Software Tools Subsystem
Manager's Guide

T. Allen Akin
Terrell L. Countryman
Daniel H. Forsyth, Jr.
Jefferey S. Lee
Jeanette T. Myers
Arnold D. Robbins
Peter N. Wan

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

September, 1984

TABLE OF CONTENTS

Overview

You are reading the **Software Tools Subsystem Manager's Guide.** The machine-readable text of this Guide is supplied with the Subsystem in the file "=doc=/fguide/mgr" (already formatted) and in the directory "=doc=/guide/mgr" (unformatted).

Purpose

This Guide addresses the needs of the Subsystem Manager: that individual or small group of individuals that is responsible for the installation, maintenance and daily operation of the Subsystem.

Bringing up a large software system is a complicated process that involves several bootstrapping steps. In this case, the Subsystem Manager will be responsible for at least three: the installation of the Subsystem, initial distribution of documentation, and creation of user accounts.

Once the system is running and a sizable user community develops, the responsibilities of the Manager will change. In particular, he will control the generation and distribution of new copies of documentation, maintain lists of active users, update the description of the local hardware configuration, and possibly modify the Subsystem itself by changing either the system code or documentation.

It is the intent of this Guide to provide the Manager with the information necessary to carry out these duties and with procedures for their performance that are recommended by the Subsystem's designers.

For further information on the philosophy and use of the Subsystem, the reader is referred to the Software Tools Subsystem Reference Manual, the Software Tools Subsystem User's Guide, and, of course, to Kernighan and Plauger's Software Tools.

Summary of Contents

This Guide is divided into five sections as outlined in the following five paragraphs.

The **Subsystem Configuration** section deals with the directory structure of the Subsystem. It describes the standard configuration supplied on the Release Tape as well as options for changing directory names, locating directories on specific logical disks, and storing certain portions of the Subsystem off-line.

The **Installation Procedure** covers the initial installation of the Subsystem. It catalogs the contents of the Subsystem Installation Package and Release Tape, provides instructions for loading the tape, and details the steps the Manager must take in

order to make the Subsystem operational.

The **Conversion Procedure** section describes the steps necessary to update your current Subsystem to this release. If you are a new customer, then you need only peruse this section for various caveats, but not necessarily act upon the information contained therein.

The section on **Documentation Structure** describes the nature, coverage and physical location of all Subsystem documentation. Its two sub-sections correspond to the two major Subsystem documents: the Software Tools Subsystem Reference Manual and the Software Tools Subsystem User's Guide.

The day-to-day activities of the Subsystem Manager are covered in **Subsystem Management**. Such concerns as maintaining user accounts and hardware configuration files, adding local software tools and documentation, and operating Subsystem user services are treated.

Subsystem Configuration

The Subsystem is a complex piece of software that resides in several disk directories. This section discusses the standard directory structure and the means provided for changing it, some alternative directory structures, and naming and structural conventions that must be followed if changes are made.

Standard Directory Structure

The following chart outlines the structure of the major Subsystem directories as supplied on the Release Tape:

```
aux
    primes  (file of prime numbers less than 1,000,000)
    spelling
        {dictionaries of English words, place names, etc.}

bin
    {standard Subsystem commands, supported by GT}

lbin
    {locally-supported commands}

temp
    {scratch files created by Subsystem programs}
```

```
| vars
      {subdirectory for each user}
          .mail     (old mail storage)
          .vars     (shell variable storage)
          .template (user template definitions)
|         .hist     (user shell session storage)

  extra
      bin
          {programs called by shell files in 'bin'}

      bug
          {bug reports gathered by 'bug'}

|     cron
|         {example files for use by the 'cron' program}
|
      clist (list of command names used by 'guess')

      gossip
          {file for each user, for messages sent by 'to'}

*     installation (installation name)

      mail
          {file for each user, for messages sent by 'mail'}

      memo
          {file for each user, for memoranda from 'memo'}

      moot.u
          {files used by 'moot'}

      news
*         articles
              {files containing one news article each}
          index       (index to all past news articles)
          subscribers (list of news service subscribers)

          delivery
              {file for each user, for undelivered news}

      phones (list of phone numbers used by 'phone')

      terms (list of terminals attached to the system)

      users (list of authorized Subsystem users)

      fmacro
          {miscellaneous text formatter macro files}

      incl
|         {macro definitions for Ratfor, C, and PMA}

      numsg    (propaganda message sent to new users)
```

```
    template (pathname template definition file)

    vth
        {files describing terminal hardware characteristics}

    ttypes (list of Subsystem-supported terminals and their
        characteristics)

src

    lcl
        lib
            {subdirectories for locally-supported libraries}
        spc
            {subdirectories for local programs with non-
             standard compilation requirements}
        std.r
            {source files for local Ratfor programs}
        std.sh
            {source files for local shell programs}
        std.stacc
            {source files for local Ratfor/'stacc' programs}

    lib
        c$main
                {files to build the C startoff routine ---
                only for sites which license the C compiler}
        cio
            {source files for C I/O library --- only for
            sites which license the C compiler}
        edt
            {source files for line editor routines}
        math
            {source files for 'vswtmath' library}
        sh
            src
                {source files and directories for 'vshlib'}
            {miscellaneous files for 'vshlib'}
        swt
            obj
                {object code files for 'vswtlb' routines}
            src.a
                (archive containing source code for 'vswtlb' routines)
            {miscellaneous files for library 'vswtlb'}
        vcg
            {source files for the vcg support library ---
            only for sites which license the C compiler}
        vcg_main
            {source for main routine for vcg for individuals
            that have written their own front ends for vcg ---
            only for sites which license the C compiler}

    spc
        {subdirectories for programs with nonstandard
         compilation requirements}
```

```
    std.r
        {source files for standard Ratfor programs}

    std.sh
        {source files for standard shell programs}

    std.stacc
        {source files for standard Ratfor/'stacc' programs}

    ext.c
        {source files for C programs in "=ebin=" --- only
        for sites which license the C compiler}

    ext.r
        {source files for Ratfor programs in "=ebin="}

    ext.sh
        {source files for shell programs in "=ebin="}

    misc
        {Subsystem support and maintenance routines}

doc
    build
        guide   (format a new version of the User's Guide)
        man     (format a new version of the Reference Manual)
        rebuild (format a new Reference Manual entry)

    fguide
        {files containing formatted portions of the
         User's Guide}

    fman
        s1
            {formatted standard command documentation}
        s2
            {formatted standard library documentation}
        s3
            {formatted local command documentation}
        s4
            {formatted local library documentation}
        s5
            {formatted low-level command documentation}
        s6
            {formatted low-level library documentation}
        {miscellaneous formatted portions of Reference Manual}

    guide
        {subdirectories containing portions of the User's
         Guide, unformatted}

    hist
        history (history of changes to the Subsystem)
```

```
man
    s1
        {unformatted standard command documentation}
    s2
        {unformatted standard library documentation}
    s3
        {unformatted local command documentation}
    s4
        {unformatted local library documentation}
    s5
        {unformatted low-level command documentation}
    s6
        {unformatted low-level library documentation}
    {miscellaneous unformatted portions of Reference Manual}

print
    guide (print a copy of the User's Guide)
    man   (print a copy of the Reference Manual)

se_h
    {files containing on-line help information for 'se'}

misc
    {Documentation support and maintenance routines}
```

Top-Level Directories. The top-level directories 'aux', 'bin', 'doc', 'extra', 'lbin', 'src', 'temp' and 'vars' are dedicated to the Subsystem. In addition, use of the Subsystem requires that several files be added to the Primos directories 'cmdnc0', 'lib', and 'system'. (A list of these files may be found in the section on Installation Procedures in this Guide.)

Previously, the 'cmdnc0' directory used for Subsystem files had to be the directory to which the console was attached after a cold start. Under the new revision (Revision 19) of the operating system, the 'cmdnc0' directory used for Subsystem support commands must be the 'cmdnc0' located on the lowest numbered logical disk partition, to ensure that users can enter the Subsystem properly. The 'lib' directory used for Subsystem libraries must also be on the lowest-numbered logical disk, so the libraries will be locatable by the loader. The 'system' directory used for Subsystem shared segment files should be the directory in which all standard Primos shared code resides, so that the shared Subsystem programs may be installed during a cold-start.

Directory Security and Placement on Disk. The subsystem is supplied with ACL protections but if the tape is restored onto a password partition, the password protections will override the ACL protections. Although the Subsystem will operate properly regardless of the placement of its top-level directories, substantial reduction in overhead may be had by following these recommendations. The following discussion normally describes the necessary protections for password directories and then follows that with the protections needed in the case of ACL directories.

The directories 'bin' and 'lbin' are accessed very frequently (about once per command) and so should be located on a low-numbered logical disk. The files in these directories must be readable by non-owners but should be protected against alteration by ordinary users. This can be accomplished by placing an owner password on the directories or by making the directories ACL protected directories with permissions of "list", "use", and "read" for everyone. (But see the User's Guide for the Software Tools Subsystem Command Interpreter, for information on "search rules.")

Scratch files created by Subsystem programs reside in the directory 'temp'. The concept of a "temporary file directory" is necessary to allow editing of files on read-only disks or in directories in which the user has non-owner status. Depending on the application at hand, files in 'temp' may grow to excessive size, so it should be placed on a logical disk with plenty of available storage space. It is not accessed frequently, so its placement is otherwise unconstrained. 'Temp' must be public; that is, it must have either a blank owner password or, if it is an ACL protected directory, permissions of "list", "use", "add", "delete", "read", and "write" for everyone.

Subdirectories of the directory 'vars' are used for storage of personal profile information. 'Vars' is accessed infrequently, and is typically small; it may be placed on any convenient logical disk. 'Vars' itself should have an owner password to preserve the privacy of individual users; it may not have a non-owner password. If 'vars' is made an ACL protected directory, it should have permissions of "list" and "use" for everyone. The Subsystem manager must create in 'vars' a subdirectory for each Subsystem user, named by that user's login name. Each of these subdirectories should be protected by an owner password of the user's own choosing and should have no non-owner password. If they are ACL protected, they should be given ACL protection for the owner of "list", "use", "add", "delete", "read", and "write" and "list" and "use" permissions for everyone else. If the directory is ACL protected, the Subsystem will not request a password to allow entry. If it is password protected, the 'swt' command prompts for the owner password and records it internally before entering the Subsystem; this saved password is then used in all future references to the directory by the Subsystem. If the user wishes to change the directory's password, he must do so outside of the Subsystem, so that the Subsystem will be able to exit normally.

Miscellaneous information that pertains to the Subsystem resides in the directory 'extra'. 'Extra' is relatively small and is frequently referenced (to check for messages sent from user to user via the 'to' command), so it should be placed on a low-numbered logical disk. All contents of 'extra' and its subdirectories should be readable by non-owners and free of non-owner passwords. If it is ACL protected, each file should be protected so that everyone can read it and each directory should be protected with "list", "use", and "read" protections for everyone. The subdirectories 'mail', 'gossip', and 'memo' must

be public so that anyone can create a file in them ("list", "use", "add", "delete", "read", and "write"). The files in the subdirectory 'news' should normally be writeable by anyone and its subdirectories public; however, the Subsystem Manager may see fit to restrict subscriptions to the news service by removing non-owner write access to the 'subscribers' file, and publishing of news articles by removing non-owner write access to the 'index' file.

'Src' contains all Subsystem source code. It is extremely large and very infrequently used. It should be placed on a high-numbered logical disk, and, at the discretion of the Subsystem Manager, be protected to prevent unauthorized access.

'Aux' contains several large auxiliary files, particularly the dictionary of English words and the list of prime numbers less than one million. It should be placed on a high-numbered logical disk. Files in 'aux' and its subdirectories should be readable by non-owners, and there should be no non-owner passwords. An owner password may be employed at the discretion of the Manager to enforce security. The ACL protections would be "list", "use", and "read" permissions for everyone. If you uncomment the template =new_word=, and leave it as =aux=/spelling/new_words, then this file needs to writeable by everyone. (Permissions of a/rw, or "read" and "write" ACL permissions.)

'Doc' contains the formatted and unformatted versions of both the Reference Manual and the User's Guide. It should be placed on a high-numbered logical disk. Generally, its contents should be readable by non-owners. It may be owner password protected at the discretion of the Manager, but should not have a non-owner password (ACL permissions of "list", "use", and "read" for everyone). The same applies to all of its subdirectories.

Alternative Directory Structures

For various reasons (lack of disk space or naming conflicts, for example) the Subsystem manager may need to restructure the Subsystem or even remove portions of it entirely. This section describes the actions necessary to reconfigure the Subsystem directory structure to meet local needs.

Templates and Top-Level Directories. File names (alias "pathnames") in the Subsystem feature a number of extensions beyond the capabilities of Primos treenames. (For a full discussion of pathnames, please see the User's Guide to the Primos File System in the Software Tools Subsystem User's Guide.) The extension that bears on Subsystem directory structure is a simple macro substitution facility that goes by the name of "templates." When an identifier enclosed in equals bars (=) appears in a pathname, it is automatically replaced by some appropriate substitution text. In particular, such "templates" have been provided for the names of all Subsystem top-level directories, and all Subsystem code follows the convention that top-level directories

are always named by a pathname containing the appropriate template.

Reconfiguration of the Subsystem's directory structure may be accomplished simply by changing the substitution text that replaces the top-level directory templates. The templates and their substitution text may be found in the file 'template' in the directory 'extra' (on the Release Tape). (This file may be edited with either the Primos editor or one of the Subsystem editors.) For example, suppose that the directory 'doc' conflicts with a local directory of the same name. Edit the template definition file, and change the following line

```
doc         //doc
```

to

```
doc         //tools_doc
```

then change the name of 'doc' to 'tools_doc'. The reconfiguration is complete.

It should be noted that if the name or location of the directory 'extra' or of the template definition file itself is changed, the 'initswt' program run at cold start time must be given a command line argument that specifies the new location of the template file. See the section on Subsystem Installation for further details.

As supplied, the template definitions for all top-level Subsystem directories use the omitted-packname option of the pathname syntax. This means that any time one of these directories is referenced, an ascending search of the MFDs on all logical disks is made until the directory is found. If circumstances prevent placement of the frequently-referenced Subsystem directories on low-numbered disks, it is still possible to avoid the overhead of long directory searches by changing the template definitions to include explicit packnames or logical disk numbers. If this is done, however, the Subsystem must be reinitialized any time one of its directories is moved to another pack.

Off-Line Storage. Certain portions of the Subsystem are not required for everyday usage, and may be removed in order to conserve disk space. The following paragraphs list the directories that may be stored off-line.

The source code directory 'src' is extremely large and may be useless on a production system. It may be stored on tape with impunity (although doing so will cause the 'locate' and 'source' commands to cease functioning).

The on-line documentation supplied with the Subsystem has been found extremely useful in the past, both to new users learning the system and to expert users needing a refresher course on the usage of particular commands. However, none of it is

essential to the operation of the Subsystem; the entire directory 'doc' may be stored off-line. As a less drastic measure, the unformatted versions of the Reference Manual and User's Guide that reside in the subdirectories 'man' and 'guide' may be stored off-line, while everything else remains on disk. (This allows the 'help' and 'guide' commands and the 'h' command of 'se' to function properly.)

If the dictionary of English words and the list of prime numbers are not frequently used, the directory 'aux' may be stored off-line. This affects the commands 'spell', 'speling' and 'rsa', the template =new_words= (if it is defined), and the local math library routine 'prime'.

Installation Procedure

This section covers the procedures necessary for installation of the Subsystem. It lists the contents of the Installation Package and the Release Tape, and provides instructions for loading the tape and initializing the Subsystem. Before reading this section, a thorough study of the **Subsystem Configuration** section of this Guide is recommended.

Subsystem Installation Package

For new customers, the Subsystem Installation Package as sent from Georgia Tech contains the following items:

1 Release Tape
1 Copy of the Subsystem Manager's Guide
1 Copy of the Reference Manual
1 Copy of the User's Guide

Old customers who are updating to Release 9 will only receive the Release Tape, the Conversion Guide, and the Manager's Guide.

Release Tape Contents

The Subsystem Release Tape contains all files and directories necessary for proper operation of the Subsystem. It is in standard MAGSAV/MAGRST format and contains four "logical tapes." Each logical tape contains a number of separate directories that normally would reside on the same logical disk.

Logical Tape 1. The first logical tape contains the following three directories:

cmdnc0 lib system

and these directories contain the following files:

```
        cmdnc0>swt          used to enter the Subsystem
        cmdnc0>swtseg       latest revision of SEG, modified slightly
                            for the Subsystem.  Its output is
                            completely compatible with standard SEG.
|       cmdnc0>snplnk       snaps dynamic pointer links (see below)

        lib>vswtlb          shared Subsystem I/O and utility library
        lib>nvswtlb         unshared version of vswtlb
*       lib>p4clib          bootstrap Pascal compiler run-time-support
                            library
        lib>vedtlb          line editor library
|       lib>vswtmath        high precision mathematical function library
|       lib>shortlb         shortcall routines for FORTRAN
|       lib>vlslb           linked string library
|       lib>vrnglb          ring support library
|       lib>vshlib          shared Shell utility library

|       system>cron.comi    example startup file for 'cron'
|       system>ring.comi    example startup file for 'ring'
        system>sh2030       shared portion of the command interpreter
        system>st2030       shared data area for templates
        system>se2031       shared portion of the screen editor
        system>sw2035       shared portion of the Subsystem library
|       system>sh4000       used to install the command interpreter
        system>sw4000       used to install the Subsystem library
*       system>initswt      used to initialize pathname templates
```

These files must be placed in the appropriate Primos directories
| at your installation. They should be placed in 'cmdnc0', 'lib',
| and 'system' on the lowest-numbered logical disk containing those
| directories.

Logical Tape 2. The second logical tape contains the following directories:

```
bin     lbin     extra
```

'Bin' is the standard Subsystem command directory. It contains the executable versions of all Georgia Tech-supported Subsystem commands.

'Lbin' is a command directory for locally-written tools. Commands in 'lbin' are normally useful at only one installation, or have not been found valuable enough to merit full support.

'Extra' contains miscellaneous information used by various parts of the Subsystem. In particular, it houses the mail, news and memo delivery directories, which tend to grow steadily in size over a period of time.

Logical Tape 3. The third logical tape contains the following directories:

vars temp

These directories should be placed on a disk partition with a large amount of free space, since files in 'temp' may become arbitrarily large.

'Vars' is used to store personal profile information for all Subsystem users.

'Temp' is a special directory dedicated to containing scratch files.

Logical Tape 4. The fourth logical tape contains the following directories:

doc src aux

These directories are all very large and infrequently accessed. They do not normally vary much in size.

'Doc' contains formatted and unformatted copies of all Subsystem documentation.

'Src' contains all releasable Subsystem source code.

'Aux' contains miscellaneous auxiliary files, such as the dictionary of English words and the list of prime numbers.

Loading the Tape

To load the release tape, follow the instructions below:

1. Assign a tape drive:

```
ASSIGN MT0
```

2. Mount the release tape on the assigned drive.

3. Attach to the master file directory on the logical disk containing 'cmdnc0', 'lib', and 'system' (usually disk 0):

```
ATTACH MFD <owner-password> <disk-number>
```

or if the tape is being restored to an ACL or priority ACL protected partition, type

```
ATTACH MFD <disk-number>
```

4. Load the contents of the first logical tape with MAGRST:

```
MAGRST
Tape Unit (9 Trk): 0
```

```
          Enter logical tape number: 1
          <tape label information>
          Ready to Restore:  yes
```

(This will load the files in 'cmdnc0', 'lib', and 'system'.)

5. Attach to the master file directory on the logical disk selected for the 'bin', 'lbin' and 'extra' directories.

6. Load the contents of the next logical tape (i.e., reply "0" to the "Enter logical tape number:" prompt) with MAGRST. (This will load the directories 'bin', 'lbin' and 'extra'.)

7. Attach to the master file directory on the logical disk selected for the 'temp' and 'vars' directories. It should have ample free space.

8. Load the contents of the next logical tape with MAGRST. (This will load directories 'vars' and 'temp'.)

9. Attach to the master file directory on a logical disk with a great deal of free space.

10. Load the contents of the next logical tape with MAGRST. (This will load directories 'aux', 'doc', and 'src'.)

This completes the loading of the Subsystem from tape.

Reconfiguration of Primos for the Subsystem

Primos Revisions 18.0 and above have now used all normally available private memory segments. In order to bring up the Subsystem, it is necessary to increase the NUSEG parameter in the Primos configuration file to at least 43 (octal), up from the default of 40 (octal), to provide private segments for the Subsystem that do not conflict with standard Prime programs. It also implies that you cannot bring up the Subsystem without rebooting your system, unless you already have the NUSEG parameter set high enough.

Initialization of Shared Segments

Several important portions of the Subsystem reside in shared memory segments. Once the release tape is loaded, these segments must be initialized.

One of the enhancements provided with Version 9 is increased security of shared segments. The SNPLNK ("Snap Link") program shown in the commands below runs through a given segment and "snaps" the dynamic subroutine linkages. In other words, all pointers which are set up as dynamic links are turned into real pointers. This is usually done when a program runs, by the Ring 3 pointer fault handler. By snapping all the links at one time, these segments can then be shared as read only. This will

prevent an errant program from scrambling the shared libraries.

Type the following commands on your system console:

```
OPR 1
SHARE SYSTEM>SW2035 2035 700
SHARE SYSTEM>SH2030 2030 700
SHARE SYSTEM>ST2030 2030 700
SHARE SYSTEM>SE2031 2031 700
R SYSTEM>SW4000
R SYSTEM>SH4000
R SYSTEM>INITSWT

SNPLNK 1/2030; SHARE 2030 600
SNPLNK 1/2031; SHARE 2031 600
SNPLNK 1/2035; SHARE 2035 600
OPR 0
```

Ideally, the preceding commands would be placed in your cold-start procedure file CMDNC0>C_PRMO or CMDNC0>PRIMOS.COMI, so they will be performed automatically after every cold-start. Note: if you have changed the name or location of the template definition file or the 'extra' directory, you must specify the new name of the template file on the invocation of 'initswt'. For example, if you have changed the name of the 'extra' directory to 'etc', use the following command instead of "r system>initswt":

```
R SYSTEM>INITSWT ETC>TEMPLATE
```

For installations that had a previous release of the Subsystem, this completes the installation procedure. The Subsystem should now be ready to go. Otherwise, new Subsystem managers should read the next subsection, which describes the remaining steps.

Initial Log-in by SYSTEM

If this is your first Subsystem release, several further steps are necessary to complete installation. As delivered, the Subsystem has only one active user account: that for the login name SYSTEM, which is assumed to be used only by system administrative personnel. Once the Subsystem is loaded and initialized, the Subsystem Manager should log in as user SYSTEM and verify that the Subsystem is working.

Login as user SYSTEM and type the following command:

```
swt
```

If the 'vars' directory was restored on a password partition, 'swt' will prompt for the owner password of SYSTEM's profile directory. The Subsystem is delivered with a null password for SYSTEM, so just strike the RETURN key. The shell (Subsystem command interpreter) will then be executed. If the 'vars' directory was restored on an ACL partition, the 'swt' will not prompt for any password, but will immediately execute the shell. Before it

will accept any commands, the shell will prompt you with "Enter terminal type: ". You should respond with the mnemonic for your terminal type; if you do not know the correct mnemonic, respond with a "?" and the shell will provide a list of acceptable responses. After you have entered an acceptable mnemonic or a RETURN (if you do not wish a terminal type associated with your login session), the shell will be ready to accept commands. You should see a "]" prompt, indicating that the Subsystem is up and running.

Modify the file "=installation=" to contain the name of your installation. (The easiest way to do this from the Subsystem is to type a command similar to the following:

```
echo "Georgia Tech System B" >=installation=
```

Simply replace "Georgia Tech System B" with the name of your installation.)

Before the Subsystem can be released for general use, profile storage directories must be created for all potential users, and their names must be entered in the "=userlist=" file. In addition, descriptions of all terminals attached to the computer must be entered in the "=termlist=" and "=ttypes=" files. For information on these tasks, see the **Subsystem Management** section of this Guide.

Resolving Shared Segment Conflicts

If the segment numbers used by the shared Subsystem programs and libraries conflict with those used by other programs at your installation, you can change the Subsystem segment numbers; however, you must first install the Subsystem as supplied. Also note that you must change the SHARE commands used in your 'c_prmo' or 'primos.comi' cold start command file to reflect the changed segment numbers.

The Subsystem makes use of three shared segments: 2030 for the Shell and system template table, 2031 for the screen editor 'se', and 2035 for the shared library.

The directory for building the Shell is "=src=/lib/sh". In this directory there is a file named "segment" which contains the segment number to be used for the shared portion of the object code. First, change the contents of this file to the desired segment number; then simply execute the Shell program 'build'. This will produce three object codes files, 'sh', the interlude program which should be placed in "=bin=" as 'sh' and "=system=" as 'sh4000', "sh", the shared code which should be copied to "=system=" for automatic sharing by your 'c_prmo' or 'primos.comi' cold start procedure, and 'vshlib', which should be copied to "=lib=". This copying can be done by executing the Shell program 'install'. For example, the following would fix the shell to run in segment 2037:

```
cd =src=/lib/sh
echo "2037" >segment
build
install
```

Also in segment 2030 is the shared portion of 'swt', the Subsystem initialization program. To change its segment, attach (using 'cd') to the directory "=src=/spc/swt.u", change the contents of the file "segment" to the desired segment number, and execute the shell program 'build', just like changing the Shell's segments. Then execute 'install' to copy the shared portion in the file "st" into "=system=" and copy 'swt' into "=cmdnc0=". (Please note that if you change the code for 'swt', it must generate no sector-zero links. If it does, you will wipe out the shell when sharing it!)

Segment 2030 is also used for the storage of system templates. If you must change the location of their storage area, you must alter the loader interface program 'ld' in order to specify a new absolute address for the storage area, then rebuild the Shell (as outlined above), the libraries (as outlined below), the program 'initswt', and any local program that uses the unshared version of the Subsystem library ('nvswtlb'). For further information on the implications of moving the template storage area, please contact Georgia Tech.

The screen editor normally resides in segment 2031. To move it, attach to the directory "=src=/spc/se.u", change the contents of the file "segment" to the desired segment number, then execute the Shell program 'build'. This will yield two object code files: 'se', the interface program that should be placed in "=bin=", and "se", the shared portion that should be placed in "=system=" to be shared in at cold-start time. This copying can be done by executing the Shell program 'install'.

The shared libraries normally reside in segment 2035. To move them, attach to the directory "=src=/lib/swt", change the contents of the file "segment" to the desired segment number, then execute the Shell program 'build'. The object code files, 'vswtlb' and 'nvswtlb' should be copied to the "=lib=" directory; the shared code file "sw" should be copied to "=system="; and the file 'inst' should be copied to "=system=" and renamed "sw4000". This copying can be done by executing the Shell program 'install'.

Segments Used

The following table lists the segments in virtual memory, and how they are used by the operating system and various Subsystem programs.

Segment	Use
0000 - 0401	Operating System

2030	Software Tools Shell
2031	Software Tools Screen Editor
2035	Software Tools Library
2050	Fortran Library
4000 - 4037	User Program
4040	Software Tools Common
4041	Software Tools Stack
4042	Software Tools Common
6000	Operating System Data
6001	Fortran Library
6002	Primos Ring 3 Stack
6003	Operating System Stack

Some user programs use certain predefined user segments for their own use, so you have to be careful where you load your programs. For instance, if a program uses segment 4006, and you run it from the shell, from within the screen editor, you will destroy the screen editor's common blocks. If any of the programs or routines that use these predefined segments are not in use, they are available for user programs. For example, If the screen editor is not in use, segments 4006 and 4007 can be used with no problem, and if the Primos routine P$ALC is not being used (it is used by C, Pascal, and PL/I programs) then segments 4010 through 4027 become available.

Segment	Use
4000 - 4005	User Program
4006 - 4007	Screen Editor Per User Common
4010 - 4027	Primos Dynamic Memory -- P$ALC routine
4036 - 4037	SEG Symbol Tables

Changes for Primos Rev. 19.4

When Rev. 19.4 of Primos is released, you will need to make two changes to allow you to run the Subsystem under it. First, change the definition of =cldata= to be "6002 12", instead of "6002 6". There is a commented out template definition for this in the =template= file. All you have to do is remove the leading comment symbol, for this definition, and comment out the old one. Secondly, in the file =src=/spc/swt.u/init_s.s, change the definition of CLDATA$SM_FAULT_ERR from "XB%+90" to "XB%+91". This is because several Primos internal data structures change their location at Rev. 19.4.

Conversion Procedure

This section contains pointers for re-installing the Subsystem on a system running an older version of the Subsystem.

User Impact

Before trying to use a newer Subsystem release, first study the Conversion Guide included with the new release to determine what, if any, impact will be felt by your user community. If you need complete information on the changes made to programs, you can load the documentation directories "//doc/fman" and "//doc/fguide" from the release tape.

Usually, a Subsystem release is largely compatible with the release it replaces. Those incompatibilities that do exist are noted in the Conversion Guide. If incompatible changes have been made to a command so that you determine it unreasonable to force your user community to convert to the new command, you have several alternatives: (1) you can write shell programs to cover the incompatibilities, (2) in many cases, you can install the old command from the previous release (you may have to recompile it, though), or (3) you can not install the new Subsystem release. If you find it necessary to take the third alternative, please contact us so that we can try to find a better solution to your dilemma.

Compatibility is a different story when you have locally modified versions of Subsystem programs or you have locally written programs that take advantage of "secret knowledge" of the Subsystem's internals. In this situation, you must examine the newly released programs that interface with your local software to determine the changes necessary to interface with the new release. If you have difficulty in this area, please contact us and perhaps we can suggest possible solutions.

Installing the New Subsystem

Once you have determined the suitability of the new Subsystem release and have mapped out a conversion plan, you are ready to test and install the new release. Unfortunately, two different versions of the Subsystem cannot run simultaneously because the Primos shared library mechanism has no provision for duplicate shared entry points. Therefore, to test and install the new Subsystem, you must bring up the new Subsystem only when no other users are running with the old Subsystem.

Before loading the new release, you must first save the old Subsystem files and directories. You can then immediately restore the old Subsystem in the case that the new one malfunctions. If you have about 16 million bytes of disk available, you can just change the directory names:

```
CNAME BIN OLD_BIN
CNAME LBIN OLD_LBIN
CNAME DOC OLD_DOC
CNAME SRC OLD_SRC
CNAME EXTRA OLD_EXTRA
CNAME AUX OLD_AUX
```

Otherwise, you must copy the directories to tape (or removable disk) and delete the original versions. If you copy the files to tape, copy 'extra' and change its name; do not remove it from disk -- you will need it later. Do not change the directories 'vars' and 'temp'; these can be used as they are. Be sure to save the Subsystem files in 'cmdnc0', 'lib', and 'system'.

Then, load the release tape just as explained in the **Installation Procedure** section, leaving out the load of 'vars' and 'temp'. After you place the new Subsystem files in 'system', 'cmdnc0', and 'lib', re-boot your system. (Unless you are familiar with the shared library re-installation procedures, a re-boot is the only safe way to re-install a shared library.) The new Subsystem should now be available for use.

What To Do About Pre-8.1 Programs

Any calls to the subroutine 'init' should be removed from your programs. You should then recompile them, making sure that nothing depends on the value of EOS being less than 0. In fact, no program should depend on any properties of EOS. The Version 8 Compatiblity libraries are no longer supported. You may, at your own risk, continue using the V8-compatible libraries supplied with Release 8.1.

Modifications to Subsystem Files

Once you have installed the new release, you must move your local modifications into 'extra'. First, compare the files "=extra=/template" and "//old_extra/template" (using 'diff', if you like); add any local templates to "=extra=/template". Then update the templates: Exit the Subsystem and type

```
OPR 1
SHARE 2030 700
R SYSTEM>INITSWT
SHARE 2030
OPR 0
```

Copy the following files from 'old_extra' to 'extra':

```
installation
phones
terms
users
```

The 'users' file has changed format (see the section on adding and deleting users under **Subsystem Management**) so you might want to copy it the following way

```
=ebin=/cvusr old_extra/users extra/users
```

=ebin=/cvusr is a shell file that takes the pathnames of the old userlist and the new userlist as arguments, and expands the login names to the 32 character length needed by the 'whois' program. This conversion only needs to be done once, when the new Subsystem is first installed and brought up. New customers do not have to do this, of course.

Delete the following directories in 'extra' and copy them from 'old_extra' using 'cp':

```
gossip
mail
memo
moot.u
news
```

You may want to examine the articles in 'news' before replacing them with your local articles.

You must examine the following files and directories in 'extra' to determine if local changes need to be made:

```
bug
fmacro
incl
numsg
ttypes
vth
```

The shell uses a slightly different shell variable save file format to allow special characters to be encoded as mnemonics and prevent the file from getting scrambled if NEWLINE characters are accidently entered in a variable's value. A program, 'csv', is supplied to make this change simpler for each of the users. 'Csv' takes a list of user names as standard input, opens up the file "=vars=/<user-name>/.vars" and changes the appropriate shell variable values. This will only need to be done when the system is first brought up. If the directory "=vars=" contains only user variable directories then a simple command to perform the conversion is

```
lf -c =vars= | =ebin=/csv
```

or another way, if it contains other files, is to list the names into a file, edit the file, and then redirect the input into 'csv'. For example:

```
lf -c =vars= >user_names
se user_names              # make any changes here
user_names> =ebin=/csv
del user_names             # don't need it any more
```

Now, you may perform any tests that you like. We suggest that at minimum, you try the screen editor, a few shell files, and any other commands that are frequently used on your system.

Documentation Structure

Given a question about the Subsystem, where does one go in order to find the answer? This section attempts to address this problem, at least to the extent of enabling the Subsystem Manager to identify the document required and produce a printed copy of it if needed.

Software Tools Subsystem documentation is divided into two parts: the Reference Manual and the User's Guide. The Reference Manual is mostly technical information: usage summaries, listings of differences from standard Software Tools programs, etc. The User's Guide is mostly "soft" information: tutorials, applications notes, and the like. Each has its place, and must be accessed in its own way.

Reference Manual

The Reference Manual is normally the first port of call for anyone seeking an answer to a specific question. The Manual is composed of a number of entries, one for each command or subprogram in the Subsystem, divided into six sections: standard (i.e., supported by Georgia Tech) commands, standard subprograms, local (i.e., supported by the local installation, or not at all) commands, local subprograms, low-level commands, and low-level subprograms. The Manual is indexed by a simple list of entries and by an automatically generated key-word-in-context index.

A copy of the Reference Manual, formatted for 8.5" by 11" paper at 10 characters per inch horizontal, 6 lines per inch vertical, may be spooled for printing with the command

```
=doc=/print/man
```

Frequent use of this command is not recommended because of the
* manual's sheer size.

Individual Manual entries may be printed with the 'help' command. Simply typing

```
help <command-name>...     or
help <subprogram-name>...
```

will cause the selected Manual entries to be printed on the user's terminal, with a pause between each screenful. The user may also type

```
help -p <command-name> | os >/dev/lps/f
```

to obtain a printed copy of an entry exactly as it appears in the Manual.

'Help' may also be used to read the Manual's index. If "-i" is used in place of a command name, then the Manual's index will be printed. If "-f <pattern>" is used in place of a command name, then only those index entries matching the given pattern will be printed. For example, all commands and subprograms whose Manual entry heading lines contain the word "string" could be identified by typing

```
help -f string
```

This may be useful for people just learning to use the Subsystem, who may know several names for a function they wish to perform, but not the exact command or subprogram they need.

An unformatted copy of the Reference Manual resides in the directory "=doc=/man". A formatted version resides in "=doc=/fman". Should it ever be necessary to rebuild the formatted version, simply type

```
=doc=/build/man
```

User's Guide

The Guide is recommended for anyone just learning to use the Subsystem, as well as for those requiring a deeper knowledge of the workings of some of the more complex tools. It is actually a collection of separate papers, arranged roughly in the order required by a novice user. It is not intended as a quick reference; the Reference Manual performs that function.

Most papers in the Guide contain a tutorial section and an applications notes section, for beginners and experienced users, respectively. Some papers also contain a formal definition section; these would be of use only to those requiring a very complete understanding of the workings of the Subsystem.

The entire User's Guide, formatted for 8.5" by 11" paper, 10 characters per inch horizontal, 6 lines per inch vertical, may be spooled for printing by executing the following command:

```
=doc=/print/guide
```

The Guide is small enough that this operation is not overly expensive in terms of time or paper.

The 'guide' command may be used to print the individual papers that comprise the Guide. For example,

```
guide ed
```

would print the Introduction to the Software Tools Text Editor on the user's terminal, pausing after each screenful. To obtain a printed copy instead, one should type

```
guide -p ed | os >/dev/lps/f
```

It is generally good policy to have a number of copies of the Software Tools Subsystem Tutorial and the Introduction to the Software Tools Text Editor on hand for distribution to new users.

An unformatted copy of the User's Guide resides in the directory "=doc=/guide". Should it ever be necessary to rebuild the formatted copy in "=doc=/fguide", it can be done by typing

```
=doc=/build/guide
```

Subsystem Management

This section outlines the day-to-day responsibilities of the Subsystem Manager: adding and removing user accounts, keeping track of local hardware configuration changes, maintaining local tools, etc.

Adding and Deleting Users

Adding and deleting Subsystem user accounts boils down to the maintenance of one file and one directory.

The list of authorized users is addressed by the template "=userlist=". On a standard Subsystem, it resides in the file //extra/users. There is one line in the user list corresponding to each authorized Subsystem user. Users may appear in any order in the user list, although conventionally it is kept sorted alphabetically. The format of a line in the user list is as follows:

Columns	Information
1-32	User's login name, in upper case, left-justified, blank-padded
33	Blank
34-80	User's name and commentary information

Example:

```
1234567890123456789012345678901234567890...
BURDELL                          George P. Burdell (Development)
```

Whenever someone is added to the Subsystem user community, an appropriate entry must be made in "=userlist=". Whenever a user is no longer authorized to use the system, his entry must be removed.

Each Subsystem user possesses a "profile" directory, which must be created for him by the Subsystem Manager. When adding an account, create the profile directory with the command

```
mkdir =vars=/<login-name> -o <password>
```

if it is a password directory or just

```
mkdir =vars=/<login-name>
sacl =vars=/<login-name> <login-name>=adlurw $rest=lu
```

if it is an ACL protected directory. "<Login-name>" represents the login name of the new user. "<Password>" represents a standard file system owner password, which must be cited by the user when he enters the Subsystem if his variables directory is password protected. Example:

```
mkdir =vars=/gpb -o sesame
```

When a user is removed from the system, his profile directory must be deleted. This can be done most conveniently with the 'del' command:

```
del -sd =vars=/<login-name>:<password>
```

For example,

```
del -sd =vars=/gpb:sesame
```

In addition to the above measures for removing a user's account, the Subsystem Manager should check for any undelivered mail, gossip messages, or news articles. See the **Operation of Communications Systems** subsection below.

Specifying Local Hardware Configuration

The screen editor 'se' and a number of other programs that employ the virtual terminal handler library need to know the type and make of the terminals attached to each AMLC line on the system. This information is contained in the terminal list, which resides in the file "=termlist=" (nominally "//extra/terms"). Each line of the terminal list has the following format:

Columns	Information
1-3	Octal AMLC line number (000-177)
4	Blank
5-9	Three digit decimal user number of associated process, in parentheses
10	Blank
11-16	Terminal type (as recognized by the Subsystem); blank if unknown
17	Blank
18-80	Comments (usually physical location of terminal, etc.)

For example:

```
12345678901234567890...
007 (009) b200   George Burdell's office (Room 1729)
```

The contents of the "=termlist=" file must be kept up-to-date, or the 'e', 'whereis', 'se', and 'term_type' commands will cease to operate properly.

Adding Terminal Types

To extend the terminal type knowledge of the Subsystem, there are three places where changes need to be made in Subsystem files. These locations are the "=ttypes=" file, the "=vth=" directory, and various files under the screen editor directory, "=src=/spc/se.u". In all cases, the mnemonic for the new terminal may not be longer than six characters.

The "=ttypes=" file contains the terminal's general attributes. Its format consists of a mnemonic for the terminal name, the full terminal name, and then a series of flags. These flags currently indicate whether the terminal type is supported by 'se', whether the terminal type is supported by the VTH package, and whether the terminal type represents an upper-case only terminal.

The file "=src=/spc/se.u/how_to_add_terminal_types" gives the details on adding new terminal types to the screen editor. Basically, new code must be added for the cursor movement routines and the editor must be recompiled and installed so that it incorporates the new code. Also, the screen editor's 'usage' routine, as well as the Reference Manual entry, should be updated to include the mnemonic of the new terminal.

To add a new terminal type to the VTH package, an initialization file must be created in the "=vth=" directory, with the same name as the mnemonic that you have chosen for that terminal type. To find out what the format and contents of this file should be, please refer to the other files in that directory for examples.

* Adding Local Tools and Library Routines

When the Subsystem is used for program development, there is a tendency for an installation to collect a set of locally-useful tools. Here are a few suggestions on how to incorporate these local tools into the Subsystem environment.

Of course the primary repository for local commands is the directory 'lbin'. The Georgia Tech 'lbin' is supplied on the release tape as an example of a local command library, and because some of the commands contained therein may be of general use. To place a local tool in 'lbin', simply copy its object code into the directory and use the 'chat' command to make sure it is readable by all users, or if the 'lbin' is ACL protected then the command will automatically be readable.

Since the command search rule employed by the command interpreter may be changed by the user, any number of local command directories may be created. For example, a directory named 'games' might be created for the purpose of keeping employees out of the local arcade at lunchtime. A search rule including this directory might be

```
^int,^var,&,=lbin=/&,=bin=/&,//games/&
```

Any of the programs in 'games' may then be invoked without need for using their full pathnames.

Of course, local subprogram libraries may also be created at will. As with standard Primos, the only steps necessary to make such libraries accessible are to place them in 'lib' and make them readable by all users.

Adding Local Documentation

If local tools and libraries are added to the Subsystem (as outlined above), there will be a need for some means of documenting them. Sections three and four of the Reference Manual are provided for this purpose.

Section three of the Manual deals with local commands. To document such a command, one places a standard documentation file in "=doc=/man/s3" and uses "=doc=/build/rebuild" to place a formatted copy in "=doc=/fman/s3". The documentation may then be extracted by the 'help' command, or printed with the entire manual by executing "=doc=/print/man".

A standard documentation file for a command has several distinguishing characteristics. First, the documentation for a tool named "xxx" resides in a file named "xxx.d". Second, the structure of the file's contents is determined by a number of standard text formatter macro commands. Each macro begins a separate subject in a manual entry. They must appear in the following order: ".hd" (heading), ".ds" (description), ".es" (examples), ".fl" (files used), ".me" (messages issued), ".bu" (bugs and deficiencies), and ".sa" (see also). If a section is empty, it should be omitted entirely.

Once a documentation file has been entered, it must be formatted and the formatted copy placed in a separate directory. The shell program "=doc=/build/rebuild" has been provided for this purpose. An example:

```
=doc=/build/rebuild s3 xxx
```

This would format the file "=doc=/man/s3/xxx.d" and place the result in "=doc=/fman/s3/xxx.d", where it may be accessed by 'help' and 'usage'.

Section four of the Manual deals with local library subprograms. The documentation procedure for subprograms is similar to that for major tools; an unformatted copy of the documentation is placed in "=doc=/man/s4", and a formatted copy in "=doc=/fman/s4". Again, this makes the documentation available through 'help'.

Formatter macros for library routine documentation differ somewhat from those for command documentation. In order, they are: ".hd" (header), ".fs" (function of subprogram), ".im" (implementation sketch), ".am" (arguments modified by subprogram), ".ca" (other subprograms called by this subprogram), ".bu" (bugs or deficiencies), and ".sa" (see also). Again, if a section is empty, it should be omitted entirely.

The rebuild procedure for subprogram documentation is very similar to that for command documentation; simply use "s4" in place of "s3" in the "rebuild" command.

Operation of the 'Cron' Program

One of the new features of Version 9 of Software Tools is the 'cron' program, which allows the system administrator to arrange for the computer to automatically do tasks of a periodic nature. The manual entry for 'cron' (help cron) will give you the information you need in setting up 'cron'. As distributed, "=cronfile=", contains an example entry and a brief description of what cronfile entries should look like, and "=system=/cron.comi" contains an example startup file to initialize 'cron'. 'Cron' executes as an ordinary Software Tools user so it must have an entry in "=varsdir=" and "=userlist=" (see **Adding and Deleting Users** in this section). The 'cron' user must have all permission to the directory "=crondir=". As supplied, 'cron' expects to be run as the system administrator with an ACL protecting "=crondir=" ("=crondir=" is protected with SYSTEM having $all access and everyone else has "list" and "use" privileges).

The supplied Primos routine SPH should be used to start 'cron'. The following command should be entered at the console or placed in the Primos cold start command file ('c_prmo' or 'primos.comi'):

```
SPH SYSTEM>CRON.COMI -U <user-name> -P <project> -V 1 -G <groups>
```

where <user-name> is the name under which 'cron' should run, <project> is 'cron's login project, and <groups> is the list of file system groups with which the 'cron' user should be associated. For example, the following will startup 'cron' with all the default attributes and no groups:

```
SPH SYSTEM>CRON.COMI -U SYSTEM -P DEFAULT -V 1 -G
```

Operation of Communications Systems

The Subsystem provides three different means of passing information from user to user: the postal service, the grapevine, and the news service.

Postal Service. Most electronic communication between Subsystem users is accomplished through the mail system. The 'mail' command stores arbitrary messages in the directory "=mail=" (nominally "//extra/mail"), from where they may be retrieved by the addressee at a later time. All letters are postmarked with the time of mailing and the login name of the sender. Whenever a user enters the Subsystem via 'swt', he is informed if there is any undelivered mail addressed to him.

Grapevine. 'Mail' is not real-time; a letter sent is generally not received until the next time a user enters the Subsystem, or if the user has set his mail notification in the Shell, the next time the Shell notifies him. The "grapevine" managed by the 'to' command alleviates some of this problem. Messages sent from user to user via 'to' are stored in the directory "=gossip=" (nominally "//extra/gossip"), which is searched by the command interpreter before executing each terminal-level command. Thus, the delay between sending a message with 'to' and its receipt by the addressee is no longer than the longest time he spends executing a command. Since users typically spend a great deal of time text editing, the screen editor has also been given the ability to display a message sent by 'to'. See the "om" command in 'se' for details.

News Service. Occasionally an item of general interest or special importance must be delivered to the user community at large. The Subsystem news service provides a means of delivering these articles, while making an archival copy of them and placing their headlines in an index file for later reference.

Since the disk space required to store undelivered news articles may be prohibitively expensive, users who wish to receive news must "subscribe" to the service with the 'subscribe' command. This need only be done once in an account's lifetime. The list of subscribers may be found in the file "=news=/subscribers".

News articles are published with the 'publish' command. 'Publish' takes one file name argument. The named file is copied into the news boxes ("=news=/delivery/<user>") of all subscribers, an archival copy is made in "=news=/articles", and the first line is entered into the index file "=news=/index" for later reference.

News articles that were published incorrectly or are outdated may be removed with the 'retract' command. 'Retract' can remove one or more articles at one time by specifying the article numbers as arguments. A notice of retraction for each article removed is placed in the news boxes of all subscribers who have seen the retracted article; subscribers who have not seen a retracted article are not notified of the retraction. Since removal of outdated news articles is not of great importance, such articles may be retracted quietly by using the "-q" option.

When a subscriber enters the Subsystem, he is informed if there is any news he has not yet seen. He may then retrieve the article with the 'news' command. At any time, he may also use the 'news' command to review the index or any archived articles.

Modifying the Dictionary of English Words

The dictionary of words supplied with the Subsystem is still rather incomplete, and may require additions from time to time.

The template =new_words= is commented out in the system template file. If you make it an active template (by removing the comment symbol), the 'spell' program will write into =new_words= any words it finds which are not in the dictionary. (=new_words= is defined to be =aux=/spelling/new_words.) You may then wish to periodically clean up the file as follows:

```
cd =aux=/spelling
sort new_words | uniq >new_words
```

which will sort the file and remove duplicate entries. You can then go through the file with a dictionary, and remove words which are misspelled.

To add new words to the dictionary, the following procedure is recommended.

Obtain a list of words to be added, or use =new_words= as described above (or both). Obtain from each word as many derivative words as possible, by changing prefixes and suffixes, forming compounds, etc. Check each of these for correct spelling.

Attach to the directory "=aux=/spelling/new" and split the new words into the word files there according to the following scheme:

dictionary	ordinary English words
gazetteer	names and trademarks
abbreviations	abbreviations, acronyms
glossary	computer science terms

Words may appear in more than one file; for example, "assemble" may appear both in the dictionary and in the glossary.

When the new words have been split to their respective files, append these files onto the files of the same name in "=aux=/spelling." You may then empty out the files in "=aux=/spelling/new" by 'echo'ing into them. Go back to "=aux=/spelling", and execute the shell program 'build', which combines the files to form the file 'words', which is used by the spelling check program. You may also run the program 'info' in "=aux=/spelling" for more information.