UNIX-CLI

Gerado por Doxygen 1.10.0

Capítulo 1

Índice dos ficheiros

1.1 Lista de ficheiros

Lista de todos os ficheiros documentados com uma breve descrição:

Capítulo 2

Documentação do ficheiro

2.1 Referência ao ficheiro constants.h

Header file containing constant definitions.

Macros

- #define **PROGRAM_NAME** "UNIX-CLI"
- #define **VERSION** "0.1"
- #define **EXIT_CMD** "termina"
- #define **MAX_ARGS** 64
- #define **BUFFER_SIZE_BYTES** 4096
- #define **FILE_INFO_STR_SIZE** 50

2.1.1 Descrição detalhada

Header file containing constant definitions.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

This header file defines various constants that are used throughout the application. These constants are used to represent specific values or settings that are used in multiple parts of the codebase.

Constants in this file are organized into logical groups based on their purpose or usage. Each constant is given a descriptive name to indicate its meaning or significance.

For example, constants related to file permissions may be grouped together, while constants representing error codes may be in a separate group.

Constants defined in this file are intended to improve code readability, reduce the risk of errors due to magic numbers, and provide a centralized location for managing shared values.

Versão

0.1

Data

2024-03-20

Copyright

2.2 constants.h

Ir para a documentação deste ficheiro.

```
00001
00026 #ifndef CONSTANTS_H
00027 #define CONSTANTS_H
00028
00029 /* GENERAL CONSTANTS */
00030 #define PROGRAM_NAME "UNIX-CLI"  // program name
00031 #define VERSION "0.1"            // current version number
00032 #define EXIT_CMD "termina"       // command to exit CLI
00033 #define MAX_ARGS 64              // maximum number of arguments
00034
00035 /* BUFFERS */
00036 #define BUFFER_SIZE_BYTES 4096  // max buffer size
00037
00038 /* FILE INFORMATION */
00039 #define FILE_INFO_STR_SIZE 50  // size of strings in `FileInfo` structure
00040
00041 #endif /* CONSTANTS_H */
```

2.3 Referência ao ficheiro execute.h

Header file for functions that execute files.

Funções

- void **execute_command** (const char *command_path, char *args[])

  Executes a command with the given arguments.

2.3.1 Descrição detalhada

Header file for functions that execute files.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

Versão

0.1

Data

2024-04-21

Copyright

Copyright (c) 2024

2.3.2 Documentação das funções

2.3.2.1 execute_command()

```
void execute_command (
            const char * command_path,
            char * args[] )
```

Executes a command with the given arguments.

This function creates a child process using fork() and executes the specified command with the provided arguments using execvp(). If the fork or execvp operation fails, an error message is printed, and the function returns.

Parâmetros

command_path	The path to the command to be executed.
args	An array of strings containing the arguments for the command.

2.4 execute.h

Ir para a documentação deste ficheiro.

```
00001
00012 #ifndef EXECUTE_H
00013 #define EXECUTE_H
00014
00025 void execute_command(const char *command_path, char *args[]);
00026
00027 #endif /* EXECUTE_H */
```

2.5 Referência ao ficheiro find.h

Header file for functions that find and check for executable files.

```
#include <stdbool.h>
```

Funções

- bool **is_executable_file** (const char ∗path)

  Checks if a file is executable.
- bool **find_command_in_path** (const char ∗command, char ∗command_path)

  Finds the full path of a command in the PATH environment variable.

2.5.1 Descrição detalhada

Header file for functions that find and check for executable files.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

Versão

0.1

Data

2024-04-21

Copyright

2.5.2 Documentação das funções

2.5.2.1 find_command_in_path()

```
bool find_command_in_path (
            const char * command,
            char * command_path )
```

Finds the full path of a command in the PATH environment variable.

This function searches for the specified command in the directories listed in the PATH environment variable. If the command is found, its full path is copied to the provided buffer.

Parâmetros

command	The name of the command to search for.
command_path	A buffer to store the full path of the command.

Retorna

true if the command is found, false otherwise.

2.5.2.2 is_executable_file()

```
bool is_executable_file (
            const char * path )
```

Checks if a file is executable.

This function checks if the file at the specified path is executable.

Parâmetros

path	The path to the file.

Retorna

true if the file is executable, false otherwise.

2.6 find.h

Ir para a documentação deste ficheiro.

```
00001
00012 #ifndef FIND_H
00013 #define FIND_H
00014
00015 #include <stdbool.h>
00016
00025 bool is_executable_file(const char *path);
00026
00038 bool find_command_in_path(const char *command, char *command_path);
00039
00040 #endif /* FIND_H */
```

2.7 Referência ao ficheiro input_parser.h

Header file for input parsing functions.

Funções

- int **parse_input** (char ∗input, char ∗args[ ], int max_args)
  Parses an input string into arguments.

2.7.1 Descrição detalhada

Header file for input parsing functions.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

Versão

0.1

Data

2024-04-21

Copyright

Copyright (c) 2024

2.7.2 Documentação das funções

2.7.2.1 parse_input()

```
int parse_input (
            char * input,
            char * args[],
            int max_args )
```

Parses an input string into arguments.

This function tokenizes the input string by spaces and stores the tokens in the provided array of strings (args). The maximum number of arguments that can be stored in the args array is specified by max_args.

Parâmetros

input	The input string to be parsed.
args	An array of strings to store the parsed arguments.
max_args	The maximum number of arguments that can be stored.

Retorna

int The number of arguments parsed and stored in the `args` array.

2.8 input_parser.h

Ir para a documentação deste ficheiro.

```
00001
00012 #ifndef INPUT_PARSER_H
00013 #define INPUT_PARSER_H
00014
00027 int parse_input(char *input, char *args[], int max_args);
00028
00029 #endif /* INPUT_PARSER_H */
```

2.9 Referência ao ficheiro utils.h

Contains utility declarations.

`#include <stdbool.h>`

Funções

- bool **should_exit** (const char *input)

  Checks if the input string indicates the program should exit.

2.9.1 Descrição detalhada

Contains utility declarations.

Autor

Enrique Rodrigues (`a28602@alunos.ipca.pt`)

Versão

0.1

Data

2024-04-21

Copyright

Copyright (c) 2024

2.9.2 Documentação das funções

2.9.2.1 should_exit()

```
bool should_exit (
            const char * input )
```

Checks if the input string indicates the program should exit.

This function checks if the input string starts with the exit command defined in the constants header file. If the input string matches the exit command, the function returns true; otherwise, it returns false.

Parâmetros

input	The input string to check.

Retorna

true if the input string indicates program exit, false otherwise.

2.10 utils.h

Ir para a documentação deste ficheiro.

```
00001
00011 #ifndef UTILS_H
00012 #define UTILS_H
00013
00014 #include <stdbool.h>
00015
00026 bool should_exit(const char *input);
00027
00028 #endif /* UTILS_H */
```

2.11 Referência ao ficheiro execute.c

Contains functions for executing external commands.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
```

Funções

- void **execute_command** (const char *command_path, char *args[])
  Executes a command with the given arguments.

2.11.1 Descrição detalhada

Contains functions for executing external commands.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

Versão

0.1

Data

2024-04-21

Copyright

Copyright (c) 2024

2.11.2 Documentação das funções

2.11.2.1 execute_command()

```
void execute_command (
            const char * command_path,
            char * args[] )
```

Executes a command with the given arguments.

This function creates a child process using fork() and executes the specified command with the provided arguments using execvp(). If the fork or execvp operation fails, an error message is printed, and the function returns.

Parâmetros

command_path	The path to the command to be executed.
args	An array of strings containing the arguments for the command.

2.12 Referência ao ficheiro find.c

Contains functions for finding executable files in the PATH.

```
#include <constants.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
```

Macros

- #define **_XOPEN_SOURCE** 700
- #define **_DEFAULT_SOURCE**

Funções

- bool **is_executable_file** (const char ∗path)

  Checks if a file is executable.
- bool **find_command_in_path** (const char ∗command, char ∗command_path)

  Finds the full path of a command in the PATH environment variable.

2.12.1 Descrição detalhada

Contains functions for finding executable files in the PATH.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

Versão

0.1

Data

2024-04-21

Copyright

Copyright (c) 2024

2.12.2 Documentação das funções

2.12.2.1 find_command_in_path()

```
bool find_command_in_path (
            const char * command,
            char * command_path )
```

Finds the full path of a command in the PATH environment variable.

This function searches for the specified command in the directories listed in the PATH environment variable. If the command is found, its full path is copied to the provided buffer.

Parâmetros

command	The name of the command to search for.
command_path	A buffer to store the full path of the command.

Retorna

true if the command is found, false otherwise.

2.12.2.2 is_executable_file()

```
bool is_executable_file (
            const char * path )
```

Checks if a file is executable.

This function checks if the file at the specified path is executable.

Parâmetros

path	The path to the file.

Retorna

true if the file is executable, false otherwise.

2.13 Referência ao ficheiro input_parser.c

Contains functions for parsing input strings into arguments.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Funções

- int **parse_input** (char ∗input, char ∗args[ ], int max_args)
  Parses an input string into arguments.

2.13.1 Descrição detalhada

Contains functions for parsing input strings into arguments.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

Versão

0.1

Data

2024-04-21

Copyright

Copyright (c) 2024

2.13.2 Documentação das funções

2.13.2.1 parse_input()

```
int parse_input (
            char * input,
            char * args[],
            int max_args )
```

Parses an input string into arguments.

This function tokenizes the input string by spaces and stores the tokens in the provided array of strings (args). The maximum number of arguments that can be stored in the args array is specified by max_args.

Parâmetros

input	The input string to be parsed.
args	An array of strings to store the parsed arguments.
max_args	The maximum number of arguments that can be stored.

Retorna

int The number of arguments parsed and stored in the `args` array.

2.14 Referência ao ficheiro main.c

This file contains the main entry point of the program.

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include "constants.h"
#include "execute.h"
#include "find.h"
#include "input_parser.h"
#include "utils.h"
```

Macros

- #define **_XOPEN_SOURCE** 700

Funções

- int **main** ()
  Main entry point of the program.

2.14.1 Descrição detalhada

This file contains the main entry point of the program.

Autor

Enrique Rodrigues (`a28602@alunos.ipca.pt`)

Unix-CLI is a versatile command-line utility featuring a custom command interpreter, allowing users to execute a variety of commands directly from the terminal. With a focus on efficiency and user-friendliness, Unix-CLI utilizes system calls for low-level operations, ensuring broad compatibility across Unix-like operating systems.

Versão

0.2

Data

2024-03-20

2.14.2 Modifications

- 2024-04-18: Updated program to v0.2, documented on the Github repo. Enrique George Rodrigues (a28602@alunos.ipca.pt)
- 2024-04-22: CLI tries to execute as a file first and if it fails it looks in the users PATH variable. Enrique George Rodrigues (a28602@alunos.ipca.pt)

2.14.3 Documentação das funções

2.14.3.1 main()

```
int main ( )
```

Main entry point of the program.

The main function serves as the entry point of the program. It executes the command-line interface, allowing users to execute commands and programs.

Retorna

int Returns 0 upon successful execution or 1 in the case of error.

2.15 Referência ao ficheiro utils.c

Contains utility functions.

```
#include <stdbool.h>
#include <string.h>
#include <unistd.h>
#include "constants.h"
```

Funções

- bool **should_exit** (const char ∗input)
  Checks if the input string indicates the program should exit.

2.15.1 Descrição detalhada

Contains utility functions.

Autor

Enrique Rodrigues (a28602@alunos.ipca.pt)

Versão

0.1

Data

2024-04-21

Copyright

2.15.2 Documentação das funções

2.15.2.1 should_exit()

```
bool should_exit (
            const char * input )
```

Checks if the input string indicates the program should exit.

This function checks if the input string starts with the exit command defined in the constants header file. If the input string matches the exit command, the function returns true; otherwise, it returns false.

Parâmetros

input	The input string to check.

Retorna

true if the input string indicates program exit, false otherwise.

Índice