COLORS MODULE

The module documentation

A module for color customization

Author

Rômulo Peres de Moraes

Table of Contents

1. The purposes of this document

This document has the purpose of describe the module and its interfaces with the goal of improve the Developer Experience (DX) once the developers have to create new features or even for maintenance.

2. Colors module

The colors module is a component from the Rclock that customize screen components like date, colons and digits with new colors. All the settings about colors are defined at the beginning of the program.

3. The layout of the documentation and the module

The module is divided into two parts, the public code and the private code, they will be placed inside the directories public/ and private/ respectively. All private interface names shall begin with an underscore. The public interfaces may use the private interfaces, the private interfaces may use another private interface, however, a private interface can’t use a public interface.

4. Interfaces

4.1. Public interfaces

void loadBuiltinColors( );

Purposes:

This procedure defines all possible colors that the Rclock can use for user customizations.

This procedure only define all colors using the Ncurses library, however, to use them is necessary call the **setComponentsColors( )** to set specific colors for each interface component.

Preconditions:

This routine must be the first called routine from this module for configuration purposes, if it is not called first, unexpected behavior may happen.

Postconditions:

All colors defined inside this routine may be used for draw elements on screen.

void setComponentsColors(struct ColorsModule userArguments, char* errorOutput);

Purposes:

This procedure will assign colors for all available interface components, that are Colons, digits and date.

Preconditions:

The user arguments must be fetched before calling this procedure**.**

Postconditions:

The component's color ID becomes available after calling this procedure and can be fetched using other function from this module.

Special considerations:

This procedure uses other private routines that set the colors for specific components.

The variables that store the colors of the components are static variables inside the module and defined at the beginning of the file.

Additional comments:

Any error generated inside this procedure shall be available on the **errorOutput** pointer.

ColorID getDigitColor(unsigned char digitIndex);

Purposes:

This function will fetch the ColorID from the array of digit's colors located inside the public part of the module as a static variable, the digit that needs to be fetched is identified by the digitIndex, with a range between 0 and 5.

Preconditions:

The loadBuiltinColors( ) and setComponentsColors( ) should be called before calling this function.

Postconditions:

The ColorID returned from this function may be used to set a custom color to the terminal using the Ncurses library.

Special considerations:

If a out of range value is provided as a digit index to the function, the returned value should be the default color of the clock (I.e the blue color).

ColorID getColonColor( );

Purposes:

This function returns the color ID assigned to the clock’s colons, the variable that holds this value is located inside the public part of the module.

Preconditions:

The loadBuiltinColors( ) and setComponentsColors( ) should be called before calling this function.

Postconditions:

The ColorID returned from this function may be used to set a custom color to the terminal using the Ncurses library.

ColorID getDateColor( );

Purposes:

This function returns the color ID assigned to the clock’s date, the variable that holds this value is located inside the public part of the module.

Preconditions:

The loadBuiltinColors( ) and setComponentsColors( ) should be called before calling this function.

Postconditions:

The ColorID returned from this function may be used to set a custom color to the terminal using the Ncurses library.

4.2. Private interfaces

void _setColorToTheDigits(ColorID newColor, ColorID digitColors[ ]);

Purposes:

This procedure will receive a color and the array of color IDs and will update all colors to the new value.

Postconditions:

All clock’s digits will be drawn using the new color.

Special considerations:

The array of colors is located inside the public part of the module.

void _setColorToTheClock(ColorID newColor, ColorID *colonsColor, ColorID digitColors[ ]);

Purposes:

This procedure will set a new color for the digits and the colons of the clock.

Postconditions:

After setting the new value, the colors of both components may fetched to be used for drawing the pixels on screen.

Special considerations:

This procedure uses other private routines to set the new color to the digits.

The variables that store the colors of the components are static variables inside the public part of the module and defined at the beginning of the file.

struct RclockColor* _searchForColor(char *colorName, struct RclockColor *availableColors, size_t sizeOfAvailableColors);

Purposes:

This function will search the color given by the user identified by name, the returned struct reference contains the color name, the color ID for the clock and the color ID for the date.

Postconditions:

The returned value gives you the oportunity of change the color of the clock’s digits/colons and the clock’s date with the respective Ids located inside the struct.

Special considerations:

If the color doesn’t exist, a NULL reference should be returned.

The struct RclockColor array is located inside the public part of the module.

Additional comments:

the sizeOfAvailableColors argument is the count of elements that the array of available colors have.

void _setGlobalDigitsColor(struct ColorsModule userArguments, char *errorOutput, struct RclockColor availableColors[ ], size_t sizeOfAvailableColors, ColorID digitColors[ ]);

Purposes:

This procedure checks if the color provided by the user exists and update the color of the digits to the new value.

Postconditions:

If no errors were reported, the digits will be shown with the new color.

Special considerations:

This routine uses _**searchForColor( )** and _**setColorToTheDigits( )** to accomplish its tasks.

The errors that this procedure might report are generated by the **generateErrorMessage( )** function, from the **errors** module.

Additional comments:

Any error reported by this procedure have to be copied to the *errorOutput.

void _setClockColor(struct ColorsModule userArguments, char *errorOutput, ColorID *colonsColor, ColorID digitsColor[ ], struct RclockColor availableColors[ ], size_t sizeOfAvailableColors);

Purposes:

This procedure checks if the color provided by the user exists and set a new color to the digits and colons of the clock.

Preconditions:

The user arguments have to be feched before calling this subroutine.

Postconditions:

If no errors were reported, the colons and digits of the clock will be displayed with the new color.

Special considerations:

This routine uses _searchForColor( ) to verify if a color is available in the software.

This routine uses _setColorToTheClock( ) to set the color to the digits and colons.

The digits’ colors and the colon’s color pointers are static variables located at the beginning of the public part of the module.

The errors that this procedure might report are generated by the generateErrorMessage( ) function, from the errors module.

Additional comments:

Any error reported by this procedure have to be copied to the *errorOutput.

void _setDateColor(struct ColorsModule userArguments, char* errorOutput, ColorID *dateColor, struct RclockColor availableColors[ ], size_t sizeOfAvailableColors);

Purposes:

This procedure checks if the color provided by the user exists and set a new color to the clock’s date.

Preconditions:

The user arguments have to be feched before calling this subroutine.

Postconditions:

If no errors were reported, the clock’s date will be displayed with the new color.

Special considerations:

This routine uses _searchForColor( ) to verify if a color is available in the software.

The date’s color pointer references a static variable that is located at the beginning of the module’s public file.

The errors that this procedure might report are generated by the generateErrorMessage( ) function, from the errors module.

Additional comments:

Any error reported by this procedure have to be copied to the *errorOutput.

void _setColonColor(struct ColorsModule userArguments, char* errorOutput, ColorID *colonsColor, struct RclockColor availableColors[ ], size_t sizeOfAvailableColors);

Purposes:

This procedure checks if the color provided by the user exists and set a new color to the clock’s colons.

Preconditions:

The user arguments have to be feched before calling this subroutine.

Postconditions:

If no errors were reported, the clock's date will be displayed with the new color.

Special considerations:

This routine uses _searchForColor( ) to verify if a color is available in the software.

The colon's color pointer references a static variable that is located at the beginning of the module's public file.

The errors that this procedure might report are generated by the generateErrorMessage( ) function, from the errors module.

Additional comments:

Any error reported by this procedure have to be copied to the *errorOutput.

void _setColorForEachClockDigit(struct ColorsModule userArguments, char* errorOutput, ColorID digitColors[ ], struct RclockColor availableColors[ ], size_t sizeOfAvailableColors);

Purposes:

This procedure set a new color for each digit of the clock that was specified by the user through command-line arguments.

Preconditions:

The user arguments have to be feched before calling this subroutine.

Postconditions:

If no errors were reported, every single clock's digit will be shown with its own color.

Special considerations:

This routine uses _searchForColor( ) to verify if a color is available in the software.

The digits' colors is array array located at the beginning of the module's public file.

The colors are inside the **struct ColorsModule** and they are collected by the arguments module.

The errors that this procedure might report are generated by the generateErrorMessage( ) function, from the errors module.

Additional comments:

Any error reported by this procedure have to be copied to the *errorOutput.