-

Performing Map Calculations on GRASS Data: r.mapcalc Program Tutorial

Marji Larson
Michael Shapiro
Scott Tweddale

U.S. Army Corps of Engineers
Construction Engineering Research Laboratory
Environmental Division
Spatial Analysis Systems Team
P.O. Box 9005
Champaign, IL 61826-9005

December 1991

Table of Contents

1. INTRODUCTION

1.1. Background

This document details the capabilities of **r.mapcalc**, a program that allows users to manipulate, analyze, and create map data by performing mathematical calculations on raster map layers. It is included among Geographic Resources Analysis Support System (GRASS) software. GRASS is a public domain, image processing, and geographic information system (GIS) originally developed by researchers in the Environmental Division of the U.S. Army Corps of Engineers Construction Engineering Research Laboratory (USACERL-EN) in Champaign, Illinois. The system is used to input, manipulate, analyze, and output geographic data by users in both military and non-military and public and private agencies based in North America, Europe, and other parts of the world.

Although most GRASS development has been conducted at USACERL, system integration, development, testing, distribution, training, and support is performed by numerous publicly- and privately-operated sites throughout the world. GRASS version 4.0 implemented significant additions and modifications to system libraries and programming code. Mechanisms are needed to transfer current technology and information about GRASS to user and development sites.

This document reflects modifications made to GRASS version 4.0 and guides the user through the construction of raster map analyses using **r.mapcalc**.

1.2. Objective

The objective of this work is to transfer knowledge of GRASS version 4.0 raster map analysis functions to the field. This document guides the user through the process of using **r.mapcalc** to manipulate, create, and analyze raster data layers in GRASS using mathematical expressions.

1.3. Approach

Current GRASS **r.mapcalc** functions are examined in this report.

1.4. Scope

This document discusses the 4.0-release version of the GRASS **r.mapcalc** program, completed in the Summer of 1991. Some elements of this document will be dated and inaccurate for post-4.0 software releases.

1.5. Mode of Technology Transfer

The information in this report will aid the technical transfer of USACERL's GRASS image processing and geographic information system. GRASS is being transferred to the field through the following mechanisms: training programs, hands-on experience, a user support center, newsletters, extensive documentation, institutional structures at the Army and interagency levels, communication networks, and other forums.

User feedback on GRASS program capabilities, documentation, and other technology transfer mechanisms is important to the development of the system. Users are encouraged to communicate such feedback to GRASS development staff at USACERL, via existing electronic communication networks and via the GRASS Information Center at USACERL, P.O. Box 9005, Champaign, IL 61826-9005; phone 217-373-7220; fax 217-373-7222; or e-mail grassbug@zorro.cecer.army.mil.

2. ORIENTATION

This paper describes the use and syntax of the GRASS software tool **r.mapcalc.** This tool uses mathematical syntax to perform analyses on existing map layers within GRASS. It allows a user to create a new map layer which is the result of a mathematical expression which may contain raster map layer names, integer or floating point constants, and known mathematical functions. **r.mapcalc** is designed to act as a calculator for maps. Mathematical calculations are performed for each grid cell of a raster map layer, applying the mathematical functions supplied by the user.

It is assumed that the reader is familiar with the meanings of raster data format, map layers, the current mapset and location, regions, and masks in GRASS. These and other basic GRASS concepts are reviewed in the GRASS hard copy and on-line help facility (accessible through **g.help**), in the *GRASS Programmer's Manual* (Shapiro et al., 1992), and in other GRASS documents available through the GRASS Information Center at USACERL.

Land managers are given the job of protecting, maintaining, and developing areas of land. To perform this duty, a land manager needs various types of geographical information. Some of this information may be available in the GRASS database, represented by raster map layers. Other information may not be available directly, but can be derived by analyzing and/or combining information from one or more existing raster map layers. Often, the manner in which information needs to be interpreted or combined can be represented by a mathematical equation. In such a case, the GRASS tool **r.mapcalc** provides the user with a means for quickly obtaining the desired resultant geographical information.

r.mapcalc is a calculator for raster map layers, performing mathematical calculations on a cell-by-cell basis to obtain a new raster map layer. Calculations are done in the usual precedence of operations with parentheses used to designate an otherwise desired ordering.

3. SYNTAX

The **r.mapcalc** command expects input in the form of:

result = expression

Within the equation "result=expression", *result* is the name of the new raster map layer to contain the result of the calculation, and *expression* is any legal mathematical expression involving raster map layers, integer or floating point constants, and functions known to the **r.mapcalc** calculator. Known operators and functions are listed in following sections, along with the order of computation for the operations.

r.mapcalc will create the resulting raster map layer in the current mapset. It should be noted that **r.mapcalc** works with the current region and mask settings. Caution should be used to insure correct results. If the user finds that the output from a **r.mapcalc** equation is not as expected, the current region and mask setting should be checked.

The **r.mapcalc** command can be used in one of three ways:

- entering the equation on the command line
- entering only the command name (no equation), and then entering equations, one at a time, at the new (MAPCALC) prompt
- passing one or more equations to the map calculator via redirected standard input (either from a file or across a pipe)

Because the UNIX shell interprets some characters to have special meanings (see the **Special Characters** section below), the user is advised to use the second of these methods. However, an explanantion of each input method follows.

If the equation (result=expression) is given on the command line, only one equation can be provided. This equation is evaluated, the new map layer is created, and the user is returned to the GRASS

4.0 prompt.

If the command **r.mapcalc** is entered alone, then a new (MAPCALC) prompt appears, and the user can enter equation after equation. After the entry of each equation, it is evaluated, the new map layer is created, and the user is returned to the MAPCALC prompt. After all desired equations have been evaluated, the user is returned to the GRASS 4.0 prompt by simply hitting the RETURN key (without entering anything on the line) to denote that no more equations are to be given.

In the event of lengthy formulas in the expression, a line continuation feature has been added. By adding a \ to the end of the input line, **r.mapcalc** assumes that the formula continues onto the next input line. There is no limit to the number of imput lines or to the length of the formula. Please note that the line continuation feature only works when entering formulas from standard input and does not work if the user enters input on the command line. Therefore, if the user is working with lengthy expressions, they should enter the standard input mode by typing **r.mapcalc** alone on the command line and hit return before entering input.

Input can also be provided through redirected standard input (from a file or across a pipe). If one or more equations have been entered into a file, they can be evaluated using **r.mapcalc** in one of the following ways:

```
r.mapcalc < inputfile
```

or

```
cat inputfile | r.mapcalc
```

The **cat** command simply types out the contents of a file. The pipe character | tells UNIX to "pipe" the results of one command into another command. Thus, in this last example, the contents of the file *inputfile* are piped to the **r.mapcalc** command as input. It is not the intention of this document to cover either the **cat** command or the piping capabilities of UNIX in detail. No further examples using the pipe capability will be given.

When equations are sent to the **r.mapcalc** command via redirected standard input, all entered equations are evaluated, one after another, and the user is returned to the GRASS 4.0 prompt when the end of the file or input stream is reached.

3.1. OPERATORS AND ORDER OF PRECEDENCE

Operator	Meaning	Type	Precedence
%	modulus (remainder upon division)	Arithmetic	1
/	division	Arithmetic	1
*	multiplication	Arithmetic	1
+	addition	Arithmetic	2
-	subtraction	Arithmetic	2
==	equal	Logical	3
!=	not equal	Logical	3
>	greater than	Logical	3
>=	greater than or equal	Logical	3
<	less than	Logical	3
<=	less than or equal	Logical	3
&&	and	Logical	4
||	or	Logical	4

Division by 0 and modulus by 0 are acceptable and give a 0 result.

The operators are applied from left to right, with those of lower numbered precedence applied before those with higher numbered precedence. Parentheses are allowed, and can be used to override stated precedence. Nested parentheses are allowed to any depth.

3.2. FUNCTIONS

The following functions are currently supported:

abs(x)	return absolute value of x
eval([x,y,...,]z)	evaluate values of listed expr, pass results to z
float(x)	convert x to floating point
if	decision options:
if(x)	1 if x not zero, 0 otherwise
if(x,a)	a if x not zero, 0 otherwise
if(x,a,b)	a if x not zero, b otherwise
if(x,a,b,c)	a if x > 0, b if x is zero, c if x < 0
int(x)	convert x to integer [truncates]
log(x)	natural log of x
log(x,b)	log of x base b
max(x,y[,z...])	largest value of those listed
median(x,y[,z...])	median value of those listed
min(x,y[,z...])	smallest value of those listed
round(x)	round x to nearest integer
sqrt(x)	square root of x
tan(x)	tangent of x (x is in degrees)

The variables x, y, z, a, b, and c above can be replaced by integer or floating point constants, a raster map layer name, or another expression involving constants, raster map layers, and other functions, provided the numbers represented are within the allowable domain of the given function. (For instance, ’tan(expression)’ makes sense only if ’expression’ evaluates to a number representing degrees.)

Note that although functions are available which produce floating point results, the resulting raster map can hold only integer (category) values. Thus, if the final "answer" to a calculation is a floating point number, the number will be rounded to the nearest integer before being stored as a value in the resulting map layer.

Calculations will be done according to the type of values being used in the expression. For instance, (8 / 3) will be calculated using integer arithmetic, and the answer will be 2. If the user wants the answer to be a floating point number which is rounded, care should be taken to create floating point arithmetic; for example, (8.0 / 3) and (float(8) / 3) would be evaluated as floating point values (here, 2.66) that are finally rounded to 3 after all calculations are completed, when assigned as map category values in the resultant map layer.

Anything given in the mathematical expression which is not a constant, an operator, or a known function name will be taken as a raster map layer name. If a GRASS raster map layer has been given a name which could be interpreted as a constant or an arithmetic expression, the raster map layer name must be enclosed in double quotes for correct use with the **r.mapcalc** command. For instance, suppose a map layer has been created which is a reclass of a soils map layer, and it has been named *soils-reclass*. The expression

```
newmap=soils-reclass * 2
```

would attempt to create a raster map layer named *newmap* which is the raster map layer *soils* minus the raster map layer *reclass* times 2, while the expression

```
newmap="soils-reclass" * 2
```

would attempt to create a raster map layer named *newmap*, which is the raster map layer *soils-reclass* times 2.

Using quotes around the resulting map name is an error.

3.3. MAPSET SEARCH PATHS

The **r.mapcalc** command will look for the map layer names given in the mathematical expression according to the current mapset search path. It is possible to use a raster map layer found in another mapset by explicitly specifying the mapset (mapname@mapset). For instance, in the following expression:

```
newmap = x@PERMANENT / y
```

the raster map layer *x* will be looked for specifically in mapset PERMANENT, while the raster map layer *y* will be searched for in the mapsets included in the user's current GRASS mapset search path.

The "@mapset" form is not allowed in the name for the resulting map layer, which is automatically created in the current mapset.

3.4. SPECIAL CHARACTERS

For evaluating only one equation, it is simplest to enter the equation on the command line. However, there are two things about this usage of the **r.mapcalc** command that should be brought to light here.

Since the equation is being given on the command line, care must be taken when using characters which have special meaning to the UNIX shell. It is advisable to enclose the expression portion in single quotes to avoid unexpected results. These special characters include, among others:

```
* ( ) > & | [ ] <
```

For instance, since the * character has special meaning to the UNIX shell, the expression in:

```
r.mapcalc newmap = elevation * 2
```

would be altered before being sent to the map calculator, and would result in error. Instead, using single quotes around the expression portion:

```
r.mapcalc result = 'elevation * 2'
```

would let the UNIX shell know that any special characters should not be altered, and the expression would be sent as is to the **r.mapcalc** command. (Without the use of single quotes around the expression portion, parentheses may also be misinterpreted, and, although the expression may be correct, the message "Badly placed ()'s" would be given.)

The second thing to note about this type of usage of the **r.mapcalc** command is that if the result map given on the command line is the name of a currently existing raster map layer, it is assumed that the user wants to overwrite it, and the raster map layer will be overwritten without warning.

If the user wishes to input more than one equation, a different usage of the **r.mapcalc** command would be logical. For this case, only the command name is given. A new (MAPCALC) prompt is then provided, and equations are entered one after another. All characters given are taken as is, and are not modified by the UNIX shell. If the result raster map layer given at this MAPCALC prompt already exists, the user is warned, and asked for a decision about overwriting it.

3.5. NEIGHBORHOOD MODIFIERS

Maps and images are database files stored in raster format, i.e., two-dimensional matrices of integer values. In **r.mapcalc**, maps may be followed by a neighborhood modifier that specifies a relative offset from the current cell being evaluated. The format is map[r,c], where r is the row offset and c is the column offset. For example, map[1,2] refers to the cell one row below and two columns to the right of the current cell, map[-2,-1] refers to the cell two rows above and one column to the left of the current cell, and map[0,1] refers to the cell one column to the right of the current cell. This syntax permits the development of neighborhood-type filters within a single map or across multiple maps.

3.6. CATEGORY LABELS vs. CATEGORY VALUES

Sometimes it is desirable to use a value associated with a category's contents instead of the category value itself. If a raster map layer name is preceded by the @ operator, then the labels in the category file for the raster map layer are used in the expression instead of the category value. For example, if you specified "@elevation.dted" in an **r.mapcalc** equation, it would read the actual elevation values for each category, rather than the category numbers themselves, which are simply scaled between 1-255.

This use of the "@" symbol is different from that earlier described in the section **Mapset Search Paths**. In that section, the "@" symbol is *preceded* by a GRASS map layer name, and is followed by the name of a GRASS mapset. It is used to specify the use of a particular map layer (i.e., the named map layer stored under the named mapset), rather than any other map layer of the same name stored under any other mapset listed in the user's mapset search path.

4. SAMPLE SESSION/EXAMPLES

This sample session can be replicated by running the GRASS within the location *spearfish.*

Commands that you are requested to enter are displayed in this document with the following format:

PROMPT:> **your_command**

Results that you should see after entering a command are displayed in this document with the following format:

Some result displayed here

The PROMPT you see will depend on whether you are at the main GRASS 4.0 shell level, or whether you have typed in the **r.mapcalc** command alone (without arguments) and are therefore at a MAP-CALC prompt.

4.1. COMMAND-LINE EXAMPLES

Suppose that you have a raster map layer describing elevation in which the category values are the true elevations. If you need to have a map layer which also depicts the elevation, but you want the numbering to start at 1 (instead of at the minimum elevation value for that geographic region), you could use the **r.mapcalc** command with a simple subtraction to create the new map. First, you should find out the range of elevation values contained in the original map layer. This can be done using the GRASS command **r.describe**.

```
GRASS 4.0>  r.describe elevation.dem
```

```
READING [elevation.dem in PERMANENT] ...   100%
0 1066-1840
```

(Note that in the result section above, 100% is given after the statement about reading a map layer. As the command is actually executing, this will start as 0% and be updated as the execution progresses.)

Since the minimum elevation value is 1066, you would subtract out 1065 to start the numbers in the new map layer at 1.

```
GRASS 4.0> r.mapcalc new.elev = elevation.dem - 1065
```

```
PARSING EXPRESSION elevation.dem - 1065
EXECUTING new.elev = elevation.dem - 1065
complete:    100%
CREATING SUPPORT FILES FOR new.elev
minimum value -1065, maximum value 775
```

The map calculator has to first parse the expression given, and see if it makes sense mathematically. If the map calculator does not understand something (a function used is not known to the calculator, a parenthesis is unmatched, or the expression is mistyped), a message will be given that the calculator has been confused. If the expression is understood, the calculations begin. The equation is calculated for each grid cell location in the current region, substituting the category value from that grid location for each map layer name in the expression. Once all calculations are complete, the new map layer is created, and the minimum and maximum values in the new map layer are reported.

Note that the output from the above command may not be the desired result. The grid cell locations which contained a "0" (no data) value in the original map layer now contain a value of -1065. To obtain the desired result (renumbering the values to start at 1) without changing the cell values that were originally 0, we will use a slightly more complex equation, given below. Single quotes should be used around the expression. Otherwise, the UNIX shell tries to interpret the parentheses in a special manner, and the command will not complete.

```
GRASS 4.0> r.mapcalc new.elev = ’if(elevation.dem,elevation.dem-1065)’
```

```
PARSING EXPRESSION if(elevation.dem, elevation.dem-1065)
EXECUTING new.elev = if(elevation.dem, elevation.dem-1065)
complete:    100%
CREATING SUPPORT FILES FOR new.elev
minimum value 0, maximum value 775
```

This form of the "if" command within the map calculator is interpreted as: if the category value of "elevation" is non-zero, then set the result equal to "elevation-1065". Otherwise, set the result equal to 0. Note that you were not asked if you desired to overwrite the cell file *new.elev*. The assumption is that you do want to overwrite it.

4.2. STANDARD INPUT EXAMPLES

Next we will perform a few more calculations, and to simplify the process, we will type the **r.mapcalc** command alone without any accompanying equation, then entering our equations, one at a time, at the MAPCALC prompt. (This will simplify the need for quotes around the expression portion of the equation, and will warn us about overwriting raster files that already exist. Also, there will not be a need to re-enter the command name each time, only the new equation.)

GRASS 4.0> **r.mapcalc**

```
MAPCALC>
```

A new (MAPCALC) prompt is given, and now we can enter equations one after another.

Suppose you would like a map layer depicting an area which has been severely flooded. We will alter the elevation file from the *spearfish* data set so that only elevations below 1200 feet will be represented, assuming a flooding to 1200 feet. (This is a bit high for flooding, but will illustrate the technique to be used.) Enter the following equation:

MAPCALC> **new.elev = if(elevation.dem <=1200)**

```
PARSING EXPRESSION if(elevation.dem<=1200)
new.elev - already exists. ok to overwrite? [n] y
EXECUTING new.elev = if(elevation.dem<=1200)
complete:     100%
CREATING SUPPORT FILES FOR new.elev
minimum value 0, maximum value 1
```

The color assigned to category 1 should be made a blue color to represent the water level at this elevation.

Note that since the equation is being entered at the MAPCALC prompt, a warning is given that the map layer *new.elev* already exists, and the choice is provided to overwrite or not. If you do not wish to overwrite the map layer *new.elev*, enter **n** in response to the question (or simply hit RETURN to accept the default response of ’n’ shown in brackets). Then you must re-enter the equation with a new resulting map layer name.

Now, suppose we want to determine the depth of flooding.
MAPCALC> **depth= if(new.elev,1200-elevation.dem)**

```
PARSING EXPRESSION if(new.elev,1200-elevation.dem)
EXECUTING depth = if(new.elev,1200-elevation.dem)
complete:     100%
CREATING SUPPORT FILES FOR depth
```

4.3. CREATING NEW DATA LAYERS

In the *spearfish* data set, raster map layers exist representing aspect and streams. It would be interesting to display the streams file overlaid on the aspect map layer to see if the streams seem to line up with changes in aspect direction. This could be done by displaying the *aspect* map layer, and then using the GRASS command **d.rast -o** with the map layer *streams*. If you wanted to actually create a new map layer representing the two map layers together, you could use **r.mapcalc**.

The aspect map layer contains 25 categories--24 directions of aspect, plus category 25 representing "no aspect". (The category range of the aspect map layer can be obtained using the GRASS command *r.describe*.) We will create a new map layer retaining the 25 categories from the aspect map layer and assigning the streams data to category number 26. Enter the following equation:

MAPCALC> **aspect.str = if(streams,26,aspect)**

```
PARSING EXPRESSION if(streams,26,aspect)
EXECUTING aspect.str = if(streams,26,aspect)
complete:     100%
CREATING SUPPORT FILES FOR aspect.str
minimum value 0, maximum value 26
MAPCALC>
```

When the streams file contains a non-zero value, the new map layer is assigned a value of 26. Otherwise, the new map layer receives the category value of the aspect map layer. The new map layer *aspect.str* will not have any assigned color table. One should be assigned using the GRASS command **r.support** (probably the color table type labeled "Aspect"). For the streams (represented by category 26) to stand out against the rest of the map, the color for category 26 should then be altered. This can be done using the **d.colors** program.

Hit the RETURN key now to return to the GRASS 4.0 prompt.

The next example will illustrate how **r.mapcalc** can be used to perform an analysis on various raster map layers. Given the following information about the categories for map layers *transport.misc*, *roads*, and *railroads*, an analysis can be done to represent all areas within the *spearfish* location where transport lines are available.

map:	transport.misc	roads	railroads
c	0 No data	0 No data	0 No data
a	1 Power transmission	1 Interstate highway	1 railroad line
t	2 Landing strip	2 Primary highway, hard surface	2
e		3 Secondary highway, hard surface	
g		4 Light duty road, improved surface	
o		5 Unimproved road	
r			
y			

We will say that transport lines are available when *transport.misc* has category value 2, when *roads* has category values 1, 2, or 3, or when *railroads* has category value 1. The result raster map layer (*trans.avail*) will have only the one category, denoting that one of the above was true and therefore a transportation line of some sort is available. Since the equation for this is a little complicated, it might be easiest to create a text file which can be used to provide the input to **r.mapcalc.** This makes it easier to correct any typing errors. The file created for input is called *map.input* and is printed out below using the UNIX command **cat**.

```
GRASS 4.0> cat map.input
```

```
trans.avail = transport.misc==2 || roads==1 || roads==2 || roads==3 || railroads==1
```

4.4. REDIRECTING INPUT

We will evaluate this equation using the redirected input format of the **r.mapcalc** command as shown below.

```
GRASS 4.0> r.mapcalc < map.input
```

```
PARSING EXPRESSION
  transport.misc==2 || roads==1 || roads==2 || roads==3 || railroads==1
EXECUTING
  trans.avail = transport.misc==2 || roads==1 || roads==2 || roads==3 || railroads==1
complete:    100%
CREATING SUPPORT FILES FOR trans.avail
minimum value 0, maximum value 1
EOF
```

Each test for equality (like "transport.misc==2") is a logical expression which evaluates to 0 or 1 (false or true). The tests are joined with logical or's, ||, so that the result of the expression will evaluate to 1 if any one of the tests for equality is true. The resulting map layer *transport.misc* will be a zero-one map, where category 1 represents the fact that one of the above conditions is true.

The **r.mapcalc** tool can be used to create map layers with new types of information, derived from information contained in other map layers. Whenever an equation exists that calculates a value from other values, **r.mapcalc** will be useful provided the original variables in the equation are available as existing raster map layers. For instance, given a drainage accumulation map layer (where category values represent the number of cells within a given watershed which would drain to the given grid location) and a map layer depicting slope values, a map layer can be derived which approximates the distribution of a topographic index characterizing the degree of saturation in the soil mantle. The index is mathematically expressed as the

ln (drainage_accumulation/tan(slope_magnitude))

An **r.mapcalc** statement expressing this mathematical formula can be written using the log and tan functions. High saturation areas are flat and have large contributing areas. A drainage accumulation file for a watershed within the *spearfish* location can be derived from elevation data using the GRASS program **r.watershed**. A slope map layer can be derived from elevation data using the GRASS program **r.slope.aspect**. A slope map should already be available in the *spearfish* database. You should check to be sure this is not a reclassed version of the slope map, but instead represents degrees of slope. The slope map derived from the **r.slope.aspect** program reserves category 0 to represent "no data," and so zero degrees is represented as category 1, one degree by category 2, and so on. In developing our mathematical expression for **r.mapcalc**, we should therefore substitute "slope-1" for "slope" in the equation given above. We do not want to subtract 1 when the slope map layer contains a zero value (recall the example with elevation-1065), so we should use the expression "if(slope,slope-1)." The map calculator for **r.mapcalc** represents the natural log by "log" (not "ln"), so the expression would look like: log(drain/tan(if(slope,slope-1))) for a drainage accumulation map layer named *drain* and a slope map layer named *slope*. The last adjustment to make is to take into consideration what happens when either the drainage accumulation map layer or the slope map layer contains "no data" (category value 0). This would result in a calculated value that is negative and large. We will take care of this case of evaluating using "no data" by eliminating all negative results of the equation (removing results representing a negative soil saturation index). This can be done using the max function as shown below. If the result of the expression is greater than zero, its result will be assigned to the new map layer. However, if the result is less than zero, then zero will be the result of the max function. The **r.mapcalc** equation used to produce this product map would be:

GRASS 4.0> **r.mapcalc index.sat='max(0, log(drain / tan(max(0,slope-1))))'**

```
PARSING EXPRESSION max(0, log( drain / tan(max(0,slope-1)) ) )
EXECUTING index.sat = max(0, log( drain / tan(max(0,slope-1)) ) )
complete:    100%
CREATING SUPPORT FILES FOR index.sat
minimum value 0, maximum value 12
```

You are now finished with the examples and the sample session. See the entry for **r.mapcalc** in the *GRASS User's Reference Manual* for additional examples. Unless you want to keep the new map layers created with this tutorial, you may want to remove them now to conserve disk space.

5. CONCLUSIONS AND RECOMMENDATIONS

This manual is part of an ongoing process to document Geographic Resources Analysis Support System (GRASS) program functions. It is one of many technology transfer mechanisms being implemented to explain current, and future, GRASS program capabilities to resource managers and system users.

Documentation, along with training programs, hands-on use, a user-support center, newsletters, institutional structures at the Army and interagency levels, communication networks, and other forums, can be used to aid this ongoing technology transfer effort.

User feedback on GRASS program capabilities, documentation, and other technology transfer mechanisms is also needed. Users are encouraged to communicate such feedback to GRASS development staff at USACERL, via existing electronic communication networks and via the GRASS Information Center at USACERL, P.O. Box 9005, Champaign, IL 61826-9005; phone 217-373-7220; fax 217-373-7222; e-mail grassbug@zorro.cecer.army.mil.

REFERENCES

Ruiz, Marilyn S., and Jean M. Messersmith, *Cartographic Issues in the Development of a Digital GRASS Database*, USACERL Special Report N-90/16 (U.S. Army Construction Engineering Research Laboratory [USACERL], September 1990).

Shapiro, Michael and James Westervelt, *r.mapcalc: An Algebra for GIS and Image Processing*, Draft ADP Report (USACERL, Summer 1992).

Westervelt, James, and William D. Goran, *Introduction to GRASS 4*, Draft ADP Report (USACERL, July 1991).

Westervelt, James, Mary Martin, and Deborah Brinegar, "GRASS 4.0 Programs," in *GRASS User's Reference Manual*, ADP Report N-87/22 rev (USACERL, September 1988, last revised December 1991).

Westervelt, James, Michael Shapiro, William D. Goran, et al., *GRASS User's Reference Manual*, ADP Report N-87/22 rev (USACERL, September 1988, last revised December 1991).