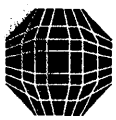
CRAY-3 Software Introduction Manual

CSOS 1.0 Publication Number: 3102



Cray Computer Corporation
1110 Bayfield Drive
Colorado Springs, CO 80906

Copyright

Revision History

Release Date	Summary of Changes
Nov 8, 1991	Initial Release of Manual.

Reader Comments

If you have any comment about the technical accuracy, content, or organization of this manual, please tell us. You can contact us in any of the following ways:

Call our Technical Publications department at (719) 579-6464 during the hours of 8:00 A.M. to 5:00 P.M. (Mountain Time).

Send us electronic mail from a CSOS or UNIX system to pubs@craycos.com

Write to us at the following address:

```
Cray Computer Corp.
Technical Publications Department
1110 Bayfield Drive
Colorado Springs, CO 80906
```

We value your comments and will respond to them promptly.

Table of Contents

Conventions

An effort has been made to utilize the following conventions throughout Cray Computer Corporation manuals and documentation:

Typographic:

Body text is set in 12 point Times Roman font.

boldface	**Boldface** indicates any literal value or name, including CSOS commands, command options, file names, and directory names which a user is expected to type verbatim.
italics	*Italics* represent terms being defined, words or names of variables for which the user supplies exact information and for emphasis.
`typewriter`	Attributes, procedures, macros, and anything resembling source language code are all set in a `typewriter` font. Representations of anything that might appear on your screen are also set in a `typewriter` font.
UPPERCASE	Uppercase is primarily used for programming language statements or functions (e.g.: Fortran statements) and acronyms.
underscore	Underscored words in command lines indicate default values, underscoring is also used to specify accepted option abbreviations.
[ ]	Brackets enclose optional portions of a command or directive format.
a | b	A vertical bar in a command format separates two or more possible choices, one of which you may specify.
choice1	Stacked items indicate two or more literal parameters when only one

choice2	of those parameters may be used.
name(*number*)	Items in text with a *number* in parentheses after them are references to CSOS manual page descriptions. The (*number*) denotes the section of CSOS documentation in which they are described, for example **f77**(1) refers to the CRAY-3 Fortran user command manual page and **ctime**(3C) refers to the C library ctime function.
"*string*"	Quotation marks are used to delimit literal strings, groups of words which are used as a single word or in describing a unique concept or in the invention of new terminology.
...	Ellipses indicate the optional use of the preceding item two or more times in succession.
O'	The capital letter O followed by an apostrophe is used to indicate an octal number; O'177777 means 177777 octal.

Acronyms:

CCC is an acronym of Cray Computer Corporation.

CRI is an acronym of Cray Research, Incorporated.

NFS is an acronym of Network File System.

SVR4 is an acronym of the System V Release 4 system of UNIX Systems Laboratories.

TCP/IP is an acronym of Transmission Control Protocol/Internet Protocol.

Nomenclature:

dataset/file	The terms *dataset* and *file* are used interchangeably in some manuals, except where a difference is explicitly noted. Under CSOS, a file specification can be a file path name.
on | off	The terms on and off are used interchangeably with enabled and disabled, true and false, or with 1 and 0, respectively.

Manual Page Formats:

Many Cray Computer Corporation manuals are comprised of information which is available on-line for use with the **man**(1) command. To retrieve a manual page entry, the following command may be typed, substituting the entry of interest for *entry*:

man *entry*

If there is more than one entry of the same name, all entries are printed. To retrieve the entry for a particular section, an optional section specification may be supplied between the man command and the desired entry:

man *section entry*

For example, the "**man write**" command will display three different man pages, one for the **write** command (section 1), one for the **write** system call (section 2) and one for the **write**

Fortran I/O library function (section 3u). If the user is interested solely in the write system call, then the command "**man 2 write**" may be used. For more information on the man command, and the various sections available, see **man**(1) by issuing the command "**man 1 man**".

Standard typesetting conventions for printed manual pages include:

bold	**Boldface** indicates literal strings, including command names, directory names, file names, path names, man page entry names, options, shell or system variable code names, system call names, C structures and C reserved words.
italic	*Italics* indicate variables, user-supplied (non-literal) options, terms or concepts being defined within the text and for additional emphasis.

It should be noted that these conventions are not necessarily adhered to by various text filters available on the CSOS system (e.g.: **pg**(1), **more**(1)). Some filters will render **boldface** indistinguishable from normal Roman body text. Some filters will render *italics* as underscored. Thus these conventions are not immediately applicable to the visual perception of information displayed with on-line utilities and documentation.

Manual page entries are based upon a common format. The following list shows the order of sections within a manual page and provides a brief description of the information content of those sections. Not all sections shown below are found in each manual page entry.

NAME	Shows the name of the entry and briefly states its function.
SYNOPSIS	Presents the syntax of the entry. The following conventions are used in the SYNOPSIS section: Brackets [ ] enclosing a command line component may indicate that the component is optional. When an argument or operand is given as *name* or *file*, it always refers to a file name. Ellipses ... indicate that the preceding command line component may be repeated. An argument beginning with a minus, plus or equal sign (-, +, or =) is usually an option.
DESCRIPTION	Discusses the entry and its purpose or function in detail.
OPTIONS	Lists and describes the entry's options, their purpose, use and potential interactions.
IMPLEMENTATION	Provides details for using the entry.
NOTES	Points out items of particular importance.
CAUTIONS	Describes actions which may alter data or produce undesirable results.
WARNINGS	Describes actions which may produce harmful effects on the system or its users.
EXAMPLES	Provides examples of usage for the entry.

FILES	Lists files that are either part of the entry or related to it.
RETURN VALUE	Describes return values for the entry and possible error return codes.
MESSAGES	Describes the informational, diagnostic and error messages that may be produced by the entry. Self-explanatory messages are generally not listed.
DIAGNOSTICS	Describes diagnostic messages produced by the entry.
BUGS	Lists known bugs or deficiencies of the entry.
SEE ALSO	Lists man page entries or manuals which contain information related to this manual page entry.

Chapter 1 INTRODUCTION

1.1 Purpose

This manual provides an introduction to the software products offered with the Cray Computer Corporation CRAY-3 supercomputer system. These software products include the operating system, utilities and libraries, external interface software, and applications software. Software development, at Cray Computer Corporation, is an ongoing activity with improvements in performance, reliability, functionality, and maintainability released on a periodic basis.

The software offers a consistent user interface to the architectural features of the CRAY-3 system. The system software provides facilities that enhance user productivity through easy-to-use and flexible interfaces. These interfaces provide the ability for users to develop and maintain very large, complex applications programs, locate and remove program errors, and verify the correct performance of the applications.

1.1.1 Organization of This Manual

This chapter provides a general introduction to CCC's software philosophy, heritage, and product set. Subsequent chapters explore each of the key

components of the product set in more detail. References to other appropriate documentation are signified by the use of a book icon:

1.2 Software Philosophy

CCC's general philosophy on software is to take advantage of commercially available software solutions wherever practicable. This has the dual benefits of:

- Allowing CCC's software development resources to focus on CRAY-3 specific attributes, especially as these relate to delivering optimal performance from the CRAY-3 system.

- Allowing CCC to exploit "standard" solutions that are also available to other computing platforms. This provides CCC users the opportunity to exploit common products across a heterogenous network.

A few years ago, there were few commercially available software products that had real applicability in the supercomputer environment. Fortunately, today the picture is very different. While there are still many areas in which supercomputer users are the first to recognize the need for a particular capability, these same needs are increasingly being mirrored in other computing environments. It therefore follows that, in the future, one may expect to see an increasing proportion of supercomputer software needs satisfied by commercially available, multi-platforms products.

CCC will play its part in promoting this concept by making the software products that it develops, and which might have a wider application within the industry, available via recognized third party providers. CCC's purpose here is not entirely altruistic, but rather to continually reduce the amount of generic software that it needs to maintain and integrate with new third party products.

1.3 Software Heritage

Much of CCC's current software is based on software inherited from Cray Research at the time that CCC was spun off as a separate company. This gave CCC a base of mature, facilities-rich software upon which to build. CCC's ongoing software development may be viewed as an evolution from this initial base with a specific focus on meeting the needs of the CRAY-3 user. With the passing of time, there will naturally be some divergence in the CRI and CCC product offerings, particularly as CCC seeks to utilize standard third party, rather than proprietary, software solutions wherever possible. However, CCC remains conscious of the value of providing a simple migration path for the CRI user to a CRAY-3 system and will therefore maintain the inherited CRI user interfaces where this is practicable.

1.4 Product Set

CCC's software product set provides an environment that allows the CRAY-3 user to fully exploit the capabilities of the system through standards-conformant interfaces. In addition, it recognizes that in the near term, many CCC users will be migrating from a CRI environment and would therefore benefit from a degree of compatibility with CRI interfaces. CCC software supports the following user environment:

- Applications written in Fortran or C
  - with CRI extensions for compatibility
  - With automatic optimization, vectorization, and aids for multiprocessing

- UNIX operating system interface
  - currently based on an AT&T SVR2 kernel and SVR3 commands
  - with CRI and CCC extensions, eventually to be replaced by an SVR4 base

- Full multiprocessing support
  - multiple job streams, multitasking of individual jobs

- Full symbolic debugging capabilities
  - including debugging of multitasked codes

- Windows based user interface
  - via OSF/Motif, OPEN LOOK, X Windows or Curses environments

- Network access via standard UNIX protocols
  - Transmission Control Protocol/Internet Protocol (TCP/IP), Network File System (NFS)

This manual describes each of the major components of CCC's software product set under the following headings:

- CRAY-3 Colorado Springs Operating System (CSOS)
- CRAY-3 compilers and multitasking features
- Utilities, tools, and libraries
- Communications software
- Applications software
- Foreground processor software

Appendix A to this manual reviews the various types of documentation provided by CCC and provides a catalogue of existing and planned materials.

An AT&T UNIX System V Release 4 source license is required for CSOS. Customers must obtain this directly from UNIX Systems Laboratories, a subsidiary of AT&T.

Chapter 2 CSOS

The CRAY-3 operating system, CSOS, is based on Cray Research's UNICOS 5.0 operating system which, in turn, is based on AT&T UNIX System V, with extensions ported from 4.3BSD UNIX. CSOS includes supercomputer-specific extensions developed by Cray Research (as part of UNICOS 5.0) and further extensions developed by Cray Computer Corporation.

CSOS is an operating system that can execute many processes at the same time. It provides multiprogramming and multiprocessing services, permitting a single user or many users to execute processes simultaneously. In addition, CSOS supports the application of multiple processors (CPU's) to a single process through multitasking. These characteristics enable CSOS to provide exceptional problem-solving performance and ease-of-use, which complements the computational capability of the CRAY-3.

A primary advantage of CSOS is its ability to fit effectively into existing environments as a part of a computer network. CSOS provides connectivity to a variety of user environments, including multi-drop, high-performance networks, and point-to-point access.

Cray Computer Corporation developers are committed to improving the performance of all parts of CSOS with each release of the product. Enhancements have been made to basic UNIX, but the traditional UNIX theory

of operations is preserved in the CSOS environment. The enhancements that enable CSOS to exploit the power of the CRAY-3 include the following:

- Enhanced I/O capabilities to deliver supercomputer performance
- File system enhancements that improve traditional UNIX file-space allocation methods. These enhancements provide more effective disk usage and optimal I/O throughput.
- Multiprocessor and multitasking support
- Additional networking software
- Batch processing and process/job recovery capabilities
- Accounting features
- Enhanced resource usage for CPU's, memory, batch access, and interactive access
- Debugging aids
- Various tuning and optimization utilities
- Implementation of the group membership method available in 4.2BSD, providing more effective control of access to the file system
- An optional security feature that provides support for concurrent processing of sensitive information at multiple security levels. Design specifications for this feature were derived from the U.S. Department of Defense (DoD) Trusted Computer System Evaluation Criteria.

2.1 CSOS DIFFERENCES FROM UNICOS 5.0

CSOS is an incrementally enhanced version of the CRI UNICOS 5.0 product, designed to provide a functional yet stable software environment for early users of the CRAY-3 system. Where practicable, it takes advantage of technologies being developed for the AT&T SVR4 product platform, where these do not compromise the goal of stability. CCC's intent is that the proprietary components of UNICOS will be progressively phased out in favor

of SVR4 standards conformant, portable versions as these become available. An important difference between CSOS and UNICOS 5.0 is that CSOS is written in Standard C, as opposed to the Portable C (PCC)/Standard C mixture found in UNICOS 5.0. This simplifies maintenance and improves the inherent stability and performance of the product.

Several new features have been added to the original UNICOS 5.0 base. These additions support new capabilities in the applications development and system administration areas (see following sections) as well as providing additional tools for systems programming.

The major, essential enhancement in CSOS is support for the new CRAY-3 hardware and its associated subsystems. Where practicable, all of the code associated with new hardware support has been tested under simulation in advance of the availability of actual hardware.

2.2 CSOS HARDWARE SUPPORT

- CRAY-3:
  - Up to 16 CPU's
  - Up to 1 Gigaword memory
- CRAY-2:
  - Up to 8 CPU's
  - Up to 512 Megaword memory
  - 1 or 2 foreground processors
- Disk:
  - CRI DD49 and DS40 subsystems
  - CCC RAID via HIPPI[1]
- Network:
  - Network Systems' HYPERchannel
  - VME
- Console:
  - Sun compatible

1. From mid-1992.

2.3 CSOS CONNECTIVITY

Since the CRAY-3 system will usually operate as the computational node in a network, physical and logical connection mechanisms are critical. CSOS supports network connections through Network Systems Corporation (NSC) interfaces as well as VME for workstations. In the future it will support network connections via HIPPI, as the appropriate hardware becomes available from network vendors. In the protocol area, NFS and TCP/IP protocols are fully supported, as is the new BSD 4.4 Telnet capability which provides improved line mode processing, removing inappropriate interrupts and character handling from the CRAY-3.

2.4 THE STRUCTURE OF CSOS

The structure of CSOS is pictorially represented by a set of five concentric circles as shown in Figure 1. The innermost circle (1) represents the hardware system architecture. The next outward radiating circle (2) represents the operating system, called the kernel. Circle 3 outside of the kernel represents the system calls or software protocols of CSOS. Circle 4 represents CSOS utilities, libraries, languages, and other Cray Computer and vendor software applications. The outermost radiating circle (5) represents the shell.

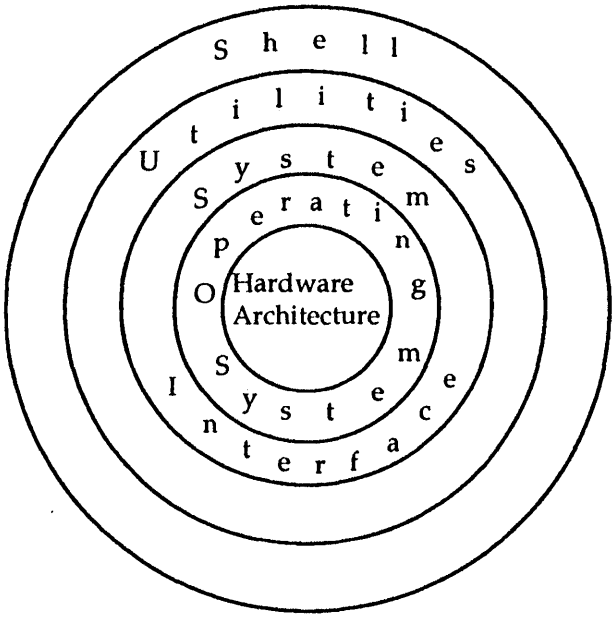


Figure 1 The Structure of CSOS

2.4.1 The Kernel

The CSOS kernel is a collection of memory-resident routines that interact with the underlying hardware architecture through hardware interfaces such as interrupts, status words, registers, and interact with programs through system calls. The kernel routines perform the following major functions.

- Manage fundamental hardware resources: CPU's, memory, channels, and peripheral devices
- Transfer data and computer programs between the CRAY-3 common memory and peripherals
- Allocate space for files
- Monitor the use of resources: log and retain information about errors encountered and recovery from these errors where possible
- Provide information regarding processes and resources
- Schedule the central processors by activating user and system processes by using a prioritized scheduling algorithm

2.4.2 System Interface

Programs make requests for kernel services through the system calls. System calls provide the functional building blocks for all the utilities and application programs that use the CSOS operating system.

2.4.3 Utilities

A large set of simple utilities exists that provides frequently used functional services for the user. CSOS is designed to accommodate virtually an unlimited number of additional utilities; existing utilities do not need to be modified to work with new ones. CSOS provides an environment that allows complex tools

to be built by combining several utilities to work together. Additional information on the utilities is provided later in this section.

2.4.4 The Shell

The shell is a powerful utility that provides the user interface to CSOS. Shells process user commands and provide a programming language to invoke utilities and application programs and control the flow of users' work. The shell initiates the processing of commands entered interactively by a user or as a script for batch processing. The shell interprets a command as a shell command, a program name, or a script. Through system calls, the shell requests programs to be executed or services to be performed. With CSOS, users are able to create their own powerful shell programs or scripts, thus customizing the environment for their needs. CSOS supports the two most popular shells, the Bourne shell (System V UNIX) and the C shell (BSD).

2.5 THE CSOS FILE SYSTEM

A CSOS file is defined as a sequential stream of data. CSOS does not impose a file structure upon the file. A file may contain data, programs, or both. With the necessary access permissions, the user can store, retrieve, and modify the information contained in a file; the user can also create new files and delete existing ones.

The CSOS file system is structured in the form of a hierarchical "tree"; all disk space is grouped into manageable file system pieces that become "branches" of the tree. This arrangement of files is efficient because associated files may be grouped together by the user, and the amount of searching for files is bounded; that is, each level of searching is restricted to the current directory unless the user specifies alternative search paths. This structure limits possible damage due to hardware failure and provides for effective control of the allocation of disk space.

CSOS maintains six kinds of files: directories, regular files, block special files, character special files, fifo (first in first out) special files, and restart files. See Job and Process Recovery Features for more information about restart files.

2.5.1 Directories

Directories provide user-specified structure to the CSOS file system and provide mapping between the disk parameters that describe a file's location and its file name. A directory entry points to a file using one or more pointers, also referred to as links.

The system rules governing links to directories determine the tree structure on the file system. The origin of the file system is called the root and is specified by the / character. A list of directories, separated by slashes and followed by a file name is called a path name and allows the user to specify any file or scan any part of the tree structure on the file system. Directories can be read, written, or copied (shared) provided appropriate permission has been established.

As noted earlier, CSOS groups available disk space into manageable file system pieces. This has two effects at the user level: file space can be exhausted in one but not all of these pieces, and no links can exist between one file system hierarchy and another.

The individual user has a directory called home for the user's personal files. CSOS allows the user to create files and subdirectories within the user's home directory; similarly, the user can create files and additional subdirectories within any subdirectory they create.

CSOS maintains several directories for its own use, including the root directory and directories containing information such as the utility tool set, system logs, peripheral device configuration data, and administrative and configuration programs.

2.5.2 Regular Files

Regular files contain an arbitrary sequence of bytes such as program text, executable binary files, or data. The system neither requires nor imposes structuring of these files. Therefore, regular files are sequences of bytes that are randomly addressable. When a structure must be introduced in a file (for example, to format compiler output to be acceptable to the loader), the programs involved accomplish the task without the intervention of CSOS.

2.5.3 Special Files

Each I/O device is associated with a special file, which is read and written as an ordinary file permitting device-independent I/O operations. Any request to read and write a special file results in the activation of the associated I/O device. Device files contain the specifications for interfacing with any of the named devices. Special files are protected from indiscriminate use. This treatment of I/O devices results in several operational advantages:

- I/O operations to files, devices, or fifo special files are usually indistinguishable from one another.
- Files and devices have the same syntax and meaning, so a device name can be passed to programs expecting a file name as an argument, and vice versa.
- Files and devices are protected by the same security mechanism.

There are two types of special files: block special files and character special files. When reading from or writing to a block device, data may be staged through a cache of system buffers. Disks are the current block devices supported. Those devices that do not use the system buffer cache are loosely referred to as character devices. For example, a console is a character device that uses its own buffers. The disk can also be read or written as a character device; in this case, the system buffer cache is not used, and the data is moved directly between the device and the user's buffer. The following are special files currently supported under CSOS.

- Disk drive interface
- Error logging interface
- Memory-resident files
- Network entry device
- Pseudo terminal driver
- General terminal interface
- Controlling terminal interface
- Special CPU functions interface
- High Performance Parallel Interface (HIPPI)
- Low-speed channel interface
- SECDED maintenance function interface
- HYPERchannel interface
- Security log interface

2.5.4 Processing Flow

This section explains the hierarchical structure of the file system and the operation of commands.

2.5.4.1 The Hierarchical Structure

The hierarchical structure of the file system is, in essence, a network of directories and files. Since the directories are used for routing searches through the file structure for the desired directory or file, the directories act as indexes. Each directory contains the addresses of directories and files of the next subordinate level of the hierarchical structure; thus, the tree structure is used to describe the file organization. Figure 2 is an example showing one possible configuration for part of the user's file system. The contents of the directories and subdirectories shown are often as follows.

Directory	Contents
/	Root Directory
/bin	Executable files containing the compiled versions of user level CSOS commands.
/usr	Subdirectory containing system files and/or user subdirectories and files; other subdirectories may also contain system files.
/dev	Special files describing software interfaces to system hardware devices; generally, all files that represent peripheral devices such as terminals and printers are kept in this directory.
/usr/bin	More executable files; this directory commonly contains additional user level CSOS commands.
/usr/local	Subdirectories containing compiled versions of local commands, data files, and so on. The names of subdirectories within /usr/local are often defined by the site.
/usr/include	Header (.h) files and subdirectories containing header files. Header files contain definitions of symbols required by the kernel or used by library routines, structure definitions, and so on.
/usr/include/sys	System header files.
/usr/man	Files containing on-line manual entries.

The /etc directory (not shown in the diagram below) usually contains administrative commands and data files.

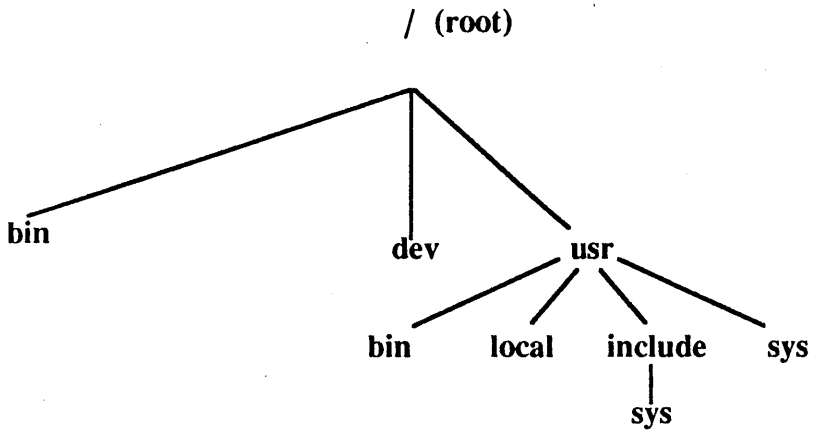


Figure 2 The Hierarchical Structure of CSOS

2.5.4.2 The Operation Of Commands

When a user enters a command, the shell, serving as a command interpreter, reads the command and begins to search in one of several directories, in the order specified by the user's path. The operation of the shell is suspended when the command (the process) is being executed. The process, which is normally an executable file, contains the instructions for controlling the operation and making requests of CSOS to perform the user's work. When the process completes, CSOS resumes operation of the shell. The shell then prompts the user to indicate it is ready for the next command. The shell also supports the ability to run processes in the background while continuing the execution of other work in the shell.

Many commands support the compilers and software products that Cray Computer Corporation provides to enhance the power, usability, and portability of the software.

The Fortran and C language compilers, as well as the assembler, relocatable loaders, and other utilities, are all accessed by commands. These software products are described later in this manual.

2.6 STANDARD CSOS SECURITY MECHANISMS

CSOS security provides protection from unauthorized users gaining access to the computer system and from authorized users tampering with other users' files, and permits access and sharing of files when an authorized user needs to access shared files. The security features specified in the DoD Trusted Computer System Evaluation Criteria are optionally available in CSOS and are described later in this chapter. The following standard security measures are available in CSOS:

- Password protection
- File and directory access permissions
- Encrypted files

2.6.1 Password Protection

The user password is maintained in a password file in an encrypted form. The encryption is a one-way transformation. Access is permitted after matching the encryption of the entered password and the encrypted password stored in the file.

The user data base (UDB) contains one entry for each user and contains the login name, the encrypted password, the numerical user ID (UID), the numerical group ID (GID), the comment field, the initial working directory, and a default shell.

2.6.2 File and Directory Access Permissions

Each directory and file in the file system is associated with a set of permissions that specifies who can access the file and how it can be accessed. The permissions specify read, write, and execute permission for the user, group, and all legitimate users. The permissions are checked when any user except the superuser attempts to gain access to the file. CSOS supports the multiple simultaneous group membership method available in 4.2BSD. This means that all authenticated group ID's are available for simultaneous access permission testing. This method of group ID management permits the sharing of files within an authenticated group or among project team members.

2.6.3 Encrypted Files

The crypt command allows the user to encrypt files to make them unreadable. A key, supplied on the command line, provides a transformation that the data undergoes before it is stored. The decrypt command restores the file to its original state.

2.7 PROCESSING FEATURES

CSOS and UNIX have a similar philosophy, structure, and function. The major differences consist of the following processing features available to CSOS users. These features enable CSOS to exploit the power of the CRAY-3 supercomputer:

- Enhanced I/O capabilities
- Multitasking features
- Batch processing features
- Job and process recovery features
- Interactive environment
- Real time capabilities
- CSOS security feature
- Accounting features

2.7.1 Enhanced I/O Capabilities

The following I/O extensions have been included in CSOS:

- Asynchronous I/O
- Large I/O block sizes
- File system that span disk volumes
- Disk striping
- Handling of disk flaws
- File allocation
- Raw I/O operations
- Varying input modes

- Logical Volume Manager/Configuration Manager
- Support of RAID disks

2.7.1.1 Enhanced I/O Capabilities - Definitions

Asynchronous I/O. Asynchronous I/O capabilities are supported by the queued compound asynchronous read and write system calls allowing a user to send a list of I/O requests to the system to be handled asynchronously. The queued compound asynchronous I/O system calls support multiple I/O requests on a file to be queued with a single system call. Each I/O request in the list provides for maximum control of the desired I/O characteristics. This allows a user to overlap CPU processing with the transfer of data to or from memory.

Large I/O Block Sizes. CSOS supports file systems with very large block sizes. This permits optimum space usage on a disk volume, and also improves file contiguity improving I/O performance and data transfer rate. Large I/O block size reduces the number of disk I/O operations required to locate and transfer file information. Users are able to request contiguous disk space for a file.

File Systems that Span Disk Volumes. CSOS allows a file system to span physical device volumes by use of a cluster descriptor file. File systems can be linked into one file system either at configuration time or after the system is booted. File size is restricted only by the combined size of the disks in a file system.

Disk Striping. CSOS supports disk striping which allows the distribution of a single file across several physical devices or volumes to decrease access time. This enables the parallel transfer of data to all devices in the striped group for enhanced performance and for maintaining high transfer bandwidth. Striping files onto several disks makes concurrent access to sequential blocks of a file possible. In some applications, I/O performance can be significantly improved with striping. An example of foreground processor striping is for a system swap file. Striping is specified at file open time or during CSOS configuration.

Handling of Flawed Disk Media. Flawed disk blocks are marked and not used for file allocation, allowing devices with physical flaws to remain in service.

File Allocation. A sector-linked list method and a track-bit map method are used to allocate disk space. Allocations of eight sectors (32,768 bytes) or less are allocated with the sector-linked list method. Allocations with greater than 8 sectors are allocated with the track-bit map method. This permits optimum space usage on a disk volume and improves contiguous use of file space and I/O performance.

Raw I/O Operations. CSOS supports direct I/O transfers into user memory by bypassing the system I/O buffer. The direct transfer block size is 172,032 bytes for the DD49 disk and 196,608 bytes for the DS40 disk subsystem. Raw I/O capability can be selected at file open time. Once selected, raw I/O is done with the normal read and write system calls. Bypassing the system I/O buffer yields improved performance for those programs that select the raw I/O capability.

Varying Input Modes. CSOS supports both line-at-a-time and character-at-a-time input modes. This allows the front end computer systems running the TCP/IP communications utility telnet to perform line editing. In the front end mode, the front end system forwards every character to the CRAY-3 as soon as it is typed. CSOS instructs a front end system running telnet to alternate between line-at-a-time and character-at-a-time input modes, as appropriate to the application running on the CRAY-3. This adds enhanced flexibility and performance to system operation.

Logical Volume Manager. The Logical Volume Manager provides a new level of flexibility and control in the way in which file systems may be allocated and administered on-line. Used in conjunction with the graphical application support libraries described in Chapter 4, this provides the system administrator with a graphical interface through which he can monitor and redistribute the use of available disk space, without impacting the running system. The Logical Volume Manager is a collection of kernel, user mode utilities, and user mode daemons that provides:

- Partitioning of physical disks into slices
- Grouping of slices into partition groups that act like single disk partitions
- Configuration data distributed over managed disks
- Flaw recovery and management

System Configuration Manager. The System Configuration Manager works in conjunction with the Logical Volume Manager to provide a graphical icon and menu-driven interface for managing, administering, and configuring the CSOS system software and the hardware on which it runs.

RAID Disk Support. CSOS provides support for CCC RAID disk subsystems via HIPPI channel connections[2].

2.7.2 Multitasking Features

CSOS supports the parallel processing of a user's application program. Parallel processing with several processors can significantly improve program performance and programmer productivity. Independent parts of a single program can be executed concurrently by several processors using a software technique called multitasking. CSOS also supports the execution of multiple job streams across the multiple processors of a CRAY-3 system.

2.7.3 Batch Processing Features

The CSOS Network Queuing System (NQS) provides for the execution and control of batch work within the CRAY-3. NQS permits the creation of a batch environment and manages the workload and resources.

The batch environment can be composed of UNIX front-ends running RQS for job submission, UNIX peers running NASA/COSMIC NQS, and one or more CRAY-3's running NQS.

NQS allows the user to perform the following:

- Submit requests to a batch queue. The user can specify a set of qualifications for the batch request, including time, memory and CPU resource limits, exporting of environment variables, class name, and the queue to which the request is submitted.
- Submit a batch request interactively or submit a batch job from a front end computer batch job stream.
- Display the status of NQS queues. The **qstat** command displays the ordering of NQS requests and provides information about the requests in their respective queues.

[2] Available mid-1992.

- Delete or signal NQS requests.
- Display the status of NQS queues.
- Display supported batch limits and shell strategies for each host.

NQS allows the system administrator to perform the following:

- Establish queues that manage batch requests by their resource requirements.
- Establish priorities for queues and set a selection algorithm.
- Define a multiple system batch environment.
- Control the flow of requests through the queues.
- Initiate orderly shutdown of batch environment with checkpointing of executing jobs.

2.7.4 Job and Process Recovery Features

Two CSOS commands, **chkpnt** and **restart**, allow the user to checkpoint and restart a process or an entire job.

All checkpoint and recovery decisions are made at the user level. For an orderly system shutdown, the system sends out a SIGSHUTDN signal to all processes, indicating that a shutdown is imminent. The **chkpnt** and **restart** commands call the following:

- The **chkpnt** system call, which causes a user process or job to checkpoint its current operating environment to the named restart file. Processes with open pipes can be checkpointed and restarted if their pipe connections do not go outside the job or multitask group being checkpointed.
- A restart file contains sufficient information to restart the process or job described by the checkpoint image, provided that the required files are present.

- The **restart** system call, which causes recovery of the process or job described in the named restart file. The system verifies that the process or job can recover from the last checkpoint image.

Jobs that are run under NQS are automatically recovered over orderly scheduled shutdowns.

The **qchkpnt** command permits the batch job to perform a checkpoint, and resume processing. This capability allows protection from unscheduled system interruptions by creating a restart file that NQS can use to restart jobs during the next system start-up.

2.7.5 Interactive Environment

CSOS provides an extremely productive interactive environment based on use of the DoD TCP/IP protocol.

TCP/IP provides high-performance communications for interactive environments. The ease-of-use and flexibility of the CSOS command syntax provide a user-friendly, interactive environment. Productivity tools are available to efficiently use interactive file access, file manipulations, text editors, and on-line interactive debuggers.

The CSOS environment, in conjunction with the software network protocols, permits a very high productivity environment to be established for the user community. As an example, a highly compatible command environment exists between CSOS and workstations running software based on UNIX and interfacing to the CRAY-3.

This computational environment allows the user to determine where to execute a process. The highly interactive functions may take place locally on a workstation, while computationally-intensive parts may be distributed to the CRAY-3, and a high resolution graphics device may be used to display the results. This computational environment also supports the traditional batch, large-scale computational processes.

2.7.6 Real Time Capabilities

Real time features on CRAY-3 systems include least-time-to-go, real time process scheduling and very fast direct communication with real time external devices. The CSOS real time capabilities support fast response times for processes operating in real time mode. The specific response time that can be attained depends on the actions required by CSOS to service the requests, the real time external device hardware and software, and the media connecting that device with the CRAY-3 system.

Real time external devices that can be supported by these real time features include analog-to-digital conversion equipment, satellite telemetry or radar antenna data acquisition equipment, or other similar high-speed, low-latency devices. These devices can be connected to either the low-speed channels capable of 50 or 100 Mbits/second (6 or 12 Mbytes/second), or to the high-speed HIPPI channel capable of up to 100 Mbytes/second.

Foreground direct I/O, another real time support feature on the CRAY-3, is described in the foreground processor section of this document.

The following list specifies the CSOS features that are provided to support real time processes for the CRAY-3:

- Ability to schedule real time processes based on fixed, high real time priorities
- Ability to schedule real time processes using the Least-Time-To-Go (LTTG) scheduling algorithm
- Ability to preempt non real time processes
- Ability to exert scheduling control over a group of processes
- Ability to pre-allocate disk file space
- Ability to dedicate one or more CPU's to a real time process
- Ability to lock real time processes into physical memory

- Ability to maintain an interval timer with 1-millisecond granularity
- Ability to interface to real time capabilities through special system calls, while retaining access to all non real time CSOS system calls
- Ability to access real time capabilities from Fortran or C

2.7.7 CSOS Security Feature

CSOS provides a multiuser environment in which information and resources can be easily shared. In many cases, proper administration of physical security and adherence to standard system security measures ensure adequate protection for both user data and the system itself. However, at some sites and under certain circumstances, system integrity and the protection of sensitive information may require using the CSOS security feature.

The secure CSOS system facilitates concurrent processing of sensitive information at multiple security levels. No user may have access to sensitive information unless both discretionary and mandatory access control rules are followed. To provide a secure framework for a CRAY-3 running CSOS, the system exercises control over the flow of information between a CRAY-3 and remote network hosts; that is, the secure CSOS system provides protection for sensitive information and controls access to a CRAY-3 from networks, preserving security within CSOS system boundaries.

However, CSOS does not provide security enhancements to connected networks, nor does it guarantee a secure environment, which depends upon proper administration of physical security and proper application of system security features.

Design specifications for this feature were derived from the DoD Trusted Computer System Evaluation Criteria. These criteria describe the system software capabilities needed to satisfy government and private industry security requirements. The CSOS security feature offers the following:

- It supports concurrent processing of sensitive information at multiple security levels. This reduces the need to dedicate CRAY-3 to a specific sensitive process, thereby allowing efficient system use in environments where both sensitive and unclassified processing take place.

- It sustains system performance. The CSOS security feature requires minimal overhead, and it accommodates customer regulation of functions (such as security logging) that could affect overall system performance. Security log conditionals may be regulated by each installation, based upon system workload and required security audit coverage.

- It maintains compatibility with the architectural features and user interfaces of the UNIX operating system, as developed by AT&T Bell Laboratories. Because the CSOS security feature preserves familiar security mechanisms and user interfaces, it has a minimal impact on the usability of the system. Integration of the security feature is confined to a limited set of kernel functions; access controls for all processes are centralized and exercised in a manner consistent with the existing architecture.

2.7.7.1 CSOS Security Feature - Definitions

Subject. A subject is a validated user or process.

Object. An object can be a process, a regular file, a directory, a block or character special file, a fifo special file (named pipe), a message, or a socket. A socket is an interactive bidirectional communication, based on a client/server relationship and supported by the Transmission Control Protocol (TCP), between two or more programs running on different computer systems.

Discretionary Access Control. Discretionary access control is a set of rules that controls and limits access to an object, based on an identified individual's need to know and without intervention by a security officer for each individual assignment. This is accomplished by the use of standard CSOS mode permission bits (read (r), write (w), and execute (x)) and an access control list (ACL); the ACL allows the owner of a file to control r/w/x access to that file. The owner entersthe individual or group identifiers and the r/w/x privileges for the subjects that will be allowed to access the file into the ACL.

File owners usually establish the discretionary access rules for files they own; however, file access is always governed by the mandatory policy restrictions established by the security administrator.

Mandatory Access Control. Mandatory access control is a set of rules that controls access based directly on a comparison of the subject's mandatory access values (security level, compartments, and permission) and the object's

security level and compartments. The security administrator establishes a site's mandatory access rules and each user's access control by the assignment of these security levels, compartments, and user permissions in the validation file (/etc./udb). These mandatory access controls are the significant security features activated by the secure state of CSOS.

A maximum of 16 hierarchical classifications, called security levels, are used to represent a given subject's security clearance and the sensitivity of any object in the system.

At login, an individual user is uniquely identified and validated for access to CSOS as part of a successful login procedure. During the validation processing, the user is given minimum security level, maximum security level, active security level, authorized compartments, active compartments, and permissions.

The security level range is defined by minimum and maximum security levels; this range establishes a security window within which the user is allowed to operate. The active security level, automatically set to the user's default security level at login, constitutes the user's current clearance while executing under CSOS. In general, the user's active security level must be equal to the security level of any object that is accessed. The user can raise the active security level, but it cannot be set higher than the maximum level assigned by the security administrator.

The security window is bounded at each level by a set of compartments that restrict the access privileges to objects in specified categories. The user is assigned a set of active and authorized compartments during login; the active compartments define the current set while executing under CSOS. The authorized compartments are those compartments that a user can activate by command. The user's active compartments must be equal to or a superset of an object's compartments for access. The user can add compartment(s) to the set of active compartments, as long as the request is within the set of the authorized compartments assigned to the user.

At login, a set of permissions can also be granted to a user. The security administrator can use these permissions to define the site's security policy on a per-user basis. For example, permission may be given to read objects with a security level lower than that of the user (read down) and/or to append information to an object with a security level higher than that of the user (write up). Permission to read down or write up is always constrained by the user's minimum or maximum security level.

CSOS supports the principle of least privilege, which requires that each subject in the system be granted the most restrictive set of privileges commensurate with the performance of authorized tasks. The application of this principle limits the damage that can result from accident, error, or unauthorized use. In the secure CSOS system, restricted privileges can be administered through discretionary and mandatory access controls, by assignment of user permissions, and by the establishment of restricted environments in the form of restricted shells and restricted directories. Except for some system directories and files, all objects in the system inherit the active security level and/or active compartments of the creating subject.

Trusted Facility Management Capability. Some security compartments are reserved by CSOS to provide for trusted facility management. The trusted facility management capability allows a site to distribute security administration, system administration, and system operator responsibilities among separate users. The security administrator is responsible for assigning such compartments to specific utility programs and can define additional compartments as needed.

Security Log Feature. The secure CSOS system also ensures that all events and actions that have an impact on system security can be documented and traced to an individual user. To do so, CSOS provides a login validation mechanism by which each user is uniquely identified and validated before accessing the system. As each user performs security-related tasks, the system records the user and the task in the security log. CSOS commands allow the security administrator to access and produce reports of log entries.

Access Control List. Subject to the mandatory access controls described in the preceding paragraphs, standard CSOS access permissions and, optionally, the access control list (ACL) specify who may read, write, and/or execute the user's file. The ACL feature refines the discretionary access mechanism to allow the application of access permission on a per-user basis; that is, the ACL allows the user to restrict access of files to users who need such access to do their jobs or to perform specific tasks.

2.7.8 Accounting Features

CSOS provides the CRAY-3 user and the system administrator with accurate measurements of the user's resource usage. Both standard AT&T System V process accounting and the enhanced System Accounting, ConSolidated

Accounting, (CSA) are provided. Information available for user jobs includes interactive sessions and NQS jobs. CSOS accounting provides the following:

- Process accounting
  - elapsed time
  - CPU time
  - memory usage
  - I/O requests
  - I/O wait time - lock and unlock in memory
  - device usage and connect time
  - multitasking usage

- Job accounting
  - memory high water mark
  - summary of file system usage
  - accumulation of process statistics
  - NQS queue time

- Administration tools
  - flexible billing
  - flexible nonperiodic/periodic accounting report generation
  - accounting file management

Chapter 3 COMPILERS AND MULTITASKING FEATURES

The CCC software product set includes enhanced versions of the inherited CRI CFT77 and Standard C compilers. The concept of a modular compiling system with common optimization and code generation modules for both Fortran and C remains. However, the restructurer (optimization) module has been completely rewritten and a new intermediate text form has been introduced to provide a better basis for communication between the compiler modules.

The new restructurer incorporates improved vectorization capabilities which are already resulting in improved program execution times (as compared to the inherited CRI CFT77 compiler). Future enhancements will include parallelization and further vector, scalar, and whole program optimizations. A major goal of the restructurer is to be able to detect and exploit parallelism without user intervention. This will include inner and outer loop analysis, defining the scope of variables, benefit prediction, and conditional parallelization. Explicit parallelism will also be supported, allowing users to guide the compiler in code segments where the user can provide additional information or where the compiler cannot automatically detect parallelism.

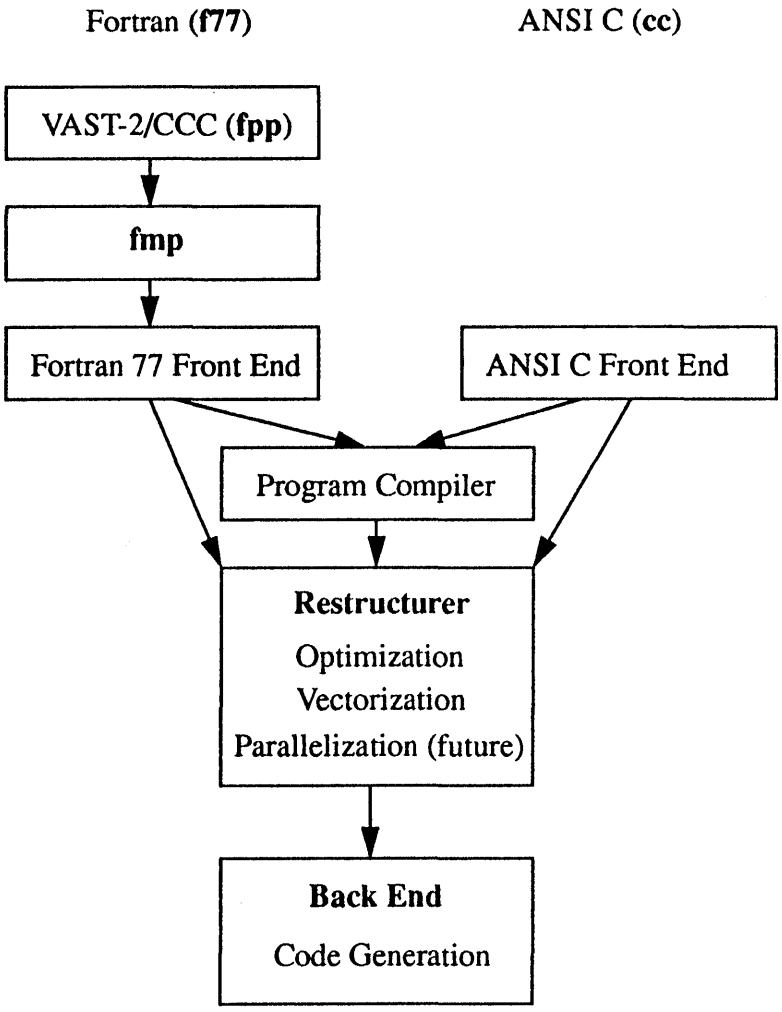


Figure 3 Cray Computer Corporation Compiler Development

Improvements to code generation for the CRAY-3 will be made by focusing and simplifying the back end; for example, by permitting instruction and register assignment interaction, improving peephole optimizations, and developing compiler techniques for increasing local memory usage.

CCC's compiler development may be seen as an evolutionary process with the gradual replacement of the compiler technology inherited from CRI with improved capabilities such as the new restructurer, within the framework provided by the new intermediate text representation (termed ir) internal to the compiler and its components. CCC currently supports both its new compiler, **f77**, and a CRAY-3 version of the inherited CRI CFT77 compiler, **cf77**. **f77** incorporates the new optimizer (restructurer) and provides some significant

execution performance improvements over the **cf77** compiler. **f77** retains the CFT77 compiler front end, thereby providing simple migration from CRI systems. The CFT77 code generator (back end) has been modified to exploit the CRAY-3 instruction set and is included in both **f77** and **cf77**. A new back end, with improved instruction scheduling capabilities, specifically targeted to the CRAY-3 is under development and is expected to be released during the second half of 1992.

The latest version of Pacific-Sierra Research Corporation's VAST-2/CCC (**fpp**) Fortran preprocessor and the **fmp** Fortran preprocessor which provide a superset of optimization and parallelization capabilities to those afforded by the Fortran compiler itself are supported, as are enhanced versions of appropriate Fortran, C, and I/O libraries.

The CRAY-3 Standard C compiler, in common with **f77** and **cf77**, utilizes the new restructurer and includes new capabilities for improved detection and recovery of compilation errors.

See the *CRAY-3 Fortran Reference Manual*, *CRAY-3 C Reference Manual*, and *VAST-2/CCC User Guide* for complete details.

3.1 CRAY-3 FORTRAN COMPILER, f77

The **f77** compiler is a high-performance compiler that is a full implementation of the ANSI X3.9-1978 standard and includes most ANSI X3.9-1966 features. **f77** supports a number of extensions to this standard to offer broader capabilities and to take advantage of the features of Cray Computer Corporation computer systems.

f77 is an integral part of CCC's compiling system. This compiler gives a Fortran program the ability to make use of all available processors through automatic parallelization of program components.

The design of **f77** encompasses traditional as well as more recently developed techniques in optimization and vectorization. **f77** compiler development continues to focus on methods which will deliver the highest performance and functionality from the CRAY-3 architecture.

Optimizations performed by the restructurer currently include the following:

- Elimination of common subexpressions
- Forward propagation of constants
- Extraction of invariant expressions from loops
- Movement of loads and stores out of loops
- Store elimination
- Dead code elimination
- Arithmetic simplification
- Short-circuiting of logical expressions
- Constant expression evaluation
- Strength reduction
- Scheduling of instructions to maximize functional unit parallelism
- Global register assignment

The vectorization of inner loops in Fortran programs allows them to take advantage of the great speed of vector operations. As a general rule, a vectorized loop executes 5 to 10 times faster than its scalar equivalent. No special syntax is required to specify vector operations in Fortran programs. Existing vectorizable code can immediately take advantage of vector speed. **f77** vectorizes loops containing nested IF statements, loops that use indirect (gather/scatter) addressing, and loops that search for particular conditions, as well as a variety of other types of loops.

f77 links vector computations, saving variables and subexpressions in intermediate registers rather than in memory. Doing so reduces the need for time-consuming memory accesses.

Diagnostic messages help end users create more efficient code by explaining what was vectorized in a program and what was not, and why. Often, simple code changes or compiler directives result in full vectorization of loops that previously appeared unvectorizable.

A wide range of compiler options are available. One feature in particular, flowtrace, provides the programmer requiring additional optimization with a valuable diagnostic tool. By enabling flowtrace, the programmer obtains a complete listing of CPU time spent in each subroutine as a percentage of total CPU time, the number of times each subroutine was called, and the names of programs calling each subprogram. Finer granularity information is available

with a new jumptrace library which utilizes the compiler's flowtrace code generation paradigm. Other compiler options enable listings of assembly code and cross-reference maps, and provide debugging aids.

3.1.1 ft77 Extensions

The Fortran compiler has many extensions to support other dialects of Fortran and to accept many nonstandard syntax structures that are common in programs written for other manufacturers' equipment. As an option, **f77** will flag statements that do not conform to the 1978 ANSI Fortran standard. The **f77** is an extended version of the 1978 ANSI Fortran standard. Some major **f77** extensions are identified below:

- Some Fortran 90 array processing, which permits operations on whole arrays with an abbreviated syntax
- User-controllable extent of symbol table generation for enhanced source-level symbolic debugging
- Recursive functions and subroutines
- Stack storage allocation
- Shifting and masking operations
- O (octal) and Z (hexadecimal) format descriptors
- Pointer data type
- Hollerith constants
- Boolean constants (octal or hexadecimal)
- Variable names of up to 31 characters and external and common block names containing up to 8 characters
- Type * (n) in TYPE and IMPLICIT statements
- Comments embedded within a line
- TASK COMMON storage for multitasking
- Asynchronous I/O (BUFFER IN/BUFFER OUT), which allows I/O operations to execute simultaneously with other program statements
- ENCODE and DECODE statements
- NAMELIST I/O
- Extra edit descriptors, including those for right justification and octal or hexadecimal output

3.1.2 Fortran I/O Capabilities

The following I/O capabilities are available to the Fortran user:

- Fortran formatted and unformatted I/O
- Fortran list-directed I/O
- Fortran direct access I/O
- BUFFER IN and BUFFER OUT I/O
- NAMELIST I/O
- Word-addressable random I/O (synchronous/asynchronous)
- Record-addressable random I/O (synchronous/asynchronous)
- Word-addressable, random access dataset I/O
- Dataset copy, skip, and positioning

See the *CRAY-3 Fortran Reference Manual* and the *CSOS 1.0 Library Reference Manual* for complete details.

3.2 CRAY-3 C COMPILER

The CRAY-3 ANSI Standard C compiler provides users of the CRAY-3 with a C compiler that conforms to the standards of the American National Standard Institute (ANSI) X3.159-1989, Programming Language C.

The ANSI Standard C compiler is also expected to conform to the International Standards Organization (ISO) standard for C language.

The ANSI Standard C compiler makes use of the same modular compiling system utilized by the **f77** compiler described above. The ANSI C front end generates the appropriate intermediate text form to take advantage of the language-independent optimizer and code generator. The C front end allows macro definition and substitution, conditional compilation, and the inclusion of named files in the compilation process.

Compiler options can be selected to do the following:

- Create an assembly listing of the code generated by the compiler

- Generate a flowtrace at execution time showing the record of CPU usage at a subroutine level
- Control vectorization or optimization
- Generate full symbol tables to enable source-level symbolic debugging

The C compiler implementation includes the following distinguishing features:

- Identifier name size up to 255 characters
- Support a standard calling sequence which allows access to Fortran and other languages
- Automatic vectorization of for, while, and do while loops
- Integer precision of 32-bits, 46-bits, or 64-bits

Extensions to the ANSI Standard C are identified below:

- 64-bit precision for multiply and divide operations
- Argument range check for some math functions

See the *CRAY-3 C Reference Manual* and the *CSOS 1.0 C Library Reference Manual* for complete details.

3.3 ASSEMBLY LANGUAGE, VERSION 2

Assembly Language, Version 2 (CAL), provides a powerful macro assembly language that allows a user to take advantage of all the CRAY-3 instructions. CAL allows the user to tailor programs to the architecture of the CRAY-3 computer and to write optimized code.

A set of symbolic machine instructions represents all the functions of the background processor. They translate one-for-one to binary machine instructions.

CAL also provides the user with a set of pseudo instructions which simplifies the task of creating assembly language programs. Pseudo instructions direct the assembler in interpreting source statements and generating object code.

CAL produces relocatable code usable as input to the loader. Subroutines written in CAL can be called from Fortran or C programs and vice versa.

The CAL assembler executes under control of the CRAY-3 operating system, CSOS. The **as** command assembles the source file, creating an output file for the loader. The **as** command-line options can specify characteristics of the assembler run such as the file containing source statements and list output.

3.4 MULTITASKING

Multitasking is a technique whereby an application program can be partitioned into independent tasks that can run in parallel on a CRAY-3 with multiple CPU's. Multitasking results in substantial throughput improvements over the performance of programs executed with one CPU. To achieve this, three methods can be used: macrotasking, microtasking, and autotasking. These three techniques can all exist in the same program.[1]

Multitasking improves productivity by reducing the execution time of a program due to the fact that a single program is executed by several CPU's, and the CPU idle time is minimized in the process. Multitasking is most effective when it is used on programs that have a high degree of parallelism.

3.4.1 Macrotasking

Macrotasking is an implementation of multitasking at a large granularity level and allows parallel execution of code at the subprogram level on multiple processors. Macrotasking is best suited to programs with very high levels of parallelism. The user interface to the system's macrotasking capability is a set of subroutines that explicitly defines and synchronizes tasks at the subprogram level.

1. A replacement library (**libpll**) for **libmt** and **libauto** is under development. This will integrate macrotasking and microtasking into a unified and optimal multitasking interface for the CRAY-3. This new library is planned for first customer availability mid-1992.

3.4.2 Microtasking

Microtasking is an implementation of multitasking at a finer granularity level and exploits parallelism at the DO-loop level in Fortran codes. Microtasking does not require code modification; rather, users insert directives in the form of Fortran comments that indicate where parallelism exists. The synchronization cost of microtasking is extremely low, which implies that small segments of code can be successfully partitioned over multiple CPU's.

When extra CPU cycles are available on the system, microtasking will use them effectively; when there are no idle cycles, the multitasked code will run at about the same speed as its nonparallel counterpart.

3.4.3 Autotasking

Autotasking is an implementation of microtasking which exploits parallelism at the DO-loop level in Fortran codes. Autotasking is designed to automatically partition parallel code sequences across multiple processors by using conditional vectorization, loop exchange, IF statement conversion, and automatic in-lining. Its flexible design allows users to insert processing directives.

Autotasking is based on the microtasking design, but is completely automatic and requires no programmer intervention. Autotasking provides the same performance characteristics and dynamic CPU usage as microtasking, while providing several additional features beyond microtasking. Autotasking introduces the concepts of parallel regions, parallel loops that carry values out from the last iteration, parallel loops that require private arrays, and the provision for the experienced programmers with specialized scientific skills to tune the performance of a parallelized program beyond that achieved by autotasking based on the dependence analysis data generated by the autotasking system.

Autotasking in the Fortran compiling system consists of three phases: dependence analysis, translation, and object code generation.

The dependence analysis phase looks for parallelism within program units, recognizing when iterations of Fortran DO loops operate on independent elements of arrays and inserting directives expressing the parallelism. The input to the dependence analysis phase is the Fortran source code and the

output is Fortran source code, possibly restructured, with autotasking directives added to express the parallelism. These autotasking directives are treated as comments by other vendor's Fortran compilers. This preserves the portability of the Fortran source code. The dependence analysis may be used to improve the performance of programs where little parallelism exists.

The translation phase uses the autotasking directives to restructure the code for parallel execution. The output is Fortran source code with calls to machine-dependent library routines and intrinsic statements embedded in the source code to control parallel execution.

The code generation phase uses the translation phase output to generate executable machine object code.

Autotasking on the CRAY-3 is facilitated through the use of the VAST-2/CCC Fortran preprocessor, developed by Pacific-Sierra Research Corporation and offered as part of the CCC software product set.

See the *CSOS 1.0 File Formats and Special Files Reference Manual* for further details.

3.5 COMPILER PREPROCESSORS

Two compiler preprocessors are available to enhance the optimization capabilities of the CCC Fortran compiler. Both can be automatically invoked as part of the compiler or as a separate command, and translate Fortran source code into optimized and/or parallelized Fortran source.

3.5.1 fpp/vast - Pacific-Sierra Research Corporation's VAST-2/CCC

fpp (or **vast**) analyzes Fortran code and inserts optimization and, optionally, parallelization directives. These commands are the Pacific-Sierra Research Corporation's VAST-2 product specifically designed for CCC software and the CRAY-3.

VAST-2/CCC improves the performance of Fortran codes in four major ways:

- Enhanced vectorization

- Recognition and generation of parallel constructs (called autotasking or concurrentization)
- Automatic in-line expansion
- Special code sequence recognition

VAST-2/CCC recognizes vectorization opportunities and then utilizes various techniques to produce code that can be vectorized. These techniques include statement reordering, ambiguous subscript resolution, reference reordering, splitting calls out of loops, loop nest restructuring, and loop exchanges.

VAST-2/CCC also recognizes parallelization opportunities and inserts directives which, when processed by **fmp**, exploit the inherent parallelism in a user's code. In general, concurrentization is performed on the outermost possible loop.

See the *VAST-2/CCC User Guide* for complete details.

3.5.2 fmp - Fortran Microtasking Preprocessor

fmp is the microtasking preprocessor that interprets the microtasking directives, and rewrites the program to enable it to microtask. Microtasking improves the wall-clock execution time of a program by employing extra processors during some of the computationally intensive parts of the program. **fmp** is automatically invoked, under user control, by the CRAY-3 Fortran compiler and thus provides autotasking capabilities.

Chapter 4 UTILITIES, TOOLS AND LIBRARIES

CSOS supports a comprehensive set of utilities and tools to provide a powerful and flexible programming environment for both application programmers and system programmers. CSOS supports the following more notable utilities:

- Editors: **ed, emacs, vi**
- CRAY-3 loader, **ld** and **segldr**
- Debuggers, **bdb, debug, symdump**()
- System activity report, **sar**
- Performance analyzers, flowtrace, jumptrace, **memtrace, prof, profview, procstat, trace, vpm, vsm**
- Source control maintenance utilities, **sm** and **nupdate**, and Source Code Control System, SCCS
- Global cross-reference facility, **ftref**
- Multitasking history trace processor, **mtdump**
- Support tools
- Simulators - **sim** and **vm**

In addition, CSOS supports a wide range of libraries including the standard Fortran, C and performance I/O libraries, X11R4, OSF/Motif and OPEN LOOK graphical user interfaces, and a completely new set of debugger support and graphical applications development libraries, the latter collectively referred to as Holmes.

4.1 EDITORS

The common UNIX editors (e.g,. **ed**, **vi**) are available with CSOS. In addition, the GNU Project Emacs editor (**emacs**) is a fully supported product. Further editing capability is afforded by the X Window System, OSF/Motif and OPEN LOOK text widget intrinsics.

4.2 CRAY-3 LOADER

The CRAY-3 loader is an automatic loader for code produced by the Fortran, C and CAL compilers. It is available to CSOS users via a traditional UNIX command, **ld**, or via the **segldr** command. Executing under the control of CSOS on a CRAY-3, **ld** and **segldr** are efficient and full featured loaders which produce both segmented and nonsegmented executable programs.

Segmented programs are those having overlaid code modules and nonsegmented programs are those not having overlaid code modules. Program segments are loaded as required without explicit calls to an overlay manager. The CRAY-3 loader helps users produce and execute segmented programs without extensively modifying the code and automatically produces run-time checks to determine whether subroutine calls require loading new segments into memory. A system-provided, memory-resident routine loaded with the object module manages memory overlays.

Full details regarding the **ld** and **segldr** commands may be found in the *CRAY-3 Loader Reference Manual.*

4.3 DEBUGGING

The following debuggers are a standard part of the CRAY-3 software:

- **bdb**, symbolic source-level interactive debugger
- **debug**, a symbolic dump utility
- **symdump**(), a run-time symbolic dump library utility

4.3.1 bdb

CCC has developed a new source-level, symbolic interactive debugger, **bdb**, designed to offer markedly superior functionality to products previously available in the supercomputing environment and provide portability between the CSOS and future SVR4 environments.

Some of the key features supported in the current release of **bdb** include:

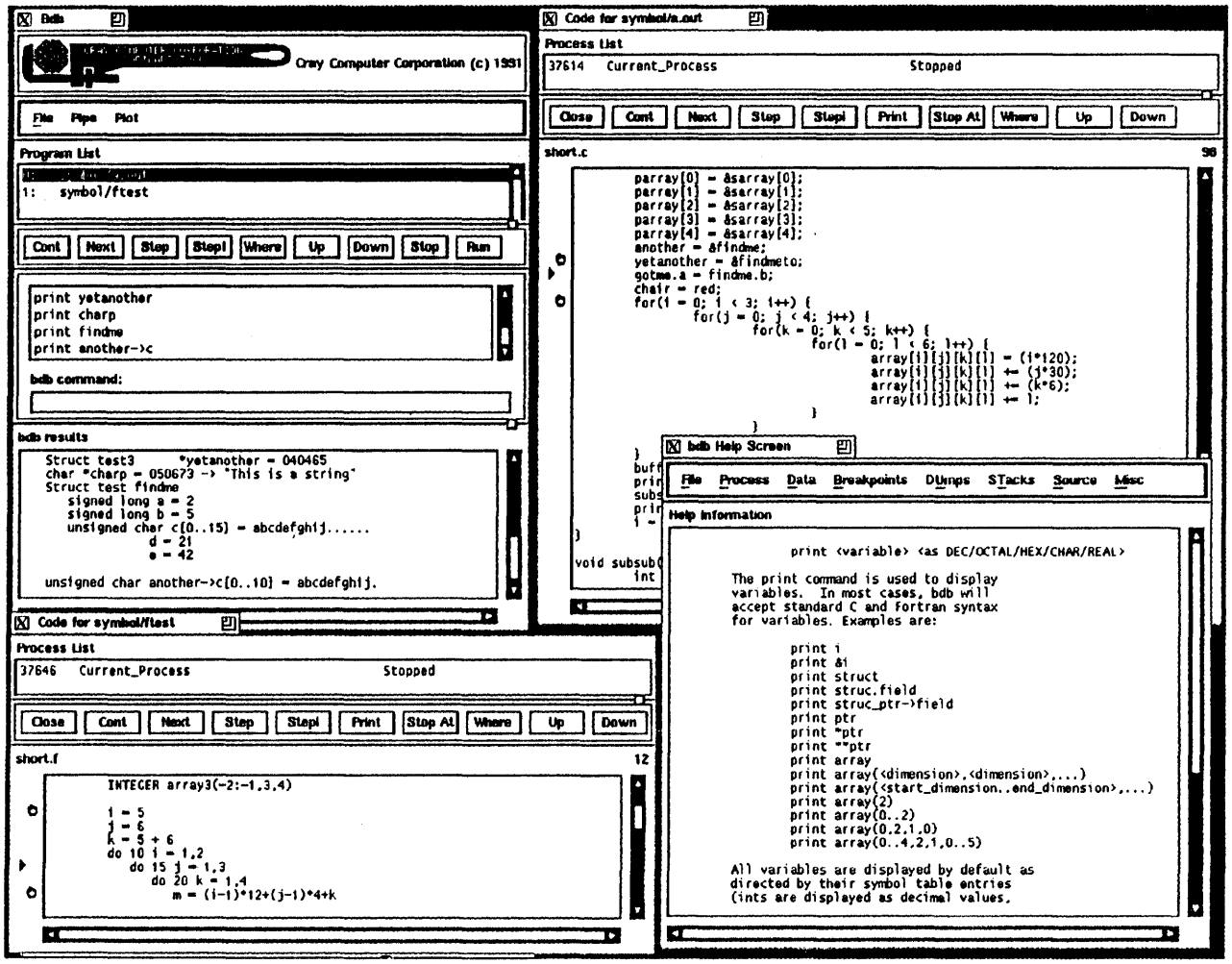
- Library-based design: allowing functions normally available solely to a debugger to be available to user applications.
- Full capabilities for printing and setting process variables (structures, structure elements, arrays, array sections) symbolically.
- Full capabilities for printing and setting hardware registers.
- Support for both Fortran and C.
- The ability to start processes, attach to and detach from existing processes, examine core files, alter process state, and run processes unimpeded by the debugger.
- An advanced user interface, utilizing the Holmes system libraries, offering the user the options of line, X11R4 Athena widget, OSF/Motif or OPEN LOOK[1] mode.
- The ability to control and debug multiple processes simultaneously.
- The ability to control and debug multitasked processes.

Key features to be added to the debugger include:

- The ability to run the debugger in a distributed mode: with the user interface executing on a workstation and the process control interface executing on the CRAY-3.
- Graphical symbolic data displays.
- Checkpoint/restart of a debugging session
- Native language expression evaluation for Fortran and C.

1. Available early 1992.

Figure 4 Sample bdb Session with the Holmes Libraries Graphical Interface



Shown in the above figure is the OSF/Motif widget version of **bdb**, which utilizes the Holmes libraries graphical user interface. Three sample screens are presented: the main screen (upper left) from which the user controls all procesesses, two code windows (upper right and lower left) from which those individual processes can be controlled, and the on-line help facility (lower right).

Full details of the new CCC debugger may be found in the *bdb User Guide and Internal Reference Manual.*

4.3.2 dasm - CRAY-3 disassembler

For those users versant in CRAY-3 assembly language, a disassembler is available for inspecting an executable binary. **dasm** produces a CAL-like listing, along with symbolic information (if it is available in the binary) for both CRAY-3 background processor and foreground processor binaries.

4.3.3 debug

debug provides a batch-oriented mechanism of obtaining symbolic interpretation of a user core file. The **debug** command produces a full, symbolic dump of the contents of all active variables in the application at the time the corefile was obtained. This listing can then be viewed off-line. Options on the command line provide limited options for controlling the amount of information output.

4.3.4 symdump()

symdump() is a standard library subroutine that can be called from within an application to obtain output similar to that of **debug** during execution of the application. This provides a convenient alternative to inserting print statements for variables being examined during debugging.

4.4 PROCESS PERFORMANCE TOOLS

This section provides a description of a subset of the performance tools which are included with the system software.

4.4.1 Flowtrace - Tracing Procedure Calls

Flowtrace generates printed information about all procedure calls in an executed program. Flowtrace is implemented with a compiler option (**-h flow** or **-F**) and the flowtrace library (**libflow**) and postprocessor (**flow**). Support for flowtracing multitasked applications is provided with a new flowtrace library

(**libnflow**) and postprocessor (**nflow**)[2]. Flowtrace summary information is written to file **stdout** and contains the following:

- The time spent in each routine: amount, percentage of the total execution time, and average time per call
- Number of calls to each procedure and total calls in the program
- Lists of routines that call and are called by each routine
- A calling tree of the main program and all subprogram

4.4.2 Jumptrace - A Finer Granularity Flowtrace

Jumptrace uses the compiler-generated flowtrace paradigm, but is implemented by replacing the flowtrace library with the jumptrace library (**libjtrc**) when producing an executable program binary. The jumptrace library contains routines which monitor performance at the loop level, rather than at the subroutine level and a summary report is automatically generated at program termination.

4.4.3 memtrace, memtrck

memtrace allows a user to trace memory allocation calls in a process. This is particularly useful in applications which perform memory management functions. **memtrck** provides a postprocessing capability for the **memtrace** log and partially automates the process of validating memory allocation and deallocation activity.

4.4.4 prof and profview - Non-Invasive Execution Profiler

The **prof** profiling system available on CSOS allows a user to determine areas of high execution activity of the program being monitored. Since the statistics are gathered by the CSOS operating system, this tool is non-invasive and has little impact on the user's program. Areas of high execution activity will be made visible by **prof**, showing how often the particular instruction program area was executed by percentage, hit count, and estimated time. The **prof**

[2] Aavailable December 1991.

utility program indicates which routine contained the high activity, showing the localized area down to the level of an individual Fortran line number. By studying the data provided by **prof**, the user can determine which routines need further optimization efforts, and which routines can be safely de-emphasized from the optimization efforts.

The interactive utility **profview** makes it easier for the user to review and study the statistics generated by the **prof** command.

4.4.5 procstat and procrpt - Input/Output and Memory Usage Statistics

The **procstat** utility gathers statistics on the input/output activity of the program being monitored, with only a minimum of overhead time incurred by procstat itself. The statistics gathered include the number of characters read and written, the number of I/O calls made, and the amount of wait time associated with the I/O requests. These statistics are gathered and reported for each individual file name.

In addition, **procstat** statistics show the amount of memory activity for the program being monitored including the amount of memory requested, and the number of times the system was called to service a memory size change request.

The post-processor utility program, **procrpt**, organizes the statistics generated by **procstat** into a more readable report.

4.4.6 trace - Trace System Calls

trace makes use of the **/proc** interface to intercept the system calls of a process, allowing the user to monitor system call activity for that process. Details about each system call are displayed as they occur or in summary form depending upon command line options. **trace** can be run on a command or on a pre-existing process.

4.4.7 vpm - Visual Process Monitor

The Visual Process Monitor[3] (**vpm**) is a Holmes system based tool which can be used to monitor process performance characteristics as a process executes. Both textual and graphical information is presented to the user regarding memory, CPU, and I/O usage and to trace a user's process symbolically.

4.5 SYSTEM PERFORMANCE TOOLS

Overall system performance is an important factor in a supercomputing environment. Two products provided by CCC aid in measuring and analyzing system performance:

4.5.1 sar - System Activity Report

The system activity report utility, **sar**, is supported under CSOS. This utility displays performance and resource usage statistics for the running system (for example, CPU use, number of system calls made, and reads and writes to each configured device in the system). The information is valuable for monitoring and analyzing system performance.

4.5.2 vsm - Visual System Monitor

The Visual System Monitor (**vsm**) is a Holmes system based tool which allows a user to monitor total system performance characteristics and system-level process information. Both textual and graphical data displays provide **sar**-like data, updated at user-specified intervals. A graphical display allows the user to monitor physical CPU utilization in three categories: system, user, and idle.

3. Available mid-1992.

4.6 SOURCE CONTROL MAINTENANCE UTILITIES

4.6.1 CSOS Source Manager (sm)

The CSOS source manager (**sm**) is a program that provides a mechanism for version control of applications source code in a UNIX environment. The purpose of **sm** is to keep track of the updates and the corrections made to files in the form of modification files. The **sm** structure is based on a new update format (**nupdate**) program library (PL). This allows **sm** to support programs or systems based on COS, CSOS, UNICOS and CTSS. **sm** provides the following capabilities:

- Storing text files
- Retrieving particular versions of files
- Controlling updating privileges to files
- Recording each change as it is applied

To accomplish these tasks, **sm** maintains each file along with all inactive lines. To retrieve the current version, all active lines are output. To retrieve previous versions, these inactive lines are reactivated.

4.6.2 CSOS Source Control Manager (nupdate)

nupdate is a line-oriented text editor for maintaining programs in the form of source code, as well as other types of text data. **nupdate** creates and modifies program libraries (PL's) and produces output that can be used as input to other programs, particularly compilers and assemblers. **nupdate** can create a new PL or modify an existing PL. **nupdate** is provides compatibility with program libraries generated on COS, UNICOS, and CTSS systems with UPDATE.

For a creation run, input must include source decks, and can include a source dataset from an earlier update run, modification sets, and input directives. Output from a creation run can include a new PL, listings, a compile dataset, and a source dataset.

For a modification run, input must include a PL and can include new decks, modification sets, and input directives. Output from a modification run can be a selected listing, a compile dataset, source decks, and a new PL.

4.6.3 Source Code Control System (SCCS)

The source code control system (SCCS) is a collection of programs ported from the AT&T UNIX system that run under CSOS. The purpose of SCCS is to track modifications to files, which is useful when programs and documentation undergo frequent changes because of development, maintenance, or enhancement. It provides the following capabilities:

- Storing text files
- Retrieving particular versions of files
- Controlling updating privileges to files
- Identifying the version of a retrieved file
- Recording when, where, and why a change was made and who made each change to a file

As its name implies, SCCS performs these tasks on source code (high-level or assembly language) or text but not on binary executable files.

To accomplish the previously described tasks, SCCS stores the original file, all changes to the file, comments describing the changes, and control information in a single control file. As users make revised versions of the file, SCCS automatically updates the control file.

4.7 GLOBAL CROSS-REFERENCE FACILITY

Two utilities are supplied which produce program cross-reference listings:

4.7.1 ftref - Fortran Global Cross-Reference

ftref is a tool that generates a listing containing several forms of information about a Fortran application. **ftref** reports on the common block variables used in the subroutines within an application. **ftref** provides tabular information that consists of entry names, calling routines, and called routines for each subroutine; it displays this information as a static calling tree. For multitasked applications, **ftref** summarizes the use of multitasking subroutines and reports

whether a common variable or a subroutine is locked when it is referenced or redefined. It also shows information about use of the multitasking barrier routines.

4.7.2 cxref - C Program Cross-Reference

cxref analyzes a collection of C files and attempts to build a cross-reference table and produces a listing of all symbols (auto, static, and global) in each file.

4.8 MULTITASKING TRACE ANALYSIS

4.8.1 Multitasking History Trace Processor (mtdump)

mtdump processes multitasking history trace buffer entries, produced by **libmt**, which are generated by the multitasking routines. It also handles trace information from the multitasking barrier routines.

4.8.2 Microtasking Trace Buffer

The currently supported microtasking library (**libauto**) contains user-controllable instrumentation for tracing events of interest during the execution of a micro- or autotasked application. The MICRO_TRACE and MICRO_STATS environment variables may be set by the user to control the size of the trace buffer and enable/disable tracing, respectively.[4]

[4] A replacement library (**libpll**) for **libmt** and **libauto** is under development. This will integrate macrotasking and microtasking into a unified and optimal multitasking interface for the CRAY-3. Graphical trace analysis utilities are planned to provide both monitoring and post-analysis capabilities. These new products are planned for first customer availability mid-1992.

4.9 SUPPORT TOOLS

CSOS includes a large number of support tools to assist programmers and end users. Descriptions of a few of the more notable support tools are provided in this chapter.

4.9.1 awk

The **awk** utility provides a means of scanning and processing text patterns within a file utilizing a "programming language" concept, and generally provides more powerful and extendable capabilities than those afforded by **grep**.

4.9.2 diff

The **diff** utility displays the differences between two files on a line-by-line basis. It displays the differences as instructions that users can use to edit one of the files to make it the same as the other.

4.9.3 grep family

The **grep** (**egrep**, **fgrep**) utilities search one or more files, line by line, for a pattern. The pattern can be a simple string or another type of a regular expression. The **grep** utility takes various actions, specified by options, each time it finds a line that contains a match for the pattern indicated. Users may specify on the command line the files that the **grep** utility will use for input.

4.9.4 mail

The **mail** program allows the electronic exchange of messages between users on CSOS and connected systems or across a nationwide network.

4.9.5 make

The **make** program provides a method for maintaining up-to-date versions of programs that result from many operations on a number of files. The **make** program can keep track of both the sequence of commands that create certain files and the list of files that require other files to be current before the operations can be done. A useful feature of the CSOS **make** utility is its inherent ability to multiprocess; this is controlled by the user's NPROC environment variable.

4.9.6 perl

perl is an interpreted language for scanning arbitrary text files, extracting information, and printing reports based on that information. It combines some of the most useful features of the C language, and the **sed**, **awk** and **sh** utilities without the arbitrary limits on the size of files imposed by many other UNIX text scanning utilities.

4.9.7 tail/head

The **tail** utility displays the last part of a file. It takes its input from the file users specify on the command line or from its standard input. In default, the **tail** utility displays the last ten lines of a file. Users may specify any number of lines in a file to be displayed and may indicate which lines, blocks, or characters to display. The **head** utility provides the same function as **tail**, except that it displays the first portions of a file.

4.10 SIMULATORS - sim and vm

Enhanced versions of the CRAY-3 / CRAY-2 instruction simulator (**sim3/sim**) and virtual machine environment (**vm**) are made available for the first time to customers. **sim3** and **sim** provide non-native simulation; i.e., simulation of CRAY-3 binaries on a CRAY-2 system; **vm** provides native instruction simulation. These products provide background and foreground processor and disk simulation. Enhancements include support for simulated file systems, copying between simulated and real file systems, debugging, tracing, and symbolic capabilities as well as SVR4 support.

See the *Instruction Simulator Reference Manual* for complete details.

4.11 CRAY-3 SOFTWARE LIBRARIES

The CRAY-3 software includes subprograms that are callable from CAL, Fortran, and C. The subprograms have been divided among the libraries generally on a functional basis. The major libraries and their functions are listed below:

- asdef - Assembler definitions library
- csu - Common start-up library
- libauto - Microtasking/autotasking support library
- libc - Standard UNIX system call and C support library
- libdb - **symdump**() library*
- libf - Fortran library
- libflow - Flowtrace support library
- libjtrc - Jumptrace support library
- libio - Fortran and high-performance I/O interface library
- libm - Math library
- libmalloc - Special memory manager library with limit checking
- libmt - Multitasking library
- libpll - CRAY-3 parallel processing library[5]
- libsci - Scientific library
- libu - Fortran interfaces to CSOS system and C libraries
- lmset - Local memory definition library
- Traceback libraries
  - libnotr - Dummy entries to disable traceback
  - libtr - Routines to perform traceback
  - libvtr - Traceback with vector register dump
- Special-purpose libraries
- libcurses - Screen-oriented terminal I/O library
- libenv - Environment-check-enable library

5. Available mid-1992.

- libl - Lex library
  - libnet - TCP/IP network library
  - libprof - Profile-enable library
  - librpc - Remote procedures call library

Both libraries have also been enhanced specifically for the CRAY-3 architecture. The **libsci** changes include the addition of LAPACK routines, higher performance versions of the FFT algorithms and a Sparse Matrix Solver. Both **libsci** and **libm** have been enhanced to support the improved floating point accuracy of the CRAY-3 system.

In addition to the above listed libraries, there are a set of new libraries supplied by Cray Computer Corporation which fall into two functional categories: debugger support and graphical applications support libraries. More information is provided in the following sections.

4.12 CRAY-3 DEBUGGER SUPPORT LIBRARIES

When Cray Computer Corporation embarked on the design and implementation of a new source-level debugger for supercomputer applications, it made a decision to make components of the debugger reusable and thus chose to implement those components in the form of library entry points. The result is the availability of the following libraries to the general user:

4.12.1 libbdb

A process control library which supplies the ability, via the CSOS **/proc** interface, to attach to and control, or inspect existing processes. All **bdb** debugger process control functions are supported via this library.

4.12.2 libdis

A library containing all data formatting and symbolic display routines utilized by the debugger.

4.12.3 libsym

A library containing interface routines to the symbol tables generated by Cray Computer Corporation compilers which allow the extraction of symbolic information from an executable binary.

4.12.4 libtool

A set of miscellaneous library routines utilized by Cray Computer Corporation software tools, such as a CRAY-3 / CRAY-2 background and foreground disassembler, simple string parsing, and manipulation functions.

See the *CSOS 1.0 Debugging Tools Libraries Reference Manual.*

4.13 CRAY-3 GRAPHICAL APPLICATION SUPPORT LIBRARIES

Cray Computer Corporation provides a suite of graphical user interface libraries, including those accepted as industry standards. The Holmes System libraries have been wholly developed by Cray Computer for the purpose of supplying a consistent user interface regardless of the look-and-feel implemented by the lower-level Graphical User Interface (GUI) libraries. Several new software tools supplied with CSOS (e.g., **bdb**, **va**, **vsm**, **vpm**) are built upon the Holmes System libraries, and provide a consistent interface regardless of whether the user's preference is X/Athena widgets, OSF/Motif, or OPEN LOOK.

4.13.1 MIT X Window System

Cray Computer Corporation presently supports the X11R4[6] system from MIT. The X Window System allows the CRAY-3 to write graphics and text output directly to any bitmap display terminal that runs this software. The CSOS implementation of the MIT X Window System follows the client-server model: the client is a highly transportable package that can be moved from host to host, and the server provides workstation-dependent support. The client portion of this model resides on a CRAY-3 running CSOS.

6. X11R5 will be supported in mid-1992.

4.13.2 OSF/Motif

Cray Computer Corporation has ported the OSF/Motif 1.1 toolkit to the CRAY-3 and CSOS and supports this industry-standard graphical user interface in client applications.

4.13.3 AT&T OPEN LOOK

Cray Computer Corporation has ported the OPEN LOOK toolkit to the CRAY-3 and CSOS and supports this "industry-standard" graphical user interface in client applications.

4.13.4 Holmes System Libraries

The Holmes libraries were developed to fill a specific need for a complete and easy-to-use distributed application development environment for supercomputers, as well as to address the pragmatic issue of providing a reusable set of graphical user interface components rather than designing application-specific components. Recent advances in workstation technology have prompted the need for distributing applications across differing platforms, allowing the supercomputer to work on problems best suited for the platform, while allowing graphic workstations to take over the load of a graphical front end. It should be easy for a user to perform this distribution of power, and the Holmes libraries are our solution. All new Cray Computer Corporation software tools are being developed utilizing the Holmes libraries and their functionality as a foundation.

Specifically, Holmes was built using the following set of goals:

- Users should be able to develop applications which utilize graphical user interfaces without becoming a window system expert or a communications expert.
- Windowed applications impose an overhead on a supercomputer system that should be eliminated if possible.
- Transition between window systems (e.g., X11R4/Athena, OSF/Motif, OPEN LOOK), or variations of a single window system, should not require source code to be changed.

- A front end interpreter should be available, both to speed the development process and to ease user customization.
- Data I/O, between processes or devices, should never result in an application blocking. Blocking is unacceptable in windowed applications as well as service providers.
- A standard interface, one that aids in the development of modular code, makes it easier to split programs into sections that may be distributed.

The Holmes System is comprised of a set of libraries which implement individual aspects of the computational environment:

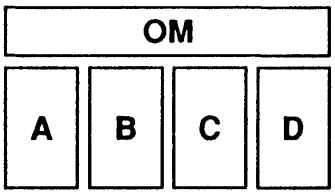
- **libtcl** - The Tool Command Language library as developed by Professor John Ousterhout at the University of California, Berkeley.
- **libtclp** - Cray Computer Corporation extensions to the Tool Command Language Library.

- **libom** - The Object Manager library which supports a simple, object oriented programming interface for application development.
- **libwigins** - The Wigins library which provides consistent callback and periodic event processing.
- **libdoyle** - The Doyle library which supports distributed applications and hides the various communication idiosyncrasies from the application programmer.
- **libwatson** - The Watson library which, along with the **libwaw, libwmw** and **libwow** support libraries (which are window system dependent), provides a complete graphical user interface to application programs.
- **libhudson** - The Hudson library which supplies a number of "canonical" widget packages for supporting simple data plotting and visualization (e.g,. X-Y plots, pie charts, etc.).

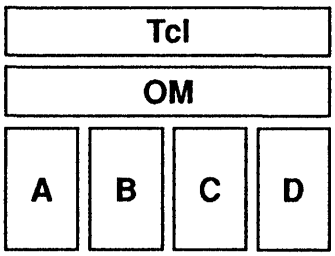
The following figure depicts the interfunctionality of the Holmes libraries and the stages a program evolves through during the normal development cycle of a Holmes based application. The eventual result of this design and development cycle is an application which is easily distributed across heterogenous platforms.

For a complete description of the Holmes Application Development Environment see the *Holmes Reference Manual.*

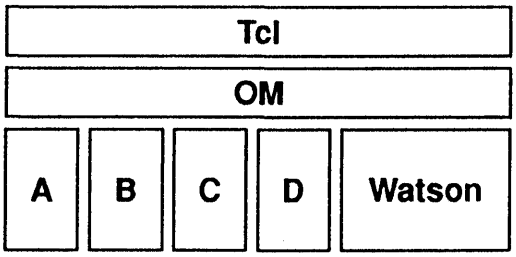
This monolithic application contains all sections of a normal application.



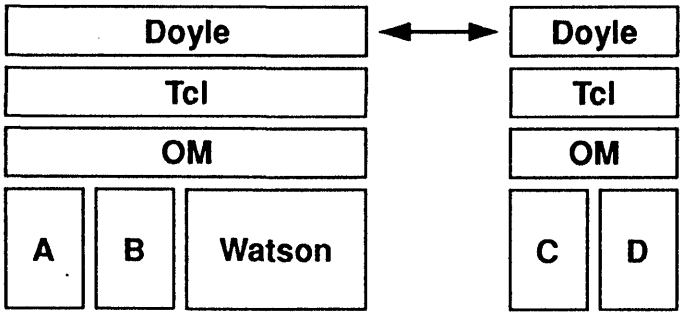
The monolithic application has been separated into logical classes and provided with a standard interface through ***libom****. This application behaves in the same way as the monolithic application described above.*



A Tcl interface has been added to the application allowing easy prototyping of new capabilities and functions. The Tcl interface also provides a standard command line interface to the application that allows quick customization of the application without changing the source code.



A graphical user interface has been added to the application by including Watson in the load module. With few exceptions, the control of the user interface may be completely contained in the Tcl layer as Tcl code.



Simply by changing the Tcl, this application is separated into two separate processes which may then be distributed on separate processors or systems.

Chapter 5 NETWORKING AND COMMUNICATIONS

A primary advantage of CSOS is its ability to fit effectively into existing environments as part of a computer network. CSOS supports communications and connectivity via the DoD transmission control protocol/internet protocol (TCP/IP) suite.

Cray Computer Corporation communication software products provide a variety of ways to fit a CRAY-3 into the existing communications environment. CSOS supports the following products.

- Transmission Control Protocol/Internet Protocol (TCP/IP)
- CSOS Network File System facility (CSOS NFS)

These products are briefly described in the following subsections.

5.1 TCP/IP PROTOCOL SUITE

TCP/IP is a set of computer networking protocols that allows two or more computer systems, called hosts, to communicate over a network. TCP/IP was originally defined by the Defense Advanced Research Projects Administration (DARPA), an agency of the DoD. It is now a government standard, and many

vendors and third parties support TCP/IP on a variety of computing equipment. A license is required for TCP/IP.

This implementation consists of the entire set of lower layer protocols, which include the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Iinternet Control Message Protocol (ICMP); the DoD defined user applications file transfer protocol (**ftp**), simple mail transfer protocol (**smtp**), and virtual terminal access (**telnet**); Sun's Network File System (NFS), MIT's X Window System for graphic applications, and other utilities. In addition, Cray Computer Corporation supports the 4.3BSD extensions that include socket user interface, remote copy (**rcp**), and remote shell (**remsh**).

The TCP/IP communication protocol suite, as defined by the DoD and enhanced by the University of California at Berkeley, has been implemented on the CSOS system. This allows a CRAY-3 to act as peer and communicate with other vendor systems on a TCP/IP supported network. These include most mainframe vendors and virtually all workstations.

De facto standard networking applications such as Sun's NFS and MIT's X Window System packages use TCP/IP as the transmission protocol. Such applications on top of TCP/IP provide a seamless interface from the workstation to and from the CRAY-3. TCP/IP also provides a platform upon which the users may build distributed applications. NFS provides transparent access to data distributed over a network. Data resident on one system can be accessed transparently from other systems supported within the network.

TCP/IP allows users to send both ASCII text files and binary files across the network; differences in file formats are automatically resolved during the transfer. Within TCP/IP, a set of procedures allows communication among all hosts on the network whether they are similar or dissimilar systems. TCP/IP packages on non CRAY-3's are available from other vendors or third party suppliers. Implementations currently exist on most systems based on UNIX and many systems not based on UNIX. TCP/IP allows communication between physically diverse networks.

TCP/IP provides a number of commands that assist the user in communicating not only with CRAY-3's, but with any systems that support the TCP/IP protocol.

5.2 CSOS NETWORK FILE SYSTEM (NFS)

The CSOS Network File System (CSOS NFS) is a software product that allows users to share directories and files across a network of machines. CSOS NFS users can use standard CSOS I/O calls and commands with standard permission controls to access files from any file system. Similarly, other NFS users can make use of CSOS file systems from anywhere in the local network environment. The user interface to CSOS NFS is transparent.

CSOS NFS uses a server/client system to provide access to files on the network. A server is any machine that allows a portion of its local disk space to be exported (made available for remote mounting). A client is any machine that makes a request for an exported file system. When a CSOS user issues an I/O call (such as **read, write, open, close, create,** or **delete**) for a file that resides on a file system mounted by CSOS NFS, the call is transmitted to the server machine. When the server receives the request, it performs the indicated operation. In the case of **read** or **write** requests, the indicated data is returned to the client or written to disk, respectively.

CSOS NFS also includes a set of distributed processing tools developed by Sun Microsystems. It includes **nfs, rpc, xdr, yellow pages, lock manager,** and others. These tools will continue to be supported as part of the NFS package. The Remote Procedure Call (RPC) routines allow users to create distributed applications that can call procedures residing in any host on the network. These routines extend procedure-call semantics into the network environment and user External Data Representation (XDR) routines to ensure that data sent across the network is in a machine-independent format.

CSOS NFS administrative tools are provided as well as User Identification (UID) mapping, which allows users located in different administrative domains on a large network to all concurrently access a single CRAY-3, even though a user may have different user ID's between his local system and the CRAY-3.

Chapter 6 APPLICATIONS

Cray Computer Corporation supports vendors of application software in the conversion and optimization of their softwarepackages for the CRAY-3 CSOS environment. The *Directory of Applications Software*[1] provides descriptions of the packages and information on the vendors. Available packages cover a wide range of disciplines including the following categories:

- Structural and mechanical engineering - analysis
- Structural and mechanical engineering - design
- Electronics and electrical engineering
- Nuclear engineering and energy
- Computational fluid dynamics
- Petroleum - reservoir simulation
- Petroleum - seismology
- Chemistry
- Graphics and iImaging
- Mathematics, econometrics, and statistics
- Simulation and mathematical programming
- Languages, data management, and other tools

1. Available December 1991.

The availability of applications for the CSOS environment is driven largely by customer requirements communicated to the vendors of the applications. The Cray Computer Applications and Benchmark Group encourages applications developers to migrate their applications to the CRAY-3 CSOS environment and supports them in optimizing the packages. We also encourage development of new applications and algorithms to take advantage of the vast memory and multiple processors available on the CRAY-3.

Cray Computer does not develop or provide these packages directly. They are distributed by the respective vendors or developers as noted in the *Directory of Applications Software*. The availability of third party packages is expected to expand rapidly as the population of installed CRAY-3 systems increases.

Chapter 7 FOREGROUND PROCESSOR SOFTWARE

The foreground processor is responsible for monitoring and supervising system operations. The foreground processor scans each of the four hardware channels, searching for a channel request. When a request is on the channel, the foreground processor initiates the required activity and then starts scanning the channels again.

The firmware that runs in the foreground processor is organized into the following parts.

- Monitor
- Background processor interface
- Macros
- Foreground processor real time capabilities
- Foreground configurator

In addition, the system console, based on a Sun compatible, UNIX workstation, is connected to a special channel on the foreground processor which also facilitates deadstart and dead-dump of the system. This console workstation includes a number of capabilities not previously provided in CRAY-2 console environments.

7.1 MONITOR

The CRAY-3 foreground processor monitor processes requests from the background processors, controls the operation of all external I/O devices such as disk storage units and CRAY-3 low-speed channel connections, and provides an interface to the maintenance and operator workstation.

The foreground processor monitor consists of code to govern four channel scans, clock and console handlers, channel time-out routines, and deadstart initialization. The foreground processor monitor code uses macro statements to configure the clock and console, each background processor, disk storage facilities, 6 and 12-Mbyte bandwidth channel connections and the HIPPI channel.

The monitor source code is on one file, and the macro definitions and related constants are on a second file called the foreground processor text file. The monitor is configured for a particular installation by adding the desired macro statements to the foreground processor monitor source file and then assembling the source file using the foreground processor text.

7.2 BACKGROUND PROCESSOR INTERFACE

The operating system runs in the background processors and makes requests to the foreground processor by issuing exit instructions. These requests can be I/O requests or state change requests such as switching from system mode to user mode. The jk field of each system mode exit instruction is decoded by the foreground processor to determine the nature of the system request.

7.3 MACRO DEFINITIONS

The foreground processor has drivers for each device or channel it manages. When one of these macros is called, it generates the configured executable code block and the associated local memory control block that make up a foreground processor device driver.

7.4 FOREGROUND PROCESSOR REAL TIME CAPABILITIES

On the CRAY-3, a process in real time mode can obtain very fast response times by using the direct I/O capability. This foreground processor, direct I/O capability provides response times of about 30 microseconds, allowing support of millisecond or even sub-millisecond frame times. When using this capability, the real time process is locked into central memory and assigned to one or more dedicated processors.

Library routines accessible from both Fortran and C languages are provided to issue I/O requests and to process I/O completions between the memory of the real time process and the real time external device(s). Using these routines, the real time process communicates directly with the foreground processor, which in turn manages the I/O between the real time process and one or more real time external devices. Foreground direct I/O bypasses the CSOS operating system, eliminating operating system overhead for these requests. The foreground processor will ensure that no I/O operations can take place outside the assigned memory space of the real time process.

Real time external devices that can be supported by foreground direct I/O include analog-to-digital conversion equipment, satellite telemetry or radar antenna data acquisition equipment, or other high-speed, low-latency devices to connect with other computer resources. These devices can be connected to either the CRAY-3 low-speed channels capable of 50 or 100 Mbits/second (6 or 12 Mbytes/second), or to the CRAY-3 high-speed HIPPI channel capable of up to 100 Mbytes/second.

Other CSOS operating system real time features are described in the processing features section of this document.

The foreground direct I/O feature provides the following capabilities.

- Ability to bypass the CSOS operating system and manage I/O directly between a real time process and one or more real time external device(s).
- Ability to enter real time mode and dedicate one or more CPU's.
- Ability to respond to an external event within about 30 microseconds, and thereby support millisecond and sub-millisecond frame times.

7.5 FOREGROUND CONFIGURATOR

CSOS includes a new foreground configuration tool which removes the need to generate new versions of foreground processor software (to reflect changing configurations) on the CRAY-3 itself. Instead, these can be generated on the console in a matter of seconds.

7.6 CRAY-3 CONSOLE FACILITIES

The CRAY-3 system includes a new Sun compatible operator console running Sun's UNIX operating system. This provides some important new capabilities:

- The ability to install, mount and boot from a CRAY-3 file system on the console, without requiring CRAY-3 production disks to be on-line
- An optical disk drive for all install materials
- A common interface both for on-line, production running, and for off-line diagnostic activity
- Visual configuration of subsystems using the System Configuration Manager
- All operational functions, except deadstart / dead-dump, can be remoted to an alternative UNIX workstation

Appendix A CCC DOCUMENTATION

A.1 PRESENTATION OF INFORMATION

Cray Computer Corporation provides information about its products at a number of levels and in a variety of ways. All key product information is provided in an electronic form that may be tailored by individual customers to reflect their specific needs. The tools used to create and manipulate this information have been carefully chosen to ensure portability across as wide a range of platforms as possible.

The three basic styles of presentation used are:

- Multimedia
  - Used where interactive or animated presentation of information significantly simplifies understanding. This form of documentation is primarily targetted to a site's training department.

- On-line Electronic
  - Extensively used for product descriptions, real time help, and problem tracking.

- Hard Copy
  - Primarily used in situations where, for logistic reasons, neither of the above mechanisms is readily accessible. Cray Computer's general philosophy is to provide its users with electronic versions of materials and allow them to choose how to reproduce this data in hard copy form.

A.2 DOCUMENTATION TOOLS

Cray Computer employes the following commercially available tools for the preparation of documentation . Users wishing to exploit or modify CCC documentation will need to obtain the appropriate tool, if thay are not already available locally.

- Multimedia
  - Hardware: Apple Macintosh
  - Software: PowerPoint, Hypercard 2

- On-line Electronic
  - Manuals: Any platform that supports FrameMaker 3.0 or later (includes UNIX X Window, Sun View, Apple Macintosh). An individual site can tailor this documentation to meet its own needs.
  - Man Pages: Any platform providing TCP/IP access to the CRAY-3. Man pages are supplied in both nroff source and output forms.
  - Problem Reports: Via TCP/IP access to dBASE IV database, either locally or remotely to CCC headquarters.

- Hard Copy
  - All hard copy documentation is generated from one of the above electronic forms. Certain manuals are generated from man pages to produce hard copy versions. This capability is only currently available at CCC. These manuals are not provided in FrameMaker format and cannot, therefore, be tailored on site.

A.3 DOCUMENTATION CATALOG

A.3.1 Multimedia

- CRAY-3 Architecture
  - CRAY-3 Memory Operation (Macintosh/PowerPoint)
  - CRAY-3 Tailgating Operation (Macintosh/PowerPoint)

A.3.2 Electronic Documentation - Manuals

General

3101 Documentation Catalog

3102 CRAY-3 Software Introduction Manual

Hardware

3201 CSOS On-Line Diagnostic Maintenance Manual, CRAY-3

3202 Hardware Reference Manual

3203 On-Line Diag. N/W Comm. Prgm. (OLNET) Maintenance Manual

3204 On-Line Diagnostic Ready Reference, CRAY-3 CSOS

3205 Site Planning Guide

System Administration

3301 Administrator's Guide for CRAY-3 Systems, CSOS

3302 Data Center Managers, Overview of CSOS

3303 NFS Administration

3304 Security Administration Reference Manual

3305 TCP/IP Administrator's Guide

System Administration

3306 TERMINFO, Defining & Compiling terminfo/csos 1.0

End User Documents

3401 Assembler, CAL Assembler Version 2

3402 bdb User Guide and Internal Reference Manual

3403 CRAY-3 C Programmer's Reference Manual

3404 CRAY-3 Fortran Reference Manual

3405 CRAY-3 Loader Reference Manual

3406 CRAY-3 User Mode Internal Programming Information

3407 CSOS Autotasking Guide

3408 CSOS Primer

3409 CSOS Shell and Variable Ready Reference

3410 Holmes Reference Manual

3411 Foreground Processor Reference Card

3412 Background Processor Reference Card

3413 Multi-Tasking Programmer's Manual, CRAY-3

3414 Performance Utilities Reference Manual

3415 Support Tools Guide

3416 Symbolic Debugging Package Reference Manual

3417 UPDATE Reference Manual

3418 VAST-2/CCC User's Guide

3419 vi Editor Reference Card

System Programmer

3601 CRAY-3 CSOS Foreground Processor Manual

System Programmer

3602 CRAY-3 System Programmer Reference Manual

3603 CSOS 1.0 Kernel Error Message Manual

3604 CSOS Internal Reference Manual - CRAY-3 Computer Systems

3605 DEBUG Tables Internal Reference Manual

3606 Instruction Simulator Reference Manual

A.3.3 Electronic Documentation - Man Pages

A complete set of approximately 2,500 man pages is provided with the CSOS operating system, organized in the conventional UNIX subdirectories. Cray Research's modifications to this structure, for example for system administration man page, have been retained for compatibility with UNICOS environments.

Man pages may be viewed and printed as required on any system which can access CSOS. The modification and reformatting of man pages requires access to the BSD or AT&T versions of the DOCUMENTER'S WORKBENCH. The following manuals, formed from man pages are currently available only in hard copy format from CCC:

Electronic Documentation -- Man Pages

3801 CSOS 1.0 Administrator Commands Reference Manual

3802 CSOS 1.0 Networking Libraries Reference Manual

3803 CSOS 1.0 Miscellaneous Information Manual

3804 CSOS 1.0 Multi-Tasking Libraries Reference Manual

3805 CSOS 1.0 C Library Reference Manual

3806 CRAY-3 Macros & Opdefs Reference Manual

3807 CSOS 1.0 Debugging Tools Libraries Reference Manual

3808 CSOS 1.0 File Formats and Special Files Reference Manual

3809 CSOS 1.0 Fortran Libraries Reference Manual

3810 CSOS 1.0 Holmes Libraries Reference Manual

Electronic Documentation -- Man Pages

3811 CSOS 1.0 Math and Scientific Libraries Reference Manual

3812 CSOS 1.0 OSF/Motif Products Reference Manual

3813 CSOS 1.0 System Calls Reference Manual

3814 CSOS 1.0 User Commands Reference Manual

3815 CSOS 1.0 X Windows (X11R4) Reference Manual