Shore Software Installation Manual[1]

The Shore Project Group
Computer Sciences Department
UW-Madison
Madison, WI

Version 1.1.1

October 27, 1997

[1]This research is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-92-C-Q508.

Contents

1 Introduction

The Shore Software distribution consists of documentation, binary, and source releases. This document describes how to install the releases, and how to build Shore from the source release. To begin using Shore to write applications, it is sufficient to install just the binary and documentation releases.

If you have any problems installing the software, please follow the directions in *How to Submit a Problem Report*.

2 Requirements

To install and run the Shore documentation and binary releases, you will need,

- Solaris running on a SPARCstation or Intel X86 workstation (we built this release under Solaris 2.5.1 on a SPARCstation-10 and on Pentium Pro);
- An up-to-date version of the GNU C/C++ compiler (programs compiled with other compilers will not be link-compatible with the libraries supplied). Binary release 1.1.1 was built with gcc and g++ version 2.7.2.3 and #include files from libg++ release 2.7.2. Use of older versions of the compiler or libraries may cause problems. This software is available from the Free Software Foundation, http://www.fsf.org;
- the *gzip* file (de)compression facility to unpack the release. This program is also available from the Free Software Foundation;
- lots of free disk space: roughly

  - 5 MB for the documentation release (1.5 MB for HTML, 3.3 MB for Postscript). You can conserve disk space by throwing the Postscript version after (or before printing) it. You can even throw away both versions and read the documentation over the world-wide web directly from the University of Wisconsin site, as well as
  - 70 MB (or less, depending on the architecture) for the binaries configured with debugging,
  - 25 MB (or less, depending on the architecture) for the binaries configured without debugging, or
  - 200 MB (or less, depending on the architecture) for the source release and space to build from scratch;
- a World-Wide-Web (WWW) browser for reading the HTML versions of the documentation, or software for reading or printing Postscript.

If you also wish to build Shore from the sources, you will also need

- an up-to-date version of GNU gcc and associated libraries (version 2.7.2.1 or later–we are using version 2.7.2.3);
- disk space to build (in addition to space for sources):
  - 115 MB without debugging symbols and without debugging code (-UDEBUG) or
  - 170 MB with debugging symbols and with debugging code (-DDEBUG);
- Perl 5 (we are currently using version 5.003). For more information about Perl, see the Perl home page at http://www.perl.com/perl;
- GNU Make (we are currently using version 3.74), Flex (we are using version 2.5.2), and Bison (we are using version 1.24). These programs are available from the Free Software Foundation. Flex and Bison are only necessary if you wish to rebuild the SDL compiler. The GNU version of `make` is absolutely essential. None of the Makefiles in the source distribution are likely to work with other versions of `make`;
- the TCL (Tool Command Language) library (we use version 7.4). TCL is used to provide a command-line (shell-like) interface to various parts of Shore, including the Storage Manager and the Shore Server. We hope in future releases to get rid of the dependency on TCL. TCL is currently available from ftp://ftp.smli.com/pub/tcl or http://sunscript.sun.com/; and
- the normal Unix utilities /bin/sh, /bin/sed, /bin/mv, etc.

For all the following installation steps, we use two variables, so that you can “cut and paste” the instructions below.

```
set TARDIR=directory-that-holds-tar-files
set SHROOT=directory-in-which-to-install-shore
```

(If you use the Bourne shell (`/bin/sh`) or one of the shells derived from it, such as ksh, omit the word "set".)

Whatever shell you use, make a directory in which to install Shore. (If you plan to install both the debugging and no-debugging versions, create a separate directory for each.)

```
mkdir $SHROOT
```

3 Installing the Documentation Release

If you are installing Shore for the first time and you just copied the tar files with FTP and followed the instructions in the README file there, you will already have finished this step, and you can skip to the next section.

Set the shell variables TARDIR and SHROOT and create the directory $SHROOT. Then extract the documentation.

```
set TARDIR=directory-that-holds-tar-files
set SHROOT=directory-in-which-to-install-shore
```

If you use the Bourne shell (`/bin/sh`) or one of the shells derived from it, such as ksh, omit the word "set".

```
mkdir $SHROOT
cd $SHROOT
gzip -d -c $TARDIR/doc.tar.gz | tar -xvf -
```

This step creates three directories, `docs.html`, `docs.ps`, and `examples`. The `$SHROOT/docs.html` directory contains HTML versions of the documentation. You can read these documents with a World-Wide-Web (WWW) browser such as Netscape or Mosaic. Click the "`Open...`" button and type

```
file://localhost/SHROOT/docs.html/homepage.html
```

where *SHROOT* is the directory where you un-`tar`'ed the documentation release.

These documents are also available over the network at

```
http://cs.wisc.edu/shore/doc
```

You may delete `docs.html` if you do not want to keep local copies.

The directory `$SHROOT/docs.ps` contains Postscript versions of the documents suitable for printing on a laser printer. You may delete this directory after printing the documentation.

The `examples` directory contains example and test programs discussed in the documentation. You can get the examples without the HTML or Postscript documentation by fetching `examples.tar.gz` rather than `doc.tar.gz`.

4 Installing a Binary Release

A Shore binary release contains contains everything needed to write, compile and run Shore applications. A binary release is specific to a particular hardware/OS platform and is compiled with either debugging/auditing support or extra optimization and no debugging support.

Set the shell variables TARDIR and SHROOT and create the directory $SHROOT as explained in the **Requirements** section. Follow these steps to install a binary release for SPARC Solaris 1.5 with full debugging support.

```
cd $SHROOT
gzip -d -c $TARDIR/sparc.dbg.tar.gz | tar -xvf -
```

(Substitute `x86` for `sparc` and/or `opt` for `dbg` as appropriate.)

This step creates sub-directories `bin, lib` and `include` in `$SHROOT`. If you are installing both debugging and non-debugging binaries, un-`tar` them in distinct subdirectories of $SHROOT.

That's it! You can test your installation by following the instructions in the Section Testing Your Installation.

5 Compiling and Installing the Source Release

This section describes how to re-build Shore from the source release. First, un-`tar` the release somewhere. Set the shell variables TARDIR and SHROOT and create the directory $SHROOT as explained in the **Requirements** section. You can extract the sources in the same directory as a binary or documentation release, or you can make a new directory for them.

```
# Go to a directory where you want to install
# the sources.
# For simplicity, we let that be $SHROOT.
# If you use another directory, make the
# appropriate substitution for SHROOT in
# the next step.
set SRCDIR=$SHROOT
cd $SRCDIR
gzip -d -c $TARDIR/src.tar.gz | tar xvf -
```

(users of sh, ksh, etc. should omit the word "set").

This step creates directories `$SRCDIR/config`, `$SRCDIR/tools`, and `$SRCDIR/src`. The directions below tell you how to determine where the binaries will be installed when you build Shore from the sources.

5.0.1 Setting Configuration Options

Before you build Shore, you must give values to some configuration options. These options are located at the top of the file `$SRCDIR/config/config.h`.

```
vi $SRCDIR/config/config.h
```

For compilation to succeed you *must* set the values of FlexLib and FlexInclude to the absolute paths of the flex library libfl.a and the directory containing FlexLexer.h, respectively. Similarly, you must set TclLib and TclInclude to the absolute paths of the TCL library libtcl.a, and the directory containing tcl.h. In addition, either make sure that gcc, g++, bison, flex, perl, and tcl are in your *path* or modify the corresponding variables in $SRCDIR/config/config.h as appropriate. Be sure that you are you have the correct versions of these programs as indicated in the **Requirements** section. Also be sure that your *path* is set up so that "make" invokes GNU make.

You may also change the IncludeDebugCode, Debugging, Optimize, RcDebugging, and InstallDir options as indicated by comments in the config.h file.

5.1 Compilation Steps

```
cd $SRCDIR

# generate the Makefiles.
perl tools/makemake -r

# start compiling
cd $SRCDIR/src
make
```

The Perl script tools/makemake generates a Makefile from the Imakefile in each source directory, as controlled by various files in the $SRCDIR/config directory. (Unlike previous releases of Shore, we no longer used the imake utility from the X Window system to generate Makefiles. The script makemake, although inspired by imake is written from scratch in Perl and has many features lacking in imake.) Once Makefiles are generated, later changes to Imakefiles can be installed by typing make make in the src directory.

The command make depend can be used to generate .depend files in all the source directories. It is not necessary to invoke this command before compiling Shore, but if you intend to make modifications to any source files, it is a good idea to do it before proceeding, to ensure that modifications of header files, sdl grammar files, etc. properly cause recompilation of all affected files. Typing make in one directory will only build programs in that directory. Therefore, if you are making changes in one source directory, you can do a local make to check for compilation errors, but you should then do a make in the directory $SRCDIR/src to make sure that all other directories that depend on your changes are rebuilt. You can type make depend before make all if you like. It will build the tools it needs to complete its job. For example, make depend effectively does make all in $SRCDIR/src/rpc4.0/rpcgen because rpcgen is needed to make the dependencies in the Shore Value-Added Server.

The principal make targets (all, depend, install, clean, etc.) descend recursively into all sub-directories of the directory in which they are invoked. There are counterparts all_local, depend_local, install_local, clean_local, etc.) that affect *only* the current directory.

5.2 Installing the Compiled Files

The make target install installs executable files in $SRCDIR/installed/bin, include files in $SRCDIR/installed/include, and other supporting files in $SRCDIR/installed/lib. It also installs code and README files from src/examples and src/oo7 in $SRCDIR/installed/examples. (You can replace $SRCDIR/installed by another location by changing the definition of InstallDir in $SRCDIR/config/config.h and re-running make make).

```
# edit InstallDir in config/config.h if you
# want to change the destination

cd $SRCDIR/src
make install
```

6 Testing Your Installation

This section lists steps to test your installation. Included are instructions for running the Shore server, compiling and running some example programs, and mounting the Shore file system. There are also instructions for compiling a simple value-added server built with the Shore Storage Manager. Only brief explanations are provided. Pointers to relevant documents are given.

6.1 Running the Shore Server

Set the environment variable SHORE to point to the directory containing the directories bin, lib, etc. containing the Shore programs and libraries. If you installed a binary release, this directory will be $SHROOT (the shell variable SHROOT is explained in the **Requirements** section). If you built the binaries from the source release, it will be $SRCDIR/installed.

```
setenv SHORE $SHROOT
# or
setenv SHORE $SRCDIR/installed

# if you use sh, ksh, etc, do this instead
SHORE=$SHROOT
# or
SHORE=$SRCDIR/installed

export SHORE
```

Copy the configuration files to your home directory and modify a line in the .shoreconfig file to point to installation location. (In the sed command below, treat PUT-YOUR-INSTALLED-DIR-HERE literally. You can yank this entire command with your mouse.)

```
sed -e "s,PUT-YOUR-INSTALLED-DIR-HERE,$SHORE," \
     $SHORE/lib/options > ~/.shoreconfig
```

```
    cp $SHORE/lib/shore.rc ~
    # in case you want to edit them later,
    chmod u+w ~/.shoreconfig ~/shore.rc
```

Make a directory in which to run the Shore server. This directory will have subdirectories `log` and `volumes`. You must run the server while in this directory since the `~/.shoreconfig` and `~/shore.rc` files refer to `./log` and `./volumes`. The directory should be in a file system with at least 100 MB of free space (this is because the log and volume sizes in the distributed configuration files are this large).

```
    # create a directory for running the Shore server
    mkdir shoreserver
    cd shoreserver
    mkdir log volumes
```

Now you can run the Shore server.

```
    $SHORE/bin/shore
```

You can safely ignore the warning

```
Warning: no database administrator -- running under userid ...
```

Expect the server to print a few lines like these:

```
Looking for run command file "~/shore.rc"...
format --./volumes/miniroot-- 5000 true
mkfs ./volumes/miniroot 5000 10
Done serving devices and making filesystems.

root = 0
setroot  10
/ is now 0.0.0.0:10.20007
See if  /  exists in the Shore namespace.
/  does exist
Done putting together the namespace.

Done reading ~/shore.rc.
Shore Server
Release 1.1.1
(0)%
```

Try typing the following commands to the Shore server.

```
    # list directories, registered objects, pools in root directory
    ls -al
    help
```

To shut down the server try any one of these:

```
# at the server prompt
q[uit]
bye
exit
ctrl-D

# or open another window and run this program
$SHORE/bin/sshutdown
```

The document, *Running a Shore Server*, gives more information on configuring and running the Shore Server.

6.2 Compiling and Running an Application

The first test uses one of the examples shipped in the documentation release. See `pscan/README` for details. For other examples, see the `README` in each of the examples directories.

For the following examples, you must have a server running. If you don't, start one as described above in another window, or in the background. (The server needs to be running for the `make`, since it invokes the SDL compiler, which needs to contact the server to register types.) The commands in the following paragraphs should be typed to a Unix shell in another window, not to the Shore VAS shell.

You must also fetch and unpack a copy of the example programs either by installing the documentation release or the examples release as described above, First copy the entire `examples/pscan` directory into another directory.

```
mkdir pscan
cp $SHROOT/examples/pscan/* pscan
```

(The directory $SHROOT/examples is part of the documentation release).

Then build the test programs `build`, `pscan`, and `destroy` and try running them.

```
cd pscan
make
build 10 testpool
pscan testpool
destroy testpool
```

The output should look something like this:

```
% build 10 testpool
Created 10 parts
% pscan testpool
0, Part0
1, Part1
2, Part2
3, Part3
4, Part4
```

```
    5, Part5
    6, Part6
    7, Part7
    8, Part8
    9, Part9
    Found 10 parts
    % destroy testpool
    Destroyed 10parts
```

The document *Getting Started with Shore* gives information about writing Shore applications, using `$SHROOT/examples/stree` as an example.

6.3 NFS-Mounting the Shore File System

The Shore Server creates and manages a name space of objects, the Shore file system, that looks like a Unix file system. The Shore file system can be mounted as an NFS remote file system, allowing you to use standard commands such as `ls` to navigate it.

To mount the Shore file system you must have super-user (root) privileges on your machine.

```
    su
    # create a mount point (an empty Unix directory)
    mkdir /shoremnt

    $SHORE/bin/mnt localhost /shoremnt $SHORE/bin
    exit
```

To check the mount (you don't have to be super-user):

```
    # use your system's mount command to see what's mounted
    mount
    # The mount command may not be in your path by default.  For Solaris, the
    # command is /usr/sbin/mount
```

You should see something like this:

```
/shoremnt on localhost:/
    soft/intr/port=2999/mport=2997/timeo=60/rsize=8192
    /wsize=8192/mport=2997/remote
    on Tue Aug  5 15:40:27 1997
```

(the exact format varies from platform to platform; it will probably all be on one line).

```
    # at an ordinary shell prompt in one window, type
    cp /etc/motd /shoremnt
    # at the prompt from the shore server that you started earlier, type
    ls -l /
    # at the ordinary shell prompt, type
    ls -l /shoremnt
```

```
cat /shoremnt/motd
rm /shoremnt/motd
ls -l /shoremnt
# results should indicate the same directory contents,
# although the syntax will vary

# at the prompt from the shore server, type
log all trace

# repeat the previous sequence of commands at the shell prompt
cp /etc/motd /shoremnt
ls -l /shoremnt
cat /shoremnt/motd
rm /shoremnt/motd
ls -l /shoremnt

# You should see a trace of the NFS operations performed by the server
# emulating a Unix file system.
# To turn off the tracing, exit from the Shore server and restart it,
# or type the following at the server prompt.
log all error
```

After you're done, unmount the Shore file system (again, as super-user).

```
su
$SHORE/bin/sumount /shoremnt
exit
ls -l /shoremnt
# result should be 'total 0'
```

For Linux, there is no special Shore version of umount.

```
su
umount /shoremnt
exit
```

You may ignore the message

```
Cannot MOUNTPROG RPC: RPC: Program not registered
```

The document Running a Shore Server gives more information on NFS-Mounting the Shore file system.

6.4 Compiling and Running a Value-Added Server

If you plan on writing your own value-added server (VAS) using Shore Storage Manager (SSM) you should perform the following tests on the SSM installation. This test uses the "hello world" example shipped in the documentation release. See `examples/vas/hello/README` for details about the program. For other VAS examples, see the file `examples/vas/README`.

The first thing to do is copy the program to a new directory.

```
mkdir hello
cp $SHROOT/examples/vas/hello/* hello
cd hello
make
```

Before running the `hello` program, copy `exampleconfig` to `config` and replace `DISKRW` to the pathname of the installed shore `diskrw` program. If you have set the shell variable SHORE correctly, you can yank this entire command with your mouse. You will also need to make a directory for holding the log. The log directory used is set by the `sm_logdir` option in `exampleconfig`. Running `hello` will leave log files in `./log.hello` and a storage device file called `./device.hello`. These can be removed when you are done.

```
sed -e "s,DISKRW,$SHORE/bin/diskrw," exampleconfig > config
mkdir log.hello
./hello
rm -r log.hello device.hello
```

You should see something like this.

```
processing configuration options ...
forking ...
waiting ...
Starting SSM and performing recovery ...
Formatting and mounting device: ./device.hello...
Creating a new volume on the device ...
Creating a file for holding the hello record ...
Creating the hello record ...
Pinning the hello record for printing ...

Hello World!

Shutting down SSM ...
Finished!
```

The document *Writing Value-Added Servers with the Shore Storage Manager* gives information about writing value-added servers using `$SHROOT/examples/vas/grid` as an example.