Stanford Exploration Project, Report 93, November 11, 1997, pages 225–236

Short Note

SEP Computing environment

Robert G. Clapp and Sean Crawley[1]

keywords: *algorithm, computing*

INTRODUCTION

One of the more mysterious things when you begin at SEP is the computing environment. After a year or two you know how things work, but in the meantime you have probably spent many hours rewriting pre-existing code or struggling with fairly pedestrian software which is documented either cryptically, not at all, or in a location unknown to all but the documenter. This paper is an effort to minimize time sinks by giving a basic overview of the SEP computing environment, stating some of the 'unwritten rules', and providing the answers or directions to the answers to common questions of new students.

COMPUTING ENVIRONMENT

There are currently six different brands of computers at SEP, with varying functionality:

- HDS Xterm (machine names: fujiyama, avila, and ciuc)
  - You can't login into these machines. They are directly connected to the HPs with each Xterm assigned a primary HP to connect with.
  - They do not have the same windows environment as all other platforms so part of the setup section below is invalid. To open a window on another machine, type `xon machinename`.
- SUN (oas, taal, pele, zarand, and etna)
  - Oas serves as:

[1]**email:** not available

    * the server for the SUNs, which is to say it holds the `/usr/local/disk`, which contains files that are platform specific;
    * the SUN which holds `/usr/pubsw`, the location of numerous programs available through afs;
    * our ypmaster;
    * our main platform to run Promax and Focus;
    * our mail server, along with etna and taal;
    * our host for the home directories;
    * one of the primary hosts for personal devices.
  - Pele (aliased sepwww, sepftp, taishan) is our web server.
  - Zarand is used for testing CDs.
  - The SUNs are also a good place to check for programs and man pages that you can't find on other machines. For years the SUNs were our primary platform and therefore things were installed most completely on them; this is no longer the case.

- CM-5 (Connection Machine) (vesuvio is actually this is the SUN front-end)
  - To use the CM, you need to program in CM-Fortran, a dialect similar to High Performance Fortran and Fortran 90.
  - The status of the CM at SEP is questionable so large time investments learning to code for CM might not be worthwhile.

- PC running Linux (koko and fuego)
  - Koko is the server for the Linux machines.
  - These computers are very similar to the ones all SEP students will be having at home.
  - GNU software comes pre-installed on these machines so they are going to be the most robust when running Emacs, gcc, etc.

- Hewlett Packard (spur, okmok, sthelens, rainier, hakone, picacho, and pinatubo)
  - Spur
    * serves as then home of many of the personal drives, and the global source directories;
    * s our fastest single processor (135 Mhz);
    * runs afs, files located in /usr/pubsw; and
    * is one of our top graphics machines.

- Silicon Graphics (santorin)
  - Santorin:

    * is a 16-node Power Challenge super computer;
    * is one of our better graphics workstations;
    * is co-owned with Petroleum Engineering Department; and
    * does not have the complete SEP environment (for example Emacs is not installed) to discourage wasting valuable compute time.
  - Make sure to read and *follow* the rules(?).

DIRECTORY STRUCTURE

Over the years much thought has been put into designing the most effective, intuitive directory structure possible. Certain problems arise because diverse people, with different visions, have contributed enthusiastically to certain subsets of the environment over the years. Thus some lack of consistency exists. This section is an attempt to explain what the current vision is, some general rules and guidelines, and places to start looking when you want to find more information.

Home Directories

`/homes/sep/[your-name]` is where you should keep all of your *source* files. By source files we mean:

- your C, C++, Fortran77, Fortran90, Java code;
- any papers you write (see the Report and latex section);
- the makefiles and History files relating to your research; and
- any other important, small, not easily regeneratable file.

This does not mean data, executables, or other large and/or easily reproducible files. We try to limit the content of home directories because:

- Home directories are backed up daily. Large files waste both the tape and time required in backing them up.
- There is a limitation imposed by the disk partition in which the /homes/sep directory resides. When it becomes too full the entire computer system will slow down or crash completely.

There are places for both data and other important, big files that will be covered later (such as seismic data). To help remind you when you are breaking the rules, you are sent a daily message if you have large files (anything over .5MB). If you receive a message, **remove the offending file**.

Personal Devices

`/[your-name]` is your own personal space. It is a place for you to store executables, data, and anything large that you plan to keep for more than a few weeks.

Something you should do in your personal device directory structure is make space for executables. You should create a directory `/[your-name]/bin` and then under this directory create the sub-directories `all` (for executables that are machine independent, such as shell scripts and perl, awk, and sed programs); HP700 (Hewlett-Packard computers); I586 (PC running Linux); SGI64 (Silicon-Graphics with 64-bit OS); SUN4 (SUN running 4.xx OS); and SOLARIS (SUN running Solaris).

Scratch Space

Scratch space is the primary location for data. The name of the scratch space takes the form `/scr[first two letters of the machine name][scratch device number]`. The name for the second scratch device on spur is thus: `/scrsp2`.

These drives are large, ranging in size from 1.7 GB - 40 GB and are attached to all of the main computers at SEP. In order to minimize network traffic, try to write your data to scratch drives attached to the machine you are running on. This will make your job run faster, minimize the chance of the job failing because of network problems, and keep the whole network running at a higher level. This is generally taken care of for you by your ~/.datapath file.

A couple of cautionary notes about scratch devices:

- Auto-erase: Files which have not been accessed for a while are automatically erased. The erasure period varies from 2 weeks to a month depending on the drive. So if something is critical to you it is much better to leave it on your personal device.
- Hogging: If a scratch device gets too full, the top four users of the device will be sent a message. It is fine to to be the major user of scratch space occasionally, but avoid hogging for long periods.

Work Space

These drives take the form `/wrk[first two letters of machine name][number]` (e.g. */wrksp1*)

They are meant for group projects where working in someone else's home directory or personal device can be troublesome. This is also where reports are usually constructed. They are on purpose small, and *shouldn't* be used for scratch drives. The cleaning of these drives is left up to its users unless an obvious case of hogging is observed.

Other conventions and useful directories

- `/usr/local/XXXX` : contain directories trees associated with different software packages. For example `/usr/local/SU` contains `bin, include, lib,` and a link to a `src` directories for the CWP Seismic Unix Software package..
- `/usr/local/src` : location for shared source files
  - `our` : SEP source code (such as sep-distr, the home of SEPlib)
  - `gnu` : GNU source code (Emacs, gmake, etc.)
  - `pub` : public domain source code
  - `their` : 3rd party source code that is not publicly distributed.
- `/advance` : directly for Promax flows
- `/promax#` : Promax data directories
- `/focus#` : Focus flows and data directories

BACKUPS

- Home directories: are backed up daily to disk and weekly to tape. You can restore the daily backups by use of the `restoresep` command on oas (go `man restoresep` to obtain a more detailed description). Only the previous two days are stored on tape, if what you are looking for is older than that, you need contact Computer Administration Support Team (CAST)(**?**).
- Personal devices: are backed up twice annually immediately following each SEP report. The person responsible for backups should provide you with a tape containing your drive contents shortly after the SEP report.
- Others : /usr/local/src, /usr/local/share, /advance, and the root drives on all the servers are also backed up weekly by CAST. Recover files by following the same procedure as for home directories.

PARALLEL COMPUTING

Reflection seismology in general, and SEP in particular, is always dealing with larger datasets and implementing more more complex algorithms to image these datasets. SEP has a long tradition of parallel computing to tackle these problems. Currently SEP has several options available:

- CM5 (vesuvio): Uses message passing and CM Fortran. The continuing status of the CM at SEP is in doubt, and the processor (25 Mhz for each of the 32 processors) makes learning to code on it a questionable decision.

- Cray: We have access to Cray supercomputers as part of our sponsorship agreement. In general this is an underutilized option.

- SGI Power Challenge (santorin): This is the most convenient path to parallel coding. The Power Challenge uses a shared memory model that enables you to code essentially the same as on other machines. The Power Challenge provides automatic parallelization, which means you don't have to do any work to start taking advantage of multiple processors. This automatic parallelization has its limits. It is not allowed in C++, limited in C, and sometimes deceiving in Fortran 77 and Fortran 90. You can also use explicit directives (check the on-line documentation `/usr/sbin/insight` for more information).

SECURITY

Being a user on a UNIX system makes you a potential security hole. Do a couple of things to avoid becoming a cracker's way in.

First, choose a smart password. This means do not have all or part of your login name in your password; do not have your cat's name or your mother's maiden name or Mao's middle name in your password. Do not have any English words in your password. Do have at least 6 characters, at least one of which is non-alphanumeric (such as an ampersand or an equals sign or something). To keep you honest, there is a password cracking program which runs periodically. If it guesses your password it will disable your account.

Second, be careful about logging in as root. If possible, do so on the machine's console. Avoid using 'rlogin' to become superuser, and avoid sending a machine's root password through your modem.

In case you are wondering, SEP has had a major breakin. In the early 1990s, somebody managed to gain access and erase every file on the system. Of course, everything is backed up, and therein lies our real security; but it is tedious, annoying, and a little embarrassing to have to restore your system from tape.

RESPONSIBILITY LIST

Every six months a list is made of computer responsibilities for each SEP researcher. The idea is to share the support workload and to provide a name when a problem or question comes up.

Whenever a new list comes out, read it to find out what your jobs are. When you have a question, read the list to find out whom you should ask.

The list will probably be copied and handed out periodically, but it's easiest to find it on the web (`http://sepwww.stanford.edu/Jobs/jobs.html`).

WHERE DO I RUN PROGRAMS?

There is a long list of computers available to you, and until you connect names with cpu's it is not obvious which computer to use for a given task. The right choice depends on the task; if you are running a program which does geophysics, pick a fast one; if you are running Netscape, stay out of the way of people doing geophysics.

Our more powerful compute servers are:

- Santorin, a parallel 16 processor Silicon Graphics machine.
- Spur, an HP, our fastest single processor machine.

Powerful with an asterisk:

- Oas, a fairly powerful but busy SUN Sparc 20.
- Zarand, another Sparc 20, often unavailable do to CD production and testing.
- Vesuvio, a parallel machine (CM-5) which is very fast if used properly, but requires some expertise.

Use these machines as you need them for research, homework, and so forth. Don't be inconsiderately gluttonous, but don't be shy either; these things are here to be used.

For Netscape, Emacs, and so forth, you should use your desktop machine (in other words, use your 'HERE' window).

SEP SETUP

SEP's setup is quite powerful, and quite flexible, *if* you know what you're doing. If you do not, you can can cause problems both for yourself and others. Before changing the default setup (what you got when your account was created) read the following carefully.

- RULE#1 : Do not modify the global setup.
- RULE#2 : Limit your editing to the files listed in the customization section.

Background

The idea of a global setup was started at SEP in the early 1990s. Before that, a new student would typically hack together parts from other SEP students, struggling to get everything to work for them. The purpose of the global setup was to provide everyone the essential windows environment, Xdefaults, and shell environment that they needed to do their daily work. From this base, it was built to allow further customization and add-ons at the individuals discretion.

Backbone

The basic global setup can be found in `/usr/local/share/setup`. In general, **stay away**. When you first get your account the following files and directories will be created, don't touch them unless otherwise noted:

- .cshrc : If you run csh or tcsh, this is the first file accessed by the shell.
- .bashrc : If you run bash (not supported completely at SEP), this file is accessed.
- .login : Contains instructions for when you login, telnet, rlogin into an account.
- .logout : Contains instructions for when you logout.
- .xinitrc,.X11Startup, .xsession : Contains instructions for when you begin an xsession.
- .Xdefaults : Defaults for X11 applications (such as colors, placement, etc.). You might want to copy this file from $SETUP/skel/ and then modify it.
- .vueprofile : This is the first file checked if you login at an HP console.
- .rhosts : Contains a list of safe hosts. You might want to occasionally add to this, but be careful, this a potential security hole. You can get an updated version (one that has the current list of SEP machines) by typing $SETUP/exec/machines while in your home directory.
- .datapath : This is used by SEPlib to determine where to put scratch files. It is usually set to something intelligent, but you might edit it occasionally. You can update by following the same procedure described in .rhosts.
- Setup : This is directory where you put the customization files described below.

Customization

Here is list of files you can add to your setup directory and what they do:

- shell environment
  - bash_aliases : A list of your own person bash aliases.
  - bash_aliases.[type] (type=sun, convex, ibm, hp,i586, dec, other) : A list of aliases for individual types of machines. Only the aliases corresponding to the machine you are logged into will be added to your environment.
  - alias.generic : Same thing as bash_aliases, only for csh and tcsh.
  - alias.[type] : Same thing as bash_aliases.type, only for csh and tcsh.

  - cshrc.generic : A place to put your personal environmental variables, add to your path, or set other variables.
  - cshrc.[type] : Same as cshrc.generic, but for things specific to a given platform.
  - Xres.screen1 : Change your default screen fonts, etc.
  - login : Personal instructions when logging in.
  - login.[hostname] : Personal instructions when logging into a specific machine.

- windows environment
  - clients.clock : Your choice of an on-screen clock.
  - clients.remote,clients.extra : To add additional windows, such as extra xterms,xload, xbiff, etc.
  - Xres.[type] : Xdefaults for specific platforms
  - tvtwmrc : You can add this to change the look of your windows environment, such as your pull down menus and the color scheme.

MAKE

Make is a smart scripting language for generating commands. The syntax for make can be quite simple or quite complex. Schwab (**?**) provides information on SEP's philosophy and how we use make for our reproducible documents. For a complete description of makes features type `xinfo` (or look in `/usr/local/src/gnu/make-*` the files of the form make.info-* contain the source for xinfo). Another good source for examples are old SEP reports. You can find the latest versions of the SEP report in `/wrkoa1/sepXX`, where XX is the report number

Compile Rules

SEP has a fairly sophisticated set of make rules to make compiling programs easier. To use SEP's make rules you need to put at the top of your makefile:

```
include ${SEPINC}/SEP.top
```

and at the bottom your Makefile:

```
include ${SEPINC}/SEP.bottom
```

All of SEP's compiling rules are based on using a standard suffix convention so let's start by listing the acceptable suffixes:

- .c : C code
- .C : C++ code
- .java : java code
- Fortran77
  - .f : strict fortran77 code
  - .r : ratfor code
  - .rs : ratfor with SAW conventions
  - .rst : ratfor with SAW conventions and temporary arrays
  - .rt : ratfor code with temporary arrays
- Fortran90
  - .f90 : strict fortran90 code
  - .r90 : ratfor90 code (similar to ratfor)
  - .rs90 : ratfor90 code + saw conventions
- .fcm : CM-Fortran

As your program becomes more complex (more files, libraries, compiling options, etc.) their similar increase in your `Makefile` complexity increases similarly.

- In the simplest case, when you write a program that is contained in a single C, Fortran77, Fortran90, or C++ file (for example `program.c`) you just have to type `gmake program.x` to compile the program and to link it with SEP and system libraries.
- If you want to link additional object files to create the executable add the rule:

  ```
  program.x: subs.o
  ```

  where `subs.suffix` (the list above) is another source file in which you wish the executable to be linked with.
- If this level of sophistication is not sufficient there exists a number of predefined variables that you can add to your `Makefile` that perform special functions. Below is a list of the variables, what they do, and where they must be placed in the `Makefile`. All of the variables need to be assigned in the following manner:

  ```
  VAR = mydefinition
  ```

  - Debugging/Optimization

- `DEBUG` [before SEP.top], set it equal to anything and the program will compile in debug mode. If you plan to use a debugger on your code YOU MUST set this flag.
  * `OLEVEL` [before SEP.top] sets the optimization to manual control. It defaults to a fairly intelligent value, so you should normally not set this.
  * `NO_FIX` [before SEP.top], keeps .f90 files (created from .r90 and .rs90 files). You need this if you're using a debugger.
- Adding libraries : when you write more complex programs you will often create your own library. To link with this library using the standard make rules you need to define the following variables:
  * `UF90LIBDIR, UF77LIBDIR, UCLIBDIR` [after]: by using this option you can add a path to the list of directories. The linker looks for libraries of the form `-lmylib.a`.
  * `UF90LIBS, UF77LIBS, UCLIBS` [after]: you can add additional libraries to link with. You can use either the standard form `-lmylib.a`, which will search the system, SEP, and any library path you defined for the libray `libmylib.a` or you can explicitly link in a library using the `-l/my/lib/is/here/libmylib.a`.
- Adding compiling flags : sometimes you might want to have additional compiling flags. Three common reasons are: to add an additional directory to look for include files (of the form `-I/my/include/dir`); to specify the form of a fortran90 code (fixed or free); or to define some preprocessors flags (of the form `-DMYPREFLAG`).
  * `UF90FLAGS, UF77FLAGS, UCLFLAGS` [after] :to add compile flags for a given type of a compiler.
- Additional flags for parallel computing: when writing code for the SGI Power Challenge you often want an extra
  * `PAR_ANA` to keep files that give information about automatic parallelization.
  * `PROCESSORS` to set the number of processors that a code is compiled for. The defaults is to compile the code for 4 processors.

REFERENCES