNU configuration file". The contents of the custom configuration file are described below in Table C.4 and illustrated in Figure C-7.

Table C.4
Description of Variables Included in the UNIX Custom Configuration File

Variable	Significance
MCUSER	User designation, in UNIX **whoami**
OS	Operating system designation, in UNIX **uname** (e.g., IRIX64)
sys	System designation (e.g., sgi)
CODE	Code name (e.g., MCNP)
VERSION	Code version (e.g., 5.0)
menugra	Graphics option
menugpath	Graphics library path
menuglib	Graphics library name
menugincl	Include path
DATAPATH	Data path
NPVM	Number of PVM processes used for running the test suite
NTRD	Number of threads used for running the test suite
NMPI	Number of MPI processors used for running the test suite
menupre	Option to generate a post-processed version of the code
menucomp	Option for file comparison of post-processed code
compdir	Target directory used in file comparison option
TESTDATA	Option to generate and test type 2 data

Table C.4
Description of Variables Included in the UNIX Custom Configuration File

Variable	Significance
CONFIG	General configuration variable, includes the following options: [plotting multiprocessing cheap debugging]. Multiprocessing keywords are listed in Table C.6. For example, CONFIG=plot seq cheap debug. (On the AIX platform, IEEE is included as an additional option to specify various floating point options, and is used to compare with results from other systems.) (On the LINUX platform, portland, lahey, and absoft are included as additional options to specify the Fortran compiler to be used.) (On the Windows platform, compaq, lahey, and absoft are included as additional options to specify the Fortran compiler to be used. The options gcc, fcc, acc, and cl are included as additional options to specify the C compiler to be used.)
The variables CODE, VERSION, DATAPATH are then incorporated in zc_mod.F90 via C-preprocessor directives.	
premake	Logical switch to tell the Makefile that the install script or the custom_${OS}.gcf file is being used.
FC	Fortran 90 compiler
CC	C compiler
PVM_ROOT	Path to PVM distribution
PVM_ARCH	PVM machine type
MPI	MPI distribution (mpich or other)
MPIFC	MPI Fortran 90 compiler
MPICC	MPI C Compiler

STANDARD SYSTEM-DEPENDENT CONFIGURATION FILE

The standard system-dependent configuration file is called ${OS}.gcf, and contains system-dependent defaults, values of compiler and CPP flags, and specific files which are not used for certain options such as plotting. It is necessary to filter out unused files because some systems will give compiler warnings or errors if a file, after preprocessing, contains no Fortran statements. The key system defaults for the systems tested to date include:

Table C.5
UNIX System Defaults via ${OS}.gcf

System	Default Values
sgi	CONFIG=seq plot DATAPATH: /usr/projects/data/nuclear/mc/type1
alpha	CONFIG=seq plot DATAPATH: /usr/projects/data/nuclear/mc/type1
aix	CONFIG=seq plot DATAPATH: /usr/local/udata/mcnpxs
sun	CONFIG=seq plot cheap DATAPATH: /usr/local/udata/mcnpxs
linux	CONFIG=seq plot cheap DATAPATH: /usr/local/udata/mcnpxs

This standard system-dependent configuration file is included in the following Makefiles, along with the custom configuration files (if present): *Source/*Makefile, *Source/src/*Makefile, *Source/ datasrc/*Makefile, and *Testing/Regression/*Makefile.

C. Source/src Directory

The Makefile in this directory generates the executable for the MCNP code. The name of the executable is defined by the CODE and VERSION variables previously referenced, for example MCNP5. This Makefile also invokes the generation of a post-processed version of the code that is conditionally set by the *install* utility, as mentioned in Section III. These intermediate files are traditionally given the .i suffix. These .i files are edited with a couple of simple lines of *Perl* to remove trailing white space and to delete the lines containing #ident. The purpose for this editing is to facilitate comparison with other similar source files located in another directory. The results of such a comparison are summarized in a listing of dif_.f files which is automatically generated when the post-processed file comparison option is enabled.

D. Source/datasrc Directory

The Makefile in this directory generates the MAKXSF executable. This executable translates type 1 cross-section data (ASCII text format) into type 2 cross-section data (binary format dependent upon the platform OS).

E. Source/dotcomm/src Directory

The Makefile in this directory governs the creation of the dotcomm library, which is only needed when the MPI or PVM versions of MCNP are built. The Makefile in this directory compiles all the .F files, also in this directory, and then passes control to the internals/mpi or internals/pvm directory Makefile, depending on the value of DOTCOMM_INTERNAL, which is set in the ${OS}.gcf config file. The appropriate .h files for MPI or PVM are also included, based on paths given in the ${OS}.gcf config files. After the .c source files in the appropriate subdirectory are built, control is passed back to the place where the *gmake* was invoked.

F. Testing/Regression Directory

The Makefile in this directory will execute the regression test suite and present a summary of the comparisons with standard *testoutp* and *testmctl* templates. The summary also includes the number of threads and/or processors used for multiprocessing runs. An additional feature, which is set by the *install* utility, is the option to generate and test type 2 data. In this mode, the regression testing of type 1 data is first performed, and then MAKXSF is executed with an appropriate SPECS file, and the regression testing is repeated with the type 2 data. The summary table indicates when type 2 data results are being shown. This could be further generalized in the future to generate and test type 2 cross sections for data other than those included in the libraries developed for the testing regression suite. This is necessary due to an incompatibility of direct access binary cross-section files.

It is important to note that any option that is either a system default or included in the custom configuration file can be overridden on the *gmake* command line. For example, on the SGI IRIX system, the configuration default is 'plot seq.' If a user desired to modify this to run an MPI problem (build and test the code) with the number of processors equal to 3, the following options are available to implement this change in configuration. It is necessary to choose only one of these options to effect this desired change.

1. Rerunning the installation utility and modifying the response to the multiprocessing option selections
2. Modifying the custom configuration file, if present
3. Typing on the command line in the *Source* directory: gmake CONFIG='plot omp' NTRD=3

An additional method of operation involves the utilization of the file ${OS}-modes.gcf as shown in Table C.1 to define different targets and modes, as shown in Table C.3. This is useful for building and testing a spectrum of distributed and shared memory multiprocessing options, as well as incorporating combinations of other configuration variables such as cheap.

Table C.6
UNIX Commands

Directory/ Sub-Directory	**User Commands**	**Mode of Operation**	**Comments**
Source	`install`	General configure setup; creates answer files, cleans all directories, generates post-processed files (conditional), builds the code executable, builds the makxsf executable, and executes the regression test suite.	Create configuration setup through MCNP setup menu: Enable all *or* individual default options *or* enter user specific selections for setup.
Source	`install <sys>`	General configuration setup	Create configure files for UNIX systems *other* than current OS
Source	`gmake` *see note below on gmake	Cleans directories, generates post-processed files (conditional), builds the code, creates the makxsf executable, and runs the tests	
		• Using a custom configure file previously generated by *install* *or* • Without running *install* to generate custom configure files	• Uses information from custom configuration file *if present* *or* • Uses system-dependent default options if custom configuration file is *not* present
Source	`gmake reinstall`	Builds the executables, and runs the tests, omitting the initial file cleaning of all directories	Used for development

Table C.6
UNIX Commands

Directory/ Sub-Directory	User Commands	Mode of Operation	Comments
Source	`gmake clean`	Cleans	Cleans all directories
Source	`gmake preproc`	Generates post-processed files	Creates files
Source	`gmake build`	Builds the MCNP executable	Generates executable
Source	`gmake build CONFIG=mpi`	Builds the MPI version MCNP executable	**see note below on CONFIG
Source	`gmake build CONFIG=pvm`	Builds the PVM version MCNP executable	**see note below on CONFIG
Source	`gmake build CONFIG='mpi omp'`	Builds the MPI and threads version of MCNP	**see note below on CONFIG
Source	`gmake build CONFIG='pvm omp'`	Builds the PVM and threads version of MCNP	**see note below on CONFIG
Source	`gmake makxsf`	Builds the MAKXSF executable	Generates executable
Source	`gmake test`	Runs the tests	Executes the regression tests
Source	`gmake mode` *or* `gmake cheap-modes`	Builds the code and executes the regression test suite for many configurations consecutively	Uses the modes definitions, such as multiprocessing options and cheap, indicated in the ${OS}-modes.gcf files
Source/src	`gmake` *or* `gmake EXEC=<exec-name>`	Builds the MCNP executable *or* Changes the name of the executable from the default to <exec-name>	Generates object files, links, and creates the executable. If the object files and/or executable exist and are up to date, they are not regenerated
Source/src	`gmake clean`	Cleans the current directory	Removes files
Source/src	`gmake preproc`	Generates post-processed files	Creates files
Source/src	`gmake filename(s).o`	Generates all the specific object files requested	Useful if only one or a few files have changed

Table C.6
UNIX Commands

Directory/ Sub-Directory	User Commands	Mode of Operation	Comments
Source/datasrc	`gmake` *or* `gmake makxsf`	Generates the MAKXSF executable	Cross-section file handling
Source/ dotcomm/src	`gmake clean`	Cleans all dotcomm directories	Support for multi-processing options
Source/ dotcomm/src	`gmake libdotcomm.a DOTCOMM_INTERNAL=mpi`	Builds MPI version of libdotcomm.a	Support for multi-processing options
Source/ dotcomm/src	`gmake libdotcomm.a DOTCOMM_INTERNAL=pvm`	Builds PVM version of libdotcomm.a	Support for multi-processing option
Testing/ Regression	`gmake` *or* `gmake tests` *or* `gmake test1` *or* `gmake test2`	Cleans the current directory and executes the regression test suite	The number of threads or processors can be changed on the command line. This overrides the setting in the custom configuration file.
Testing/ Regression	`gmake ELOC=<path> EXEC=<exec-name>`	Changes the path and file name of the executable to the values given, cleans the current directory and runs the tests	
Testing/ Regression	`gmake clean`	Cleans the current directory only	
*Type `gmake` | `&tee file.log` to create a log file.			

Table C.6
UNIX Commands

Directory/ Sub-Directory	User Commands	Mode of Operation	Comments

**The CONFIG gmake command line parameter may contain several different values to control which version of MCNP is built. If more than one value is given, all should be contained in single quotation marks. Possible values include one from each of the following categories:
PARALLEL: seq, mpi, pvm, 'mpi omp', or 'pvm, omp'
GRAPHICS: plot
DEBUG: debug
CHEAP: cheap
 	
Example 1:
To build a graphics threaded MPI version on an IRIX64 operating system, type the following command:
`gmake build CONFIG='mpi omp plot'`
Example 2:
To build a graphics MPI version using the Portland Group PGI compiler on a LINUX operating system, type the following command:
`gmake build CONFIG='mpi plot cheap portland'`
If CONFIG is not set on the command line, the default is used, which is sequential mode (seq).			

Figure C-7. Example of a UNIX Custom Configuration File with Default Options Set for a Sun System (Letter M from Top Level Menu)

```
# --- User identification ---
MCUSER=eolus
# --- Operating system identification ---
OS=SunOS
# --- System identification ---
sys=sun:
# --- Code name ---
CODE=mcnp
# --- Code version ---
VERSION=5
# --- Graphics option ---
menugra=XLIB
# --- Graphics library path ---
menugpath=/usr/openwin/lib
# --- Graphics library name ---
menuglib=libX11.a
# --- Include path ---
menugincl=/usr/openwin/include
# --- Datapath ---
DATAPATH=/usr/local/udata/mcnpxs
# --- Number of PVM processors ---
NPVM=1
# --- Number of threads ---
```

```
NTRD=1
# --- Number of MPI processors ---
NMPI=1
# --- Option to generate post-processed code ---
menupre=no
# --- Option for file comparison of post-processed code ---
menucomp=no
# --- Target directory used in file comparison option ---
compdir=/home/eolus
# --- Option to generate and test type 2 data ---
TESTDATA=
# --- General configuration options: Plotting Multiprocessing CHEAP Debugging
---
CONFIG=plot seq cheap
# --- Switch to say that this file has been read ---
premake=premake2
# --- Fortran 90 Compiler ---
FC=f90
# --- C Compiler ---
CC=cc
# --- MPI Implementation ---
MPI=other
# --- MPI Fortran 90 Compiler ---
MPIFC=f90
# --- MPI C Compiler ---
MPICC=cc
```

VII. INSTALLING AND BUILDING MCNP5 ON WINDOWS PCs

There are two different ways to install MCNP5 on a PC running a Windows operating system (95/98/NT/2000/XP/ME). The simplest method is to use the InstallShield® setup programs, similar to that of other Windows programs. The first setup program copies the MCNP executables, source code, and test problems to a user-selected directory and then sets two environmental variables. The MCNP Visual Editor and MCNP documentation are also installed. The second setup program installs the data libraries, MAKXSF (a cross-section library compression program), the files XSDIR and SPECS, and sets an environmental variable. The user is then asked to log out, log back in, and then run the test problems to verify that MCNP has been installed correctly. The main advantage of using this method is that no compilers are needed and no source code needs to be built. This option meets the needs of most users.

Alternatively the user can copy the MCNP directory tree to the desired location and use the supplied install script to build MCNP5, MAKXSF, and/or run the test problem suite. The install script can be used to build MCNP only if the appropriate compilers are already installed. The advantage to this method is that the executables can be rebuilt to apply patches or modifications to the source.

After installing or building the MCNP executable, additional software may need to be installed and appropriate environmental variables may need to be set or changed to take advantage of X11 graphics or parallel communications capabilities in MCNP. X-windows client software, not provided with MCNP5, needs to be running to display geometry, tally, or cross-section plots. X-Windows software is discussed on page C-24 of this appendix. To use the parallel versions of the MCNP executables, parallel communications software will need to be installed prior to running with this capability, and prior to building a parallel version of MCNP, if the install-shield executables are not used. Specific instructions on how to install MPI and PVM are given on page C-24. The environmental variables PATH and DATAPATH can be modified to make file management easier. DISPLAY may need to be set to use the plotting capabilities of MCNP. MCNP environmental variables are discussed on page C-25.

A. Installing MCNP5 on Windows PCs

THE INSTALLSHIELD® SETUP PROGRAM

The InstallShield® programs for MCNP5 are similar to that of other windows applications. Double clicking on the setup file will start the InstallShield® program. After starting the MCNP5 Executables installer, the initial setup window is displayed, then the next two windows present the Copyright notice and Software License Agreement, and request for user information. The following window asks where MCNP should be installed. It does not need to be in the default directory of /Program Files/LANL/MCNP5/. The installation package will then copy the plotting-sequential and parallel executables, MCNP source code, documentation (including the MCNP Manual), Visual Editor, and problem test suite into the chosen directory. The final screen queries the user if it can change the appropriate environmental variables: PATH and DISPLAY. Since PATH is already present, the MCNP5 directory path is appended to this variable. If these environmental variables are changed, the user must log out and then log back in (or reboot for some operating systems) before they will take effect. The user should be aware that if another executable with the name MCNP5 is already present in the path, the first executable in the path, i.e. the previously existing MCNP5, may be unintentionally used. The InstallShield® setup program cannot be used to build MCNP executables.

The second InstallShield® setup program installs the data libraries, the MAKXSF program and the XSDIR and SPECS files. The installer queries the user for a directory to place these files, which may or may not be a subdirectory where MCNP5 was installed. After this directory is specified, the environmental variable DATAPATH is set. Administrative privileges are also required, as well as write permission to at least 2.5 Gigabytes of hard disk space. This large amount of space is mostly used by the ASCII format (type 1) updated ENDF/B-VI cross-section libraries, which can be compressed with MAKXSF and the SPECS file to ~800 Megabytes. For more information about using MAKXSF, see section XII. beginning on page C-39 of this appendix.

After the InstallShield® programs are completed, the user should run the test suite to verify that the executable has been installed and operates correctly on the user's specific operating system and hardware. This testing procedure can be started by double clicking on the runprob.bat icon located in the directory Installation, where MCNP5 was installed. After the test problems are run, files that list the differences between the tally or output files generated and the expected results are displayed. These difference files should be reviewed by the user to determine if the differences are simple round-off errors or something more substantial, indicating incompatible hardware or software and that MCNP may give incorrect results.

To uninstall either the MCNP5 Executables or MCNP Data install packages, the user should remove them via the Windows Control Panel, with the Add/Remove Programs function. This will delete any files that were installed (files created while running the test problems will not be deleted) and will modify the registry appropriately. The environmental variables will not be removed, however, but these can be removed manually.

UNINSTALLING MCNP5

When MCNP5 is installed with the InstallShield® setup program, it can and should be uninstalled like any other Windows application. The user simply goes to the Add/Remove Programs feature inside the Control Panel within My Computer, selects MCNP5, and clicks on Change/Remove. Windows will then uninstall MCNP5, along with any files that it placed on the computer as part of the installation. Any files that were written subsequently, such as files that were written during the testing process, will not be removed, and folders that are not empty will not be removed. However, in such cases, the user has two simple alternatives. If the only additional files are ones that were written during the testing process, the user can run the cleanup.bat script to remove them by double-clicking on its icon in the Installation subfolder within the MCNP5 folder prior to uninstalling MCNP5. Otherwise, the user can manually remove the remaining folders and the files they contain.

It should be noted that uninstalling MCNP5 does not reset the environmental variables that were set as part of the installation. The user can manually remove or reset those variables, although in most cases they are ignored by other Windows applications.

If the user anticipates reinstalling MCNP5 at a later date, it is imperative that the uninstaller be used, because it cleans up the Windows Registry as part of the process. Simply deleting the folders created by the InstallShield® setup program leaves the Registry unaltered, which may cause problems during a subsequent reinstallation.

THE INSTALL SCRIPT

A second method to install MCNP5 uses an install script, which interactively queries the user for various build options and then executes the make utility to build MCNP, MAKSXF and/or run the test problem suite. The various build options include which Fortran and C compilers should be used, the location of appropriate X11 files, and the path to the XSDIR file. The script also gives the user the opportunity to only generate custom makefiles. These custom makefiles contain the build options selected, and will be automatically used whenever the make utility is used. An answer file, which lists the options chosen in the install script, is also created. The answer file can only be used to set options in the install script, and only if it is specified on the install command line. If MCNP is to be built, or the test suite is to be run, then install script will then execute *gmake* (the GNU version of *make*) and write most of the output to the file install.log.[5,6,7] This script, which uses the *gmake* utility to build the code, can only be used if a UNIX-based shell is installed.

INSTALLING A UNIX SHELL - CYGWIN

If the install script or *gmake* utility is used, a UNIX command shell must be installed. UNIX command shells are not standard on Windows operating systems and must be installed. Cygwin, a freeware port of a UNIX command shell for Windows PCs, can be obtained at www.cygwin.com or http://www.redhat.com/apps/download/. The setup program can be downloaded or run from the

Redhat website. This setup program will step users through the Cygwin installation process, allowing them to select web installation or download installation files to a local drive. The location where the Cygwin software should be installed (a path without spaces is recommended) and a temporary directory where files can be downloaded are specified in the next two windows.

After selecting a website to download or install from, the user selects Cygwin packages to install. In addition to the Cygwin packages that are selected by default, the *gcc* and *make* packages (located under the Devel directory) will also need to be selected to build MCNP with the *make* utility. The *perl* package (located under the Interpreters directory) is also recommended, and is required if the Absoft or Lahey Fortran compilers are going to be used to compile MCNP.

INSTALLING PLOTTING SOFTWARE - X WINDOWS CLIENT

In order to display MCNP plots, an X11 windows client software package is needed. Several commercial X-windows clients are available: Reflection X (http://www.wrq.com/products/), Hummingbird's Exceed_NT (http://www.hummingbird.com/products/nc/exceed/index.html), and Starnet's X-win32 (http://www.starnet.com/). No single commercial product is recommended. A freeware X client is also available with Cygwin, X-Free86. It has also been tested with MCNP5. These client software packages do not need to be the developer or professional versions, since the X11 header and library files are included with the MCNP5 distribution.

INSTALLING PARALLEL SOFTWARE - MPICH.NT OR PVM

In order to use or build MCNP5 with parallel capabilities, appropriate parallel communications software must be installed. Either MPI or PVM communications protocols are supported. To build a parallel version of MCNP5 for Windows, see page C-29.

The MPI port for Windows is MPICH.NT, developed at Argonne National Laboratory, and can be downloaded from http://www-unix.mcs.anl.gov/~ashton/mpich.nt/. This website also offers the helpful references MPICH.NT FAQ and MPICH Users Manual. If there is no need to rebuild parallel MCNP5, only the runtime dlls and MPIRun package will be needed (mpich.nt.1.2.4.exe). This package uses an InstallShield® setup program which requires installation from an administrative account on all PCs in the cluster. If MCNP5 needs to be built, then the source code (package mpich.nt.1.2.4.src.exe) should be downloaded and unzipped as well. The program MPIConfig must be run on each computer after MPICH.NT installation. The local host name must be added and the settings applied.

Once MPICH.NT is installed, a few additional steps are required. MPI enabled MCNP must be copied to the same directory on all hosts. MCNP can be executed through either the Windows MPIRun GUI or command line MPIRun. For the MPIRun GUI, hosts must be added by selecting or typing their names in the hosts section. The DATAPATH may need to be set under the advanced options. For the command line MPIRun, the hosts must be specified by the `-hosts` option, which can be used to specify the number of processes started on each host. Typically the number of processes is equal to the number of CPUs utilized plus one. The first process listed is the master process and does not run any histories. For example, the command to start three MCNP MPI processes on ComputerA (a dual CPU machine) and one process on ComputerB is:

```
    mpirun -hosts 2 ComputerA 3 ComputerB 1 mcnp5mpi inp=test
```

Alternatively, MCNP5 can be built and run with PVM, developed at Oak Ridge National Laboratory. The PVM port for Windows can be downloaded from http://www.csm.ornl.gov/~sscott/PVM/Software/. The file ParallelVistualMachine3.4.3.zip contains the source and binaries needed to install and run PVM on a single computer. This InstallShield® program must be run on all Windows PCs in the cluster. PVM requires additional communications software, a remote shell (RSH) client/server package, before it will run across a cluster of Windows PCs. Two commercial RSH packages can be obtained from http://www.winrshd.com/ and http://www.ataman.com/. The RSH package must be installed on each computer in the cluster, and the permissions must be set to allow RSH or REXEC connections for the desired user accounts. The Ataman RSH package was successfully tested.

Similar post-installation steps are required to run the PVM version of MCNP5 on Windows PCs. The MCNP executable must be copied into the %PVM_ROOT%/pvm3/bin/WIN32 directory on all hosts. PVM must be started on a single computer before a PVM enabled MCNP can be executed. After PVM is started, additional hosts can be added to the PVM cluster with the "add host" command from the PVM console prompt. PVM operability can be tested and verified with the PVM example programs, such as hello. MCNP can then be started from a separate command shell with the following command:

```
mcnp5pvm inp=test tasks n
```

where `n` is the number of slave processes. The number of tasks is usually the number of CPUs in the cluster. A negative number entered for `n` causes MCNP to skip the initial load-balancing feature, and is recommended for a homogeneous cluster. Additional information on how to install MPI or PVM can be obtained in their respective user manuals. Additional information on running MCNP in parallel can be found in Appendix C section VIII. beginning on page C-30. For more information about running MCNP in parallel on Windows clusters, see Reference 16 on page C-42.

SETTING ENVIRONMENTAL VARIABLES

After installation, it may be necessary to change or add three Windows environmental variables. Variables are set differently for Windows 95/NT/2000/XP/ME, but for each of these operating systems the variables can be viewed the same way. The value of an environmental variable PATH, for example, can be printed in a command shell window (i.e. a "DOS prompt") with the command

```
echo %PATH%.
```

The first environmental variable that may need to be changed is PATH, a semicolon-separated list of directories used to find executables and dynamic link libraries. The directory where MCNP is installed should be included, so that MCNP can be executed from any directory, making file management more convenient. If several programs with the same name exist within PATH, then the executable in the first occurring directory will be executed. Appending the directory of the newly installed version of MCNP to the PATH may not change which MCNP is executed if an older version with the same name is given earlier in the directory listing. Failure to change PATH will mean that the MCNP executable must be located in the directory where the input file is located. To use the plotting features of MCNP, the location of the dynamic link library X11.dll or Xlib.dll may need to be added to PATH as well. Failure to make sure that X11.dll or Xlib.dll is in your path may mean that a plotting-enabled executable cannot run, even if the plotting features of MCNP are not used.

The second environmental variable that may need to be added or changed is DATAPATH. It is the directory path to the file XSDIR, which is used by MCNP to locate the cross-section data libraries. MCNP also searches in the local directory and a directory specified at compile time. If XSDIR does not exist in any of these three locations, MCNP will issue a FATAL error. Additionally, the command line option XSDIR=name may be used to specify an XSDIR formatted file with a different name located in the directory given in DATAPATH. Failure to set DATAPATH means that the file XSDIR must be located in the directory where the input deck is located.

The third environmental variable is DISPLAY, which is only needed for a plotting version of MCNP. This environmental variable is used by the X windows client to route X windows. It is usually set to display windows on the same computer where MCNP is executed. In this case, the value of DISPLAY should be localhost:0.0. Failure to set DISPLAY may mean that the plotting executable will not be able to open a window and plot. A graphics version of MCNP will still be able to create a postscript file of the plots, even without the X-windows client software or the environmental variable DISPLAY.

B. Building MCNP on Windows PCs

MCNP5 must be built if the install script is used or the source needs to be modified or patched. The install script uses the *make* utility to direct one of three supported Fortran 90/95 compilers, and one of three supported C compilers, to compile and link MCNP. Alternatively, the interactive graphical interface, Compaq Developer Studio, uses Compaq Fortran 90 and Microsoft C to build MCNP. Neither the *make* utility nor a UNIX shell is needed to build MCNP with Compaq Developer Studio. These two methods will be discussed in the following paragraphs.

THE *MAKE* UTILITY

Make is a utility which understands user defined relationships between different files used to build programs. In an attempt to supervise the building of a target program, *make* controls preprocessors, compilers, linkers, archive utilities, or other programs to manipulate these files. MCNP5 can be built on all supported platforms, including Windows PCs, with the *make* utility.

Unlike most UNIX or LINUX based operating systems, the *make* utility is not standard on Windows. The recommended version of *make* for a Windows PC is the GNU *make* utility, which is an optional addition with Cygwin, described above. The *make* utility does not include any compilers, which also must be installed to build MCNP. The *make* utility can be used to build MCNP by typing "make build" in a Cygwin command prompt in the MCNP5/Source directory.

One of the first things *make* does is read an operating-system dependent file in the MCNP5/Source/ config directory. For all Windows installations, this file is Windows_NT.gcf. It contains all the default paths to the supported compilers, header files, and libraries, and specifies the compiler and linker options. This file should be modified if the compilers and other necessary files are not in the default locations. The *make* utility will also read the custom_Windows_NT.gcf file created by the install script if present. Additional information about these files and *make* commands is given in Appendix C section II. beginning on page C-2.

THREE FORTRAN COMPILERS

The *make* utility uses one of three Fortran 90/95 compilers and one of three C compilers to build MCNP5 on Windows PCs. The supported Fortran 90 compilers are: Compaq Visual Fortran (CVF version 6.6B) [formerly known as Digital Visual Fortran], Lahey Fortran 95 Pro (LF95 version 5.70c), and Absoft Pro Fortran (F95 version 8.0).[8,9] The supported C compilers are the Microsoft C/C++ compiler (MSC version 12.00.8168) and GNU gcc compiler (version 2.95.2-5 [Cygwin special]), either of which may be used with any of the three Fortran compilers. The Fujitsu C compiler (FCC version 3.0) is also supported, but only in conjunction with Lahey Fortran 95. Table C.7 shows the versions of MCNP that can be compiled with the *make* utility. For example, the command "make build CONFIG='plot cheap compaq cl' " at the Cygwin command prompt in the MCNP5/Source directory will build a plotting version of MCNP5 with the Compaq Fortran and Microsoft C compilers.

Table C.7
Supported Versions and Compilers with the Cygwin *Make* Utility

MCNP Version	Supported Compiler Sets*	Make Command Line	Notes
Sequential	Compaq (v6.6B)	CONFIG='seq compaq'	GNU gcc is used***
	Lahey (v5.70c)	CONFIG='seq lahey'	Absoft and Lahey need preprocessor.
	Absoft (v8.0)	CONFIG='seq absoft'	Absoft does not support control-c interrupts.
Plotting	Compaq + MSC**	CONFIG='plot compaq cl'	X11 library and headers required to compile.
	Compaq + gcc	CONFIG='plot compaq gcc'	X client running required to display.
	Lahey + MSC	CONFIG='plot lahey cl'	Absoft and Lahey need preprocessor.
	Lahey + gcc	CONFIG='plot lahey gcc'	Absoft does not support control-c interrupts.
	Lahey + fcc	CONFIG='plot lahey fcc'	
	Absoft + MSC	CONFIG='plot absoft cl'	
	Absoft + bcc	CONFIG='plot absoft gcc'	

*Only the professional version of the three Fortran compilers is supported.
**The default. The command "make build" will build a plotting executable with CVF and MSC.
***The *make* utility expects there to be an object file from the c source, so gcc is used to build a nearly empty object file which the Fortran compilers link.

All of the compilers use the environmental variables LIB and INCLUDE to locate the compiler's libraries and include files, respectively. These should be set when the compilers were installed. Additionally, the directory path to the compiler executables (f90, lf95, f95) should also be located in the PATH environmental variable. If more than one compiler with the same name is installed, the explicit path and compiler can be specified in the Windows_NT.gcf file.

A PERL PREPROCESSOR

Unlike the Compaq Fortran compilers, the Absoft and Lahey Fortran compilers do not have the capability to preprocess #ifdef statements that are used in the MCNP Fortran and C source to specify version (sequential, plotting, etc) and platform (UNIX, LINUX, DEC, etc.) specific code. The MCNP Fortran source must be preprocessed before the Absoft and Lahey compilers can be used. This can be done with the *perl* script fpp.pl, which is provided with the MCNP distribution. This script processes out the #ifdef statements and substitutes the appropriate values for the MCNP version, compile date, and version number. If the environmental variable DATAPATH is set, it will also be substituted directly into the source code before MCNP is compiled. The processed files are saved with .F95 extensions for Absoft or .i extensions for Lahey, which are then compiled. The intermediate files are deleted after linking.

COMPAQ DEVELOPER STUDIO

As an alternative to the *make* utility, the Compaq Developer Studio build utility is also supported. It is an independent GUI which requires neither Cygwin nor the *make* utility. Using the Developer Studio, it is possible to build a sequential, plotting, MPI or PVM version of MCNP5. Included in the MCNP5 distribution are Developer Studio projects and project workspaces (.dsp and .dsw files, respectively) for each of these versions. To build one of these four versions, open the corresponding .dsw file located in the /Source/CVF directory. If CVF is installed on the computer, this can be done simply by double clicking on the appropriate file in the Windows Explorer.

The only settings that may need to be changed are the paths to the X11, MPICH.NT or PVM files, if they were not installed in the default directory. The paths to these files are specified in the C preprocessor and link input fields of the project settings. Unlike *make*, the CVF Developer Studio has no problems when paths to these files contain spaces. An additional path to the file XSDIR can also be specified at compile time. The Fortran preprocessor keyword DPATH and its value can be set on the project settings' Fortran tab. While the directory path must not contain spaces, the DOS style format will work. For example, instead of C:\Program Files\LANL\MCNP5\data, C:\Progra~1\LANL\MCNP5\data should be specified.

The option to build is the second item under the build menu on the topmost menu bar. Alternatively, the F7 button can be pressed to start the build. In some cases Developer Studio cannot determine which files need to be built prior to others, and the code may need to be rebuilt after the first failed build completes. Additionally, the user is encouraged to build the release version of MCNP by setting the active configuration (under the build menu item). The versions of MCNP5 that have been built and tested with the Compaq Developer Studio are shown in Table C.8.

Table C.8
Supported Versions with Compaq Developer Studio

MCNP Version	Supported Compiler Sets	Notes
Sequential	Compaq	No C compiler is needed.
Plotting	Compaq + MSC	X11 library and headers required to compile. X client required to display.
Parallel (MPI or PVM)	Compaq + MSC	PVM or MPICH.NT also required.

BUILDING PLOTTING VERSIONS

To display MCNP geometry, tally, or cross-section plots, it is necessary to have a plotting enabled version of MCNP and X window client software. A plotting version of MCNP can be built with either the *make* utility or Developer Studio. With *make*, a plotting executable is the default. To force *make* to build a plotting enabled version of MCNP with *make*, it is necessary to specify the plot on the CONFIG keyword.

One of the enhancements for Windows PCs is that all of the appropriate source files (including the X11.lib, X11.dll, and header files) are included in the MCNP distribution and are located in the Source/X11r6 directory. The original source can be downloaded from www.X.org. The X11 library may be downloaded from ftp://ftp.cc.utexas.edu/microlib/nt/x11r6/ or built from the X.org source. Proprietary X11 header files and library files may also be included with a commercial X windows client (if the developer/professional version was purchased), but using the X11 files with the MCNP distribution is recommended.

BUILDING PARALLEL VERSIONS

To use the parallel features of MCNP, it is necessary to have a parallel enabled version of MCNP and the appropriate parallel communications software installed and running. Parallel enabled versions of MCNP can only be built with Compaq Developer Studio after the header files and appropriate libraries have been installed.

The Developer Studio project files mcnp5mpi.dsw and mcnp5pvm.dsw already have all the appropriate settings configured. These changes include the addition of the preprocessor definition MULTP (and MPI for the MPI version), additional Fortran and C source, MCNP include files, and the additional paths to these directories and the MPI or PVM directories. The only changes needed are the paths to the MPI or PVM include files and libraries if these are not installed in the default directories. The include file path will need to be listed in the C++ tab (preprocessor category). The MPI or PVM library should be listed in the Object/library modules line on the link tab (the general category). The necessary libraries for MPI are the ws2_32.lib and mpich.lib libraries. The PVM libraries are ws2_32.lib, libpvm3.lib, and libgpvm3.lib. The path to this library should be listed in the input category.

VIII. PARALLEL CONFIGURATION INFORMATION

BUILDING THE DOTCOMM LIBRARY

The dotcomm library, libdotcomm.a, is only needed when the MPI or PVM version of MCNP is built. The dotcomm.a library will be automatically built if the *gmake* command-line parameter definition of CONFIG contains either MPI or PVM, or the MPI or PVM option is selected in the install script. To build the dotcomm library only, the command `gmake libdotcomm.a DOTCOMM_INTERNAL=x` should be typed in the Source/dotcomm/src directory, where x is either MPI or PVM. If either parallel option was selected, the dotcomm library and either the MPI or PVM library is linked when the MCNP executable is linked. If the MPI or PVM libraries are not found while linking, then the appropriate library include path should be set in the ${OS}.gcf file by adding the path to the definition of DMMP. While the dotcomm library cannot be built with both MPI and PVM, either option can be combined with the OpenMP threads (CONFIG=omp) option, which does not use the dotcomm library.

EXECUTION IN PARALLEL MODES

MCNP may be built for either MPI or PVM and then executed in the same fashion as other parallel programs on your system. In general, for MPI the execution line will look like:

```
mpirun -np <m> mcnp5.mpi i=inp01
```

where <m> is the total number of processes including master, and m-1 slave processes will be available to transport particles. To provide load balancing with MPI, add the keyword "balance" to the command line. For load balancing with MPI, the execution line will look like:

```
mpirun -np <m> mcnp5.mpi i=inp01 balance
```

If the MPICH implementation of MPI is used, then the keyword "eol" must be added after all other MCNP keywords to distinguish MCNP keywords from directives added by MPICH. In this case your execution line will be:

```
mpirun -np <m> mcnp5.mpi i=inp01 eol
```

If you combined the build with OpenMP for a multiprocessor per node SMP machine, then each slave may be optionally threaded by setting the tasks option:

```
    mpirun -np <m>   mcnp5.mpi   i=inp01 tasks <n>
```

The syntax required to allocate enough resources for the threading varies by system. For instance, on an alpha Tru64 System 5.1, use:

```
    prun -n <m>  -c <n> mcnp5.mpi    i=inp01 tasks <n>
```

to reserve n processors for each <m> MPI process.

The combined-option executable can be executed in any of the following ways: sequential, all-MPI, all threads, or hybrid. Note that the minimum number of MPI slaves accepted by MCNP is 2, so you need at least three MPI processes.

For a PVM execution, you first need to assure there is a copy of (or link to) the MCNP5.pvm executable in subdirectory pvm3/bin/<pvm arch> of the home directory. Launch PVM. At the `pvm>` prompt, add machines to the "Parallel Virtual Machine" using the `'add'` command. The syntax of the `'add'` command is `'add <hostname>'`, where `<hostname>` is the name of the machine to be added to the "Parallel Virtual Machine." Next, give the command `'quit'` to exit the console, but leave the pvm daemon running.

Execute the MCNP job(s), and then at the end of the session again enter PVM to return to the console. Give command `'halt'` to kill the pvm daemon.

The MCNP command line will be of the form:

```
    mcnp5 i=inp01 tasks <j>
```

where the absolute value of <j> is the number of slaves that will be spawned to track particles. If the user built with PVM OpenMP combined, the command line will be

```
    mcnp5  i=inp01 tasks <j>x<n>
```

where each of the j spawned slaves runs particles on n threads.

For either PVM option, if j is positive, a load balancing operation is enabled early in the run to account for a heterogeneous environment. To omit that step, as for a homogeneous cluster, enter a negative value for j.

The simplest parallel run would be to build for one shared memory node and to just run openMP threading (No PVM or MPI). Build with CONFIG='omp' and execute with just tasks option on the following command line:

```
    mcnp5  i=inp01  tasks <n>
```

Running the automated test suite in any of these parallel configurations may require editing the Testing/Regression/Inputs/ runprob.mpi to satisfy your environment. Script runprob.mpi is used for MPI testing; runprobmt for all other configurations (seq, OpenMP only, PVM).

IX. TESTING PERFORMED TO DATE

The testing performed to date includes the following platforms: sgi, alpha, aix, sun, and PC. There were several limitations and problems encountered, which are mentioned in this section.

On some platforms, such as the Alpha OSF1 Q machine, it is necessary to load modules upon login in order to access compilers or debugging tools. The compiler versions corresponding to the tested platforms include:

Table C.9
UNIX Platforms Tested to Date

Platform Name	Compiler Version	Mode	Comments
SGI IRIX64	MIPSpro 7.3.0 MIPSpro 7.3.1.2m	Sequential Seq and Parallel	host kaji hosts theta and bluemountain
Alpha OSF1	Fortran 5.3-915 Fortran 5.4.1.a	Sequential Seq and Parallel	host ratbert hosts QSC and Q
AIX 4.3.3 AIX 4.3 AIX 5.1	xlf90 7.1 xlf90 7.1 xlf90 7.1	Sequential Seq and Parallel Seq and Parallel	host toji host blue at LLNL host frost at LLNL
SunOS	f90 version 6.2 f90 version 6.1	Sequential Seq/PVM	host glitter host glitter
Linux	PGF90 4.0-1	Seq/MPI/PVM/ OMP	host lambda
Linux	LAHEYPro_6.1e	Seq/MPI/PVM/ OMP	host lambda
Linux	Absoft Pro-8.0 QF3	Seq/MPI/PVM	host lambda

In Table C.10, Table C.11, and Table C.12, the software and settings used to test the PC version of MCNP5 are documented.

Table C.10
PC Platforms Tested to Date

Software	Version	Used in MCNP Mode	Notes
Compaq Visual Fortran	6.6B	seq, plot, mpi, pvm	
Compaq Developer Studio		seq, plot, mpi, pvm	
Microsoft Visual C/C++	6.0	plot, mpi, pvm	
Lahey Pro Fortran 95	5.70c	seq, plot	
Fujitsu C/C++	3.0	bplot	Only links with Lahey
Absoft Pro Fortran 95	7.5, 8.0	seq, plot	
GNU gcc	2.95.3-5 (cygwin special)	seq, plot	
Cygwin	Setup version 2.249.2.5	seq, plot	
GNU Make	3.79.1 (i686-pc-cygwin)	seq, plot	
GNU Perl	5.6.1 (cygwin multi)	seq, plot	Only required with build using Absoft or Lahey
fpp (Perl Script)		seq, plot	Only required with build using Absoft or Lahey

Table C.11
Files, Libraries, and Client Software for Plotting on PCs

Software	Version	Used in MCNP Mode	Notes
X11 .h include files	X11R6.6	plot	
X11.lib	X11R6.6	plot	
Reflection X	9.0.3	plot	X windows client

Table C.12
Software for Parallel Jobs on PCs

Software	Version	Used in MCNP Mode	Notes
PVM	3.4.3	PVM	
ATRLS (*TCP Remote Logon Services*)	3.1	PVM	Not needed for single multiprocessor PC. Needed for cluster of PCs. Only tested with Windows 2000 PCs.
MPICH_NT	1.2.4	MPI	

X. MODIFYING MCNP WITH PATCHES

This section describes the process for modifying MCNP. The method is based on the GNU tools *diff* and *patch.*[5]

NOTE: *Before making any changes to the MCNP distribution, backup the original files for recovery and for comparison. It is necessary to have an unmodified copy of the entire distribution to generate and/or apply patches.*

The method for creating and applying patches has changed completely from how it was done with earlier versions of MCNP. Patches are not written; they are generated. Also patches can be applied to the entire MCNP directory tree.

The commands described here to build or rebuild the code assume a UNIX operating system or a similar shell running on Windows (Cygwin).

Why Use Patches?

Why might you need to use a patch?

- A patch may be issued by the MCNP team to fix bugs or add new features.
- You may want to distribute your changes to others.
- You may want to adapt your changes to a new MCNP version.

In each of the above scenarios, the ability to summarize the changes in a short file makes the work easier. For example, if the MCNP team issues a patch and you have also made local changes, you can use two patches to merge the two together.

Modifying the Source Code

To make local changes to a copy of MCNP, just edit the appropriate source files. The directory structure for MCNP5 is:

```
Source/
     Makefile
     install
     config/
     CVF/
     datasrc/
     dotcomm/
          src/
     X11R6/
Testing/
     Regression/
          Makefile
          Inputs/
          Templates/
     config/
          test_options.mk
```

The radiation transport source code is in *Source/src/.* If you need to make changes to existing files, edit them and leave them in their current locations.

If you add new files, they will be automatically included in builds if you follow these steps.

1. Create the new file conforming to the Fortran ANSI Standard[14] in the free-style format. The name of new Fortran files should be of the form `<name>.F90`. See Reference 15 for the definition of the MCNP coding style guide.
2. Edit the file *Source/src/FILE.list* to add your new Fortran files to the *F_SRC* macro. A backslash is necessary to continue to the next line. So if you add a whole new line at the end, you must append a backslash on the line before it. If you insert a line in the middle, put a backslash at the end of the new line.
   Note: *There can be no white space after the backslash.*
3. Add a dependency line for each of your new files. If your files do not USE any MCNP modules (existing or new), this step can be skipped. If you make modifications to an existing file that adds a module dependency, add the new dependency condition to the appropriate line. See the *Source/src/Depends* file for the format.
4. In *Source/,* type `'make clean'` to prepare for a full rebuild.

When your changes are ready, change into the *Source/* directory and type `'make build'`.

GENERATING A PATCH

To generate a patch from a modified version of MCNP, you need to have access to an unmodified version (or other reference version) for comparison. Comparisons and patches are generated with the GNU utility *diff.*

To see the extent of the modifications, you can generate a list of modified, added and deleted files with the command

```
> diff -qr <base_dir> <modified_dir>
```

Be sure to examine the output of this command to look for unintended differences. Object files, editor-backup files or executables may be in the list. Clean up any extraneous files before proceeding with patch generation.

For example, comparison of two MCNP directories *MCNP/Source/* and *MCNP/Modified/* will generate output like the following. In this example, there are no added/deleted files or extraneous files, only modified ones.

```
> cd MCNP
> diff -qr Source Modified
Files Source/src/Makefile and Modified/src/Makefile differ
Files Source/src/dmmp.F90 and Modified/src/dmmp.F90 differ
Files Source/src/dynamic_arrays.F90 and Modified/src/dynamic_arrays.F90 differ
.
.
.
```

The patch itself is generated with the command

```
>diff -Naur <base_dir> <modified_dir> > <patch_file>
```

where <patch_file> is the name you want to give the file containing the patch.

To generate the patch, you should position the two directory trees adjacent to each other. In the above example, the unmodified MCNP distribution, `<base_dir>`, is in *MCNP/Source*, the modified version, `<modified_dir>` is in *MCNP/Modified.* Let the desired output file, `<patch_file>` be *MCNP/my_mods.txt.*

For this example, the commands to generate the patch are

```
> cd MCNP
> diff -Naur Source Modified > my_mods.txt
```

The first few lines of `my_mods.txt` would look something like the following patch if the file *Source/src/Makefile* was modified.

```
diff -Naur Source/src/Makefile Modified/src/Makefile
--- Source/src/Makefile 2002-10-17 12:32:40.000000000-0600
+++ Modified/src/Makefile2002-10-21 21:07:40.000000000-0600
.
.
.
```

After generating a patch, you can add commentary to the patch-file to describe its function. Start all comment lines with `'#'` to mark them as comments. See the GNU documentation on *patch* (type `'man patch'` on UNIX systems) for more information on the flexibility of this excellent tool.

APPLYING A PATCH

Applying a patch is as simple as generating one. It is especially easy if it is a patch to the base MCNP distribution and you want to apply it to your unmodified copy of the same version of MCNP.

Note: *In all cases, make backups of the patches, the original distribution, and your modified directory before proceeding.*

To see the files that a particular patch will affect, you can use the GNU utility *grep* as follows. This can be useful to determine if two patches are independent or may be in conflict (modifying the same files). For the example used above, partial results are given below.

```
>grep '^diff' <patch_file>
diff -Naur Source/src/Makefile Modified/src/Makefile
diff -Naur Source/src/dmmp.F90 Modified/src/dmmp.F90
diff -Naur Source/src/dynamic_arrays.F90 Modified/src/dynamic_arrays.F90
.
.
.
```

If an unmodified copy of the MCNP distribution (*not your only copy!*) is stored in *MCNP/Source*, the commands to apply a patch named *mcnp_patch.20021102* is

```
>cd MCNP
>patch -p1 -d Source < mcnp_patch.20021102
```

The output will look something like the following with one line per patched file:

```
patching file `src/Makefile'
.
.
.
```

If all the output is of this form, the application of the patch is fully successful.

Complications can arise if you need to apply more than one patch to the same version. Patches from the MCNP team will be documented for compatibility with other such patches. If a specific order of application is required, that will be made clear.

If you have made local modifications to the base distribution, follow these steps:

1. Generate a patch with your changes from the base MCNP distribution or a version patched only with MCNP-team patches. See the discussion above for instructions. Back up a copy of your patch.
2. Apply any new patches from the MCNP team in the documented order before applying your patch. You should also back up the MCNP-team patches for future use.
3. Back up the newly-patched MCNP distribution for future reference.
4. Apply your patch to the newly-patched MCNP.

If your patch modifies sections of MCNP that are also modified by other patches, portions of your patch may be rejected by the *patch* utility as being in conflict with the newly-patched version. In these cases, a file will be created with the rejected portions. For example, if the patch to *Source/src/Makefile* fails, a file named *Source/src/Makefile.rej* will be created with the problem section. You will have to resolve any such rejections manually. In this case, a file named *Source/src/Makefile.orig* will also be created and represents a copy of the original.

If you receive an MCNP-team patch, you *can* try to apply it directly to a locally modified version. The format of the generated patches makes the patch utility able to find the most likely place to insert modifications. If the patching fails, fall back to the procedure given above.

Note: It is possible that either method of applying your own patches on top of MCNP-team patches will succeed, but still generate incorrect code. *The MCNP team cannot and does not give any warranty that your modifications will be compatible with its own. It is the responsibility of the author of the non-MCNP-team modifications to make them compatible with team releases.*

XI. MCNP VERIFICATION

MCNP5 comes with a set of test problems that appear in a directory called Testing. This directory should be placed at the same level as the Source directory. In other words, the parent directory of both Testing and Source should be the same. This allows the install and test script to find the scripts that run the set of test problems. Input files and Templates of expected test results for each supported platform are included in the content of the Testing directory. Table C.13 below documents the names and contents of the scripts.

Table C.13
Scripts for Running Test Problems

File	Description
Makefile	Makefile for MCNP verification.
testinp.tar	Compressed input files for MCNP verification.
testmctl.sys	Compressed tally output files for MCNP verification.
testoutp.sys	Compressed MCNP output files for MCNP verification.
testdir1	Cross-section directory for MCNP verification.
testlib1	Cross-section data (type 1) for MCNP verification.

Upon completion of the regression test set, there should be a set of inp??m and inp??o files (?? = 01, 02, etc.). If the appropriate files to be compared exist, they are compared; if they do not exist, an error message is produced for that problem number. Differences between these runs and the standard show up in the dif?? files. Exact tracking is required for MCNP5 verification. Significant differences, that is, other than round-off in the last digit, may prove to be serious (e.g., compiler bugs). In such cases, the cause of the difference should be fully understood.

The test problems are neither good nor typical examples of MCNP problems. Rather, they are bizarre test configurations designed to exercise as many features as possible. The test set is constantly changed as new capabilities are added to MCNP and as bugs are corrected. The INPnn files are the same for all systems, but the answers, mctl??, differ slightly from system to system because of differences in arithmetic processors. The test set works on the basis of "particle tracking" in which the random walks must be identical. The test problem data library testlib1 is also only for testing purposes because it contains bad data used to test the code. The testlib1 data should not be used for real transport problems.

XII. CONVERTING CROSS-SECTION FILES WITH MAKXSF

The auxiliary code MAKXSF can be used to convert cross-section libraries from one format to another and to construct custom-designed cross-section libraries.

MCNP can read cross-section data from two types of files. Type 1 files are formatted and have sequential access. Type 2 files are unformatted, binary files and have direct access. RSICC distributes type 1 files because they are portable across platforms, but in that format the files are slow. The auxiliary program MAKXSF is provided for translating type 1 files into the faster access type 2 files. In the same manner, type 2 cross-sections can also be converted into type 1 cross-sections. You can also use MAKXSF to delete cross-section tables that you do not need and to reorganize the cross-section tables into custom-designed cross-section libraries. Please note that it is necessary to rebuild your type 2 cross-section files for MCNP5, but the procedure has been simplified. The program MAKXSF is compiled appropriately for your platform using the makefile system at the same time as MCNP is built. It is located in directory Source/datasrc.

The input files to MAKXSF are one or more existing cross-section libraries; a directory file, XSDIR, which describes the input cross-section libraries; and a file called SPECS that tells MAKXSF what it is supposed to do. The output files are one or more new cross-section libraries, a new directory file that describes the new cross-section libraries, and a file called TPRINT that contains any error messages generated during the run. The input and output cross-section libraries can be any combination of type 1 and type 2 files. The various types of cross-section libraries and the form and contents of the cross-section directory file are described in detail in Appendix F. The directory file XSDIR in the MCNP data package contains complete descriptions of all the cross-section files in that package. Printing it provides a useful reference. The sample SPECS file in the MCNP data package can be used with MAKXSF to create a complete set of type 2 files from the type 1 files provided. It should be noted that when re-executing MAKXSF, it is necessary to first delete a pre-existing TPRINT output file.

The SPECS file is a formatted sequential file with records not exceeding 80 characters. The data items in each record may start in any column and are delimited by blanks. The structure of the SPECS file is given in Table C.14 and an example is given in Table C.15.

Table C.14
SPECS File Record Structure

Record	**Contents**		
1	Name of old dir file	Name of new dir file	
2	Name of old xs lib*	Name of new xs lib	Type Recl* Epr*
3	Access route* entered into new directory file (or blank line)		
4 +	Nuclide list, if old xs lib is absent		
Blank record			
Where	* = optional		

The default for Epr (entries per record) is 512. Recl (record length) will be set appropriately by MAKXSF, depending on whether your platform specifies records in terms of words (OSF and PC) or bytes (all others), resulting in 8 bytes per entry. All data is stored double precision. You should NOT need to use these options.

Records 2 through 4+ can be repeated any number of times with data for additional new cross-section libraries. The SPECS file ends with a blank record. If "name of old cross–section library" exists on record 2, all nuclides from that library will be converted.

Table C.15
Example SPECS File

Record	**Contents**		
1	xsdir1	xsdir2	
2	el1	el2	2
3	home/scratch/el2		
4	rmccsab2	2	
5	datalib/rmccsab2		
6	7015.55c		
7	1001.50c		
8			

In Table C.15, the SPECS file starts with type 1 cross-section directory file XSDIR1, electron library EL1, and neutron libraries RMCCSA1 and RMCCS1. All nuclides on the electron data file EL1 are to be converted to a type 2 file called EL2. Records 4–7 tell MAKXSF to search all libraries listed in XSDIR1 until it finds nuclides 7015.55c and 1001.50c (which happen to be on RMCCSA1 and RMCCS1, respectively) and construct a new type 2 library RMCCSAB2 consisting of only these nuclides. The new directory file XSDIR2 will tell MCNP to look for the electron cross sections in /home/scratch/el2 and for the neutron cross sections in /datalib/rmccsab2.

If the type of the new cross-section file is specified to be 1 in record 2, only the name of the new cross-section file and the 1 for the type are read in that record. If the type in record 2 is 2, the record length and the number of entries per record can be specified, but it should not be necessary. If there is any difficulty, be sure that you are compiling MAKXSF with the option

```
-DDIRACCESS_RECL_WORD
```

for platforms specifying records in terms of words (OSF and PC) and without that flag for platforms specifying record lengths in bytes. Running the type 2 tests of the test suite from the install script is a good way to be sure the settings are correct.

The optional access route on record 3 of the SPECS file is a concatenation of a UNIX data path with the library name and becomes the fourth entry for each nuclide in the library in the XSDIR file.

It is not necessary to generate all the cross-section files that you will ever need in one MAKXSF run. You can combine and edit directory files at any time with a text editor or with another MAKXSF run. The only requirement is that you must give MCNP a directory file that points to all the cross-section files that are needed by the current problem. If you plan to run a long series of

MCNP problems that all use the same small set of cross-section tables, it might be convenient to generate with MAKXSF a small special-purpose cross-section file and directory file just for your project.

XIII. REFERENCES

1. E. C. Selcow, "New Fortran 90 Build System for MCNP (U)," Los Alamos National Laboratory research notes X-5:RN(U)01-11 (Rev. 1), May 8, 2001.
2. R.M. Stallman and R. McGrath, *GNU MAKE*, Free Software Foundation, Boston, MA (available URL: http//www.gnu.org) July 1998.
3. J. F. Briesmeister, Ed., "MCNP - A General Monte Carlo N-Particle Transport Code, Version 4C," Los Alamos National Laboratory report LA-13709-M, March 2000.
4. G. W. McKinney, "Enhancement of the MCNP Install Package," Los Alamos National Laboratory memorandum X-6:GWM-93-611, September 1993.
5. Richard Stallman, Paul Visscher, Bret Smith, and Luis M. Arteaga, "GNU's not Unix!," (available URL: http://www.gnu.org/software) October 22, 2002.
6. Richard Stallman, Bob Chassell, Melissa Weishaus, Roland McGrath, and Paul D. Smith, *GNU make*, Version 3.80 (stable), Free Software Foundation, Inc. Boston, MA, October 2002.
7. D. Hagerty, M. Weisshauss, and E. Zaretskii, *GNU Software for MS-Windows and MS-DOS*, Free Software Foundation, Boston, MA (available URL: http//www.gnu.org/order/windows.html) June 2001.
8. "Fortran Compilers," (available URL: http:\\www5.compaq.com/fortran) 2002.
9. "Fortran" Lahey Computer Systems, Inc., Incline Village, NV (available URL: http:\\www.lahey.com) August 2002.
10. Mike Loukides and Andy Oram, "Programming with GNU Software," *Thompson Learning,* Chapter 7 (O'Reilly & Associates, Inc., Sebastopol, CA) April 1995.
11. Andy Oram and Steve Talbott, *Managing Projects with make*, 2nd Edition (O'Reilly & Associates, Inc., Sebastopol, CA) October 1991.
12. L. J. Cox, "Scripts for Manipulating MCNP Source Code," Los Alamos National Laboratory memorandum X-5:99-13(U), April 2000.
13. L. J. Cox, "DMMP Upgrade for MCNP4C," Los Alamos National Laboratory memorandum X-5:00-44(U), November 2000.
14. "American National Standard for Programming Language - Fortran - Extended," American National Standards Institute, ANSI X3.198-1992, New York, NY, September 1992.
15. L. J. Cox, "Standards For Writing and Documenting Fortran 90 Code," Los Alamos National Laboratory research notes X-5-RN(U)-00-28, March 28, 2000.
16. J. T. Goorley, F. B. Brown, and L.J. Cox, "MCNP5 Improvements for Windows PCs," Nuclear Mathematical and Computational Sciences: A Century in Review, A Century Anew, Gatlinburg, Tennessee, April 6-11, 2003, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2003).