Chapter 5

Raster Maps

This chapter provides an explanation of how raster map layers are accommodated in the GRASS database.[1]

5.1. What is a Raster Map Layer?

GRASS raster map layers can be conceptualized, by the GRASS programmer as well as the user, as representing information from a paper map, a satellite image, or a map resulting from the interpretation of other maps. Usually the information in a map layer is related by a common theme (e.g., soils, or landcover, or roads, etc.).

GRASS raster data are stored as a matrix of *grid cells*. Each grid cell covers a known, rectangular (generally square) patch of land. Each raster cell is assigned a single integer attribute value called the *category* number. For example, assume the land cover map covers a state park. The grid cell in the upper-left corner of the map is category 2 (which may represent prairie); the next grid cell to the east is category 3 (for forest); and so on.

land cover

2	3	3	3	4	4
2	2	3	3	4	4
2	2	3	3	4	4
1	2	3	3	3	4
1	1	1	3	3	4
1	1	3	3	4	4

1 = urban 3 = forest
2 = prairie 4 = wetlands

In addition to the raster file itself, there are a number of support files for each raster map layer. The files which comprise a raster map layer all have the same name, but each resides in a different database directory under the mapset. These database directories are:

[1] The descriptions given here are for GRASS 4.x data formats only. Previous formats, still supported by GRASS but no longer generated, are described in documents from earlier releases of GRASS.

directory	function
cell	binary raster (cell) files
cellhd	raster header files
cats	raster map category information
colr	raster map color tables
colr2	alternate raster map color tables
hist	raster map history information
cell_misc	miscellaneous raster map support information

For example, a raster map named *soils* would have the files *cell/soils, cellhd/soils, colr/soils, cats/soils,* etc.

Note. Database directories are also known as database *elements.* See *§4.4 Mapsets* [*p. 16*] for a description of database elements.

Note. *GIS Library* routines which read and write raster files are described in *§12.9 Raster File Processing* [*p. 98*].

5.2. Raster File Format

The programmer should think of the raster data file as a two-dimensional matrix (i.e., an array of rows and columns) of integer values. Each grid cell is stored in the file as one to four 8-bit bytes of data. An N*x*M raster file will contain N rows, each row containing M columns of cells.

The physical structure of a raster file can take one of 3 formats: uncompressed, compressed, or reclassed.

Uncompressed format. The uncompressed raster file actually looks like an N*x*M matrix. Each byte (or set of bytes for multibyte data) represents a cell of the raster map layer. The physical size of the file, in bytes, will be *rows*cols*bytes-per-cell.*

Compressed format. The compressed format uses a run-length encoding schema to reduce the amount of disk required to store the raster 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. If the data is single byte data, then each pair is 2 bytes. If the data is 2 byte data, then each pair is 3 bytes, etc. (see **Multibyte data format** below). The rows are encoded independently; the number of bytes per cell is constant within a row, but may vary from row to row. Also if run-length encoding results in a larger row, then the row is stored non-run-length encoded. And finally, since each row may have a different length, there is an index to each row stored at the beginning of the file.

Reclass layers. Reclass map layers do not contain any data, but are references to another map layer along with a schema to reclassify the categories of the referenced map layer. The reclass file itself contains no useful information. The reclass information is stored in the raster header file.

 - 25 -

Multibyte data format. When the data values in the raster file require more than one byte, they are stored in *big-endian* format,[2] which is to say as a base 256 number with the most significant digit first.

Examples:

cell value		base 256	stored as			
868	=	3*256 + 100	3	100		
137,304	=	$2*256^2 + 24*256 + 88$	2	24	88	
174,058,106	=	$10*256^3 + 95*256^2 + 234*256 + 122$	10	95	234	122

Negative values are stored as a signed quantity, i.e., with the highest bit set to 1:[3]

<table>
<tr><th>cell value</th><th></th><th>base 256</th><th colspan="4">stored as</th></tr>
<tr><td>-1</td><td>=</td><td>-(1)</td><td>1| 0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>-868</td><td>=</td><td>-(3*256 + 100)</td><td>1| 0</td><td>0</td><td>3</td><td>100</td></tr>
<tr><td>-137,304</td><td>=</td><td>$-(2*256^2 + 24*256 + 88)$</td><td>1| 0</td><td>2</td><td>24</td><td>88</td></tr>
<tr><td>-174,058,106</td><td>=</td><td>$-(10*256^3 + 95*256^2 + 234*256 + 122)$</td><td>1| 10</td><td>95</td><td>234</td><td>122</td></tr>
</table>

All data values in a given row are stored using the same number of bytes. This means that if the value 868, which uses 2 bytes, occurred in a row that uses 3 bytes to represent the largest data value, 868 would be stored as $\boxed{0}\boxed{3}\boxed{100}$

Also, one row may only require 2 bytes to store its data values, another 4 bytes, and yet another 1 byte. The rows are stored independently and would be stored using 2 bytes, 4 bytes, and 1 byte respectively.

File portability. The multibyte format described above is (except possibly for negative values) machine independent. If raster files are to be moved to a machine with a different cpu, or accessed using a heterogeneous network file system (NFS), the following guidelines should be kept in mind. All *4.1 format*[4] raster files will transfer between

[2] The fact that the values are stored *big-endian* should not be construed to mean that the machine architecture must also be *big-endian.* The programs which read raster files perform the necessary arithmetic to construct the value. They do NOT assume anything about byte ordering in the cpu.

[3] This means that the value is stored using as many bytes as required by an integer on the machine (usually 4).

[4] The raster file format did not change from 3.0 to 4.0.

machines, with two restrictions: (1) if the file contains negative values, the size of an integer on the two machines must be the same; and (2) the size of the file must be within the seek capability of the lseek( ) call.[5] The *pre-3.0 compressed* format is not stored in a machine-independent format, and cannot generally be used for intermachine transfer, unless the two machines have the same integer and long integer format.

5.3. Raster Header Format

The raster file itself has no information about how many rows and columns of data it contains, or which part of the earth the layer covers. This information is in the raster header file. The format of the raster header depends on whether the map layer is a regular map layer or a reclass layer.

Note. *GIS Library* routines which read and write the raster header file are described in *§12.10.1 Raster Header File* [*p. 106*].

5.3.1. Regular Format

The regular raster header contains the information describing the physical characteristics of the raster file. The raster header has the following fields:

raster header

proj:	1
zone:	18
north:	4660000
south:	4570000
east :	770000
west :	710000
e-w resol:	50
n-s resol:	100
rows:	900
cols:	1200
format :	0
compressed:	0

proj, zone

The *proj*ection field specifies the type of cartographic projection:[6]

0 is unreferenced x,y (imagery data)
1 is UTM
2 is State Plane
3 is Latitude-Longitude

Others may be added in the future. The *zone* field is the projection zone. In the

[5] This usually means that the size of a long integer on the two machines is the same.

[6] State Plane is not yet fully supported in GRASS and Latitude-Longitude is still under

example above, the projection is UTM, the zone is 18.

north, south, east, west

The geographic boundaries of the raster file are described by the *north, south, east,* and *west* fields. These values describe the lines which bound the map at its edges. **These lines do NOT pass through the center of the grid cells at the edge of the map, but along the edge of the map itself.**

n-s resol, e-w resol

The fields *e-w resol* and *n-s resol* describe the size of each grid cell in the map layer in physical measurement units (e.g., meters in a UTM database). They are also called the grid cell resolution. The *n-s resol* is the length of a grid cell from north to south. The *e-w resol* is the length of a grid cell from east to west. As can be noted, cells need not be square.

rows, cols

The fields *rows* and *cols* describe the number of rows and columns in the raster matrix.[7]

format

The *format* field describes how many bytes per cell are required to represent the raster data. 0 means 1 byte, 1 means 2 bytes, etc.

compressed

The *compressed* field indicates whether the raster file is in compressed format or not : 1 means it is compressed and 0 means it is not. If this field is missing, then the raster file was produced prior to GRASS 3.0 and the compression indication is encoded in the raster file itself.

Note.

If the rows and columns of the raster matrix are not stored in the raster header, they are computed from the geographic boundaries as follows:

```
rows = (north − south) / (ns resol)
cols = (east  − west)  / (ew resol)
```

If the rows and columns of the raster matrix are stored in the raster header, the resolution values are computed from the geographic boundaries as follows:

```
ns resol = (north − south) / (rows)
ew resol = (east  − west)  / (cols)
```

development.

[7] These fields were added for 4.0. Under 3.0 the number of rows and columns were calculated from the other values in the header. However, this sometimes resulted in wrong results since the resolution values could not be stored with sufficient accuracy in ascii format. 4.0 used the resolution fields only if the row and column fields are not present (i.e., 3.0 format header files).

5.3.2. Reclass Format

If the raster file is a reclass file, the raster header does not have the information mentioned above. It will have the name of the referenced raster file and the category reclassification table.

reclass header

reclass header		
reclass		
name:	county	
mapset :	PERMANENT	
#5		*first category in reclass*
1		*5 is reclassified to 1*
0		*6 is reclassified to 0*
1		*7 is reclassified to 1*
0		*8 is reclassified to 0*
2		*9 is reclassified to 2*

In this case, the library routines will use this information to open the referenced raster file in place of the reclass file and convert the raster data according to the reclass scheme. Also, the referenced raster header is used as the raster header.

5.4. Raster Category File Format

The category file contains the largest category value which occurs in the data, a title for the map layer, an automatic label generation capability, and a one line label for each category.

category file

```
# 5 categories
title for map layer
<automatic label format>
<automatic label parameters>
0:no data
1:description for category 1
2:description for category 2
3:description for category 3
5:description for category 5
```

The number which follows the # on the first line is the largest category value in the raster file. The next line is a title for the map layer. The next two lines are used for automatic label generation. They are used to create labels for categories which do not have explicit labels. (The automatic label capability is not normally used in most map layers, in which case the *format* line is a blank line and the *parameters* line is: 0.0 0.0 0.0 0.0.) Category labels follow on the remaining lines. The format is *cat : label.*

The first four lines of the file are required. The remaining lines need only appear if categories are to be labeled.

Note. *GIS Library* routines which read and write the raster category file are described in *§12.10.2 Raster Category File* [*p. 108*].

5.5. Raster Color Table Format

The GRASS raster color tables and associated programming interface have undergone a fairly major revision to resolve problems presented by raster maps that have a large range of data values. The previous design[8] used arrays to store a color for each data value between the minimum and maximum values in the raster map. This array structure was also reflected in the format of the color table file -- each color stored as a single line in the color file. Because GRASS raster maps can have data values in the range ±2147483647[9] this method of storing color information is clearly untenable.

GRASS 4.1 solves the above problem by representing color tables as linear ramps for intervals of data values. Colors are specified (and stored) for the endpoints of each interval. Colors for values between endpoints are not stored but are computed using a linear interpolation scheme.

The following is an example 4.1 color file:

4.1 color table file

% 1387 1801		
1387:255:85:85	1456:170:170:0	*colors for categories 1387-1456*
1456:170:170:0	1525:85:255:85	*colors for categories 1456-1525*
1525:85:255:85	1594:0:170:170	*colors for categories 1525-1594*
1594:0:170:170	1663:85:85:255	*colors for categories 1594-1663*
1663:85:85:255	1732:170:0:170	*colors for categories 1663-1732*
1732:170:0:170	1801:255:85:85	*colors for categories 1732-1801*

The first line is a % character (to indicate that this is a 4.x format color file) and two numbers indicating the minimum and maximum data values which have colors. The rest of the file are the color descriptors. In this example, the minimum and maximum values are 1387 and 1801. Looking at the first color line, the color for category 1387 is red=255, green=85, blue=85; the color for category 1456 is red=170, green=170, blue=0.[10] The color for category 1400 is calculated from the colors for categories 1387 and 1456:

---

[8] See the *GRASS 3.0 Programmer's Manual,* for details on the 3.0 color file.

[9] These values are for 32-bit architectures.

[10] The colors are represented as levels of red, green, and blue, where 0 represents the lowest intensity and 255 represents the highest intensity.

red	=	interpolate(255,170)	=	239
green	=	interpolate(85,170)	=	101
blu	=	interpolate(85,0)	=	69

There are other formats which are simply variants of this format. For example, if the red, green, and blue intensities are all the same, then only the "red" value appears. This next example defines a gray scale color table:

4.1 color table file

```
% 1387 1801
1387:0     1801:255
```

Also, if the starting and ending categories are the same, only the first appears:

4.1 color table file

```
% 1 6
1:34:179:112
2:233:110:15
3:127
4:43:135:33
5:70:7:52
6:93:210:163
```

Note. *GIS Library* routines which read and write the raster color table are described in *§12.10.3 Raster Color Table* [*p. 111*].

5.6. Raster History File

The history file contains historical information about the raster map: creator, date of creation, comments, etc. It is generated automatically along with the raster file. In most applications, the programmer need not be concerned with the history file. Occasionally a program might put information into this file not known or readily available to the user, such as information about a satellite image: sun angles, dates, etc. The GRASS *r.info* program allows the user to view this information, and the *r.support* program allows the user to update it. It is the user's responsibility to maintain this file.

Note. *GIS Library* routines which read and write the raster history file are described in *§12.10.4 Raster History File* [*p. 116*].

5.7. Raster Range File

The range file contains the minimum and maximum values which occur in a raster file. It is generated automatically for all new raster files. This file lives in the *cell_misc* element as "cell_misc/*name*/range" where *name* is the related raster file name.

It contains one line with four integer values. These represent the minimum and maximum negative values, and the minimum and maximum positive values in the raster file. If there are no negative values, then the first pair of numbers will be zero. If there are no positive values, then the second pair of numbers will be zero.

Note. *GIS Library* routines which read and write the raster range file are described in *§12.10.5 Raster Range File* [*p. 117*].