GRASS 4.1 INSTALLATION GUIDE

ABSTRACT

This document explains how to install GRASS 4.1. **Please read the instructions completely before beginning. There are changes in the installation process between GRASS 4.0 and GRASS 4.1.**

April 30, 1993

GRASS 4.1 INSTALLATION GUIDE

1. SUPPORT

The Geographical Resource Analysis Support System (GRASS) is a public-domain product of the GRASS Inter-Agency Coordinating Committee. The 4.1 release is being coordinated by the Office of GRASS Integration (OGI) at the Construction Engineering Research Laboratories (CERL), Champaign, Illinois. GRASS is an integrated set of many programs designed to provide digitizing, image processing, map production, and geographical information system capabilities to the user. Authorship of the individual programs is noted in the *GRASS 4.1 User's Reference Manual.*

CERL maintains the GRASS Information Center at phone number 1-800-USA-CERL (or 217-352-6511), extension 220. Through this Center you may find out what firms and agencies currently distribute and support GRASS as well as which offer class instruction, data development, or related services.

2. TO COMPILE OR NOT TO COMPILE?

Your release tape may provide you with binary code compiled for a particular machine configuration. If this configuration (which is clearly marked on the medium's label) matches your machine set up, you will be able to run GRASS almost immediately after loading the files onto disk. The simple instructions that come with the distribution media should be followed and this entire document can safely be ignored for now.

These instructions are for sites that have received source code. Compilation of this software can be moderately to extremely difficult. You may attempt installation yourself or seek assistance through any of several commercial firms competent in GRASS installation and support. Refer to section 1 for finding assistance.

This installation document involves the compilation of computer source (human-readable) code into machine language. To be successful, you must be familiar with:

Your computer
UNIX
UNIX shell(s) (sh, csh)
An editor
A C language compiler
Makefiles
Tape handling and reading (using the **mt** & **cpio** commands)
Super-user operations

configurations:[1]

[1] See section 17, "Appendix," for notes regarding other computer configurations.

	COMPUTER SUN (4,386i)
Operating System	SunOS 4.1
C compiler version	any K&R compatible
Graphics software	Sunview, X-windows
Graphics Hardware	8-plane color

GRASS 4.1 has been compiled and runs on the following computer If you are using different configurations than those defined here, expect more than the usual share of difficulties. Even a ''new and improved'' compiler can cause problems. Installation of GRASS on systems using new (to GRASS) graphics monitors will experience the most difficult problems. The *GRASS 4.1 Programmer's Manual* is available to help the person responsible for porting GRASS to different hardware configurations.[2]

3. DATA IMPACTS

Existing GRASS data files (those created under previous versions of GRASS) can be used under GRASS 4.1 without conversion.

Please note that while there have been no significant changes to the vector, raster, site and imagery data files from 4.0 to 4.1, there is no guarantee that data created under 4.1 will be backwards compatible with 4.0.

Vector files and their support files as well as raster color files made under GRASS 4.1 (or 4.0) are not backwards compatible with GRASS 3.1 or earlier versions of GRASS.

4. PREPARATION

You must first decide which user will own the GRASS 4.1 files. In this document, we will assume that the user *grass* will own the GRASS 4.1 files.

GRASS 4.1 should be installed in new directories. DO NOT extract the 4.1 source code on top of any existing 4.0 source. If you already have a version of GRASS on your system and you want to install GRASS 4.1 in the same place, move the older version elsewhere.[3] Compiled binaries should go into a new place as well. OGI recommends that you keep the SRC and GISBASE directories separate for more flexibility; however, if you have a shortage of available disk space, you may elect to use the same directory for both.

---

[2] The *GRASS 4.1 Programmer's Manual* can be used as a reference for writing graphics drivers.

[3] A certain amount of savvy is required here. If the older version exists in the home directory for a user, and you rename the home directory, not only must you recreate the directory, but you must also restore the user's files as well (e.g., *.cshrc*, *.login*, *.mailrc*).

GRASS 4.1 source and programs can be installed anywhere in your computer system. This document will henceforth refer to the directory that houses the GRASS software as

$GIS

You should also be careful about file permissions when installing some of the GRASS programs; in particular, GRASS installs the programs **gmake4.1** and **grass4.1** in a commonly accessible directory (defined when running the **setup** program). In most cases, the GRASS installation will not be performed directly by the super-user (in fact, the installation is often performed by the user *grass*); in such cases, take care to ensure that the directory where **gmake4.1** and grass4.1 are installed has permissions allowing the user installing GRASS to copy these programs into this directory. (You can do this by allowing only members of a particular user group to write to the directory and then making the installing user a member of that group; you can also accomplish this by making the directory world-writable temporarily, i.e., until the GRASS installation is completed. Note that you must set up the permissions before starting the installation process because the **setup** program will not let the installer select a destination directory that is not writable.)

Similarly, GRASS 4.1 data can be installed anywhere in your computer system. At this time you must determine where you would like to load the sample data. This document will henceforth refer to this data directory as

$GISDBASE

You can use the UNIX shell variables to store the paths of the GRASS source directory and the GRASS data directory in the GIS and GISDBASE variables, respectively. For example, if you choose to place the software in */usr/grass4.1*, you might use one of the following commands:

```
For CSH    setenv GIS /usr/grass4.1
For SH     GIS= /usr/grass4.1; export GIS
```

Similarly, if you choose to place the data in */usr/grass.data* you might use one of the following commands:

```
For CSH    setenv GISDBASE /usr/grass.data
For SH     GISDBASE= /usr/grass.data; export GISDBASE
```

Create the software and data directories by issuing the following commands (you may have to **su** to *root* to create them – if you do, be sure to change the ownership to *grass*):

```
mkdir $GIS
mkdir $GISDBASE
chown grass $GIS $GISDBASE
```

Note that the GIS and GISDBASE shell variables must be set as defined above.

Then log in as *grass,* set the GIS and GISDBASE variables as above, and change to the directory $GIS:

cd $GIS

5. DISTRIBUTION TAPE CONTENTS

The distribution media contain one or more files. (Check your label for verification.) The following table shows the contents of these files:

File Number	Contents	Size	
		Uncompiled	Compiled
1.	GRASS Source code	49.7 Mb	< 97.5 Mb
2.	Spearfish database	17.2 Mb	N/A
3.	Imagery database	30.4 Mb	N/A
4.	World database	2.3 Mb	N/A
5.	Related Source code	12.9 Mb	< 18 Mb

6. EXTRACTING SOURCE CODE AND SAMPLE DATABASE

Mount the distribution tape and then change to the directory $GIS. Follow the instructions in the table that represents your machine.[4] It is recommended that you extract the files as the user *grass,* not as *root.* If you do it as *root,* the files may be owned by a random or unknown user on your system.

The five files can be extracted from the distribution tape individually at any time. For example, it may be appropriate to extract the source code one day for compilation, extract the Spearfish sample database another day, and then unload the sample imagery data another day.

The device names used in the tables (e.g., ‘‘/dev/rmt0’’) are only examples; you must use the correct device names for your hardware configuration. Furthermore, the rewindable and non-rewindable names for the tape device must be used in the appropriate steps of loading the tape (e.g., you must substitute your rewindable tape device name wherever ‘‘/dev/rmt0’’ is given in the instructions, and the non-rewindable device name wherever ‘‘/dev/nrmt0’’ is listed).[5]

---

[4] The files are stored on the tape with *relative* path names so they will be extracted into and below the current directory. **Make sure you are in $GIS or $GISDBASE prior to issuing the commands necessary to extract the source code or sample database respectively.**

[5] For every physical tape drive, UNIX provides two device names, corresponding to a rewinding device and a non-rewinding device. When the rewinding device is used in an operation, UNIX will rewind the tape once the operation is completed; when the non-rewinding device is used, the tape will be left at the position reached at the end of the operation. As a result, two names can be used (with different meanings) to refer to

For both of the SUN tape formats as well as for the Intergraph formats, note that if you intend to unload all five files in the order that they are present on the tape, then you should be able to unload them without having to rewind and position the tape for each step – this will shorten the unloading process considerably.

SUN ½" tape	
load tape	
Unload Programs	
go to GRASS directory	cd $GIS
rewind tape	mt -f /dev/rmt0 rew
extract source	dd if= /dev/rmt0 ibs= 20480 |cpio -icdu
Unload Spearfish database	
go to DATA directory	cd $GISDBASE
rewind tape	mt -f /dev/rmt0 rew
position tape†	mt -f /dev/**n**rmt0 fsf 1
extract spearfish data	dd if= /dev/rmt0 ibs= 20480 |cpio -icdu
Unload Imagery database	
go to DATA directory	cd $GISDBASE
rewind tape	mt -f /dev/rmt0 rew
position tape†	mt -f /dev/**n**rmt0 fsf 2
extract imagery data	dd if= /dev/rmt0 ibs= 20480 |cpio -icdu
Unload World database	
go to DATA directory	cd $GISDBASE
rewind tape	mt -f /dev/rmt0 rew
position tape†	mt -f /dev/**n**rmt0 fsf 3
extract world data	dd if= /dev/rmt0 ibs= 20480 |cpio -icdu
Unload Related Source code	
go to GRASS directory	cd $GIS
rewind tape	mt -f /dev/rmt0 rew
position tape†	mt -f /dev/**n**rmt0 fsf 4
extract related src	dd if= /dev/rmt0 ibs= 20480 |cpio -icdu
†You must specify the **n** in the device name on these commands.	

the same physical tape drive.

SUN ¼" cartridge	
load tape	
Unload Programs	
go to GRASS directory rewind tape extract source	cd $GIS mt -f /dev/rst0 rew dd if= /dev/rst0 ibs= 65536 |cpio -icdu
Unload Spearfish database	
go to DATA directory rewind tape position tape† extract spearfish data	cd $GISDBASE mt -f /dev/rst0 rew mt -f /dev/**n**rst0 fsf 1 dd if= /dev/rst0 ibs= 65536 |cpio -icdu
Unload Imagery database	
go to DATA directory rewind tape position tape† extract imagery data	cd $GISDBASE mt -f /dev/rst0 rew mt -f /dev/**n**rst0 fsf 2 dd if= /dev/rst0 ibs= 65536 |cpio -icdu
Unload World database	
go to DATA directory rewind tape position tape† extract world data	cd $GISDBASE mt -f /dev/rst0 rew mt -f /dev/**n**rst0 fsf 3 dd if= /dev/rst0 ibs= 65536 |cpio -icdu
Unload Related Source code	
go to GRASS directory rewind tape position tape† extract related src	cd $GIS mt -f /dev/rst0 rew mt -f /dev/**n**rst0 fsf 4 dd if= /dev/rst0 ibs= 65536 |cpio -icdu
†You must specify the **n** in the device name on these commands.	

Intergraph ¼" cartridge and 8mm cartridge	
load tape	
Unload Programs	
go to GRASS directory rewind tape extract source	cd $GIS mt -f /dev/rmt/0m rew dd if= /dev/rmt/0m ibs= 20480 |cpio -icdu
Unload Spearfish database	
go to DATA directory rewind tape position tape† extract spearfish data	cd $GISDBASE mt -f /dev/rmt/0m rew mt -f /dev/rmt/0m**n** fsf 1 dd if= /dev/rmt/0m ibs= 20480 |cpio -icdu
Unload Imagery database	
go to DATA directory rewind tape position tape† extract imagery data	cd $GISDBASE mt -f /dev/rmt/0m rew mt -f /dev/rmt/0m**n** fsf 2 dd if= /dev/rmt/0m ibs= 20480 |cpio -icdu
Unload World database	
go to DATA directory rewind tape position tape† extract world data	cd $GISDBASE mt -f /dev/rmt/0m rew mt -f /dev/rmt/0m**n** fsf 3 dd if= /dev/rmt/0m ibs= 20480 |cpio -icdu
Unload Related Source code	
go to GRASS directory rewind tape position tape† extract related src	cd $GIS mt -f /dev/rmt/0m rew mt -f /dev/rmt/0m**n** fsf 4 dd if= /dev/rmt/0m ibs= 20480 |cpio -icdu
†You must specify the **n** in the device name on these commands.	

7. GRASS COMPILATION

If your site has several different machine architectures for which GRASS needs to be compiled, you have the option to mount a single, shared copy of the source code on the different machines (via NFS). You may then follow these instructions once for each machine to produce a separate set of object and executable files for each architecture; the compilation process will ensure that object and binary files for the different architectures are kept separate.

These instructions presume that you are familiar with UNIX, C, make, and shell scripting.

NOTE: These instructions and scripts have been used to compile GRASS on the following machine architecture:[6]

SUN 4 with SunOS 4.1.1

Please e-mail comments regarding the compiling of GRASS on other platforms or operating systems to:

[6] But see section 17 ''APPENDIX'' for information about other platforms.

grassbug@zorro.cecer.army.mil

At this point the GRASS source code has been loaded and is ready to be compiled. Most of the compilation steps take place within the directory $GIS/*src/CMD*. While logged in as *grass*, the owner of the GRASS code, do the following:

```
cd $GIS/src/CMD
ls
```

This directory contains scripts and files used to compile GRASS. By running scripts and changing lists of programs, you generate GRASS binaries for your system. The following is a partial list of the contents of this directory:

VERSION
Current version number and date of the GRASS release

generic/
System-independent files needed by the compilation process

GISGEN.sh	shell script that controls the entire compilation process
MAKELINKS.sh	shell script used to establish links to the user executable commands
gmake.sh	shell script that does compilations
make.head	make variables
make.tail	some additional make rules

head/
This directory is empty on the distribution tape, but will contain header file(s) for this site. Header files are created by running the *utils/setup* command.

lists/
Lists of programs to be compiled:

GRASS	standard GRASS programs
local	site-specific GRASS programs (you will create this file)
$ARCH	architecture-dependent GRASS programs (you will create this file)
local.example	sample list of optional programs and drivers that can be compiled locally

next_step/
This directory is empty on the distribution tape. It will contain files that track how far along the compilation process is.

utils/
Contains the *setup* script and its related scripts and files.

7.1. COMPILATION STEPS OVERVIEW

(1) Generate files that contain site- and machine-specific information for the **make** program.

(2) Please check the Appendix for any alterations that may be required for your platforms.

(3) Edit files containing lists of local and machine-architecture-specific programs to be compiled (generally printer, digitizer, and graphics drivers).

(4) Run the GRASS compilation script.

(5) Run the GRASS program linking script.

(6) Edit device driver configuration files.

(7) Compile GRASS contributed programs.

(8) Compile GRASS related and hybrid programs.

7.2. COMPILATION SUPPORT FILES

Each machine and site needs to have GRASS compiled in ways that specify different:

- compilation and load flags
- system libraries
- installation directories
- default databases and locations

The shell script *utils/setup* assists you in defining many of the compilation options and definitions that will become part of every compile-time generated makefile. The *setup* script creates four files:

< header>
: A file containing system-dependent information used during compilation. The *setup* script will ask you for a name for this < header> file. It will be created in the *head/* directory. We suggest using the name of the architecture, similar to $ARCH.

gmake4.1
: This shell script does compilation (using the UNIX **make** command). It will use the < header> information during compilation. The script is placed into a directory (called UNIX_BIN) that you specify during the setup. This directory must be part of, or added to, your shell's PATH variable.

GISGEN.< header>
: This script compiles all of GRASS. It is created in the $GIS/*CMD* directory.

MAKELINKS.< header>
: This script links all user executables to a common program called **front.end**.

NOTE: For brevity, this document will refer to *gmake4.1* as *gmake*, *GISGEN.*< header> as *GISGEN*, and *MAKELINKS.*< header> as *MAKELINKS.*

NOTE: *GISGEN* assumes **/etc/mknod** is the command for making fifos. This is not true for all machines. Installers should modify *generic/GISGEN.sh* to replace the **/etc/mknod/** command with the one correct for their **/bin/mknod**).

7.3. COMPILATION SETUP

Run the *setup* script[7] and answer the questions it asks:

```
sh utils/setup
```

When *setup* is complete, it will ask you for the name of a file to store the information it gleaned about your system. This document refers to this file as < header> . You are encouraged to use the same name as $ARCH (the architecture name).

Examine the newly created < header> file in *head/* to make sure things are OK. A brief description for each defined variable follows.[8]

CC	The name of the C compiler.
ARCH	Name identifying the architecture of the machine on which you are compiling GRASS.
GISBASE	Directory where compiled GRASS will be installed.
UNIX_BIN	Local directory where the GRASS *grass4.1* entry program and *gmake4.1* compilation script will be installed. This directory should be writable by the user building GRASS.
DEFAULT_DATABASE	Directory where local GRASS databases are contained.
DEFAULT_LOCATION	GRASS database that first-time users get as a default.
COMPILE_FLAGS	Compilation flags.
LDFLAGS	Load flags.
TERMLIB	System library supporting low-level cursor movement.
CURSES	System library that supports screen cursor control.
MATHLIB	System math library.
LIBRULE	Method for archiving and randomizing libraries.
USE_TERMIO	Flag to use the termio library if available.
USE_MTIO	Flag to use the mtio library if available.
DIGITFLAGS	Flags to set owner and priority of the v.digit program.
XCFLAGS	Flags for X11 compilation.
XLDFLAGS	Loader flags for X11 programs.

[7] If you are using Control Data EP/IX (or RiscOS), please see the notes regarding Control Data Hardware in section 17, "Appendix."

[8] See section 17 ''APPENDIX'' for sample < header> files that have been created at CERL for sun4 and other machines.

XINCPATH	Directory for X11 include files.
XMINCPATH	Directory for Motif include files.
XLIBPATH	Directory for X11 library.
XTLIBPATH	Directory for Xt library.
XMLIBPATH	Directory for Motif libraries.
XEXTRALIBS	Platform specific X11 libraries.

7.4. EDIT LISTS

Next, you must edit files containing lists of site- and machine-specific programs. The directory *lists/* contains files that list the directories to be compiled. Directory names are relative to the $GIS directory. The file *lists/GRASS* lists all basic GRASS programs that get compiled for each machine architecture at every site. The files *lists/local* and *lists/*$ARCH, which the user can create, control site-specific and architecture-specific compilation, respectively.

$ARCH is the architecture name you approved while running the *utils/setup* script. You can determine this by running:

```
gmake -sh |grep ARCH
```

A *lists/*$ARCH file may not exist in the distribution, but you are free to create it to add names of programs you want compiled specifically for this architecture. This architecture-specific list allows NFS-linked source code to compile a set of programs for machines which have the same architecture.

Similarly, there may not be a *lists/local* file, but you are free to create this as well. The programs in this list will compile for all machines at your local site regardless of their architecture.

The *local.example* file contains a list of display, paint, and graphics drivers provided in the source; you should uncomment the drivers you plan to use. The file *GRASS.X* contains the list of Xgrass programs.

If you want any of these items compiled, you must add them either to the *lists/local* or the *lists/*$ARCH file. You are encouraged to put the graphics drivers in the appropriate *lists/*$ARCH file.

All lists may contain comment lines, indicated by a ‘‘# ’’ as the first character in the line.

7.5. XDRIVER

If you are compiling the XDRIVER under Openwindows on a SUN machine,[9] or if you have X11 release 3, you may have to edit the file

[9] Or on other platforms.

$GIS/src/display/devices/XDRIVER/XDRIVER/Gmakefile

and follow the instructions therein.

7.6. COMPILE GRASS

The script *GISGEN*[10] drives the compilation process. If all goes well, you will be able simply to enter the command **GISGEN** and wait. The compilation process takes from about four hours on the faster workstations to about twelve hours on the slower workstations.

It is recommended that you save *GISGEN* output in a file that you can later review for compilation warning messages. The following command runs *GISGEN*, sending the output both to the screen and to a file:

```
sh GISGEN |& tee /tmp/GISGEN.out   # csh (not sh)
```

For use in a Bourne shell, the output can be captured with

```
sh GISGEN 2> &1 |tee /tmp/GISGEN.out
```

GISGEN collects all of the directory names to be compiled from *lists/GRASS*, *lists/*$ARCH, and *lists/local* and begins running **gmake** in each directory in succession. The screen output is a collection of messages from *GISGEN* and from the UNIX **make** program. A failure at any step will halt compilation. Upon encountering a failure, you might do one of the following things:

1. Fix the compilation problem by modifying code in the directory that failed. After modification, return to this directory and re-run *GISGEN*. Compilation will resume at the failed directory and continue down the list of directories if successful.
2. Restart *GISGEN*. If the failure requires modifications to code already compiled, or the compilation options you set in step 1, you must remove *next_step/*$ARCH (or *next_step/next_step* if an architecture name was not specified when running the *setup* script). You may then re-run *GISGEN*.
3. Skip the failed directory and resume compilation with the next directory in the list. Simply run

   ```
   sh GISGEN -skip
   ```

   (You might want to capture *GISGEN* output into a file as you did earlier.)

   When complete, *GISGEN* will put the word DONE into the *next_step* file and will print the phrase ‘‘DONE generating GIS binary code’’ on the screen. (You may wish to review *GISGEN.out* to view any error messages.)

---

[10] The *utils/setup* program creates a file called *GISGEN.*< header> . We will refer to this file simply as *GISGEN*, but you will need to remember it as *GISGEN.*< header> .

7.7. LINK GRASS PROGRAMS

GISGEN directs a compilation process that places the GRASS programs in directories not directly accessible to the user community. Most user commands are actually links to a single program called *front.end*. Links to this program must be made for every actual GRASS program. This is done **after** *GISGEN* is finished. To make (or re-make) links for all user programs, run the script *MAKELINKS* (i.e., *MAKELINKS.*< header>).

7.8. COMPILE OTHER PROGRAMS

GRASS programs come in five flavors:

MAIN

The main programs are those under $GIS/*src*. These programs are compiled automatically by *GISGEN*. They have been in GRASS for at least one release, have been tested, and are reliable programs.

ALPHA

The alpha programs are those under $GIS/*src.alpha*. These programs are also compiled automatically by *GISGEN*. These programs have been compiled and given some testing but have not been part of GRASS for a full release.

CONTRIB

The contributed programs are in the directory $GIS/*src.contrib*. These programs are NOT compiled automatically by *GISGEN*. The state of these programs vary. Some may compile with **gmake;** others are more suitable as starting points for programmers who will be writing new software.

RELATED

The GRASS user community has discovered that there are several public domain programs that are very useful in conjunction with GRASS. These are found in the directory $GIS/*src.related* (assuming you have unloaded the related source code from the release media). Compile these programs based on the instructions (if available) in their respective directories.

GARDEN

The GARDEN programs are in the directory $GIS/*src.garden*. GARDEN programs are those that mix the capabilities of GRASS with the capabilities of one or more of the ‘‘related’’ programs (or other systems). Some require successful compilation of the ‘‘related’’ programs and generally compile using **gmake**. Of particular interest are the tools in *grass.informix,* which link GRASS with the Informix relational database system.

8. GRAPHICS DRIVER CONFIGURATION (etc/monitorcap)

Now you must create the file $GIS/*etc/monitorcap*, which tells the GRASS graphics application programs and graphics drivers how to communicate with one another. This file contains lines that describe each graphics driver on the system. After GRASS compilation, the $GIS/*etc/moncap.sample* file contains many different entries, all commented out (using the # symbol in the first column).[11] You must first copy the *moncap.sample* file to create a new *monitorcap* file. Then, edit the *monitorcap* file for your system's graphics devices. At a minimum, you will be uncommenting those lines that describe the graphics devices for which GRASS code has been compiled. (You identified these devices in the *lists/local* file before compiling GRASS.)

Each line in the *monitorcap* file contains six fields separated by colons:

name : program : description : fifos : tty : msg

name	The name the user uses to refer to the driver.
program	The actual program name as a UNIX command. It is specified as a relative path from $GIS (see examples).
description	A short 4- or 5-word description of the driver.
fifos	The names of the 2 fifo files that are assigned to this driver. These files must exist as named pipes,[12] have read and write permission by everybody, and not be used by any other driver (see NOTES).
tty	In some cases, it is desirable to force the driver to be started from a specific tty device. If this is the case, put the full path name of the tty in this field. Otherwise leave the field empty (see examples and NOTES following).
msg	If the tty field is specified and an attempt to start the driver is made from another tty, then this message is printed.

Examples:

[11] The original file is named *moncap.sample* to prevent subsequent compilation efforts from overwriting any existing *monitorcap* files.

[12] Named pipes are created with the command *mknod filename p*. This is done automatically for about 20 sets of fifo files by the *GISGEN* script. (The word ''fifo'' is short for ''first-in-first-out''.)

1. This first example is for a single device system, in which the monitor can be started from any terminal (tty). The two fifos are */usr/grass4.1/dev/fifo.1a* and */usr/grass4.1/dev/fifo.1b*. Here, the name of the driver is *x0*. Note that the driver program itself is specified as *driver/XDRIVER*. This is a relative path reference which will be translated into $GIS*/driver/XDRIVER*. Note that the path names of the fifos must be fully specified.

```
x0:driver/XDRIVER:Sun driver:/usr/grass4.1/dev/fifo.1a /usr/grass4.1/dev/fifo.1b::any terminal
```

2. The following example is for a two-monitor system, with the same graphics device available on each monitor. Note that the same program is used for both devices, but that the name field and the fifos field are different. Also note that both drivers must be started from a specific tty.

```
mass1:driver/MASS:Driver 1:/usr/grass4.1/dev/fifo.1a /usr/grass4.1/dev/fifo.1b:/dev/tty4:tty4
mass2:driver/MASS:Driver 2:/usr/grass4.1/dev/fifo.2a /usr/grass4.1/dev/fifo.2b:/dev/tty5:tty5
```

NOTES

1. When you are installing a driver, a pair of fifos (also known as named pipes) must exist for the driver to use. The names you choose are arbitrary. Fifo pairs with the names *fifo.1a* and *fifo.1b, fifo.2a* and *fifo.2b, fifo.3a* and *fifo.3b, fifo.4a* and *fifo.4b,* and *fifo.5a* and *fifo.5b* are created in the directory $GIS/*dev* by *GISGEN*.[13] It is suggested that you use these in your *monitorcap* file.
2. The tty field is not sufficiently robust to handle all situations. You should probably leave this field blank if installing GRASS on a SUN system. (In particular, SUN users running under Suntools must start the driver from the bitmap terminal, but the actual tty device number will vary from window to window.)

9. DIGITIZER DRIVER CONFIGURATION (etc/digcap)

NOTE: *The following instructions are for the new (4.1)* v.digit. *If you prefer to use the 4.0 version, see the* ''APPENDIX'' *at the end of this document.*

Now you must build the file $GIS/*etc/digcap*, which identifies the available digitizers and which I/O port each digitizer uses.[14]

[13] See section 7, "GRASS Compilation," for details about *GISGEN*.

[14] There are some sample *digcap* files in $GIS/*etc* that can be used as templates. To find these, use the command: **ls $GIS/etc/digcap***

Every *digcap* file should specify the ‘‘none’’ digitizer, which allows for on-screen digitizing using the mouse as the digitizer. In addition, there must be an entry for each digitizer locally available.

Each line in the file contains four fields separated by colons:

name : tty : driver : description

name	The name the user uses to refer to the digitizer.
tty	The tty port to be used by the digitizer.
driver	The name of the driver file (in the $GIS/*etc/digitizers* directory).
description	A short 4 or 5 word description of the digitizer.

Example:

This example is for a single digitizer system. The digitizer is an Altek digitizer on */dev/tty1*. The ‘‘none’’ digitizer is also specified.

```
altek:/dev/ttyb:al30f8_16: Altek digitizer, AC30, format 8
none:nodig:nofile: Run digit without the digitizer
```

NOTE: If the digitizer is later moved to a different tty port, the tty field must also be modified to reflect the change.

These next steps must be done while running as the user *root*, so **su** to *root*.

For each tty port */dev/tty***X** specified in the *digcap* file, type the command:

chmod 0666 /dev/*tty***X**

Also be sure that no programs (e.g., **getty** or **init**) are listening on those tty ports for logins. This may require modification to the files */etc/ttys* or */etc/inittab* to disable **getty** or **init** from running on that port. See your system manager’s guide for details.

Now exit from the **su** to become the user *grass* again.

10. PAINT DRIVER CONFIGURATION[15]

The GRASS program **p.map**[16] requires driver programs for each hardcopy color printer on your system that you intend to use with GRASS. With the exception of the ‘‘preview’’ and ‘‘preview2’’ drivers, which send their output to

[15] There is no $GIS/*etc/paintcap* file.

[16] and its potential replacement **p.map.new**

the graphics screen, the paint drivers send their output to tty ports named */dev/<* device*>* , where *<* device*>* is the driver name. For example, the ''tek4695'' printer uses */dev/tek4695* for its output.[17]

You will have to link these names to real device ports. For example, suppose that the ''tek4695'' printer is on */dev/tty10*. Link the driver to the port by typing these commands (note you may need to be logged in as *root* to create files in */dev*):

```
ln /dev/tty10 /dev/tek4695
chmod 0666 /dev/tek4695
```

Also be sure that no programs (e.g., **getty** or **init**) are listening on those tty ports for logins. This may require modification to the files */etc/ttys* or */etc/inittab* to disable **getty** or **init** from running on that port. See your system manager's guide for details.

The paint driver configuration design supports multiple printers of different types. For example, it is permissible to have both a ''tek4695'' and a ''shinko635'' printer on the system.

Multiple printers of the same type may be connected to a single system. Similarly, remote printers can also be supported. You must familiarize yourself with the paint support files and directories to enable such capabilities. The support files and directories, kept in the directory $GIS*/etc/paint*, are:

driver.sh
: A directory containing shell scripts that set driver variables and then generally call identically named programs in the *driver* directory.

driver
: A directory containing the actual binary programs that process the data and generate output to the devices.

driver.rsh
: A shell script that can be used in situations where the paint device is supported by GRASS on a remote machine.

When a user selects a driver, the selection is made from names in the *driver.sh* directory. To provide for a second or third printer of a common type, simply make a new entry in this directory. For example, say there is a *tek4695* file already here, but you want to support two ''tek4695'' printers. Copy the *tek4695* shell script to a new file, say, *tek4695b*. You now must edit this new file and identify the tty device of the second printer. Let us assume that you decided to run this device from

[17] The ''tek4695'' and ''shinko635'' printers perform parallel i/o only. If your machine does not have a parallel port, you will have to use a serial to parallel converter.

/dev/tek4695b. Edit the new script, changing only the last line from:

```
exec ${PAINT_DRIVER?}
```

to

```
exec ${GISBASE?}/etc/paint/driver/tek4695
```

The rest of the script should remain untouched. For further details on the operation of the paint driver, refer to the *GRASS 4.1 Programmer's Manual.*

If a machine running GRASS needs access to a printer on a remote machine that is available via a local network, you can make other modifications to the *driver.sh* files. For example, imagine the following situation:

machine A	Runs GRASS 4.1 has tek4695 printer connected to */dev/tek4695*
machine B	Runs GRASS 4.1 needs access to machine A's tek4695 printer

On machines A and B, edit the *driver.sh/tek4695* files (note that the files may be shared across an NFS mount if the machines are the same architecture, e.g., two SUN-4 machines). Change the following lines:

```
# for networked machines, set the host which has the printer
# printer_host= . print_host_alias= .
# case 'hostname' in
#       $printer_host|$printer_host_alias);;
#       *) exec rsh $printer_host $GISBASE/etc/paint/driver.rsh $PAINTER n
#          exit 0;;
# esac
```

Uncomment these lines and add machine names where appropriate:

```
# for networked machines, set the host which has the printer
printer_host= A print_host_alias= A.*
case 'hostname' in
     $printer_host|$printer_host_alias);;
     *) exec rsh $printer_host $GISBASE/etc/paint/driver.rsh $PAINTER n
        exit 0;;
esac
```

If $GISBASE is not identical on the two machines, change the reference to $GISBASE in the above lines to the actual full path name of $GISBASE on machine A.

This script, when executed on machine A, skips the edited session and behaves

normally. When run on machine B (or any machine other than A), a UNIX **rsh** connection is made to machine A, where the printer will be operated. Note that the **rsh** command requires that the individual users have logins on both machines and have the ability (through the *.rhosts* file in their home directories) to use **rlogin** and **rsh** between the two machines without passwords.

NOTE: The command **hostname** is used in this example. If you machine does not have a **hostname** command, you will have to substitute some script or program that returns the name of the machine (e.g., **uname -n**).

11. SUPPLY YOUR GRASS 4.0 DATA

Do you have GRASS 4.0 data you wish to provide to GRASS 4.1? If so, we suggest that you **copy** your 4.0 data into the 4.1 data directory (which you have already defined as $GISDBASE). Use the following instructions as a guide; you will need to modify them to fit the actual location of files on your system.

Note that your data is probably the most expensive component of your GRASS system – more expensive than the hardware, the training, the system support, and the extra air conditioning. It is our policy at CERL always to have at least two copies of data. This can be two copies on disk, two on tape, or on a combination of tape and disk. Keeping three or more copies, some stored in different buildings, provides necessary backups.

Using ½" tape

1. Login as *root*. (This must be done as *root*, otherwise the ownership of the database files will change.)

2. Change directory to your GRASS 4.0 data directory. For example,

```
cd /usr/grass/data
```

3. Use **tar** to copy the contents to tape. Note that this directory will contain the names of all the locations your data covers. You will **not** want to back up the **spearfish** database. Assuming your data locations are ‘‘spearfish,’’ ‘‘county,’’ ‘‘park,’’ and ‘‘base,’’ and your tape drive is */dev/rmt0,* you will use the following command:

```
tar cf /dev/rmt0 county park base
```

4. If you do not have enough disk space for two copies of your data, you will want to remove the 4.0 data at this point:

```
rm -rf county park base
```

However, if you do remove the data, you should first rerun step 3 with a second tape, just in case the first tape is written badly or becomes damaged.

5. Change directory to the GRASS 4.1 data directory:

```
cd $GISDBASE
```

(You set $GISDBASE early in the installation process. You may have to do this again if you have logged out and logged in again since the installation.)

6. Copy the data from tape back to the new directory. If you loaded the GRASS 4.1 **spearfish** database earlier in the installation process, you should find the directory *spearfish* in this directory. If all is well, read the data from tape:

```
mt -t /dev/rmt0 rew  (this command is probably not necessary)
tar xvpf /dev/rmt0
```

(In this example, replace ''/dev/rmt0'' with the device name of your tape drive.)

Copying direct disk to disk:

You may want to skip the tape step. If you have enough disk space to hold two copies of the data simultaneously, you can do the following:

1. Login as *root*. (This must be done as *root*, otherwise the ownership of the database files will change.)

2. Copy the data from the old directory to the new one. For example, assume the old data directory is */usr/grass/data* and the new is $GISDBASE, and assume the same databases as in the above example. Issue the following command:

```
cd /usr/grass/data
tar cf - county park base |(cd $GISDBASE; tar xvpf -)
```

12. DID YOU MISS SOMETHING?

In the process of installation you may have missed something. Some of the more common steps skipped or missed can be completed without going through the entire installation. The more common steps are mentioned here.

Bad compilation

The results of the compilation may not work for a variety of reasons. Past experience has shown that bad hardware, new compilers, different floating point processors, different hardware, and wrong information in the GRASS files in the $GIS/*src/CMD* directory can cause problems. If the problem is suspected to be wrong information, re-examine the ''System Configuration'' section.

Incomplete compilation

The $GIS/*src/CMD/lists* directory contains the file *GRASS* and, if you created them, the files *local* and *$ARCH*. These contain names of the directories that *GISGEN* compiles. It is possible that you:

1. told *GISGEN* to skip one of the directories; or
2. did not add a graphics driver to *src/CMD/lists/local*

You can compile the code in any directory with the following steps:

1. **cd** to that directory
2. run **gmake**

Sample data missing

The data on the release tape can be loaded at any time. Use the instructions in the ‘‘Extracting the Source Code and Sample Database,’’ section 7, as a guide.

13. GRASS INSTALLATION IS COMPLETE

At this point, GRASS is now installed on the system and can be run using the command **grass4.1.** This command will be found in the directory you specified during *setup*. You will find this directory name defined in the < header> file as UNIX_BIN.

14. CONTRIB-TEST AND OTHER PROGRAMS

In addition to the supported code under $GIS/*src* and $GIS/*src.alpha*, there are unsupported programs under $GIS.

Directory	Contents
src.contrib	GRASS programs useful to programmers
src.related	Programs that can work with GRASS
src.garden	Programs that mix GRASS and other systems’ capabilities

These programs are not automatically compiled because they are not supported by OGI. (OGI’s intention is to make these unsupported programs available. Compile and use them **at your own risk.**) If there are instructions within these directories, they should assist you in making decisions about what and how to compile.

15. CLEANING UP

If you are short of disk space you may want to remove some of the non-essential parts of GRASS.

Object files

Unless you will be actively changing all of the code, you will save at least six megabytes by removing all object (.o) files. This can be done with the command:

```
find $GIS/src* -name '*.o' -print | xargs rm -f
```

or (if your system does not have **xargs**),

```
find $GIS/src* -name '*.[o]' -exec rm -f {} \;
```

Source files

You can save about thirty megabytes of additional space by simply removing the source and manual directories:

```
rm -rf $GIS/src* $GIS/man
```

This will in no way affect the operation of GRASS. Note, however, that you should **not** remove $GIS*/man* if you have set up $GISBASE and $SRC to be in the same directory tree (since, in this case, deleting $GIS*/man* will delete the help files used by GRASS).

16. MOVING THE BINARY INSTALLATION

If you need to move GISBASE to another directory, it is no longer necessary to recompile the system. Common reasons for moving the programs include disk reorganization and installation on other systems.

For example, assume you wish to move $GISBASE from */usr/grass4.1* to */home/grass4.1*. To accomplish this, do the following steps.

1. Move the base directory. On most systems, *root* can accomplish this with the command:

   ```
   mv /usr/grass4.1 /home/grass4.1
   ```

   On others a copy will be necessary:

   ```
   (cd /usr; tar cpf - grass4.1) |(cd /home; tar xpf -)
   ```

   This command, if done as *root,* will preserve the original file ownerships and permissions.

2. Edit *grass4.1* (this for the user). Replace all references to the old directory with references to the new directory. For example, this script might be changed from:

   ```
   :
   GISBASE= /usr/grass4.1
   export GISBASE
   exec $GISBASE/etc/GIS.sh
   ```

   to:

   ```
   :
   GISBASE= /home/grass4.1
   export GISBASE
   exec $GISBASE/etc/GIS.sh
   ```

3. Edit the *monitorcap* file. In this example, the new GISBASE is */home/grass4.1*, so you would edit $GISBASE/*etc/monitorcap* and change any references to */usr* to refer to */home*.

17. APPENDIX

This appendix contains miscellaneous information about various other platforms as well as instructions for installing and using the 4.0 version of **v.digit**.

17.1. MISSING UNIX COMMANDS

GRASS is written assuming the existence of certain UNIX commands which may not be present on your system. These commands are listed in the table below. If you system does not have these commands, you will find shell scripts under $GIS/CMD/utils which emulate the functionality required. You can copy these scripts to UNIX_BIN (or any other directory that will be in everybody's PATH variable) giving them the correct name:

command	function	replacement[19]
clear	clears the terminal screen	clear.tput
tset	terminal setup	tset.tput
reset	terminal reset	reset.tput
more	pages displayed text	more.pg
whoami	prints user's name	whoami.sh
lpr	print command	< none supplied>

17.2. XDRIVER REQUIREMENTS

The XDRIVER as developed by CERL requires that the default visual be a Pseudo-color visual with at least 256 colors. If your X server does not set the default visual to this type, then you must teach it to do so. It is beyond the scope of this document to provide such instructions about your X server. However, we do know that for the Data General AViiON you can do this by editing the file */var/X11/xdm/Xservers* and placing the following line into this file:

```
:0 local /usr/bin/X11/X :0 bc -cc 3
```

17.3. XGRASS/OPENWINDOWS REQUIREMENTS

To make XGRASS function properly in a SUN Openwindows environment, you must make sure that the Motif keysyms are installed. To do this, check to see if the file */usr/openwin/lib/XKeysymDB* exists. If it does, append the file *src/xgrass/XKeysymDB* to it; if it does not, copy the file *src/xgrass/XKeysymDB* to

[19] The scripts clear.tput, tset.tput and reset.tput require the UNIX tput(1) command; the script more.pg requires the UNIX pg(1) command; and the whoami.sh command requires the Bourne Shell and the /tmp directory. There is no replacement for lpr because the command to print ascii text to a printer varies. You will have to write this one.

the directory */usr/openwin/lib*. This may require root permission. It is not clear how to fix the problem if you cannot do this. The error comes from Xt translation table parsing and cannot be fixed in Motif; this problem arises because the Motif keysyms have not been installed in the Openwindows server.

17.4. SCO UNIX

There may be problems compiling *src/libes/gis* under SCO UNIX. The **make** program on this system does not seem to be able to handle the length of the command which builds the library. The workaround is to execute the command by hand (after *GISGEN* fails), and then rerun *GISGEN*. To execute the command directly:

```
cd $GIS/src/libes/gis
ar rc ../LIB.$ARCH/libgis.a OBJ.$ARCH/*.o
```

where $ARCH is the architecture name you defined during *setup*.

17.5. CRAY LOADER PROBLEMS

The Cray UNICOS loader does not know that an archive library is a true library unless the library name is preceded by ‘‘-l’’. Since GRASS programs are compiled with the full name of the library (rather than using the ‘‘-l’’ flag) this will cause the loader to either not load the library or to load all the object files in the library. To get around this you will need to write your own **cc** command which prepends the ‘‘-l’’ flag to arguments that end in ‘‘.a’’ and then runs the real C compiler with the modified arguments.

17.6. INTERGRAPH HARDWARE PLATFORMS

There are two new programs that will not compile on the Intergraph hardware platform due to portability issues. Those users compiling on an Intergraph should be aware that the compilation process will be stopped at these two programs, or they should delete the program names from the list being used by the *GISGEN* process before beginning compilation. These programs are:

```
src.alpha/imagery/i.ortho.photo
src.alpha/mapdev/v.in.tig.lndmk
```

17.7. MIPS/CONTROL DATA HARDWARE PLATFORMS (RiscOS, EP/IX)

When running *setup*, be sure to specify */usr/bsd43/bin/cc* as the compiler to be used and specify the full path name.

After running *setup*, edit the *head/*$ARCH file to change the lines

```
XLIBPATH        = -L/usr/lib
XTLIBPATH   = -L/usr/lib
XMLIBPATH   = -L/usr/lib
```

to read

```
XLIBPATH        =
XTLIBPATH   =
XMLIBPATH   =
```

(i.e., remove the ‘‘-L/usr/lib’’ from each of these lines).

The Mips/CDC BSD compilation environment does not provide certain C header files that are required by some GRASS programs; these must be made available by taking the following steps after running *setup*. (Note that you should substitute "mips" in these examples with the values of $ARCH, i.e., the name you gave to this architecture in *setup*).

```
mkdir $SRC/include/mips
cp /usr/include/posix/stdlib.h $SRC/include/mips
cp /usr/include/posix/unistd.h $SRC/include/mips
```

Add ‘‘-I$(SRC)/include/$(ARCH)’’ to the COMPILE_FLAGS variable defined in *head/*$ARCH (or specify it during *setup*). Finally, edit $SRC/*include/mips/stdlib.h* to add the line

```
extern double strtod();
```

as the second line before the end of the file (i.e., this should not be the last line of the file!).

The Mips/CDC C compiler does not understand the ‘‘const’’ keyword, which is used in some GRASS programs, and using it causes a fatal compilation error. To work around this problem, add ‘‘-Dconst= ’’ to the COMPILE_FLAGS variable defined in *head/*$ARCH (or specify it during *setup*).

The Mips/CDC BSD compilation environment does not provide certain standard UNIX library routines that are required to build some GRASS programs. These muse be made available by adding them to the "gis" library. Follow the regular compilation steps by running *GISGEN*. After the "gis" library has been built, interrupt the compilation process and take the following steps:

```
cd $SRC/libes/gis/LIB.$ARCH
ar x /usr/lib/libc.a strtod.o ctype.o
ar ruv libgis.a strtod.o ctype.o
cd $SRC/src/CMD
```

Note also that some XGRASS programs may need to have the order of the libraries specified in their *Gmakefiles* changed in order to find these functions (i.e. you may need to re-order the library list to place ‘‘$(GISLIB)’’ at the end of the list.

17.8. SAMPLE HEADER FILES

Following are sample < header> files created during the preparation of this release.

SUN4		
CC	=	cc
ARCH	=	sun4
GISBASE	=	/grass4.1b/sun4
UNIX_BIN	=	/usr/local/bin
DEFAULT_DATABASE	=	/grass.data
DEFAULT_LOCATION	=	spearfish
COMPILE_FLAGS	=	-O
LDFLAGS	=	-s
XCFLAGS	=	-D_NO_PROTO
XLDFLAGS	=	
XINCPATH	=	
XMINCPATH	=	
XLIBPATH	=	-L/usr/lib
XTLIBPATH	=	-L/usr/lib
XMLIBPATH	=	-L/usr/lib
XEXTRALIBS	=	
TERMLIB	=	-ltermlib
CURSES	=	-lcurses $(TERMLIB)
MATHLIB	=	-lm
LIBRULE	=	ar ruv $@ $?; ranlib $@
USE_TERMIO	=	
USE_MTIO	=	-DUSE_MTIO
USE_FTIME	=	-DUSE_FTIME
DIGITFLAGS	=	-DUSE_SETREUID -DUSE_SETEUID -DUSE_SETPRIORITY
VECTLIBFLAGS	=	-DPORTABLE_3
GETHOSTNAME	=	-DGETHOSTNAME_OK

Intergraph Interpro		
CC	=	acc
ARCH	=	ig
GISBASE	=	/grass4.1b/ig
UNIX_BIN	=	/usr/local/bin
DEFAULT_DATABASE	=	/grass.data
DEFAULT_LOCATION	=	spearfish
COMPILE_FLAGS	=	-O -w -knr
LDFLAGS	=	-s
XCFLAGS	=	-DIGRAPH -DSYSV -D_NO_PROTO
XLDFLAGS	=	
XINCPATH	=	
XMINCPATH	=	
XLIBPATH	=	
XTLIBPATH	=	-L/usr/lib
XMLIBPATH	=	-L/usr/lib
XEXTRALIBS	=	-lbsd -lc -s
TERMLIB	=	-ltermlib
CURSES	=	-lcurses $(TERMLIB)
MATHLIB	=	-lm
LIBRULE	=	ar ruv $@ $?
USE_TERMIO	=	-DUSE_TERMIO
USE_MTIO	=	-DUSE_MTIO
USE_FTIME	=	
DIGITFLAGS	=	-DINTERPRO
VECTLIBFLAGS	=	
GETHOSTNAME	=	-DGETHOSTNAME_UNAME

Data General AViiON		
CC	=	cc
ARCH	=	aviion
GISBASE	=	/grass4.1b/aviion
UNIX_BIN	=	/usr/local/bin
DEFAULT_DATABASE	=	/grass.data
DEFAULT_LOCATION	=	spearfish
COMPILE_FLAGS	=	-O
LDFLAGS	=	-s
XCFLAGS	=	-D_NO_PROTO
XLDFLAGS	=	
XINCPATH	=	
XMINCPATH	=	
XLIBPATH	=	-L/usr/lib
XTLIBPATH	=	-L/usr/lib
XMLIBPATH	=	-L/usr/lib
XEXTRALIBS	=	-lPW
TERMLIB	=	
CURSES	=	-lcurses $(TERMLIB)
MATHLIB	=	-lm
LIBRULE	=	ar ruv $@ $?
USE_TERMIO	=	-DUSE_TERMIO
USE_MTIO	=	-DUSE_MTIO
USE_FTIME	=	-DUSE_FTIME
DIGITFLAGS	=	-DUSE_SETREUID -DUSE_SETEUID -DUSE_SETPRIORITY
VECTLIBFLAGS	=	
GETHOSTNAME	=	-DGETHOSTNAME_OK

Silicon Graphics IRIS		
CC	=	cc
ARCH	=	sgi
GISBASE	=	/GRASS.bin/4.1
UNIX_BIN	=	/usr/local/bin
DEFAULT_DATABASE	=	/net/GRASS.src/4.1/data
DEFAULT_LOCATION	=	spearfish
COMPILE_FLAGS	=	-cckr -O
LDFLAGS	=	-s
XCFLAGS	=	-D_NO_PROTO
XLDFLAGS	=	
XINCPATH	=	
XMINCPATH	=	
XLIBPATH	=	-L/usr/lib
XTLIBPATH	=	-L/usr/lib
XMLIBPATH	=	-L/usr/lib
XEXTRALIBS	=	-lPW
TERMLIB	=	-ltermlib
CURSES	=	-lcurses $(TERMLIB)
MATHLIB	=	-lm
LIBRULE	=	ar ruv $@ $?
USE_TERMIO	=	-DUSE_TERMIO
USE_MTIO	=	-DUSE_MTIO
USE_FTIME	=	
DIGITFLAGS	=	-DUSE_SETREUID -DUSE_SETEUID -DUSE_SETPRIORITY
VECTLIBFLAGS	=	
GETHOSTNAME	=	-DGETHOSTNAME_OK

Mips RISCos or CDC EP/IX		
CC	=	/usr/bsd43/bin/cc
ARCH	=	mips
GISBASE	=	/GRASS.bin/4.1final/mips
UNIX_BIN	=	/usr/local/bin
DEFAULT_DATABASE	=	/GRASS.src/4.1final/data
DEFAULT_LOCATION	=	spearfish
COMPILE_FLAGS	=	-O -traditional -Dconst= -I$(SRC)/include/$(ARCH)
LDFLAGS	=	-s
XCFLAGS	=	-Olimit 2000 -Wf,-XNd8400,-XNp12000 -D_NO_PROTO
XLDFLAGS	=	
XINCPATH	=	
XMINCPATH	=	
XLIBPATH	=	
XTLIBPATH	=	
XMLIBPATH	=	
XLIB	=	-lX11
XTLIB	=	-lXt
XMLIB	=	-lXm
XEXTRALIBS	=	
TERMLIB	=	-ltermlib
CURSES	=	-lcurses $(TERMLIB)
MATHLIB	=	-lm
LIBRULE	=	ar ruv $@ $?
USE_TERMIO	=	
USE_MTIO	=	-DUSE_MTIO
USE_FTIME	=	-DUSE_FTIME
DIGITFLAGS	=	-DUSE_SETREUID -DUSE_SETEUID -DUSE_SETPRIORITY
VECTLIBFLAGS	=	
GETHOSTNAME	=	-DGETHOSTNAME_OK

17.9. GRASS 4.0 DIGITIZER DRIVER CONFIGURATION--etc/digitcap

If you prefer to use the 4.0 digitizer driver configuration, use the following instructions **instead** *of those provided in Section 9.*

You will need to make sure that the following directories are compiled by *GISGEN*

src/mapdev/bin_dig
src/mapdev/v.digit2
src/mapdev/digitizers/none

as well as any 4.0 digitizer driver(s) for the digitizer(s) you have.

Now you must build the file $GIS/*etc/digitcap,* which identifies the available digitizers and the i/o port each digitizer uses.[20]

Every *digitcap* file should specify the ''none'' digitizer, which allows for on-screen digitizing using the mouse as the digitizer. In addition, there must be an entry for each digitizer locally available.

Each line in the file contains four fields separated by colons:

name : tty : program : description

name	The name the user uses to refer to the digitizer
tty	The tty port to be used by the digitizer
program	The actual program name as a UNIX command. It is specified as *digit*.
description	A short 4- or 5-word description of the digitizer.

Example:

This example is for a single digitizer system. The digitizer is a ''Kurta'' digitizer on */dev/tty1*. The ''none'' digitizer is also specified.

```
kurta:/dev/tty1:KURTA digitizer 9600 baud
none:nodig:digit: Run digit without the digitizer
```

NOTE: If the digitizer is later moved to a different tty port, the tty field must also be modified to reflect the change.

These next steps must be done while running as the user *root,* so **su** to *root.*

For each tty device */dev/tty***X** specified in the *digitcap* file, type:

chmod 0666 /dev/tty**X**

Also be sure that no programs (e.g., **getty** or **init**) are listening on those tty ports for logins. This may require modification to the files */etc/ttys* or */etc/inittab*) to disable **getty** or **init** from running on that port. See your system manager's guide for details.

Now exit from **su** to become the user *grass* again.

[20] There are some sample *digitcap* files in $GIS/*etc* that can be used as templates. To find these: ls $GIS/etc/digitcap*

Table of Contents

1: SUPPORT 1
2: TO COMPILE OR NOT TO COMPILE? 1
3: DATA IMPACTS 2
4: PREPARATION 2
5: DISTRIBUTION TAPE CONTENTS 4
6: EXTRACTING SOURCE CODE AND SAMPLE DATABASE 4
7: GRASS COMPILATION 7
7.1: COMPILATION STEPS OVERVIEW 9
7.2: COMPILATION SUPPORT FILES 9
7.3: COMPILATION SETUP 10
7.4: EDIT LISTS 11
7.5: XDRIVER 11
7.6: COMPILE GRASS 12
7.7: LINK GRASS PROGRAMS 13
7.8: COMPILE OTHER PROGRAMS 13
8: GRAPHICS DRIVER CONFIGURATION (etc/monitorcap) 14
9: DIGITIZER DRIVER CONFIGURATION (etc/digcap) 15
10: PAINT DRIVER CONFIGURATION 16
11: SUPPLY YOUR GRASS 4.0 DATA 19
12: DID YOU MISS SOMETHING? 20
13: GRASS INSTALLATION IS COMPLETE 21
14: CONTRIB-TEST AND OTHER PROGRAMS 21
15: CLEANING UP 21
16: MOVING THE BINARY INSTALLATION 22
17: APPENDIX 23
17.1: MISSING UNIX COMMANDS 23
17.2: XDRIVER REQUIREMENTS 23
17.3: XGRASS/OPENWINDOWS REQUIREMENTS 23
17.4: SCO UNIX 24
17.5: CRAY LOADER PROBLEMS 24
17.6: INTERGRAPH HARDWARE PLATFORMS 24
17.7: MIPS/CONTROL DATA HARDWARE PLATFORMS (RISCos, EP/IX) 24

17.8: SAMPLE HEADER FILES 26
17.9: GRASS 4.0 DIGITIZER DRIVER CONFIGURATION--
etc/ digitcap 30

--

--