HEWLETT® PACKARD

S-Class and
X-Class Servers

SPP-UX Exemplar User's Guide

Second Edition

SPP-UX Exemplar User's Guide

S-Class and X-Class Servers

B5655-90030

Second Edition

July 1997

Hewlett-Packard Company
Convex Division
Richardson, Texas
United States of America

SPP-UX Exemplar User's Guide

S-Class and X-Class Servers

B5655-90030

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Warranty Statement

HP warrants hardware Products against defects in materials and workmanship. If HP receives notice of such defects during the warranty period, HP will, at its option, either repair or replace hardware Products which prove to be defective. Some newly manufactured Products may contain selected remanufactured parts equivalent to new in performance. Service parts are new or equivalent to new.

This entire book is recyclable.

Printed in the United States of America

Revision Information for

SPP-UX Exemplar User's Guide

S-Class and X-Class Servers

Edition	Document No.	Description
Second	B5655-90030	Second edition, July 1997. Title changed to *SPP-UX Exemplar User's Guide*. New material includes Chapter 5 and information on X-Class Servers. Other changes made throughout.
First	B5655-90003	Initial release, January 1997. Published under the title *Exemplar User's Guide*.

Contents

Figures

Tables

Preface

Purpose and audience

This book provides an introduction to using Hewlett-Packard Exemplar S-Class and X-Class Technical Servers. An overview of the hardware is given, along with descriptions of the software that runs on S-Class and X-Class servers.

This book include details about the SPP-UX operating system, SPP-UX utilities for determining the system configuration, tools for monitoring a server's performance, and programming languages and associated development tools and libraries.

Most topics covered in this book are covered in more detail in the books listed in "Associated documents" on page xiv.

Notational conventions

This section describes the notational conventions used in this book.

`Bold monospace`

In command examples, text shown in **`bold monospace`** identifies user input that must be typed exactly as shown.

`Monospace`

In paragraph text, `monospace` identifies command names.

In command examples, `monospace` identifies command output, including error messages.

Note

A Note highlights supplemental information.

Italic

In paragraph text, *italic* identifies new and important terms and titles of documents.

In command examples, *italic* identifies text that is replaced with text supplied by the user.

manpage(1)

In paragraph text, manpage(1) refers to an entry in the online man pages, where the name of the man page is followed by its section number in parentheses. To view the man page enter

`man` *section manpage*

at the command line. For details see man(1) (enter `man 1 man`).

KEYCAP

In paragraph text, text shown in **KEYCAP** indicates keyboard keys you must press to execute the command. For example, **RETURN** refers to the carriage return key.

Two **KEYCAP** terms separated by a hyphen indicate two keys that you must press simultaneously. For example, **CTRL-d** indicates that you must press the **d** key while holding down the **CTRL** key.

Associated documents

The *Guide to Exemplar Documentation* (B5655-90044) lists and describes all documentation available to support HP Exemplar S-Class and X-Class Servers.

Many HP Exemplar books are available online via the PinPoint book browser. The PinPoint browser and online books are shipped with all S-Class and X-Class systems. For details contact your System Administrator.

To order the *Guide to Exemplar Documentation*, or the other documents listed here, see "Ordering documents" on page xvi.

Programming documentation

The following hardcopy documents may be of interest to Exemplar S-Class and X-Class application developers:

- *Exemplar Programming Guide* (B5600-90001) describes efficient parallel programming techniques that are available with the Exemplar C and Fortran compilers.
- *Exemplar C and Fortran 77 Programmer's Guide* (B5600-90002) introduces the Exemplar C and Fortran 77 compilers, which are based on Hewlett-Packard's C and Fortran 77 compilers and support the Exemplar programming model.
- *HP Fortran 90 Programmer's Reference* (B5876-90001) provides a complete description of the HP Fortran 90 programming language, including all standard features and HP language extensions.

- *Exemplar C++ Programming Guide* (B5630-90001) describes the features and operation of the Exemplar C++ compiler. It is not a complete reference for the C++ compiler but instead focuses on differences between the Exemplar C++ compiler and the standard HP C++ compiler on which it is based.
- *CXdb Quick Reference Card* (B5639-90001) lists frequently-used commands for CXdb, a visual debugger for debugging applications written in Fortran 77, Fortran 90, C, and C++.
- *CXpa Reference* (B5639-90002) describes how to use CXpa, an interactive runtime performance analysis tool for tuning Fortran, C, and C++ applications on Exemplar S-Class, X-Class, and SPP1200/1600 Series systems.
- *CXtrace User's Guide* (B4539-90003) describes CXtrace, a trace-based performance analysis tool for C and Fortran 77 MPI and PVM applications on Exemplar S-Class, X-Class, and SPP1200/1600 Series systems.
- *HP MPI User's Guide* (B6011-90001) describes how to use HP MPI (Message Passing Interface), a library of C– and Fortran-callable routines used for message-passing programming.
- *HP PVM User's Guide* (B5885-90001) describes how to use HP PVM (Parallel Virtual Machine), a library of C- and Fortran-callable routines used for message-passing programming.
- *HP MLIB LAPACK User's Guide* (B5649-90005), *HP MLIB SCILIB User's Guide* (B5649-90006), and *HP MLIB VECLIB User's Guide* (B5649-90007) describe HP's LAPACK, SCILIB, and VECLIB software libraries of Fortran-callable mathematical subprograms.

Architecture and system software documentation

The following books have detailed information on Exemplar S-Class and X-Class hardware and system software:

- *Exemplar Architecture: S-Class and X-Class Servers* (A4716-90001) provides technical information on Exemplar S-Class and X-Class Servers and on the optimization of applications that run on those servers.
- *SPP-UX System Administration Guide* (B5655-90023) describes and provides instructions for the administration and maintenance of SPP-UX V5.2.
- *Checkpoint Restart User's Guide* (B5655-90027) provides instructions on how to save the state of selected processes to disk files and later restart them from their saved states.

- *SPP-UX Large Files User's Guide* (B5655-90032) describes large files and provides information on programming and usage.
- *NQS System Administration Guide* (B5589-90008) describes how to configure and manage NQS+, a batch queuing system.
- *NQS User's Guide* (B5589-90009) provides user-level information on NQS+, a batch queuing system, and instructions on its use.
- *TMR Administrator's Guide* (B6017-90009) explains how to configure and administer the Tape Mount Request (TMR) tape management software system.
- *TMR Operator's Guide* (B6017-90010) describes how to operate TMR, including how to monitor and control tape system resources and how to respond to tape mount and unmount requests.
- *TMR User's Guide* (B6017-90008) introduces TMR, a tape management software system that provides traditional tape management. Provides tape system concepts and instruction for performing tape-related tasks.

Ordering documents

To order additional copies of this document or other documents listed in the section "Associated documents", call 1-800-227-8164 between 6 a.m. and 5 p.m. PST.

To place an order from outside the United States, or if you cannot use the 1-800 number, call 1-415-857-5027.

Please have the order number (*xxxxx*-9*xxxx*) and the exact title of the document available when ordering.

Technical assistance

If you have questions that are not answered in this book, contact the Hewlett-Packard Convex Technical Assistance Center (TAC) at the following locations:

- Within the continental U.S., call 1-800-952-0379.
- From Canada, call 1-800-345-2384.
- All other locations, contact your local Hewlett-Packard office.

You can also use the `contact` utility, if you would like to report any problems you may have. For more information refer to the contact(1) man page.

Introduction 1

This book introduces Hewlett-Packard Exemplar S-Class and X-Class Technical Servers. It describes the Exemplar hardware and software environment and covers how you can use it for running software and developing programs.

Because this book provides general overviews of the topics it covers, you should refer to the books listed in "Associated documents" on page xiv for more detailed information.

Exemplar S-Class and X-Class Technical Servers

Exemplar S-Class and X-Class Technical Servers can solve problems that are too complex or too large to be solved on workstations. They provide high levels of throughput and a reduced time-to-solution for large problems.

S-Class and X-Class servers also are compatible with other Hewlett-Packard PA-RISC platforms, including Hewlett-Packard workstations and other Hewlett-Packard servers. This is partly because of processor compatibility and partly due to compatible operating systems among the platforms.

Exemplar S-Class and X-Class servers have unique hardware features for supporting multiple processors (up to 64 PA-RISC CPUs) and greater I/O and memory capacities than found on other machines. Likewise, the SPP-UX operating system includes HP-UX compatibility as well as distinctive features for running programs in parallel on multiple processors and managing the large memory and I/O capacities.

Another benefit of Exemplar S-Class and X-Class servers is that they are scalable—the amount of processor, I/O, and memory hardware can be adjusted as computing needs demand. Also, regardless of how large or small the hardware configuration is, SPP-UX gives users a view of the server as a single machine, and the interconnecting hardware provides low communication

latency, high bandwidth, global cache coherence, and a large globally shared memory.

The Exemplar S-Class and X-Class environment

An overview of the Exemplar environment is given in Chapter 2. The overview includes both the hardware environment (the processors, physical memory, and rest of the architecture) and software environment (including the operating system, file system, and virtual memory). Also in Chapter 2 is a discussion of *subcomplexes*, which determine the processors and memory available to various users and programs.

Determining an Exemplar server's configuration is covered in Chapter 3. You can get information about:

- The server's name and internet address
- Number of processors
- Memory capacity and cache allocation
- Subcomplex configuration(s) and name(s)
- Available disk space
- Other details

Several "layered" and optional system software packages give additional capabilities to the SPP-UX operating system. Layered system software includes the ability to manipulate large files and the ability to checkpoint and restart applications with long execution times. Optional system software includes the Network Queueing System (NQS+) and the TMR tape management software system. See Chapter 5 for information on layered and optional system software.

Monitoring performance and programming

Monitoring the use of S-Class and X-Class servers is the focus of Chapter 4. You can use several SPP-UX utilities to find out how heavily a server's processors are being used, which programs are running, how many threads a process has spawned, and who is logged in to the server.

Programming tools available for S-Class and X-Class servers are briefly described in Chapter 6. Several programming languages (FORTRAN 77, Fortran 90, C, and C++) are available, in addition to optimized libraries, message passing libraries (PVM and MPI), and performance analysis and debugging tools.

Overview of S-Class and X-Class servers

2

This chapter introduces Hewlett-Packard's Exemplar S-Class and X-Class Technical Servers and the SPP-UX operating system and file system.

Topics covered in this chapter include the following:

- An overview of Hewlett-Packard Exemplar servers
- Descriptions of the S-Class and X-Class architectures, operating system, and file system
- Explanations of how S-Class and X-Class servers' resources can be partitioned (and restricted) for different uses by using subcomplexes
- A discussion of the different categories of memory
- A brief overview of the network, disk, and other I/O devices and controllers supported on S-Class and X-Class servers

Chapter 3 covers utilities for getting information about a server's hardware and software configuration. Utilities for monitoring system use are listed in Chapter 4. Optional system software and "layered" software products are described in Chapter 5. Programming languages, debugging tools, performance tools, and libraries are covered in Chapter 6.

Exemplar Technical Servers

Hewlett-Packard's Exemplar Technical Servers includes several classes of scalable high-performance servers. This section gives a brief overview of the following Exemplar servers: D-Class, K-Class, S-Class, and X-Class.

Exemplar Technical Servers have one or more PA-RISC processors, high-bandwidth distributed memory subsystems, and scalable I/O subsystems. Table 1 is an overview of several Exemplar servers' features.

Table 1 Exemplar Technical Servers

Class	CPUs	Memory	I/O
D-Class	1 to 2	32 MB to 1.5 GB	8 slots maximum (Ethernet, SCSI, SCSI-2, parallel, and RS-232)
K-Class	1 to 4	128 MB to 4 GB	8 slots maximum (Ethernet, SCSI-2, parallel, and RS-232)
S-Class	4 to 16	256 MB to 4 GB[1]	1 to 24 PCI controllers
X-Class	4 to 64	1 GB to 16 GB[2]	1 to 96 PCI controllers

1. 4 GB maximum using 16-mbit SDRAMs.
2. 16 GB maximum using 16-mbit SDRAMs.

Exemplar servers are highly interoperable among each other and with Hewlett-Packard desktop workstations. They are based upon the same processor architecture (Hewlett-Packard's PA-RISC) and provide compatible operating environments (HP-UX on D-Class and K-Class, and SPP-UX on S-Class and X-Class).

This book covers Exemplar S-Class and X-Class servers only.

Exemplar S-Class and X-Class architectures

This section gives an overview of the Exemplar S-Class and X-Class architectures.

Scalability is a major feature of the S-Class and X-Class architectures: as computing needs increase, the number of processors, amount of memory, and I/O capabilities can be increased. Regardless of how the hardware is configured, the SPP-UX operating system presents the server as a single machine.

A complete S-Class or X-Class server consists of at least one hypernode. As shown in Figure 1, an S-Class hypernode includes processors (up to 16 PA-8000 CPUs), memory, and I/O interfaces.

An X-Class hypernode, as in Figure 2, also includes hardware for tightly interconnecting multiple hypernodes. S-Class servers are single-hypernode machines; X-Class servers can include multiple-hypernode configurations, such as Figure 3 and Figure 4.

S-Class servers

The Hewlett-Packard's Exemplar S-Class server supports from four to 16 PA-8000 processors, from 256 MB to 4 GB of physical memory, and from one to 24 high-performance PCI I/O controllers (for network, disk, tape, or other use). Another major component is the crossbar, which provides the processors and I/O controllers access to and from physical memory.

Figure 1 shows an overview of an S-Class server.

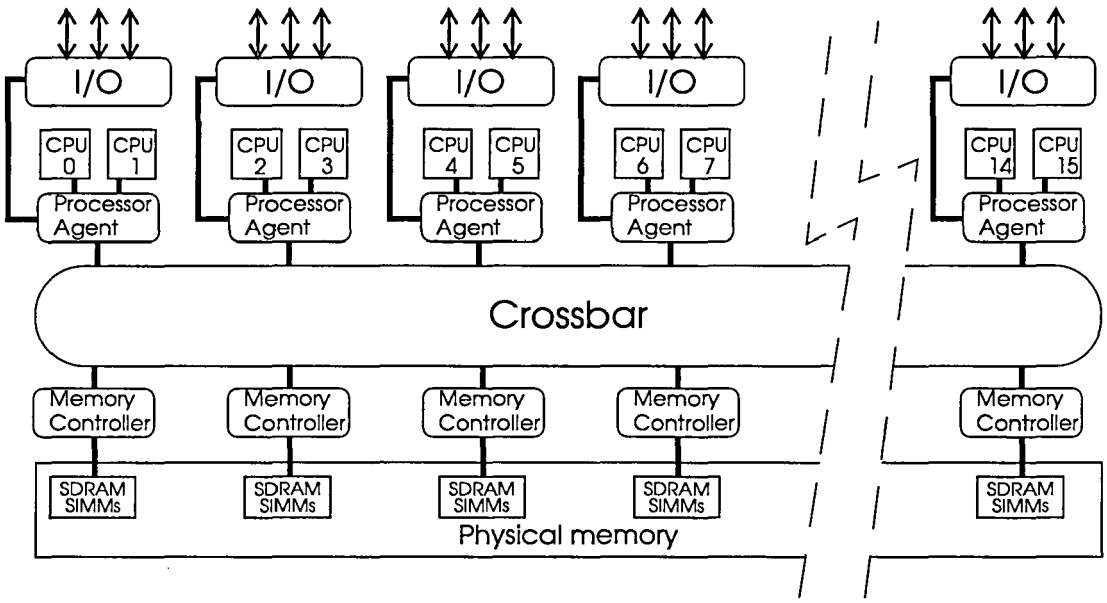


Figure 1 S-Class hardware overview

Figure 1 depicts an S-Class hypernode with the maximum number of processors (16 processors: CPU 0–CPU 15) and I/O ports. A hypernode can have fewer processors or fewer I/O ports present. In addition, a server's memory boards may be partially or fully populated with memory.

Each pair of processors connects to a port on the crossbar by means of a processor agent. This processor agent also serves as the connection for the processors' associated I/O port, if one exists.

The processor agent includes a specialized component called the DataMover, which is dedicated to moving data between memory locations. The DataMover helps increase I/O performance, can speed up memory accesses, and can be of significant benefit to programs that use message-passing libraries such as MPI.

Each I/O port can have up to three PCI controllers connected. The I/O ports are capable of directly transferring data to and from any physical memory in the system, including (on X-Class servers) memory on other hypernodes. This eliminates CPU involvement in data transfers, thus dedicating CPUs to running user programs.

On the other side of the crossbar shown in Figure 1, each port has a memory controller that is the interface to physical memory. On X-Class servers, CTI controllers also are present (see "X-Class servers" below).

The crossbar is nonblocking, so all ports can simultaneously operate at full bandwidth. So, for example, all ports on the processor side can be served their memory requests—without delay—when the requests are made to unique ports on the memory side. Ports on the same side of the crossbar also can communicate with each other, for instance, to pass messages between processors.

X-Class servers

Hewlett-Packard X-Class servers can include up to four hypernodes, each of which is similar to an S-Class system. However, all hypernodes in an X-Class server are tightly coupled into a single system. This is achieved by the coherent toroidal interconnect (CTI), which provides low communication latency, high bandwidth, global cache coherence, and global shared memory for the entire system.

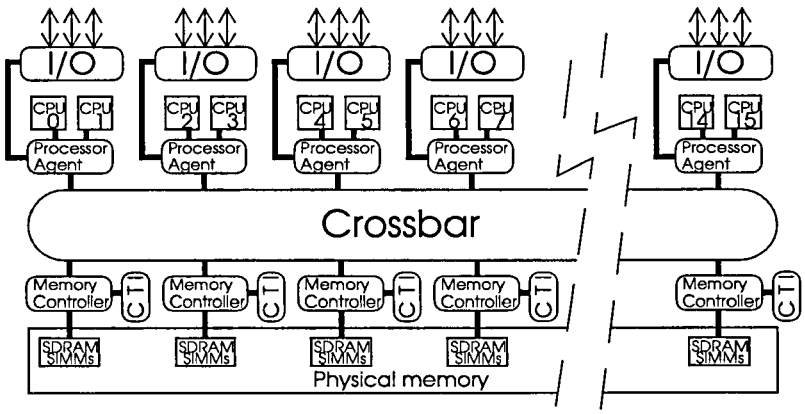


Figure 2 X-Class hardware overview (one hypernode)

Figure 2 shows a single X-Class hypernode. The difference between Figure 2 and Figure 1 is the presence of the CTI controllers, connected to the memory controllers, in Figure 2.

Multihypernode X-Class servers

Hewlett-Packard Exemplar X-Class servers currently may consist of one to four hypernodes and are capable of being scaled further. Each X-Class hypernode has the same CPU, memory, and I/O capacities as an S-Class server, allowing for a total of up to 64 CPUs, 16 GB of physical memory, and 96 PCI I/O controllers in a four-hypernode X-Class server.

The CTI controllers (Coherent Toroidal Interconnect controller) are used in multihypernode X-Class servers. The CTI controllers on different X-Class hypernodes are connected by CTI rings, which are unidirectional (data travels only in one direction on the rings).

For two-hypernode and three-hypernode X-Class servers, CTI rings provide a one-dimensional interconnect. A two-hypernode example is shown in Figure 3.

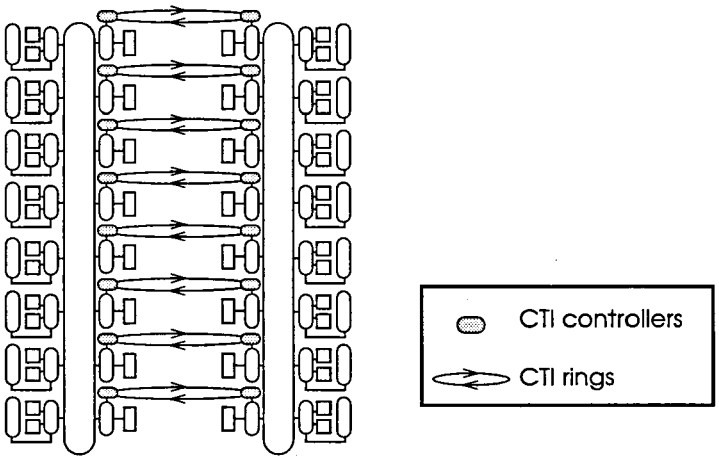


Figure 3 X-Class hardware overview (two hypernodes)

Any processor on a multihypernode X-Class server can access memory on another hypernode by routing its request through its own crossbar to a CTI ring that connects to the hypernode where the remote memory exists. The requested data then is returned via a CTI ring and is routed through the crossbar to the requesting processor.

A one-dimensional interconnect (such as in Figure 3) is sufficient for X-Class servers with up to three-hypernodes. For X-Class servers with more than three hypernodes, a two-dimensional

interconnect is used to shorten paths between the requesting and responding hypernodes.

Figure 4 shows a four-hypernode X-Class server with two dimensions of interconnect. The CTI rings in Figure 4 provide X-dimension interconnect (such as that between hypernode 0 and hypernode 1) and Y-dimension interconnect (such as that between hypernode 0 and hypernode 3, or hypernode 1 and hypernode 2).

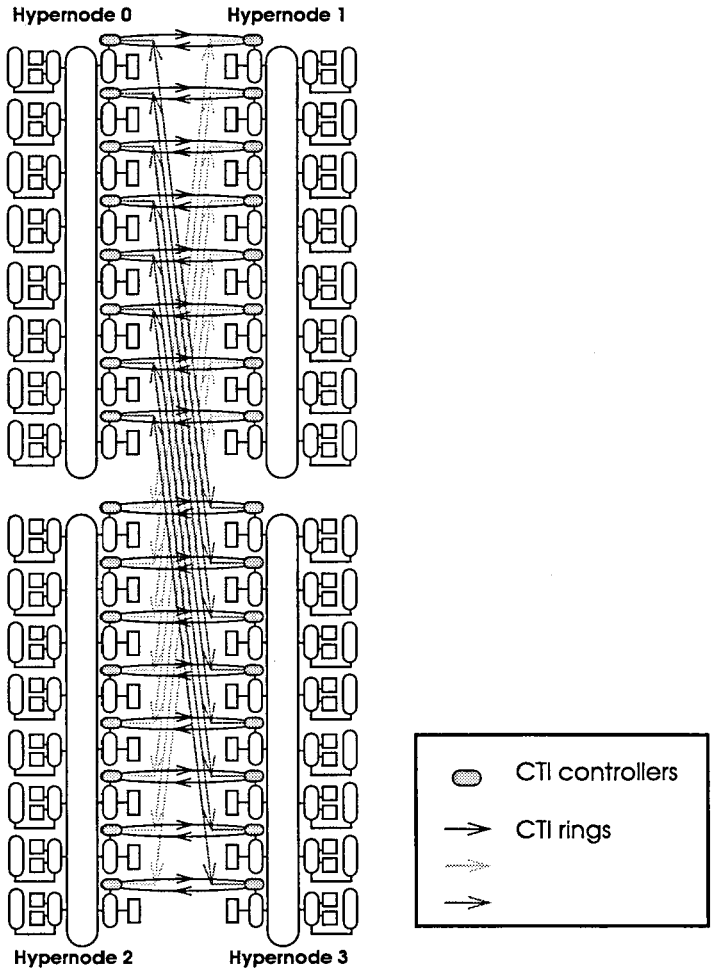


Figure 4 X-Class hardware overview (four hypernodes)

When a processor makes a request for memory from a remote hypernode, the request first travels the required distance, if any, on the X-dimension CTI ring and then, if needed, travels the Y-dimension ring. On return, the response travels an X-dimension

ring first and then a Y-dimension ring. The response does not necessarily travel the same path as the request.

The SPP-UX operating system

SPP-UX is the operating system that runs on S-Class and X-Class Exemplar Technical Servers.

SPP-UX provides compatibility with the HP-UX 10.20 operating system: nearly all HP-UX commands (system utilities) are supported by SPP-UX, and HP-UX 10.20 applications can run on SPP-UX servers without being recompiled.

SPP-UX also includes features not in HP-UX, such as utilities for configuring and monitoring S-Class and X-Class servers, and features that permit parallel applications to execute efficiently on the hardware.

Figure 5 shows an overview of the Exemplar operating environment. The items shown in the shaded area are components of the SPP-UX operating system; the other items (applications, compilers, and tools) run on top of the operating system.

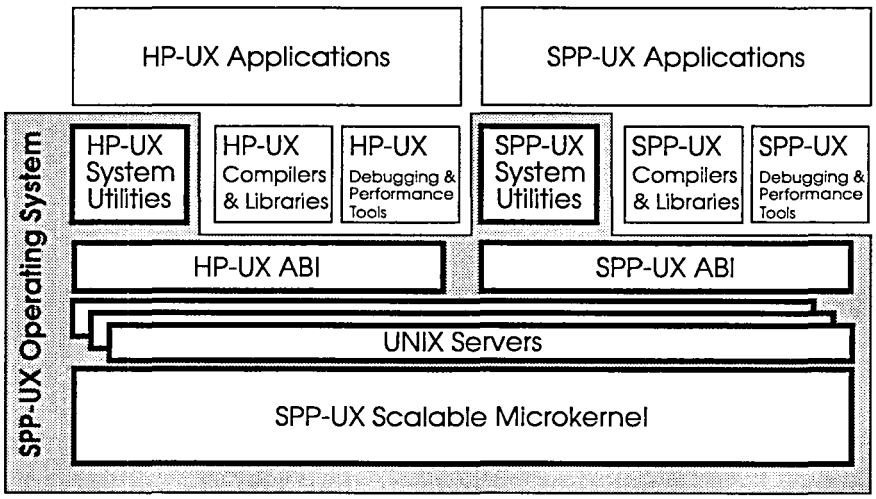


Figure 5 SPP-UX operating environment

As shown in Figure 5, the most basic SPP-UX functionality is provided through the microkernel. The microkernel is responsible for the scheduling of processors, virtual memory, and other operations. On S-Class servers, there is only one microkernel. On X-Class servers, each hypernode runs its own copy of the microkernel (thus providing improved performance by "scaling" the system's operations across hypernodes).

UNIX servers run as needed to manage the file system, networking, subcomplexes, and processes. On X-Class servers, UNIX servers may be distributed across hypernodes to help eliminate resource bottlenecks.

The two ABIs (Application Binary Interfaces) shown in Figure 5 provide the mechanisms for handling system calls that are made by HP-UX and SPP-UX compilers, applications, and system utilities (such as `ls`, `ps`, `pot`, `sysinfo`, and other commands).

Also shown in Figure 5, both HP-UX and SPP-UX system utilities are included in the SPP-UX operating system. (A few of the specialized HP-UX utilities, such as the audio tools, are not provided on S-Class and X-Class servers, which do not have audio hardware.) SPP-UX utilities provide commands not in the HP-UX operating system. Table 2 lists some of the utilities that distinguish SPP-UX from HP-UX.

Table 2 SPP-UX utilities not in HP-UX (partial listing)

SPP-UX command	Summary
`/usr/bin/mpa`	Modifies the attributes of a program for execution.
`/usr/bin/pot`	Displays and updates information about the threads on the system.
`/usr/bin/sod`	Displays format information about SOM and ESOM objects.
`/usr/bin/sysinfo`	Prints system information, including processor, memory, SPP-UX, and subcomplex information.
`/usr/bin/syspic`	Monitors performance of the system and provides real-time graphical representation.
`/usr/bin/chkpnt` `/usr/bin/restart`	Checkpoints (`chkpnt`) and restarts (`restart`) execution of a process or process family.

File system layout

This section describes the SPP-UX file system and summarizes where you can find various types of files.

The term *file system layout* refers to the organization of files and directories. The SPP-UX file system is essentially the same as that used by HP-UX, although some files (such as the kernel files) differ between the two operating systems, and SPP-UX has additional files, as shown in Table 2.

In SPP-UX, as in HP-UX, files are organized by category:

- Operating system files are kept separate from application files
- Configuration files are kept separate from executable files
- Private (local) files are separate from shared (networked) files
- Areas of the file system that may change in size are separate from areas that remain static.

Table 3 lists the main directories found at the root level on all SPP-UX systems. System Administrators may create additional root-level directories on individual systems, but the directories in Table 3 should be sufficient for nearly all purposes.

Table 3 SPP-UX root-level directories

Directory	Contents
/dev	Device files
/etc	Configuration files
/home	Users' home directories
/mnt	Mounting point for local file systems
/net	Mounting point for remote file systems
/opt	Optional products, including applications, compilers, and 3rd-party software
/sbin	SPP-UX system administration tools
/stand	SPP-UX kernel files needed at boot time
/tmp	Temporary "scratch" files
/usr	Various SPP-UX files
/var	Variable-sized files, such as data and log files used by applications

Overview of selected directories

The main directories important to users and programmers are covered in the following sections.

For more details on the file system layout on your Exemplar system, refer to the file /usr/share/doc/filesys.txt. A PostScript version of this file is available as /usr/share/doc/filesys.ps.

/home

The /home directory contains users' home directories. It is private (not networked) and is dynamic (its files may change in size). Users are free to use this area of the file system as they wish.

/opt

The /opt directory contains applications, compilers, and other optional products (programs that are not required for S-Class and X-Class servers to operate).

Files in the /opt directory are organized by product. Executables are installed in /opt/*product*/bin, libraries are placed in /opt/*product*/lib, and so on.

/etc

The /etc directory contains mostly system configuration and administration files, and it does not include any user-executable files. However, the following files are of interest to all users:

/etc/skel/.cshrc
/etc/skel/.exrc
/etc/skel/.login
/etc/skel/.profile

These files provide the system default settings for users' configuration files. When a new user account is created these files are automatically copied to the new user's home directory.

/etc/PATH

This file contains the default system list of directories to search for executable programs.

/etc/MANPATH

The MANPATH file contains the default list of directories searched for man pages when the `man` command is used.

/etc/SHLIB_PATH

This file contains a list of directories where shared libraries can be found.

/usr

The /usr directory includes a variety of subdirectories, including:

/usr/bin
: Operating system user commands.

/usr/contrib
: Unsupported software distributed with SPP-UX, such as `scinfo` and `scname`.

/usr/include
: Header files.

/usr/lib
: Object code and object code libraries.

/usr/local
: Software contributed by local users.

Subcomplexes

Exemplar S-Class and X-Class servers provide a way to balance the use of processors and memory by dividing the system's resources into units called *subcomplexes*. A subcomplex's configuration determines which processors and memory are available to programs that run on that subcomplex.

For details about commands and utilities for managing and using subcomplexes, see Chapter 3.

In effect, a subcomplex is a "logical server" that contains a portion of the server's processors and memory. When several subcomplexes are established, a single Exemplar server can act as multiple smaller servers. Every S-Class and X-Class server has at least one subcomplex and may contain as many subcomplexes as there are processors in the system.

If a subcomplex contains more than one processor, multiple programs can run on different processors within the subcomplex, or an individual program can split up its tasks to run them in parallel.

While subcomplexes do share some hardware resources (such as I/O devices and the crossbar) each processor belongs to only one subcomplex at a time, and memory cannot be shared across subcomplexes.

Exemplar subcomplexes are created and managed by software. Only certain users, such as the System Administrator, have the authority to set and change subcomplex configurations.

Subcomplex permissions

Each subcomplex has a set of *permissions* that are similar to UNIX file permissions. The permissions (read, write, and execute) can be set to apply to individual users, groups of users, or all users.

Read permission allows for reading a subcomplex's configuration information. Write permission enables the subcomplex's process scheduling policies to be changed. Execute permission allows for processes to be run on the subcomplex.

The `scm -c` command prints subcomplex permissions and other information. See Chapter 3 or the scm(1) man page for details.

Using subcomplexes

This section has simple examples of using different subcomplexes on an S-Class or X-Class server. The utilities used here (`mpa`, `scm`, and `scname`) are listed and described along with other subcomplex utilities in Chapter 3.

In the example that follows, `scname` prints the name of the current subcomplex, and then `scm` lists all available subcomplexes on the system.

```
% scname
System
% scm -s
System
Compiler
```

The `mpa` command below runs a shell session in the Compiler subcomplex, thus making it the current subcomplex, as indicated by the output from `scname`. After exiting from the Compiler shell session, the current subcomplex becomes System, as before.

```
% mpa -sc Compiler
% scname
Compiler
% exit
% scname
System
```

To run a program in a specified subcomplex, but not initiate running a shell session in the subcomplex, enter

`mpa -sc` *subcomplex program*

For more details about using these and other subcomplex utilities, refer to Chapter 3 or the utilities' man pages.

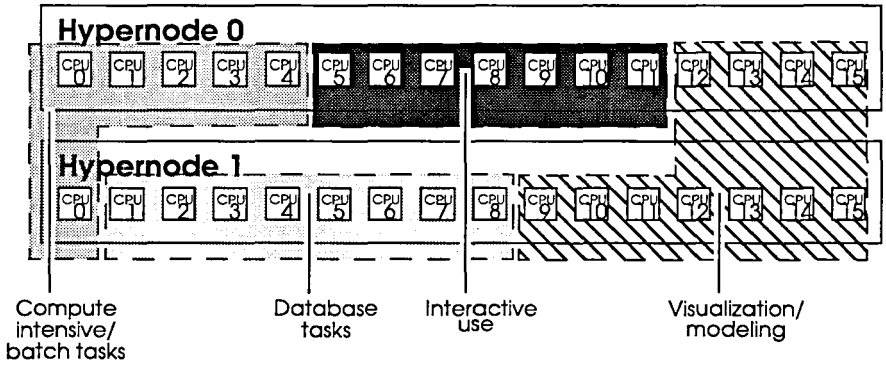
Reconfiguring subcomplexes

An Exemplar's subcomplexes can be reconfigured as needed to balance the loads on various parts of the server. When a subcomplex is reconfigured it may be assigned different processors, or it may be allocated more or less memory or processors.

In many cases a server can be reconfigured while it remains running; this permits the server to be tailored to users' needs without interruption. Some changes require restarting the server.

Exemplar S-Class and X-Class servers can be configured in thousands of different ways to provide sufficient resources for various situations. Figure 6 shows two examples.

Daytime configuration



Nighttime configuration

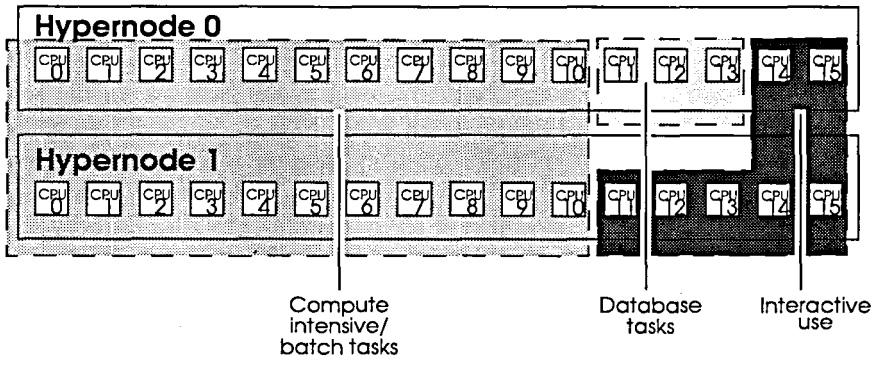


Figure 6 Sample subcomplex configurations for an X-Class server

In Figure 6, a 32-CPU X-Class server is configured for different purposes during the daytime and nighttime. Note that Figure 6 shows just the CPU allocation—but not the memory allocation—for the subcomplexes.

In Figure 6, the daytime configuration provides 6 CPUs for intensive batch processing, 8 CPUs for database tasks, 7 CPUs for interactive use (general user access to the system), and the remaining 11 CPUs for visualization and modeling purposes. In contrast, the nighttime configuration dedicates 22 CPUs to intensive batch processing, provides a different set of 7 CPUs for interactive use, and has 3 CPUs for database tasks.

Memory

This section describes memory on Exemplar S-Class and X-Class servers. Three categories of memory are discussed:

- **Physical memory**—The actual memory hardware
- **Cache memory**—Storage that provides a fast way to read or write data that is expected to be handled in the near future
- **Virtual memory**—The memory space, managed by the SPP-UX microkernel, that is available to programs

Each of these categories is discussed in more detail in the sections that follow.

Overview of memory use

Figure 7 presents a simplified high-level overview of how memory is used on S-Class servers and X-Class servers. (Hypernode-local and global memory is covered later in the section "Virtual memory".)

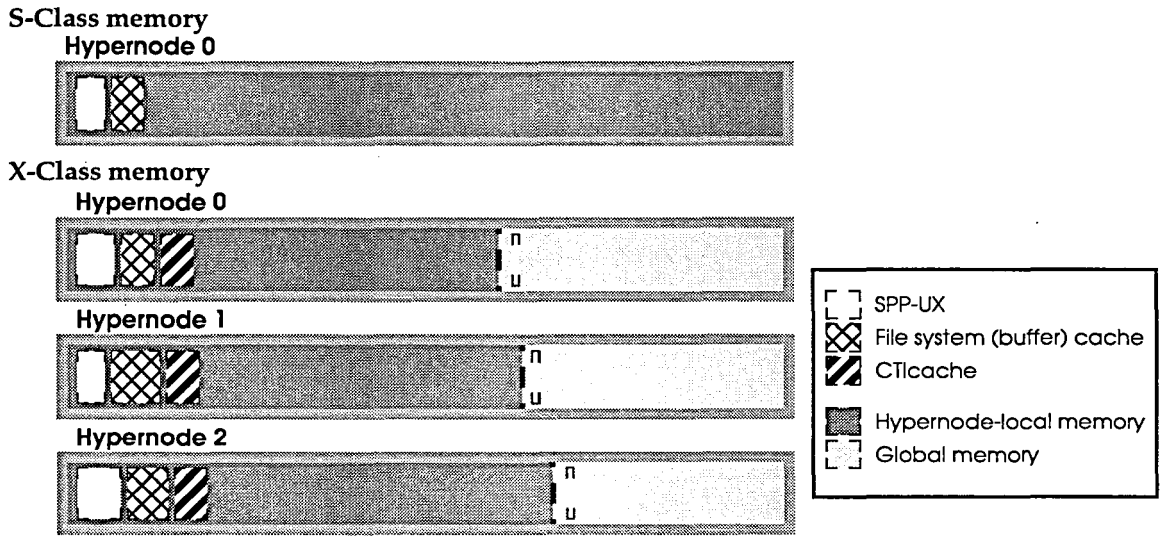


Figure 7 Overview of S-Class and X-Class memory

As shown in Figure 7, virtual memory on S-Class servers consists entirely of *hypernode-local* memory, part of which is occupied by the SPP-UX operating system and the buffer cache, and the remainder of which is available for user programs.

On X-Class servers, virtual memory is partitioned into *hypernode-local* memory and *global* memory. SPP-UX occupies some of the hypernode-local memory and a small portion of the global memory on each hypernode. The buffer cache and CTIcache also occupy portions of hypernode-local memory on each hypernode. The remainder of the virtual memory is available to user programs.

The `sysinfo -memn` command prints the cache sizes and the amounts of global and hypernode-local memory for each hypernode.

Physical memory

Physical memory is the memory hardware, connected to memory controllers, that makes up part of S-Class and X-Class hypernodes (see Figure 1). A hypernode can have 4 GB of physical memory when all memory banks are populated with 16-mbit SDRAM memory modules.

Cache memory

Cache memories are used to temporarily store data in an area—the cache—that provides faster access than the data's normal storage area, such as disk space or, on X-Class servers, memory on a remote hypernode. Items that are encached (instructions or data stored in cache memory) are typically expected to be requested for reading or writing in the near future.

Cache memory on S-Class and X-Class servers includes the file system cache and the processor caches. X-Class servers also have a CTIcache.

For detailed information about caches and their optimal use on S-Class and X-Class servers refer to the *Exemplar Programming Guide*. The following are brief descriptions of how cache memories are used.

- **Processor caches**—The processor caches are off-chip instruction and data caches. These caches are used by PA-8000 processors to reduce the latency of instruction and data fetches. The instruction and data caches are separate from the physical memory, and each is 1 MB in size.

- **CTIcache**—On multihypernode X-Class systems only, part of each hypernode's memory is dedicated as CTIcache. The CTIcache is used to encache global data fetched from other hypernodes.
- **File system cache (buffer cache)**—Each hypernode has part of its memory dedicated as a file system cache. The file system cache is used to encache items read from disk and items that are to be written to disk.

Virtual memory

Virtual memory is the memory space as seen by a program running on an S-Class or X-Class server. There are two types of virtual memory—hypernode-local and global—and each is divided into various classes.

When a process is started, it is given a 4 GB virtual address space. This address space cannot be shared across processes and cannot be shared across subcomplexes, unless it is allocated using the `mmap` or `shmget/shmat` routines (see the mmap(2), shmget(2), and shmat(2) man pages for details).

Depending on a system's configuration, the virtual memory space can be larger than the available physical memory space (with disk space making up the difference as necessary).

Hypernode-local memory and global memory

Access to hypernode-local memory is restricted to the part of a subcomplex that is on the hypernode where the memory resides. Hypernode-local data may exist as a single copy or may exist in multiple copies; see Table 4 on page 19 for details. On S-Class servers, all virtual memory is hypernode-local.

Global virtual memory is accessible throughout a subcomplex, including all hypernodes that make up the subcomplex. Global data exists in a single copy, which is distributed in different ways across hypernodes. On X-Class servers, part of the virtual memory is hypernode-local and part is global (unless 0 K is allocated for global memory).

Classes of virtual memory

Both hypernode-local and global virtual memory are divided into classes; each class provides a different type of data distribution.

Hypernode-local memory includes thread-private memory and node-private memory, two methods of sharing data within hypernodes. Global virtual memory includes near-shared,

far-shared, and block-shared memory; these provide ways to share data across hypernodes.

Note

Virtual memory classes are normally assigned automatically to variables by Exemplar compilers. Programmers can also designate variables' memory classes manually. See the *Exemplar Programming Guide* for details.

Table 4 and Figure 8 describe and illustrate how data in the various virtual memory classes is distributed.

Table 4 Types and classes of virtual memory

Hypernode-local memory	Thread-private	Every thread of a process has its own private copy of the data.
	Node-private	A single version of the data is provided for each hypernode in the subcomplex. All threads in a hypernode share a version of the data. A thread in a hypernode cannot access node-private data in another hypernode.
Global memory	Near-shared	A single copy is accessible by any thread on any hypernode in the subcomplex. Is located on one particular hypernode in the subcomplex.
	Far-shared	A single copy is accessible by any thread on any hypernode in the subcomplex. Is distributed in 4-K blocks across all hypernodes in the subcomplex in round-robin fashion.
	Block-shared	A single copy is accessible by any thread on any hypernode in the subcomplex. Is distributed in contiguous blocks equally among all hypernodes in the subcomplex. Must be dynamically allocated by the program at run time.

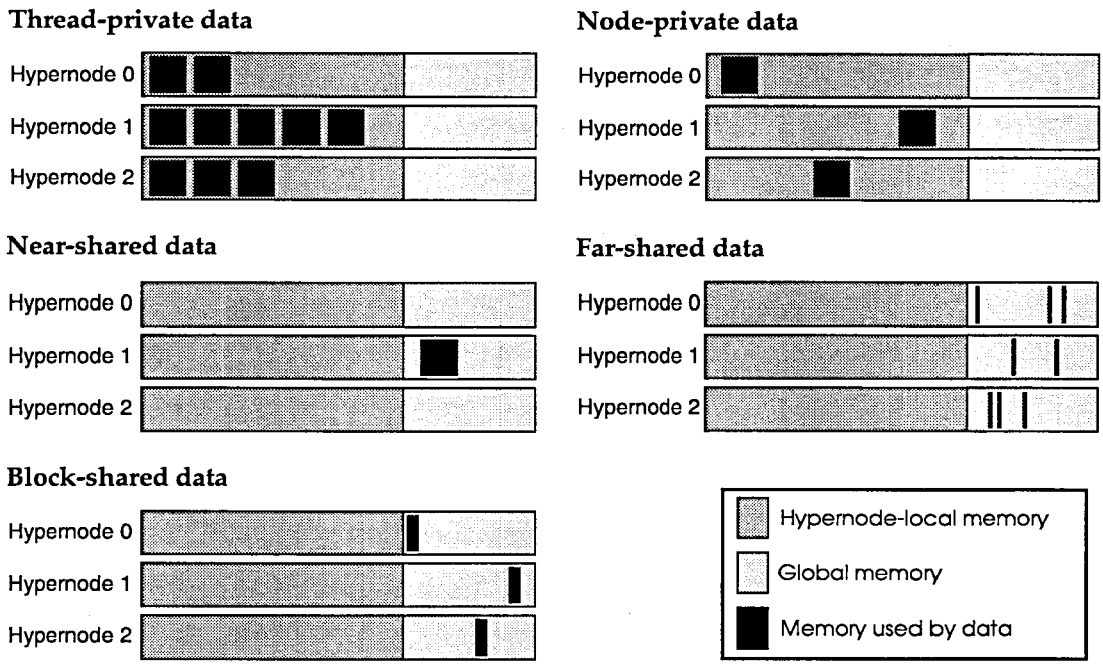


Figure 8 Classes of virtual memory

Figure 8 shows how a 32-K variable occupies the memory space of a 10-processor subcomplex that spans three hypernodes. When the variable is assigned to thread-private memory, every thread is given its own private copy of the variable, for a total of 10 copies that occupy 320 K. (This assumes the program runs 10-way parallel; if the program were run five-way parallel, there would be a total of five copies that occupy 160 K.)

Similarly, when the variable is assigned to node-private memory, every hypernode is given its own copy of the variable, for a total of three copies that occupy 96 K. (If the program were run across two hypernodes, there would instead be two copies that occupy 64 K.)

When the variable is assigned to near-shared, far-shared, or block-shared memory, only one copy exists for a total of 32 K, regardless of the number of threads or hypernodes involved in the program's execution. In these cases, the one copy of the variable is distributed across the hypernodes on which the program is run, as described in Table 4.

I/O devices

Hewlett-Packard's S-Class and X-Class Technical Servers include multiple PCI I/O controllers for connecting disk, tape, network, and other I/O devices.

S-Class servers can include up to 24 PCI I/O controllers, each of which can potentially support multiple devices. X-Class servers can include up to 96 PCI I/O controllers.

I/O devices and controllers that are supported for connecting to the PCI controllers on S-Class and X-Class servers include:

- 9 Gb disk drives
- Disk array storage subsystems
- JBOD (Just a Bunch of Disks) subsystems
- Ultra SCSI controllers
- Digital Audio Tape (DAT) drives
- Digital Tape autoloaders
- Fast Ethernet controllers
- ATM and HiPPi network controllers
- FDDI controllers
- Tier 2 devices

Refer to the *Guide to Exemplar Documentation* for more information about I/O documentation.

Determining the system configuration

3

This chapter lists the utilities and methods you can use to gather information about an Exemplar server's hardware and software configuration.

Utilities for monitoring system use are listed in Chapter 4. Information about "layered" and optional system software packages is given in Chapter 5. Programming languages, debugging tools, performance tools, and libraries are covered in Chapter 6.

SPP-UX utilities

The following utilities permit users to get information about Exemplar S-Class and X-Class machines.

Hardware and subcomplex utilities

SPP-UX provides utilities for getting information about an S-Class or X-Class server's hardware configuration. For examples of using these utilities see the section "Hardware and subcomplex configuration" on page 25.

Table 5 SPP-UX hardware and subcomplex utilities

Utility	Description
`/usr/bin/hostname`	Display the current host system's name
`/etc/ping`	Sends "echo request" packets to a host system; can be used to obtain a machine's IP address

Table 5 (continued) SPP-UX hardware and subcomplex utilities

Utility	Description
`/usr/contrib/bin/scinfo` `/usr/contrib/bin/scname`	Unsupported utilities for printing a subcomplex's name and other information
`/usr/sbin/scm`	Subcomplex Manager utility—provides full descriptions of the current system configuration
`/usr/bin/sysinfo`	Print system information, including processor, memory, SPP-UX, and subcomplex configurations

Software utilities

The utilities in Table 6 can be used to get information about software installed on S-Class and X-Class servers.

For examples of using these utilities see the section "Software configuration" on page 30.

Table 6 SPP-UX utilities for software information

Utility	Description
`/usr/bin/chatr`	View (or change) a program's internal attributes
`/usr/bin/file`	Determine file type
`/usr/sbin/mnm`	Print current SPP-UX memory utilization
`/usr/bin/mpa`	Print (or modify) a program's execution attributes
`/usr/bin/size`	Print section sizes of object files
`/usr/bin/sod`	SOM/ESOM object file dump utility
`/usr/sbin/swlist`	Display information about installed software products
`/usr/bin/sysinfo`	Print system information, including SPP-UX details, subcomplex configurations, and other details

Hardware and subcomplex configuration

This section includes examples and details for using SPP-UX utilities to get information about Exemplar S-Class and X-Class servers.

Server name and address

The `hostname` utility returns the name of the server. In the following example, the server's name is "abednego".

```
% hostname
abednego
```

You can use the `ping` utility to get a server's IP address. In the following example, `ping` is used to get information about the host abednego. In this case the server's name is "abednego", its full name is "abednego.x.hp.com", and its IP address is 15.99.219.13.

```
% ping abednego
PING abednego.x.hp.com: 64 byte packets
64 bytes from 15.99.219.13: icmp_seq=0. time=1. ms
64 bytes from 15.99.219.13: icmp_seq=1. time=1. ms
64 bytes from 15.99.219.13: icmp_seq=2. time=1. ms
64 bytes from 15.99.219.13: icmp_seq=3. time=0. ms
64 bytes from 15.99.219.13: icmp_seq=4. time=1. ms
^C
----abednego.x.hp.com PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/1
```

To stop `ping` from sending packets type **CTRL-C**, or use the `-n` option to limit the number of packets sent. For more details see the ping(1M) man page.

Number of processors

The `sysinfo` utility has two options for getting a count of the processors in an S-Class or X-Class server; some of the other `sysinfo` options are also covered in this chapter. For complete details see the sysinfo(1) man page.

`sysinfo -cpu_count` prints the total number of CPUs in the server. The following example shows the output when run on a 16-CPU server.

```
% sysinfo -cpu_count
16
```

`sysinfo -cpu` prints detailed information for every CPU in a server. Each CPU is listed on its own line, along with information

about which node it is on, whether the CPU is running and whether it is part of the SPP-UX kernel's master or server set, and which subcomplex it is assigned to.

This example shows a three-subcomplex server where CPUs 0-5 are in the "System" subcomplex, CPUs 6-11 belong to the "Math" subcomplex, and CPUs 12 to 15 are in the "Batch" subcomplex.

```
% sysinfo -cpu
node   cpu    flags                                  subcomplex
0      0      Master, Server, Running                System
0      1      Server, Running                        System
0      2      Server, Running                        System
0      3      Server, Running                        System
0      4      Server, Running                        System
0      5      Server, Running                        System
0      6      Server, Running                        Math
0      7      Server, Running                        Math
0      8      Server, Running                        Math
0      9      Server, Running                        Math
0      10     Server, Running                        Math
0      11     Server, Running                        Math
0      12     Server, Running                        Batch
0      13     Server, Running                        Batch
0      14     Server, Running                        Batch
0      15     Server, Running                        Batch
```

Memory availability and cache allocation

Memory on Exemplar S-Class and X-Class servers is used in part for cache memories. Also, on multihypernode X-Class servers, memory is split between node-private and global memory.

The `syspic` graphical system utility has facilities for tracking a server's memory use and buffer cache use. For details refer to the program's Help menu.

The `mnm` utility displays information about a server's memory utilization. It prints the total and free amount of hypernode local (node private) memory and global memory for each hypernode and shows the server's load averages. For a list of `mnm`'s command-line options enter `mnm -h`.

```
% /usr/sbin/mnm -h
/usr/sbin/mnm [-c <x> ] [-d] [-n] [-s <time>] [-p]
        -c <x>    - How many times to display
        -d        - Don't use Curses
        -h        - Display this help
        -n        - Don't display headers
        -p        - Print number of pages instead of Kbytes
        -s <time> - Seconds to sleep, 0 = no sleep Can use decimal eg. 0.25
```

The `sysinfo` command has several options for getting details about an Exemplar server's memory configuration.

`sysinfo -memc` prints memory statistics for the entire server, including all nodes and all subcomplexes.

```
% sysinfo -memc
COMPLEX MEMORY      max    allocated     free
global               0M           0M       0M
node private      3686M         480M    3206M
buffer cache       409M
network cache        0M
total             4096M         480M    3206M
```

`sysinfo -memn` prints memory statistics for one node of a server. (On S-Class servers the `-memc` and `-memn` options provide the same information.)

`sysinfo -shmem` prints information about System V shared memory parameters; see the sysinfo(1) man page.

Subcomplex configuration and name

An Exemplar S-Class or X-Class server's processors and memory can be divided among one or more *subcomplexes*, as described in Chapter 2. (Chapter 2 also has examples of using subcomplexes.)

You can print a server's subcomplex configuration using several SPP-UX utilities, shown in Table 7. Each of the utilities (`scinfo`, `scm`, `scname`, and `sysinfo`) presents subcomplex information in different formats. The `mpa` utility facilitates using subcomplexes.

The `scm` utility also can show a graphical presentation of a server and its subcomplexes. See Table 7 for details.

Table 7 SPP-UX utilities for subcomplex information

Utility (and options)	Description
`mpa`	Executes a file on a specified subcomplex. Also used for printing and changing an ESOM file's parallel attributes.
`scinfo`	An unsupported utility that prints information about the current subcomplex, including: • Subcomplex name • Subcomplex number • Number of CPUs in the subcomplex • Number of nodes the subcomplex spans
`scm`	Displays graphical representations of the server and subcomplex configuration. When invoked with no options, `scm` runs in interactive mode with a graphical user interface; for details refer to the scm(1) man page or the application's Help menu
`scm -c`	Prints a full description of the current system configuration, including the following for each subcomplex: • Subcomplex ID number • Subcomplex name • Subcomplex permissions (read, write, execute) • The user and group ID of the subcomplex's owner • Subcomplex scheduling policy • Subcomplex restrictions (if any) on memory use • List of processors in the subcomplex • The amount of global memory allocated for the subcomplex
`scm -s`	Prints the names of all subcomplexes for which the user has read permission
`scname`	An unsupported utility that prints the name of the current subcomplex
`sysinfo -cpu`	Prints the subcomplex assignment and other details for a server's CPUs. For each CPU the following information is printed: • Which node it is on • Whether the CPU is running • Whether it is part of the SPP-UX kernel's master or server set • Which subcomplex it is assigned to

Disk space

The `bdf` command displays the amount of free disk space available on a file system or set of file systems. This command works identically in both the HP-UX and SPP-UX operating systems. Two examples of the `bdf` command are given here. For complete information see the bdf(1m) man page.

In the following example, the `bdf` command is given the current directory, represented by a period (`.`), as its argument. This causes `bdf` to report usage statistics for the file system that contains the current directory. (In this example the current directory is `/users/kaputnik`, which is printed by the `pwd` command.)

```
% pwd
/users/kaputnik
% bdf .
Filesystem          kbytes     used    avail capacity  Mounted on
/dev/dsk/sd5a      1983928   340904  1444624     19%    /users
```

As shown above, the file system is at 19% capacity (it is 81% unused) and the mount point for the file system is /users. The file system has a total capacity of 1983928 kilobytes, of which 340904 kilobytes are used.

Note

The `bdf` command does not account for any disk space reserved for swap space. For this reason the amount of used and available disk space may not equal the file system's total capacity.

The example below gives a report of free space on all mounted file systems. When no arguments are given to `bdf`, reports for all file systems are printed.

```
% bdf
Filesystem          kbytes     used    avail capacity  Mounted on
/dev/dsk/sd0a       867078   519945   260425     67%    /
/dev/dsk/sd5b      1983928   189304  1596224     11%    /scratch
/dev/dsk/sd5a      1983928   340904  1444624     19%    /users
/dev/dsk/sd4a      7807920    91512  6935616      1%    /work1
/dev/dsk/sd3a      7807920  3612296  3414832     51%    /work2
/dev/dsk/sd6a      3967928  1418744  2152384     40%    /work3
```

To list only locally-mounted file systems, enter `bdf -l`. For a list of just NFS-mounted file systems, enter `bdf -t nfs`. See the bdf(1M) man page for more information.

Software configuration

This section describes ways to get information about the software installed on Exemplar servers. You can display software information including:

- The version number for SPP-UX libraries and executables
- A listing of the optional software installed on a server
- Settings for system tunables
- Other details about the size, properties, and contents of files, executables, objects, and libraries

For details on optional and "layered" system software packages, see Chapter 5.

SPP-UX version

To display the current operating system version use the `sysinfo` command. Entering `sysinfo` with no arguments displays the versions of: the SPP-UX kernel, the SPP-UX server, and the Architectural Interface Library (AIL—see the ail(3x) man page for details).

In the following example, the Exemplar server is running SPP-UX version 5.1.

```
% sysinfo
SPP-UX_mk      5.1 L34 tentacle:/OOW/OOW_5_1 [SPP1_FAST]
SPP-UX_server  5.1 L34 tentacle:/OOW/OOW_5_1 [SPP1_FAST]
SPP-UX_ail     5.1 L34 tentacle:/OOW/OOW_5_1 [SPP1_FAST]
```

The AIL provides interfaces to cache management, synchronization, the DataMover, and other features. The AIL version should match the kernel and server versions.

Library and executable versions

This section covers ways to display the version number for libraries and executable programs. For information about the various types of libraries see Chapter 5.

Some programs have a command-line option for printing the program version. See the program's man page for details.

The `vers` command prints the release version for many libraries and SPP-UX utilities. For information see the vers(1) man page.

The `what` utility extracts and prints version information, if possible, from executable programs and libraries. The following examples show the `what` command printing information about the Fortran 90 library libblas.a (the 10.20.05 version) and the

executable program `netscape` (Netscape 2.0, Motif version 1.2.2).

```
% what /opt/fortran90/lib/libblas.a
/opt/fortran90/lib/libblas.a:
        HP-UX libblas.a PA1.0 960802 (091335)  B3906AA/B3908AA B.10.20.05


% what /usr/local/bin/netscape
/usr/local/bin/netscape:
        Netscape 2.0/export, 23-Jan-96; (c) 1995,1996 Netscape Communications
Corp.
        OSF/Motif Version 1.2.2
```

See the what(1) man page for more information.

Installed software

Installed software includes all files, such as applications, compilers, libraries, and other optional products that are not required for Exemplar servers to operate.

For details on optional and "layered" system software, see Chapter 5.

You can use several methods for finding out what software is installed on an S-Class or X-Class server. This section briefly describes the `swlist`, `whereis`, and `which` SPP-UX utilities and recommends directories to inspect. If these methods do not answer your questions about installed software you can also contact your System Administrator.

The `swlist` utility lists software products that were installed using the `swinstall` program. Most software packages are installed with `swinstall`, including SPP-UX and most compilers, libraries, and other optional products. For details refer to the swlist(1m) man page.

If you are interested in optional software regardless of how it was installed, you can use `ls` to list it. Listing the contents of the /opt, /usr/local/bin, and /usr/contrib/bin directories reveals what is in the primary locations for optional software products.

The `whereis` utility locates programs, man pages, and source files whose name you specify. `whereis` can find programs that are stored in directories not listed in the PATH environment variable. In the following example both the "syspic" executable and man page are located.

```
% whereis syspic
syspic: /usr/bin/syspic /usr/share/man/man1/syspic.1
```

When `whereis` reports more than one executable for a name, the `which` command can determine the one that would be executed. For details see the whereis(1) man page.

The `which` command searches for the file that would be executed if the specified name were given as a command. `which` displays the absolute path of the file or, if the name is aliased, displays the name's alias, as shown in the following examples.

```
% which scinfo
/usr/contrib/bin/scinfo

% which ls
ls:      aliased to ls -F
```

For more information see the which(1) man page.

Tunables information

SPP-UX system *tunables* control various components of the operating system. Tunables (tunable parameters) can be adjusted by settings in the file /stand/tunables, which is maintained by the System Administrator.

Tunable parameters control aspects of the server's behavior in different events, the manner in which system logging is performed, file and process services, and other SPP-UX features. For some tunables of interest to users and programmers see Table 8.

All tunable parameters have default values, which are listed in the tunables(4) man page. These values take effect unless a parameter is given a different value in the file /stand/tunables.

Tunable parameters are established at SPP-UX boot time. For a complete list of SPP-UX tunables see the tunables(4) man page. Table 8 describes some of the user- and process-related tunables.

Table 8 SPP-UX system tunables (partial listing)

Parameter name	Description
buffer_cache_percent	The file system cache (buffer cache) for each node, specified as a percent of physical memory
dfldsiz	The default number of bytes in a process's data segment
maxdsiz	The maximum number of bytes in a process's data segment

Table 8 (continued) SPP-UX system tunables (partial listing)

Parameter name	Description
maxfiles	The maximum number of files a process can have open at once
maxssiz	The maximum number of bytes in a process's stack
maxuprc	The maximum number of processes a user can have at once
maxusers	The maximum number of simultaneous users
msgmax	The maximum number of queued messages a user can have
npty	The maximum number of pseudo-terminals

Device (disk), file, and directory sizes

To print the amount of available disk space on the current disk (or all available disks) use the `bdf` command. See the bdf(1m) man page or the section "Disk space" on page 29.

The `ls` command has options for listing the sizes of files. The `-l` option gives listings in long format, which includes files' sizes in bytes. In this example, `ls` reports that the file libU77.a is 64112 bytes in size.

```
% ls -l /opt/fortran90/lib/libU77.a
-r--r--r--   1 root     10          64112 Sep 24 15:30 /opt/fortran90/lib/libU77.a
```

See the ls(1) man page for more information.

The `du` command gives the number of 512-byte blocks allocated for the files, directories, or both that you specify. For example, `du` would display the following for the file libU77.a.

```
% du /opt/fortran90/lib/libU77.a
128     /opt/fortran90/lib/libU77.a
```

For libU77.a, `du` indicates it occupies 64K (128 512-byte blocks).

In the following example, `du` prints the amount of disk space the current directory occupies (including all files and subdirectories it contains).

The `-s` option forces `du` to print only summary information. In the following example, the directory occupies a total of 2304K (4608 512-byte blocks).

```
% du -s .
4608    .
```

For more details refer to the du(1) man page.

File, object, and executable attributes

You can get information about files and executable programs using several SPP-UX utilities. These utilities can reveal, for example, whether a file is an ASCII text file, a source code file, an executable program, a file in SOM or ESOM format, or whether it has some combination of these attributes.

Table 9 lists utilities for printing a file's attributes and information about its contents. The `chatr` utility and `mpa` utility also can change a file's attributes.

For more details about the utilities in Table 9 refer to their man pages. For examples of the `chatr`, `mpa`, and other utilities see also the *Exemplar C and Fortran 77 Programmer's Guide.*

Table 9 SPP-UX file attribute utilities

Utility	Description
`chatr`	By default, prints a file's magic number and file attributes to standard output; also can be used to change a program's internal attributes.
`chmod`, `chown`, and `chgrp`	`chmod` changes file mode access permissions. The chown command changes a file's owner ID, and `chgrp` changes the group ID.
`file`	Performs a series of tests on a file in an attempt to classify it. Classifications include ASCII text, language source file, SOM or ESOM executable file, and other classes.
`ls`	Lists the contents of directories, and lists file names and any other information requested, such as file permissions (the `-l` option) and group information (the `-g` option).
`mpa`	Print (or modify) a program's execution attributes, such as subcomplex, CPU, memory, and other specifications. Also recommended for executing a file on a specific subcomplex.
`nm`	Displays an object file's symbol table (the list of symbol names referenced or defined in the object file).

Table 9 (continued) SPP-UX file attribute utilities

Utility	Description
`size`	Prints the size of the text, data, and bss (uninitialized data) sections for an object file, along with the total size of the object file.
`sod`	Displays object files in a human-readable form. The `-a` option displays all headers, which contain information such as the minimum and maximum number of CPUs and the memory types.

Monitoring system use 4

This chapter describes ways to monitor how Exemplar S-Class and X-Class servers are being used. Getting information about the following is covered in this chapter:

- How heavily a server is being used
- When the server was booted (how long it has been up and running)
- What programs, processes, and threads are running
- How memory (including cache memory) is being used
- Which users are logged in

The available utilities are listed and described briefly in this chapter. You can find more detailed documentation in the man pages for the utilities.

Utilities for getting information about a server's hardware and software configuration are listed in Chapter 3. Optional and "layered" system software packages are described in Chapter 5. Programming languages, debugging tools, performance tools, and libraries are covered in Chapter 6.

Process, thread, load, and server monitoring

SPP-UX includes utilities for monitoring the programs running on an S-Class or X-Class server, as well as utilities for monitoring the amount and duration of a server's use.

Table 10 lists utilities for getting details about the processes running on a server, the threads running on the server, the server's average load over time, how long a server has been up and running, or combinations of these statistics.

Refer to the man pages for more details about the utilities in Table 10, including additional command-line options and ways to customize the output generated by the utilities. For detailed information about the `syspic` graphical performance monitor refer to that program's Help menu.

Table 10 SPP-UX process, thread, and load utilities

Utility	Description
`pot, top`	`pot` displays and updates information about the threads running on a server. Allows the user to specify which information is displayed and how it is sorted. `top` display and update information about the top processes on the system.
`ps, cnx_ps`	`ps` prints information about selected processes and has options to specify which processes and what information to report. `cnx_ps` includes two additional options: `-T` for providing thread information and `-s` for showing processes from a given subcomplex.
`sysinfo`	Provides a wide variety of system information. System load options include the following: • `sysinfo -lc` prints the mean load average of the entire system. • `sysinfo -ls` prints the load average for all subcomplexes, or for a particular subcomplex if its name is given.
`syspic`	Has a graphical user interface for displaying and, optionally, recording performance details. Recorded statistics can also be played back. Provides a way to monitor the system load and a wide variety of other activities.
`uptime`	Prints the current time, the length of time the system has been up, the number of users logged on to the system, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes.

Memory monitoring

The `sysinfo`, `syspic`, and `mnm` SPP-UX utilities are available for monitoring memory and cache use on Exemplar servers. For details see the section "Memory availability and cache allocation" on page 26.

User information

This section lists commands for getting information about the users logged on to an Exemplar server.

Some system monitoring utilities, such as the `pot` and `ps` commands, print details about users' activities on a server. Additional SPP-UX commands give complete lists of the users on a system and, in the case of `w` and `who`, what they are doing.

The `users` command lists the login names of the users currently on the system in a compact, one-line format.

The `who` and `w` commands print information about users, such as the user name, terminal, login time, and current activity. Details are given in the who(1) man page.

The `finger` command lists details about a specified user, including the following:

- Login name
- Full given name
- Terminal write status (if write permission is denied)
- Idle time
- Login time
- User's home directory and login shell

For complete information see the finger(1) man page.

Layered and optional system software

5

This chapter provides an overview of "layered" and optional system software available for Exemplar S-Class and X-Class Technical Servers.

Layered system software is delivered as part of the standard SPP-UX operating system and includes the following:

- Large files
- Checkpoint Restart (CR)

Optional system software is shipped as part of the SPP-UX operating system but requires that you purchase a license to use it. Optional system software includes:

- Network Queueing System (NQS+)
- Tape Mount Request (TMR)

Large files

The SPP-UX operating system provides *large files* capabilities in many situations. Large files functionality allows for creating and manipulating files larger than two gigabytes in size.

This section provides a brief overview of issues related to using large files on SPP-UX. For more detailed information about limitations, features, and programming support for large files see the *SPP-UX Large Files User's Guide.*

Large files overview

In SPP-UX, the term *large files* refers to files that are greater than 2^{31}-1 bytes in size (approximately two gigabytes). The current upper limit on the size of a large file is one terabyte minus 512 bytes (2^{40}-512 bytes).

The maximum size of a record and the maximum number of records in a direct-access file continue to have the same limit, two gigabytes—that is, MAXINT, or 2,147,483,647.

For you to create or manipulate large files, the file system containing the file must meet the following criteria:

1. The file system must not be an NFS-mounted file system—it must be locally-mounted.

   You can check this using the `bdf` command; see the `bdf` example below.

2. The file system must have both a *magic number* and *featurebits* that indicate support for large files (`FD_MAGIC_2` and `FSF_LARGEFILES FSF_LFN`, respectively).

   Check this using the `/usr/sbin/cnx_dumpfs` command; examples are shown in the following section.

Determining large file support

This section demonstrates how to determine if a file system supports large file creation and manipulation.

For more information about a particular system's configuration, refer to the System Administrator for the system.

Determining local or remote mounting

In the following example, the `bdf` command (entered with no arguments) lists all currently mounted file systems.

Locally-mounted file systems have the potential to support large files; NFS-mounted file systems do not allow large file creation or manipulation. (The currently supported NFS protocol limits the size of files a user may access via NFS to two gigabytes or less.)

```
% bdf
Filesystem            kbytes     used    avail capacity  Mounted on
/dev/dsk/sd0a         877646   626569   163312    79%    /
/dev/dsk/sd12a        500895   168091   282714    37%    /scratch1
/dev/dsk/sd3b        3415736   412064  2662096    13%    /opt
/dev/dsk/sd1a        3903928  3213720   299808    91%    /home
toolbert:/work9      2103721  1438170   644513    69%    /rmt/toolbert/work9
horse:/devsw         4102998  3366918   325780    91%    /rmt/horse/devsw
```

This example shows two NFS-mounted file systems (toolbert:/work9 and horse:/devsw). The others are locally-mounted.

To list only locally-mounted file systems, enter `bdf -l`. For a list of just the NFS-mounted file systems, enter `bdf -t nfs`. See the bdf(1M) man page for more information.

Viewing magic number and featurebits settings

The `cnx_dumpfs` utility prints a file system's magic number and featurebits settings when you specify the file system's mount point. To determine the mount point, refer to the "Mounted on" column printed by the `bdf` command.

As shown in the following examples, the magic number and featurebits settings indicate whether the file system supports large file creation and manipulation.

```
% /usr/sbin/cnx_dumpfs -l /scratch1

magic    FS_MAGIC_LFN
featurebits      FSF_LFN
```

The file system mounted on /scratch1 *does not* support large files. Its magic number is `FS_MAGIC_LFN` (not `FD_MAGIC_2`) and the featurebits setting is not `FSF_LARGEFILES FSF_LFN`.

As shown below, the file system mounted on /opt has support for large files.

```
% /usr/sbin/cnx_dumpfs -l /opt
magic    FD_MAGIC_2
featurebits      FSF_LARGEFILES  FSF_LFN
```

Checkpoint restart

Checkpoint Restart (CR) consists of utilities and library routines that allow you to save the state of a selected process, or process hierarchy, to disk files for subsequent restart.

The information in this section is cursory and general in nature. For greater depth on specific CR issues, please refer to the *Checkpoint Restart User's Guide*.

For an overview of CR features, see "Checkpoint restart overview". For information on using CR, see "General operation of CR" on page 45.

Checkpoint restart overview

You can perform CR operations from the command line and from programs written in C, C++, and Fortran. CR files, including commands and libraries, are installed in the /usr/bin/chkpnt/ and the /usr/bin/restart/ directories.

CR is especially useful for pausing(checkpointing) and restarting applications with lengthy execution times that otherwise, if halted, would have to be restarted from the beginning.

In addition to checkpointing a single process, you can checkpoint and restart—as a group—an entire process hierarchy (that is, a group of processes that have a common origin). However, CR is not designed to save and restart an entire system; in other words, it is not for "rolling-out" and "rolling-in" all processes running on a machine at one time.

Checkpoint restart and large files

The checkpoint utility can checkpoint and restart processes that have large files open.

For additional information on large files refer to the *SPP-UX Large Files User's Guide.*

Checkpoint restart and NQS+

It is possible to checkpoint or restart processes under NQS+ control. NQS+ is a set of utilities for scheduling, queueing, and executing processes in batch mode. For details see "Network Queueing System (NQS+)" on page 47.

The following NQS+ utilities have CR functionality: `qchkpnt`, `qmgr`, `qrestart`.

Checkpointing with open files, pipes, and devices

Information about any regular files, device files, pipes, or named pipes that have been opened by the checkpointed process is recorded and can be restored when the process is restarted. Any data that is in transit through a pipe is stored and replaced in the pipe when the hierarchy is restarted.

The contents of files open to a checkpointed process are not automatically saved; you must explicitly request this action with the `chkpnt -C` option. Ordinarily, only the file pathname and the current position for each open file are saved in the checkpoint file—the file data is not saved.

CR for IPC processes

Processes that use interprocess communication (IPC) can be checkpointed and restarted on Exemplar systems. IPC channels include message queues, shared memory, and semaphores.

Using CR in C, C++, and Fortran

CR provides library routines for use in C, C++, and Fortran programs. The library routines are detailed in the *Checkpoint Restart User's Guide* and the following man pages:

chkpnt(3)	restart(3)	rmckpt(3)
cnx_chkpnt(3)	cnx_restart(3)	

General operation of CR

When you checkpoint a process using the `chkpnt` command (or the `cnx_chkpnt()` function) the following happens:

1. The `cnx_pcontrol` system call stops the execution of that process.
2. A checkpoint file named *process.pid* is written to the current directory (or, if you specify the `-f` option, the file is written to a specific filename and/or directory).

   In the checkpoint file name, *process* represents the program, command, or process being checkpointed. *pid* is the process identification number (PID) of the process being checkpointed.

The checkpoint file contains all information necessary to restart the process in the same condition it was in when it was checkpointed.

CR commands

Two commands allow you to use CR at the SPP-UX prompt or in SPP-UX shell scripts:

`chkpnt`

Checkpoints processes (saves their states to a checkpoint file). For details refer to the chkpnt(1) man page.

`restart`

Restarts processes from checkpoint files. See the restart(1) man page.

The `restart` command may be invoked as a single command, or as an interactive program by using the `-i` option.

The `restart` process must have permission to access files that are needed by the target process. To ensure this, you might need to run `restart` as root or with the same UID as the target file.

Checkpoint file size

The exact amount of disk space used by the checkpoint file for any given process cannot be calculated in advance. The entire address range in use by the checkpointed process—including text, data, and stack—is written to the file.

Generally, the checkpoint file is *at least* as large as the virtual address space of the process.

Restart PID conflicts

When you restart a process, it is by default given the same process identification number (PID) as it had when it was checkpointed. If that PID is the same as the PID of another process already on the system, restarting fails. For information on forcing a restart, refer to the *Checkpoint Restart User's Guide.*

Restarting on a different system

You can checkpoint processes or process hierarchies on one Exemplar system, then restart them on a different system.

Resources needed by the process must be present on the new system, including files, CPUs, and memory. For multinode processes, the same number of nodes and an equal amount of global memory must be present.

The following are considerations when checkpointing and restarting on different systems:

- The operating system releases should be the same between the systems used for both the checkpoint and restart functions.
- The hardware architecture of the two machines must be compatible. In general, if the executable is portable between two architectures (it executes correctly on both machines), then you can checkpoint the process on one and restart it on the other.
- The subcomplexes used when checkpointing and restarting can be different, but the same resources must be available upon checkpoint and restart.
- A copy of the files—if any—that the process requires access to must be present on the new system. Because CR identifies files by pathname, the complete pathname of the copied file must be the same as its pathname on the original machine.

Network Queueing System (NQS+)

The Network Queueing System (NQS+) is optional system software that is shipped as part of SPP-UX. However, NQS+ requires that you purchase a separate license to use it.

NQS+ allows users to submit requests (jobs) to queues for batch execution at a later time.

The information contained in this section is cursory and general in nature. For more information on NQS+, please refer to the *NQS User's Guide*, or *NQS System Administration Guide.*

Note

NQS+ for Exemplar S-Class and X-Class Technical Servers differs from other versions of NQS in several major ways; understanding the differences is important if you combine several systems at one site. For details refer to the *NQS User's Guide.*

Determining if NQS+ is installed

Customarily you will locate NQS+ in the /opt/nqs/ directory (for NQS+ versions 2.3 and later which run on SPP-UX V5.0 and later). NQS+ is shipped with SPP-UX and is installed if the System Administrator selects it during SPP-UX installation.

Use `swlist` to determine if NQS+ is installed on your system. The `swlist` utility provides information on software products installed using the `swinstall` program. For more information on the `swlist` utility consult the swlist(1M) man page.

Contact your System Administrator if you are unable to locate NQS+.

NQS+ overview

NQS+ provides for the submission of user *requests* to a *queue* for subsequent batch execution.

A *request*, by definition, is one or more commands submitted by a user, or a user program, to a queue. A *queue* is an ordered list, employing a prioritized stack process, that contains requests (jobs) waiting to be executed.

Users may submit requests to queues residing on either local or remote machines configured with NQS+.

When submitting requests, you can establish requirements that must be met before the request can be executed (such as dependent program completion, event transaction, or time constraints).

There are three types of NQS+ queues:

- **Batch**—Batch queues run requests. They hold requests for scheduled, perhaps delayed, processing by subsystems within NQS+.
- **Demand**—Demand queues are special batch queues that accept requests only if they can place the requests into immediate execution.
- **Pipe**—Pipe queues are routing queues that do not directly run requests, but instead, transmit requests to other queues. Pipe queues can route requests to all three types of NQS+ queues (batch, demand, and pipe).

NQS+ commands and utilities

Table 11 lists several commands and utilities for configuring and operating NQS+.

Table 11 NQS+ commands

Command	Function
`qstat`	Displays the current status of a specified NQS+ queue. The specified queue may reside locally or remotely.
`qwatch`	Interactively displays the status of NQS+ queues that reside on the local machine.
`qlimit`	Displays the batch queue resource limits supported by the operating system on a specified (local or remote) machine.
`qps`	Displays the status of NQS+ processes associated with batch queues that reside on the local machine.
`qjlist`	Displays the contents of a specified NQS+ request - originated from either a local or remote machine.
`qmapmgr`	Builds and maintains a network database used to establish mapping connections between NQS+ and each machine in the NQS+ configuration.
`qmgr`	Controls queues and requests on the local machine. Allows you to abort, create, configure, delete, enable, and disable requests.
`qdel`	Delete requests.

Using `qmgr` to control requests

The `qmgr` utility provides several commands for controlling NQS+ requests. Table 12 shows the most common `qmgr` commands.

Table 12 qmgr control commands

qmgr command	Description
`delete request`	Deletes a request.
`hold request`	Places a queued request on hold.
`release request`	Releases a hold on a queued request.
`move request`	Moves non-running requests.
`modify request`	Changes a non-running request's priority.

Tape mount request (TMR)

The tape mount request (TMR) software is optional. It is shipped by default with SPP-UX but requires that you purchase a separate license to use it. TMR is a tape management software system. TMR ensures that only one user may access a tape at a time. It also provides a tape labeling process.

The information contained in this section is cursory and general in nature. For more detailed information please refer to the *TMR Administrator's Guide, TMR Operator's Guide,* or *TMR User's Guide.*

Information covered in this section includes the following:

- Determining if TMR is installed
- End user tape operations
- TMR command summary
- Tape label format
- Reports and logs

Determining if TMR is installed

Customarily, you will locate TMR in the /opt/tmr/ directory.

TMR is shipped with SPP-UX and is installed if the System Administrator selects it during SPP-UX installation.

Use the `swlist` utility to see if TMR was installed using the `swinstall` program. For more information on the `swlist` utility, consult the swlist(1M) man page.

Contact the System Administrator with any questions about how a specific SPP-UX Technical Server has been configured.

End user TMR access

TMR functionality is divided into three user groups, based upon how much access and control each group requires. These user groups are: *System Administrator*, *Operator*, and *End User*.

This section concentrates only on the End User level functionality.

Tape management sessions

TMR supports both interactive tape sessions and batch operations. TMR prioritizes incoming requests, allocates resources, and gives the user directions for any necessary actions.

Tape access

TMR permits access to individual files, groups of files, or complete volumes. System Administrators can grant and revoke a restricted set of TMR privileges.

- **Access by file**—Allows files to be selected by filename. Also allows file access to be limited to distinct user groups.
- **Access by volume**—Allows the entire tape volume to be accessed. Each file access request may include instructions to the tape device to seek/forward the tape to a specific position.
- **Privileged access**—A restricted set of TMR privileges are controlled by the System Administrator. The System Administrator may grant or revoke privileges for all users, specific users, or user groups.

TMR command summary

Table 13 lists basic tape session commands.

Table 13 Basic TMR commands

Command	Function
`rlaccess`	Access offline tapes
`rlnext`	Access next offline object
`rlemsg`	Display specified error message
`rlopmsg`	Send a message to the TMR operator
`rlr`	Generate TMR reports

Table 13 (continued) Basic TMR commands

Command	Function
`rlrelease`	Release media and/or device resources
`rlstat`	Display tape management status
`rlxeov`	Cross tape volume boundary

For further information about TMR commands, refer to the *TMR User's Guide.*

Tape label format

TMR supports IBM and ANSI standard label formats. It allows users to specify the exact contents of the labels by using the `rlaccess` command. For further information refer to the *TMR User's Guide.*

Reports and logs

The TMR program provides reporting and logging functions which allow users to check environment configurations, file details, and troubleshooting information. For additional information refer to the *TMR User's Guide.*

Programming tools, languages, and libraries

6

This chapter gives an overview of the programming languages, programming tools, and libraries available on Hewlett-Packard's Exemplar S-Class and X-Class servers. A short summary of programming strategies is also provided.

For detailed information about programming Exemplar servers refer to the *Exemplar Programming Guide*, the *Exemplar C and Fortran 77 Programmer's Guide*, the *HP MPI User's Guide*, the *HP PVM User's Guide*, or other documents listed in the "Associated documents" section of this book's preface.

Utilities for getting information about a server's hardware and software configuration are listed in Chapter 3. Utilities for monitoring system use are listed in Chapter 4. Chapter 5 has information about optional and "layered" system software.

Programming strategies

There are three methods for creating programs that run on Exemplar S-Class and X-Class servers: shared-memory programming, message-passing programming, and hybrid shared-memory/message-passing programming.

Shared-memory programming

In the shared-memory paradigm, the compiler handles most optimizations and, if requested, provides parallelization. Many compiler directives and pragmas are available to enhance this programming method.

The *Exemplar Programming Guide* is the primary book on Exemplar shared-memory programming.

Message-passing programming

The message-passing paradigm involves using functions, such as those in the PVM and MPI libraries, to explicitly spawn parallel processes, share data among the processes, and coordinate activities among them. There is no shared memory in this paradigm—data that is shared is explicitly passed between processes.

The *HP MPI User's Guide* and the *HP PVM User's Guide* are the main books about message-passing programming for Exemplar servers.

Hybrid shared-memory/message-passing programming

Combining the shared-memory and message-passing paradigms allows multiple shared-memory programs to coordinate activities via message passing. By using this approach you can write the majority of a program in the shared-memory style and exploit the benefits of message-passing programming as well.

The *Exemplar Programming Guide* contains examples of both styles of programming. See also the *HP MPI User's Guide* and the *HP PVM User's Guide.*

Parallel programming techniques

This section lists key concepts and alternatives for creating parallel programs. These concepts are not mutually exclusive; so, for instance, a program can take advantage of both automatic and explicit parallelism, while exhibiting both control and data parallelism.

For details about parallel programming, refer to the *Exemplar Programming Guide.*

- **Automatic and explicit parallelism**—Automatic parallelism involves having the compiler do all the work to generate a parallel executable, if possible, for a source file. Explicit parallelism involves manually inserting directives or functions in a program's source code. Explicit parallelism directives or functions intruct the compiler how to parallelize (or not parallelize) portions of a program. The directives also can indicate preferences or hints for dealing with variables or sections of code.

- **Control parallel and data parallel programming**—Control parallel programming assigns different functional sections of code to different processors for simultaneous execution. Data parallel programming divides a data object (or multiple data objects) so that it is distributed to be operated upon concurrently by separate processors.
- **Shared memory and message passing programming**—In shared-memory programming, the compiler, if requested, can handle most optimizations and provide automatic parallelism. The Pthreads the CPSlib libraries provide routines for explicit shared-memory programming.

  Message-passing programming uses functions—such as those in the PVM and MPI libraries—to explicitly spawn parallel processes, share data among the processes, and coordinate activities among them.

Programming languages and compilers

The four main programming languages available for Exemplar S-Class and X-Class servers are: FORTRAN 77, Fortran 90, C, and C++.

Each of these languages is distributed with compilation tools (compilers and related utilities), collections of libraries, and man pages and complete reference books.

Additional programming tools and libraries are also available, as discussed in the following sections ("Programming tools" and "Libraries").

For details about the Exemplar programming languages, refer to the "Associated documents" section for books covering the FORTRAN 77, Fortran 90, C, and C++ languages and related subjects.

Programming tools

This section briefly covers four tools for debugging and analyzing Exemplar programs: CXdb, CXpa, CXtrace, and XMPI. More information is available in the man pages for these products. See also these products' documentation, which is listed in the "Associated documents" section.

CXdb

CXdb can debug C, C++, and Fortran programs. It provides both a graphical user interface and a line-mode interface for working

with programs being debugged. CXdb has many features, including support for the following:

- Debugging of executable programs, core files, and active processes—including multithreaded program debugging features
- Single point of control for debugging multiple processes created with MPI applications
- Views of source code, assembly-language code, process memory, process stack frames, contents of various registers
- Interactive input and output from processes
- Breakpoints and tracepoints for routines, lines, and instructions in a program—including conditional breakpoints
- Watchpoints to monitor address ranges
- Information on program arguments, variables, expressions, arrays, and memory
- Running, re-running, stopping and continuing, and killing programs and processes

CXpa

CXpa is a runtime performance analysis tool for Fortran, C, and C++ programs that run on Exemplar S-Class and X-Class technical servers. CXpa provides three interfaces: an X/Motif graphical user interface (GUI), a character-oriented tty interface (line mode), and a batch interface for integration with scripts and Makefiles.

You can run your application in line mode or batch mode to collect profiling data, then use CXpa's GUI to view a graphical analysis of the data.

CXpa measures a program's entire execution time and reports the total time spent in individual routines, loops, and parallel loops. This produces profiling results that are more complete than statistical sampling.

Using CXpa, you can collect the following performance data for routines, loops, and compiler-generated parallel loops in your program:

- Wall clock time
- CPU time
- CPU/wall clock time (parallel efficiency)
- Execution counts
- Dynamic call graph
- Cache miss counts and latency time for memory accesses

CXpa also includes the ability to:

- Analyze profiling data in 2D and 3D graphs or text reports
- Instrument libraries and object files for routine-level profiling with `cxoi`, a separate utility shipped with CXpa
- Profile MPI and PVM applications
- Clickback to source code during analysis
- View performance data for individual threads or summed across all threads of a process

CXtrace

CXtrace is a performance analysis tool particularly useful for analyzing MPI and PVM message-passing programs. It can be used with programs coded in Fortran 77, C, or both. CXtrace's instrumentation and monitoring systems capture and visualize execution trace information.

XMPI

XMPI is an X/Motif graphical user interface for running applications, monitoring processes and messages, and viewing trace files. XMPI provides a graphical display of the state of processes within an HP MPI application.

XMPI is useful when analyzing programs at the application level (for example, examining HP MPI data types and communicators). Unlike other profilers and debuggers, you can run XMPI without having to recompile or relink your applications.

XMPI runs in one of two modes: postmortem mode or interactive mode. In postmortem mode, you can view trace information for each process in your application. In interactive mode, you can monitor process communications and take snapshots while your application is actually running.

Libraries

Libraries are included as part of the Hewlett-Packard Exemplar programming language environments. Additional, optional message-passing libraries (PVM and MPI) and mathematical routine libraries (HP MLIB) are also available.

This section briefly discusses the differences between shared and archive libraries. It also gives a brief introduction to the PVM, MPI, and HP MLIB collections of libraries.

Shared and archive libraries

Two kinds of libraries are available: archive libraries and shared libraries. Archive library file names end with `.a`, and shared library file names end with `.sl`.

When the linker maps a reference to an archive library routine, the resulting executable contains its own copy of the library routine referenced in the program. An executable that uses only archive libraries is a "complete executable" because it is completely self-contained.

When a shared library is used by the linker to resolve an external reference, the executable does not contain a copy of the library routine but instead contains a linkage table that has the routine's address. An executable that uses a shared library is an "incomplete executable" (and nearly always is *much* smaller than an equivalent complete executable).

Shared library routines are shared among all processes that use the library.

PVM and MPI message-passing libraries

PVM and MPI are both standard specifications for interfaces to libraries of message-passing routines. Both provide a portable way to create parallel message-passing applications.

The HP PVM and HP MPI libraries are available for creating message-passing programs that run on Exemplar servers. Tools such as XMPI are also provided to facilitate tuning programs that use these library routines.

For details refer to the PVM and MPI documentation listed in the "Associated documents" section.

MLIB math libraries

The HP MLIB set of math libraries includes three main components: Scilib, Veclib, and Lapack. These libraries include Fortran-callable (and C-callable) mathematical routines that are optimized for Hewlett-Packard Exemplar parallel servers.

Scilib provides a look-alike implementation of the Scientific Library portion of Cray Research Incorporated's UNICOS Math and Scientific Library V5.0.

Veclib provides mathematical software and computational kernels for application programs involving arrays. Veclib

incorporates BLAS, LINPACK, EISPACK, and other sets of routines.

Lapack is a collection of Fortran-callable subprograms that provides mathematical software for application programs involving linear algebra. (Lapack is an acronym for "Linear Algebra PACKage".)

Refer to the documentation for these MLIB packages for more details.

Index

N

O

P

Q

R

S

T

U

V

W

X

B5655-90030