Lab Goal - To learn about the Informix 4GL Terminal Capabilities.

Change to your Informix4GLclass directory and create a new directory for this lab:

```
cd ~/Informix4GLclass
mkdir lab08-termtest
cd lab08-termtest
```

Note: Most modern Unix and Linux systems use terminfo for terminal capacities. Some older versions of Informix also use terminfo. Check your Informix 4GL Release notes to see what your version may be using. The version Informix 4GL 7.51.FC3 that I am using requires TERM and TERMCAP be used. That makes it much easier for us to test changes.

Three Windows 4GL Program

1. First we will create a program to demonstrate windows in a 4GL program. This program will display 3 windows and a prompt for you to select the active window.
Type the following 4GL code into a file named threewindows.4gl

```
##########################################################################
## Module: @(#)threewindows.4gl     version: 1.0     Date: 10/1/2021
## Author: Lester Knutsen  Contact: lester@advancedatatools.com
## Copyright: Advanced DataTools Corporation - 2021
## Description: Informix 4GL for Developers Training Class Example Lab
##########################################################################

main

define ans char(1)

open window window1 at 5,10 with 10 rows, 30 columns
attribute ( border , prompt line last )
display "Window 1" at 1,2

open window window2 at 5,45 with 10 rows, 30 columns
attribute ( border , prompt line last )
display "Window 2" at 1,2

open window window3 at 18,25 with 10 rows, 30 columns
attribute ( border , prompt line last )
display "Window 3" at 1,2
```

```
while ( true )
    prompt "Window 1, 2, 3 or Q to Quit" for char ans
      case
      when ans = 1
            current window is window1
            exit case
      when ans = 2
            current window is window2
            exit case
      when ans = 3
            current window is window3
            exit case
      when ans = "q" or ans = "Q"
            exit while
      end case
end while

end main
```

2. Compile the 4GL code
   c4gl -o threewindows.4ge threewindows.4gl

3. Run the program
   threewindows.4ge

4. Compile the 4GL code using the RDS compiler
   fglpc threewindows.4gl

5. Run the program using the RDS compiler
   fglgo threewindows.4go

What line does the prompt appear at? Enter a number between 1-3 and change the current window. What key do you need to press to exit the program?

6. Edit the program and change the prompt to say "Change Current Window to 1, 2, 3 or Q to Quit". Compile the program.

Run the program, what is the runtime error you will get. Why?

Window size errors are a runtime error. Change the prompt to a shorter text string so it will fit and compile and test the program.

7. Does your windows show up as graphic boxes as show in figure 1 or with character boxes as in figure?

Figure 1

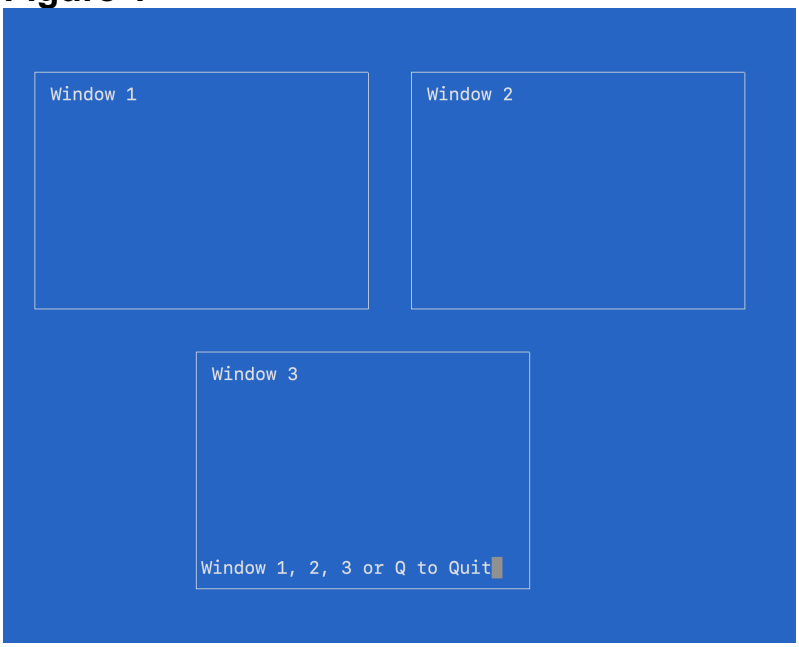
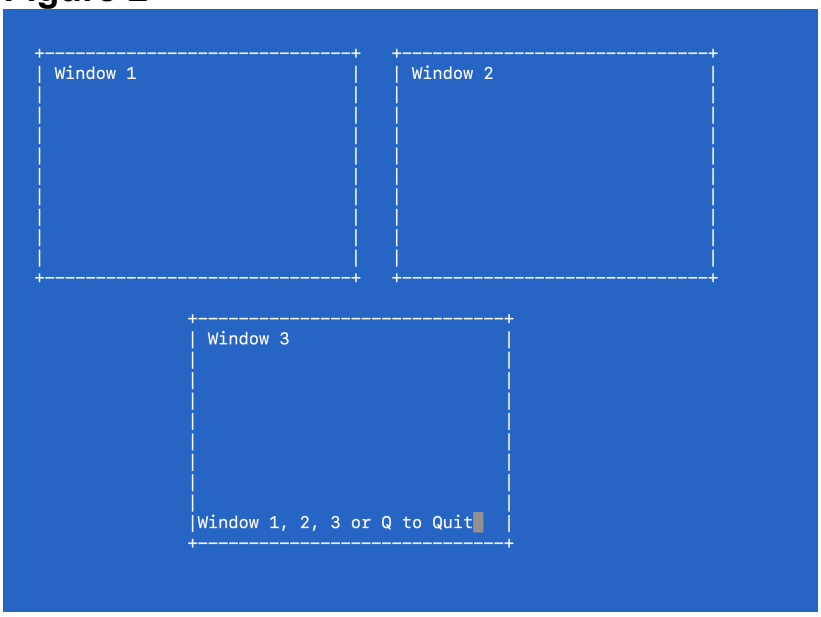


Figure 2



The Informix 4GL display is controlled by the TERM setting and the Unix TERMCAP file

Termtest 4GL Program

8. Termtest.4gl is a program I wrote in 1993 to test capabilities and the termcap entries for a new terminal. It creates a window with a border to test the line graphics capability. Then it displays 12 lines with different display attributes. Finally, using the prompt command you can test each of the function keys.

The program was originally developed to test 1993 type terminals, Wyse50, Procomm, X-term and the SCO ANSI Console. I have continued to use it over the years to test newer terminials. The bold, dim, and underline attributes may display somewhat differently from what you would expect. Using Procomm, an X term or SCO ansi console, bold displays as red and dim displays as blue. The underline attribute displays as blue under Procomm, as an underline under X term, and as white under SCO ansi. All the other attributes display as expected except the last attribute. The last line displays the attribute black which may be the same as the background color so it will be invisible.

The following four screen shots are some examples of how it may display when you run it depending on the terminal emulation you use and the termcap entries for your terminal.

The termcap entry has three basic parts for a 4GL program

1. The basic screen and cursor capabilities
2. The line drawing characters
3. The Function key available for your environment.

What you get in your environment may be different from what I get and if you have a working termcap and terminal system there is no need to change anything. Changes may affect items like cursor keys, backspace, terminal lines, and width so please only make changes in a test environment with test files. As an example, in working on termcap I have made changes an then not been able to use vi or and editor to change it back.

I am using a Apple Mac system with the default terminal program to do most of these examples. Using Putty or the console or another program

may have different results.

Figure 3 - Default xterm using /opt/informix/etc/termcap

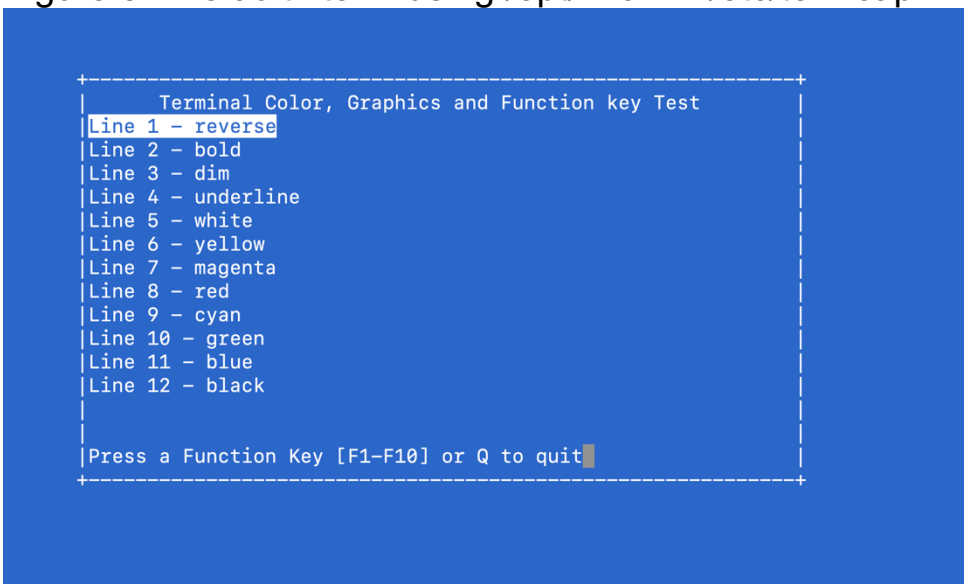


Figure 4 - Modified xterm using modified termcap for colors and lines

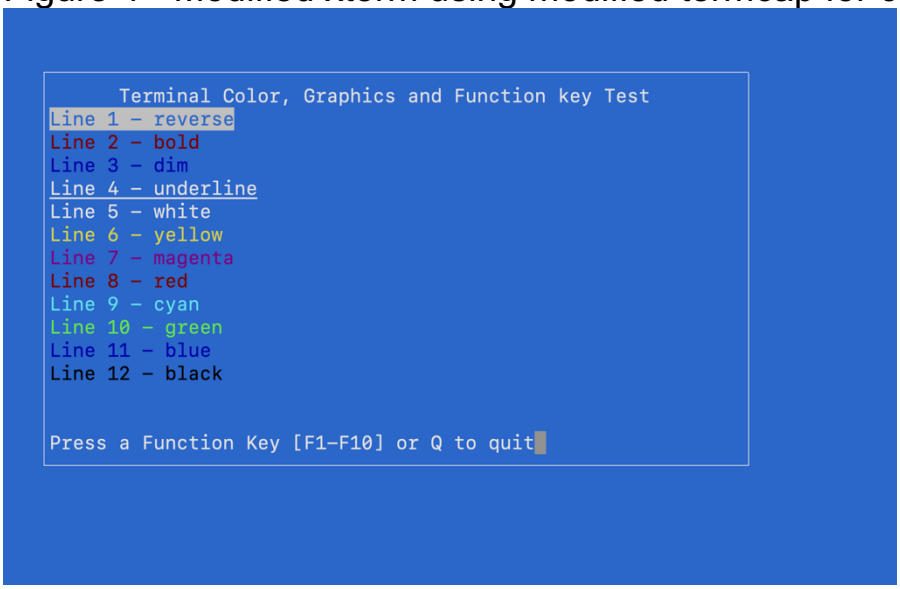


Figure 5 - Modified xterm using modified termcap

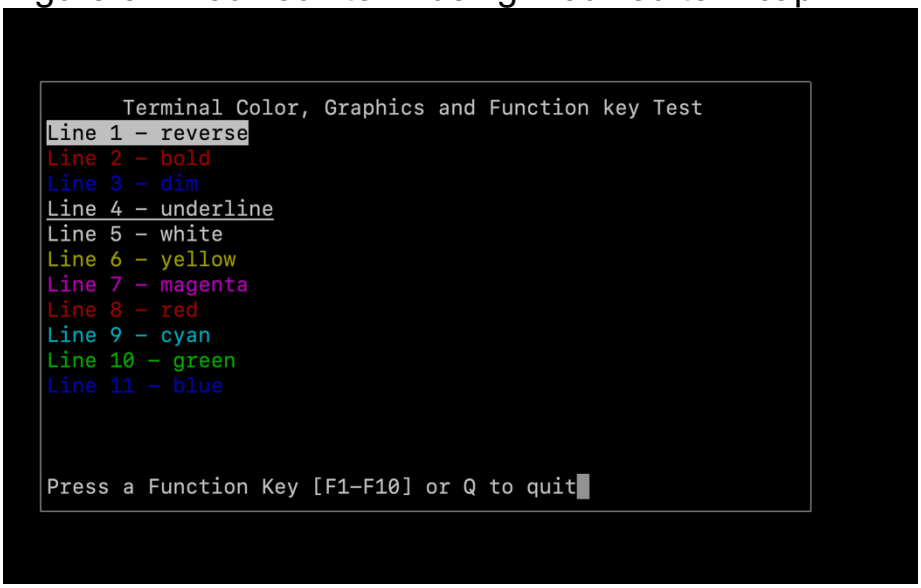
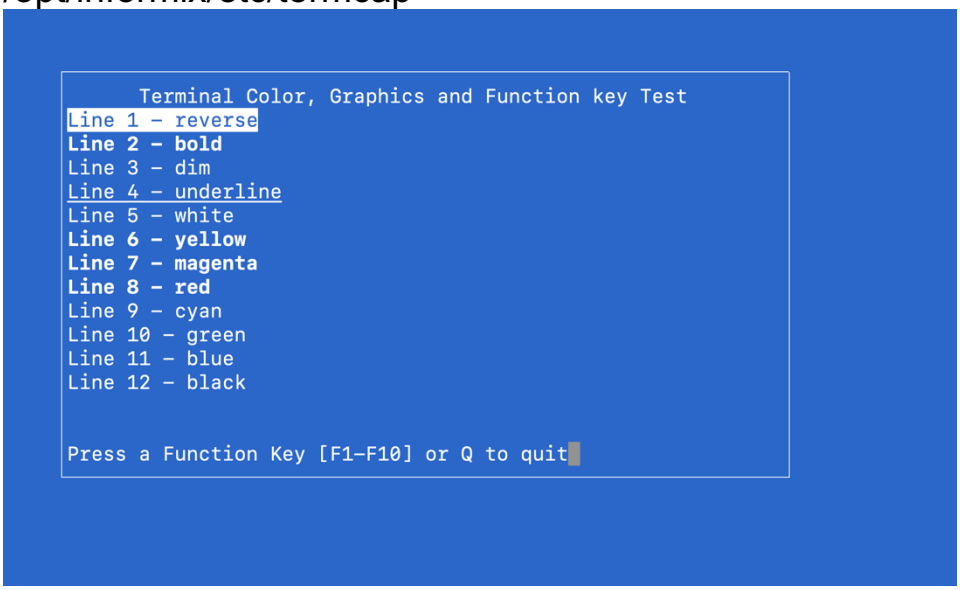


Figure 6 - Default xterm using vt100 TERM entries in /opt/informix/etc/termcap



9. Type the following 4GL code into a file named termtest.4gl

```
################################################################################
# Copyright 1993 Advanced DataTools Corporation
# Module: @(#)termtest.4gl 1.4 Date: 05/10/1993
# Author: Lester B. Knutsen
# Description: Program to test the Informix 4GL capabilities of a terminal
################################################################################
main
define ans char(1)

# Test if border line graphics are displayed correctly
open window test_box at 5,10 with 16 rows, 60 columns
attribute ( border , prompt line last )
display "Terminal Color, Graphics and Function key Test" at 1,7

# Test color and display attributes
display "Line 1 - reverse" at 2,1 attribute( reverse )
display "Line 2 - bold" at 3,1 attribute( bold )
display "Line 3 - dim" at 4,1 attribute( dim )
display "Line 4 - underline" at 5,1 attribute( underline )
display "Line 5 - white" at 6,1 attribute( white )
display "Line 6 - yellow" at 7,1 attribute( yellow )
display "Line 7 - magenta" at 8,1 attribute( magenta )
display "Line 8 - red" at 9,1 attribute( red )
display "Line 9 - cyan" at 10,1 attribute( cyan )
display "Line 10 - green" at 11,1 attribute( green )
display "Line 11 - blue" at 12,1 attribute( blue )
display "Line 12 - black" at 13,1 attribute( black ) # invsible line

# Test Function keys F1 to F10
while ( true )
    prompt "Press a Function Key [F1-F10] or Q to quit" for char ans
        on key (F1) error "F1 Function key pressed"
        on key (F2) error "F2 Function key pressed"
        on key (F3) error "F3 Function key pressed"
        on key (F4) error "F4 Function key pressed"
        on key (F5) error "F5 Function key pressed"
        on key (F6) error "F6 Function key pressed"
        on key (F7) error "F7 Function key pressed"
        on key (F8) error "F8 Function key pressed"
        on key (F9) error "F9 Function key pressed"
        on key (F10) error "F10 Function key pressed"
    end prompt
    if ans = "q" or ans = "Q" then exit while end if
end while
end main
```

10. Compile the 4GL code using the c4gl

    c4gl -o termtest.4ge termtest.4gl

11. Run the program

    termtest.4ge

12. Compile the 4GL code using the RDS compiler

    fglpc termtest.4gl

13. Run the program using the RDS compiler

    fglgo termtest.4go

14. Test

    1. Function Key F1 to F10, do they work?
    2. Did the box around the window draw a smooth line?
    3. Do the colors display correctly?
    4. Some terminals have limited or do not have color capabilities, function key and the ability to draw smooth lines. An xterm or programs like Putty or SecureCRT should be able to do this correctly.

15. Check your TERM and TERMCAP environment variables

    echo $TERM

    echo $TERMCAP

16. If everything works then the rest of this lab is informational only. You may find it helpful if you get a new terminal emulation program.

17. The follow is example termcap files to experiment with if you are using xterm or an SCO ansi terminal emulation. This termcap was used to get the screen shots in Figure 4 and Figure 5 above. Download the example form the class files as cut an paste may not work here.

This example only has three entries, xterm, xtermd and ansi. If you are using another terminal you may have to create your own entry. A good place to start is copy the informix termcap files in $INFORMIXDIR/etc/termcap to your class directory.

```
##############################################################################
## Module: @(#)termcap  version: 1.0     Date: 10/1/2021
## Author: Lester Knutsen  Contact: lester@advancedatatools.com
## Copyright: Advanced DataTools Corporation - 2021
## Description: Informix 4GL for Developers Training Class Example Lab
## ZA = COLOR Settings
## f0-f9 = Function Key Settings
## gs = Graphic Draw boxes using ansi character set
##############################################################################
## Modified xterm  - This is my current working one.
v1|xterm|vt100|DEC vt100:\
        :bs:am:xn:xv:pt:cd=\E[J:ce=2\E[K:cl=45\E[H\E[J:\
        :cs=%i\E[%d;%dr:cm=%i\E[%d;%dH:co#80:li#24:\
        :Hi=\E=:Hf=\E>:HI=^|:Po=\E1:Pe=\E2:\
        :bc=\E[D:do=\E[B:nd=\E[C:sr=\EM:up=\E[A:so=\E[7m:se=\E[m:\
        :ku=\E[A:kd=\E[B:kr=\E[C:kl=\E[D:kh=\E[H:us=\E[4m:ue=\E[0m:\
        :KA=\EOu:KU=\EOM:KB=\EOv:KC=\EOt:\
        :Ki=\EOr:KT=\EOl:KY=\EOS:KZ=\EOq:\
        :Kd=\EOm:k6=\EOs: \
        :sc=\E7:rc=\E8: \
      # function keys F1 (k0) to F10 (k9) \
        :k0=\EOP:k1=\EOQ:k2=\EOR:k3=\EOS:k4=\E[15~:\
        :k5=\E[17~:k6=\E[18~:k7=\E[19~:k8=\E[20~:k9=\E[21~:\
      # capability to display graphic boxes using the ansi character set \
        :gs=\E(0:ge=\E(B:gb=lmkjqxntuvw:\
      # informix 4gl color support  - the next line is one very long line \
      :ZA=\E[0%?%p1%{0}%=%t;37%e%p1%{1}%=%t;33%e%p1%{2}%=%t;35%e%p1%{3}%=%t;3
1%e%p1%{4}%=%t;36%e%p1%{5}%=%t;32%e%p1%{6}%=%t;34%e%p1%{7}%=%t;30%;%?%p2%t;7%
;%?%p3%t;5%;%?%p4%t;4%;m

## Dark Mode Screen
v2|xtermd|vt100|DEC vt100:\
        :bs:am:xn:xv:pt:cd=\E[J:ce=2\E[K:cl=45\E[H\E[J:\
        :cs=%i\E[%d;%dr:cm=%i\E[%d;%dH:co#80:li#24:\
        :Hi=\E=:Hf=\E>:HI=^|:Po=\E1:Pe=\E2:\
        :bc=\E[D:do=\E[B:nd=\E[C:sr=\EM:up=\E[A:so=\E[7m:se=\E[m:\
        :ku=\E[A:kd=\E[B:kr=\E[C:kl=\E[D:kh=\E[H:us=\E[4m:ue=\E[0m:\
        :KA=\EOu:KU=\EOM:KB=\EOv:KC=\EOt:\
        :Ki=\EOr:KT=\EOl:KY=\EOS:KZ=\EOq:\
        :Kd=\EOm:k6=\EOs: \
        :sc=\E7:rc=\E8: \
      # function keys F1 (k0) to F10 (k9) \
```

```
        :k0=\EOP:k1=\EOQ:k2=\EOR:k3=\EOS:k4=\E[15~:\
        :k5=\E[17~:k6=\E[18~:k7=\E[19~:k8=\E[20~:k9=\E[21~:\
      # capability to display graphic boxes using the ansi character set \
        :gs=\E(0:ge=\E(B:gb=lmkjqxntuvw:\
      # informix 4gl color support  - the next line is one very long line \
      :ZA=\E[%?%p1%{0}%=%t0m\E[37%e%p1%{1}%=%t33%e%p1%{2}%=%t35%e%p1%{3}%=%t3
1%e%p1%{4}%=%t36%e%p1%{5}%=%t32%e%p1%{6}%=%t34%e%p1%{7}%=%t30%;m\E[40m%?%p2%t
\E[7m%;%?%p3%t\E[5m%;%?%p4%t\E[4m%;:

## My original termcap from 1993 for SCO consoles and PCs with ANSI terminal
emulation
## Modified Informix 4GL ansi color, function keys, and box borders
## Does not work with current xterm
ansi|Ansi standard crt:\
      :al=\E[L:am:bs:cd=\E[J:ce=\E[K:cl=\E[2J\E[H:cm=\E[%i%d;%dH:co#80:\
      :dc=\E[P:dl=\E[M:dn=\E[B:ei=:ho=\E[H:ic=\E[@:im=:li#24:\
      :nd=\E[C:ms:pt:so=\E[7m:se=\E[m:us=\E[4m:ue=\E[m:up=\E[A:\
      :kb=^h:ku=\E[A:kd=\E[B:kl=\E[D:kr=\E[C:eo:\
      # capability to display graphic boxes using the PC ansi character set \
      :gs=:ge=:gb=\332\300\277\331\304\263:\
      # function keys F1 (k0) to F10 (k9) \
      :k0=\E[M:k1=\E[N:k2=\E[O:k3=\E[P:k4=\E[Q:\
      :k5=\E[R:k6=\E[S:k7=\E[T:k8=\E[U:k9=\E[V:\
      # informix 4gl color support  - the next line is one very long line \
:ZA=\E[%?%p1%{0}%=%t0m\E[37%e%p1%{1}%=%t33%e%p1%{2}%=%t35%e%p1%{3}%=%t31%e%p1
%{4}%=%t36%e%p1%{5}%=%t32%e%p1%{6}%=%t34%e%p1%{7}%=%t30%;m\E[40m%?%p2%t\E[7m%
;%?%p3%t\E[5m%;%?%p4%t\E[4m%;:
```

17. Download the example from the class files an put it in the directory lab08-termtest.

18. To Test this termcap and any termcap you modify you need to change the TERMCAP environment variable. Assuming the test termcap is in your local directory type:

    export TERMCAP=`pwd`/termcap.4glclass

    export TERM=xterm

19. Run the program

    termtest.4ge

20. To Test xterm in Dark Mode

    export TERM=xtermd

21. The Rest of this chapter is resource for developing and testing you own termcap file. Resources to help you with this are available on the web:

- ANSI Terminal Emulation and Informix 4GL Terminal Capabilities Published on May 1, 1993 by Lester Knutsen
    - https://advancedatatools.com/article/ansi-terminal-emulation-and-informix-4gl-terminal-capabilities-by-lester-knutsen/
- GNU Term Manual and Library
    - https://www.gnu.org/software/termutils/manual/termcap-1.3/html_chapter/termcap_toc.html
- IBM Docs on Defined Capabilities
    - https://www.ibm.com/docs/en/zos/2.4.0?topic=syntax-defined-capabilities
- Wikipedia Termcap
    - https://en.wikipedia.org/wiki/Termcap
- Using man termcap on your system may also provide helpful information.

22. You will need the manuals for your terminal or terminal emulation program to get the capabilities. Google and the internet may help you fine information.

23. The display graphic boxes using the ansi character set is set in termcap with the gs ge and gb settings. Here is the example I have found works best for xterm to get the smooth line boxes around a window. Check your manuals for your terminal type.

```
# capability to display graphic boxes using the ansi character set \
 :gs=\E(0:ge=\E(B:gb=lmkjqxntuvw:\
```

24. Color is set by the ZA setting. This must be one long continues line

informix 4gl color support - the next line is one very long line \

:ZA=\E[0%?%p1%{0}%=%t;37%e%p1%{1}%=%t;33%e%p1%{2}%=%t;35%e%p1%{3}%=%t;31%e%p1%{4}%=%t;36%e%p1%{5}%=%t;32%e%p1%{6}%=%t;34%e%p1%{7}%=%t;30%;%?%p2%t;7%;%?%p3%t;5%;%?%p4%t;4%;m

25. Function Keys are the easiest to discover. On your system if you have the system utility showkey, it will display the control codes that pressing the function keys send.

Type showkey -a and press a function key to see what is sent.

```
Press any keys - Ctrl-D will terminate this program

^[OP     27 0033 0x1b
         79 0117 0x4f
         80 0120 0x50
^[OQ     27 0033 0x1b
         79 0117 0x4f
         81 0121 0x51
```

In the example above I pressed Fn1 on my keyboard and it sent ^[OP and Fn1 sent ^[OQ. ^[is the Escape sequence and is entered in termcap as \E

You can also use the system utility for Octal Dumps od to show you keys pressed.

Type od -c and it will show you what is sent.

Function keys in termcap start with f0 for Fn1 f1 for Fn2. The entry below shows Fn1 as expecting Escape-OP or \E0P or ^[OP

:k0=\EOP:k1=\EOQ:k2=\EOR:k3=\EOS:k4=\E[15~:\