NAME

Installation − EXODUS Storage Manager 3.1 Installation Manual

DESCRIPTION

This document tells how to get a copy of the EXODUS Storage Manager source and install it on your system.

GETTING THE SOFTWARE

The Exodus Storage Manager is available without charge by anonymous FTP from **ftp.cs.wisc.edu.** All of the Exodus software is located in the **exodus** directory. The Storage Manager sources, documentation, and test programs are available in: **sm/sm.src.3.1.tar.Z**

To eliminate the need to compile the Storage Manager, we provide libraries for SPARCstations/SUN-OS, HP 7xx/HP-UX, and DECstations/ULTRIX in **sm.sparc.3.1.tar.Z**, **sm.hp.3.1.tar.Z** and **sm.dec.3.1.tar.Z**, respectively. The instructions below describe how to un-tar the release and install the Storage Manager by building binaries from the libraries provided. We also include instructions for compiling the Storage Manager for those who wish to do so.

PREREQUISITES

The Storage Manager uses **System V shared memory** and semaphores, so your kernel must have them installed. The kernel should support a shared memory segment of at least 4 megabytes. If you do not know if your system is configured with shared memory, see your system administrator. If you try to run the Storage Manager server on a system that does not have shared memory, the server will print an extensive message about its failure to allocate shared memory, and it will tell you how much shared memory it was trying to acquire; then it will exit. The amount of shared memory needed is based on the server buffer pool size option called bufpages. See the user guide for instructions on setting options.

The Storage Manager is written in C++, and is made with **GNU C++** (g++ version 2.4.5 (we have also made it with 2.3.3 in the past)), so **g++ must be available on your system to install the SM**. The Storage Manager’s client application library has a C interface, so applications can be written in C or C++. If you wish to use the client library (`libsm_client.a`) we provide, you must link applications with g++ (or eg++, the Exodus E compiler, an extension of g++).

Because the include files for the various C++ compilers differ, and in many cases are site-dependent, you may encounter some compilation problems due to differences in include files and their locations. These can usually be resolved with minor changes to the code. If you have problems with compiling because certain system include files cannot be found, see your system administrator. Some system include files (particularly network-related ones) are not included with the standard compiler and library releases, and your system administrator might have to install them for you.

DISK REQUIREMENTS

The compressed tar file for the source release (`sm.src.3.1.tar.Z`) is about 2.5 megabytes. The uncompressed tar file for the source release (`sm.src.3.1.tar`) is about 8.4 megabytes.

The table below lists the minimum disk requirements for making a Storage Manager (compilation and linking), and for the installed files. It includes the doc/ and tests/ directories. These figures are approximate.

Architecture	everything installed but tests	binaries & libraries only
MIPS (DECstations)	13 Megabytes	3.2 Megabytes
SPARCstations	11.5 Megabytes	2.5 Megabytes
HP 7xx	24 Megabytes	5.7 Megabytes

PRELIMINARIES

If the Storage Manager is to be used by multiple users, we recommend (but do not require) creating a user and account for the Storage Manager. For the purpose of this discussion, we call the account **exodus**, but you can give it any name you please. All files created by the Storage Manager’s volume formatting program will be owned by the user who runs it and will be given permission bits 0604 (user read and write permissions, public read permission). If raw disk partitions are to be used with the Storage Manager, they must be mounted so that the Storage Manager can read and write them.

Normally, the Storage Manager listens on a port, named in `/etc/services`, for client messages. The default name of this port is "exodussm". Clients, by default, also look up this port in `/etc/services`. Therefore, we suggest installing in `/etc/services` an entry such as:

```
exodussm  1152/tcp     #exodus Storage Manager
```

Note: You do not need to install this port name now. The SM can run without it as long as certain options are set correctly. The examples in the tests/ directory show how to set these options. More information on `/etc/services` can be found by reading "man services". More information on the server's port usage, including alternative ports, can be found in section 5.2 of the user guide.

INSTALLING THE STORAGE MANAGER

Create the directory in which to un-tar the release. For the purpose of this discussion, call it `SMROOT`.

Extract the sources from the tar file. Perhaps you haven't got enough disk space on one disk to put both the source tar file and the extracted sources in the same directory, so we assume that you put the tar file in /tmp:

```
cd SMROOT
zcat /tmp/sm.src.3.1.tar.Z | tar -xvf -
```

The source release contains the following directories: src, doc, tests, include, lib, and bin. The src directory contains the Storage Manager source code. The doc directory contains documentation on installing and using the Storage Manager. The tests directory contains test and example programs. The include, lib and bin directories are shell directories in the source release. After you install an object release (from `sm.`<*machine*>`.3.1.tar.Z`), the include directory will contain an include file that your applications use, the lib directory will contain the libraries from which you built your executables, The doc and tests directories contain "README" files that give information about their contents.

The libraries we distribute are not complete because there are configuration options and extra functions that some users may wish to include. The client library (`libsm_client.a`), and the server (sm_server) must be built from the libraries contained in the release. There are four steps to install the libraries and build the binaries. The first is to un-tar the object release file:

```
cd SMROOT
zcat /tmp/sm.<machine>.3.1.tar.Z | tar -xvf -
```

The second step is to edit the file `SMROOT/src/config/configOptions.c` to indicate the location of your site's default configuration files and the location of the disk I/O program used by the server. See the comments in configOptions.c for instructions on what to change. As you learn more about the Storage Manager, you may wish to establish a configuration file that is used by everyone at your site. If so, src/config/configOptions.c will need to be updated and the final step (below) repeated.

The third step is to set up a few things for your compiler. If your C++ compiler is not called "g++" you need to change src/makefile.options to use the proper names. The variables to change are CC and CPLUSPLUS located at the bottom of the file. The system can also be installed with the E compiler we distribute. By default, the E compiler attempts to link in `libsm_client.a` (since E programs usually require persistence), but this can cause problems at the link stage of installing or compiling the Storage Manager. Before using E for this purpose, you need to set an environment variable as follows:

```
setenv SM_CLIENT_LIB
```

Once the compiler issues have been settled, the fourth and final step is to make the binaries:

```
cd SMROOT/src
make binary_install
```

This will generate `bin/sm_server` and `lib/libsm_client.a`. Note, you may see warnings from "make" concerning "multiple rules" or "too many command lines". These messages can be ignored.

The user guide, "Using the EXODUS Storage Manager," refers to this installation guide for the locations of the files needed to run the Storage Manager. Here is the list of files and their location in the release:

```
libsm_client.a:        SMROOT/lib
sm_client.h:           SMROOT/include
sm_server:             SMROOT/bin
formatvol:             SMROOT/bin
shutserver:            SMROOT/bin
diskrw:                SMROOT/bin
producer/consumer example:  SMROOT/tests/producer_consumer
```

The tests that are included with the release refer to binaries and libraries in the `SMROOT/bin/`, `SMROOT/lib/`, and `SMROOT/include/` directories. Once you have a working Storage Manager, you can copy or move the contents of these directories to their final destinations.

Print the documentation. See the README in the `SMROOT/doc` directory for instructions.

Test the binary versions of the Storage Manager.

```
cd SMROOT/tests
more README
```

In `SMROOT/tests` are tests and example programs. The README tells you how to run the tests. If the tests are successful, you are done. If the binaries you install do not work on your machine, check the prerequisites, above. If your system meets all the prerequisites, you may want to try making a Storage Manager from scratch.

COMPILING THE STORAGE MANAGER

As distributed, the makefiles will compile the entire system with g++. If wish to use E, remember to set the SM_CLIENT_LIB environment variable to null (as described in the installation instructions above), to avoid having E link in its version of `libsm_client.a`.

If you use a compiler other than g++, you must determine if the compiler understands the keyword **volatile.** Synchronization of the server processes uses the Silberschatz & Peterson algorithm #4 (p. 88, Operating System Concepts, Alternate Edition, Addison-Wesley, 1988), which relies on the **volatile** storage class. There are two ways to make the few files that implement the synchronization primitives.

Volatile works

If your C++ compiler understands and properly handles **volatile,** (or if it does not optimize away the references to the volatile data, even if the keyword **volatile** does not appear), and your C++ passes the keyword along to its underlying C compiler, the entire system can be made with C++ without changes to any source code.

Volatile does not work

Your C++ compiler does not handle **volatile,** (it optimizes the memory references away). If your machine has a test-and-set instruction (e.g.,SPARC), an alternative synchronization algorithm is used. The system is compiled with C++ and with -DTESTANDSET. If you are porting the Storage Manager to a new architecture, you will have to write a test-and-set function (see `SMROOT/src/serverlib/disk/testandset.S`).

The determination is made by make(1) on the basis of the macros CPLUSPLUS (found in `SMROOT/makefile.options`), HOSTTYPE (defined by make on some architectures) and CPUTYPE (defined by make on other architectures). You must not redefine HOSTTYPE or CPUTYPE when you make a Storage Manager.

Be sure that your shell limits are high or unlimited (see csh(1)). To run build the Storage Manager, do

```
cd SMROOT/src
make install
```

This will install the binaries in `SMROOT/{bin,lib}`. More information on making the Storage Manager is available in `SMROOT/doc/Makefiles.man`. Tests can then be run as described above.

BROWSING THE STORAGE MANAGER SOURCE

The Storage Manager source is in `SMROOT/src`. General information on the Storage Manager source is available in `SMROOT/doc/source.man`. Information on the Storage Manager's architecture is available in `SMROOT/doc/arch_overview.me`. If you wish to browse the source using tags you must first make the tags files by running

```
make tags
```

in the `SMROOT/src` directory. Also, put in your `~/.exrc` the following:

```
set tags=./tags ./tags.all
```

SEE ALSO

Makefiles – making an Exodus Storage Manager