1 INTRODUCTION

1.1 What's GRASS?

GRASS is the Geographical Resources Analysis Support System. It is a public domain raster geographical information system (GIS), vector GIS, image processing system, and graphics production system. The software for GRASS is free. The GRASS source code is provided to users, who may make changes to it. GRASS is ready-made for interfacing with RIM (an RDBMS), MAPGEN (a cartographic output generator for plotters), GCTP (a co-ordinate translation package), pbmplus (an image manipulation package), and xgen (a language for generating X-windows interfaces). GRASS comes free with three sample data bases: *Spearfish, Imagery* (currently not available), and *Global*.

GRASS can be used for preparing geographical data that are eventually used in some other system, or for processing and analyzing geographical data, including displaying the results. In the former case, geographical data are imported or digitized, converted, edited, analyzed and processed in GRASS and then exported to some other system, for instance a modeling or decision support system or an expert system, where they are used to build a geographical information database. For analysis, GRASS offers a number of analytical tools and functions for problem oriented displays. With the 1993 release of version 4.1 the system consists of some 300 different computer programs implementing a broad range of functionality. However, most analysis functions are based on grid cell files, i.e., raster format[1].

1.2 Documentation

Grass Reference Manual

Explanations of all the commands used in the main and alpha distribution of GRASS (which will be called 'tools'

[1] This paragraph copied from Fedra, K. and M. Kurbat, Hybrid Geographical Information Systems. In *EARSeL ADVANCES IN REMOTE SENSING* Vol. 1, No. 3 - VII, 1992, pp. 89-100.

in the remainder of this text) are provided by the *GRASS 4.1 Reference Manual*[2]. This consists of two volumes and is thematically organised; for example, all raster tools are put together. The tool type is indicated by its first letter, e.g. *r.tool* for raster tools. Within each theme the tools are presented in alphabetical order. For each tool a short description of its purpose, a synopsis of its parameters, a description of its use, notes explaining some of the background, and a reference to related tools are provided. For an extended discussion of tools, including mistakes and omissions in the Reference Manual, see sections 4.3, 6.1 - 6.5, 7.1 and 7.2 below.

You will save yourself and your System Administrator a lot of time if you start your Grass apprenticeship by leafing through the Reference Manual at least once before you start using Grass.

Grass Tutorials

For many of the more involved GRASS procedures separate tutorials have been written. Tutorials are available for the following subjects:

- Image Processing,
- Map Digitizing (v.digit),
- Map Calculations (r.mapcalc),
- Inference Engine (r.infer, r.binfer),
- Boolean Operations,
- Tool Access,
- Weighting (r.weight),
- Watershed Analysis (r.watershed) and
- Data Extraction for DLG-3, DTED/DEM and TIGER formats.

If you plan to use a procedure for which a tutorial exists, you are strongly recommended to get the tutorial (see 'The Grass File Server' below) and read it carefully. This manual will refer the reader to the appropriate tutorials in the introduction to each main section.

Other Manuals

All over the world people using GRASS feel the need for introductory, beginner-level or expert-level texts describing Grass and how to use it. This paragraph discusses four of those texts, which have been offered for general use by their authors, in rising order of complexity.

• If you're put off by reference manuals in general, try the *Grass Training Source Materials* folder written by Fred W. Limp. This provides the training materials for a 5-day intensive hands-on course, using the *spearfish* dataset. Although it's geared to Grass 3.0 (it's currently being updated) most of the stuff can still be used in Grass 4.0 and 4.1. The source materials are presented in small chunks so as to fit comfortably on a transparancy.

• Mara F. Tongue (US Dept. of the Interior, Geological Survey, National Mapping Division, 345 Middlefield Rd., Menlo Park, CA 94525, phone: (415)329-4290, fax: (415)329-0496, email: mtongue@shiva.wr.usgs.gov) has written a *beginner's manual*. It contains Grass basics (login, monitor, help, exit), notes on the creation of Location and Mapset, importing, georeferencing and rectifying vector and raster files, editing vector files, and mosaicking raster

[2] Tools that are part of the *main* distribution of Grass are well tested, well documented, and (supposedly) bug free. Tools that are not (yet), are distributed as *alpha* code so that the user knows there may be trouble ahead. Normally it takes one or two revision cycles before alpha tools migrate to the main distribution.

files. This manual is not freely available, but asking politely may help.

• *Grass 4.0: Introduction to GIS Using GRASS* by John E. Parks (Center for Advanced Spatial Technologies, National Center for Resource Innovations, SW, University of Arkansas, Fayetteville AR), which dates from November 1991 and is currently available in draft form only, contains an introductory chapter on the history and origins of GIS, a chapter on map projections, and a number of chapters describing data display, digitizing, a simple analytical GIS application, a complete watershed analysis, how to convert sites data to Grass format, notes on networking and hardware, and a handy glossary in which you can find explanations for much of the GIS jargon you will hear and read. You can get this tutorial by anonymous FTP from **moon.cecer.army.mil**: grass/grass4.0/documents/tutorials/parks.

• John Q. Ressler's *Grass Problem-Solving Manual* takes you a few steps further, through a series of increasingly complicated problem analyses. It provides chapters on how to structure a problem in Grass, on problems that can be solved using just one map layer, and on problems requiring multiple map layer analyses. Nine problem situations are described, the solutions to six of them are given in an appendix. A useful manual once you're through the first stage of Grass apprenticeship. Resslers manual can be ordered from Non-Credit Programs, Central Washington University, Ellensburg WA 98926, tel (509) 963-1504.

I would like to thank Alan McPherron (Dept. of Anthropology at Pittsburgh University) for pointing out the existence of some of these manuals and helping me to get hold of them.

Other Documentation

Besides the documentation that I've already mentioned, you can also find useful information in the Grass Programmer's Manual, the Grass Clippings quarterly, and the daily electronic mail traffic on the GrassHopper mailing lists.

Those who intend to use Grass for a research project of more than six months duration, and who are not afraid of programming, should consult the *Grass Programmers' Manual*. This is particularly useful where the Grass Reference Manual does not provide enough information to solve a problem. The Programmers' Manual also contains sections with general background information about Grass (design philosophy, file formats, etcetera).

The quarterly *GRASS Clippings* keep you informed of current developments, new releases, past and future research, etc. You can often find useful hints and scripts in it. There's an index on Volumes 3 - 6 of the Clippings in Index B at the end of this manual.

The Systems Manager may help you contact the GRASS Mailing List, a public mailing list that you can use to ask or answer questions and to find out if other GRASS users have already solved your problems. Note that traffic on this list can be heavy (up to 15 messages a day), so your mailbox may become clogged if you take out a regular subscription but do not regularly read your mail. Appendix A of this manual contains all the information you need to make use of the GrassHopper lists.

The Grass File Server

The Grass file server (moon.cecer.army.mil) can be reached by anonymous FTP. From its main directory 'grass' on down it has the following file and directory structure (dd 30-8-93):

File/Directory name	Contents
README	
ls-lR.Z	An up-to-date overview of the contents of the file server
ftpHelp/	Helpful texts to deal with the file transfer protocol)
grass4.0/	(contains databases, documents, source and updates dirs)
documents/	Miscellaneous helpful documents (including this manual)
tutorials/	
4.0/	Tutorials for a number of difficult to use Grass tools
grass4.1/	(contains README and BUG_FIXES files, and documents, release, releaseInfo, and update_packages dirs)
documents/	Useful text files (tutorials, guides, overviews) about Grass
grass4.2/	(contains releaseInfo dir)
grassHopper/	Text files with instructions about the grassHopper lists, and archives dir
archives/	
grassp/	Monthly archived mail from the Grass Programmers' list
grassu/	Monthly archived mail from the Grass Users' list
grassInfoCenter/	(contains GICdocs dir)
GICdocs/	Useful textfiles relating to Grass and GISs in general
incoming/	(contains README and XKeysymDB files, files contributed by Grass-users, and OGI dir)
	Tools not present in the normal (main) distribution of Grass
OGI/	
source/	Tools not present in the normal (main) distribution of Grass
otherPrograms/	(contains icons, rim, and xgen1.1 dirs)
icons/	Long list of icons that you can copy and use to create your own (hardcopy) output

1.3 About This Manual

The user is urged to start his/her project by reading as much documentation as possible. This will save your own time and that of the system manager or staff member assigned to assist you. This manual aims to provide information that is not readily available from the other documentation mentioned in the previous section, or which is particular to the configuration of hard- and software that is used at the Faculty of Environmental Sciences of the University of Amsterdam.

So if you can't find a particular subject in this manual, you should first make sure that it is not already treated in one of the other texts mentioned. In general, the answers to many frequently asked questions are right there in the Reference Manual - you just did not read them. Please remember that the staff have their own research and education to do and are not supposed to spend their time answering questions that you can solve yourself.

A *GRASS Beginner's Manual* is provided with every terminal and can be bought for production costs. There is a set of *Grass Rference Manuals* next to each terminal; the other documents mentioned can be borrowed from the Systems Manager.

2 CONFIGURATION

2.1 Configuration

Hardware configuration

The complete hardware configuration is shown in the diagram below. At the moment of writing this, GRASS has been installed both on the PS4 and RS(adam) machines and on the mainframe called *diamond*, which is located at SARA. This in itself is not important for most users, since GRASS can be accessed over the network from all other terminals. However, in some cases the user may wish to login twice to GRASS - something that cannot be done within one version. Depending on the terminal that you are using, you may then have to tell the computer where to get the data and/or where to write or display the results to.

[HARDWARE DIAGRAM]

Software configuration

GRASS is installed under AIX, the IBM version of the Unix Operating System, and its graphics are handled through X-windows. Both Unix and X-windows are standard university environments for the workstation level. Network software has been installed to connect the filesystems on various computers together, creating in effect one large filesystem. This means that some problems may occur that have nothing to do with GRASS itself - and these can generally be solved by knowing something about AIX and X-windows (manuals for both are available).

Furthermore, since your data are likely to travel a lot over the local, university and national networks, it is well to know something about this too. For networking the TCP-IP network protocol is used. This provides facilities for remote login ('telnet') and file transfer ('ftp'). It is also possible to 'mount' part of one system's directory structure onto another system's directory structure; it can then be used as if it were an integral part of the second system.

2.2 Operating System Software

The Unix Operating System

Unix is the most widely used OS on university level sites. Here at the GIS Group an IBM version of Unix, called

AIX, is used. In many ways Unix resembles DOS, IBM's PC-OS; you will recognise many of the commands, although they may have other names. You will almost certainly need to work with some 10 to 20 very important Unix commands. Moreover, you will have to teach yourself the use of Unix's standard file editor, called "vi" (for "very incomprehensible").

To find out what you can do with Unix, the best thing is to simply browse through the Reference Manuals, of which there are two large volumes (for the PS), or using the CD-Rom (for the RS) which contains *Info Explorer*, a hypertext-like interactive manual.

Network Facilities

First of all, there is the 'remote login', which allows you to connect your terminal directly to some other computer. All the work is done on that computer, but the input and output come from and go to your terminal. This procedure is performed with the command **telnet** or **tn**.

Secondly, you may want to process your data using the processing power of a more powerful computer (the RS or even the mainframe). This can be done by 'mounting' your part of the directory-tree over a directory on this more powerful computer[3], or by executing a command in a so-called 'shell' on another computer.

Thirdly, you may want to physically transport a file from one computer to another. Instead of copying onto a floppy disc and recopying onto the new hard disk, you can use the **file transfer program** or **ftp**.

Fourthly, facilities for the sending and receiving of mail are provided. All Unix-computers provide the mailing software **mail**, and other mail handling software is generally available on most sites.

Help

You can get general help concerning the operating system by issueing the command **help**, or help concerning a particular operating system command by typing **man command**. Similar commands will provide help with ftp.

2.3 Unix Startup

Preparing the Text Screen

Most workplaces consist of two terminals, a small one for alphanumerical output (text) and a large one mainly for graphical output (pictures). After you have been assigned a user name, number, and directory by the Systems Manager, you can log in on your text terminal using your own name if you plan to use the 'local' machine, and as 'guest' on any other machine. In the latter case you have to do a remote login ('tn') to reach the machine that you are a user on.

[3] Since the mounting procedure will temporarily make whatever was in that directory unavailable the command can only be initiated by the Systems Manager.

Preparing the Graphics Screen

Before you start (and if nobody worked on it before you) you must prepare the graphics terminal. You must open a 'shell', start X-windows, and decide what kind of monitor you want to use.

If the graphics terminal has not been used yet, it displays the 'login' screen. In that case you log in as 'guest' (password 'gisguest'). Now, to be polite to other users who may wish to temporarily use your graphics terminal, you must provide them with a way to temporarily override your display. This is done by opening a new command shell with the command **open sh**. You will see the screen 'jump' and a new 'guest' prompt appear.

Since GRASS's graphics output is handled by X-windows the next thing to do is to start X-windows with the command **xinit**[4]. Now X-Windows can be run with or without something called a 'Motif Window Manager'. The command 'xinit' will automatically start this window manager, which you will recognise by the blue-grey 'worktable' edges that appear around the display windows[5]. You can stop these edges from appearing by pressing the left-hand mouse-button somewhere in the grey area, and choosing 'Quit MWM' from the menu that appears. What's the use of all this? You will find out in section 3.1.

Changing shells

You can see the effect of having opened a shell if you press the key combination **[Alt]-[Ctrl]**. You will change over from one 'shell', displaying the X-windows screen, to another, displaying the Unix prompt. Repeat the command to return to your own shell.

Using the Motif Window Manager (mwm)

If you have more than one Motif windows on your screen, note that at any one time only one of these is active, i.e. can be written in. You can see which window is active because that window will have brighter edges than the others. By clicking in a window you activate it.

The Motif edge around each window can be used to move, rescale, iconify or kill the window[6]. By holding down the left mouse button in the upper middle (name) bar, and dragging it to another spot, you can move the window around. By dragging the sides, top, or bottom, you can change the size of the window in one direction; by dragging one of the four corners of the window, you can change its size in two directions. To the left and right of the upper bar are three special buttons. To the left is the dot button, which hides a menu from which you can make a choice. This is mainly used to kill or hide the window. To the right there are the minus and square buttons. The minus button, when clicked into, will iconify (minimize) the window and transport it to the lower left corner of the screen.

---

[4] Except in the case of the 'xstat' monitor. There, X-windows is started as soon as the computer is turned on.

[5] Except for the *xstat* terminal; here you have to issue a separate command to start the Motif window manager.

[6] Resizing an X-windows display while in Grass will result in a window erase. You will then have to reissue your display commands.

You can restore it by clicking into the icon and using the menu that appears. The square button is used to enlarge the window until it fills the entire screen.

2.4 Exiting Unix

Exiting X-Windows

To stop the window manager (undo the Xinit command), type **[Ctrl]-D** using the keyboard belonging with the graphics terminal. This kills the current shell and returns you to the other shell.

Exiting Unix

Both terminals now show the 'local' prompt for the machine(s) that you've logged in on. Logout is done by repeating the command **exit**, which will eventually return you to the startup 'login' screens on both terminals.

3 ACCESSING GRASS

3.1 Starting GRASS

Login, .Profile

Directly after login you start GRASS by typing **grass4.0**. If the operating system cannot find GRASS and returns an error message, ask teh Systems Manager to add the path to the Grass program to your *.profile*. In the meantime, you can always start GRASS on the PS with the extended command **/u/bin/grass4.0**, and on the RS with **/usr/local/bin/grass4.0**.

The file called *.profile*, which lives in your home-directory, contains special settings for each user. Make things easy for yourself and have your profile take care of all kinds of chores![7]

Location

Once in GRASS, you are invited to define three things; the Location, Mapset, and Database. All three are in fact directories in Unix, and together they organise your data.

The Location is the overall name (top directory) for all the work you are going to do in GRASS within one specific geographic area. This often coincides with the subject of your project, e.g. 'MARS', 'BELGIUM' or 'INNERCITY'. Besides being the directory that contains all the files of that project, the location also stores the map coordinate system (projection) and the minimum and maximum map co-ordinates that you are going to use.

The recommended projection in the GRASS system is the Universal Transverse Mercator grid system, UTM. GRASS uses a number of different, predefined versions or spheroids, each with its respective regional advantages and disadvantages and distortions. GRASS includes utilities for converting individual latitude/longitude points as well as latitude/longitude grid cell based data into UTM. The basic limitation regarding UTM consists in the fact that the system supports only processing information within one of 60 UTM grid zones (...) at a time. If the user wants to work with a larger area than one single UTM zone, the UTM zone has to be extended or the information has to be split into pieces belonging to individual UTM zones, that have to be processed separately. If the source data contain no UTM information, the system can treat them in a simple x-y coordinate system, without any specific projection.

[7] The $GISDBASE, for instance, should not be equal to $HOME, but should be set to $HOME/data to avoid clogging up your $HOME-directory with all sorts of files and directories.

Alternatively, data can be fitted to a UTM zone if a sufficient number of reference points can be found. GRASS contains utilities for marking points with known latitude/longitude information, computing the parameters of the rectification and carrying out the rectification[8].

It is important to realise that whatever you do in Grass, you can never go outside the Location. All database queries and modifications are made to the chosen location only; it is not possible to access multiple locations within one session, and you can't (for instance) change UTM zones within a particular location through any Grass tool.

Mapset

Within the location you must define at least one Mapset. This is the name of a subdirectory, in which you will store all the files relating to one subject, e.g. within the location BELGIUM you may want to define a Mapset 'OECO' for all maps with an oecological subject, and a Mapset 'INFRA' for all maps with an infrastructural subject. GRASS always defines a mapset 'PERMANENT' which you are supposed to use as a kind of archive; you will work in a mapset defined by yourself while making use, when necessary, of base maps stored in PERMANENT. The mapsets are transparent to reading operations, but if you want to write, rename or delete maps in another mapset, you must exit GRASS first and re-enter into the correct mapset.

Datapath

In order to enable GRASS to find your data, you must define a Database. This is the directory that will be used to create the Location and Mapset(s) in. You are not free in the choice of datapath: you must use your HOME-directory (assigned to you by the staff). If you do not know which directory that is, leave GRASS and type the commands **cd** (change directory) and **pwd** (print which directory). Most likely the result will be something like '/u/username/data'; this is what you fill in, when asked for the database.

Once you have defined all three and confirmed the definitions when GRASS askes for it, you arrive at the GRASS starting screen. This displays a statement of ownership (not important) and a prompt: GRASS-GRID>. You are now at liberty to enter either Unix or GRASS commands.

Starting a Monitor

To make available (part of) the graphics screen to GRASS, you must use the command **d.mon start=<monitorname>**. For <monitorname> you fill in whatever is the name of your monitor (a list of these can be got by entering just **d.mon -L** and following directions). A succesfully started monitor appears as a small black window on the screen. If MWM was running when you started the monitor, it can be enlarged and moved with the mouse. If MWM was not running, you're stuck with what you've got.

Note that there are many monitornames available. So which one do we choose? Generally, the important difference is that between the monitors that have an 'x' in them, and the others. Normally, you will choose the monitor that is named after your terminal. If you want more than one monitor at the same time, you may use one of the 'x' monitors.

---

[8] This paragraph copied from Fedra, K. and M. Kurbat, Hybid Geographical Information Systems. In *EARSeL ADVANCES IN REMOTE SENSING* Vol. 1, No. 3 - VII, 1992, pp. 91.

The Region

When you create a Location you are required to specify the coordinate system, boundaries and resolution of that Location. This information is stored in the file named DEFAULT_WINDOW and will bug you for the rest of you Grass-life under the slightly different name of 'default region'. Grass assumes that all maps you are going to import or create under that Location have the same coordinate system and have boundaries that fall within its boundaries. If you specify something else, Grass will complain. Moreover, Grass will quietly assign the same resolution to any new map unless you specify something else. This is why you should be familiar with the concept of 'region' in Grass.

You can change the region settings in a number of ways - have a look at the options of the 'g.region' tool in the Reference Manual. But whatever the current settings are, GRASS implicitly resamples map data by Nearest Neighbour resampling to them when layers are created, combined or displayed. This can be most clearly seen when linear features are displayed in low resolution. The errors that may be caused by this are of two kinds: the first is due to the misalignment of regions (difference in position of cell boundaries), the second to a difference in resolution of regions (Jim Aanstoos, Research Triangle Institute, NC). If you are not aware of this your data may become changed without you knowing it.

3.2 Logging Off

Exiting Grass

Before exiting GRASS, you must stop the monitor(s) that you have used. This is done with the command **d.mon stop=<monitorname>**. If you don't remember the name of your monitor, remember to look at the top bar of the monitor, which displays its name.

Exiting GRASS is easy; you just type **exit** after the GRASS prompt and presto. Except that there are a few questions to be answered: Do you want to save all of the maps (and mess) you just made? Are you sure? Things are organised in such a way here that you can simply give two **[Enter]**s and you've left GRASS for good.

3.3 Grass Tool Access

Access Levels

As with most other software, GRASS offers various ways of accessing its tools. For beginners the easiest method is typing just the tool name; you are then prompted for any other information GRASS may need to begin executing that command. More advanced users may prefer typing the complete command-line (tool plus arguments) in one go. The ultimate is of course to put these command lines together with UNIX commands into a shell-script so that you only have to type in the script name.

Important Keys

Much-used keys in GRASS are the **[Esc]** key, which is used for confirmation, and the **[Ctrl]-C** key combination, which is used to cancel commands. **[Enter]** is used to move through menus.

Tutorial

For extended guidance on tool access within GRASS, read the tutorial on this subject and see the general tools section of this manual (6.1).

3.4 The Grass Directory Tree

Directory Tree

Let's have an excursion through your branch of the directory tree once you have made your first maps. We'll use Unix as our vehicle, and start by finding out where we are with the command **pwd** (print working directory). If you have not changed directories since entering GRASS, Unix will display the name of your home directory (you automatically go there everytime you log in). You can check where your HOME is with the command **echo $HOME**; if you're not 'home, use **cd** to get there.

Now, to find out which files and directories your HOME contains, type **ls -l** (long listing). The result resembles that of the 'dir'-command in DOS. You can travel down your tree with **cd <directoryname>**, just as in DOS. You'll find out that your tree looks essentially like this (but upside-down):

/
(Root-directory; this is the base of the tree)

adam
(Machine directory; this indicates the file system)

/u
(Users-directory; this contains separate directories for all users)

/<name>
(Your Home-directory; it is named after you)

/<locations>
(One or more locations defined by you)

/<mapsets>
(One or more mapsets defined by you, plus the mapset PERMANENT,
which is created automatically by GRASS)

/cell /dig /site_lists etc

(All maps of the same type - for instance, all binary vector maps - are put
together in a number of directories created automatically by GRASS - see below)

/<maps>

(Here, at last, are your maps)

The mapset level is the most important one in the tree. There, file access rules are applied. There, the maps you create are automatically stored in the various subdirectories. The structure of a mapset is explored in the next section. To return from wherever you may be in the tree to your HOME-directory, type **cd**.

Mapset directory structure

It will often be useful to know the directory structure created by GRASS in each mapset, not only to understand more fully what the GIS does to your data, but also if you want to meddle with files using UNIX. Here a synopsis of the structure of the standard mapset-subdirectories is given.

Some general information is stored in directories with capital letters: the default region is stored in *DEFAULT_WIND*, the current imagery group is *GROUP*, the current region in *WIND*, the search path used for finding files is stored in *SEARCH_PATH*, *MYNAME* stores a one line description for the location. Other directories have lower-case names, such as *windows* and *3d.view*, which contain unused region definitions and 3d view parameters, respectively.

Some GRASS tools will, when used for the first time, create their own directory structure below the mapset level. These directories are *COMBINE* (created by the r.combine tool), *WEIGHT* (the r.weight tool), *WATERSHED* (the r.watershed tool), *group* (the i.group tool), *icons* (the p.icons tool), and *paint* (the p.label and p.map tools). Their structure is discussed in the next section under the header **Tool directories**.

The main body of GRASS data consists of site, vector, and cell map layers. Each of these is stored in a number of directories:

- Information about the creation history for each data layer is contained in the directory *hist*.
- Site files are stored in the directory *site_lists*.
- Vector maps are stored in the directories *dig, dig_ascii, dig_att, dig_cats, and dig_plus*.
  The dig_ascii directory is used for the storing of ASCII-type vector files during import and export procedures. GRASS's own (binary) vector files are stored in the dig directory; however, these files contain 'bare' vectors and need to be supported by other files in the dig_att (attributes), dig_cats (categories), and dig_plus (topology support) directories.
- Cell maps are stored in the directories *cell, cell_misc, and cellhd*. The cell directory stores the main part of the maps (the cell values); these files are supported by the corresponding files in the cell_misc (histogram, range) and cellhd (header) directories. (The cell_misc directory in fact contains a subdirectory for each map, containing a file 'range'.) Each cell map has a color table assigned to it in the directory *colr*, with an optional secondary color table in *colr2*.

Tool Directories

Tool directories are created by some GRASS tools under the current mapset.

- The COMBINE directory contains scripts to produce one or more r.combine results.
- The WATERSHED directory contains scripts to produce one or more r.watershed results.
- The WEIGHT directory contains scripts to produce one or more r.weight results.

- The group directory contains definitions of groups of imagery files belonging together (e.g. the bands of a satellite image), as produced with the i.group command.
- The icons directory contains the icons created by you with the p.icons command and used by the d.icons and p.map commands.
- The paint directory contains labels and comments files used by the p.map command.

This is a practice worth copying if you are working with scripts for some special purpose. Create a directory PURPOSE and store all scripts for that purpose there.

3.5 Practice Data: Spearfish

Tutorial

Practice data are provided with GRASS. They are located in the directories /ps4/u4/grass4data and /adam/rsprogs/grass4.0/data. The database '**spearfish**' contains numerous maps for the spearfish area (Black Mountains, USA); the database '**imagery**' contains several raw satellite images of the same area (currently not available). The database '**global**' contains a number of lat-lon maps.

Practice data are accessed by linking the practice mapsets to a practice location under your HOME-directory. You can then use the data without damaging them. The process of linking is done as follows:

cd $HOME [Go to your Home-directory]
mkdir spearfish (or **global**) [Create a Location to receive the practice data]
ln -s /ps4/u4/grass4data/spearfish (or **global**) **/PERMANENT .** or
ln -s /adam/rsprogs/grass4.0/data/spearfish (or **global**) **/PERMANENT .** [Create a symbolic link between the directories]

Now you can start GRASS and fill in the starting screen as follows:

Location: **spearfish** (or **global**)
Mapset: **practice** (or any other name except PERMANENT)
Database: **the.path.to.your.home.directory**

3.6 Problems & Solutions

Documentation

If you're too lazy to consult the manuals, or somebody pinched them, there's an on-line help facility in GRASS. This means that, without leaving GRASS, you can consult a file that contains the very same text as the *GRASS Command Descriptions* do. It even does the searching for you! Of course you do have the disadvantage of seeing only one screen (half a page) of text at a time...

To ask for help, type **g.help**. GRASS will ask you what subject you are interested in, and will display a list of possible subjects. By following GRASS's directions you will obtain information about the tools and options that are available, and a little background.

Login

You may not be known as a user to the machine. Reasons: you have not been added as a user by the system manager; or you have not logged in on a machine where you are known as a user. Solution: first, logoff, read the manual and try again; second, ask the Systems Manager to help you.

Access

If your machine cannot find **grass4.0**, try **/u/bin/grass4.0** and **/usr/local/grass4.0/bin**. If these don't work you probably forgot to do a 'remote login' or 'mount' (whichever is applicable); or the Systems Manager that gave you a user ID forgot to adjust your .profile; get him to do it for you. Another access problem may occur if you try to login when you are already in GRASS; in this case you should get a straight refusal-with-message from GRASS.

If, after having entered a Database, GRASS can't find any of your data, don't panic. You probably did one of two things wrong: either you want to work with data that are not on the local machine and you have forgotten to do a 'remote login' or 'mount', or you made a mistake entering the Datapath. Go back and make repairs. If it still doesn't work, get the Systems Manager to help.

Monitor

As explained above, a GRASS monitor can only be started if the graphics display is properly prepared. If there's something wrong here, take note of GRASS's error message:

- If it says something about a 'Graph_set' you simply have to tell GRASS again where the monitor is. This is done with the command **DISPLAY=<terminal>:0.0 ; export DISPLAY**, where you fill in the name of your own terminal. Repeat the d.mon start=<monitorname> command and this problem should be solved.
- If GRASS complains that "the monitor is not running" and tries again, the problem is slightly more complicated. The monitor has been maltreated and must be reset. The Systems Manager knows how to solve this one quickly.
- A third kind of monitor problem may occur when the monitor you wish to start is already running, or is in use by someone else. If it's already running, use **d.mon select=<monitorname>**. If it's in use, start another. This kind of problem can be avoided by always stopping the monitor (**d.mon stop=<monitorname>**) before you exit GRASS.
- Problem four occurs when the monitor you've called up appears on somebody else's screen. This is a nasty one, because he/she will probably think that the fault is local, and will start messing up things even more. So if this happens, stop the monitor at once. Read this manual carefully to find out what you've done wrong and/or call the Systems Manager.

A problem for which there is no message occurs when, while displaying a cell file, the monitor displays only part of the columns, leaving the rest black. It is not known how this problem is caused, but it can be solved by stopping and re-starting the monitor. For any other problems with the monitor, contact the Systems Manager (boy, is he going to be busy...).

System speed

Network problems will occur when there are many users at the same time. Processes will slow down, and eventually the total amount of data transport over the net is so great that the special programs handling these data streams (NFS servers) stop working. You won't normally loose any data if this happens. There is only one thing you can do and that is to wait until the servers have regained their breath (or try to get other users to log off).

Sometimes users create a lot of data in a short time, which may fill up the available disk space and lead to a general halt of all processes. It may then be necessary for the whole system to shutdown and reboot; if this occurs you will be warned beforehand.

While it is possible to login from a computer outside the GIS rooms, this is not actually allowed, because the Systems Manager cannot keep track of these outside users.

Logoff

There are two things you should keep in mind when logging off: first, you must always **stop** your monitor before exiting GRASS, and second, you must **exit** from all levels that you opened during login. So don't leave your terminal as long as the "login:" prompt isn't displayed on both screens!

4 DATA INPUT

4.1 Data entry and editing

Digitizing tutorial

If you have digitised maps in other software and are now starting to digitize in Grass, you will have a distinct advantage over those who come unprepared. Still, digitizing procedures differ for different software. Although the basic steps (registration, vector digitizing, node snapping, topology building, labelling) are the same in most software, the methodology, nomenclature, and menu interfaces may be very different. To learn to digitize in GRASS the tutorial *v.digit* is available. Also, in the tutorial by Parks (mentioned in section 1.2) chapter 4 contains an extensive description of digitizing procedures in *v.digit*.

Digitizing is done with the help of digitizing tables, the settings of which have to be adjusted to GRASS beforehand; the Systems Manager will do this for you upon request.

Data entry in UNIX

To convert a raster map to a sites list, the command 'r.stats -1gz input=cellmap | s.in.ascii sites=sites_file' suffices (Malcolm Williamson).

Category labels for cell files can be typed directly into the relevant $LOCATION/cats file, using the format 'catnumber:label.'

4.2 Importing Maps

Different methods must be used to import vector maps and raster (cell) maps. Generally, vector conversions proceed through some form of ASCII text format. Raster conversions may sometimes be done more directly, using binary format, but file compression and header information still require special attention. Even if no ready tools exist for raster reformatting, you can in many cases still convert by using either the *PBMPLUS* conversion package by Jeff

Poskanzer[9], or the *Xv* utility by Bradley. These two programs read many formats and provide facilities for conversion to many other formats.

Grass is more strict than many other programs about topological requirements (e.g., about the closing of polygons). That's why you may have to execute the 'v.support' command with a snapping criterion even though the input file was topologically correct. The 'v.digit' command has options to show you where the open area lines are, if you want to edit these out manually. A related problem occurs when you import vector maps from other GIS, which may be more sophisticated in vector topology than Grass. In Grass the region boundaries are not regarded as valid area edges; all areas bounded by the region boundaries are therefore seen as 'open'. You must therefore edit in the missing edges before these areas are recognised and can be rasterized.

4.2.1 Importing Sites Lists

s.in.ascii converts an ASCII site-list file in GRASS to binary format.

4.2.2 Importing Vector Maps

To import a vector file frome some other format into Grass, a number of tools are provided (see below). Many of these are specifically targeted at importing DLG and TIGER files, formats that are not used in Europe. Useful tools include those importing ARC/Info, DXF, and Genamap formatted vector files. These and other (unsupported) importing procedures are discussed in the rest of the paragraph. Note: the tools marked by an asterisk are not part of the main GRASS source.

Tools

v.import converts ASCII Digital Line Graph (DLG) files, binary DLG files, and ASCII vector files into binary vector files, and creates the needed vector support files.

v.in.arc* converts Arc/Info data to GRASS vector format and stores output in the current mapset. **NB: This tool is still under alpha development. See instructions below.

v.in.ascii converts ASCII vector map layers into binary vector map layers.

v.in.dlg converts an ASCII or binary USGS DLG-3 (bdlg) file to a binary GRASS vector (dig) file. **NB:** This tool reads from a directory '$LOCATION/dlg' that must be created by the user. Unlike what it says in the Reference Manual, the tool understands only 'optional' format, not 'standard' format. The 'bdlg' files created as a by-product of the importing may be

[9] PBMplus is a comprehensive format conversion and image manipulation package. Its latest version (probably better than the code currently included in the Grass source) is available via anonymous FTP at **ftp.ee.lbl.gov**: pbmplus*.tar.Z, **wuarchive.wustl.edu**: graphics/graphics/packages/pbmplus/pbmplus*.tar.Z, and **export.lcs.mit.edu**: contrib/pbmplus*.tar.Z.

removed by the user.

v.in.dlg2 identical to *v.in.dlg* in grass 4.1.

v.in.dxf converts files in DXF format to ASCII or binary GRASS vector file format. **NB:** Use the '-a' option to ensure that a Grass format dig_ascii file is created directly. **NB 2:** Importing CAD spot elevations with this tool may pose the problem that point and label are contained in the same CAD layer. If you need both, you will have to import this layer twice (once as 'lines', once as 'labels'). **NB 3:** This tool may contain a bug that causes it not to find vector attributes shorter than two digits. Changing single digit attributes ('2') into double digit attributes ('02') allows v.in.dxf to find the attribute and place it in the dig_att file (Doug Hersh).

v.in.genamap* converts an ASCII Genamap export file into a GRASS ASCII vector file. **NB: This tool contributed by Jan Hartmann, FRW-UvA, The Netherlands. It can be found under ....

v.in.tig.basic* creates GRASS vector map from TIGER files. **NB: This tool is still under alpha development.

v.in.tig.lndmk* creates GRASS vector map from TIGER Landmark files. **NB: This tool is still under alpha development.

v.in.tig.rim* imports Census Bureau TIGER line data to GRASS vector format. **NB: This tool is still under alpha development.

v.in.transects* imports transect data to a GRASS vector map. **NB: This tool is still under alpha development.

Vector maps from Genamap

If you want to import vector maps made in Genamap in GRASS, the first thing you have to do is prepare the proper Genamap-output (the procedure is described in the *Genamap 5.1 Beginner's Manual*, section 15.1). When you are finished with that, log in to GRASS and use the File Transfer Program to enter your directory on the RT (**ftp rt**). After you logged in on the RT, you travel to the 'ZF25' directory that exists under your work area. Transport the ASCII export file (ending in .EE) with **get <filename>**, and leave FTP with **quit**. The export file will end up in your HOME-directory.

Conversion to a GRASS vectorfile proceeds in three steps:

- from Genamap ASCII format to GRASS ASCII format, with the command **v.in.genamap**. GRASS will ask you for the name of the .EE file and will offer to save points, lines and areas in separate maps (each of which has to be named by you). The resulting maps are saved in the Grass directory called 'dig_ascii'.
- from GRASS ASCII format to GRASS binary format, with the command **v.in.ascii**. Again you get the opportunity to change the name of the map; the result is written to the directory 'dig'.
- from an unsupported binary vectorfile to a supported binary vector file, with the command **v.support**. 'Supporting' a vectorfile entails two things. First, topology is generated. Since GRASS is more precise with this than Genamap, you may find that some areas cannot be closed and their tags are rejected. Refer to the GRASS-tutorial 'v.digit' for directions on solving this. Second, category support is created. This entails the assignment of an integer value to every different tag encountered in the map. This information is written to the directories 'dig_cats' and 'dig_att'.
- clean up the files that you do not need anymore, using the unix command **rm** for the files in '/arc' and the Grass tool **g.remove** for the ascii files.

If all went well, you can now call up your GRASS vector map with the command **d.vect** (- but remember first to adjust the region to your new map).

Vector maps from ARC/Info

The following information was compiled by Darell McCauley (mccauley@ecn.purdue.edu), Kenn Gardels, (gardels@ced.berkeley.edu) and Jim Gasprich (jimg@ocean.rutgers.edu):

- Export the ARC/Info vector file in 'ungenerate' format. Do not use the 'poly' option. This results in three files suffixed **.lin, .lab, .txt.** (The latter must be exported using Info).
- Create a directory 'arc' under your Grass mapset and transport the files to it.
- Import the files using **v.in.arc type=polygon lines_in=<.lin file> points_in=<.lab file> text_in=<.txt file> vector_out=<new_name> idcol= catcol= attcol=**. Fill in the correct field numbers from the .txt file with the latter three options.
- Run v.support on <newname>, building topology.
- Check that the results are satisfactory, then remove the files in /arc.

  NB: A problem may occur if the ARC/Info lines were in double precision. Malcolm Williamson (malcolm@cast.uark.edu) provided a script to fix this (see appendix D).

  NB 2: Importing vector files from ARC/Info using v.in.arc may run into problems when the ARC/ungenerate command generates files in exponential notation. In those cases the files must be edited to change a) the exponential "D" to "E", b) exponential notation to floating point notation.

  NB 3: V.in.arc won't read the .lab file if you do not specify a .txt file during import - you can trick it by specifying the .lab file for text import and selecting column 1 for both id and cat.

Vector maps from ROOTS

For maps digitised in ROOTS the following procedure should be followed:

- export your map in ARC/INFO format. Export both 'chains' and 'labels', using extensions **lin**, **lab** and **txt**. **NB:** since the 'labels' are to become categories in Grass, you should only use numerical labels in ROOTS.
- move the files by ftp or diskette to your Grass-directory. They should end up in a directory *$LOCATION/arc* (if this doesn't yet exist you should make it). If you are using a diskette, import the files with the command **dosread -a filename.old filename.new**. If you are importing the file over the network you can simply **get filename.old**, and convert it afterwards with **dos2aix filename.old filename.new**.
- use **v.in.roots linesfile labelco-ordinatefile label-txtfile grass-filename** to convert the import files to a Grass ASCII-file. **NB:** a bug in this step creates a pair of 0's at the end of the dig_ascii file in Grass; these must be removed (using the file editor *vi*) before you go on with the next step.
- convert the Grass ASCII-file to binary format with **v.in.ascii**.
- assign topology to the new map with **v.support**.
- clean up the files you don't need, using the unix command **rm** for the files in '/arc' and the Grass tool **g.remove** for the ascii files.

Vector maps from Autocad/SDR3

Acquiring data prepared in packages that are used in the field is done through conversion and import of DXF (Drawing Interchange Format) formatted files. In particular, vector files from Autocad and SDR33 will be imported using this format. Use the following procedure:

- export the map from Autocad or SDR3 in *DXF format*.
- use **v.in.dxf** to convert the dxf files to Grass binary vector format
- use **v.support** to create topology
- clean up the mess

Vector maps from IDRISI

To import vector maps from IDRISI, use the following procedure:
- create an Arc/Info 'ungenerate' file in IDRISI
- transport the file to a directory called 'arc' (you have to create this yourself) under your mapset in Grass
- run v.in.arc to import the file
- remove the temporary files

Vector maps from MOSS

According to Jim Hinthorne (GIS lab, CWU) this conversion is very difficult because MOSS, like Autocad, does not work with topologically correct arc-node structures. Before conversion is possible, MOSS vector files must therefore be cleaned up (slivers, overlaps, etc). This can be done in vector editing programs such as Genamap or LTPlus, which also have options to import MOSS files and export GRASS files.

Having said this, moss vector files CAN be imported into Grass if they are very clean (i.e. the polygon boundaries are exact duplicates), by appending the moss file to a normal dig_ascii file header and editing the vector part to put in the indicators (such as 'A 15' for Area line consisting of 15 coordinate pairs). Run 'v.in.ascii' afterwards to get a vector file that will consist solely of islands, then 'v.to.rast' to create a normal raster file (Kenn Gardels).

Scott Tweddale (tweddal@gis.uiuc.edu) has recently promised to come up with a conversion program AMS2GRASS and a tutorial to do the unthinkable.

Vector maps from ATLAS

Currently no easy route from ATLAS vector format to Grass vector format is known (Susan Stitt, National Biological Survey); known problems include 'blocking' by Atlas of DXF output files (which forces you to make an extra step through Autocad), formatting differences between Atlas ascii output files en Grass ascii input files (easting and northing interchanged, line or area indicator different from GRASS's).

4.2.3 Importing Cell Maps

Tools

r.in.ascii	converts an ASCII raster text file into a binary raster map layer.
**r.in.elas*	copies and converts an ELAS raster file to a GRASS cell file. **NB:** Tool contributed by Bruce Powell, US National Park Service. It can be found under /grass/incoming.
**r.in.erdas*	converts an ERDAS cell file to Grass cell format. **NB:** During importing of a .GIS or .LAN file you must specify the upper left coordinate as 1,1 (Ronald Thomas, Colorado State U).
r.in.ll	converts raster data referenced using lat-lon coordinates to a UTM-referenced map layer in GRASS raster format.
**r.in.poly*	creates raster maps from ascii polygon/line data files. **NB:** This tool is still under alpha

development.

r.in.sunrast* converts SUN raster files to GRASS raster files. **NB: This tool is still under alpha development.

Cell maps from IDRISI

To import cell maps from IDRISI, use the following procedure:

- export an uncompressed binary cell file and a cellheader file from IDRISI
- transport the files to Unix, using diskettes or ftp
- run r.support to create support files for the cell file, using the information from the cellheader file, **or**
- convert the files to a Grass cell file with **r.in.idrisi old.name new.name**.
- It may be that a problem occurs if your IDRISI file was of the '2 byte per cell' type, but IDRISI provides a module to undo this.
- clean up the mess.
- **NB:** if you're really lazy, there is a conversion program called 'grassidr.exe' available from ftp-server **midget.towson.edu**, directory /gis/modules, which takes uncompressed IDRISI cell files.

Other cell formats

Many conversions of raster/cell formats to Grass can be accomplished using the utilities in the PBMPLUS package by Jeff Poszkanzer. A number of examples are given here by Bruno Minoja at Consorzio Interuniversitario Lobardo:

- TGA to Grass: **tgatoppm <infile> | pnmtorast | r.in.sunrast**
- LAN or TIFF to Grass: **tifftopnm <infile> | pnmtorast | r.in.sunrast**
- TIFF can be imported more easily by using **r.in.tiff**. Other conversions can be simplified by using **r.in.pnm** once the files have been converted to PNM format.

4.2.4 Importing Imagery

**i.in.erdas* creates raster files from ERDAS files.
NB: This tool is still under alpha development.

4.3 Reading In Tapes

Reading In Tapes

Tapes containing satellite images have to be read in through SARA. This is done with the following procedure: first (1) you should prepare a suitable Location and Mapset in GRASS, then (2) you contact SARA to have the tape mounted onto the reading device, next (3) you read in the files using GRASS tools, and (4) you clean up the mess.

1) GRASS must be run on SARA to be able to read in tapes. You establish contact with SARA by using the

command **tn diamond** and logging in as user **a907tape** (the Systems Manager will tell you what the password is). Then start GRASS creating a new Location and Mapset. The Location should be defined as having x/y coordinates, the default region should be defined as minimum values (0, 0), maximum values (3500, 3500), resolution 1 x 1. The choice of Mapset name is free.

2) The tape will be mounted by SARA if you type the UNIX command **rlvmount <tapenumber>**. The tape number has earlier been provided by SARA, when you handed them the tape. It always starts with the code S0, followed by three numbers and one capital letter. Until the tape has been mounted you will see the message "Awaiting mount ..." on screen. If the tape has not been mounted after 15 minutes, you should contact the SARA information desk (tel. 5928016). The tape has been mounted when the message "Tape mount complete" appears.

3) After the tape has been mounted you type **i.tape.other** (for SPOT imagery) or **i.tape.mss / i.tape.tm** (for Landsat imagery) to enter a menu in which you must specify tape parameters.

Enter device name: **/dev/rmt0**
Please mount and load tape <enter>
Enter tape identification: **<tape id as provided by SPOT or Landsat>**
Enter image description: **(fill in your own description of the image)**
Enter title: **(ditto)**

Only the device name is compulsory; you may run through the other prompts with <enter>.

Generic Tape Extraction

Tape Layout

Number of files to be skipped: **2** (to avoid reading headers and trailers)
Number of records in the remaining files to be skipped: **0** (to avoid reading useless lines)

Band Files

Number of bands on the tape: **3** (for SPOT; 7 for Landsat)

Data Format

- *BSQ* (Band Sequential, for Landsat imagery)
- *BIL* (Band Interleaved by Line, for SPOT imagery)

Enter an **x** before the option of your choice.

Length of the longest record on the tape: **5500** (for SPOT imagery)
Extract which bands:
- *Band 1*
- *Band 2*
- *Band 3* (depends on the number you entered earlier)

Extract all three bands.

Enter prefix name for bands to be extracted: **(all image bands will get this name in addition to a numerical suffix corresponding to the band number)**
Start row: **1**
End row: (mentioned in your image documentation under: "Number of multispectral lines")
Start col: **1**
End col: (mentioned in your image documentation under: "Number of pixels per monospectral line")

Now, at last, the images are going to be read in. This will take about 5 minutes per band. Display all bands separately on the screen with the **d.rast** command for a final checkup.

NB: Sometimes image band 2 (in SPOT images) is not read in correctly the first time around. You must remove it (with the **g.remove** command and re-execute the **i.tape.other** command, selecting just Band 2 the second time around. Note that, when describing the tape layout, you should still enter 3 as the number of bands on the tape.

4) Tell SARA to dismount the tape by issueing the command **rlvunmount**. If you want to use the imagery locally instead of on diamond, you must transport the files by FTP to the right spot in your own directory tree. Don't forget to remove the files on diamond after you checked that they were transported correctly, and note that satellite imagery usually needs large files (> 10Mb) so you can't simply transport and use them on our computers (ulimit too low, disk space too low).

4.4 Data Manipulation Tools

Tutorials

Tutorials are available on TIGER in GRASS, DLG-3 data extraction, and DTED and DEM data extraction. See also chapter 7 of the tutorial by Parks (mentioned in section 1.2), which discusses the conversion of sites data from Lat/Lon to UTM and vice versa. The currently available commands are listed below.

Tools

GRASS tools starting with 'm.' are used in the manipulation of external data. During the process of importing data into GRASS it is often necessary to resample data in various ways. You should be aware that this always results in a certain loss of data. When information about the tool is found lacking in the command description manuals, remarks are added here.

m.datum.shift	returns geographic coordinates based on different spheroid (and datum) than the one used to obtain the original coordinates.
m.dem.examine	provides a terse description of USGS Digital Elevation Model (DEM) data files stored on 1/2 inch magnetic tape.
**m.dem.extract*	extracts USGS DEM data from 1/2 inch magnetic tape. **NB:** This tool still under alpha development. **NB:** Extracting from disc may not be possible. To remove CR's or LF's from DEM files, use the command *fold $*|sed 's/^M/g'|awk '{printf("%s",$0)}'* in which ^M stands for <Ctrl>M.
m.dmaUSGSread	extracts DEM data produced by the DEfense Mapping Agency (DMA) but supplied by the USGS (in a different tape format) on 1/2 inch magnetic tape.
m.dted.examine	provides a terse desciption of level 1 and 2 digital terrain elevation data (DTED) files produced by the DMA on 1/2 inch magnetic tapes.
m.dted.extract	extracts DTED level 1 and 2 data produced and supplied by the DMA on 1/2 inch magnetic tape.
m.examine.tape	provides a description of the files on a 1/2 inch magnetic tape.
m.eigensystem	computes eigen values and eigen vectors for square matrices.
**m.flip*	flips elevation data extracted from systems that retrieve data by rows from south to north. **NB:** This tool still under alpha development.
m.gc2ll	converts geocentric to geographic coordinates.
m.geo	coordinate transformation tool. **NB:** This is 'contributed' code, which can be gotten from the moon.

**m.in.pl94.db3*	imports Demographics from Census PL94-171 dBase3 files on CDROM. **NB:** This tool still under alpha development.
**m.in.stf1.db3*	imports Demographics from Census STF1A dBase3 files on CDROM. **NB:** This tool still under alpha development.
**m.in.stf1.tape*	extracts lines from a text file based on column contents, especially Census STF1A dBase3 files. **NB:** This tool still under alpha development.
m.ll2gc	converts geographic to geocentric coordinates.
m.ll2u	converts geographic coordinates to Universal Transverse Mercator (UTM) coordinates. **NB**: This tool takes two kinds of LAT/LON input files (see Reference Manual). The one with degrees and decimals need not be in the format indicated in the Reference Manual, but may be of the form '-91.417 34.250'. So no E,W,N,S indicator is necessary if you specify the minus sign for coordinates in the Western and Southern hemispheres. **NB 2:** execution of this command will **not** convert the LAT/LON map header region description to UTM, nor are the coordinates in the attribute file converted to UTM. Solution: first, put the region coordinates in a file, have them converted to UTM by *m.ll2u* and replace the old header coordinates with the new ones; secondly, convert the binary file to ascii with *v.out.ascii*, edit the ascii attribute file to remove the A's (for Area) with the unix command **sed '1,$s/A//g' att_filename**. Now use *m.ll2u* with the option **-z** to convert the files, put back the A's in the attribute file with the command **sed '1,$s/^/A/g'**. Make sure all files have the same name before you reconvert to binary GRASS format with *v.in.ascii*.
**m.lulc.USGS*	creates raster map layers from a Composite Theme Grid (CTG) file created by *m.lulc.read*. **NB:** This tool still under alpha development.
**m.lulc.read*	extracts Landuse/Landcover data in the ASCII CTG data format distributed by the USGS into a working file for *m.lulc.USGS*. **NB:** This tool still under alpha development.
**m.proj*	calculates conversion coordinates for geographic positions. **NB:** This tool still under alpha development.
m.region.ll	converts UTM coordinates falling within the current geographic region to geographic coordinates.
m.rot90	rotates elevation data extracted by either *m.dted.extract* or *m.dmaUSGSread*. **NB**: This tool is not meant to rotate GRASS cell maps, but can be tricked into doing so. Uncompress the cell map, rotate the uncompressed cell file, and edit the 'cellhd' file to interchange row and column numbers (Michael Shapiro, shapiro@zorro.cecer.army.mil).
**m.tiger.region*	finds geographic region information for US Census TIGER input data. **NB:** This tool still under alpha development. Currently bugged but being fixed.
m.u2ll	converts UTM coordinates to geographic coordinates.

5 DATA OUTPUT

5.1 Display Tools

General

The 'd.mon' command calls up monitors that are sized according to default values coded into the monitor drivers. In most cases the heighth/width ratio of the monitor will not be equal to that of your default region. Instead of resizing the monitor by using the Xwindows mouse options, you may want to consider setting the variables XDRIVER_WIDTH and XDRIVER_HEIGHT in your .grassrc file (this hidden file lives in your HOME directory), to determine the size of your monito. In the same way the XDRIVER_LEFT and XDRIVER_TOP variables can be used to determine placement with respect to the upper left corner of your screen (Darell McCauley and Bob Harrington, bobh@hwr.arizona.edu). For enterprising users, these variables could be set automatically from the region information using a shell script.

Tools

GRASS tools starting with 'd.' are used for the display of graphics on the monitor and for the management of the monitor itself. Display abilities include 2-D display, 3-D display, profiles, histograms, pie charts, graphs and screen dumps to hardcopy devices. See also chapter 3 of the Parks tutorial (mentioned in section 1.2) discusses Grass display functions. The currently available commands are listed below. When information about the tool is found lacking in the command description manuals, remarks are added here.

d.3d displays three-dimensional images based on raster map layers.
NB: This is an interactive tool. Since building a view may take a long time, you should start by choosing a low resolution and line-frequency, determine the viewing angles, then increasing the resolution. Your view of an image depends on the definition of two points for which you must enter x, y and z coordinates: the viewpoint and the center of view. Note that if you change the 'vertical exaggeration' parameter, you should also adjust your viewing height.
NB 2: This tool yields incorrect height results when used in a LAT/LON

	Location (Ronnie Lassche, VU Amsterdam).
d.ask	prompts the user to select a GRASS data base file from among files displayed in a menu on the graphics monitor.
d.colormode	allows the user to establish whether a map will be displayed using its own color table or the fixed color table of the graphics monitor. **NB:** If you choose colormode float, all screen colors outside the current window may be displayed incorrectly.
d.colors	allows the user to interactively change the color table of a raster map layer displayed on the graphics monitor. **NB:** Whether these colors get displayed as intended on the screen depends on your setting of the colormode.
d.colortable	displays the color table associated with a raster map layer.
d.display	a menu-driven, highly interactive display program for viewing maps and producing final map products.
d.erase	erases the contents of the active display frame on the user's graphics monitor.
d.font	selects the font in which text will be displayed on the user's graphics monitor. **NB:** Some fonts cannot be used although they are displayed with **show.fonts.sh**.
d.frame	manages display frames on the user's graphics monitor.
d.geodesic	displays a geodesic line, tracing the shortest distance between two geographic points along a great circle, in a longitude/latitude data set.
d.graph	generates and displays simple graphics on the graphics display monitor.
d.grid	overlays a user-specified grid in the active display frame on the graphics monitor.
d.his	produces and displays a raster map layer combining hue, intensity, and saturation (his) values from user-specified input raster map layers.
d.histogram	displays a histogram in the form of a pie or bar chart for a user-specified raster file.
d.icons	displays points, as icons, at user-defined locations in the active display frame on the graphics monitor. **NB:** A sites file may be used as input for d.icons if you remove its header, |-symbols and site descriptions with the command *s.out.ascii*.
d.label	creates and displays text labels in the active display frame on the graphics monitor.
d.labels	creates/edits GRASS paint label files for display on the graphics monitor. **NB**: this tool yields incorrect labelsize results when used in a LAT/LON Location (Ronnie Lassche, VU Amsterdam).
d.legend	displays a legend for a raster map layer in the active frame on the graphics monitor.
**d.leg.thin*	Contributed code by Bill Brown (USA-CERL, ECS), to allow thinning of the number of legend entries of a raster map before display, drawing of a single legend box with a color gradient, and interactive placement of the box. Code available from **moon.cecer.army.mil**: grass/incoming. WARNING: not tested extensively. If you specify a horizontally placed smoothed legend box, no category values are printed - use *d.label* for that.
d.mapgraph	generates and displays simple graphics on map layers drawn in the active

	graphics monitor display frame.
d.measure	measures the lengths and areas of features drawn by the user in the active display frame on the graphics monitor.
d.menu	creates and displays a menu within the active frame on the graphics monitor.
d.mon	to establish and control use of a graphics display monitor. **NB**: When the CELL monitor is in use, you can't use the mouse nor set colormode to float (Ronald Wiemer, ARCHIS).
d.paint.labels	displays text labels formatted for use with GRASS paint (*p.labels, p.map*) output to the active frame on the graphics monitor.
d.points	displays point graphics in the active display frame on the graphics display monitor.
d.profile	displays profiles of a user-specified raster map layer.
d.rast	displays and overlays raster map layers in the active display frame on the graphics monitor.
**d.rast.arrow*	draws arrows representing cell aspect direction for a raster map layer. **NB:** This tool is still under alpha development.
**d.rast.edit*	allows interactive editing of the cell category values of raster map layers displayed on the graphics monitor. **NB:** This tool is still under alpha development. **NB:** You can speed up the editing process by performing it on a subimage only (using *d.zoom* and *r.mapcalc*) which is replaced in the original map with *r.patch* after resetting the region.
**d.rast.num*	overlays cell category values on a raster map layer displayed on the graphics monitor. **NB:** This tool is still under alpha development.
**d.rast.zoom*	zooms in or out of regions on a raster map displayed on the graphics monitor. **NB:** This tool is still under alpha development.
d.rgb	displays three user-specified raster map layers as red, green, and blue overlays inthe active graphics display frame.
d.rhumbline	displays the rhumbline joining two user-specified points, in the active frame on the user's graphics monitor.
d.save	creates a list of commands for recreating screen graphics.
[*d.savescreen*	screen capture utility for some MASSCOMP graphics monitors.]
d.scale	overlays a bar scale and north arrow for the current geographic region at a user-defined location in the active display frame.
d.sites	displays site markers in the active display frame on the graphics monitor.
d.text	draws text in the active display frame on the graphics monitor.
d.title	outputs a title for a raster map layer in a form suitable for display by *d.text*.
d.vect	displays GRASS vector data in the active frame on the graphics monitor.
**d.vect.cats*	displays user specified vector categories. **NB:** This tool contributed by James Farley, CAST/NCRI-SW. It can be found under /src.contrib.
**d.vect.dlg*	displays a USGS Digital Line Graph (DLG-3) binary vector file in the active frame on the user's graphics monitor.
**d.vect.lu*	uses a lookup table for line attributes to decide in what color a (line) vector should be displayed. **NB:** This tool contributed by Ronald Wiemer, ARCHIS, The Netherlands. It can be found under /grass/incoming. A readme file describes the format required for the lookup table.

d.what.rast allows the user to interactively query the category contents of multiple raster map layers at user-specified locations within the current geographic region.

d.what.vect allows the user to interactively query the category contents of a binary vector map layer at user-selected locations within the current geographic region.
NB: This tool appears not to be reporting area totals correcly under some circumstances (Christine Weiss, NYC Dept. Parks & Recreation). Use only with caution for this purpose.

d.where identifies the geographic coordinates associated with point locations in the active frame on the graphics monitor.

d.zoom allows the user to change the current geographic region settings interactively, with a mouse.

5.2 Paint Tools

Tools

GRASS commands starting with 'p.' are used for the painting of graphics on paper. During the composition phase of a paint-image, the screen is selected as output device with the command **p.select preview**. Hardcopy output on the Tektronix color printer consists of either paper sheets or transparencies for overhead projection, both A4-sized (8 x 11 inches). The currently available commands are listed below. When information about the tool is found lacking in the command description manuals, remarks are added here.

p.chart prints the color chart of the currently selected printer.
NB: this tool doesn't work.

p.colors allows the user to develop a color table that associates map categories with user-specified printer colors.

p.icons creates and modifies icons for map display and output.

p.labels creates labels for hardcopy maps. **NB:** Pascal Weidema, FRW-UvA, The Netherlands describes a method for creating hardcopy labels from an attribute file. Copy a /dig_att file to the /site_lists directory; edit the site list to conform to the following example: **xcoord|ycoord|attribute**. Use *p.map*, option **sites**, suboption **desc y** to get the site labels printed (the label is printed to the right of its coordinate). This method can also be used to print labels from site_lists created in any other way: just edit out the **#**-signs.

p.map hardcopy color map output utility.
NB: Using the interactive version of this tool, you may stumble over the 'Display site-labels **y**' option if the relevant site-list still contains '#' before the comment field.
NB 2: When using the preview painter (see the p.select command), the low screen resolution (usually 72 dpi) will make text and patterns look differently

	on the screen than it will in the hardcopy version. **NB 3:** Under some conditions the 'scale', 'comments', and 'colortable' options do not work. If 'p.map' thinks the driver can't handle scaled output it ignores any scaling request (painting just one panel) and doesn't produce a legend or header (Michael Shapiro, USA-CERL). To make this work okay, the ppm driver must be edited by specifying the output paper size and resolution, and by setting the number of vertical pixels to 'zero'.
**p.map.new*	hardcopy color map output facility. **NB:** This is a new version of *p.map*, still in alpha testing.
p.ppm	reads portable pixmap (ppm) files created by PPM utilities.
[*p.screen*	prints a graphics monitor display image file that has been saved by *d.savescreen*. **NB:** this tool is not available for RS6000.]
p.select	selects a device (printer) for GRASS hardcopy output. **NB:** Using *p.map* with the 'ppm' device results in a file called 'paint.ppm' in your home directory, unless you specify another directory by setting an environment variable called MAPLP. **NB:** if an error message appears when you try to change over from one device to another, it can temporarily be solved by editing the file **.grassrc** in your home-directory, deleting the line 'PAINTER=....'
**ps.icon*	creates and modifies icons for use with *ps.map*. **NB:** This tool is still under alpha development.
**ps.map*	hardcopy PostScript map output utility. **NB:** This tool is still under alpha development. **NB 2:** A bug causes the interactive use of this tool to be faulty. Use a prepared input file instead (Lars Schylberg, Dept of Geodesy and Photogrammetry, Royal Institute of Technology, Stockholm). **NB 3:** Printing masked vectors with this tool is only possible if you have specified a raster file to be printed. If you don't want the raster file to appear, use an all white raster map (Sue Huse, REGIS, UC Berkeley). **NB 4:** This tool cannot deal with LAT/LON vector data if the DEFAULT region is larger than 1 panel (Rosalie Buck).
**ps.select*	selects a PostScript device for GRASS hardcopy output. **NB:** This tool is still under alpha development.

5.3 Silicon Graphics 3d

One of the GIS displays (Meteoor) is powerful enough to allow the running of a special 3-d GRASS tool, SG3d, which has a graphics user interface and produces shaded 3d graphics of cell, vector, and sites maps. An SG3d manual is available.

5.4 Hardcopy Maps and Images

General

The most important thing you should know about getting hardcopy maps out of Grass is its relative inability to deal with vectors. That's why a separate program (MAPGEN) has been added to Grass's 'garden' - if you have it installed, refer to the MAPGEN manual for further help in getting hardcopy. Other hardcopy output by Grass is always of the raster type, which means that considerations of resolution become very important.

Any method that uses screen memory to create a hardcopy map (see the 'screendump' below) is automatically bound to the screen resolution (typically somewhere near 75 ppi), which may be much lower than what your printer is capable of. For instance, the Grass monitor called 'CELL' can be used to write a 3d image to a normal cell file. Thus the image becomes available to hardcopy producing tools such as 'p.map'. However, the CELL monitor will display and create files at screen resolution, unless you set the GRASS_WIDTH and GRASS_HEIGHT variables in the CELL driver to some higher number of rows and columns. By contrast, methods that use region settings to create a hardcopy map (such as 'p.map') do not suffer from these resolution problems.

A third problem arises with the conversion of map colors as you see them on the screen to the colors that are printed. Screen colors never exactly match the printer's, but that's only a small part of the trouble - see the discussion of colors under Grass in paragraph 8.2.

Tektronix Color Wax Printer

To send maps or images to the color printer, you have a choice between using the p.map tool or making a screendump.

For hardcopy output from p.map, you should run the p.select command and set PAINTER to "ppm". P.map will then produce a file called paint.ppm. This file must be converted to PostScript format with the command **pnmtops <ppmfile> > <psfile>**, and sent to the printqueue by **lpr -h -Pps <psfile> &**.

If you want to make a screendump, you can use the command **xwd | xwdtopnm**. This creates an X-window dump of the window that you indicate with a mouse-click. To see what you've dumped, use **xwud -in <xwd-name>**. The pnm-file can be converted to PostScript and sent to the printer with the command given in the previous paragraph.

IBM Page Printer / Digital Printserver

You can also send images to the black-and-white printers in the GIS room or at the uvabalie (Building B, room C 217). Prepare the images as black-and-white PostScript files and print them on the Page Printer with **lprint <filename> &**, on the Printserver with **lpr -Psaruvalps2 -Cuva/13 -J<yourname>**

<filename> &.

You can also send black&white screendumps to these printers.

5.5 Hardcopy Texts

IBM Page Printer / Digital Printserver

Texts can be sent to the Page Printer using the **lpr -h <filename>** command, and to the Printserver with the **lpr -Psaraps <filename>** command (if they are in PS format).

5.6 Exporting maps

5.6.1 Exporting Sites Lists

s.out.ascii converts a binary site-list file in GRASS to ASCII format.

5.6.2 Exporting Vector Maps

Tools

**v.out.arc* converts GRASS vector files to ARC/INFO's "Generate" file format.
NB: This tool is still under alpha development.
NB 2: If you specify text output, a .txt file is created which cannot be imported with ARC 'generate' directly. You first have to replace the original space delimiters with commas, then 'add' the file using Info. See also Kenn Gardels 46538-46566.
NB 3: Type out parameters completely ('polygon', 'line', etc).

v.out.ascii converts a binary GRASS vector map layer to an ASCII GRASS vector map layer.
NB: The file produced with this tool will show up any lines and areas deleted earlier, if these have not been removed by 'v.clean'. Deleted features are indicated by lowercase letters 'l' and 'a' instead of uppercase 'L' and 'A' (Sue Huse). See also Appendix C.

v.out.dlg converts binary GRASS vector data to binary DLG-3 Optional vector data

	format.
v.out.moss	converts GRASS site, line or area data into MOSS import format.

5.6.3 Exporting Cell Maps

The PBMplus package by Jeff Poskanzer will come in handy when you want to export Grass raster format to some other cell format not supported by Grass (see note on page .. paragraph 4.2)

Tools

r.out.ascii	converts a raster map layer into an ASCII text file.
**r.out.elas*	copies and converts a GRASS cell file to an ELAS raster file. **NB:** Tool contributed by Bruce Powell, US National Park Service. It can be found under /grass/incoming.

Cell file to Arc/Info

Two methods are available for exporting a Grass cell file to ARC/Info. The first uses ASCII format, the second uses binary format.

ASCII conversion:

- Be sure to uncompress the Grass cell file using **r.compress -u <name>** before trying to export it, since Grass automatically compresses all cell files.
- Export the cell file in ascii format using **r.out.ascii**, then change the header of this file to conform to the format required by Arc/Info's **asciigrid** (use Arc/Info help function for this if necessary).

Binary conversion:
(Information by Kenn Gardels, gardels@ced.berkeley.edu)

- Make sure the raster file is not a reclassed file, its name contains no periods and that no 'colr' files of the same name exist.
- Set the GISDBASE environment variable to your GRASS datapath.
- Start up ARC and issue the command **imagegrid <locationname:mapsetname:rastername> <gridname>**.
- Create a new shade file in ARC.

Cell file to Idrisi

(Thanks to Jim Hinthorne)

This conversion proceeds from the ability of IDRISI to import ascii files with one cell per line:

- Make sure the region is correctly set with **g.region rast=cellmap**.
- Create an ascii file with **r.stats -1c input=cellmap output=ascii.file**.
- Use the information in the /cellhd/cellmap file to tell IDRISI the correct number of rows and columns and the correct coordinates of the file.

6 DATABASE: RIM

6.1 Creating a database in RIM

RIM User's Manual

The RIM Users Manual provides extensive discussion of all RIM capabilities, and can be consulted in the GIS-rooms. Sections 3.3 to 3.5 discuss the creation and accessing of a database and the loading of data into that database.

Importing a Database

It is best to prepare your database as a fixed-format ASCII file. This can be read without problems by RIM; it is also a standard export format for most database programs (e.g. dBase).

6.2 Using RIM with GRASS

Tools

Three tools are used in the handling of database information from GRASS: s.db.rim to select sites from the database, and s.menu for further processing of the selected sites, and v.db.rim for the creation and management of RIM vector databases.

7 GRASS TOOLS

7.1 General Tools

Tools

GRASS commands starting with 'g.' are general data (file) management commands. The currently available commands are listed below. When information about the tool is found lacking in the command descriptio manuals, remarks are added here.

g.access	controls user access to the current GRASS mapset.
g.ask	prompts the user for the names of GRASS data base files.
g.copy	copies available data files in the user's current mapset search path and location to the appropriate element directories under the user's current mapset.
g.filename	prints GRASS data base file names.
g.findfile	searches for GRASS data base files and sets variables for the shell.
g.gisenv	outputs the user's current GRASS variable settings.
**g.gnuplot*	a scientific plotting program specifically adapted for GRASS. **NB:** This tool contributed by James McCauley, McCauley Technical Services, Indiana. Its source and a tutorial can be found under /grass/incoming. To compile this for use with the *gnuplot3.4* program, use the Gmakefile from pasture.ecn.purdue.edu: pub/mccauley/grass/Gmakefile.g.gnuplot.
g.help	GRASS help facility.
g.list	lists available GRASS data base files of the user-specified type to standard output.
g.manual	accesses GRASS User's Reference Manual entries.
g.mapsets	modifies the user's current mapset search path, affecting the user's access to data existing under the other GRASS mapsets in the current location. **NB**: the order in which you select the mapsets with this tool will determine which of any number of maps of the same name will be accessed by a Grass program.
g.region	program to manage the boundary definitions for the geographic region.
g.remove	removes data base element files from the user's current mapset.

g.rename	renames data base element files in the user's current mapset.
g.setproj	allows creation of PROJ_INFO and PROJ_UNITS files to record the projection information associated with the current location.
g.tempfile	creates a temporary file and prints the file name.
g.version	outputs the GRASS version number and date.

7.2 Point Tools

Tools

GRASS commands starting with 's.' are used for the manipulation and analysis of site data. Sites abilities include surface generation (interpolation), statistical operations, reclassification, transformation to raster and vector, coordinate registration, database searches and retrievals, proximity analysis, and import/export to the S statistical package. The currently available commands are listed below. When information about the tool is found lacking in the command descriptio manuals, remarks are added here.

**s.buffer*	generates circles of variable radius around sites from a site-list. Radii can be read from the site-list. **NB**: Contributed program by Janne Soimasuo (Forestry Fac., U of Joensuu, Finland).
s.db.rim	RIM data base management/query interface for GRASS.
**s.kvc*	randomly partitions sites into test/training sets. **NB:** This tool contributed by James McCauley. It can be found at pasture.ecn.purdue.edu, directory: pub/mccauley/grass, filename: s.kvc.tar.Z).
s.menu	accesses and manipulates GRASS sites data. **NB**: The option provided by this tool to write a sites selection to a new file cannot handle decimal points (Philip Verhagen, RAAP, Amsterdam).
s.surf.idw	generates a surface from sites data.
**s.surf.tps*	interpolates and computes topographic analysis from given site data, using spline with tension. **NB:** This tool contributed by Helena Mitasova. **NB:** This tool is still under alpha development. Defining a non-zero smoothing (~0.1) reduces overshoots during interpolation significantly and still preserves topography.
**s.univar*	calculates simple univariate statistics of sites. **NB:** This tool contributed by James McCauley, Dept. of Engineering, Purdue Univ., Indiana. It can be found under /grass/incoming or on pasture.ecn.purdue.edu in directory /pub/mccauley/grass, filename: s.univar.tar.Z.

7.3 Vector Tools

Tutorial

A tutorial is available to learn how to digitize and edit vector files using the v.digit tool.

Tools

GRASS commands starting with 'v.' are used for the manipulation and analysis of vector data. Vector abilities include digitizing, editing, labeling, measurements, import/export, full topological linking, display, patching, transformation to raster, grid generation, and contour labeling. The currently available commands are listed below. When information about the tool is found lacking in the command description manuals, remarks are added here.

**v.alabel*	bulk-labels unlabeled area features in a binary vector file. **NB:** This tool is still under alpha development.
**v.apply.census*	calculates/imports demographics from Census STF1 data. **NB:** This tool is still under alpha development.
v.area	displays GRASS area and perimeter information for GRASS vector map. **NB**: This tool currently works with UTM data only, and yields incorrect results when used in a LAT/LON location.
v.cadlabel	attaches labels to (binary) vector contour lines that have been imported to GRASS from DXF format.
**v.circle*	creates vector polygons of specified radius or area, based on a site list. **NB:** Tool contributed by Bruce Powell, US National Park Service. It can be found in /grass/incoming.
v.clean	cleans out dead lines in GRASS vector files.
**v.cutter*	polygon cookie cutter (Boolean and Overlay). **NB:** This tool is still under alpha development.
v.db.rim	RIM data base management/query interface for GRASS vector files. **NB:** running this tool without problems is rather difficult (see section 6.2).
v.digit	a menu-driven, highly interactive map development program used for vector digitizing, editing, labeling and conversion to raster format. **NB:** The empty options in the WINDOW menu ('display overlay map', 'display backdrop CELL map') can be activated in the CUSTOMIZE menu.
v.digit2	identical to *v.digit* in Grass 4.1.
v.mkgrid	creates a binary GRASS vector map of a user-defined grid. **NB:** A slightly enhanced version of this tool, providing an option to rotate the grid, is available from /src.contrib/PURDUE. SysAdm info: it's author, James McCauley, warns that before compilation the binary name should be changed to v.mkgrid.contrib to avoid doubling links by MAKELINKS.
v.mkquads	creates a GRASS vector map layer and/or sites list and/or geographic region definition file for a USGS 7.5 minute quadrangle.
v.patch	creates a new binary vector map layer by combining other binary vector map layers. **NB**: to make a clean patch of two or more vector files with this tool, it may be

	useful to do a 'v.spag -i map=mapname threshold=0' to reduce double lines at the common boundaries, and a 'v.reclass -i' to remove common boundary lines altogether. Any labels problems resulting from this can always be solved later in 'v.digit' (Sue Huse). Alternatively, removal of boundary vectors ('neat lines') can be accomplished by using v.db.rim, because such lines have no attributes and are therefore not imported.
**v.proj*	allows projection conversion of vector files. **NB:** This tool is still under alpha development.
v.prune	prunes points from binary GRASS vector data files.
**v.reclass*	creates a new vector map layer based on a reclassification of categories in an existing vector map layer. **NB:** This tool is still under alpha development.
**v.report*	reports on properties of vectors (length, polygon area, numbers of sites within). Disregards current region and mask. **NB**: This is a program contributed by SCS. **NB 2**: This tool works only if the category file is either nonexistent or complete, i.e. contains 146 lines if '146' is the highest cat (Susan Stitt, National Park Service, GIS Division).
v.spag	processes spaghetti-digitized binary vector files.
v.stats	prints information about a binary GRASS vector map layer.
v.support	creates GRASS support files for binary GRASS vector data. See also the section on Grass Trouble. **NB:** By using the '-r' flag with this tool you can force a window around the data if you don't like the way Grass creates the smallest possible window. **NB 2:** Applying this tool to TIGER master files may take a long time. If you only need the geometric information from such files, you'd better extract just that part, and 'v.support' will run much faster.
v.to.rast	converts a binary GRASS vector map layer into a GRASS raster map layer. **NB**: This tool uses the current region settings during rasterization. **NB 2**: This tool will only create rasterized areas if these are labeled as such in the vector file; otherwise it will only rasterize labeled vectors.
**v.to.sites*	converts point data in a binary GRASS vector map layer into a GRASS site-list file.
v.transform	transforms an ASCII vector map layer from one coordinate system into another. **NB**: Since this tool uses linear transformation, it does not recognise LAT/LON input.
v.trim	trims small spurs, and removes excessive nodes from a binary GRASS vector file. **NB:** A bug is reported which mistakenly deletes polygon islands if a 'trim factor' of 0.0 is chosen.
**v.what.vect*	a tool for querying vector files, modified from d.what.vect, to be found in grass/incoming.

7.4 Raster Tools

Tutorials

Tutorials are available for the r.weight, r.combine, r.infer, r.mapcalc and r.watershed tools. The tutorial by Parks (mentioned in section 1.2) also contains chapters on watershed analysis (chapter 6) and a number of other raster tools (chapter 5).

Tools

GRASS commands starting with 'r.' are used for the manipulation and analysis of raster data. Raster abilities include overlay (boolean, weighed, cellophane or rule based), full mathematical functions, filters, proximity analysis, measurements (length, area, volume, counts), clustering, import/export, line-of-sight, cost analysis, transformation to vector, rotation, patching, reclassification, network flow analysis, thinning, and elevation transformations. The currently available commands are listed below. When information about the tool is found lacking in the command description manuals, remarks are added here.

r.average	finds the average of values in a values map within areas assigned the same category value in a user-specified base map.
r.basins.fill	generates a raster map layer showing watershed subbasins.
r.binfer	Bayesian expert system development program.
r.buffer	creates a raster map layer showing buffer zones surrounding cells that contain non-zero category values.
r.cats	prints category values and labels associated with user-specified raster map layers.
r.clump	recategorizes data in a raster map layer by grouping cells that form physically discrete areas into unique categories.
**r.coarsen*	provides more options than 'r.resample' for resampling. **NB:** Tool offered by Jianping Xu (jianp@ocean.rutgers.edu; Center for Remote Sensing and Spatial Analysis, Rutgers University, NJ).
r.coin	tabulates the mutual occurrence (coincidence) of categories for two raster map layers.
r.colors	creates/modifies the color table associated with a raster map layer. **NB:** the color table 'grey.eq' stretches the color histogram, not the cell values.
r.combine	allows category values from several raster map layers to be combined. **NB:** Darell McCauley reports a bug in this tool, in that the option not to use the graphics monitor cannot be used.
r.compress	compresses and decompresses raster files.
**r.contour*	creates a vector map of specified contours from a GRASS raster map layer. **NB:** This tool is still under alpha development.
**r.cost*	outputs a raster map layershowing the cumulative cost of moving between different geographic locations on an input raster map layer whose cell category values represent cost. **NB:** This tool is still under alpha development.
r.covar	outputs a covariance/correlation matrix for user-specified raster map layers.
r.cross	creates a cross-product of the category values from multiple raster map layers.

r.describe	prints terse list of category values found in a raster map layer.
**r.digit*	allows on-screen digitizing of a raster map layer. **NB:** This tool is still under alpha development.
r.drain	traces a flow through an elevation model on a raster map layer. **NB**: After installation of Grass update1, this tool can take more than one pair of coordinates on the command line or in an input raster file.
r.grow	generates an output raster map layer with contiguous areas grown by one cell.
r.infer	outputs a raster map layer whose category values represent the application of user-specified criteria (rules statements) to other raster map layers' category values.
r.info	outputs basic information about a user-specified raster map layer.
**r.line*	converts a raster map layer into a vector map layer. **NB:** This tool is still under alpha development.
r.los	line-of-sight raster analysis program.
r.mapcalc	raster map layer data calculator.
r.mask	establishes or removes the current working mask.
**r.mask.points*	examines and filters lists of points constituting lines, to determine if they fall within the current region and mask and, optionally, an additional raster map. **NB:** This tool is still under alpha development.
**r.median*	finds median of values in a cover map within areas assigned the same category value in a specified base map. **NB:** This tool is still under alpha development.
r.mfilter	raster file matrix filter. **NB**: If you specify a divisor using this tool, '0' has the same effect as '1' (Jim Hurst).
**r.mode*	finds mode of values in a cover map within areas assigned the same category value in a specified base map. **NB:** This tool is still under alpha development.
r.neighbors	makes each cell category value a function of the category values assigned to the cells around it, and stores the new cell values in an output raster map layer.
r.patch	creates a composite raster map layer by using known category values from one or more map layers to fill in areas of "no data" in another map layer.
r.poly	extracts area edges from a raster map layer and converts data to GRASS vector format.
r.profile	outputs the raster map layer values lying on user-defined lines.
r.random	creates a raster map layer and site list file containing randomly located sites.
r.reclass	creates a new map layer whose category values are based upon the user's reclassification of categories in an existing raster map layer.
r.report	reports statistics for raster map layers.
r.resample	GRASS raster map layer data resampling capability.
r.rescale	rescales the range of category values in a raster map layer.
r.slope.aspect	generates raster map layers of slope and aspect from a raster map layer of true elevation values. **NB**: Alternative methods for calculating slope and aspect are described in the r.mapcalc tutorial by Shapiro and Westervelt.
r.stats	generates area statistics for raster map layers.
r.support	allows the user to create or modify raster map layer support files.

r.surf.contour	surface generation program. **NB**: This tool produces a terracing effect (Jo Wood, U. of Leicester, UK) and may take a very long time to run.
r.surf.idw	surface interpolation utility for raster map layers.
r.surf.idw2	surface generation program.
**r.thin*	thins line-like features in raster map layers, to be used as input for *r.line*. **NB:** This tool is still under alpha development.
r.traj.data	reviews the ammunitions and weapons database used by *r.traj*.
r.transect	outputs raster map layer values lying along user-defined transect lines.
r.volume	calculates the volume of data "clumps", and (optionally) produces a GRASS site-list file containing the calculated centroids of these clumps. **NB**: This tool yields incorrect volume results when used in a LAT/LON Location (Ronnie Lassche, VU Amsterdam).
r.water.outlet	watershed basin creation program.
r.watershed	watershed basin analysis program for use in GRASS 4.1.
r.watershed4.0	watershed basin analysis program for use in GRASS 4.0.
r.weight	raster map overlay program.
**r.weight2*	weighted overlay raster map layer analysis program. **NB:** This tool is still in alpha development.
r.what	queries raster map layers on their category values and category labels.

7.5 Imagery Tools

Tutorial

A tutorial is available on image processing, discussing all imagery tools.

Tools

GRASS commands starting with 'i.' are used for the manipulation and analysis of imagery (multi-spectral) data. They allow the user to read remotely sensed images from LANDSAT multi-spectral scanner (MSS) and thematic mapper (TM) and SPOT, and to read scanned maps or photographs. Imagery abilities include tape handling for multiple formats, scene and sub-scene extraction, color enhancements, image registration, terrain correction, automated classification, user directed classification, filtering, histograms, and full integration with raster operations.

Imagery data generally come in multiple BANDS that are read into GRASS as ordinary cell maps, with row and column numbers functioning as x/y coordinates. The bands of an image are kept related by GROUPING.

Imagery is unregistered and unrectified; it should be processed BEFORE rectification because this causes information loss (through resampling). However, mosaicking (patching multiple images into one coordinate system) should only be done AFTER rectification, because the images may have differing distortions which must be removed first. The complete processing sequence for rectifying imagery is: i.group, i.target, i.points, i.rectify.

The currently available commands are listed below. When information about the tool is found lacking in the command description manuals, remarks are added here.

i.cca	canonical components analysis (CCA) program for image processing.
i.class	generates spectral signatures for an image by allowing the user to outline areas of interest.
**i.cluster*	generates spectral signatures for land cover types in an image using a clustering algorithm. **NB:** This tool is still under alpha development. **NB 2:** This program may contain a bug in the 'class separation value:' option. **NB 3:** This tool requires at least two input bands, but may be fooled by a 'dummy' band containing 1's. **NB 4:** You can find more explication of the i.cluster output by Katarina Johnsson 22144-22161, Michael Shapiro 26393-26459, Hong Zhuang 22144-22161, and about clustering in general by Peter Sincak 28197-28221.
i.colors	creates colors for imagery groups.
i.composite	creates a color composite image from three imagery band files specified by the user. **NB:** This tool uses a 6x6x6 color cube, thus resulting in a file with 216 categories.
i.fft	fast Fourier Transform (FFT) for image processing.
**i.gensig*	generates statistics for *i.maxlik* from raster map layer. **NB:** This tool is still under alpha development.
**i.gensigset*	generates statistics for *i.smap* from raster map layer. **NB:** This tool is still under alpha development.
i.grey.scale	interactively assigns a histogram contrast stretch grey scale color table to a raster map layer. **NB:** this tool does not change the cell values, just the color table.
i.group	creates and edits groups and subgroups of raster imagery files.
i.his.rgb	hue-intensity-saturation (his) to red-green-blue (rgb) raster map color transformation function.
i.ifft	inverse fast Fourier Transform (IFFT) for image processing.
i.maxlik	classifies the cell spectral reflectances in imagery data based on the spectral signature information generated by *i.cluster* or *i.class*. **NB:** This tool uses the covariance as well as the means to determine whether a pixel belongs to a given class. **NB 2:** I.maxlik considers non-symmetric covariance matrices (such as may be generated by i.cluster, but not by i.class) and negative eigenvectors are invalid (ill-conditioned) and ignores them (Michael Shapiro).

	NB 3: The 'reject' map generated by i.maxlik can be used for single class classifications. John Isaacson at USA-CERL used i.class to extract the signature of one visible archaeological site, ran i.maxlik on it, then used the values in the reject map to determine where other such sites might be located.
i.median	interactively creates a raster map layer whose color table is based on the red, green and blue color values present in existing, user-specified imagery group files. **NB:** This tool issues a warning when no histogram information is available for one of the input files; use 'r.support' to create one. **NB 2:** This tool combines three files into one 8-bit (256 color) file; to display the result you should therefore use 'd.colormode float'.
**i.ortho.photo*	interactive imagery function for the ortho-rectification of imagery group files. **NB:** This tool is still under alpha development.
i.pca	principal components analysis (PCA) program for image processing. **NB:** To approximate the first component in a PCA for four input images, use the formula (band1 + band2 + band3 + band4) / 4.
i.points	enables the user to mark coordinate system points on an image to be rectified and then input the coordinates of each point for creation of a coordinate transformation matrix.
**i.quantize*	interactive imagery function that creates a raster map layer with a color table based on the RGB values present in existing specified imagery group files.
i.rectify	rectifies an image by computing a coordinate transformation for each cell (pixel) in the image using the transformation coefficient matrix created with *i.points*. **NB:** This tool does not work with LAT/LON data.
**i.rectify2*	similar to *i.rectify*. **NB:** This tool is still under alpha development.
i.rgb.his	red-green-blue (rgb) to hue-intensity-saturation (his) function for image processing.
**i.smap*	imagery function that performs contextual image classification using sequential maximum a posteriori estimation. **NB:** This tool is still under alpha development.
i.tape.mss	extracts Multispectral Scanner (MSS) imagery data from half-inch tape.
i.tape.mss.h	extracts header information from LANDSAT MSS imagery data stored on half-inch tape.
i.tape.other	extracts scanned aerial imagery (NHAP, etc.) and SPOT imagery from half-inch tape.
**i.tape.spot*	extracts SPOT imagery from half-inch tape. **NB:** This tool is still under alpha development.
i.tape.tm	extracts LANDSAT Thematic Mapper (TM) imagery from half-inch tape.
**i.tape.tm.fast*	extracts LANDSAT Thematic Mapper (TM) imagery from tape media. **NB:** This tool is still under alpha development.
**i.target*	establishes a GRASS target location and mapset for an imagery group. **NB:** This tool is still under alpha development.
**i.vpoints*	identifies coordinate pairs of points from a vector map or keyboard entry and corresponding points in an image. **NB:** This tool is still under alpha

development.

i.zc zero-crossing "edge detection" raster function for image processing.

7.6 Shell-scripting

Make Your Own Tools

Shell-scripts are little programs that combine GRASS-commands with commands to the AIX Operating System. Essentially, scripts just contain a list of all the commands that you could have entered on the keyboard, but which are now executed by the operating system. They can be compared to 'macros' on the PC.

If you want to do something complicated, or repetitious, or taking a lot of time, the script can do it faultlessly and it won't get tired. Learning to write scripts is not difficult; just read the *UNIX Shell Programming* manual (available in the GIS-rooms). You will have to learn to use the Unix file editor VI to do the actual writing of the scripts (see section 8.1).

Some example scripts can be found in *Grass Clippings*, the quarterly newsletter. Scripts contributed by GRASS-users all over the world can be found in GRASS's ftp directory under "src/related".

8 TROUBLESHOOTING

8.1 Unix Trouble

Backups

Weekly backups of all user data are made by the Systems Manager. If you want more frequent backups or want to transport your data to a sytem that cannot be reached through the networks, contact the Systems Manager. You will be ask to provide the proper means of storage (diskette, tape or other) yourself.

File Size

There is a limit to the size your files may have in the computer, the so-called **ulimit**. The default ulimit is about 4.2 Mb. So if you are using maps with a lot of cells you may run into this particular problem, which causes process interrupts and all kinds of warnings. If you're serious about using large files, ask the Systems Manager to change your ulimit.

Compressing

When you're moving data around over the network, it may be to your advantage to compress your files first. GRASS files are automatically compressed before they are stored, but other files (e.g. export files) are not compressed. You can use the **compress** command.

Background Procedures

You will find that some procedures in GRASS allow the user to continue working while they are running. This is called "running in the background". Not all GRASS-tools use it, so if you want to perform some time-consuming operation and do not want to wait for it to finish, you can run it in the background by using the UNIX command **&**. By placing the & after the command and before the <enter>, the command is placed in the background. A process number will be displayed on your screen

and, by giving an <enter>, you are returned to the GRASS prompt. You will not be warned when the process is finished.

VI

Any file editing in UNIX must be done with the standard editor VI. This is a bit complicated to work with, so here is some advice on how to use it. You start by creating a file with **vi filename**. Your screen is then filled with ~-signs in the first row. The cursor is on the first line and in so-called *command mode*. This means that anything you type will be interpreted as a command by VI. Your first command must be to change into *edit mode*, in order to start writing. You do this by typing an **i** (for insert). Now you can start writing whatever text you have in mind.

The important thing to remember is, that VI will not do anything for you without receiving a specific command. For instance, it will not break off a line once it has reached the right side of the screen (usually at position 80); it will just "wrap around" to the next line. If you want to end a line, you will have to give an <enter>. Equally, if you have made a typing error, you will have to change over to command mode by pressing <Esc>, then move the cursor with the arrow keys, then delete (with **x**) or replace (with **r**) the mistake.

Changing over continually from edit mode to commmand mode will take some time to learn. You had best start by reading the pages on VI in the AIX Reference Manual (Vol. 2), so that you will have an idea what VI can and can't do.

8.2 GRASS Trouble

Section 3.6 of this manual discusses problems occurring during GRASS startup. This chapter will focus on problems occurring during the use of GRASS, and includes problems related to dependent hard-and software. The list will be updated as new problems occur and old ones are solved.

Supporting vector files

(Thanks to: Susan Stitt, US National Park Service, GIS division)
v.support may give three types of error warnings. WARNING: "label x matches another label x" found in the same area or WARNING: "label x matches a label y" in the same area indicate that the labels file contains two records for the same area. The first type of warning may be disregarded, because GRASS will effectively use only one of the labels. The second type of warning is more serious, in that it indicates that two different labels for one area were found. This should alert you to problems in the topology (eg, open polygons) that you must correct using *v.digit*.
The third type of warning: WARNING: "PNT_TO_AREA failed" indicates that Grass was unable to find

an area at the stated location. This may be caused either by errors in topology (open polygons) or by the area being too small relative to the scale of the map (this is set in the first *v.support* screen).
All double labels will disappear after editing if, on leaving *v.digit*, you answer 'yes' to the prompt: compress Atts file?

Reclassified Maps

Cell maps that are reclassifications (made with the *r.reclass* tool) of other maps can only be processed as long as the base map exists. If the base map is deleted or renamed, the reclassification still exists but can no longer refer to the base map and therefore cannot be processed. Thus it is important to know which maps are base maps and which are reclassifications.

Renaming and Removing Maps

GRASS provides tools (*g.remove* and *g.rename*) for the renaming and removing of map layers together with all their support files. Do not rename or remove GRASS map layers in UNIX! You are likely to forget some support file and leave it lying around. However, note that some GRASS files leave you no choice, since you can only remove them in UNIX. This is the case with imagery group signature files, for instance.

Color tables

The color table structure of GRASS gives rise to many misunderstandings. The default color table for each raster map is stored in the $LOCATION/colr directory. If another color table is specified by the user, this will be stored in the $LOCATION/colr2 directory. GRASS colr files have different formats depending on whether the colors are specified distinctly or as a ramp between categories. If two categories are mentioned on one line, with a from: and a to: category, a color ramp is made between. Furthermore, if the R,G,B values specified are equal GRASS stores only one value. (Sue Huse, REGIS, Univ Calif Berkeley).

Screen colors

Unfortunately, you may not get all or even some of the colors you specified in the color table to display on the screen. In view of claims that your screen can display a Zillion (24-bit) colors or, more modestly, at least 256 (8-bit) colors, this is rather disappointing.

Although those in the know do not agree on the subject entirely, it seems clear that you don't have more than 256 colors at any time during a Grass session. Bill Baker (bakerwl@corral.uwyo.edu) guesses that the XDRIVER sets the number of colors Grass knows by looking at a so-called 'hardware default', which is either 'PseudoColor' (8-bit) or 'TrueColor' ('DirectColor', 24-bit) displays. Kenn Gardels adds that presently Grass requires a PseudoColor type visual, which is why you have only 256 colors. But

it gets worse...

Of those 256 colors, Peter Briggs (pbriggs@laurel.ocs.mg.edu.au) says that GRASS reserves 16 (numbers 241 - 255) for the drawing of vectors (these are the colors that you keep having to choose from when you display sites, vectors, or text). In principle this leaves another 240 colors, still a respectable number. But... ...this is where the 'd.colormode' tool comes in. There are two modes ('fixed' and 'float') that determine how many of those 240 colors actually are available for display. The most generous mode ('float') uses a 6x6x6 color cube (don't ask me why) so that's 216 colors you've got left. The 'fixed' mode leaves you with a paltry number (16?) of colors. And Simon Cox (simon@cerberus.earth.monash.edu.au) says that even if he uses colormode float he can't get more than 65 colors to display. It also seems to him that normally, only part of the 6x6x6 color cube is used, and this is not optimised for the dataset that is displayed.

Data management

One problem the user will have to deal with as soon as the GIS project attains a certain size, is to manage the data that are produced. This problem appears in two ways: first comes an increasing difficulty of keeping track of the various data layers in a mapset and a location, later you will find that you want to have more information at hand about the history of particular map layers.

In all but the simplest projects, you will soon find yourself producing large amounts of map layers that are often of temporary significance only. In the beginning you will be able to store and find th erelevant map layer by apt naming conventions, but this strategy will fail after you've acquired about a hundred data layers. Grass provides no means of grouping map layers under an explanatory header or in a directory tree. The best advice therefore is to make good use of the possibilities that are present in Grass to efficiently store and retrieve data layers: the mapset, the data layer description, and the map name.

- a mapset is a means of keeping a group of maps separate from other maps in the same location. You can always acccess maps in other mapsets by judicious use of the g.mapsets tool, so there is no need to be stingy and do all the work in one mapset. To begin with, you shoud use the PERMANENT mapset in each Location to store your base maps, ie the basic map layers that you will use to create your project from. Don't use this mapset for storing non-basic map layers such as reclass layers, masks, temporary layers etc. Create a new mapset for each major part of your project.
- the header file of a cell data layer, and the header of a vector data layer, contain lines for the description of that layer. Always use it to provide a concise description! You can make use of it to search your mapbase for specific subjects and words (see below).
- The map name itself provides a way to group data layers, because Grass will list data layers in alphabetical order. So by giving maps that belong together names that start with the same letter, they are also listed together. This strategy certainly works with temporary map layers that you produce during a project, which you know that you don't want to save afterward: you name them 'tmp1', 'tmp2' etc, which makes it easy to remove these maps later on. You can also try to have the map name reflect the map history ('soil.clump.dist' reflecting that the map is the result of a distance analysis of a clumped version of the soil map), but this approach will break down with the more involved analyses.

Suppose you stick to these rules, then a good way of retrieving a shortlist of maps that you are looking for is the command 'g.list -f rast| grep clump', which will search through all the raster data layer names and descriptions that you have, and select the ones that have the string 'clump' in them.
To provide for the need for more data about GRASS data, one possible solution would be to create an extensive map history file. You already know that the /hist directory contains a short map history for cell data layers. No such facility exists for sites or vector data layers. As a stopgap, one could of course create and update a text metadata (data about data) file. At USA-CERL they have a 'blank' text of about 5 pages called a "GRASS map layer history report" and a "GRASS imagery layer history report" which you might copy and update as required. But such a text file is not supported by GRASS and therefore the user is responsible for the updating, moving, renaming and other actions to ensure that the metadata stay with the layer they describe. USACERL does not think this is a viable solution, but at the same time it recognizes the need for metadata. A slight improvement can be realised by putting the metadata file, named 'metadata', for cell map layers under the /cell_misc/<map>/ directory. There it will be automatically dealt with by the g.remove, g.rename, and g.copy tools. See also the script by Darell McCauley that creates metadata files (appendix D).

Mapset Access

A great deal of confusion appears to exist among beginning users of Grass about the various meanings of the word "access". Greg Ewing initially confused the grass 'mapset access' with the unix 'file and directory access' permissions. To have (write) access to a mapset means that you must be the directory **owner** in Unix. 'Group' or 'Other' permissions are not enough. Chris Rewerts at Purdue University argues that the original intent was for Grass users to share the same login account (and Location?) but to have their own mapsets. For multiple users this doesn't really work, he says. Instead, his site uses symbolic links (ln -s) between a permanent database and each user's workspace, putting the PERMANENT mapset into each user's search path with 'g.mapsets'. Laxmi Prasad Musunur adds that, as long as a user works under the same Location as the PERMANENT Mapset, he can make use of its data.

INDEX

INDEX A: On this manual

INDEX B: On Grass Clippings

The following list was created from the copies of GRASS Clippings (Vols. 3 - 6) available to me on 01/10/92:

Keyword	TITLE	AUTHOR	Volume, Number, Page
Accuracy, of digitizer	Graphic Digitizer Accuracy	Dominic Giovachino	6,2,19
Analysis, of topography	New Capabilities for Interpolation and Topographic Analysis in GRASS	Helena Mitsukova	6,2,13
Archaeology	GRASS Applications For Archaeological Inquiry	-	4,2,11
Bar scales	Bar Scales and Histograms	Melissa Records	6,1,4
Boundary determination	Growing GRASS	Fred Limp	6,1,15
Color tables, toggling	Growing GRASS	Fred Limp	6,1,15
DEM, processing	Processing DEM Data	Susan Stitt	4,1,1
G.average	Growing GRASS	Fred Limp	4,1,20
G.help	g.help - An On-Line Help System for GRASS 4.0	Deb Lister	6,1,14
G.report	Growing GRASS	Fred Limp	4,2,15
Fort Lewis	GRASS Used For Forest Management At Fort Lewis	-	3,3,15
Fort Lewis	GRASS Provides Historical Overview of Fort Lewis	Teresa Hansen	6,1,7
Help, on-line	g.help - An On-Line Help System for GRASS 4.0	Deb Lister	6,1,14
Histograms	Bar Scales and Histograms	Melissa Records	6,1,4
Icons, multi-colored	Growing GRASS	Fred Limp	5,2,15
Images, processing	Image Processing in GRASS	Hong C. Zhuang	6,2,10
Images, enhancement	Videographic Enhancement of GRASS Images	Bob Sullivan	6,2,4
Interface, DBMS	New GRASS Seeds	Bill Goran	4,2,7
Interface, IRDMIS	IRDMIS / GRASS / ISM Interface Programs	-	5,2,1
Interface, ISM	IRDMIS / GRASS / ISM Interface Programs	-	5,2,1
Interface, RIM	GRASS / RIM Interface Programs in GRASS 4.0	David Satnik	6,1,5
Interface, SEEPPAGE	Natural Resources Using GRASS and SEEPPAGE	-	5,2,1
Interpolation	New Capabilities for Interpolation and Topographic Analysis in GRASS	Helena	

			Mitsukova 6,2,13
Jamaica	Jamaica's UWI-MAGIC	David M. Johnson	6,2,1
Labeling, interactive	Interactive Labeling	-	3,3,11
Legends	How to make a Legend on a GRASS Display Map	Melissa Records	5,2,13
MAPGEN	GRASS-MAPGEN: Expanded Plotting Capabilities in the GRASS Environment	-	5,1,2
Modeling, hydrologic	GRASS For Hydrologic Modeling	-	4,2,1
Modeling, land cover	New GRASS Seeds	Bill Goran	4,2,7
Output, hardcopy	Growing GRASS	Fred Limp	6,1,15
Output, plotting	GRASS-MAPGEN: Expanded Plotting Capabilities in the GRASS Environment		- 5,1,2
Output, patterns	p.map Patterns	-	3,3,16
P.map, patterns	p.map Patterns	-	3,3,16
R.mapcalc	Growing GRASS	Fred Limp	6,2,15
Scripts, g.average	Growing GRASS	Fred Limp	4,1,20
Scripts, g.report	Growing GRASS	Fred Limp	4,2,15
Scripts, variability	Growing GRASS	Fred Limp	3,3,10
SG3d	SG3d Upgrade Provides Easier Positioning, Lighting	Bill Brown	6,2,1
Stereo pairs	Stereo Pairs	-	4,1,24
Tutorial, TIGER	GRASS / TIGER Tutorial Available with GRASS 4.0	Jacinda Mountz	6,1,10
Variability	Growing GRASS	Fred Limp	3,3,10
Wetlands	Wetlands Data In GRASS Vector Format	-	4,1,11
Xgen	Using Xgen in GRASS 4.0	Christine Poulsen	6,1,21

APPENDIX A

Introduction to the Office of Grass Integration (OGI) GRASS Mailing Lists

[This document is sent out to the GRASS Users' Electronic Mailing List once a month as a reminder and introduction to the list manager software. The current version dates from November 25, 1991]

The Office of GRASS Integration has set up a service that will be of interest especially to those GRASS users who have access to electronic mail. We have set up two e-mail lists to foster communication between GRASS Users and GRASS Programmers. The lists work on the principle of "mail exploding" -- a user mails a note to one address, and the computer at that address
"explodes" the letter, re-mailing it to everyone who subscribes to the list. Replies to the original letter generally go to the list as well (if there is general interest in the answer), but can be directed solely to the original author if desired. Both lists are also automatically preserved in an archive, and users may at any time request a file containing an archive of list messages.

These lists exist to foster communication between parties interested in the GRASS software; many CERL employees will participate in the lists, but on an infrequent and informal basis. The list will be maintained and stored on CERL computers; however, CERL will not officially monitor the content, intent, or accuracy of any messages that pass through the list. The point-of-contact for the list is:

grass-lists-owner@moon.cecer.army.mil

The list for GRASS users and friends is called "grassu" (short for "GRASS users"), while the list for programmers and system-level users is called "grassp" (short for "GRASS programmers"). Topics for discussion on the **grassu-list** may include questions about various GRASS applications, sources of (or reviews of) third-party support, and various and sundry other experiences with GRASS. **Grassp-list** topics will include discussion of the algorithms and intricacies of current GRASS programs, programming hints and ideas for new GRASS applications, and other prorgrammer-oriented issues. Note that the grassp-list is **not** the best (nor even an appropriate) place to report suspected bugs in GRASS code; the current <bug.report.sh> program and correspondance with OGI are the preferred problem-report procedures.

Accessing the mailing lists

Each mailing list has two addresses. The first is the destination of messages that should be "exploded" out to all subscribers. This address takes the form of

grassu-list@moon.cecer.army.mil

for the user's list, and

grassp-list@moon.cecer.army.mil

for the programmer's lists[10].

[10] These, and all other e-mail addresses in this article, are provided in an "Internet-standard" format; different host computers might require different specifications of the address. The system administrator at each site should be able to help if the addresses here do not work as specified.

The other important address for each list is the destination of commands to the list-management software (such as subscribe, unsubscribe, etc.). They should be sent to

grassu-request@moon.cecer.army.mil

or

grassp-request@moon.cecer.army.mil

as appropriate. Multiple commands may be included in messages. Commands may be located in the subject line or in the body of the letter, but there should be only one command per line. Upper- or lower-case letters may be used for commands, and they may be abbreviated to simply the first three letters if desired. All commands are acknowledged with return e-mail.

Commands that the server will accept are:

HELp This command mails back a copy of the list's help file from the archive directory.

PINg This command "pings" the mailing list, causing a simple "I'm alive" response to be returned. This is a good way to check the list's address, to check if the list is reachable from your machine, and to check if the list is up and running. This command is meant to be an alternative to mailing a message to the everyone on the list asking "Is the list working" and so forth.

SUBscribe [name]

This command adds [name] to the mailing list. The subscriber's e-mail address is taken from the "From:" line of the subscription letter, and this address is what is added to the list. The [name] parameter is optional -- it is used to specify the user's real name. If [name] is simply a dash (-), the WHO command (see below) will not display the user's e-mail address in a list of subscribers.

UNSubscribe

This command removes the user's subscription from the list. It relies on the "From:" field of the request letter. If this address does not match any name in the list, the software will notify the sender and the CERL list-maintainer, who will resolve the problem by hand.

WHO [address]

If no [address] option is given, this command returns a sorted list of list subscribers (minus all users who subscribed with the "-" option). If [address] is supplied, the software searches for a user with that address and returns that user's name.

DIRectory

This command returns the mailing list's archive directory. The primary purpose of this is to see which list-archive files are available. The list software will mail back a list of available files.

GET [file]

This command returns the named [file] from the list's archive directory. Case and spelling are important; the name must **exactly** match a file shown by the "DIR" command. A message acknowledging the request will be mailed to the user.

If the file is short, the user will simply receive a copy of it in mail. For example, the command "GET LOG" (capitalization of "LOG" is important) will cause the list-server to mail back a copy of the most recent messages to the list (which are always stored in the file named "LOG").

If the file is very long, the list server software will <uuencode> the file and split it into several mail-sized chunks. ("Uuencode" and "uudecode" are common UNIX programs to encode files for e-mail transport. The local computer must have the <uudecode> program in order to decode these files. The acknowledgement

note will inform the user how many chunks to expect, and each chunk will have a subject line that specifies "filename: part X of Y".

Examples of List Usage
To "ping" the list (i.e. check that it is functioning and reachable from your computer):

Send a mail message to:

grassu-request@moon.cecer.army.mil

In the subject or the message body, type "PING" or "ping" or "pin" (without the quotes; case is unimportant and only the first three letters of commands are significant). The list server will reply with a brief "I'm here" message. To ping the programmer's list, substitute "grassp" for "grassu" in the address above.

To subscribe to one of the lists:

Send a mail message to either

grassu-request@moon.cecer.army.mil
or
grassp-request@moon.cecer.army.mil

with "subscribe [yourname]" in the subject or body. Replace [yourname] with your real name. The list will add the e-mail address that it finds in the "From:" field of your letter.

Summary
These two lists are provided by OGI as a service for the GRASS community; we hope that they will prove to be useful resources. We encourage you to try several commands to experiment with the lists. If there are any questions, feel free to send mail to

grass-lists-owner@moon.cecer.army.mil

or contact the OGI by phone or mail.

APPENDIX B

Introduction to the vi editor

(Based on the article 'Vi, een editor met mogelijkheden' by Mia Robbemont in *SARA Bulletin* 119 (september 1992))

Vi, the standard text editor in UNIX, is not very user-friendly. Yet it is very versatile, which is why it is the preferred editor for writing shell-scripts, editing your Grass files, and other daily tasks. This appendix will introduce you to the use of **vi** and list most of its functions.

1) The first thing you need to know about **vi** is that it operates in either of two modes: the **command mode** or the **input mode**. This means that the text you type will mean different things depending on the mode. For example, typing 'a' in command mode means 'append' and enables you to start typing text into the file, while 'a' in input mode results in the letter 'a' being added to the file. Most problems of beginning users with **vi** are direct consequences of confusing the command mode with the input mode. Let's open a file and see how this works.

2) To open a file in **vi**, type **vi <filename>**. If you opened a new file, you will see a mostly empty screen with a tilde (~) at the beginning of each line. **Vi** is now in the command mode. To make things easier on yourself, type the command '**:**'. This lands your cursor at the bottom line of the screen, where you can enter commands that will affect the whole or part of the document you're editing. More on this later, but for now type the command **set showmode** and return to the document by pressing **ESC**. You will now see the text 'INPUT MODE' appear each time you're entering the input mode of **vi**.

3) The **ESC** key will always put you back into command mode (pressing it more than once will result in beeps). We now have an open document <filename> and want to start filling it with text. There are many commands that will put you into input mode ('a' being just one possibility), but '**i**' for 'insert' is the basic one. You will now see the 'INPUT MODE' message appear, and everything you type will be taken as text input. **Vi** has no automatic return at the end of each line, so if you want a line to end, you must press **ENTER**.

4) When you're finished with your text, and assuming you have made no mistakes in it, you can save it and exit **vi** by entering the command mode (with **ESC**, remember?) and typing **:wq** (for 'write quit') or, if you want to impress others with your knowledge of **vi**, with **:x** (for 'exit'). Saving the text while continuing **vi** is done with **:w** or **:w <newname>** if you want to save the text under a different name than <filename>. Leaving **vi** without saving is done with **:q!**.

5) Now you know enough about **vi** to create your own files, it's time to delve a little deeper and find out about this versatility thing... Supposing you discover that you have made some mistake in the text, you will (a) want to move to the offending spot and (b) edit out the mistake.

6) Moving through the text is certainly not the same as text input, which is why you should enter command mode before moving. Although moving with the arrow keys and the PGup and PGdown keys while in input mode is possible, you'll find that it isn't straightforward. So use the **ESC** key before going anywhere.

Small moves are performed by lowercase letters (**h** left, **l** right, **k** up, **j** down; **w** to the beginning of the next word, **e** to the end of the current word, **b** to the beginning of the current word or to the previous word) and signs (**0** to the beginning of the line, **$** to the end of the line, **+** or **ENTER** one line down). Larger moves are performed by uppercase letters (**H** to top left of screen, **M** to middle line of screen, **L** to last line of screen, **G** to last line of document) and by Ctrl-key combinations (**Ctrl-u** up half a screen, **Ctrl-d** down half a screen, **Ctrl-b** up a screen, **Ctrl-f** down a screen). If you happen to know the line number of the line you want to move to, simply type it after the colon (eg, **:20**) and presto, you're there.

It is also possible to search the text for strings in the command mode. This is done by typing **/string** (for forward searches) or **?string** (for backward searches). The cursor will land at the first occurrence of 'string' it finds.

7) Editing a text probably means you'll have to delete some parts, replace or move others, and add yet others. Deleting text is lots of fun, unless you make a mistake (especially with the more powerful commands). To delete the character under the cursor, type **x** (Remember, if you type **:x** it means save and exit, so be careful!). If you've more than one character to delete, there's **dw** to delete the (rest of) the word, **D** to delete the rest of the line, **dd** to delete the whole line. Some more options are listed at the end of this appendix. By combining a number with one of the lowercase commands you can have it repeat itself (thus, **7dd** will delete the current line and six more).

Replacing text is done with **r** (to replace a single character), **R** (to replace the current line until **ESC** is pressed), **cw** (to replace the current word), and **s** (to replace the current character by a string until **ESC** is pressed). If you find all this manual replacing laborious, 'search and replace' commands are discussed later on. Adding text is done with the **i** and **a** commands discussed earlier, with their uppercase counterparts **I** (insert text at the beginning of current line) and **A** (append text at the end of current line), and with **o** or **O** (open new line below resp. above current line).

Replacing and adding can be automagically repeated as already mentioned for deleting, by prepending the number of times you want to perform the command. To join one or more lines below the current line to the current line (ie deleting the **ENTERs** that create a new line), type **<number>J**.

8) Typing a colon in command mode gets you into the *line editor*. We have already seen a number of examples of how the line editor can be used. In general, colon commands are used to define settings (such as **:set showmode**), to do text searches and replacements (such as **:20**), and to perform document and shell functions (such as **wq)**. These subjects are discussed in some more detail below.

Some of the more useful settings are **:set number** (to see the document with line numbers), **:set all** (gives a listing of all available settings), **:set warn** (warns you if you try to leave **vi** without saving), **:set ic** (ignore case) and **:set ai** (do automatic indent). All of these settings are turned off with no-settings (eg **:set nonumber** turns off line numbering).

Text searches and replacements after the colon make use of a powerful syntax, the base of which is an expresion of the following type:

from,to substitute / <search_pattern> / <replace_pattern> / options.

You may already know this pattern from other editors, notably UNIX sed and ed. An example of a search-and replace command is **:5,$s/this/that/gc**, which searches lines 5 to end of document for all occurrences (option **g**lobal) of the string 'this' and asks for your confirmation (option **c**onfirm) that these be replaced by the string 'that'. The from,to part is optional and may be replaced by **%** if you intend to search the whole document. The search string may contain special characters such as the asterisk, dot, dollar and others - their uses are listed below. To maintain the difference between a search for a special character and a search making use of that special character, the first category is always preceded by a backslash. Thus, if you want to replace all '\' with 'backslash', you must have the search string preceded by a backslash, like this: **3,23s/\\/backslash/g**. As you see, entering a search string can become a complex task in itself.

Lastly, colon commands include **:r <document>** (to include <document> in your document), **:!<command>** (to issue a Unix command while temporarily leaving **vi**), **:!sh** (to open a Unix shell and use it until Ctrl-D is pressed), **:nu** (returns the current line number).

8) If you want to be considered the ultimate **vi** guru at your site, you should learn about buffers in **vi**. Most of what you delete, replace or move in **vi** ends up in one of a number of buffers. Normally you're not even aware that these exist, but one day the time will come that you or someone else inadvertently deletes something very important. That's why I've added a list of buffer commands below.

Finally, if you mastered all of the above and below, but don't know the answer to a question that is put to you by a **vi** novice, try to look bored and tell them to read the online **vi** documentation in Unix (**man vi**).

General:
To repeat the last command given, type **.**.
To undo the last command given, type **u**.
To repeat a string search, type **n**.
To get information about the document, type **:f** or **Ctrl-g**.

Deleting:

d*t c*	deletes from current position to character *c*.
dL	deletes from current position to bottom of screen.
dM	deletes from current position to middle of screen.
dH	deletes from current position to top of screen.
d$	deletes from current position to end of line.
d0	deletes from current position to beginning of line.

Copy and paste:

<number>y	'yanks' or copies current plus <number> lines.
p	pastes whatever is yanked at the current position.
<number>dd	deletes <number> lines.
p	pastes whatever is deleted at the current position.
:*a,b*t*c*	copies lines *a* to *b* behind line *c*.
:*a,b*m*c*	moves lines *a* to *b* behind line *c*.

Search strings:

^	beginning of line
$	end of line
.	any character or digit
*****	any number of characters or digits
[abc]	a, b, or c
[^abc]	anything but a, b, or c
[a-zA-Z]	any character
[0-9]	any digit
[a-zA-Z]*	any number of characters
[^a-zA-Z]	anything but a character

Replace strings:

&	places the search string in the replacement string

Buffer commands:

"5p	pastes whatever you deleted five deletions ago back into the current position.
"a<number>y	places current plus <number> lines into a buffer called 'a'.
"ap	places content of buffer 'a' at cursor position.

APPENDIX C

Grass file formats and conversions between them.

raster

Grid cell files are a central element in GRASS. GRASS grid cell data are stored as a matrix of grid cells. Each grid has an absolute spatial reference in terms of its position and size. Each cell is assigned a single integer attribute value called the category number. Each cell is stored in the file as one to four 8-bit bytes of data. The physical structure of a cell file can take one of three formats: uncompressed, compressed or reclassed. The uncompressed cell file actually looks like an *NxM* matrix. Each byte or set of bytes for multi-byte data represents a cell of the map layer. The physical size of the file, in bytes, will be *rows x columns x bytespercell*. The compressed format uses a run-length encoding schema to reduce the amount of disk required to store the cell file. Run-length encoding means that sequences of the same data value are stored as a single byte repeat count followed by a data value. A reclass map layer does not contain any data, it contains references to another map layer along with a schema to reclassify the categories of the referenced map layer. The reclass file itself contains no directly useful information. The reclass information is stored in its cell header file.

In addition to the cell data file itself, there is a number of support files of the grid cell map layer. These files are: grid cell header file, map layer category information, map layer color table and map layer history information. These files, which together comprise a grid cell map layer, all have the same name, but each resides in a different database directory.

The cell file itself has no information about how many rows and columns of data it contains, or which geographical part of the layer it covers. This information is in the cell header file. The format of the cell header depends on whether the map layer is a regular map or a reclass layer. The regular map layer cell header contains information describing the physical characteristics of the cell file. It has the following fields: projection, UTM zone, geographic boundaries, resolution, format containing bytes-per-cell information and compress indicator. A cell header for a reclass cell file contains the name of the referenced cell file and the category reclassification table.

The cell category file contains a title for the map layer, the largest category value which occurs in the data, and a one line label for each category. The cell color table file contains one line of color description for each category of data. The colors are represented as levels of red, green, and blue. The cell history file contains historical information about the cell file like creator, date of creation, comments, etc. The history file is generated automatically along with the cell file[11].

cell	- contains the cell values of a raster map in binary format. Unreadable.
cellhd	- contains a text header file for a raster map.
cell_misc	- contains 'extra information' for a raster map. This is organised in directories under a directory named after the map.
range	- contains the minimum and maximum values present in a raster map.
histogram	- contains a listing of the number of cells for each category in a raster map.
colr/colr2	- contains a description of the color table to be used when displaying a raster map. 'colr' contains a standard color table; 'colr2' is optional and contains a user-defined color table.

vector

dig_ascii	- contains a text vector map. The file begins with a header, then follows a listing of all vector coordinates in the map, with codes indicating the type of vector and

[11] This section copied from Fedra, K. and M. Kurbat, Hybid Geographical Information Systems. In *EARSeL ADVANCES IN REMOTE SENSING* Vol. 1, No. 3 - VII, 1992, p 90.

number of coordinates.

dig - binary vector file. Unreadable.

dig_att - contains attribute information for a vector map

dig_cats - contains category information for a vector map

dig_plus -

sites

site_lists - contains coordinates and description of a site (point). The full format is:

Sites need the # symbol preceeding the comment string if they are to be interpreted as cell values. This is important for tools such as *s.menu* option 6 and *s.surf.tps*.

Conversions

Sites to cell:
By using *s.voronoi* with the -l option and rasterizing afterwards, you can create an isochor cell map out of a sites file (Darell McCauley).

Vector to cell: use the v.to.rast tool.

Cell to sites: use the r.stats tool.

Cell to vector: use the r.thin and r.line tools to convert linear raster features to a vector map; use the r.poly tool to convert raster area features to vector areas.

APPENDIX D

Shell scripts

script name	**description**
shade.clr.sh	may work only with grey-scale images.
v.autotiger autotiger.run v.tiger.feat	Scripts by Malcolm Williamson (malcolm@cast.uark.edu), to facilitate extraction of Census data from tape. FTP from cast.uark.edu: outgoing. Don't forget the README.tigerscripts file which explains what the scripts do.
-	Script by Bill Jackson (USA-CERL), to click on an icon and get a description.
v.reg	Script contributed by Phillip Chaney, to simplify the process of registering a USGS 7.5' Quad map.
r.distance.sh	Script by Lars Schylberg, to compute shortest distance between two interactively determined data clumps.
-	Script by Lars Schylberg 46161-46174, to determine neighbours for each clump in a map.
intens.sh	Script by Dave Gerdes 3688-3736.
clean	Script by Ronald Thomas 40571-40591, to clean up archived mail.
shade.clr.sh	Script by Jim Westervelt 977-1137, to combine 3 shade relief maps in one RGB image.
m.export.arc.covers.sh	Script by Lars Schylberg 1510-1578, to facilitate exporting arcfiles for use by v.in.arc.
-	Script by Dave Gerdes 3916-3954, to create 360 degree aspect maps instead of 15-degree grouped aspect maps.
-	Script by Kenn Gardels 12643-12770, to force ascii dig files into an adjacent UTM zone.
-	Script by Lars Schylberg, to determine neighbours for each clump in a map.
-	Script by Greg Koerper 14355-14448, to streamline r.reclass operations.
-	Script by Michael Shapiro 14802-14826, to filter out small areas from the r.clump output.
-	Script by Lars Schylberg, to copy raster maps from the current location to another location.
-	Script by Chris Rewerts 25134-25164, to calculate the standard deviation of cell values in a map (zie ook s.univar).
-	Script by Michael Shapiro 26252-26283, to calculate min, max, mean, mode, and SD (zie ook s.univar).

-	Script by Michael Shapiro 44403-44422, to remove 1-cell islands.
r.prox.sh	Scripts by Lars Schylberg 37933-38079 and 38081-38206, neighborhood filters based on resp. 4 and 8 neighbors.
-	Xgen Script by John Cartwright 36626-36708, to receive user input via text entry and slider object.
-	Time loop script by Fred Limp. 10E.
label.obj.sh	Script by Lars Schylberg, to autopaintlabel raster areas resulting from r.clump. 10J10.
digatt_2_site	Script by Sue Huse to convert dig_att files to site files. 10J10.
s.labels	Script by Sue Huse to convert site files to paint labels file, to be used with d.paint.labels. 10J10.
r.meta	Script by Ronald Thomas, to create a raster metadata file, can be got from picea.cfnr.colostate.edu:pub/r.meta.Z.
-	Script by Lars Schylberg, to calculate mode for neighborhoods and multiple layers. Email to: larss@fmi.kth.se.
cp.colr.sh	Script by Lars Schylberg, to copy a color table from one raster map to another.
xgen script	Script by Paul Loechl (USA-CERL), to serve as an example script for XGEN users.