ANSI command-sequences

A look at some terminal emulation features utilized in the “console-redirection” mechanism

Clearing the screen

- Here is an ANSI command-sequence that clears the terminal's display-screen:

```
char  cmd[] = "\033[2J";

int     len = strlen( cmd );
write( 1, cmd, len );
```

Reposition the cursor

- Here is an ANSI command-sequence that moves the cursor to row 12, column 40:

```
char  cmd[] = “\033[12;40H”;

int     len = strlen( cmd );
write( 1, cmd, len );
```

ANSI color-codes

0 = black
1 = red
2 = green
3 = brown
4 = blue
5 = magenta
6 = cyan
7 = gray

Setting text attributes

- Here is an ANSI command-sequence that sets foreground and background colors:

```
char  cmd[] = “\033[32;44m”;

int     len = strlen( cmd );
write( 1, cmd, len );
```

Cursor visibility commands

- Here are ANSI command-sequences that will ‘hide’ or ‘show’ the terminal’s cursor:

```
char  hide[] = “\033[?25l”;      // lowercase L
char  show[] = “\033[?25h”;   // lowercase H
```

Demo-program

- We created a boot-time program (called ‘emitinfo.s’) that shows how a protected-mode exception-handler can send some diagnostic information via the serial null-modem cable to a remote machine that is running a ‘terminal emulator’ application

In-class exercise

- Modify this simple C++ program so that it will print its “Hello” message in colors and be located in the center of the screen:

```
#include <stdio.h>

int main( void )
{
        printf( “Hello, world! \n” );
}
```

In-class exercise #2 and #3

- Modify the ‘emitinfo.s’ program so that it will also display the FS and GS registers

- Modify the ‘emitinfo.s’ program so that its ‘isrGPF’ exception-handler will execute in 64-bit mode (on your ‘anchor’ machine)