UNIX Graphics Overview

A. R. Feuer

Bell Laboratories
Murray Hill, New Jersey 07974

1. INTRODUCTION

UNIX† Graphics, or just *graphics*, is the name given to a growing collection of numerical and graphical commands available as part of UNIX [1]. In its initial release, *graphics* includes commands to construct and edit numerical data plots and hierarchy charts. This memorandum will help you get started using *graphics* and show you where to find more information. The examples below assume that you are familiar with the UNIX shell [1].

2. BASIC CONCEPTS

The basic approach taken in *graphics* is to generate a drawing by describing it rather than by drafting it. Any drawing is seen as having two fundamental attributes: its underlying logic and its visual layout. The layout encompasses one representation of the logic. For example, consider the attributes of a drawing that consists of a plot of the function $y=x^2$ for x between 0 and 10. The logic of the plot is the description as just given, viz. $y=x^2, 0 \leq x \leq 10$. The layout consists of an x-y grid, axes labeled perhaps 0 to 10 and 0 to 100, and lines drawn connecting the x-y pairs 0,0 to 1,1 to 2,4 and so on.

The way to generate a picture in *graphics* is

gather data | transform the data | generate a layout | display the layout.

To generate the specific plot of $y=x^2, 0 \leq x \leq 10$ and display it on a Tektronix display terminal would be

```
gas -s0,t10 | af "x^2" | plot | td
```

where

gas generates sequences of numbers, in this case starting at 0 and terminating at 10.

af performs general arithmetic transformations.

plot builds x-y plots.

td displays drawings on Tektronix terminals.

The resulting drawing is shown in Figure 1.

The layout generated by a *graphics* program may not always be precisely what is wanted. There are two ways to influence the layout. Each drawing program accepts options to direct certain layout features. For instance, in the previous example we may have wanted the x-axis labels to indicate each of the numbers plotted and we might not have wanted any y-axis labels at all. To achieve this the *plot* command would be changed to:

```
plot -xi1,ya
```

producing the drawing of Figure 2.

---

† UNIX is a trademark of Bell Laboratories.

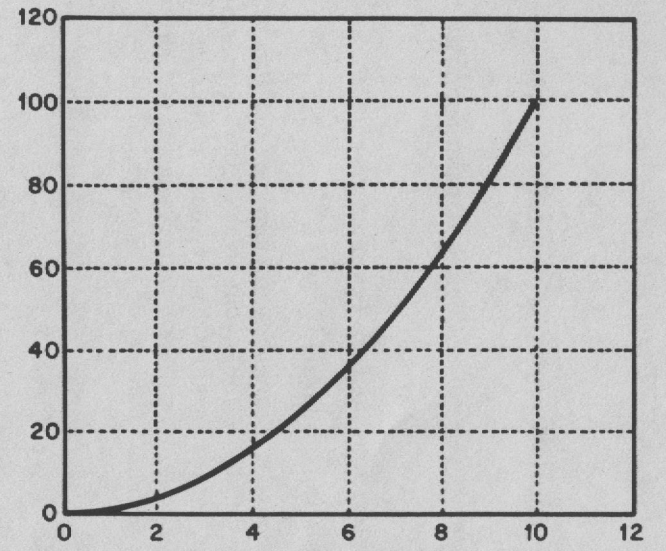

Figure 1. gas -s0,t10 | af "x^2" | plot | td

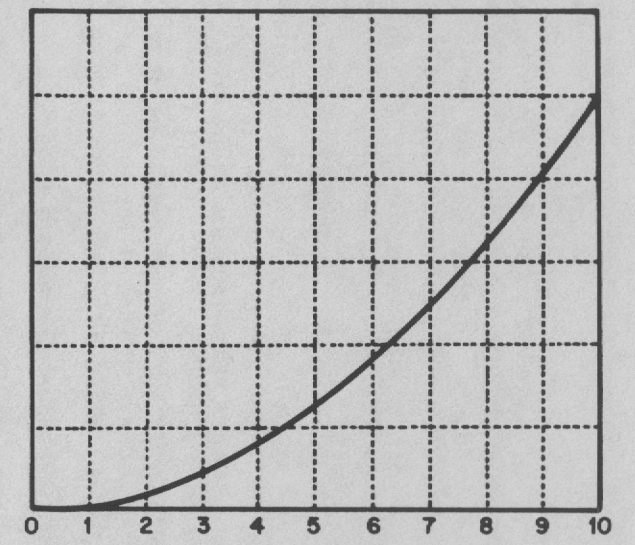

Figure 2. gas -s0,t10 | af "x^2" | plot -xl1,ya | td

The output from any drawing command can also be affected by editing it directly at a display terminal using the graphical editor, *ged*. To edit a drawing really means to edit the computer representation of the drawing. In the case of *graphics* the representation is called a graphical primitive string, or GPS. All of the drawing commands (e.g., *plot*) write GPS and all of the device filters (e.g., *td*) read GPS. *Ged* allows you to manipulate GPS at a display terminal by interacting with the drawing the GPS describes.

GPS describes graphical objects drawn within a Cartesian plane 65,534 units on each axis. The plane, known as the *universe*, is partitioned into 25 equal sized square regions. Multi-drawing displays can be produced by placing drawings into adjacent regions and then displaying each region.

3. GETTING STARTED

To access the *graphics* commands when logged in on a UNIX system type **graphics**. Your *shell* variable **PATH** will be altered to include the *graphics* commands and the *shell* primary prompt will be changed to ". Any command accessible before typing **graphics** will still be accessible; *graphics* only adds commands, it doesn't take any away. Once in *graphics*, you can find out about any of the *graphics* commands using *whatis*. Typing **whatis** by itself on a command line will generate a list of all the commands in *graphics* along with instructions on how to find out more about any of them.

All of the *graphics* commands accept the same command line format:

A *command* is:	a *command-name* followed by *argument*(s).
A *command-name* is:	the name of any of the *graphics* commands.
An *argument* is:	a *file-name* or an *option-string*.
A *file-name* is:	any file name not beginning with −, or a − by itself to reference the standard input.
An *option-string* is:	a − followed by *option*(s).
An *option* is:	letter(s) followed by an optional value. Options may be separated by commas.

You will get the best results with *graphics* commands if you use a display terminal. *Tplot*(1G) filters can be used in conjunction with *gtop* (see *gutil*(1G)) to get somewhat degraded drawings on Versatec printers and Dasi-type terminals. And since GPS can be stored in a file, it can be created from any terminal for later displaying on a graphical device.

To remove the *graphics* commands from your **PATH** *shell* variable type **EOT** (control-d on most terminals). To log off UNIX from *graphics* type **quit**.

4. EXAMPLES OF WHAT YOU CAN DO

4.1 Numerical Manipulation and Plotting

Stat(1G) describes a collection of numerical commands. All of these commands operate on vectors. A vector is a text file that contains numbers separated by delimiters, where a delimiter is anything that is not a number. For example:

```
1 2 3 4 5, and
arf tty47 Mar 5 09:52
```

are both vectors. (The latter being the vector: 47 5 9 52.)

Here is an easy way to generate a Celsius-Fahrenheit conversion table using *gas* to generate the vector of Celsius values:

```
gas −s0,t100,i10 | af "C,9/5*C+32"
```

The output is:

```
0.0     32
10      50
20      68
30      86
40      104
50      122
60      140
70      158
80      176
90      194
100     212
```

This is what is going on:

gas −s0,t100,i10 We have seen *gas* in an earlier example. In this case the sequence starts at 0, terminates at 100, and the increment between successive elements is 10.

af "C,9/5*C+32" We have also seen *af*. Arguments to *af* are expressions. Operands in an expression are either constants or file names. If a file name is given that does not exist in the current directory it is taken as the name for the standard input. In this example **C** references the standard input. The output is a vector with odd elements coming from the standard input and even elements being a function of the preceding odd element.

Here is an example that illustrates the use of vector titles and multiline plots:

```
gas | title −v"first ten integers" >N
root N >RN
root −r3 N >R3N
root −r1.5 N >R1.5N
plot −FN,g N R1.5N RN R3N | td
```

The resulting plot is shown in Figure 3.

title −v"*name*" *Title* associates a *name* with a vector. In this case, **first ten integers** is associated with the vector output by *gas*. The vector is stored in file **N**.

root −r*n* *Root* outputs the *n*th root of each element on the input. If **−rn** is not given then the square root is output. Also, if the input is a titled vector the title will be transformed to reflect the root function.

plot −F*X,g Y(s)* This command generates a multiline plot with *Y(s)* plotted versus *X*. The **g** option causes tick marks to appear instead of grid lines.

The next example generates a histogram of random numbers:

```
rand −n100 | title −v"100 random numbers" | qsort | bucket | hist | td
```

The output is shown in Figure 4.

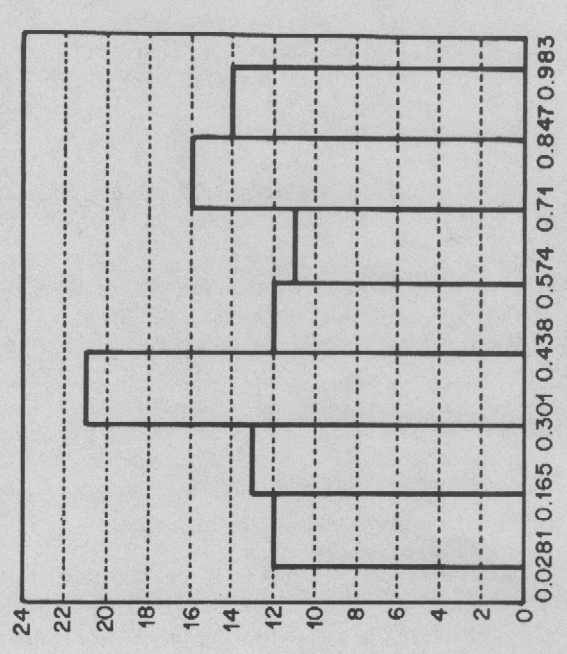

Figure 4. Histogram of 100 random numbers

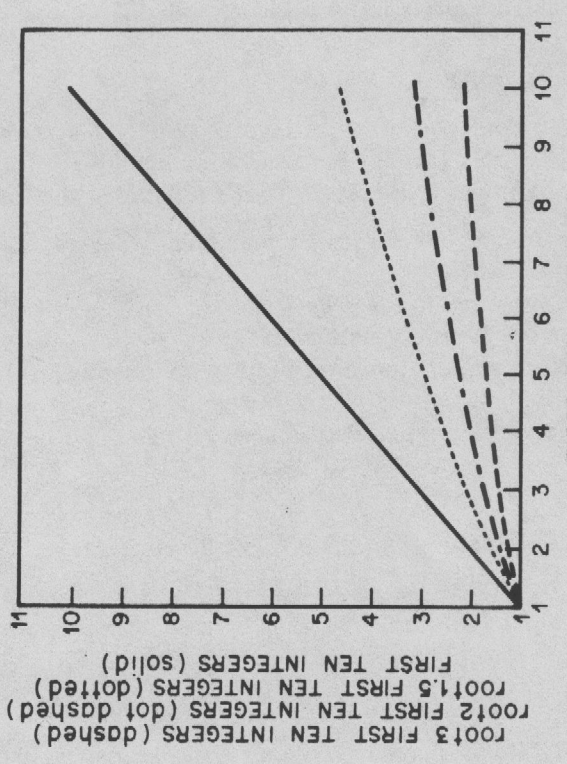

Figure 3. Some roots of the first ten integers

rand −n100	*Rand* outputs random numbers using *rand*(3C). In this case 100 numbers are output in the range 0 to 1.
qsort	*Qsort* sorts the elements of a vector in ascending order.
bucket	*Bucket* breaks the range of a vector into intervals and counts how many elements from the vector fall into each interval. The output is a vector with odd elements being the interval boundaries and even elements being the counts.
hist	*Hist* builds a histogram based on interval boundaries and counts.

4.2 Drawings Built from Boxes

There is a large class of drawings composed from boxes and text. Examples are structure charts, configuration drawings, and flow diagrams. In *graphics* the general procedure to construct such box drawings is the same as that for numerical plotting. Namely gather and transform the data, build and display the layout.

As an example, consider hierarchy charts. The command line:

dtoc | vtoc | td

outputs the drawing shown in Figure 5.

Dtoc outputs a table of contents that describes a directory structure (Figure 5a). The fields from left to right are level number, directory name, and the number of ordinary readable files contained in the directory. *Vtoc* reads a (textual) table of contents and outputs a visual table of contents, or hierarchy chart. Input to *vtoc* consists of a sequence of entries, each describing a box to be drawn. An entry consists of a level number, an optional style field, a text string to be placed in the box, and a mark field to appear above the top right hand corner of the box.

5. WHERE TO GO FROM HERE

The best way to learn about *graphics* is to log onto a UNIX system and use it. Tutorials exist for *stat*(1G) [3] and *ged*(1G) [4]; [2] contains administrative information for *graphics*. Reference information can be found in the *UNIX User's Manual* in the following manual entries:

gdev(1G), a collection of commands to manipulate Tektronix 4000 series terminals; and
ged(1G), the graphical editor;
graphics(1G), the entry point for *graphics*;
gutil(1G), a collection of utility commands;
stat(1G), numerical manipulation and plotting commands;
toc(1G), routines to build tables of contents;
gps(5), a description of a graphical primitive string.

6. REFERENCES

[1] T. A. Dolotta, S. B. Olsson, and A. G. Petruccelli (eds.). *UNIX User's Manual*—Release 3.0, Bell Laboratories (June 1980).

[2] R. L. Chen, D. E. Pinkston, and A. Guyton. *Administrative Information for the UNIX Graphics Package*, Bell Laboratories.

[3] A. R. Feuer and A. Guyton. *STAT—A Tool for Analyzing Data*, Bell Laboratories.

[4] A. R. Feuer. *A Tutorial Introduction to the Graphics Editor*, Bell Laboratories.

Figure 5. Directory Structure for Graphics

```
0.      "source"        2
1.      "glib.d"        1
1.1.    "gpl.d"         12
1.2.    "gsl.d"         14
2.      "gutil.d"       6
2.1.    "cvrtopt.d"     7
2.2.    "gtop.d"        8
2.3.    "ptog.d"        5
3.      "stat.d"        54
4.      "tek4000.d"     5
4.1     "ged.d"         37
4.4.    "td.d"          8
5.      "toc.d"         3
5.1.    "ttoc.d"        3
5.2.    "vtoc.d"        22
6.      "whatis.d"      108
```

Figure 5a. Dtoc output

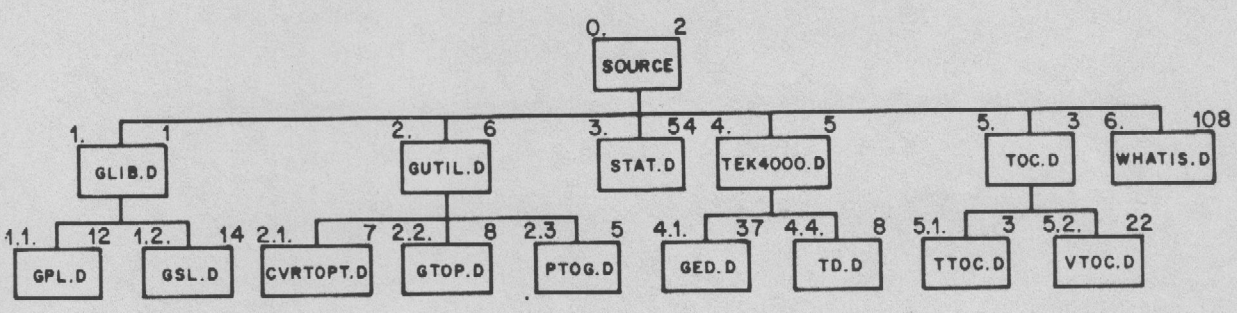

Figure 5b. Vtoc output