ARGUMENTS MODULE
The module documentation
A module for command-line inputs

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. Arguments module

The arguments module is a component from the Rclock that read and parse command-line arguments with the goal of enable, disable and customize the Rclock's features. This module uses an external library called **Anemone**, this library is useful for reading the command-line arguments and pass them to the control of the algorithm that will split and fill structs for each kind of feature.

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 interfaces, however, a private interface can't use a public interface.

4. Interfaces

4.1. Public interfaces

anemone_struct createProgramArguments(int argc, char *argv[ ]);

Purposes:

This procedure fully uses the features of the **Anemone** to define all available arguments that the Rclock will work on. It will initialize the library, create all optional arguments and compile everything. The returned value is the main structure of the Anemone library, all future operations use its data.

Preconditions:

This function have to be called first, a great part of the Rclock's features work on the values that this module will return.

Postconditions:

After calling this function, all the arguments will be available in a raw way inside the **anemone_struct**.

Special considerations:

This function only works correctly if the Anemone library be linked with the main program.

ProgramArguments fetchProgramArguments(anemone_struct *anemone, char *errorOutput);

Purposes:

This function will read all the collected arguments from the **anemone_struct** and split them into different structs that will be useful for each kind of feature. The returned value is a collection of these structs.

Preconditions:

The library have to be initialized before calling this function using the **createProgramArguments( )** function.

Postconditions:

All arguments given by the user will be returned and available for using inside the **ProgramArguments** variable.

Special considerations:

This function only works correctly if the Anemone library be linked with the main program.

Any error reported inside this function will be available on ***errorOutput**.

4.2. Private interfaces

ErrorID _validateHoursMinutesSecondsDaysMonthsAndYears(struct DatetimeModule *argumentsOutput, anemone_struct *anemone);

Purposes:

This function will check for errors on the values that represent hours, minutes, seconds, days, months and years, returning the arguments inside the **DatetimeModule**.

Preconditions:

The **createProgramArguments( )** have to be called before calling this function.

Postconditions:

If no errors were reported, all these values (if given by the user) will be moved to the **DatetimeModule** variable.

Special considerations:

This function only works correctly if the Anemone library be linked with the main program.

If any error happened inside this function, the returned value is the ID of the error.

void _fetchColorsArguments(anemone_struct *anemone, char *errorOutput, ProgramArguments *arguments);

Purposes:

This procedure will fetch all command-line arguments that are part of the color’s features.

Preconditions:

The **createProgramArguments( )** have to be called before calling this function.

Postconditions:

All those values that enable/disable color features will be moved and available inside the **ProgramArguments**.

Special considerations:

This function only works correctly if the Anemone library be linked with the main program.

ErrorID _fetchDatetimeArguments(anemone_struct *anemone, char *errorOutput, ProgramArguments *arguments);

Purposes:

This function will fetch all command-line arguments that are part of the datetime’s features.

Preconditions:

The **createProgramArguments( )** have to be called before calling this function.

Postconditions:

All those values that manipulate date and time will be moved and available inside the **ProgramArguments**.

Special considerations:

This function only works correctly if the Anemone library be linked with the main program.

This function uses the **_validateHoursMinutesSecondsDaysMonthsAndYears( )** to fetch the hours, minutes, seconds, days, months and years individually.

void _fetchScreenManagementArguments(anemone_struct *anemone, char *errorOutput, ProgramArguments *arguments);

Purposes:

This function will fetch all command-line arguments that manipulate components on screen.

Preconditions:

The **createProgramArguments( )** have to be called before calling this function.

Postconditions:

All those values that manipulate the screen components like hide/show the date and the seconds will be moved and available inside the **ProgramArguments**.

Special considerations:

This function only works correctly if the Anemone library be linked with the main program.