Installing SM

1 Installing SM

First a disclaimer: SM is copyright © 1987, 1989, 1990, 1991, 1992, 1993 Robert Lupton and Patricia Monger. This programme is not public domain, except where specifically stated to the contrary, either in the code, or in the manual. If you have a legally acquired copy you may use it on any computer "on the same site", but you may *not* give it away or sell it.

SM is provided 'as is' with no warranty, and we are not responsible for any losses resulting from its use.

Next a note about compiler-compilers. The SM command interpreter is written in YACC ('Yet Another Compiler-Compiler'), which is a Unix utility which converts a set of rules into compilable code. Unix is of course not in the public domain, but fortunately there is a version called 'Bison' written by the the Free Software Corporation, the people who wrote Gnu-emacs. As far as I know, it doesn't matter which you use if you have a choice. So that is what the cryptic comments about Yacc and Bison are about.

These notes assume that you are setting up SM on a Berkeley Unix system (in which we include Sun O/S). If this isn't true don't panic, there are notes on other operating systems at the end of this file. If these are not sufficient then you will have to get a bit more, er, innovative. Look up 'porting to new machines' in the index of the manual if you want help with getting SM running on a new system. If you need more help send us mail; we will try to provide as much guidance as the constraints of time, distance, and email permit. At the least we would like to include your horror stories in the next release of this installation guide.

In the best of all possible worlds (which are actually quite common these days), all that should be involved in building SM is:

```
cd sm
set_opts
make
make install
```

If you haven't build SM before you may have to remove a copyright from the top of '`sm/src/options.h`'; if this is necessary, '`set_opts`' should tell you. If the make succeeds, all you have left to do is to read about '`.sm`' files.

If '`set_opts`' fails and you send us mail we'll try to make it a little better and more general in the next release. In the mean time, you'll have to read the rest of these installation notes.

To build SM you should use the provided Makefile – `make all` in the main directory. It might be wise to run `make empty` first, in case there are unsuitable object files on the distribution tape. As distributed we assume that you will be using Bison not Yacc as your compiler-compiler, but it should make no difference. If you prefer to use Yacc, edit the Makefile in the src directory to remove all of the references to Bison, and all should be well.

Before running `make` you must edit the file '`sm/src/options.h`' to suit your particular needs. This is where you specify which devices you want (you have an imagen printer? Here's where to tell SM about it). In addition, this file is used to describe peculiarities of your environment, for more details see the machine-specific sections below.

You probably won't know offhand how to set all the options in this file, but don't be alarmed by this. If you are not sure if you have (e.g.) `alloca` you can just ignore the mutterings about it in '`options.h`', but then when you make SM, you may get a message that this function isn't defined: In this case you'll have to edit '`options.h`' and remake everything. A test programme like:

```
int
main()
{
   alloca(123);
   ftruncate("/dev/null",0);
   memcpy(0,"foo",3);
   select(1,0,0,0);
   strchr("bar",'r');

   return(0);
}
```

should be enough to tell you what you don't have.

Compile-time errors are likely to be due to ANSI or non-ANSI compliance of your C compiler. The best way to approach this problem is to look at the line of code where the compiler error occurs and see if it depends on any of the preprocessor definitions that are set up in '`options.h`'; if so, edit '`options.h`' accordingly and try compiling again.

You are now ready to start SM for the first time, so execute the file `sm`. Try `load demos square 20` after selecting a graphics device with the `device` command, e.g. `device tek4010`. It would be as well to run `LIST DEVICE` or to look at the graphcap file and see which entries are useful to your users as they are sure to ask. If you can't decide which to use, experiment. The same thing goes for terminals. Here the only problem is with cursor motions, some terminals work better with cursor motions disabled (e.g. `TERMTYPE selanar -21` for some flavours of graphons). Before unleashing SM on the general public, you might want to edit the macro `startup` in the file `default` in the macro directory to add local comments/warnings.

After successfully making and testing SM you will probably want to install it in some standard place. To do this `make install` after editing the Makefile to decide where the files should go (if you used `set_opts`, there should be no need for any edits).

If you use info (and TeXinfo), you may want to install the info files; this can be done with the `make installinfo` command. Info files can be read with a variety of programmes, for example `info` and `xinfo` (from the GNU project), `tkinfo` (available from `messier.EECS.Berkeley.EDU:/pub/misc/tkinfo-`

`0.7-beta.tar.Z`), or `xmosaic` (xmosaic, which is an HTML browser, needs a special gateway to read TeXinfo files). The SM macro `info` assumes that your system has `info`.

if you want to install it instructions are given in '`sm/info/README`'. As is noted in the `README`, we assume that you have at least version 3.0 of makeinfo

2 Environment (*.sm*) Files

Each user needs a *.sm* file, but starting with version 2.1.1 they no longer need a private one (although most users will probably want one of their own). There is a search path for '`.sm`' files which is set in the top-level Makefile. The default value instructs SM to search the current, then the home, and then a system directory, usually the one where files such as graphcap reside (called `$(DESTETC)` in the Makefile).

Make a copy of the relevant '`.sm`' file from the sm directory ('`vms.sm`' for vms machines, '`unix.sm`' for unix machines), edit it to suit your environment, and put it in `$(DESTETC)`. The file name should be changed to '`.sm`' (i.e. drop off the prefix 'vms' or 'unix') or else SM will not recognize it. (Actually, you can specify the file to be used on the command line - see manual for details). Despite what I said about these notes referring to setting up SM on a unix system, I have reproduced the vms version here, with an explanation of each line:

`device hirez`
: This is used by the `hcopy` and `hmacro` macros to reset the device when you do a hardcopy plot. If you want to know more see section "Hardcopy" in *The SM Manual*. It is also the default device defined when you start SM. It's changed by the `dev` macro. Optional.

`edit $disk:[SM_dir]maps.dat`
: This files gives the key mapping for the command line editor. The SM manager should change this line to point to the main SM directory. You may want to change this to '`vms_maps.dat`' to get DCL-style editing (see the SM manual for a description of the differences). Keymaps can also be set with the `READ EDIT` command. Optional.

`filecap $disk:[SM_dir]filecap`
: This file describes the type of 2-dimensional files that SM can read. The SM manager should change this line to point to the main SM directory.

`file_type C`
: Specifies the type of 2-D files to be read. Sets the SM `file_type` variable. Optional

`fonts $disk:[SM_dir]fonts.bin`
: This is the file that defines the SM fonts. You should change this line to point to the main SM directory

`graphcap $disk:[SM_dir]graphcap`
: This is the file used by stdgraph to describe the output graphics devices You should change this line to point to the correct place, initially the main SM directory

`help $disk:[SM_dir.help]`
: This defines the location of the help files. The SM manager should change this line to point to the SM help subdirectory on your machine

`history 80`

Specifies the number of commands to remember, and also the maximum number that can be put on the playback buffer. If omitted, the default value is 80

`history_file .smhist`

If this line is in the user's '`.sm`' file, SM will keep a record of all the commands executed while running SM in the file defined here (e.g. in '`.smhist.`') This file is saved from one SM session to the next

`macro $disk:[SM_dir.macro]`

This directory is the location of all currently defined SM macros. The SM manager should change this line to point to the main SM directory

`macro2 $disk:[rhl.SM]`

This line may be added to the '`.sm`' file by a user if s/he has a set of SM macros defined locally. Optional

`name Robert`

This line should be edited by the user. This name is the one SM will greet you with when it starts up. If you omit it (and SM can't find it out somehow), you'll be asked for it, and your reply will be appended to your '`.sm`' file. On Unix systems if you omit this SM will find your name in the password file.

`printer qms`

This defines a default laser printer device. This is used by the hcopy and hmacro macros to generate a hardcopy plot. See the macro definition for more detail. Sets the value of the SM variable printer. Optional

`prompt :` Set your SM prompt to `:` — a colon is the default if you omit this entry.

`prompt2 >>`

Set your secondary prompt tp `>>`. This is the prompt used when you have typed an incomplete command, and SM is waiting for the rest.

`save_file sm.sav`

The default file to `SAVE` your SM environment to. See the SAVE command for details. Optional

`TeX_strings 1`

Tell SM to use TeX-style syntax in strings. This sets the variable `TeX_strings`; for more details see the manual.

`temp_dir disk$scratch:[rhl]`

This is where plot files will be written, prior to be printed to a laser printer. If you have disk quota problems, you may want to use this. By default, the plot file will go to '`sys$login`'. (On unix machines, you probably want `/tmp/`). Optional

`term hirez`

this is just to tell SM how to do command line editing. This variable overrides the value specified as the logical (environment) variable `TERM`. Can be set with the `TERMTYPE` command. Optional

`termcap $disk:[SM_dir]termcap`

This is the file used to describe terminal to the history/command line editor. Under unix, you should probably omit this. If it isn't defined, the file '`/etc/termcap`' is used (which is meaningless to VMS). If the logical (environment) variable `TERMCAP` is defined, this takes preference over this entry. The SM manager should change this line to point to the main SM directory

The final thing that will need editing by the SM manager is the graphcap file. The entries for the hardcopy devices have a string `SY=<print command>` the print command is of course site dependent, and system dependent, and will need to be edited to give the correct name of the print queue. Or, if you do not have a hardcopy device directly attached to your machine, you will want to change this to say (in vms)

```
:SY=write sys$output "File is $F.":
```

and then generate the hardcopy by manually disposing the file to the printer

(`$F` is the name of the plot file, which SM generates by using the name given in the `OF` entry of the graphcap file, and generating a unique name if the `OF` entry ends in `XXXXXX,` and then prepending the name of the directory (this will be '`sys$login`' - in vms - if the user does not have a `temp_dir` entry in her (his) '`.sm`' file, and will be the directory defined by `temp_dir` otherwise))

3 Operating System or Hardware Specifics

Due to various peculiarities and deficiencies (or sometimes excessive cleverness) of different operating systems some need special treatment.

3.1 Alpha Running OSF/1

SM seems to build without problem on alphas running OSF/1. Unfortunately, bison does not, so if you need to rebuild control.c from control.y you will need to obtain a more recent version of bison than the one that we supply; version 1.18 is known to work. It may be found on any of the GNU source sites (for example prep.ai.mit.edu in /pub/gnu).

If you don't expect to change the SM grammar (as is probably the case) then you can avoid this problem by saving copies of control.c and control.h from the distribution, and then when make fails to build new ones, copying the saved ones to the src directory (do not move them! Their modification dates must be updated).

The easiest way to use your new bison is to go to the src directory and type

```
make rm -f control.c ; YACC="bison -y" control.c
```

An alternative would be to change the definition of YACC directly in the Makefile and rebuild (remembering to remove the old control.c).

3.2 Berkeley Unix

SM should build smoothly on Berkeley systems as most of its recent development has been on such machines.

3.3 Alliant running Concentrix

This information is based on reports from an Alliant FX/80 running Concentrix 5.6.00.

The compiler options in the Makefile should be set to

```
CC = fxc
CFLAGS = -Og -nc -g -Isrc
```

SM has been built with version 2.2 of the FX/C C compiler. It is not clear if there would be any significant difficulty with an earlier version.

3.4 Convex O/S

Define `_POSIX_SOURCE` in '`options.h`', and make sure to compile with

```
cc -D__STDC__=1 -Dunix
```

(or add the definition of `__STDC__` and `unix` to `CFLAGS` in '`sm/Makefile`', or carefully define the proper things in '`options.h`').

If you are using the X11 driver there is a problem with '`/usr/include/strings.h`', which is included by '`X11/Xos.h`'. The easiest workaround is to define `SYSV` when compiling x11.c (use `-DSYSV` on cc's command line, or edit '`x11.c`' to `#define SYSV`). It is probably OK to simply add `-DSYSV` when you add `-D__STDC__=1` to `CFLAGS`).

3.5 Hewlett Packard

Compile with

```
cc -Dhp
```

We have had reports that the ANSI version of the compiler gives problems; if you use it successfully please let us know.

3.6 Linux

SM should build without modification on PCs running linux; the only available display device is x11 (although RHL has about 80% of an svgalib driver written, it is not currently shipping with SM. Send mail if you want to help).

The only problem that we are aware of is that some older linux installations used to require an explicit `-lc` as part of the linker command required to build e.g. `makeyyl`. This is *not* detected by `set_opts`, as we believe that it's caused by an incorrect installation of gcc, and because it seems to have been fixed in recent linux releases.

3.7 IBM PC (or compatible) running Win95/98/NT/XP/2000/etc

SM should build without modification on PCs running a MS Windows 32-bit operating system, if you first install the free unix environment for Windows, cygwin, now part of RedHat (cf www.cygwin.com). You will need the development environment, and also the X11 run-time and development environment.

The only driver that is supported, apart from the stdgraph drivers, is for X11, so you need to have an X11 server for Windows on your PC. There are many of these available, as freeware, shareware, or commercial products.

3.8 IBM PC (or compatible) running DOS (or Win 3.x)

SM can be built on PC's using Borland's C++, version 3.0 or later. For these versions, the '`.sm`' file has been renamed to '`sm.rc`' due to PC file system restrictions (and is distributed as '`sm_rc`' so as not to confuse unix users).

We have seen problems with the optimizer in some releases of Borland's compiler. The Makefile.dos as distributed uses `-O1`. We also use `-ml`. If you use some other combination of switches, and get better results, please let us know.

If you don't have a working version of `bison` (you can get the executable from various places) and don't feel like building one, make sure that the files '`src/control.c`' and '`src/control.h`' are *younger* than '`src/control.y`'. If this is true, `bison` is not required.

You'll have to edit '`options.h`' to define which devices you want to use (If anyone wants to write a script like set_opts, please send us a copy!). The graphics devices available are `bgi`, the Borland Graphics Interface, and `win` or `msw`, the Microsoft Windows API. These two versions produce files with the same name ('`SM.EXE`', '`*.OBJ`') but the object files are incompatible. If you want to build both versions, make sure you clean up between the makes (with `make empty`).

Currently (due to lack of time and interest) the author of the PC port does not support the Plain version; he only checks that the Windows version compiles and runs.

You should then be able to say `make -fMakefile.dos` in the top level directory, or `make -fMakefile.dos -DWIN` if you want the windows version. Note that you must specify twice whether you want the plain DOS or the Windows version; this is due to the general structure of sm and will not be changed (uhm, fixed).

The drivers use some svga code from the net. It's in the directory '`sm/dos`', and you'll have to move the files to the proper places before you can build SM (for example, to build the bgi driver, you'd typically copy the '`.h`' files to '`/borlandc/include`' and the '`.bgi`' to '`/borlandc/bgi`'). There should be no problem using more recent versions, if you can find them.

When you run SM with `device bgi`, the graphics drivers, the '`.bgi`' files, are assumed to be in a directory given by the DOS `BGIPATH` environment variable; alternatively you can specify a `bgipath` variable in your '`sm.rc`' file.

If you use devices that need the stand-alone programme `rasterise` you'll find that SM doesn't leave enough memory to run both at once. It would be possible to rewrite rasterise to use much less memory, but as this isn't a real concern on unix boxes this is unlikely to happen soon. You'll have to exit SM (maybe using the `save` command to save your environment) before you can submit your plots. One way to remember to do this is to put a line

```
BIN = echo "Remember to run the command: "
```

in your private graphcap file.

3.9 IBM PC (or compatible) running OS/2

I don't know much about this port, so comments would be welcome. The build is done with the set of Makefiles called '`Makefile.os2`'. The device name is `os2pm`, and the terminal type `os2pc`.

SM can be built on IBM PCs (or compatibles) running OS/2 2.X using Borland's C++ compiler for OS/2 version 1.5. (See below if you have version 1.0) On screen graphics output is provided by a separate Presentation Manager program, sm_pmdev.exe, which is spawned by the SM command line session.

Installation Instructions:

1. Edit sm\src\options.h to define OS2PM as an output device.
2. From the sm\ directory build everything using the command

   ```
   make -fmakefile.os2 all
   ```

   This should build everything and place the two executables, sm.exe and sm_pmdev.exe in the sm\src\ directory. If you will be using a raster device you must also make rasterise.exe with the command

   ```
   make -fmakefile.os2 rasterise.exe
   ```

   This will build rasterise.exe and place in in the sm\ directory.
3. Define two environment variables HOME and SMPATH to your config.sys file, for example

   ```
   SET HOME=D:\efgh\
   ```

   Makes D:\efgh\ your HOME directory

   ```
   SET SMPATH=. ~ E:\abcd\ D:\
   ```

   In this example SM will look for the .sm file first in the current directory, then in HOME, then E:\abcd\ and last in D:\

   (Remember, after placing the SET statemanet in the config.sys file, the system must be re-booted for the definition to take effect)
4. Rename the file sm\os2.sm to .sm, and place it in a a directory consistent with your choice in 3). Edit the various entries to point to the appropriate directories in your system - note the use of forward slashes in all entries except `temp_dir`.
5. Copy `sm.exe`, `sm_pmdev.exe`, and `rasterise.exe` (if you want to use a raster printer) to a directory included in the system's PATH variable.

If all is well you are ready to run. After starting SM the statement device os2pm (or device pm) should produce a Presentation Manger graphics window. By default retain mode is enabled for this window so it can be resized by grabbing an edge or corner, and the drawing will scale accordingly. When minimized the graphics window will appear as an icon in the Minimized Window Viewer. Use help os2pm for futher information on this driver.

Since the directory separator in OS\2 is interpreted as an special character in the graphcap file, specifying directories explicitly can lead to problems. A convenient way to enable the raster device is to use a :SY entry in graphcap something like

```
:SY= rasterise -r $0 $F $F.tmp \n PRINT /B $F.tmp \n del $F.tmp:
```

and be sure your .sm file has a line like

```
temp_dir = d:\temp\
```

3.9.1 Borland C++ v1.0 Compatibility

Currently the only incompatability with V1.0 of the Borland C++ for OS/2 compiler arises from a Borland error in the declaration of signal() function. This can be remedied by adding the lines

```
#if (defined(__BORLANDC__ ) && defined(__OS2__))
typedef void (__cdecl *pCatch )(int);
#define signal(a,b) signal((a),(pCatch)(b))
#define SIGBUS  SIGSEGV
#endif
```

after the #include statements in sm\src\sm_main.c .

3.10 IBM RISC/6000

Compile using `CC=c89 -Drs6000` in '`sm/Makefile`'. In versions of AIX prior to 3.2, you may have to add `-bnodelcsect` to the `LDFLAGS` passed to bison in '`sm/src/Makefile`', i.e.

```
$(MAKE) CC="$(CC)" CFLAGS="-c" LDFLAGS="-s -bnodelcsect" bison
```

(This is a required by a bug in their compiler)

The RS/6000 supports Silicon Graphic's GL graphics as well as X11.

If you want to call SM from fortran you must explicitly link with the C runtime library (add a `-lc` to the end of the linker flags, which are called `LDFLAGS` in the sample Makefile in the '`callable`' directory), or else the brain-dead system will find its own function `getenv` not the standard C function with that name.

3.11 Silicon Graphics

Compile using `CC=cc -ansiposix -Dsgi` in '`sm/Makefile`'. If you want SM to use GL routines for graphics, define SGI in '`options.h`'; X11 will work just as well (or you could use both). As SGI doesn't provide a termcap file you'll have to use ours; the default '`.sm`' file provided should include it.

There is a bug in the mathematics library (bugid 124930); the workaround is to use -O when compiling SM. If you want to know if your system has the bug, start SM and say:

```
set x=0,10 set x=10+0*x set y=lg(x) print { x y }
```

If you have the bug, it should be pretty clear!

3.12 SunOS (including Solaris)

Sun has traditionally had a BSD-unix operating system, but with Solaris it has switched to the System V camp.

If you are running Openwindows and not the X Consortium X11 distribution, you will have to tell the makefiles where to find the X11 include files and libraries. Openwindows has them (by default - your system manager may have moved them) in '`/usr/openwin/include`' and '`/usr/openwin/lib`', respectively. So you need to add `-I/usr/openwin/include` to `CFLAGS`, and `-L/usr/openwin/lib` to `LDFLAGS`. You may also have to define the environmental variable `LD_LIBRARY_PATH` to be `/usr/openwin/lib`.

When compiling the sunview device (an obsolescent sun windowing system) you may get warnings if you are using the gcc C compiler because sun's system headers are not ansi compliant. You should ignore them, the device will work perfectly well.

3.13 System V Unix

The default configuration is for a BSD-derived unix. To compile for System V you should define SYS_V in the options.h file. Because SyS V has no ranlib command (what? ranlib?), you should edit the Makefile in sm/src to define RANLIB as "sh /dev/null". On (some?) HP machines, if you want to use gcc you may need to compile get1char.c with the -traditional flag, as the system header files are broken (or you can run the gcc script `fixincludes`).

At least some SyS_V machines don't have the system call `select`. If this describes you don't panic. Define `NO_SELECT` in '`options.h`' and proceed. This has some minor side effects that could lead to problems with refreshing display windows under (for example) X Windows. There are workarounds (e.g. you will have to make sure that the X-Windows server has backing store), if you have trouble send us mail.

Many Sys_V machines don't have a termcap file. We provide a small one for you, but if you have weird terminals you may have to figure out how to translate terminfo back to termcap; when you have succeeded (or given up) please send us mail. An alternative might be to look at some system (such as a sun) with an extensive termcap file.

3.14 Ultrix

SM seems to build without problem under Ultrix.

3.15 VMS

Most (but not all) of this is for those poor souls without MMS.

To make SM first set your default directory to '`[sm]`'. You should look at (and maybe edit) the file '`[sm.src]options.h`' to choose in your favourite device drivers. If one of those is vaxuis, you will also have to edit '`[sm.src]sm.opt`'. The comments in '`options.h`' tell you how to edit the linker options file.

Then if you are running on an old-style VMS system, execute the command file '`compile.com`', i.e.

```
$ @compile
```

This will compile all the routines in the subdirectories, including making '`bison.exe`' and the font table. You can also just make parts of SM, e.g.

```
@compile bison
@compile fonts
```

See the comments at the top of '`compile.com`' for a list

If you are using an Alpha and OpenVMS, the first argument to '`compile.com`' should be the word `alpha`, viz

```
@compile alpha
```

or

```
@compile alpha fonts
```

If you have MMS, you have to edit the '`descrip.mms`' files in the top level directory and in '`/src`' to choose the proper compiler flags, and to set the name of the linker options file. We hope that we have provided enough comments to make this easy.

Once these changes have been made, you can simply say

```
$ mms all
```

in the '`[sm]`' directory, and make the program that way (after editing '`[sm.src]options.h`' as before)

You may have quota problems when you try to run SM. The following is a report from a VMS site on what quotas were found sufficient for running SM (n.b. - may not be up to date):

Subprocess limit > 2
Buffered I/O byte count quota > 5000
Direct I/O limit 18
Buffered I/O limit 18
AST limit 22
Paging file quota largish, we reckon > 10k
Default page fault cluster 64

There is a command file to allow you to run SM as a kept command, called '`kept_SM.com`'. To use it, define a foreign command that runs SM (e.g. `_sm :== $ disk:[sm.src]sm.exe`), and say `@kept_SM _sm`. Then you can suspend SM with the `CONTROL-Z` key (or whatever you decided to use – see the section on the editor), and restart it with the same command.

You will also need to define a foreign command to run '`rasterise.exe`' if you have any raster devices.

Because SM uses the '`termcap`' file invented by Unix, you'll need a copy yourselves. There is a small one in the main source directory and you'll have to make sure that either there is a termcap entry in the '`.sm`' file pointing to it (better), or a logical variable.

4 How to Generate Hardcopy or Online Manuals

There are a number of files in the info directory that you might find useful, namely

'`install.txi`'
: Installation notes

'`notes.txi`'
: Release notes

'`sm.txi`' The full SM manual

'`tutorial.txi`'
: The SM tutorial

All of these are written in TEXinfo, which means that they can either be printed with TEX, or processed into a hypertext system called info. If you are in Europe (in which I include the UK) you may want to use A4 paper; to do this un-comment the line `@afourpaper` found near the top of each file (i.e. change `"@c @afourpaper"` to `"@afourpaper"`).

To print them using TEX you can just treat them like any other TEX file (despite their weird suffix and format), or simply use the provided makefile (e.g. `make sm.dvi`). If you want to do it by hand, read on.

If you have to build the full SM manual with indexes and cross references from scratch, you need to say (on a Unix system):

```
tex sm.txi
texindex sm.??
tex sm.txi
texindex sm.??
tex sm.txi
```

Under VMS, you must first define texindex as a command, viz.

```
$ texindex :== $[<smdir>.info]texindex.exe
```

and then you have to type out the file names explicitly, so you say

```
$ dir sm.%%
```

and then

```
$ texindex <file names you get from the dir command>
```

(The first run of TEX and texindex produces a first draft of the indexes, which change the page numbering a bit as one of the indexes appears in the middle of the appendix. The next run allows for this and produces the correct page numbering; the final TEX run produces the manual. If you omit the first tex + texindex iteration the page numbers of the last few appendices will be off by

one or so). You can build texindex with `cc -o texindex texindex.c`. You need to go to this much trouble only if you have changed the pagination, or if files like sm.tps are not present.

If you want to use info you should ensure that you have modern copies of the two program required, `makeinfo` and `info`, at least as recent as those that we provide on the SM distribution tape. If you didn't get a tape, they are available as texinfo.tar.Z in all the usual places; The last time that I checked it was on prep.ai.mit.edu as

```
/pub/gnu/texinfo-3.1.tar.gz
```

and, as the name suggests, this is a gzipped file, so if you need it pick up gzip at the same time. I know that version 1.55 of makeinfo works; if yours is older you should upgrade or be prepared to make some minor changes, e.g. change `--footnote-style` to `+footnote-style`.

You can (or we did) process the reference manual part of the manual into help files – see '`make_help`' for details. Or you can use the SM macro `info` to read the info node for any of SM's commands.

An alternative would be to convert the TeXinfo files to HTML. We provide a slightly-patched version of Lionel Cons's `texi2html` converter; he's been sent the patches so I hope that in the future a vanilla version will convert SM's manual. Part of the processing is done by the file '`texi2html.ext`', which must be present in either the current directory or the directory where `texi2html` is installed. If this is done, you should simply be able to say

```
texi2html -menu -inline_toc sm.txi
```

and generate '`sm.html`' automatically (along with sm_toc.'`html`' and '`sm_foot.html`'). Because of bugs in many readers (including `xmosaic`), you may need to add the option

```
-invisible " "
```

to make the index's hypertext links work correctly.

If you'd rather obtain the manual as a large number of little files (which most HTML browsers will digest more quickly), try the

```
-split_chapter
```

```
or
-split_node
```

option (don't use -inline_toc at the same time); you might want to do this in a different directory.

If you use -split_chapter you can set up a link directly to the Command Reference (it's sm_22.html as I write this).

5 What to do if you have Problems

The worst sort of bug is the sort that leads to a core dump/traceback map. If this ever happens to you, *please* note down carefully what you were doing. If you are running Unix, please use dbx to examine the core file, and use the `where` command; under VMS please note down the stack trace. If dbx refuses to cooperate, please use the script '`dump`' in the main directory, which uses adb to get at least some information. Then send the information to Robert Lupton or Patricia Monger. If you can repeat the problem, use `SAVE` to save your killer programme, and send it to one of us. Then we'll be able to `RESTORE`, `playback`, and see the problem for ourselves.[1] If you don't complain, bugs won't get squashed.

The next class of problems are things that you don't like, or things that you would like to be able to do. Incomprehensible manuals come under this heading too. Please send them to us as well, and be as specific as possible. Please don't just say 'the manual was awful' or or 'Reading the manual was like my first exposure to James Joyce', but rather, 'I found the manual hard to get into. More explanation of how to manipulate files of macros would have been useful, and more examples', or 'I took 3 weeks to find how to draw logarithmic axes'.

[1] monger@mcmaster.ca or rhl@astro.princeton.edu. If you don't like computer mail, send a letter to Robert Lupton, Princeton University Observatory, Peyton Hall, Ivy Lane, Princeton NJ 08544