The Geographic Information Systems: GRASS HOWTO

David A. Hastings

U. S. Department of Commerce
National Oceanic and Atmospheric Administration
National Geophysical Data Center

Boulder, CO 80303
USA
dah@ngdc.noaa.gov

Revision History

Revision 1.0	13 November 1997	Revised by: dah
Initial Release		

This document describes how to acquire, install and configure a powerful scientific public−domain Geographic Information System (GIS): the Geographic Resources Analysis Support System (GRASS). It provides information on other resources for learning more about GRASS, GIS in general, for acquiring data, etc.

This document also encourages the Linux community to consider enhancing this software as a major application of UNIX/Linux. ("When will Linux become bundled with public domain or Linux Public License 'killer apps'"?) For more on this topic, see Section 8 below.

Table of Contents

1. What is a GIS? 1

2. What Is GRASS? 2

3. A Brief History of GRASS 3

3.1. A Re−Invogorated GRASS Management Model 5

3.2. Continued Assessment of Future GRASS Management 6

4. System Requirements for GRASS 7

5. How to Acquire GRASS 8

6. How to Get GRASS Running on Your Linux−based Computer 9

7. Web−based Support for GRASS (and for GIS Matters in General) 11

8. The Future of GRASS? 13

9. Copyright Notice, and Notice on Support of this Document 15
- 9.1. Copyright notice 15
- 9.2. Notice on support of this document 15

10. References 16

A. Acquisition/Installation of GRASS4.13 Binaries 17

B. Acquisition/Installation of GRASS4.1.5 Binaries 19

C. Acquisition/Compilation of GRASS Source Code 21

C.1. GRASS 4.2 Quick Start 23

D. Enhancing GRASS 28

E. Sample Files 29

1. What is a GIS?

There are many ways to describe a Geographic Information System. Here are three working definitions (from David A. Hastings, 1992, Geographic Information Systems: A Tool for Geoscience Analysis and Interpretation):

1. (The minimal definition): A GIS is a hardware/software system for the storage, management, and (with hardcopy or screen graphic) selective retrieval capabilities of georeferenced data. Definitions like this one are often used by vendors and users of vector–only GIS, whose objective is sophisticated management and output of cartographic data.
2. (A parallel definition): A GIS is a hardware/software system for managing and displaying spatial data. It is similar to a traditional Data Base Management System, where we now think in *spatial* rather than in tabular terms, and where the "report writer" now allows output of maps as well as of tables and numbers. Thus we can consider a GIS a "spatial DBMS" as opposed to traditional "tabular DBMSs." Few people use this definition, but it might help to explain GIS to a DBMS user.
3. (A more aggressive definition): A GIS is a system of hardware, software, and data that facilitates the development, enhancement, modeling, and display of multivariate (e.g. multilayered) spatially referenced data. It performs some analytical functions itself, and by its analysis, selective retrieval and display capabilities, helps the user to further analyze and interpret the data. Properly configured, the GIS can model (e.g. synthetically recreate) a feature or phenomenon as a function of other features or phenomena which may be related – where all features or phenomena are represented (characterized) by spatial and related tabular data. The analytical objectives described here are sometimes controversial – and often given lip service by cartographic GIS specialists who have not yet seen what can be accomplished scientifically by a select few GISs that go beyond cartographic approaches.
4. Another definition can be found at *the University of Edinburgh.*

---

2. What Is GRASS?

GRASS (Geographic Resources Analysis Support System) is a public domain raster based GIS, vector GIS, image processing system, and graphics production system. Created by the US Army Corps of Engineers, Constriction Engineering Research Laboratory (USA/CERL) and enhanced by many others, it is used extensively at government offices, universities and commercial organizations throughout the world. It is written mostly in C for various UNIX based machines. Linux is one of its more robust implementations.

GRASS contains over 40 programs to render images on monitor and paper; over 60 raster manipulation programs; over 30 vector manipulation programs; nearly 30 multi−spectral image processing manipulation programs; 16 data management programs; and 6 point file management programs.

GRASS' strengths lie in several fields. The simple user interface makes it an ideal platform for those learning about GIS for the first time. Users wishing to write their own code can do so by examining existing source code, interfacing with the documented GIS libraries, and by using the GRASS Programmers' Manual. This allows more sophisticated functionality to be fully integrated within GRASS.

Other strengths include GRASS' pioneering of mixed resolutions in a data base, mixed geographic coverage areas in a data base, raster image compression techniques via run−length encoding and reclassification lookup tables, GRASS' rescaling of display images on the fly to fill the display screen, plus its fundamental design criterion of powerful computer−assisted scientific analysis of environmental issues (as opposed to merely going for intricate cartographic output of relatively simple processes).

GRASS is usually supplied as free, non−copyright source code to be compiled on host machines. Some compiled binaries are also easily obtainable at no cost via the Internet. It runs on a variety of UNIX platforms.

Copied from Project Assist *Intro to GRASS*.

3. A Brief History of GRASS

In the early 1980s the U. S. Army Corps of Engineers' Construction Engineering Research Laboratory (USA/CERL) in Champaign, Illinois, began to explore the possibilities of using Geographic Information Systems to conduct environmental research, assessments, monitoring and management of lands under the stewardship of the U. S. Department of Defense. Part of the motivation for this action was new responsibility for the environment encoded into the National Environmental Policy Act of the late 1970s.

Bill Goran of USA/CERL conducted a survey of available GISs, assuming that he could find several systems capable of environmental analysis, from which he could select one or more to recommend for use by CERL and perhaps others in the Department of Defense. However, he was surprised to find no GIS that satisfied his needs. What started as a selection process turned into a design exercise for his own GIS development program.

USA/CERL hired several programmers, and began by writing a hybrid raster−vector GIS for the VAX UNIX environment. This made the team one of the first to seriously develop GIS for UNIX. Though they still faced challenges with different versions of UNIX, they developed procedures of coding in ANSI standard UNIX, avoiding "tweaking" the code toward any particular vendor−specific flavor of UNIX.

GRASS developed a programming style characterized by:

- Use of UNIX libraries where possible, plus the creation of GRASS libraries for repeated GIS−specific acts such as opening raster files that might be compressed (by run−length encoding) or not.
- The ability to handle both major GIS data types: raster and vector.
- The favoring of raster data processing, as scientific analysis was easier to encode with raster (than for vector) data models.
- The ability to handle raster grids of mixed grid sizes in the same data base. This was a departure from raster's image processing tradition of requiring identical (and perfectly registered) grid cell arrays in each and every data layer.
- The ability to handle raster grids with different areas of coverage. Again, this was a departure from raster tradition of having all grids be identical in geographic coverage.
- The ability to run−length encode raster data files, in order to greatly reduce file sizes of most files.
- The separate structure of reclassification files. Such files merely contained a look−up table noting the previous and new classes. This is MUCH more compact than replicating the original grid with different numerical values. A reclassified file of a 100x100 km square area of 10 metre grid cells would be a few hundred bytes, rather than 100 megabytes of uncompressed 8−bit raster data.
- The acceptance of de−facto standard data models. While competitors created cumbersome (and in many cases secretive) data formats, GRASS accepted the de−facto standard Digital Line Graph vector format and unheaded binary raster grid format. GRASS later abandoned DLG as its internal vector file format, and let its raster format evolve. However, DLG and the unheaded binary raster grid are still routinely handled formats for GRASS, and its new formats are as open as its previous ones.
- GRASS code was managed in several directories. Initial contributions were placed in the src.contrib directory. More solid code was moved to the src.alpha directory. After remaining in the src.alpha for one full release cycle, the code, with resultant bug fixes, moved to the most honorable level, the src directory.

GRASS was overseen by three levels of oversight committees. USA/CERL kept the ultimate responsibility for GRASS. It implemented most GRASS development, and carried out the day−to−day management of

GRASS testing and release. The GRASS Interagency Steering Committee (GIASC), comprised of other Federal agencies, met semi−annually to review development progress, and evaluate future directions for GRASS. (Academic and commercial participants in GRASS also attended GIASC meetings; only part of each meeting was "Federal−Agencies−only." GRASS eventually became nominally and officially a "product" of the GIASC, though everyone recognized USA/CERL's leadership role. The GRASS Military Steering Committee met periodically to review the progress of GRASS in serving its original intent: meeting the Department of Defense's needs to evaluate and manage the environment of military lands.

The public interacted with CERL and GIASC through USA/CERL's GRASS Information Center. GRASS Beta testing was very widespread, and quite intensive for the leading users of GRASS. Several leading users, such as the National Park Service and the Soil Conservation Service, selected GRASS as its prime or only GIS. They made significant commitments to enhance and test GRASS, yet considered this investment well worth their while. They said that they had more influence over the direction of GRASS than they would over any known alternative system. They also felt that, despite their major efforts and expenses in supporting GRASS, they had a bargain in relevant power for the dollar.

Several universities adopted GRASS as an important training and research environment. Many conducted short−courses for the public, in addition to using GRASS in their own curricula. Examples of such leading academic users of GRASS are Central Washington University, The University of Arkansas, Texas A & M University, The University of California at Berkeley, and Rutgers University.

Though GRASS received some criticism (some say) for being so good and so public, it was also reputedly borrowed from liberally by some developers of other systems. Though the first group might have viewed it as unfair competition, the second group may have noted that it was not copyright, and was a valuable testbed for GIS concepts. GRASS received an award from the Urban and Regional Information Systems Association (URISA) for quality software in 1988.

As CERL and GRASS evolved through the late 1980s and early 1990s, CERL attempted to cut overhead costs associated with supporting the public domain version. It created and initially funded the Open GRASS Foundation, in cooperation with several of the leading users of GRASS. The Open GRASS Foundation has since evolved into the Open GIS Consortium, which is aiming for more thorough interoperability at the data and user interface level, but appears not to be taking advantage of the major open GIS testbed (GRASS).

In 1996 USA/CERL, just before beginning the beta testing for GRASS version 5.0, announced that it was formally withdrawing support to the public. USA/CERL announced agreements with several commercial GISs, and agreed to provide encouragement to commercialization of GRASS. One result of this is *GRASSLANDS*, a commercial adaptation of much of GRASS. Another result is a migration of several former GRASS users to COTS (Commercial Off−The−Shelf) GISs. However, GRASS' anonymous ftp site contains many enhancements to the last full version 4.1 release of GRASS. Many organizations still use GRASS feeling that, despite the lack of a major release in five years, GRASS still leads the pack in many areas.

3.1. A Re−Invogorated GRASS Management Model

In late 1997, a group at Baylor University took the lead in developing a new Website for GRASS. This quickly developing Website contains GRASS 4.1 source code and Sun Solaris binaries, GRASS 4.1 documentation, and an on− line manual. By November 1997 this site posted the first version of GRASS 4.2 source code and binaries currently for Sun Solaris) with Linux and Windows NT under consideration). GRASS 4.2 incorporates several enhancements from the CERL website, plus some of Baylor's own enhancements. Documentation for GRASS 4.2 is appearing; the group encourages cooperation in further development of GRASS, and is looking for partners. It hopes to use increased use of the World Wide Web in developing and managing GRASS. GRASS 5 development and compilation is underway. The site also links to the Blackland GRASS site at Texas A & M University, for those desiring very inexpensive access to GRASS for Windows 95.

3.2. Continued Assessment of Future GRASS Management

Note: An ad−hoc group (which includes myself) is exploring the basic issue of continued, reconfigured, yea perhaps increased, value of GRASS as a public test−bed for GIS technology. It is exploring shepherding the testing and release of GRASS5.0, and exploring possibilities for a more distributed management model for GRASS design and development. It is exploring the universe of public domain spatial data processing software (including geographic information and image processing systems), and perhaps tabular data base management systems. How can such knowledge be (1) optimized as an open, public test bed for such technology and (2) better used by the public? Might this involve a Linux management model, perhaps? See Section 8 for more discussion on this topic.

4. System Requirements for GRASS

Minimum requirements include:

- 8 Mbytes of memory (of course, more is better..)
- 100 Mbytes of free disk space
- ~40 mb for executables,
- ~40 mb for source code (which you can ignore if you merely install the Linux binaries)
- ~? for data (the veritable bottomless pit can be filled with data, if you so choose)

GRASS runs on Linux kernel versions as old as 1.2.13 (see more information in the appendices for various specific binaries).

GRASS will run in text mode. However, for displays of data, you will need X. If you are still running a version of X, it will probably work with GRASS.

If you find any other hardware/OS requirements that should be mentioned, please let me know!

5. How to Acquire GRASS

GRASS used to be available on tape from various companies that signed distribution agreements with USA/CERL. These companies usually supported specific platform environments, such as Masscomp, Sun, DEC, Hewlett Packard, IBM (risc), PC (running some flavor of UNIX), and Macintosh (running AUX). Until recently, the flavors of UNIX working on PCs generally were too low−end, or required too much added programming support (e.g. programming drivers for high−end graphics boards like the Number Nine boards of several years back) to be stable or complete. However, with robust systems like Linux, this problem is history. Similarly, few people acquire GRASS on tape, though a few do on CD−ROM.

The main way to acquire GRASS is to get it via anonymous ftp from:

1. The new site at *Baylor University*

   As of the date of this version of the mini−HOWTO, Baylor has source code for GRASS 4.1 and 4.2, as well as Sun Solaris compiled binaries. Blackland GRASS for Windows 95/NT is linked to from this site. Baylor is considering its own Linux and Windows NT binaries, as well. You should be able to compile the Baylor source code under Linux yourself, using information in this mini−HOWTO.

2. *The traditional site* at *USA/CERL* or from mirrors cited at USA/CERL's website:

   The ftp location is:

   moon.cecer.army.mil

   Appendix A describes how to acquire and install GRASS4.13 compiled binaries from USA/CERL. (See section 6 before installing GRASS!)

   Appendix B describes how to acquire and install GRASS4.15 compiled binaries from USA/CERL. (See section 6 before installing GRASS!)

   Appendix C describes how to acquire and compile GRASS4.14 and GRASS4.15 source code from USA/CERL, as well as GRASS4.2 source code from Baylor University. (See section 6 before installing GRASS!)

   Linux distribution developers! Might you be interested in including GRASS with your distribution? Remember, GRASS source code is in the completely unrestricted, copyright−free, public domain. Your distribution might be more valuable if it contained source code and/or compiled binaries for GRASS.

6. How to Get GRASS Running on Your Linux−based Computer.

Appendices A, B, and C describe how to acquire and install GRASS. Before actually installing GRASS, you will have to decide where to put three parts of the system:

1. The GRASS binaries, source code (if you install this), man pages, documentation, and the like. Many folks put this stuff off /usr/local (e.g. /usr/local/grass/bin, /usr/local/grass/src).
2. The GRASS executable and gmake utilities. Some folks put this stuff off /usr/local (e.g. /usr/local/grass/grass4.1 and gmake4.1 or /usr/local/bin/grass4.1 and gmake4.1).
3. The GRASS data directories. These can go anywhere, as they are specified in configuration files.

   I have used a different scheme for a decade. As GRASS code, binaries, and the like (except data owned by users) are all owned by the special user "grass" I don't want this stuff to get spread around my system. I create a new directory (usually on a separate file system) called /user, and put all my GRASS stuff below this. For example:

```
/user/grass4.1/bin   (I usually put grass4.1 and gmake4.1 here...)
              /data
              /dev
              /etc
              /man
              /src
              /src.alpha
              /src.contrib
```

   I'm currently building a GRASS5.0 site, which then goes under:

```
/user/grass5/bin
            /data   (some GRASS5 data formats have changed...)
            /dev
            /etc
```

   The GRASS Installation Guide (described in Section 10 and in Appendix C) is useful for getting GRASS running, even if you merely install the binaries as described in Appendices A and B. Please don't overlook one important detail: Most GRASS installations separate user from software manager accounts and UNIX permissions. You should create a "grass" (the quotes here are for emphasis, and should not be part of the actual user userid) user account on your workstation. All installation and configuration of grass should be done by user "grass". Untar (or un"cpio" files, run setup configuration utilities, run Gmakefiles (GRASS versions of makefiles), and edit configuration files as user "grass." Then only RARELY run GRASS as user "grass." (I only run GRASS as user "grass" when I am making archival data files in the PERMANENT mapset.) This is done for much the same reason as not running user software as user "root". YOU CAN DO TOO MUCH DAMAGE AS USER "grass"!

   Beyond the instructions in these appendices, and information in the GRASS Installation Guide, you have some additional housekeeping to do, such as developing a data base. You can acquire sample data bases from USA/CERL (directory pub/grass/grass4.1/data at anonymous "ftp moon.cecer.army.mil"), start from scratch following instructions in the GRASS Programmer's Manual (and, to a lesser degree, buried in the functional descriptions of the GRASS User's Reference

Manual).

I personally recommend that you start with the Spearfish and Global databases available from USA/CERL:

a. The Spearfish data base covers two 7.5 minute topographic sheets in the northern Black Hills of South Dakota, USA. It is in the Universal Transverse Mercator Projection. It was originally created by Larry Batten (now of the Environmental Systems Research Institute's office in Boulder, Colorado) while he was with the U. S. Geological Survey's EROS Data Center in South Dakota. The data base was enhanced by USA/CERL and cooperators. It is an excellent, and well−used (there are many training materials available for GRASS with this data base) example of a county−scale GIS project in the UTM projection.
b. The Global data base was developed by Bob Lozar of USA/CERL to prototype a latitude−longitude "projection" data base in GRASS for global environmental study and decision support.

Starting with these two examples, you can build your own data bases in UTM and latitude−longitude projections. (Note, many people don't call latitude−longitude a projection. Others disagree, saying that anything that transfers the Earth's surface to two dimensions is a projection.. We'll stay away from that debate here. Needless to say, lat−lon is treated as other projections are by the computer program.)

---

7. Web−based Support for GRASS (and for GIS Matters in General)

Support for a public domain program? No way, they say! Actually, as a user of Linux, you probably know better.

GRASS started by having a GRASS Information Office at USA/CERL. There were also very active users outside USA/CERL, who provided valuable user support. GRASS had annual users' meetings, listservers for users and developers, etc. Companies provided value added support services on a contractual or fee basis.

Various people have developed valuable books and training materials on GRASS. Several universities used to conduct training courses in GRASS. I don't know how many of these are continuing. If training courses interest you, try asking on the usenet newsgroup comp.infosystems.gis (see below for more on this newsgroup).

Valuable "books" available on the Internet are noted in the References (Section 10).

World Wide Web−based training materials, including training in GRASS, are highlighted in the *CyberInstute Short Course in GIS* (then scan down for the link(s) to Web−based tutorials in GIS).

One of the better GRASS tutorials is *Project Assist's − Intro to GRASS*

Other good sites:

Central Washington University was an *early GRASS user and training facility*

Starting the hunt for mostly free spatial data by Stephan Pollard This is based at the Center for Advanced Spatial Technology of the University of Arkansas, another early educator with GRASS.

Purdue University has several *GRASS features*

USA/CERL's online GRASS manual

Rutgers University's GRASS Information Center at the *Center for Remote Sensing and Spacial Analysis*

The REGIS project at *The University of California at Berkeley* has a Linux version of GRASS available via ftp, and also has *a Web−based version of GRASS called GRASSLINKS.*

After getting trained by the books and Web−based tutorials noted just above, where do you turn to for specific advice???

Probably the best source of support these days is usenet newsgroup *comp.infosystems.gis* If you're not familiar with newsgroups, ask your network administrator or Internet service provider. comp.infosystems.gis contains modestly heavy traffic on such topics as

- "how do I find data on this topic for this area?"
- "how do I convert these data for use in my Aardvark GIS?"
- "how do I get this function to work in my Aardvark GIS?"
- "which GIS can I use to solve my particular problem?"

GRASS used to be one of the top GISs discussed on this group. Traffic in GRASS is dropping slightly, as its user community matures. However, there are usually answers to your questions, if you post them. You might also do a "power search" on subject:GRASS [& your own subject of interest here?] and newsgroup:comp.infosystems.gis in *DejaNews* to see what might appear from the usenet archives.

---

8. The Future of GRASS?

Excellent question! Several possible answers have been thrown out:

1. USA/CERL's announced intention is to use GRASS and COTS (commercial off−the−shelf software) for internal uses, to leave the GRASS public web− and ftp−site on its system indefinitely, and to sign cooperative research and development agreements with three companies: (1) the Environmental Sciences Research Institute (ESRI), (2) Intergraph, and (3) Logiciels et Applications Scientifiques (L.A.S.) Inc. The first two agreements encouraged the incorporation of GRASS concepts into ESRI's and Intergraph's commercial GISs. The third encouraged the adaptation of GRASS' concepts and code into a new commercial GIS by L.A.S. L.A.S. also offered to encourage the continuation of a public domain GRASS, as a viable stand−alone system and as a potential source of new ideas and code for L.A.S.'s GRASSLAND. One observer noted that the first two agreements might be akin to someone signing Linux over to Microsoft. The same observer considers the experiment by/with L.A.S. to be an interesting possibility − an attempt to keep viable public domain and commercial versions of GRASS.
2. Some people believe that GRASS will wither without USA/CERL's central management. Some believe that the Open GIS Consortium will successfully guide industry into an open architecture that will benefit all developers and users. Others believe that OGIS' effort will lead to a cacophony of almost similar (but not quite interoperable) vendor−specific "standards," so the loss of GRASS as an open development platform will be felt sorely.
3. Some people believe that developments on some campuses and other sites may result in those institutes keeping GRASS for awhile, but in non−standard forms. In short, GRASS will undergo "cell division" and lead to a cacophony of internally valuable, but externally unused, GISs.
4. Others hope that GRASS' previous management model under USA/CERL has left it ready for a new model. Perhaps:

   a. Under a new mentor, such as NASA (which needs an open, powerful and scientific, GIS integrated with image processing system for its Earth Observing System).
   b. Under a distributed management model... perhaps somewhat like Linux?
   c. Perhaps a bit of a hybrid? Perhaps a Web−based effort could spawn a series of usenet discussion groups beginning with

      ◊ comp.infosystems.gis.grass, and evolving to:
      ◊ comp.infosystems.gis.grass.academics
      ◊ comp.infosystems.gis.grass.publicservice
      ◊ comp.infosystems.gis.grass.commercialvalueadded
      ◊ comp.infosystems.gis.grass.commercialdistributors
      ◊ comp.infosystems.gis.grass.programming
      ◊ comp.infosystems.gis.grass.users
      ◊ comp.infosystems.gis.grass.centralcommittee

   Clearly the topics are a bit tongue−in−cheek. However, under this model, a Central Committee (including representation of academic, public service [government and nongovernmental organizations], commercial distributors and value added firms, programmers, and users) would guide overall grass development and testing. The other special interest groups would serve their user communities. Academics, for example, would involve GIS and GRASS education, but would also try to pull GRASS development in its direction. Value added commercial developers would serve their own interests, including trying to pull GRASS development in a direction that would help their businesses. Users would help each other learn GRASS, develop workarounds to bugs, etc.

GRASS offers considerable potential for:

- Use as a scientific, as well as a traditional graphically oriented GIS. Many GISs can make pretty maps. Many of those GISs cannot easily perform certain scientific analytical functions as easily or powerfully as GRASS. GRASS was designed and developed in response to a perceived need for scientific GIS, specifically for environmental analysis, and the environmental management/protection of public lands. Incidentally, there is at least one Web−based GRASS version. *GRASSLINKS*, developed at *The University of California at Berkeley*, uses Web forms to submit commands to the server, which creates .gif−based display output, places the images into pages, and serves them up to the requester. More on that later.
- Education. GRASS is easier to teach and learn than some other GISs. It is easier to modify (for those that want to learn GIS as computer science, rather than as "geography") than most other GISs that come without source code and treat the program as a magical black box. And, of course, it is more affordable for the student of GIS than many other GISs.
- Applications research and development. Many universities have used GRASS. Its available source code, easy modification, easy scriptability, etc., give it distinct advantages over some more closed systems.
- Public Service. GRASS has been used as a scientific GIS for many public service applications. There is considerable value in continuing a robust GIS that can ba packaged with any UNIX workstation. There is considerably more value if that UNIX workstation universe can include Linux (but is not constrained only to Linux).
- GIS research and development. For example − do you want to experiment with a different data model? Add it to GRASS!
- Commercialization. This document gives contact information for a commercial version of GRASS. That company (and perhaps others?) may welcome your help in enhancing/supporting their product.

Who would be the Linus Torvalds equivalent in this management model? Perhaps no single person. I have been involved in GRASS for about a decade, when GRASS was the only GIS that satisfied my needs in scientific data management and GIS application. Indeed, I had been a dedicated avoider of the user−unfriendly UNIX environment until GRASS forced me to learn it. Several senior GRASS developers are active in GRASS−related activities and would like to see the continued vitality of an open GRASS. It's likely that a reborn GRASS would attract a new crop of friends. Thus the concept of a "Central Committee" to collectively lead GRASS' transition to a more open management and development style.

In short, the Linux community has an opportunity to take under its wing a killer ap. GRASS' current public domain status is slightly different from Linux's. However, that status could be discussed....

Comments would be appreciated!

---

9. Copyright Notice, and Notice on Support of this Document

9.1. Copyright notice

This document was prepared by a Federal civil servant in support of his work (but mostly on his own time). It is NOT SUBJECT TO COPYRIGHT.

9.2. Notice on support of this document

I believe that the contents of this document are accurate. However, if you use this document, you accept the risks for any errors in this document (and in any documents that are cited here).

I would greatly appreciate help in correcting any errors, or in enhancing this document. However, "my time is limited" in dealing with this issue. Any help that you can provide, that also helps me to more efficiently respond to your interest, is more likely to be responded to quickly. A complaint might be appreciated, but a suggested improvement that includes draft wording might be REALLY appreciated.

---

10. References

For general reference material on GIS, try a very good technical bookstore (in many cases these are campus bookstores at schools with good GIS programs or top–notch technical or general bookstores – you know that ones are near you..), or the following URL for the *CyberInstitute Short Course on Geographic Information Systems* (convened by myself):

Also check *Baylor University*'s growing *GRASS Home Page* and *USA/CERL*'s *GRASS Home Page*

For a good collection of references on GRASS, try this procedure, to load up on reference goodies from USA/CERL:

```
ftp moon.cecer.army.mil
login: anonymous
password: your email address
cd pub/grass/grass4.1/outgoing
image
get grassman.ps.Z  (or grassman.txt.Z, or grassman.wp.Z)
cd ../documents/programmer/postscript
image
get progman.ps.Z
cd ../../user/postscript
image
get refman.ps.Z
cd ../..
image
get installGuide.ps.Z
bye

uncompress grassman.ps.Z
uncompress progman.ps.Z
uncompress refman.ps.Z
uncompress installGuide.ps.Z

lpr *.ps   (or whatever is appropriate for your environment)
```

- installGuide => The GRASS Installation Guide (you need this to compile GRASS source code)
- grassman => The GRASS Beginner's Manual (intro to GRASS)
- refman => The GRASS User's Reference Manual (function guide)
- progman => The GRASS Programmer's Manual (and administrator's guide – this is valuable for info about data formats, etc.)

Browse around the ftp site noted just above, and you may find more stuff of interest. Particularly in the pub/grass/grass4.1/documents directory, there are tutorials on advanced GRASS functions such as r.mapcalc (think of this as math applied to raster arrays), r.combine and r.weight (think of this as how to combine spatial submodels into one type of model), and others.

A. Acquisition/Installation of GRASS4.13 Binaries

This appendix describes how to acquire and install Linux binaries for GRASS4.13 (the 3rd update to the last full release of GRASS, version 4.1).

How to get these files:

```
ftp moon.cecer.army.mil
login: anonymous
password: your email address
cd pub/grass/grass4.1/release/binaries/linux
image
mget grassa*
bye

Installation instructions:
*******************************************************************
* GRASS 4.1 Update 3 for Linux
*
* This package contains GRASS programs only, *NO* GIS
* data is included.  You can find example GRASS data at
* moon.cecer.army.mil
*
* Compiled by: Andy Burnett - burnett@zorro.cecer.army.mil
* compiled on: April 7, 1994

*******************************************************************
System Requiremnts:

        35 MB disk space to hold the binary distribution

System library requirements:

        libc4.5.21 or greater

        libX.so.3.1.0 or greater

If you are running libraries that are older than these, this binary
distribution will *NOT* run on your linux system.

-------------------------------------------------------------------------
Files in this release:

        README_4.1.3            what you are currently reading
        ginstall                simple grass installation script
        grassaa --------|
        grassab         |
        grassac         |
        grassad         |
        grassae         |--     the linux GRASS binaries
        grassaf         |
        grassag         |
        grassah         |
        grassai         |
        grassaj         |
        grassak --------|

INSTALLATION:
```

```
        To install this binary distribution of grass for linux, you
can simply run the ginstall script or you can unpack the files by
hand.  I recommend using the ginstall script ... it's very simple and
should be bullet proof.  To run the ginstall script, you will need
gawk (gnu awk) installed on your system and it needs to be in your
PATH.

If, however, you want to do things by hand, here's what you need to
do:

o  make the destination directory (/usr/grass, /usr/local/grass,
   whatever)  This will become your GISBASE for grass.

********************* LOOK HERE **************************************
from here on, replace $GISBASE with the name of the directory you just
created
********************* LOOK HERE **************************************

o  cat grassa? | gzip -d | (cd $GISBASE; tar xvf -)
   This will unpack all the grass binaries into the $GISBASE directory

o  copy $GISBASE/etc/moncap.sample to $GISBASE/etc/monitorcap and edit
   it.
o  change all occurrences of GBASE in that file to $GISBASE
o  copy $GISBASE/etc/grass4.1 into a public directory (I suggest
   /usr/bin)
o  edit the copy you just made:
   change all occurrences of GBASE to $GISBASE
```

B. Acquisition/Installation of GRASS4.1.5 Binaries

This appendix describes how to acquire and install Linux binaries for GRASS4.15 (the 5th and last update to the last full release of GRASS, version 4.1).

How to get these files:

```
ftp moon.cecer.army.mil
login: anonymous
password: your email address
cd pub/grass/grass4.1/release/binaries/linux
image
mget linuxa*
bye

Installation instructions:
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Files in this release:
        README_4.1.5            what you are currently reading
        install.sh                simple grass installation script
        linuxaa --------|
        linuxab         |
        linuxac         |
        linuxad         |
        linuxae         |--   the linux GRASS binaries, version 4.1.5
        linuxaf         |
        linuxag         |
        linuxah         |
        linuxai --------|

* * * * * * * * * * *** * * * * * * * * * * * * * * * * * * * * * * *
*

The GRASS4.15 for Linux was compiled in my Linux box with the
following configuration:
        Slackware 3.0
        kernel 1.2.13
        gcc 2.7.0
        libc 5.0.9
        flex 3.5.2

~ ~ ~ ~ ~ ~ ~
~ IMPORTANT: ~
~ ~ ~ ~ ~ ~ ~
THE LINUX GRASS 4.15 BINARIES ONLY WORK ON ELF-LINUX. THE BINARIES MAY
NOT WORK WITH EARLY VERSION OF KERNEL AND/OR GCC AND FLEX.

The binaries was tared and gziped, then split into 9 (close to 1.3 MB
- 1200 x 1K block) files named from linuxg.aa to linuxg.ai.

You should ftp all the linuxg.a* in binary mode and also get this
readme file and an installation script - install.sh.  Please put all
of these files in the same directory - source directory.

At the source directory under the UNIX prompt, type
        sh ./install.sh full_path_to_the_destination_directory

and it should automatically unzip and untar the linuxg.a* files to the
```

```
destination directory and also edit several site-specific files.  The
total space your need is about 26 MB.

At the destination directory, your can find the grass4.1 script.  It
should have been modified to reflect your installation directory.
Now, either move/copy the grass4.1 file to one of your PATH or use the
link command as below:
        cd /usr/local/bin
        ln -s destination_directory/etc/grass4.1 grass4.1

Now, you are ready to start GRASS by typing grass4.1 and you should
know how to run GRASS afterward.

There is a readme directory in the destination_directory/etc
directory.  This directory has several readme files that come with
some incoming commands.  You can find all the compiled commands of
this binaries in the commands.readme file.  I can't guarantee that all
of them work but I have tested lots of them.  If you find some
commands that don't work, please post a message on the grass user
group and we can solve it all together.

Yung-Tsung Kang,
Michigan State University
```

C. Acquisition/Compilation of GRASS Source Code

The GRASS binaries for Linux tend to work. Why would anyone want to mess with the source code?

Let's try to answer this with another question: "Why can't I get the source code to my GIS, so I can see how it works, and maybe fix some things to work the way I like them?" (You probably know the answers to this question, at least for many commercial software packages.)

If you want to

1. Add any of the numerous existing alpha and contributed GRASS functions,
2. Understand how a function works (did any programming shortcuts or performance enhancements affect the accuracy of a function? Can I improve the performance of a function?)
3. Revise or enhance the code (if you do this, please see Appendix D!),
4. Try compiling several tens of megabytes of source code, this appendix is for you. Also check Appendix E.

First, you need to acquire the source code, and the GRASS Installation Guide. You may also want to get the GRASS Programmer's Manual and User's Reference Manual. To do this:

```
ftp moon.cecer.army.mil
login: anonymous
password: your email address
cd pub/grass/grass4.1/release/source
get README.4
get README.5
image
mget s4* (or s5*, your choice)
cd ../../documents
get installGuide.ps.Z
cd /manuals/programmer/postscript
get progman.ps.Z
cd ../../user/postscript
get refman.ps.Z
bye
```

Don't forget this site. There are several tutorials on some of GRASS' more advanced programs in the pub/grass/grass4.1/document directory. There are two options for source code (I'm only discussing GRASS version 4.14 here, though version 4.15 is also available) The pub/grass/outgoing directory contains many contributed functions (and many other candidates for enhancing your system).

Follow the README.4 file for installing GRASS version 4.14 (which is sometimes called version 4.1.4) source code. Follow the README.5 file for installing GRASS version 4.15 (which is sometimes called version 4.1.5) source code.

After installing the source code, uncompress and print installGuide.ps.Z (or the troff version, if you prefer that and got it from a neighboring directory). You might also want to uncompress and print refman.ps.Z and progman.ps.Z at the same time. Note that progman.ps.Z is called the programmer's manual, but also contains valuable information about data formats and directory structures. Advanced users may also want to know the GRASS system utilities, even if they won't be calling them in code.

Now, use the GRASS Installation Guide (from installGuide.ps.Z) to guide yourself through the installation.

The thickness of this document may at first be intimidating. However, if you installed Linux yourself, you should be ready to tackle a GRASS installation. Don't be surprised if a function or two does not compile on your system. I have a couple of uncompiled functions on my own Linux system. Fortunately, these are functions that I don't use... Some day I'll get back to them, fix them, and compile them!?

Here is a late−breaking addition, on how to install the newly released *GRASS 4.2* from *Baylor University* This text is as provided by Baylor, unedited by myself due to its release only a few days ago. Please note the similarity with other installations..

C.1. GRASS 4.2 Quick Start

Installation Instructions

Warning
These instructions pertain to the 4.2 release of GRASS. Users are urged to consult the complete installation guide for more detailed instructions.

```
$GIS/src -
This directory contains scripts and files used to compile  GRASS.
By  running  scripts  and changing lists of programs you generate
GRASS binaries for your system.

You may mount a disk containing GRASS source on different types
of machines and compile without making source code copies.  You
follow the following instructions for each machine.

WARNING: These instructions presume that you have familiarity with
UNIX, C, make, and shell scripting.  Murphy's law visits GRASS
regularly and your expertise in these matters will be the best
defense against Mr. Murphy.

WARNING: These instructions and scripts have been used to compile
GRASS on various machines.  Please mail results of using this
information for compiling GRASS on your platforms and operating
system to:

grass@baylor.edu

DIRECTORY CONTENTS


    GISGEN      script which will compile GRASS

    MAKELINKS   script used after GISGEN to establish the user executable
                commands

    VERSION     current version number and date of the GRASS release

    generic/    system independent files need by gmake
                  gmake     shell script which does compilations
                  make.def  make variables
                  make.tail some additional make rules

    head/       gmake header file(s) for this site.  Header files are
                  created by running the utils/setup command.

    lists/      lists of programs to be compiled
                  GRASS     standard GRASS programs
                  local     site specific GRASS programs
                  ...       architecture dependent GRASS programs

    next_step/  files used by GISGEN to keep track of how far along
                it is in the compilation. Used to restart
                GISGEN (after a failure) where it left off.

    utils/      contains the 'setup' script and all support scripts
```

```
                and files needed by 'setup'


COMPILATION STEPS OVERVIEW

 (1) Generate files that contain location and machine specific make
    information.

 (2) Edit files containing lists of location and machine specific
     programs to be compiled (generally printer, digitizer, and graphics
     drivers).

 (3) Run GRASS compilation script

 (4) Run GRASS program linking script

 (5) Edit device driver configuratin files

 (6) Compile GRASS contributed, alpha programs.

 (7) Compile GRASS related and hybrid programs.

COMPILATION STEPS (DETAILS)

(1) Generate make support files

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

     compilation and load flags
     system libraries
     installation directories
     idefault data bases and locations

The shell script utils/setup assists you in define many of the make
options and definitions that will become part of every compile-time
generated makefile (about 350).  It also creates your shell script for
compiling individual GRASS programs - gmake4.2.

Run "utils/setup" and answer the questions.

The makefile portions are placed in the head/ under a name which you
specify/approve in the utils/setup process.  The executable shell script
which directs compilation is placed in (by default) /usr/local/bin.

Examine the just created file in head/ to make sure things are ok.
A brief description for each defined variable follows:

  ARCH            = Key name identifying the architecture of the machine
                    on which you are compiling GRASS.
  GISBASE         = Directory into which compiled GRASS will be contained
  UNIX_BIN        = Local directory where the GRASS entry program and gmake
                    will be contained

  DEFAULT_DATABASE= Directory where local GRASS data bases are contained
  DEFAULT_LOCATION= GRASS data base that users get as the first default

  COMPILE_FLAGS   = Compilation flags
  LDFLAGS         = Load flags

  TERMLIB         = System library containing low-level cursor movement
  CURSES          = System library that supports screen cursor control
```

```
  MATHLIB         = System math library
  LIBRULE         = Method for archiving and randomizing libraries

  USE_TERMIO      = Flag to use the termio library if available
  USE_MTIO        = Flag to use the mtio library if available
  CAN_CLEAR       = Flag indicating that the terminal screen can be cleared
  DIGITFLAGS      = Flags to set owner and priority of the v.digit program

(2) Edit files containing lists of location and machine specified programs

The directory lists/ contains files that list directories that will
be compiled.  Directory names are relative to the GRASS src
directory.  The file lists/GRASS lists all basic GRASS programs that
get compiled at every site.  The file lists/local and lists/$ARCH.

    -----------------------------------------------------------------
    $ARCH is the architecture name you approved while running the
    utils/setup script.  You can determine this by running:
        gmake4.2 -sh | grep ARCH
    -----------------------------------------------------------------

There man not be a lists/$ARCH file, but you are free to create it to
add names of programs you want compiled specifically for this
architecture.  It is an architecture-specific list which allows NFS
linked source code to compile one set of programs for one machine,
and another set for another machine.  All machines that share the
same source code via NFS mounts will compile the directories listed
in lists/local.

All lists may contain comment lines - indicated by a # as the first
character in the line.  The lists/local file contains lists of all
digitizer, graphics, and paint (hard-copy map) drivers.  All machine
specific devices are commented out - you must uncomment those that
are particular to your site or architecture.  You are encouraged to
move the graphics driver items to the appropriate lists/$ARCH file.

(3) Run GRASS compilation program

The script GISGEN drives the compilation process.  If all goes well
you will be able to simply enter the command GISGEN and wait.  The
entire compilation process takes from about 1/2 hour on the faster
workstations to about 8 hours on the slower workstations.

GISGEN collects all of the directory names to be compiled from lists/GRASS
lists/$ARCH and lists/local and begins running gmake4.2 in each directory.
Screen output is a collection of messages from GISGEN and from the UNIX
make program.  Failure at any step will halt compilation.  On failure
you might do one of the following things:

  1 - Fix a compilation problem by modifying code in the directory that
      failed.  After modification, return to this directory and rerun
      GISGEN.  Compilation will pick up at the failed directory and continue
      down the list of directories if successful.

  2 - Restart GISGEN.  If the failure requires modifications to code already
      compiled, or the compilation options you set in step 1, you must
      remove next_step/$ARCH (or next_step/next_step if ar architecture name
      was not specified in step 2.  You may then rerun GISGEN.

  3 - Skip the failed directory.  In this case you must seek through the
      files list/GRASS lists/$ARCH and lists/local to determine the directory
      name that follows the failed directory name.  The failed name is in
```

```
      next_step/$ARCH and must be replaced in that file with the next name.
      After editing, rerun GISGEN

When complete GISGEN will put the word DONE into the next_step file and will
print the phrase "DONE generating GIS binary code" to the screen.

(4) Run GRASS program linking script

The GISGEN directs a compilation process that stashes the GRASS programs
away in directories unavailable to the user community.  Most user commands
are actually links to a single program called "front.end".  Links to this
program must be made for every actual GRASS program.  This is done AFTER
GISGEN is finished.  To make (or re-make) links for all user programs
run the script MAKELINKS.

(5) Edit device driver configuratin files

Your compiled system may any combination of several graphics, paint, and
digitizer drivers.  Refer to the GRASS installation instructions for
details.

NOTE:  If you have trouble compiling your graphics driver, go to the directory
$GIS/src/display/devices and compile the proper drivers manually using gmake4.2.

(6) Compile GRASS contributed, alpha programs.

GRASS programs come in three flavors:

  MAIN - The programs are those compiled in step 3.  They have stood the
         test of time and are generally reliable programs.

  ALPHA - Alpha programs are intended to be included with the MAIN programs
         in the next release.

  CONTRIB - Sites generate lots of special purpose programs in GRASS to get
         some job done, but do not polish the effort sufficiently to
         even be considered alpha code can be distributed in this category.

ALPHA programs are found in the directory src.alpha.  You, the installer
may visit these programs and compile any that you desire.  In directories
that contain Gmakefile files, simply run: gmake4.2

CONTRIB programs are in the directory src.contrib.  The state of these
programs are varied.  Some programs may compile with gmake4.2; others
are suitable as a starting point for programmers who will be writing
new software.

(7) Compile GRASS related and hybrid programs.

The GRASS user community has discovered that there are several public-domain
programs that are very useful in conjunction with GRASS.  These are found
in the directory src.related.  Compile these programs based on instructions
(or lack of instructions) in the individual directories.

Hybrid programs are those that mix the capabilities of GRASS with the
capabilities of one or more of the "related" programs.  These are found
in the src.garden directory.  They require successful compilation of
the "related" programs and generally compile using the gmake4.2 and
the included Gmakefile files.

The rest of the compilation should just take some time.  If you have
already installed GRASS binaries, you should back up your system (or
```

```
at least get the working binaries out of the way of your
compilation!).

Good Luck!  And be secure in the likelihood that you can use the
compiled binaries if you bail out of a full compilation of the source
code.
```

D. Enhancing GRASS

If you plant to enhance any part of GRASS, read this first!

GRASS has been developed for over a decade as completely unrestricted public domain source code and executables. Though there was initial resistance to the existence of such robust software in the public domain, many competitors learned to take advantage of GRASS. It has reputedly been the intellectual stimulus for several enhancements to other GISs. Several companies conducted business by installing and customizing public domain GRASS for customers, and by providing other value–added services such as data base development.

As USA/CERL no longer supports the public version of GRASS, users are free to use what currently exists. They're also currently completely on their own. At least where the public domain version is concerned.

There is a *commercial version of GRASS*, adapted from the public domain version by Logiciels et Applications Scientifiques (L.A.S) Inc. of Montreal. In a recent check, LAS sold its GRASSLAND for Sun, Linux and Windows NT. LAS is trying to encourage the continuation of a robust public domain Linux, partly as a source of new ideas and code for their own developments.

E. Sample Files

Example Linux versions of some critical GRASS files

This appendix is the home of Linux−specific examples of selected GRASS configuration files. Currently, only several examples of a single file are offered. However, this is the most important file for configuration! Later, examples of database configuration files (e.g. DEFAULT_WIND) and other files may appear.

In the Installation Guide (pp. 10−11) you will see mention of the [header] file in directory $GIS/src/CMD/header (where $GIS is the directory in which you place GRASS − some folks put this in /usr/local − I put everything in a GRASS' own filesystem/directory /user/grass4.1). The installation guide favors Sun systems, as these were the development environment for GRASS4. (In case you cared, Masscomp workstations were earlier development environments.) Below are examples of this <header> file for linux (which you might want to name linux in your $GIS/src/CMD/header directory. You may want to refer to this section when running the setup ($GIS/src/CMD/utils/setup) command.

One version:

```
CC                    = gcc
ARCH                  =

GISBASE               = /user/grass4.1
UNIX_BIN              = /user/grass4.1/bin

DEFAULT_DATABASE      = /user/grass4.1/data
DEFAULT_LOCATION      = china

COMPILE_FLAGS         = -O2
LDFLAGS               = -s

XCFLAGS               = -D_NO_PROTO -DXM_1_1_BC
XLDFLAGS              =
XINCPATH              =
XMINCPATH             =
XLIBPATH              =
XTLIBPATH             = -L/usr/lib
XMLIBPATH             = -L/usr/lib
XLIB                  = -lX11
XTLIB                 = -lXt
XMLIB                 = -lXm
XEXTRALIBS            =

TERMLIB               =
CURSES                = -lcurses $(TERMLIB)
MATHLIB               = -lm

#                     LIBRULE = ar ruv $@ $?
#                     LIBRULE = ar ruv $@ $?; ranlib $@
#                     LIBRULE = ar ruv $@ $?; ar ts $@
#                     LIBRULE = ar rc $@ `lorder $(OBJ) | tsort`
LIBRULE               = ar ruv $@ $?

USE_TERMIO            = -DUSE_TERMIO
USE_MTIO              = -DUSE_MTIO
USE_FTIME             = -DUSE_FTIME
DIGITFLAGS            = -DUSE_SETREUID -DUSE_SETPRIORITY
VECTLIBFLAGS          =
```

```
GETHOSTNAME          = -DGETHOSTNAME_OK
```

Another version:

```
#CC                   = gcc
#CC                   = gcc -ggdb -traditional
CC                   = gcc -traditional
#CC                   = gcc -static

ARCH                 = linux

GISBASE              = /usr2/local/grass/grass4.1
UNIX_BIN             = /usr/local/bin

DEFAULT_DATABASE     = /usr2/local/grass
DEFAULT_LOCATION     = grass4.1

COMPILE_FLAGS        =
#COMPILE_FLAGS        = -O
LDFLAGS              = -s

XCFLAGS              = -D_NO_PROTO
XLDFLAGS             =
XINCPATH             = -I$GISBASE/xgrass
#XINCPATH             =
XMINCPATH            =
XLIBPATH             = -L/usr/lib
XTLIBPATH            = -L/usr/lib
XMLIBPATH            = -L/usr/lib
XLIB                 = -lX11
XTLIB                = -lXt
XMLIB                = -lXm
XEXTRALIBS           =

TERMLIB              =
CURSES               = -lcurses $(TERMLIB)
MATHLIB              = -lm

#                    LIBRULE = ar ruv $@ $?
#                    LIBRULE = ar ruv $@ $?; ranlib $@

#                    LIBRULE = ar ruv $@ $?; ar ts $@
#                    LIBRULE = ar rc $@ `lorder $(OBJ) | tsort`
LIBRULE              = ar ruv $@ $?; ranlib $@

USE_TERMIO           = -DUSE_TERMIO
USE_MTIO             = -DUSE_MTIO
USE_FTIME            = -DUSE_FTIME
DIGITFLAGS           = -DUSE_SETREUID -DUSE_SETPRIORITY
VECTLIBFLAGS         =
GETHOSTNAME          = -DGETHOSTNAME_OK
```

Another version:

```
#CC                   = gcc -traditional -ggdb
CC                   = gcc -traditional -m486
#CC                   = gcc
ARCH                 = linux
```

```
GISBASE              = /usr/local/grass/grass4.1
UNIX_BIN             = /usr/local/bin

DEFAULT_DATABASE     = /usr/local/grass
DEFAULT_LOCATION     = grass4.1

COMPILE_FLAGS        = -O2
LDFLAGS              = -s

XCFLAGS              = -D_NO_PROTO -DXM_1_1_BC
XLDFLAGS             =
XINCPATH             =
XMINCPATH            =
XLIBPATH             = -L/usr/lib
XTLIBPATH            = -L/usr/lib
XMLIBPATH            = -L/usr/lib
XLIB                 = -lX11
XTLIB                = -lXt
XMLIB                = -lXm
XEXTRALIBS           = -lXmu

TERMLIB              =
CURSES               = -lcurses $(TERMLIB)
MATHLIB              = -lm

#                    LIBRULE = ar ruv $@ $?
#                    LIBRULE = ar ruv $@ $?; ranlib $@
#                    LIBRULE = ar ruv $@ $?; ar ts $@
#                    LIBRULE = ar rc $@ `lorder $(OBJ) | tsort`
LIBRULE              = ar ruv $@ $?; ranlib $@

#USE_TERMIO           = #-DUSE_TERMIO
USE_TERMIO           = -DUSE_TERMIO
USE_MTIO             = -DUSE_MTIO
USE_FTIME            = -DUSE_FTIME
DIGITFLAGS           = -DUSE_SETREUID -DUSE_SETPRIORITY
VECTLIBFLAGS         =
GETHOSTNAME          = -DGETHOSTNAME_OK
```

Yet another version:

```
CC                   = cc
ARCH                 = linux

GISBASE              = /usr/local/grass4.15/linux
UNIX_BIN             = /usr/local/grass4.15/linux

DEFAULT_DATABASE     = /data/grassdata
DEFAULT_LOCATION     =

# -fwritable-strings - for ps.map only
#COMPILE_FLAGS        = -O -m486 -fwritable-strings
COMPILE_FLAGS        = -O -m486
LDFLAGS              = -s

XCFLAGS              = -D_NO_PROTO
XLDFLAGS             =
XINCPATH             =
XMINCPATH            =
```

```
XLIBPATH            = -L/usr/X11R6/lib
XTLIBPATH           = -L/usr/lib
XMLIBPATH           = -L/usr/lib
XLIB                = -lX11
XTLIB               = -lXt
XMLIB               = -lXm
XEXTRALIBS          =

TERMLIB             =
CURSES              = -lcurses $(TERMLIB)
MATHLIB             = -lm

#                   LIBRULE = ar ruv $@ $?
#                   LIBRULE = ar ruv $@ $?; ranlib $@
#                   LIBRULE = ar ruv $@ $?; ar ts $@
#                   LIBRULE = ar rc $@ `lorder $(OBJ) | tsort`
LIBRULE             = ar ruv $@ $?

USE_TERMIO          = -DUSE_TERMIO
USE_MTIO            = -DUSE_MTIO
USE_FTIME           = -DUSE_FTIME
DIGITFLAGS          = -DUSE_SETREUID -DUSE_SETPRIORITY
VECTLIBFLAGS        = -DPORTABLE_3
GETHOSTNAME         = -DGETHOSTNAME_OK
```

Intimidating? It probably shouldn't be if you've configured X Windows on your Linux box. These examples should give you patterns to look for when running the setup utility in GRASS (described in the Installation Guide).