GRASS - Automation

Automation Concepts

- Automation is simply the combination of a series of commands in a form that allows for their repeated execution with a minimum of user intervention
- Automation is accomplished through the use of *scripting languages* that support the running of executable programs
- Scripting languages that may be used include:
  - The UNIX (or DOS) shell: BASH, TCSH, etc.
  - Perl
  - Python
  - PHP
- The sophistication of the scripts that may be developed is determined by the specific capabilities of the language being used.

GRASS Execution in Scripts

- GRASS executes within a standard shell environment with a set of environmental variables set to facilitate the proper execution of GRASS commands
- The current set of environment variables that existing within an existing GRASS session may be obtained by executing the `env` command at the GRASS command prompt.

```
> env
  ... list of current environment variables
```

- The specific environment variables required by GRASS to ensure proper command execution include:

GISBASE
GISDBASE
GISRC

PATH (include GRASS binary path)

GRASS Execution in Scripts (cont.)

- The startup process for the GRASS application takes care of setting the environment variables when starting
  - This results in an ability to run simple scripts that only contain GRASS commands and required script logic *from within the GRASS environment* (i.e. from the GRASS command prompt)
- GRASS commands may be executed outside the GRASS application by explicitly setting the required environment variables
  - This results in an ability to write scripts that contain all of the information required to execute GRASS commands from outside of the GRASS application environment (in a web application, or as part of an automated data acquisition and processing script).

Elements of a Script

- If a script is be executed in a *NIX environment, the first line should have a *shebang*, the symbols "#!" followed by the path and filename of the *interpreter* to use in executing the remainder of the script. For example:

```
#!/usr/bin/perl
#!/usr/local/bin/bash
#!/bin/bash
```

  - Systems that do not require the *shebang* should treat this line as a comment and ignore it.
- Comments, represented by different symbols depending upon the specific script interpreter being used. The "#" symbol is commonly used to denote the beginning of comment text. The text following the "#" symbol is typically ignored by the interpreter.

Elements of a Script (cont.)

- The content of the script then consists of commands in the language of the interpreter, including commands that would execute GRASS commands.
- The ability to include a combination of interpreter commands and GRASS commands provides a powerful tool for performing sophisticated automated processes
  - Specifying specific data files to process - yielding different results each time the script is run, depending upon the script specifications.
  - Looping through a set of commands repeatedly
  - Optionally executing commands based upon a specified logical test
  - Branching through a process, depending upon needs
- Before a script/program can be run, it must be made executable - use the `chmod` command: e.g.

```
chmod u+x <script name>
```

Elements of a Script (cont.)

- Scripts may be created with any text editor, though some editors make script writing easier by providing helping functions like syntax coloring. Editors you might use include:
  - Notepad (windows)
  - Textedit, BBedit (Mac OS X)
  - Kate, Gedit, vi, EMACS (Linux)
- I would recommend that you use Kate (on the class disk in the 'Editors' area of the 'K menu'
- Scripts, unlike compiled computer programs, are saved as standard ASCII text. If you use a word processor to edit a script, make sure that it is saved as ASCII text - not in the word processor format.
- Before a script/program can be run, it must be made executable - use the `chmod` command: e.g.

```
chmod u+x <script name>
```

Script Examples - BASH Shell Environment

- The script examples that will be shown here are based upon the BASH shell - a *NIX shell available on many linux and UNIX platforms (including Cygwin)
- Notes about the BASH shell
  - The *shebang* will point to the location of the BASH interpreter on the system, for example (the *shebang* for BASH on the class disk)

```
#!/bin/bash
```

  - The character identifying comments is "#", for example:

```
# this is a comment
this is a command # followed by a comment
```

  - Environment variables are defined using the `export` command, e.g.

```
export GRASSRC='/home/kbene/.grassrc60
```

Script Examples 1 - Basic Screen Mapping within GRASS

This script executes the series of commands that were previously used to generate a map on the computer screen, within the GRASS application environment

```
#!/bin/bash
# sample script that generates a three frame map
# to be displayed on the computer screen

# GRASS Commands
g.region region=sw      # set the region to a previously saved one

d.mon start=x0          # start the monitor
d.erase color=blue      # erase to a blue background

d.frame -c frame=legend at=0,90,0,10      # define legend frame
d.frame -c frame=title at=90,100,0,100    # define title frame
d.frame -c frame=map at=0,90,10,100       # define map frame

d.frame -s frame=legend                   # select the legend frame
d.erase color=black                       # erase legend frame to black
d.frame -s frame=title                    # select the title frame
d.erase color=black                       # erase title frame to black

< continued >
```

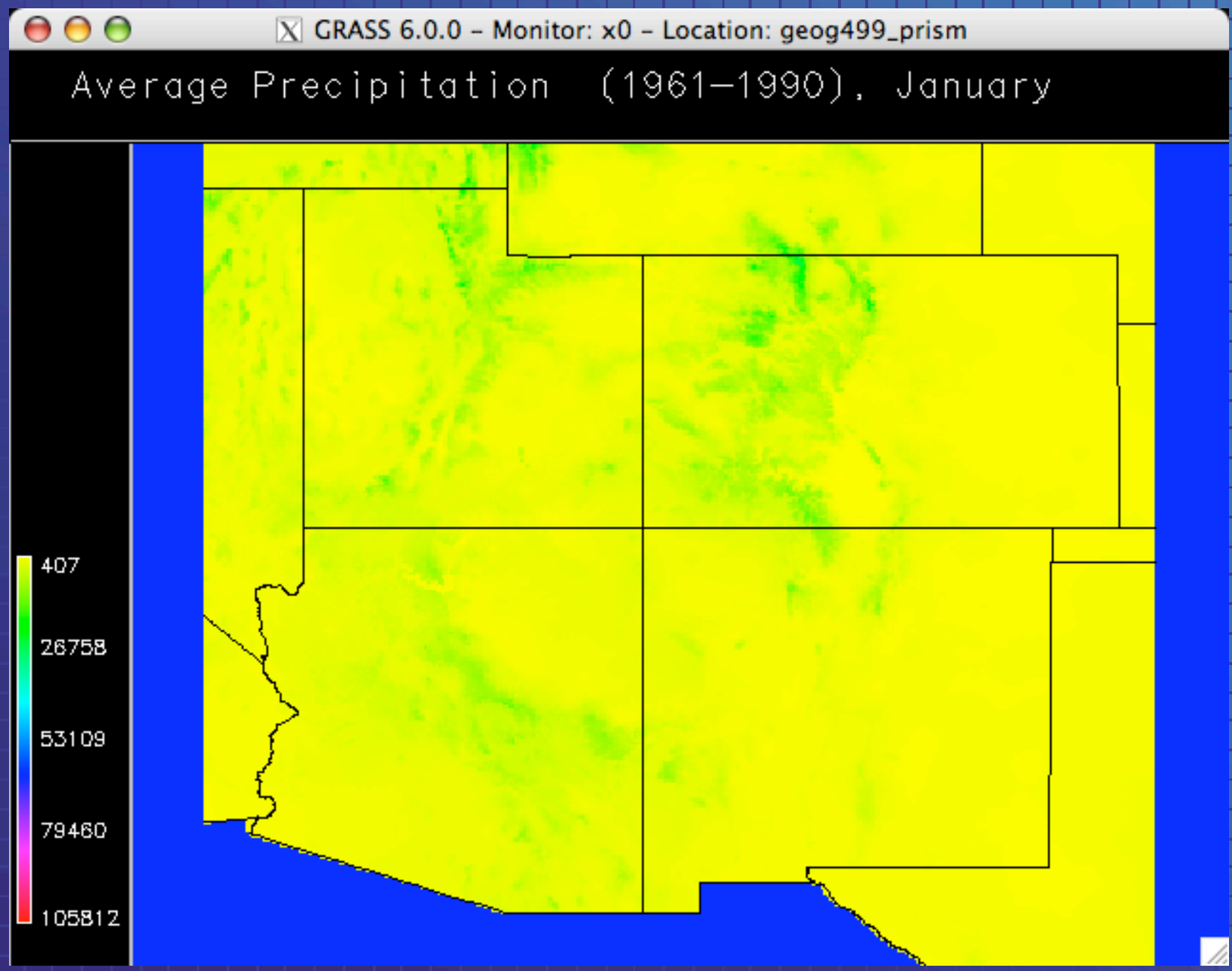
Script Examples 1 - Basic Screen Mapping within GRASS (cont.)

```
< continued >

d.frame -s frame=map                          # select the map frame
d.rast -o map=us_ppt.01.int                   # display the precip raster
d.vect map=census_states type=boundary        # display the state boundaries

d.frame -s frame=legend                       # select the legend frame
d.legend map=us_ppt.01.int thin=50 \          # display the legend for
  at=5,50,5,20 color=white                    # the precipitation raster

d.frame -s frame=title                        # select the title frame
echo “Average Precipitation  \                # print the title
  (1961-1990), January” | d.text
  size=40 color=white at=5,50

# End of script ...
```

Script Examples 1 - Process for Creating the Script

- Open a new file in a text editor (Kate for example)
- Enter the commands for the script into the text file
- Save the file in a convenient location (i.e. a directory that you designate as a storage location for scripts)
- Make the file executable by executing the `chmod` command
  - To make it executable by the owner of the file (i.e. you)
    ```
    chmod u+x myscript.sh
    ```
  - To make it executable by everyone (only do this if you want other users on the system to use the script)
    ```
    chmod ugo+x myscript.sh
    ```
- The script may then be run by either executing it within the current directory
  ```
  > ./myscript.sh
  ```
  or by specifying the complete path to the script
  ```
  > /home/user/grass_scripts/myscript.sh
  ```

Script Examples 1 - Result



Script Examples 2 - Generating a PNG File within GRASS

This script executes the series of commands that were previously used to generate a PNG file, within the GRASS application environment

```
#!/bin/bash
# sample script that generates a three frame map
# to be displayed on the computer screen

# GRASS Commands

# define the required environment variables
export GRASS_WIDTH=1280
export GRASS_HEIGHT=960
export GRASS_PNGFILE='precip.png'
export GRASS_BACKGROUNDCOLOR=050505
export GRASS_TRANSPARENT=TRUE
export GRASS_TRUECOLOR=TRUE
export GRASS_PNG_COMPRESSION=1
export GRASS_PNG_AUTO_WRITE=FALSE

g.region region=sw     # set the region to a previously saved one

< continued >
```

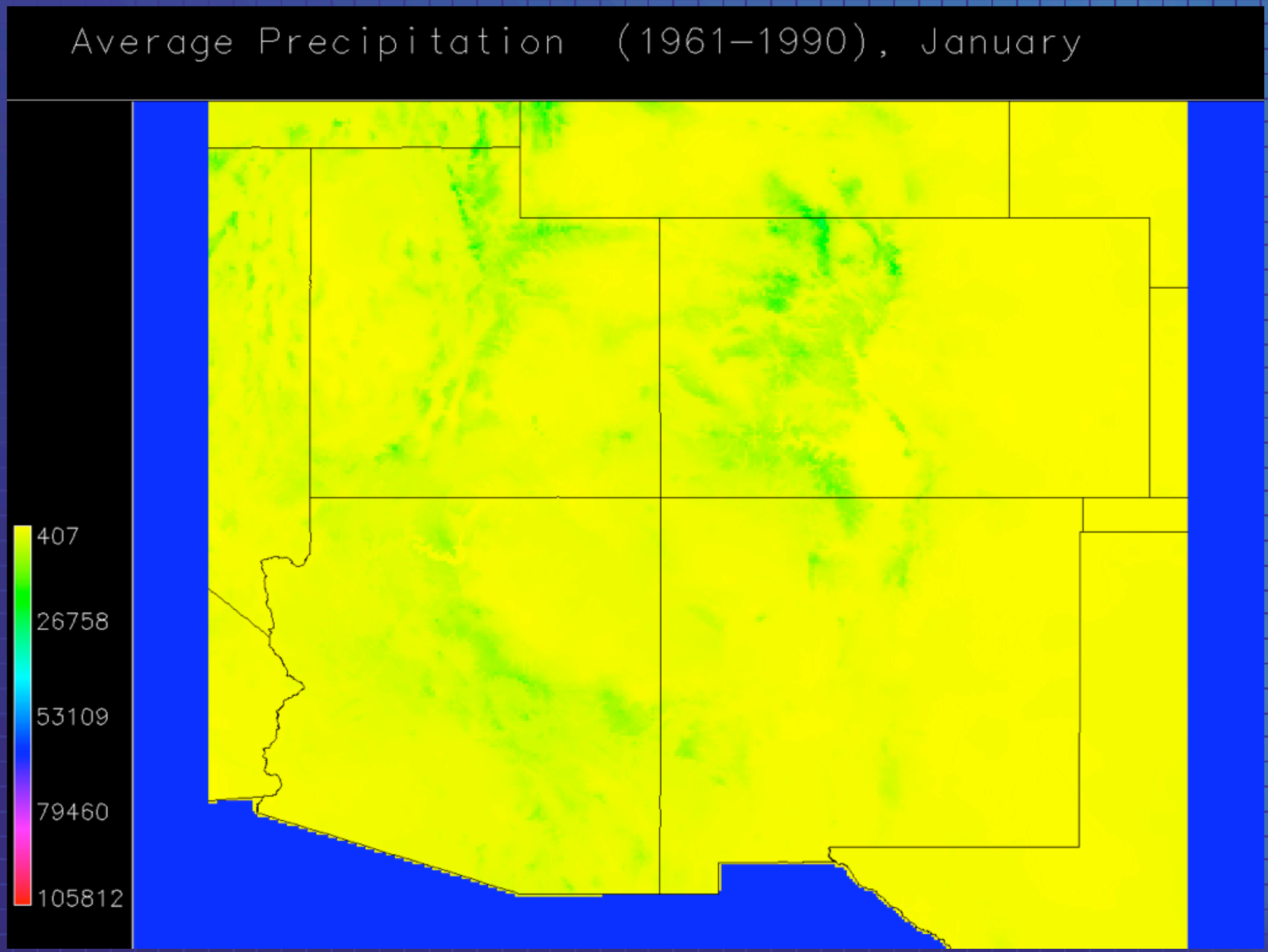
Script Examples 2 - Generating a PNG File within GRASS (cont.)

```
< continuation >

d.mon start=PNG          # start the PNG 'monitor'
d.mon select=PNG         # select the PNG 'monitor' for output

d.erase color=blue     # erase to a blue background

d.frame -c frame=legend at=0,90,0,10      # define legend frame
d.frame -c frame=title at=90,100,0,100    # define title frame
d.frame -c frame=map at=0,90,10,100       # define map frame

d.frame -s frame=legend                    # select the legend frame
d.erase color=black                        # erase legend frame to black
d.frame -s frame=title                     # select the title frame
d.erase color=black                        # erase title frame to black

d.frame -s frame=map                     # select the map frame
d.rast -o map=us_ppt.01.int              # display the precip raster
d.vect map=census_states type=boundary   # display the state boundaries

d.frame -s frame=legend                  # select the legend frame
d.legend map=us_ppt.01.int thin=50 \     # display the legend for
  at=5,50,5,20 color=white               # the precipitation raster

< continued >
```

Script Examples 2 - Generating a PNG File within GRASS (cont.)

```
< continuation >

d.frame -s frame=title                          # select the title frame
echo “Average Precipitation  \                  # print the title
  (1961-1990), January” | d.text
  size=40 color=white at=5,50

d.mon stop=PNG                                  # Stop the PNG driver and
                                                # close the generated file


# End of script ...
```

Script Examples 2 - Result



Script Examples 3 - Generation of Variable Outputs Based On Input

The execution of a script may be fundamentally modified through the use of simple command line parameters - values that are provided with the command.

The BASH shell handles provided parameters through simple numbered environment variables:

`$1` = the first parameter
`$2` = the second parameter

`$9` = the ninth parameter

References to these parameters wherever the actual values are needed

Script Examples 3 - Generation of Variable Outputs Based On Input (cont.)

The previously provided PNG generation script may be modified to accept three parameters:

- Parameter 1 - $1 - The raster to display
- Parameter 2 - $2 - The title to print on the generated PNG
- Parameter 3 - $3 - The name of the PNG file to generate

The modified code would then generate a PNG file based on the following command:

```
> myscript.sh myrasterFile “The title” mypngFile.png
```

Script Examples 3 - Generation of Variable Outputs Based On Input (cont.)

```
#!/bin/bash
# sample script that generates a three frame PNG file
# that depicts the raster specified on the command
# line as the first parameter changes the title to
# match the second parameter and saves the file to
# the filename specified as the third command
# line parameter
#
# wk12_sample3.sh <input raster> <Title Text>
<output PNG file>

# GRASS Commands
export GRASS_WIDTH=1280
export GRASS_HEIGHT=960
export GRASS_PNGFILE=$3
export GRASS_BACKGROUNDCOLOR=050505
export GRASS_TRANSPARENT=TRUE
export GRASS_TRUECOLOR=TRUE
export GRASS_PNG_COMPRESSION=1
export GRASS_PNG_AUTO_WRITE=FALSE

< continued >
```

Script Examples 3 - Generation of Variable Outputs Based On Input (cont.)

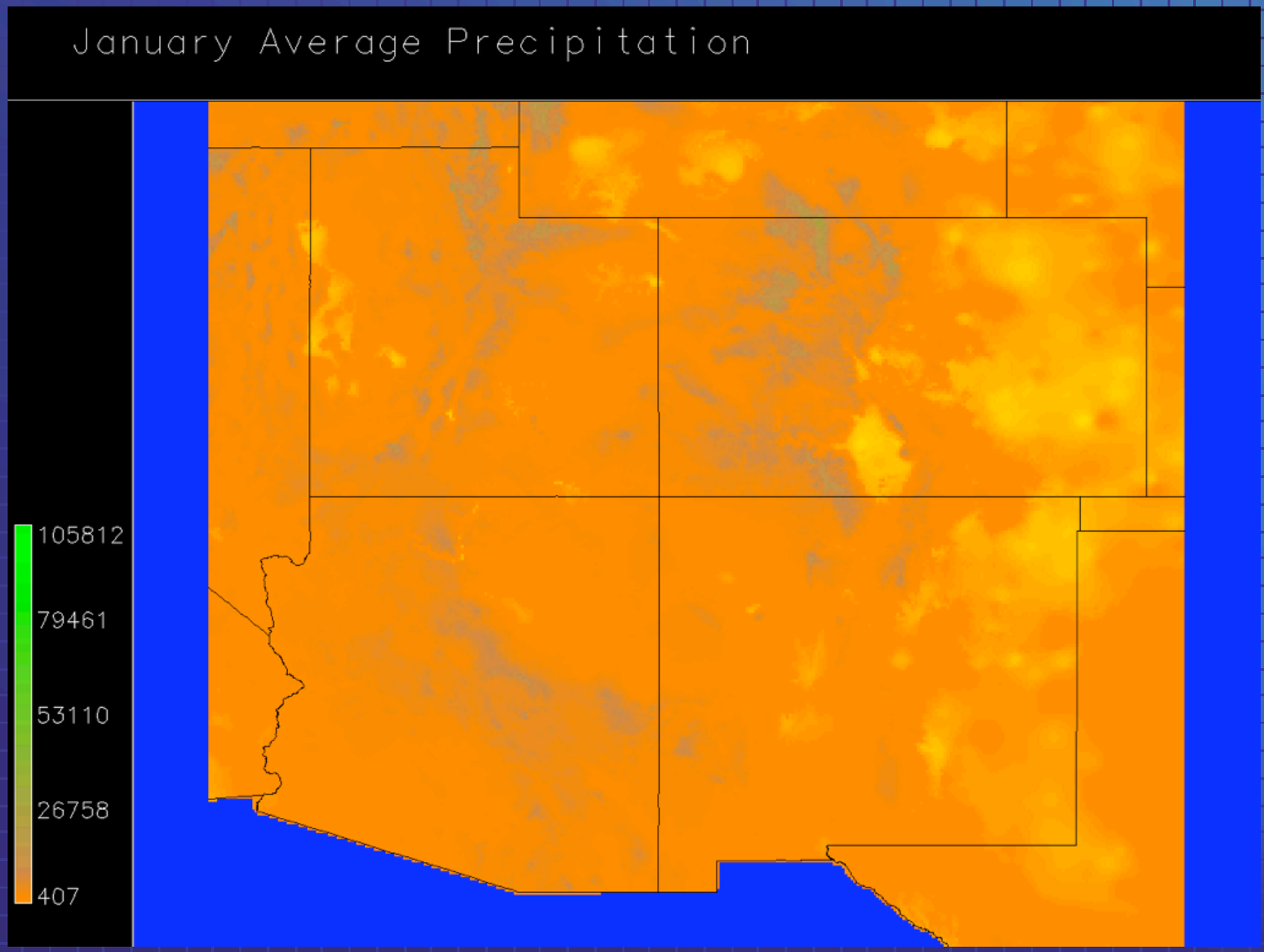
```
< continuation >

g.region region=sw

d.mon start=PNG          # start the PNG monitor
d.mon select=PNG         # select the PNG driver

d.erase color=blue       # erase to a blue background

d.frame -c frame=legend at=0,90,0,10      # define legend frame
d.frame -c frame=title at=90,100,0,100    # define title frame
d.frame -c frame=map at=0,90,10,100       # define map frame

d.frame -s frame=legend                   # select the legend frame
d.erase color=black                       # erase legend frame to black
d.frame -s frame=title                    # select the title frame
d.erase color=black                       # erase title frame to black

d.frame -s frame=map                    # select the map frame
d.rast -o map=$1                        # display the precip raster
d.vect map=census_states type=boundary  # display the state boundaries

< continued >
```

Script Examples 3 - Generation of Variable Outputs Based On Input (cont.)

```
< continuation >

d.frame -s frame=legend                          # select the legend frame

# display the legend for the precipitation
raster d.legend map=$1 thin=50 at=5,50,5,20 color=white

d.frame -s frame=title                           # select the title frame
echo $2 | d.text  size=40 color=white at=5,50 # print the title

d.mon stop=PNG

# End of script ...
```

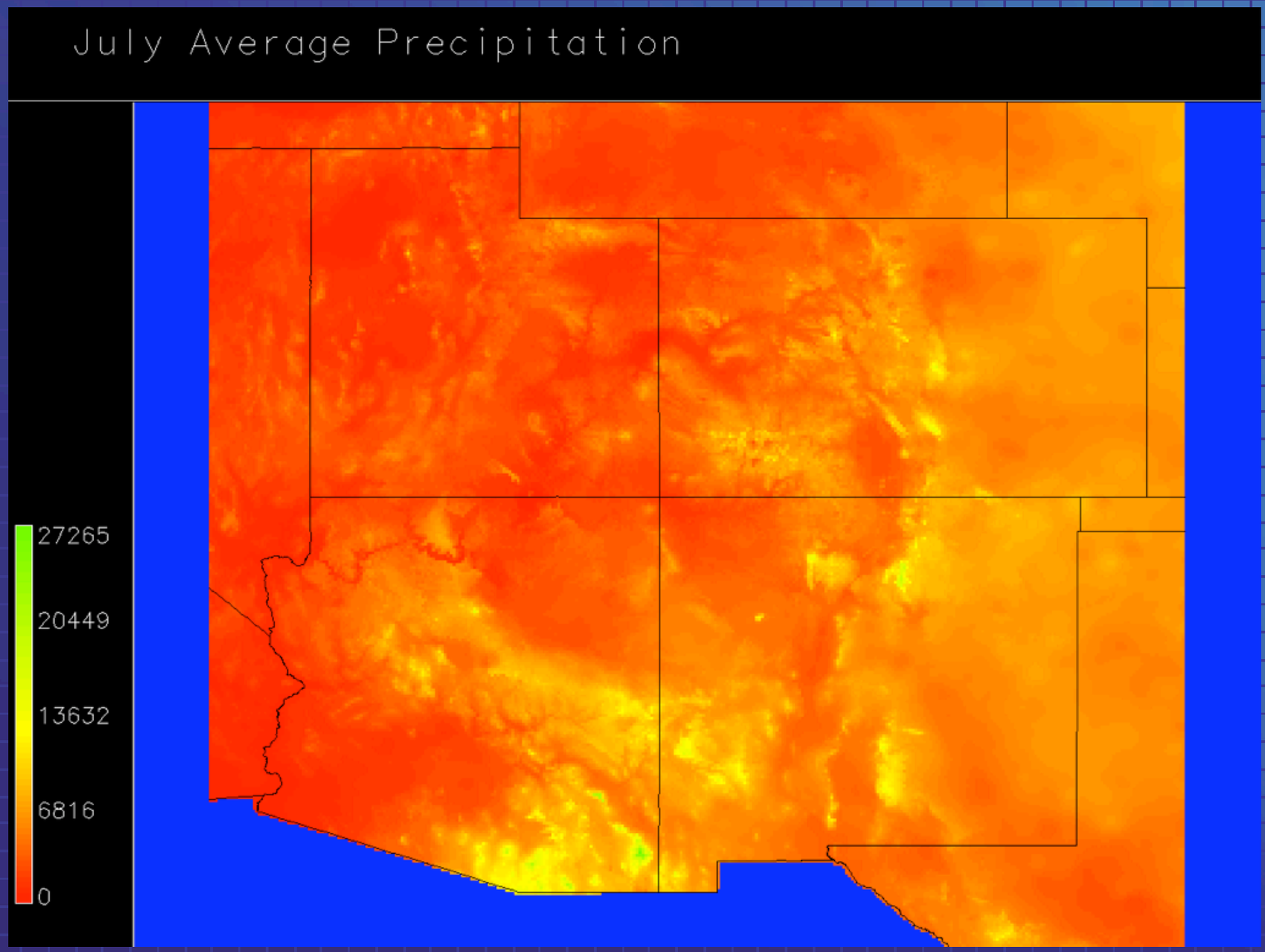
Script Examples 3 - Results

```
> ./wk12_sample3.sh us_ppt.01 "January Average Precipitation"
sample3_jan.pn
```



Script Examples 3 - Results

```
> ./wk12_sample3.sh us_ppt.07 "July Average Precipitation"
sample3_jul.png
```



Script Examples 4 - Scripting Outside GRASS

- The previous script may be modified to execute outside of the GRASS application environment by setting the appropriate GRASS environment variables.
- The GRASSRC environment variable points to a configuration file that contains the paths for the current GRASS data base directory, location, and mapset. The specified file is an ASCII text file that consists of the following:

```
GISDBASE: /Users/kbene/shared/teaching/grass.data
GRASS_GUI: text
MAPSET: PERMANENT
LOCATION_NAME: geog499_prism
```

- The GRASSRC environment variable may point to any file on the system that contains the necessary information.

Script Examples 4 - Scripting Outside GRASS (cont.)

- The GRASSBASE variable points to the base of the GRASS installation on the system - i.e. where the GRASS libraries are installed

  `/usr/lib/grass` on the class CD

- Depending upon the configuration of your system, additional information will need to be added to the PATH environment variable to point to the specific locations of the GRASS binary files and scripts.

  `/usr/lib/grass/bin`

  `/usr/lib/grass/scripts`

  on the class CD

Script Examples 4 - Scripting Outside GRASS (cont.)

The beginning of the script presented in Example 3 has been modified to run outside of GRASS. The rest of the script is the same.

```
# Set GRASS environment variables for execution
# outside GRASS
#GISBASE, GISRC, PATH (include GRASS binary path)
export GISBASE="/usr/lib/grass"
export GISRC="/Users/kbene/.grassrc_prism"
export PATH=$PATH:/usr/lib/grass/bin
export PATH=$PATH:/usr/lib/grass/scripts
```

Script Examples 4 - Result