Denkh

A tool for generating PDF files

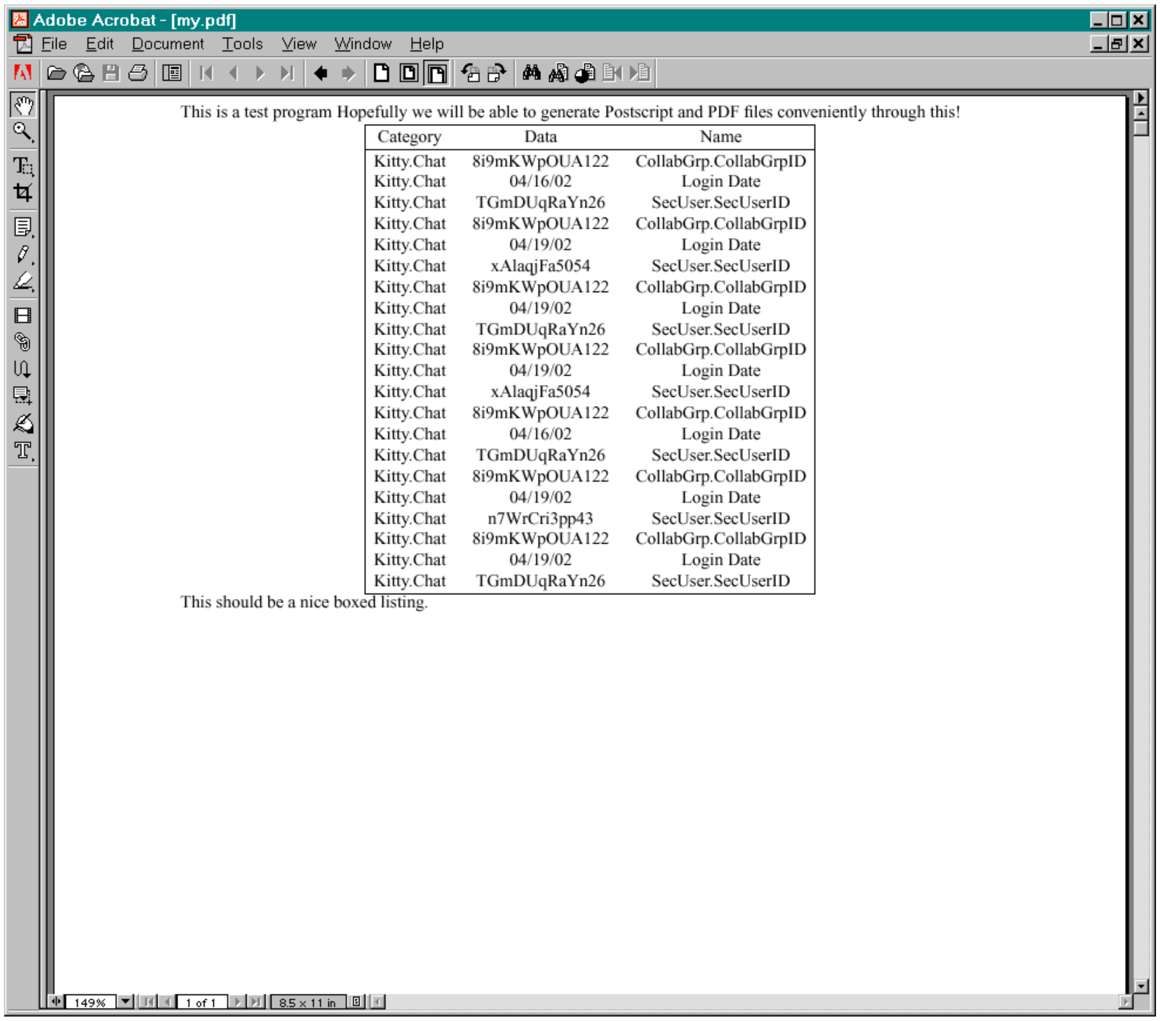
Amduus Information Works, Inc.
www.amduus.com

Table of Contents:

License

Introduction

More and more often companies are asking for output in PDF files. They can be more visually sophisticated with fonts, bolding, lines, bar graphs, etc. One of the problems with HTML is the lack of page numbering on listings. This can be accomplished in PDF with headers and footers available too!



This is a test program Hopefully we will be able to generate Postscript and PDF files conveniently through this!

Category	Data	Name
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/16/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	xAlaqjFa5054	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	xAlaqjFa5054	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/16/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	n7WrCri3pp43	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID

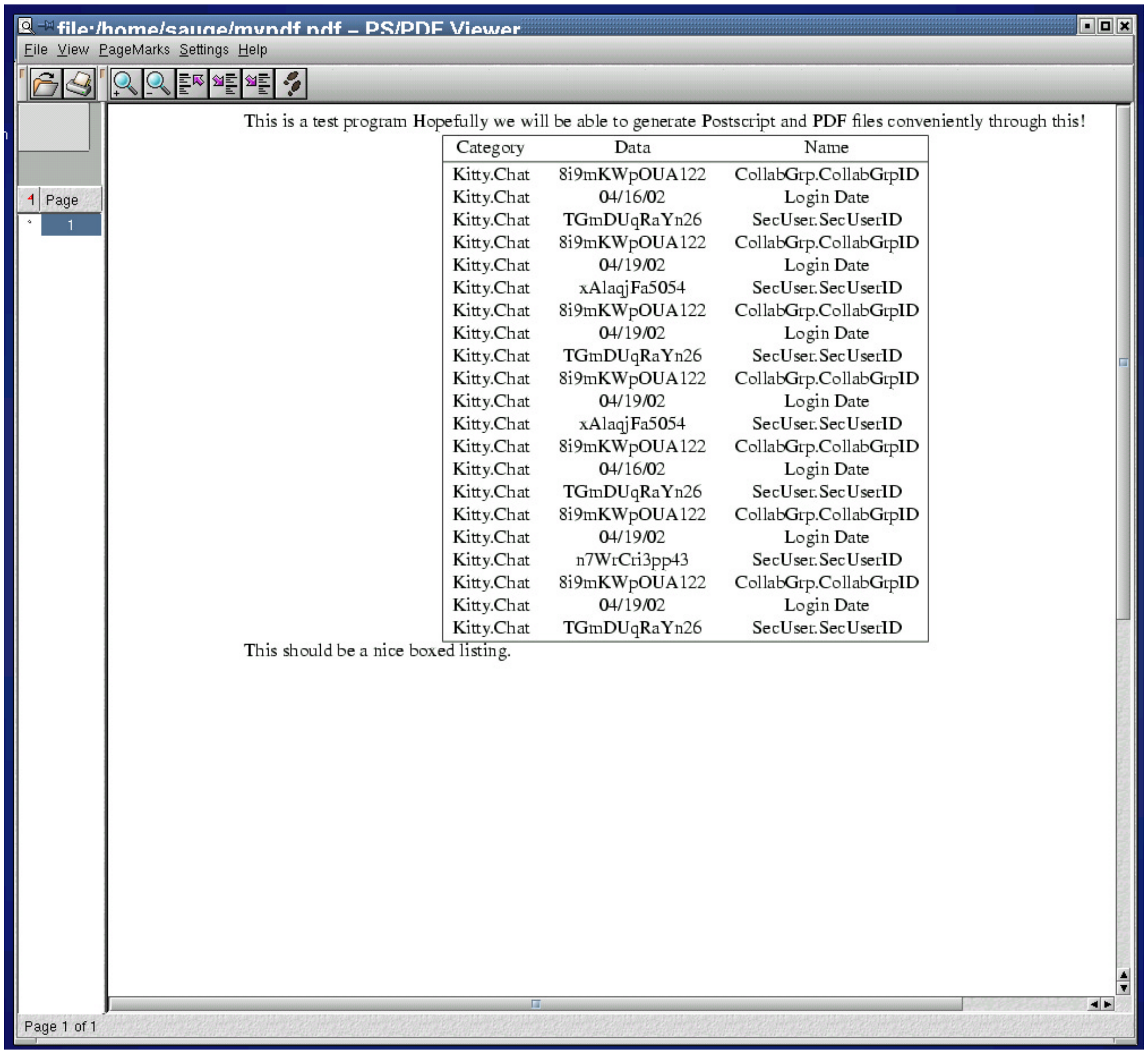
This should be a nice boxed listing.

Viewing a simple PDF file on Windows

Images can be placed into the files using encapsulated postscript (can be created on Corel Draw or other image manipulation programs) and included in the report.

PDF files are also known as non-proprietary formats. They can be transported between many operating systems without changes to the file format. The same file can be read on UNIX, Linux, Windows, and Apple computers.

PDF files are also read-only. With special programs, they can be annotated, but it is very difficult to modify a file unless the user is very technically adept.


file:/home/sauge/mypdf.pdf – PS/PDF Viewer
File View PageMarks Settings Help
Page
1
Page 1 of 1

This is a test program Hopefully we will be able to generate Postscript and PDF files conveniently through this!

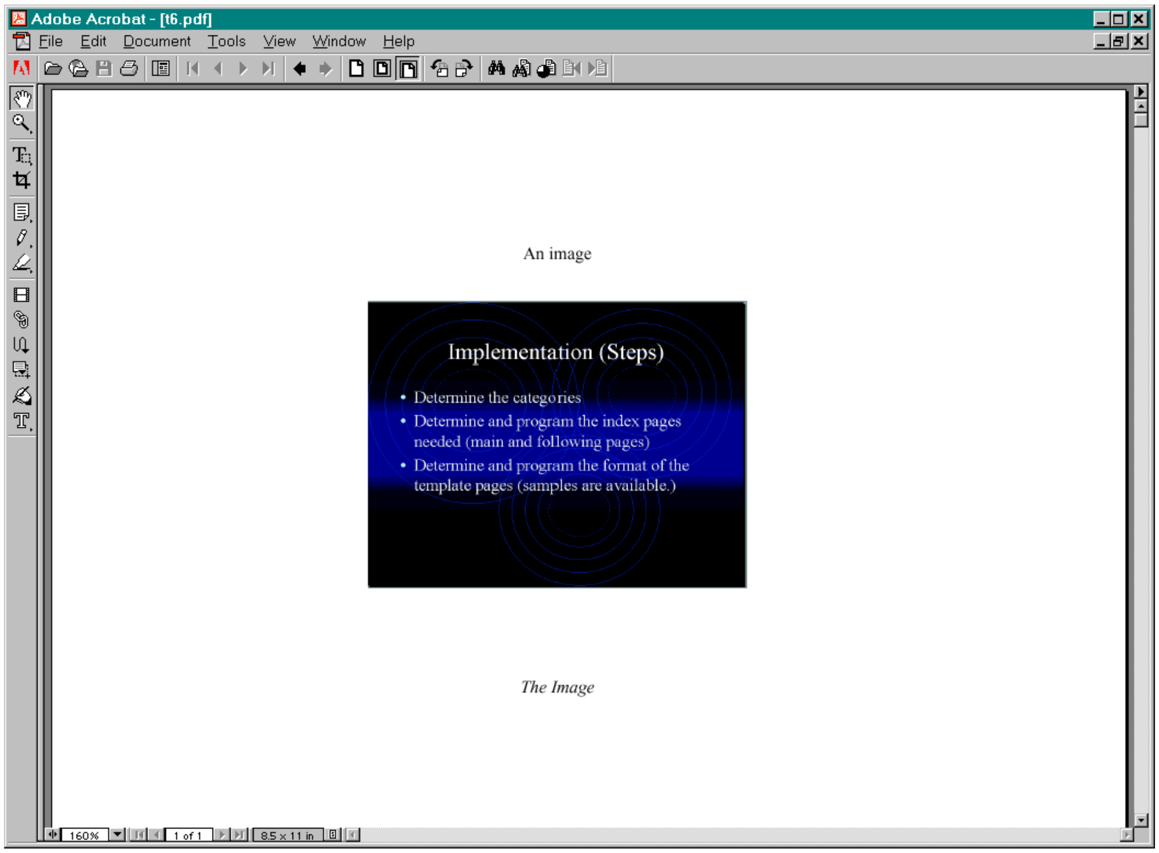
Category	Data	Name
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/16/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	xAlaqjFa5054	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	xAlaqjFa5054	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/16/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	n7WrCri3pp43	SecUser.SecUserID
Kitty.Chat	8i9mKWpOUA122	CollabGrp.CollabGrpID
Kitty.Chat	04/19/02	Login Date
Kitty.Chat	TGmDUqRaYn26	SecUser.SecUserID

This should be a nice boxed listing.

Viewing the PDF file on Linux/UNIX

As you read the document, you will see that I use the word troff and groff intermixedly. This is because they are basically the same package. The groff package is the GNU (www.gnu.org) package that does the operations of the troff package found on AT&T UNIX systems. The groff package has been ported to many different operating systems such as Linux, Solaris, AIX, etc. and is open source.

You will need the groff package on your system to create the postscript files. To convert the postscript files into PDF files, you will need the Ghostscript package on your system.



An image

Implementation (Steps)

- Determine the categories
- Determine and program the index pages needed (main and following pages)
- Determine and program the format of the template pages (samples are available.)

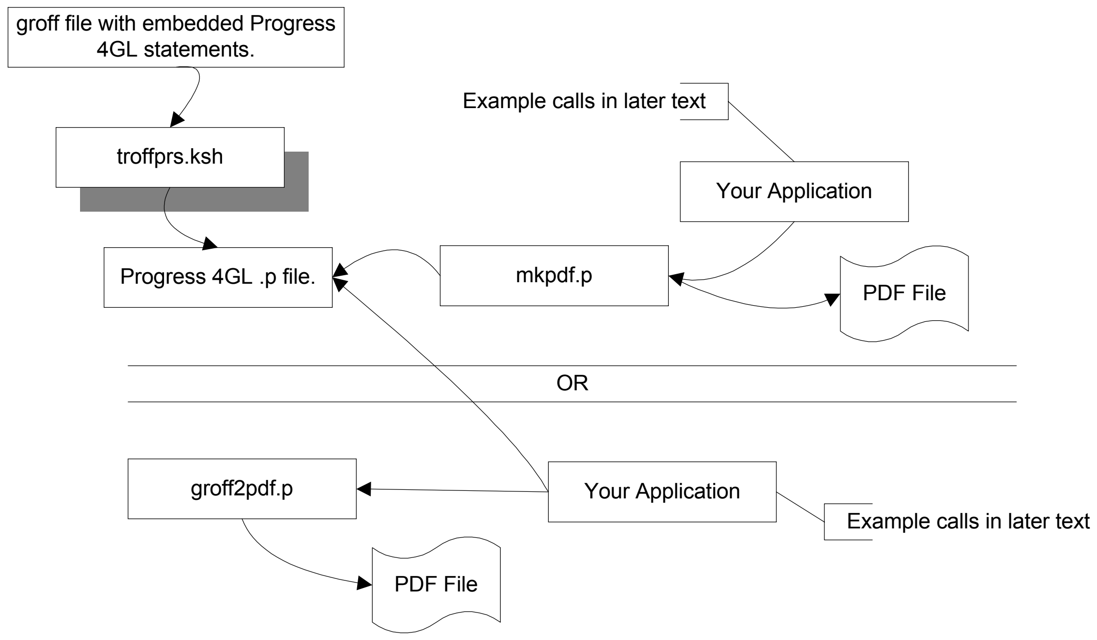
The Image

Example t6.troff shows how to insert an image into your PDF file

Amduus Information Works, Inc. provides training and support for this package. We can also help you create your reports if you do not understand groff formatting well.

The architecture

What are the pieces and how do they fit together? Below is a diagram that lays out everything:


groff file with embedded Progress 4GL statements.
troffprs.ksh
Progress 4GL .p file.
Example calls in later text
Your Application
mkpdf.p
PDF File
OR
groff2pdf.p
Your Application
Example calls in later text
PDF File

The groff file provides the formatting instructions for your PDF file. Inside the file are embedded 4GL statements to pull up data from the DB and place it into the PDF file.

There is a command that converts this file with embedded 4Gl statements into a normal 4GL program file.

One calls the program file in two ways. One way is to use the mkpdf.p program that assumes there is only one parameter to your file.

The other way is to call your file directly which will output a groff file with all the data in it. Then you would call the groff2pdf.p program which converts that into a pdf file of your naming and placement.

From there on, you would handle the pdf file under your application's logic.

Using the software:

In short the steps to using the software are these:

1. Create a file with the formatting tags for groff.[1]
2. Embed the 4GL statements into the file using the <!--TS and --> tags like one would for a SpeedScript file in Webspeed.[2]
3. Run the file through troffprs.ksh which will yield a pure 4GL program for use in your character, GUI, or Web environment.

Once you have the new .p file, you have two options of creating the PDF.

1. Use the mkpdf.p program to create a pdf file. This program is very limited however.

OR

1. Call the report program directly, always including where it should put it's groff file.
2. Call the groff2pdf.p program which converts this file into a pdf file and deletes the groff file.

[1] These tags can be found on the internet or in computer bookstores. Amduus Information Works, Inc. can create these files for you given the format you give, or we can give you training on the tags. Many of the most common tags are wrapped in 4GL functions documented later in this book.
[2] Webspeed programmers using the E4GL style of programming will pick up on this with little trouble.

Determine the format

The formatting is done with groff format rules. This is a very well known package if you are from the "old school" of UNIX programming. The package was used to create publications and is very powerful.

Here is the groff listing with embedded 4GL statements:

```
This is a test program Hopefully we will be able to generate Postscript
and PDF files conveniently through this!

.TS
center box tab(|);
c c c.
Category|Data|Name
_
<!--TS
FOR EACH WebState NO-LOCK:
-->
`WebState.Category`|`Webstate.Data`|`Webstate.Name`
<!--TS
END.
-->
.TE

This should be a nice boxed listing.
```

This is the embedded groff file that created the PDF files shown in the introduction area. It starts with some text which is merely placed in the file. One can see there are table tags showing the table should be centered contained in a box with the data delimiter set to | symbol. The data items are centered in each column, and the column headers are Category, Data, and Name. There should be a line between the header line and the data lines as shown with the _ on it's own line.

We see that the 4GL statements are delimited by <!--TR on it's own line and by --> on it's own line. Progress introduced the term SpeedScript, and so I will refer to this kind of coding as TroffScript. Within these tags you can use 4GL in anyway that you normally would in any 4GL program.

Outside the tags, if you wish to refer to some data value, they should be delimited with the ` (back tic) character. As you will see in the following section explaining the generated source code, one should consider lines outside of the TroffScript tags similar to PUT statements, using the ` to denote the data contained in a variable/field data instead the actual name.

Conversion of the embedded groff file into the Progress procedure file

Once the groff file has been completed, it is time to convert the file into a progress procedure that can be directly called by progress application programs.

This is done with the troffprs.ksh script. It is found in the /script directory of the installation directory. Be sure to edit this script to set the PROPATH and DLC environmental variables correctly. It does not need to be connected to a database to function.

It is called like this:

```
troffprs.ksh [embedded_groff_file_name] [progress_procedure_file_name]
```

use this as an example:

```
troffprs.ksh test.troff test.p
```

This will translate test.troff into the progress procedure test.p.

Lets see what this command did.

Examination of the resulting progress procedure

The groff file is converted into a source code file for Progress applications. This file can be run directly from other progress .p files.

It presents a warning in comments that the file was generated via a process. This lets the programmers know that if they manipulate this file, and then another programmer manipulates the groff file, the changes will be lost. The programmer should always manipulate the groff file.

```
/* Warning - this is a generated file from test.troff */
/* Generated by tools from Amduus Information Works, Inc. */
/* http://www.amduus.com    Scott Auge sauge@amduus.com */



DEF INPUT PARAMETER cOutputFile AS CHARACTER NO-UNDO.
DEF STREAM StdOut.
OUTPUT STREAM StdOut TO VALUE(cOutputFile).



PUT STREAM StdOut UNFORMATTED 'This is a test program~n'.
PUT STREAM StdOut UNFORMATTED 'Hopefully we will be able to generate
Postscript and PDF files conveniently~n'.
PUT STREAM StdOut UNFORMATTED 'through this!~n'.
PUT STREAM StdOut UNFORMATTED '.TS~n'.
PUT STREAM StdOut UNFORMATTED 'center box tab(|);~n'.
PUT STREAM StdOut UNFORMATTED 'c c c.~n'.
PUT STREAM StdOut UNFORMATTED 'Category|Data|Name~n'.
PUT STREAM StdOut UNFORMATTED '_~n'.
FOR EACH WebState NO-LOCK:


PUT STREAM StdOut UNFORMATTED '' WebState.Category '|' Webstate.Data
'|' Webstate.Name '~n'.
END.


PUT STREAM StdOut UNFORMATTED '.TE~n'.
PUT STREAM StdOut UNFORMATTED 'This should be a nice boxed listing.~n'.

OUTPUT STREAM StdOut CLOSE.
```

Next the program automatically inserted an input parameter that will receive the name of the output file the program should put it's groff formatted results in. You can of course use the tags in the embedded 4GL groff file to add your own input parameters, just remember that they will follow the default parameter presented here.

After this, the program begins outputting the static text and inserts the embedded 4GL in the proper location relative to that static text. These lines are coded in blue to help set them out better.

Of particular of interest are the lines using the ` character. This line is coded in red to help it stand out better. Note how the items contained within back tic characters have become part of a PUT STREAM StdOut statement. The portions of line not in the back tic character have become static strings in the statement.

As I am sure you can guess, we are not done yet. Let's examine how we call this new program from your application program to get a PDF file generated.

Calling the generated programs

There are two main ways of calling the report program generated above. One is through the mkpdf.p program, and the other is by calling it directly and then calling the groff2pdf.p program.

Calling via mkpdf.p

This routine is good if you do not have any extra parameters in the report program. It takes the arguments of what the program name is, what the pdf file name should be, and where to place that file name.

```
run mkpdf.p ("testing/test.p", "my.pdf", "/tmp").
```

Calling the program directly and using groff2pdf.p

This method requires you to remember the troff file that is created by the report program. However, it does allow you to call the program directly, including any additional parameters you may have placed into the file. Below is a sample calling needed.

```
run testing/test.p ("/tmp/mypdf.troff").
run groff2pdf.p ("/tmp/mypdf.troff", "/tmp/mypdf.pdf").
```

The groff2pdf.p program pulls in the troff file generated by the report program and creates a PDF file placing it into the location specified in the second argument.

Both these methods call down into the Groff and GhostScript packages on the system they are executing on. The 4GL programmer does not need to know about these packages or how to invoke them as the routines do that for them.

Using Denkh to convert Progress Report Programs to PDF

A programmer can automatically change a report generated by Progress into a PDF with the example calls below. Programmers need not change the report program to get a PDF file out of it.

First the programmer has a program issue a call to the report generating program. Note this program can be GUI/CHUI/WWW interfaced. In this simple program, we have hard coded the output file the report went into. The programmer will need to track the report generated by the user.

```
run examples/text1.p.
```

This program outputs it's results into /appl/Denkh/src/examples/text1.rpt.

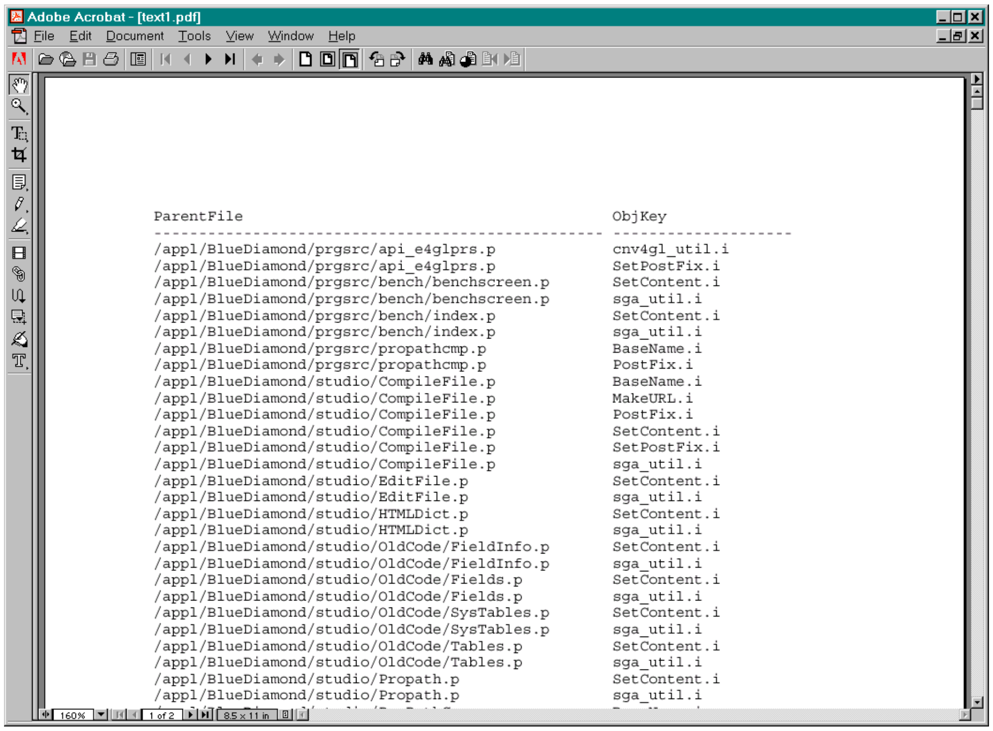
The beginning of using the Denkh system to translate the program into a PDF based report is done with two simple calls.

One then calls the txt2groff.p program which takes the program output and translates it into troff markup for the groff2pdf.p program. Note that the output file is listed first, followed by the input file.

```
run txt2groff.p
("/appl/Denkh/src/examples/text1.troff",
 "/appl/Denkh/src/examples/text1.rpt").

run groff2pdf.p
("/appl/Denkh/src/examples/text1.troff",
 "/appl/Denkh/src/examples/text1.pdf").
```

The user will then have a file called text1.pdf that contains the report. Here is a sample picture of the report in Acrobat.

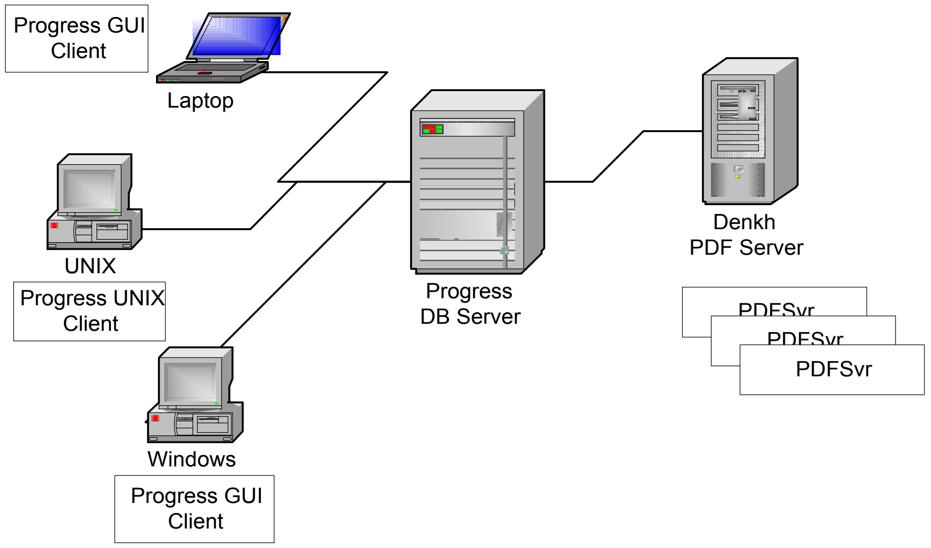

Adobe Acrobat - [text1.pdf]
File Edit Document Tools View Window Help
160%
1 of 2
8.5 x 11 in

```
ParentFile                                         ObjKey
-------------------------------------------------- --------------------
/appl/BlueDiamond/prgsrc/api_e4glprs.p             cnv4gl_util.i
/appl/BlueDiamond/prgsrc/api_e4glprs.p             SetPostFix.i
/appl/BlueDiamond/prgsrc/bench/benchscreen.p       SetContent.i
/appl/BlueDiamond/prgsrc/bench/benchscreen.p       sga_util.i
/appl/BlueDiamond/prgsrc/bench/index.p             SetContent.i
/appl/BlueDiamond/prgsrc/bench/index.p             sga_util.i
/appl/BlueDiamond/prgsrc/propathcmp.p              BaseName.i
/appl/BlueDiamond/prgsrc/propathcmp.p              PostFix.i
/appl/BlueDiamond/studio/CompileFile.p             BaseName.i
/appl/BlueDiamond/studio/CompileFile.p             MakeURL.i
/appl/BlueDiamond/studio/CompileFile.p             PostFix.i
/appl/BlueDiamond/studio/CompileFile.p             SetContent.i
/appl/BlueDiamond/studio/CompileFile.p             SetPostFix.i
/appl/BlueDiamond/studio/CompileFile.p             sga_util.i
/appl/BlueDiamond/studio/EditFile.p                SetContent.i
/appl/BlueDiamond/studio/EditFile.p                sga_util.i
/appl/BlueDiamond/studio/HTMLDict.p                SetContent.i
/appl/BlueDiamond/studio/HTMLDict.p                sga_util.i
/appl/BlueDiamond/studio/OldCode/FieldInfo.p       SetContent.i
/appl/BlueDiamond/studio/OldCode/FieldInfo.p       sga_util.i
/appl/BlueDiamond/studio/OldCode/Fields.p          SetContent.i
/appl/BlueDiamond/studio/OldCode/Fields.p          sga_util.i
/appl/BlueDiamond/studio/OldCode/SysTables.p       SetContent.i
/appl/BlueDiamond/studio/OldCode/SysTables.p       sga_util.i
/appl/BlueDiamond/studio/OldCode/Tables.p          SetContent.i
/appl/BlueDiamond/studio/OldCode/Tables.p          sga_util.i
/appl/BlueDiamond/studio/Propath.p                 SetContent.i
/appl/BlueDiamond/studio/Propath.p                 sga_util.i
```

Using the Denkh PDF Server Software

The Denkh PDF Server software allows a highly flexible means of generating PDF files for the users of Progress clients. Progress clients can be GUI/WWW/CHUI.

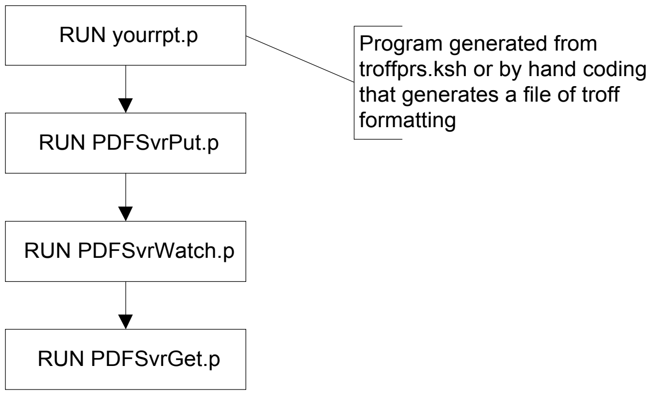
Architecture



The application that needs PDF reports run on various clients available from Progress Software. These clients use an API of routines to request reports of the PDFSvr process which is running on a server. This set up is required because the PDFSvr process needs Linux to properly render PDF files for the client application.

Using the Client API to request a PDF

Using the API is very simple. It is composed of three procedure files: one routine places a request onto the server, another watches the status of the request, and the final pulls the requested PDF (when available.)


RUN yourrpt.p
RUN PDFSvrPut.p
RUN PDFSvrWatch.p
RUN PDFSvrGet.p
Program generated from troffprs.ksh or by hand coding that generates a file of troff formatting

Using the example program from the supplied source found in the Denkh/src/examples directory, we run a program created from the t4.troff program. (See above for converting *.troff routines into .p routines.)

```
RUN examples/t4.p (INPUT "/tmp/t4.data").
```

When your operating system can handle generating PDFs directly with the Denkh package (Linux required), you can continue on as explained in the above section. When not, you need to call a Denkh PDF Server to render the report into PDF format for you.

First we make a call to the Put routine to put a request into the Denkh PDF Server:

```
RUN PDFClt/PDFSvrPut.p (INPUT "/tmp/t4.data", OUTPUT cReportID).
```

We send the filename of the output of the report program to the API, and receive a ReportID number. We need this ReportID number to reference the report in the other APIs. This is a very important number because one will loose the report in the wilds of the system – think of it as the request's name.

It may be that the Denkh PDF Server is busy processing other reports (or simply is not running) – hence an API is available to find out what is happening with the report. It returns the report's status. When the status becomes "DONE" then the Denkh PDF Server has processed the report and the programmer can call the PDFSvrGet.p routine to get the PDF report.

```
RUN PDFClt/PDFSvrWatch.p (INPUT cReportID, OUTPUT cStatus).
```

The status can be of the following:

SUBMITTED	The report has been submitted and awaiting processing
PROCESSING	The PDF Server is processing the report right now
LOCKED	The record about the report is LOCKED.
NO_RECORD	The report ID is bad, or the record has been lost
DONE	The PDF file is ready!

Once the report is ready for the client application, the PDFSvrGet.p routine is used to pull the report out of the system onto the file system of the client application:

```
RUN PDFClt/PDFSvrGet.p
(INPUT "/tmp/myname.pdf",
 INPUT cReportID,
 OUTPUT cStatus).
```

If the report is not yet ready, the routine will return the status of the report.

The arguments are the name of the report you wish the PDF file to have, the ReportID naming the report, and an output telling you the status of the report. Any status other than DONE will not result in a PDF file available for the user.

There after, refer to the report as a file on your client's machine, as in the example it would be `/tmp/myname.pdf.`

Administration of the Denkh PDFSvr process

Starting the Denkh PDF Server process:

```
Denkh/script/PDFSvr.ksh >> /tmp/DenkhLog
```

Be sure to tailor the script for your system's configuration.

There can be multiple processes running to help load distribution on heavily used systems, however, the rendering is generally very fast and many places need only one process running.

Groff for Windows

http://gnuwin32.sourceforge.net/packages/groffl.htm

I don't work on Windows, so if anyone wants to port this over to it, your welcome to do so. Above is a link the groff tools, however the ps2pdf tool cannot be found for Windows!

Conclusion

There you have it. The real magic is in learning the groff tags (or macros as they are preferably called). These tags allow the definition of fonts, sizing, page numbers, table of contents, etc. Very powerful stuff.

Creating a program that conveniently adds data from the Progress database is done very much like it is in SpeedScript for HTML pages. One can get really fancy sectioning out parts into internal procedures to be called over and over – as you can see, it truly is a progress application.

Sample Reports:

There are a few sample reports in the /src/examples directory of the install directory. The .troff file is the file composed by the programmer. The .p file is the generated tool, and the .pdf is a sample PDF file.

t1.troff	Simple text, simple table that does not cross pages
t2.troff	Simple text, headers and footers, table that crosses multiple pages, paragraphs, spacing, verticle bars in table heading with line across bottom, internal procedure used for rows of table.
t3.troff	Example of multiple page report for newspaper wraps. Hopefully one can see how this can work for invoices, mass letters, etc.
t4.troff	Example showing a Verticle and Horizontal Bar graph using the included procedures in the package.
t5.troff	Example of using a more complex table to display data
t6.troff	Example showing how to insert an image into your PDF file. Excellent for including company logo's and product pictures.

Quick Macro Reference:

This is the "me" macro package. Contact Amduus if you require it.

Tables:

You will want to get hold of the document "Tbl – A Program To Format Tables" by M. E. Lesk and L. L. Cherry. There are examples of tables and far more robust explanations of the macros.

Macros:

.TS H	Start of table
options;	See below
format.	See below
data	Data above .TH will header on multiple page tables
.TH	End of data shown on multiple page tables
data	Data for the table
.TE	End of table

Options:

center	center the table (default is left side)
expand	table becomes width of line for page
box	enclose the table in a box
allbox	all elements are in a box

doublebox	enclose table in double box
tab(x)	use x instead of tab to delimit data items.
linesize(n)	set box lines to n point size

Format:

l	left align
r	right align
c	center align
n	numerical align (aligns digits)
a	alpha align
s	spanned heading (not allowed on first column)
^	vertical spanning (not allowed on first row)
|	single vertical line
||	double vertical line

Headers and footers

.fo ‘left’center’right’	Footer for left center and right information
.he ‘left’center’right’	Header for left center and right information
%	Use the % for the current page number

Text:

.ce n	center next n lines of text
.sp n	Space n lines down
.PP	Begin paragraph
.ls n	line spacing (2 would be double spaced)
.bp	Start a new page
.ip one	indented paragraph with “one” as the “bullet word”, next line is paragraph
.lp	left aligned paragraph
.bu	next line is bulleted
.2c	begin two column output
.1c	begin one column output
.bc	begin in new column
.u text	underline text (multiple words in quotes)
.bx text	box text (multiple words in quotes)
.i text	italics text (multiple words in quotes)
.b text	bold text (multiple words in quotes)
.sz +N	change point size to N (temporary see docs for permanent)

Procedures and Functions Available:

The package has some pre-built tools to help build complex items on the page.

FUNCTION Bold()

Found In Bold.i, StdFmt.i

```
INPUT cText AS CHARACTER
```

Example Call:

```
Bold("Bolded Text")
```

FUNCTION CR()

Found In CR.i, StdFmt.i
Inserts some spacing into the report

```
INPUT iSpacing AS INTEGER
```

Example Call:

```
CR(2)
```

FUNCTION EnglishDay()

Found In EnglishDay.i, StdFmt.i
Returns the day name as day

```
INPUT iDay AS INTEGER
```

Example Call:

```
EnglishDay(WEEKDAY(TODAY)).
```

FUNCTION EnglishMonth()

Found In EnglishMonth.i, StdFmt.i
Returns the month name in English

```
INPUT dDate AS DATE
```

Example Call:

```
EnglishMonth(TODAY)
```

FUNCTION FontScale()

Found In FontScale.i, StdFmt.i
Sets font scale to n points temporarily

```
INPUT iFontScale AS INTEGER
```

Example Call:

```
FontScale(12)
```

FUNCTION InsertImage()

Found In InsertImage.i, StdFmt.i
Inserts an image into the document.

```
INPUT cAlign AS CHARACTER
INPUT cFileName AS CHARACTER
INPUT cXSize AS CHARACTER
INPUT cYSise AS CHARACTER
```

Align can be:

L – Left Align
R – Right Align
Default is centered

Sizes are in inches.

Example Call:

```
InsertImage ("", SEARCH("examples/epstest.eps"), "3", "3")
```

Creates a 3 inch by 3 inch picture from test.eps (encapsulated postscript).
The function will convert from GIF and JPG images if the convert command is available on the system (Linux).

FUNCTION Italics()

Found In Italics.i, StdFmt.i
Creates text in italics sub-font.

```
INPUT cText AS CHARACTER
```

Example Call:

```
Italics("Text in Italics")
```

FUNCTION LineSpace()

Found In LineSpace.i, StdFmt.i
Determines line spacing of the report. Two would be double spaced.

```
INPUT iSpacing AS INTEGER
```

Example Call:

```
LineSpace(2)
```

FUNCTION PageBreak()

Found In PageBreak.i, StdFmt.i
Causes a page break in the report

```
No Inputs
```

Example Call:

```
PageBreak()
```

FUNCTION PageFooter()

Found In PageFooter.i, StdFmt.i
Inserts a page footer onto page 2 through n. Use the % in place of the page number.

```
INPUT cLeft AS CHARACTER
INPUT cCenter AS CHARACTER
INPUT cRight AS CHARACTER
```

Example Call:

```
PageFooter("Left Side", "Page %", "Right Side")
```

FUNCTION PageHeader

Found In PageHeader.i, StdFmt.i
Inserts a page header onto page 2 through n. Use the % in place of the page number.

```
INPUT cLeft AS CHARACTER
INPUT cCenter AS CHARACTER
INPUT cRight AS CHARACTER
```

Example Call:

```
PageHeader("Left Side", "Page %", "Right Side")
```

FUNCTION ScaleData()

Found In ScaleData.i, StdFmt.i
Scales data for use with chart printing routines. Converts numbers into values between 0.0 – 1.0.

```
INPUT cDataList AS CHARACTER
```

Example Call:

```
RUN VertBarChart (
ScaleData(1,2,3,4),
"Jan,Feb,Mar,Apr",
4,
0.2).
```

FUNCTION TableEnd

Found In TableEnd.i, StdFmt.i
Defines the end of a table.

```
No Inputs
```

Example Call:

```
TableEnd()
```

FUNCTION TableHeader

Found In TableHeader.i, StdFmt.i
Sets up the header for table – effects multi-page tables.

```
INPUT cOptions AS CHARACTER
INPUT cFormat AS CHARACTER
INPUT cCommonData AS CHARACTER
```

Options:

center	center the table (default is left side)
expand	table becomes width of line for page
box	enclose the table in a box
allbox	all elements are in a box
doublebox	enclose table in double box
tab(x)	use x instead of tab to delimit data items.
linesize(n)	set box lines to n point size

Format:

l	left align
r	right align
c	center align
n	numerical align (aligns digits)
a	alpha align
s	spanned heading (not allowed on first column)
^	vertical spanning (not allowed on first row)
|	single vertical line
||	double vertical line

Example Call:

```
TableHeader("center tab(/)","c c","Month/Sales~n_/_")
```

FUNCTION TextCenter

Found In TextCenter.i, StdFmt.i

Determines number of lines of text to center (this refers to groff lines.)

```
INPUT iLines AS INTEGER
```

Example Call:

```
TextCenter(1)
```

FUNCTION TextUnderline

Found In TextUnderline.i, StdFmt.i

Underlines text presented as argument.

```
INPUT cText AS CHARACTER
```

Example Call:

```
TextUnderline("Text to be underlined")
```

PROCEDURE VertBarChart

Found in VertBarChart.i

```
  DEF INPUT PARAMETER cValuesList AS CHARACTER NO-UNDO.
  DEF INPUT PARAMETER cLabelList AS CHARACTER NO-UNDO.
  DEF INPUT PARAMETER fMaxHeight AS DECIMAL NO-UNDO.
  DEF INPUT PARAMETER fMaxWidth AS DECIMAL NO-UNDO.
```

Values are decimal from 0.0 thru 1.0 in a comma separated list.
Labels are in a comma separated list. Be careful they fit in the bar of the graph. They are required, so it you need blanks, enter that many commas to keep the one-to-one between values and labels.
The MaxHeight is in inches. It is the highest bar possible when the value is 1.0. The other bars will be ratios of this height.
The MaxWidth is used to specify the width of each bar.

Example Call:

```
RUN VertBarChart (
"0.1,0.3,0.7,0.4",
"0.1,0.3,0.7,0.4",
4,
0.2).
```

PROCEDURE HorzBarChart

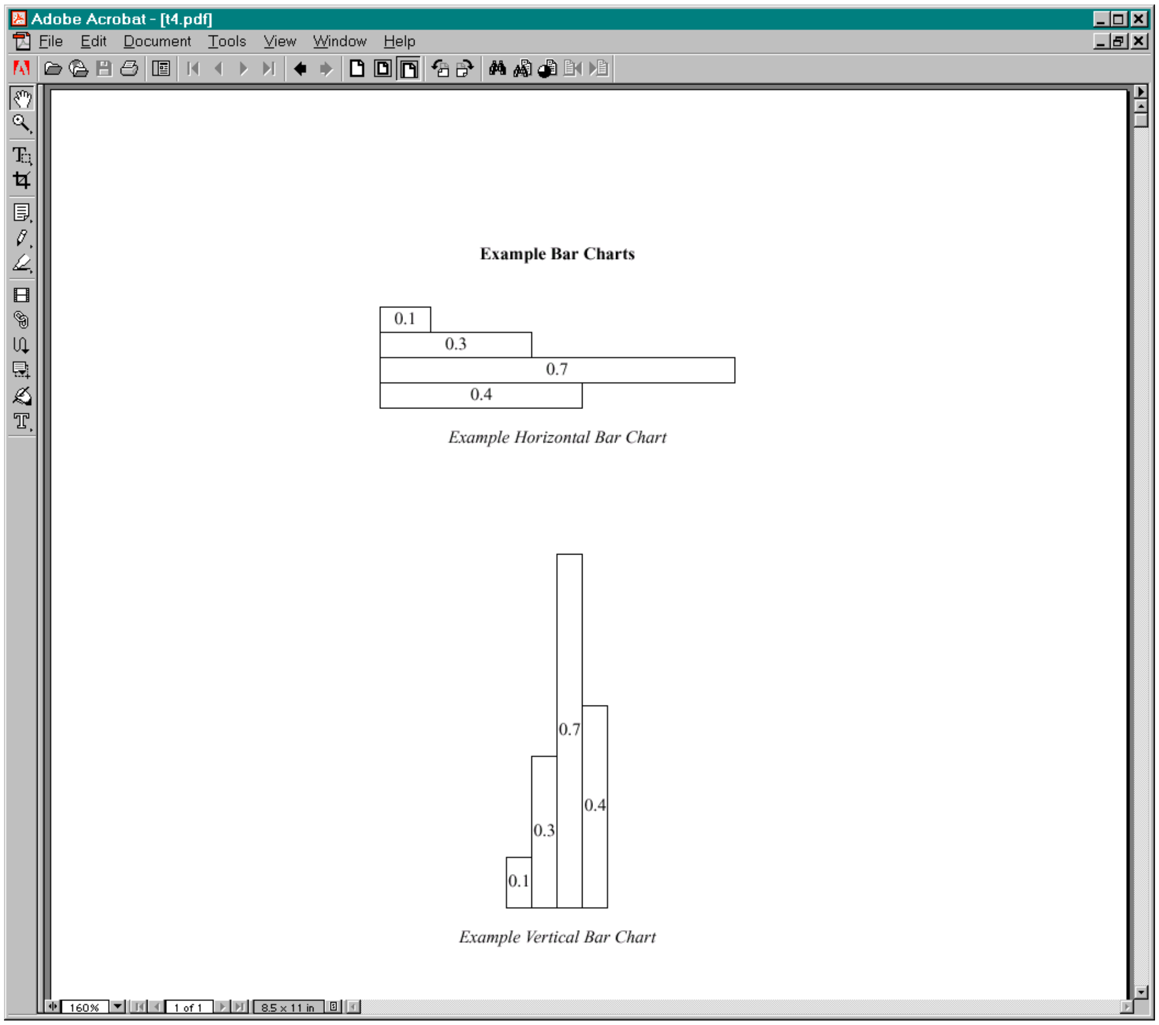
Found in HorzBarChart.i

```
  DEF INPUT PARAMETER cValuesList AS CHARACTER NO-UNDO.
  DEF INPUT PARAMETER cLabelList AS CHARACTER NO-UNDO.
  DEF INPUT PARAMETER fMaxWidth AS DECIMAL NO-UNDO.
  DEF INPUT PARAMETER fMaxHeight AS DECIMAL NO-UNDO.
```

Values are decimal from 0.0 thru 1.0 in a comma separated list.
Labels are in a comma separated list. Be careful they fit in the bar of the graph. They are required, so it you need blanks, enter that many commas to keep the one-to-one between values and labels.
The MaxWidth is in inches. It is the longest bar possible when the value is 1.0. The other bars will be ratios of this height.
The MaxHeight is used to specify the thickness of each bar.

```
RUN HorzBarChart (
"0.1,0.3,0.7,0.4",
"0.1,0.3,0.7,0.4",
4,
0.2).
```

Sample PDF:



Example Bar Charts

Example Horizontal Bar Chart

Example Vertical Bar Chart

Amduus Information Works, Inc.

We can help you implement these routines into your organization. We can provide training, programming, and installation help.

If you like, we can set up a machine and lease/sell Progress licenses to connect to your database and generate reports from this tool.

Amduus Information Works, Inc. provides strategic and integration programming for time and materials as well fixed priced packages. We specialize on the UNIX operating system.

Contact Scott Auge at sauge@amduus.com for more information.