Standards for Removing System and Directory Dependence from Programs and Makefiles

Larry Dennis
Department of Physics
Florida State University
Tallahassee, Florida, 32306
internet: larry@fsulcd.physics.fsu.edu

and

Peter Dragovitsch
Supercomputer Computations Research Institute
Florida State University
Tallahassee,Florida, 32306
intenet: drago@ibm16.scri.fsu.edu

March 18, 1994

Abstract

This document describes a set of standards for creating operating system and directory independent makefiles and programs. It is based on using the C preprocessor for dealing with include files and system dependent code, using a standard set of ENVIRONMENT VARIABLES for dealing with different flavors of unix and a standard directory structure to eliminate most normally encountered problems. These standards were developed for use with the software for CLAS, but with minor modifications they are useful for any software package.

Using Makefiles in a System and Directory Independent Manner

This is an attempt to describe the rules needed to develop makefiles which are system and directory independent. Of course, given the current variety of unix platforms, this is not strictly possible. However, it is possible to confine all of the system and directory dependence to one small set of macros. This will make it possible to localize the required modifications to environment variables, and in the case of system dependence make the required changes completely transparent to the user.

The following steps are suggested and an example makefile is given at the end of the text.

1. Use *make* to create programs and libraries.

2. Use environment variables for all system specific defintions.

3. **Any operation** within the makefile which may require system or user specific programs, libraries, flags or directories should use *macros* to define them.

4. Use the following extensions on unix systems.

   - **.a** For libraries.
   - **.o** For object files.
   - **.f** For FORTRAN source files containing a single subroutine.
   - **.c** For C source files.
   - **.h** For C include files.
   - **.src** For FORTRAN source files containing multiple subroutines.
   - **.F** For FORTRAN source codes prior to preprocessing.
   - **.Fsrc** For multiple FORTRAN source codes prior to preprocessing.
   - **.cmn** Files containing common blocks.
   - **.par** Files containing parameters.

5. Use the C preprocessor to include files and to select system specific code within source codes.

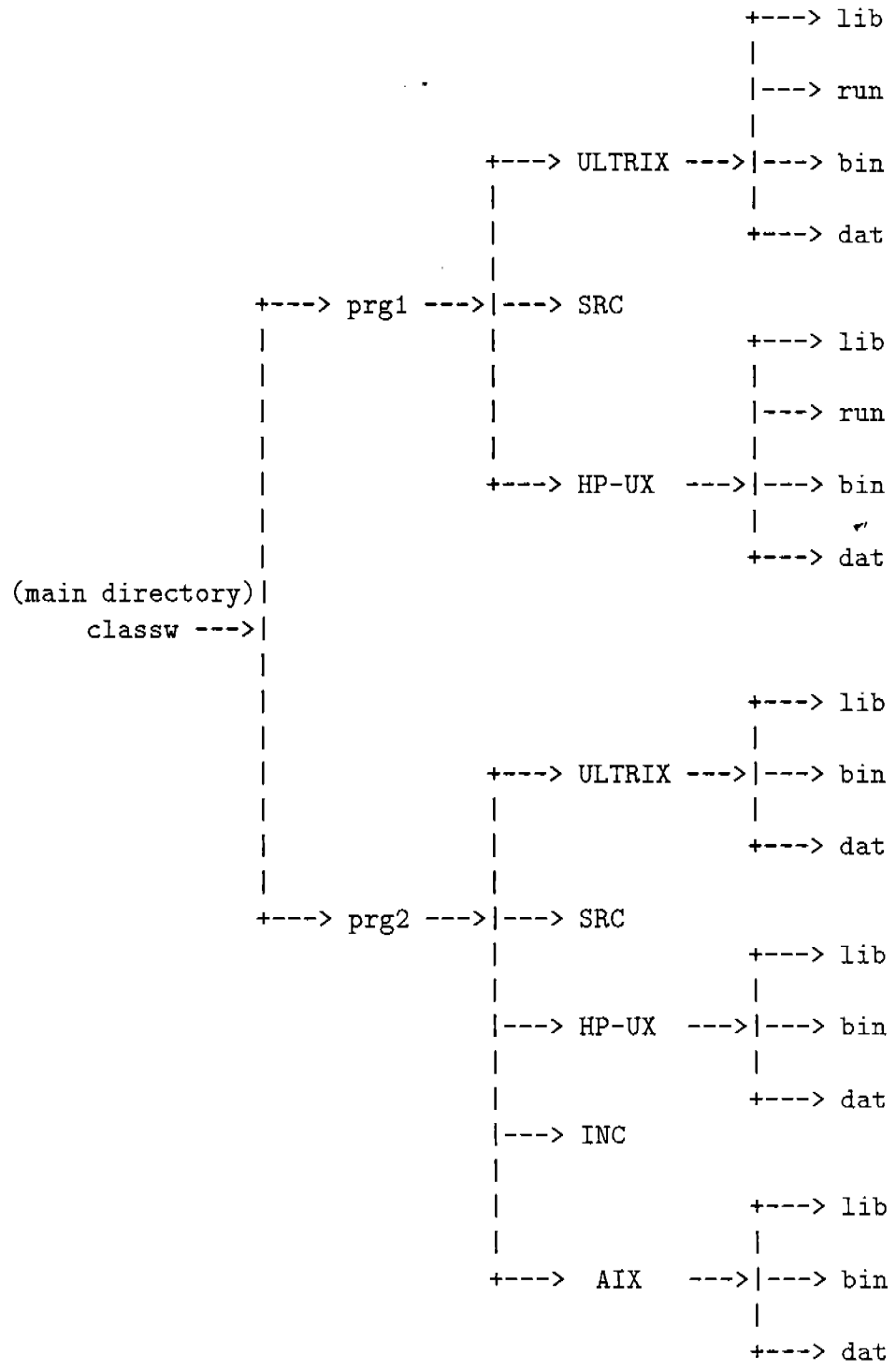
6. Use separate steps within the makefile for each process. For example, do not combine FORTRAN compilation and linking into one step (ie, such as from .f files to executable). The following steps and extenstions are suggested. The order given is the suggested order. Of course one may not need all of the steps for their application.

   - Use the C preprocessor to include files and/or select system dependent code. This converts files with **.F** or **.Fsrc** extensions into files with .f or .src extensions, respectively.
   - Use *fsplit* or the cernlib *fcasplit* to split a large fortan file, one with a .src, into many smaller FORTRAN files having .f extensions.
   - Use the FORTRAN compiler to convert .f files into .o files.

- Use the archiver, **ar**, to create a library, thereby combining several **.o** files into a **.a** file.
- Use the loader to link the files into an executable program.

Standard CLAS Directory Structure

The standard CLAS directory structure is based upon the classw structure on clas01. Thus the root directory for the structure is **classw** (whereever it happens to be). Each program within the clas library has its own directory. Each program directory contains documentation, a **SRC** subdirectory and a subdirectory for each **operating system** the program runs under. The names of the operating system subdirectories must be the string returned by the *uname* command for the systems (AIX, HP-UX, ULTRIX, etc). The structure within the SRC subdirectory is up to the code developer. Each operating system subdirectory should contain a **bin**, **lib** and **dat** subdirectories. The result is a directory structure that looks like that shown on the next page.


(main directory)
classw
prg1
prg2
SRC
INC
ULTRIX
HP-UX
AIX
lib
run
bin
dat

Standard ENVIRONMENT VARIABLES

The following file will set up the environment variables needed for writing system and directory independent makefiles and programs. The machine dependent commands and directories in this file need to be set as required for your computer system and application.

```
#   makeenv skeleton
#           Sets up environment variables for clas software.
#           This file should be executed in PROGRAM's parent
#           directory.
#
#   Created by:  dennisl
#   Created on:  09/20/93    at 14:05:18
#
#    Define Machine Name (ULTRIX,AIX,HP-UX,etc)
#    and MROOT directory (parent of your top level PROGRAM
#    directory
#
setenv MACHINE `uname`
setenv MROOT  `pwd`
#  Machine Dependent commands and flags (Examples are for
#  ULTRIX computers.
#       C Preprocessor
#
setenv CPP 'cpp -P -D$(MACHINE)=1'
#       fslit or cernlib fcasplit
#
setenv FSPLIT fsplit
#       FORTRAN compilation
#
setenv FC 'xlf -c -qextname'
#       Program linking
#
setenv LD 'xlf -v'
#       C compilation
#
setenv CC 'cc -c'
#  Set the following three environment variables as needed
#  for your computer.
setenv CERN_ROOT /users/cern/v93c
setenv XLIB_ROOT /lib
setenv CLAS_ROOT /users/dennisl/ibm/classw
```

HP-UX version for CEBAF's clas02

```
#   clasenv skeleton
#
#           Sets up environment variables for clas software.
#           CEBAF's clas02 environment
#
#   Created by:  dennis_l
#   Created on:  09/27/93    at 19:11:37
#
#  Set library directory environment variables.
#
setenv CERN_ROOT /usr/site1/cern/hp700_ux90/v93c
setenv XLIB_ROOT /lib
setenv CLAS_ROOT /usr/site2/classw
#
#  Set main root directory and operating system type.
#
setenv MROOT     /usr/site2/classw
setenv MACHINE   HP-UX
#
#  Create Environment Variables for
#  Machine Dependent Commands - HP-UX version
#
#       C Preprocessor
#
setenv CPP '/lib/cpp -P -D$(MACHINE)=1'
#
#       fslit or cernlib fcasplit
#
setenv FSPLIT fsplit
#
#       FORTRAN compilation
#
setenv FC 'fort77 -c +e -O +ppu'
#
#       Program linking
#
setenv LD 'fort77 -v'
#
#       C compilation
#
setenv CC 'cc -c'
```

AIX version for SCRI's ibm16

```
#   clasenv skeleton
#           Sets up environment variables for ibm16.
#
#   Created by:  dennisl
#   Created on:  09/29/93    at 15:33:41
#
#  Set library directory environment variables as needed.
#
setenv CERN_ROOT /u/hep/a/heplib/ibm/cernlib/v93b
setenv XLIB_ROOT /lib
setenv CLAS_ROOT /ds9/1/users/dennisl/ibm/classw
#
#  Set main root directory and operating system type.
#
setenv MROOT     /ds9/1/users/dennisl/ibm/classw
setenv MACHINE   AIX
#
#  Machine Dependent commands and flags.
#
#       C Preprocessor
#
setenv CPP 'cpp -P -D$(MACHINE)=1'
#
#       fslit or cernlib fcasplit
#
setenv FSPLIT fsplit
#
#       FORTRAN compilation
#
setenv FC 'xlf -c -qextname'
#
#       Program linking
#
setenv LD 'xlf -v'
#
#       C compilation
#
setenv CC 'cc -c'
```

ULTRIX version for CEBAF's clas01

```
#   clasenv skeleton
#           Sets up environment variables for clas01.
#
#   Created by:  dennis_l
#   Created on:  10/01/93    at 21:31:33
#
#  Set library directory environment variables.
#
setenv CERN_ROOT /usr/site3/cern/dec_ultrix4/93c
setenv XLIB_ROOT /lib
setenv CLAS_ROOT /usr/site2/classw
#
#  Set main root directory and operating system type.
#
setenv MROOT     /usr/site2/classw
setenv MACHINE   ULTRIX
#
#  Create Environment Variables for
#  Machine Dependent Commands - ULTRIX version
#
#       C Preprocessor
#
setenv CPP '/lib/cpp -P -D$(MACHINE)=1'
#
#       fslit or cernlib fcasplit
#
setenv FSPLIT fsplit
#
#       FORTRAN compilation
#
setenv FC 'f77 -c'
#
#       Program linking
#
setenv LD 'f77 -v'
#
#       C compilation
#
setenv CC 'cc -c'
```

You may obtain more information on the enviroment variables and some automatic methods for setting them from the ftp server at ftp.scri.fsu.edu.