

U.S. Department of Commerce
National Oceanic and Atmospheric Administration
National Weather Service
National Centers for Environmental Prediction
5200 Auth Road
Camp Springs, Maryland 20746-4304

Technical Note

NCEP Standards for Operational Codes and Implementations¹

Christopher Peters²
General Sciences Corporation
Laurel, Maryland 20707

June 1998

THIS IS AN UNREVIEWED MANUSCRIPT, PRIMARILY INTENDED FOR INFORMAL
EXCHANGE OF INFORMATION AMONG THE NCEP STAFF MEMBERS

¹OMB Contribution No. 158

²e-mail: wd20cp@sun1.wwb.noaa.gov

NCEP Standards for Operational Codes and Implementations

Christopher Peters
General Sciences Corporation

1. Introduction

This technical note gives an overview and description of the standards expected of Environmental Modeling Center (EMC) programmers when preparing programs to run in the NCEP production suite. Many of the standards and procedures for implementing new codes have been documented by NCEP Central Operations (NCO), however some of these documents are now outdated or, in some cases, exist only by "word of mouth". Thus, there is a need for a survey of the current operational standards and procedures, written for an audience consisting primarily of EMC programmers who may not be aware of the demands and protocols of the production suite.

This technical note is divided into several sections: 1) an overview of the NCEP production suite and the general design principles desirable in a new program, 2) some specifics of how to prepare run scripts and code for implementation, 3) a description of the procedures to follow once scripts and code are ready, and 4) an appendix containing some further documentation of NCO policies and standards.

2. Overview and general design principles

a. The Production Suite

The NCEP production suite consists of those jobs that run under the class "prod" on one of the three NCEP central computers (currently one Cray C90, two Cray J916 machines). This class of jobs is assigned a higher priority than general "batch" jobs, and a certain percentage of machine resources (*i.e.* global memory, job queues) are reserved strictly for production jobs. The amount of the machine reserved for production is not fixed, as it has to be periodically adjusted to meet the needs of production. Currently, the total global memory of 890 Mwords on the Cray C90 is partitioned so that 410 Mwords,

or about 46 %, is reserved for production jobs at all times. The rest is available for general "checkout" (batch jobs). Whereas batch jobs often have to spend long amounts of time waiting "in queue" before running, production jobs generally have to run at fixed times during the day and within a narrow time window. The Supervisor Monitor Scheduler (SMS) controls the timing and release of each production job. Once a job has been released by SMS, it should spend no more than a few seconds waiting in queue before running. Longer delays occasionally occur when the system, for whatever reason, becomes too busy and production jobs begin to back up. In general, however, production jobs must run at predictable times of the day and within a much stricter time limit than checkout jobs. Also, since SMS controls the release of operational jobs, there is no need for production jobs to resubmit themselves using "qsub", as is commonly done in batch mode for automation. These are important considerations for programmers when designing and testing their codes for implementation.

A second consideration is that all NCEP production jobs are operationally supported, 24 hours a day, 7 days a week. The Senior Production Analysts (SPAs) who work for NCEP Central Operations (NCO) maintain the production suite and respond to job failures when they occur. This includes hours outside of the normal work day, such as overnight and weekends. New programs being tested for implementation must therefore be reliable and "bullet-proof", otherwise unnecessary work and grief will occur. The SPAs are trained to handle system related failures, such as a machine being down or a disk system being temporarily unavailable; however, they are not paid to be "debuggers" or beta testers for EMC programmers. EMC staff will be contacted, including during non-working hours if necessary, when a SPA is unable to resolve a programmer-related failure and the failure is hindering the remainder of the production suite. Therefore, it is in the best interest of both EMC and NCO to make sure that all new programs being considered for implementation are fully tested and as robust as possible. Programs should be extensively tested and optimized to ensure that they run as efficiently as possible.

b. Language standards

Source code *must* be provided for all operational programs. Providing only an executable for any job step, no matter how trivial, is unacceptable. Currently, FORTRAN 90 and "C" are the only acceptable programming languages for operational programs. The vast majority of codes currently running in production are written in FORTRAN 77. At the level of run scripts, UNIX korn shell (ksh) is the standard. The distributed-brokered Networking system (dbnet), which allows for efficient and reliable data transfers between computers, is written in Perl. For the purposes of this document, comments on language standards will be restricted to FORTRAN.

In general, code that meets the ANSI standards for FORTRAN 77 will be sufficient for the FORTRAN 90 standard. Many features permissible in a particular vendors' FORTRAN 77, however, are non-standard. In other words, the mere fact that compilation is successful doesn't prove that the code is FORTRAN 77 compliant. On the Crays, compilation with the "-en" option in either F77 or F90 will flag for non-standard usage. Users should attempt this and remove or replace all non-standard features. More fundamentally, a basic design principle of any operational program is that it should be as "machine independent" as possible. With computer systems evolving and changing rapidly, codes that can be easily ported to new systems will become an increasingly more common requirement. See Appendix C for more specific examples.

c. Input/Output

The writing/reading of data to/from output/input files is one of the *least* efficient operations on the Cray computers (and on computational machines in general), particularly when performing input/output (IO) in small increments. In terms of operating on small portions of a file, a personal computer is actually many times more efficient than the Cray C90. Therefore, programmers should try to limit the amount of repetitive I/O in their programs to as little as possible. Large output files used during debugging stages should also be eliminated, as well as any scratch files, which are a programming device

used to get around core memory limitations of older mainframes. These scratch files are largely a legacy of a bygone programming era and should be eliminated. The amount of speed up that can be achieved by switching to dynamically allocating memory for large arrays (instead of using scratch files) can be substantial.

d. Job control

On the SMS workstations, there are task scripts which control the submission of production jobs to the Crays. These task scripts are triggered by either the clock or by the successful completion of other jobs. The task scripts submit jobs on the Crays, called "J-jobs". The J-job is a short UNIX script which initializes the more important shell script variables such as the cycle, the location of source, scripts, and executables. The J-job then executes another UNIX shell script, often referred to as a run script. The run script does the real work, and is usually much larger and more complex than either the J-job or the SMS task script. The run script is the level at which any compiled codes are executed. The J-job sets UNIX shell variables and then exports them for use by the run script and any "child" script. This is an important principle which should be followed at all levels of shell script programming, namely that variables should be defined only once and at the highest level possible. For example, if your run script (parent) has to call other scripts (child scripts), then the child scripts should not be setting global variables (i.e. the date, cycle, etc.) Control variables should always be set in the J-Job, or at least in the main run script, and should be set only once. From a programmer's standpoint, the run script is the only component of job control which they need to know how to modify. Task scripts and J-jobs are maintained by NCO personnel. However, before making changes to the run script, programmers should familiarize themselves with the J-job associated with the run script, and make changes accordingly.

3. Specifics

a. Getting the Date

One of the most basic things any program that runs daily needs to do is to figure out what the date is. In UNIX, there are several different ways to do this. Using the "date"