Daemon

1.0.0

Generated by Doxygen 1.9.6

Chapter 1

Strona główna dokumentacji Deamona

1.1 Opis

Demon którego głównym zadaniem jest utrzymywanie tej samej zawartości pomiędzy dwoma katalogami.
Po podaniu katalogu źródłowego i docelowego demon rozpoczyna swoją pracę.
Dodatkowo jest opcja aby program działał rekurencyjnie.

1.2 Inicjalizacja

1.2.1 Parametry

Parametr 1: Folder źródłowy
Parametr 2: Folder docelowy
Parametr 3: Wielkość która będzie dzeliła pliki na "małe" i "duże"
Parametr 4(opcjonalny): -R, służy do przeszukiwania rekurencyjnego

1.2.2 Działanie

Program działa w tle jako demon. Co 5 minut sprawdza stan folderów.
Jeżeli został znaleziony plik który znajduje się w katalogu źródłowym, a nie ma go w docelowym, zostaje do niego skopiowany. Jeżeli plik istnieje zarówno w folderze źródłowym i docelowym zostaje sprawdzona data modyfikacji, od której zależy dalsza praca, jeżeli plik ma nowszą datę w folderze źródłowym zostaje skopiowany do folderu docelowego, w innym wypadku nic się nie dzieje Jeżeli w folderze docelowym zostaje znaleziony plik któego nie ma w folderze źródłowym zostaje on usunięty.
Jeżeli została wybrana opcja -R zostają sprawdzane również katalogi, i ich zawartości, z folderów źródłowego i docelowego.

1.3 Autorzy

Mateusz Kondraciuk
Jakub Franciszek Modzelewski

1.4 Prowadzący

dr inż. Wojciech Kwedlo

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 doesItExists Struct Reference

Struktura przechowywująca wszystkie pliki które znajdują się w folderze źródłowym z uwzględnieniem podziału na foldery oraz pliki.

Data Fields

- char name [32]

  Przechowuje informacje o nazwie pliku lub katalogu.
- int isDir

  Przechowuje informacje o tym czy element jest katalogiem.
- struct doesItExists * next

  Wzkaźnik na następny element.

4.1.1 Detailed Description

Struktura przechowywująca wszystkie pliki które znajdują się w folderze źródłowym z uwzględnieniem podziału na foldery oraz pliki.

4.1.2 Field Documentation

4.1.2.1 isDir

```
int isDir
```

Przechowuje informacje o tym czy element jest katalogiem.

4.1.2.2 name

```
char name[32]
```

Przechowuje informacje o nazwie pliku lub katalogu.

4.1.2.3 next

```
struct doesItExists* next
```

Wzkaźnik na następny element.

The documentation for this struct was generated from the following file:

- C:/Users/user/Desktop/daemon/deleteNotExisting.c

Chapter 5

File Documentation

5.1 C:/Users/user/Desktop/daemon/copy.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include <sys/mman.h>
```

Macros

- #define PACKAGE "mmap"

Functions

- int copyBigFile (char *fileSourcePath, char *fileDestPath)

  Funkcja służąca do kopiowana dużych plików, to czy plik jest wystarczająco duży do używania tej funckcji ustala użytkownik.
- int copySmallFile (char *fileSourcePath, char *fileDestPath)

  Funkcja służąca do kopiowana małych plików, to czy plik jest mały ustala użytkownik.
- int copyOrNot (struct stat sourceFile, struct stat destFile)

  Funkcja służąca do sprawdzania potrzeby kopiowania pliku, tzn. czy plik z folderu docelowego nie jest nowszy niż plik z folderu źródłowego.
- int copyFile (char *fileSourcePath, char *fileDestPath, char *size, struct stat inputFileAttrib)

  Funkcja służąca do sprawdzenia rozmiaru pliku, i przekierowania do prawidłowej funkcji.

5.1.1 Macro Definition Documentation

5.1.1.1 PACKAGE

```
#define PACKAGE "mmap"
```

5.1.2 Function Documentation

5.1.2.1 copyBigFile()

```
int copyBigFile (
            char * fileSourcePath,
            char * fileDestPath )
```

Funkcja służąca do kopiowana dużych plików, to czy plik jest wystarczająco duży do używania tej funckcji ustala użytkownik.

Parameters

in	*fileSourcePath*	ścieżka do pliku źródłowego
in	*fileDestPath*	ścieżka do pliku docelowego

Return values

0	jest zwracane w przypadku sukcesu

5.1.2.2 copyFile()

```
int copyFile (
            char * fileSourcePath,
            char * fileDestPath,
            char * size,
            struct stat inputFileAttrib )
```

Funkcja służąca do sprawdzenia rozmiaru pliku, i przekierowania do prawidłowej funkcji.

Parameters

in	*fileSourcePath*	ścieżka do pliku źródłowego
in	*fileDestPath*	ścieżka do pliku docelowego
in	*size*	wielkość określona przez użytkownika dzieląca pliki na małe i duże
in	*inputFileAttrib*	struktura przechowująca metadane pliku

Return values

copySmallFile	jest zwracane w przypadku gdy plik zostaje oznaczony jako mały
copyBigFile	jest zwracane w przypadku gdy plik zostaje oznaczony jako duży

5.1.2.3 copyOrNot()

```
int copyOrNot (
            struct stat sourceFile,
            struct stat destFile )
```

Funkcja służąca do sprawdzania potrzeby kopiowania pliku, tzn. czy plik z folderu docelowego nie jest nowszy niż plik z folderu źródłowego.

Parameters

in	*sourceFile*	struktura przechowująca metadane pliku źródłowego
in	*destFile*	struktura przechowująca metadane pliku docelowego

Return values

0	jest zwracane w przypadku sukcesu

5.1.2.4 copySmallFile()

```
int copySmallFile (
            char * fileSourcePath,
            char * fileDestPath )
```

Funkcja służąca do kopiowana małych plików, to czy plik jest mały ustala użytkownik.

Parameters

in	*fileSourcePath*	ścieżka do pliku źródłowego
in	*fileDestPath*	ścieżka do pliku docelowego

Return values

0	jest zwracane w przypadku sukcesu
-1	jest zwracane w przypadku błędu

5.2 C:/Users/user/Desktop/daemon/copy.h File Reference

Functions

- int copyOrNot (struct stat sourceFile, struct stat destFile)

  Funkcja służąca do sprawdzania potrzeby kopiowania pliku, tzn. czy plik z folderu docelowego nie jest nowszy niż plik z folderu źródłowego.
- int copyFile (char ∗fileSourcePath, char ∗fileDestPath, char ∗size, struct stat inputFileAttrib)

  Funkcja służąca do sprawdzenia rozmiaru pliku, i przekierowania do prawidłowej funkcji.
- int copyBigFile (char ∗fileSourcePath, char ∗fileDestPath)

  Funkcja służąca do kopiowana dużych plików, to czy plik jest wystarczająco duży do używania tej funckcji ustala użytkownik.
- int copySmallFile (char ∗fileSourcePath, char ∗fileDestPath)

  Funkcja służąca do kopiowana małych plików, to czy plik jest mały ustala użytkownik.

5.2.1 Function Documentation

5.2.1.1 copyBigFile()

```
int copyBigFile (
            char * fileSourcePath,
            char * fileDestPath )
```

Funkcja służąca do kopiowana dużych plików, to czy plik jest wystarczająco duży do używania tej funckcji ustala użytkownik.

Parameters

in	*fileSourcePath*	ścieżka do pliku źródłowego
in	*fileDestPath*	ścieżka do pliku docelowego

Return values

0	jest zwracane w przypadku sukcesu

5.2.1.2 copyFile()

```
int copyFile (
            char * fileSourcePath,
            char * fileDestPath,
            char * size,
            struct stat inputFileAttrib )
```

Funkcja służąca do sprawdzenia rozmiaru pliku, i przekierowania do prawidłowej funkcji.

Parameters

in	*fileSourcePath*	ścieżka do pliku źródłowego
in	*fileDestPath*	ścieżka do pliku docelowego
in	*size*	wielkość określona przez użytkownika dzieląca pliki na małe i duże
in	*inputFileAttrib*	struktura przechowująca metadane pliku

Return values

copySmallFile	jest zwracane w przypadku gdy plik zostaje oznaczony jako mały
copyBigFile	jest zwracane w przypadku gdy plik zostaje oznaczony jako duży

5.2.1.3 copyOrNot()

```
int copyOrNot (
            struct stat sourceFile,
            struct stat destFile )
```

Funkcja służąca do sprawdzania potrzeby kopiowania pliku, tzn. czy plik z folderu docelowego nie jest nowszy niż plik z folderu źródłowego.

Parameters

in	*sourceFile*	struktura przechowująca metadane pliku źródłowego
in	*destFile*	struktura przechowująca metadane pliku docelowego

Return values

0	jest zwracane w przypadku sukcesu

5.2.1.4 copySmallFile()

```
int copySmallFile (
            char * fileSourcePath,
            char * fileDestPath )
```

Funkcja służąca do kopiowana małych plików, to czy plik jest mały ustala użytkownik.

Parameters

in	*fileSourcePath*	ścieżka do pliku źródłowego
in	*fileDestPath*	ścieżka do pliku docelowego

Return values

0	jest zwracane w przypadku sukcesu
-1	jest zwracane w przypadku błędu

5.3 copy.h

Go to the documentation of this file.

```
00001 extern int copyOrNot(struct stat sourceFile, struct stat destFile);
00002 extern int copyFile(char *fileSourcePath, char *fileDestPath, char *size, struct stat
      inputFileAttrib);
00003 extern int copyBigFile(char *fileSourcePath, char *fileDestPath);
00004 extern int copySmallFile(char *fileSourcePath, char *fileDestPath);
```

5.4 C:/Users/user/Desktop/daemon/copyRecursive.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include <sys/mman.h>
#include "copy.h"
#include "currentTime.h"
#include "updateTextFile.h"
```

Functions

- int copyRecursiveDir (char ∗dirSource, char ∗dirDest, char ∗argv, int iter)

  Funkcja służąca do kopiowana rekursywnego wchodzenia w poszczególne katalogi, kopiowania ich oraz ich zawartości do folderu docelowego.

5.4.1 Function Documentation

5.4.1.1 copyRecursiveDir()

```
int copyRecursiveDir (
            char * dirSource,
            char * dirDest,
            char * argv,
            int iter )
```

Funkcja służąca do kopiowana rekursywnego wchodzenia w poszczególne katalogi, kopiowania ich oraz ich zawartości do folderu docelowego.

Parameters

in	*dirSource*	ścieżka źródłowa
in	*dirDest*	ścieżka docelowa
in	*argv*	parametry wejściowe
in	*iter*	służy do robienia wcięć w tekście

Return values

1	wystąpił błąd podczas procesu kopiowania
checkFlag	zmienna przechowywująca informacje czy wystąpił błąd

5.5 C:/Users/user/Desktop/daemon/copyRecursive.h File Reference

Functions

- int copyRecursiveDir (char ∗dirSource, char ∗dirDest, char ∗argv, int iter)

  Funkcja służąca do kopiowana rekursywnego wchodzenia w poszczególne katalogi, kopiowania ich oraz ich zawartości do folderu docelowego.

5.5.1 Function Documentation

5.5.1.1 copyRecursiveDir()

```
int copyRecursiveDir (
            char * dirSource,
            char * dirDest,
            char * argv,
            int iter )
```

Funkcja służąca do kopiowana rekursywnego wchodzenia w poszczególne katalogi, kopiowania ich oraz ich zawartości do folderu docelowego.

Parameters

in	*dirSource*	ścieżka źródłowa
in	*dirDest*	ścieżka docelowa
in	*argv*	parametry wejściowe
in	*iter*	służy do robienia wcięć w tekście

Return values

1	wystąpił błąd podczas procesu kopiowania
checkFlag	zmienna przechowywująca informacje czy wystąpił błąd

5.6 copyRecursive.h

Go to the documentation of this file.

```
00001 int copyRecursiveDir(char *dirSource, char *dirDest, char *argv, int iter);
```

5.7 C:/Users/user/Desktop/daemon/currentTime.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
```

Functions

- struct tm * currentTime ()

  Funkcja służąca do pobierania aktualnego czasu.

5.7.1 Function Documentation

5.7.1.1 currentTime()

```
struct tm * currentTime ( )
```

Funkcja służąca do pobierania aktualnego czasu.

Return values

timeinfo	aktualny czas

5.8 C:/Users/user/Desktop/daemon/currentTime.h File Reference

Functions

- struct tm * currentTime ()

  Funkcja służąca do pobierania aktualnego czasu.

5.8.1 Function Documentation

5.8.1.1 currentTime()

```
struct tm * currentTime ( )
```

Funkcja służąca do pobierania aktualnego czasu.

Return values

timeinfo	aktualny czas

5.9 currentTime.h

Go to the documentation of this file.

```
00001 struct tm* currentTime();
```

5.10 C:/Users/user/Desktop/daemon/daemon.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
```

Functions

- void create_deamon ()

  Funkcja tworząca demona.

5.10.1 Function Documentation

5.10.1.1 create_deamon()

```
void create_deamon ( )
```

Funkcja tworząca demona.

5.11 C:/Users/user/Desktop/daemon/daemon.h File Reference

Functions

- void create_deamon ()

  Funkcja tworząca demona.

5.11.1 Function Documentation

5.11.1.1 create_deamon()

```
void create_deamon ( )
```

Funkcja tworząca demona.

5.12 daemon.h

Go to the documentation of this file.

```
00001 extern void create_deamon();
```

5.13 C:/Users/user/Desktop/daemon/daemon_at_work.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include "daemon.h"
#include "copy.h"
#include "currentTime.h"
#include "deleteNotExisting.c"
#include "copyRecursive.h"
#include "updateTextFile.h"
```

Functions

- void daemon_at_work (char *argv[ ], int strLenSource, int strLenDest, char *dirSourcePath, char *dirDest↩ Path)

  Funkcja wywyływana przez demona.

5.13.1 Function Documentation

5.13.1.1 daemon_at_work()

```
void daemon_at_work (
            char * argv[],
            int strLenSource,
            int strLenDest,
            char * dirSourcePath,
            char * dirDestPath )
```

Funkcja wywyływana przez demona.

Parameters

in	*argv*	parametry wejściowe
in	*strLenSource*	długość ścieżki pliku źródłowego
in	*strLenDest*	długość ścieżki pliku docelowego
in	*dirSourcePath*	scieżka pliku źródłowego
in	*dirDestPath*	scieżka pliku docelowego

Return values

"	Pusty return występuje w wypadku błędu który możemy podejrzeć w pliku "errors.txt"

5.14 C:/Users/user/Desktop/daemon/daemon_at_work.h File Reference

Functions

- void daemon_at_work (char ∗argv[ ], int strLenSource, int strLenDest, char ∗dirSourcePath, char ∗dirDest↩ Path)

  Funkcja wywyływana przez demona.

5.14.1 Function Documentation

5.14.1.1 daemon_at_work()

```
void daemon_at_work (
            char * argv[],
            int strLenSource,
            int strLenDest,
            char * dirSourcePath,
            char * dirDestPath )
```

Funkcja wywyływana przez demona.

Parameters

in	*argv*	parametry wejściowe
in	*strLenSource*	długość ścieżki pliku źródłowego
in	*strLenDest*	długość ścieżki pliku docelowego
in	*dirSourcePath*	scieżka pliku źródłowego
in	*dirDestPath*	scieżka pliku docelowego

Return values

"	Pusty return występuje w wypadku błędu który możemy podejrzeć w pliku "errors.txt"

5.15 daemon_at_work.h

Go to the documentation of this file.

```
00001 void daemon_at_work(char *argv[],int strLenSource,int strLenDest,char *dirSourcePath,char
      *dirDestPath);
```

5.16 C:/Users/user/Desktop/daemon/deleteNotExisting.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "updateTextFile.h"
```

Data Structures

- struct doesItExists

  Struktura przechowywująca wszystkie pliki które znajdują się w folderze źródłowym z uwzględnieniem podziału na foldery oraz pliki.

Typedefs

- typedef struct doesItExists doesItExists_t

  Struktura przechowywująca wszystkie pliki które znajdują się w folderze źródłowym z uwzględnieniem podziału na foldery oraz pliki.

Functions

- int deleteRecursive (const char *dirPath, int iter)

  Funkcja usuwająca rekurencyjnie.
- void addToList (doesItExists_t **start, const char *name, int dirFlag)

  Funkcja dodająca nazwy plików i folderów do struktury "doesItExists".
- void deleteList (doesItExists_t **start)

  Wyczyszczenie listy.
- int isThereThatFile (doesItExists_t *start, char *name, int dirFlag)

  Sprawdza czy podany plik lub katalog jest na liście.

5.16.1 Typedef Documentation

5.16.1.1 doesItExists_t

```
typedef struct doesItExists doesItExists_t
```

Struktura przechowywująca wszystkie pliki które znajdują się w folderze źródłowym z uwzględnieniem podziału na foldery oraz pliki.

5.16.2 Function Documentation

5.16.2.1 addToList()

```
void addToList (
            doesItExists_t ** start,
            const char * name,
            int dirFlag )
```

Funkcja dodająca nazwy plików i folderów do struktury "doesItExists".

Parameters

in	*start*	początek listy
in	*name*	nazwa pliku lub katalogu
in	*dirFlag*	ustala rodzaj tzn. czy jest folderem

Return values

"	Pusty return nie informuje o błędzie

5.16.2.2 deleteList()

```
void deleteList (
            doesItExists_t ** start )
```

Wyczyszczenie listy.

Parameters

in	*start*	początek listy

5.16.2.3 deleteRecursive()

```
int deleteRecursive (
            const char * dirPath,
            int iter )
```

Funkcja usuwająca rekurencyjnie.

Parameters

in	*dirPath*	scieżka folderu do usunięcia
in	*iter*	służy do robienia wcięć w tekście

Return values

1	Zwracane w przypadku błędu który możemy podejrzeć w errors.txt
checkFlag	zmienna przechowywująca informacje czy wystąpił błąd

5.16.2.4 isThereThatFile()

```
int isThereThatFile (
            doesItExists_t * start,
            char * name,
            int dirFlag )
```

Sprawdza czy podany plik lub katalog jest na liście.

Parameters

in	*start*	początek listy
in	*name*	nazwa pliku lub katalogu
in	*dirFlag*	ustala rodzaj tzn. czy jest folderem

Return values

1	Plik nie został znaleziony
0	Plik został znaleziony

5.17 C:/Users/user/Desktop/daemon/main.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include <signal.h>
#include "daemon.h"
#include "copy.h"
#include "currentTime.h"
#include "daemon_at_work.h"
#include "sigusr1.h"
#include "updateTextFile.h"
```

Functions

- int main (int argc, char ∗argv[ ])

  Funkcja main rozpoczynająca działanie całego programu.

5.17.1 Function Documentation

5.17.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Funkcja main rozpoczynająca działanie całego programu.

Parameters

in	*argc*	Ilość parametrów wejściowych
in	*argv*	parametry wejściowe

Return values

1	Zwraca błąd, jego szczegóły można podejrzeć w errors.txt
0	Kończy działanie programu

5.18 C:/Users/user/Desktop/daemon/sigusr1.c File Reference

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include <signal.h>
#include "currentTime.h"
#include "updateTextFile.h"
```

Functions

- void funkcja_obslugujaca_sigusr1 (int numer)

  Funkcja informująca iż sygnał sigusr1 został przechwycony i zostanie obsłużony.

5.18.1 Function Documentation

5.18.1.1 funkcja_obslugujaca_sigusr1()

```
void funkcja_obslugujaca_sigusr1 (
            int numer )
```

Funkcja informująca iż sygnał sigusr1 został przechwycony i zostanie obsłużony.

Parameters

in	*numer*	zmnienna potrzebna do prawidłowego działania kodu

5.19 C:/Users/user/Desktop/daemon/sigusr1.h File Reference

Functions

- void funkcja_obslugujaca_sigusr1 (int numer)

  Funkcja informująca iż sygnał sigusr1 został przechwycony i zostanie obsłużony.

5.19.1 Function Documentation

5.19.1.1 funkcja_obslugujaca_sigusr1()

```
void funkcja_obslugujaca_sigusr1 (
            int numer )
```

Funkcja informująca iż sygnał sigusr1 został przechwycony i zostanie obsłużony.

Parameters

in	*numer*	zmnienna potrzebna do prawidłowego działania kodu

5.20 sigusr1.h

Go to the documentation of this file.

```
00001 void funkcja_obslugujaca_sigusr1(int numer);
```

5.21 C:/Users/user/Desktop/daemon/updateTextFile.c File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include "currentTime.h"
```

Functions

- void updateTextFile (const char ∗fileName, const char ∗text)

  Funkcja dodająca tekst do wybranego pliku.
- void updateTextFileParam (const char ∗fileName, const char ∗text, const char ∗param)

  Funkcja dodająca tekst do wybranego pliku z użyciem zmiennej.
- void updateTextFileRecursive (const char ∗fileName, const char ∗text, int iter)

  Funkcja dodająca tekst rekurencyjnie do wybranego pliku.
- void updateTextFileRecursiveParam (const char ∗fileName, const char ∗text, int iter, const char ∗param)

  Funkcja dodająca tekst rekurencyjnie do wybranego pliku z użyciem zmiennej.

5.21.1 Function Documentation

5.21.1.1 updateTextFile()

```
void updateTextFile (
            const char * fileName,
            const char * text )
```

Funkcja dodająca tekst do wybranego pliku.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst
in	*text*	Tekst który dodajemy do pliku

5.21.1.2 updateTextFileParam()

```
void updateTextFileParam (
            const char * fileName,
            const char * text,
            const char * param )
```

Funkcja dodająca tekst do wybranego pliku z użyciem zmiennej.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst
in	*text*	Tekst który dodajemy do pliku
in	*param*	Zmienna przechowywująca nazwy plików lub katalogów

5.21.1.3 updateTextFileRecursive()

```
void updateTextFileRecursive (
            const char * fileName,
            const char * text,
            int iter )
```

Funkcja dodająca tekst rekurencyjnie do wybranego pliku.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst
in	*text*	Tekst który dodajemy do pliku
in	*iter*	służy do robienia wcięć w tekście @

5.21.1.4 updateTextFileRecursiveParam()

```
void updateTextFileRecursiveParam (
            const char * fileName,
            const char * text,
            int iter,
            const char * param )
```

Funkcja dodająca tekst rekurencyjnie do wybranego pliku z użyciem zmiennej.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst
in	*text*	Tekst który dodajemy do pliku
in	*param*	Zmienna przechowywująca nazwy plików lub katalogów
in	*iter*	służy do robienia wcięć w tekście

5.22 C:/Users/user/Desktop/daemon/updateTextFile.h File Reference

Functions

- void updateTextFile (const char ∗fileName, const char ∗text)

  Funkcja dodająca tekst do wybranego pliku.
- void updateTextFileRecursive (const char ∗fileName, const char ∗text, int iter)

  Funkcja dodająca tekst rekurencyjnie do wybranego pliku.
- void updateTextFileParam (const char ∗fileName, const char ∗text, const char ∗param)

  Funkcja dodająca tekst do wybranego pliku z użyciem zmiennej.
- void updateTextFileRecursiveParam (const char ∗fileName, const char ∗text, int iter, const char ∗param)

  Funkcja dodająca tekst rekurencyjnie do wybranego pliku z użyciem zmiennej.

5.22.1 Function Documentation

5.22.1.1 updateTextFile()

```
void updateTextFile (
            const char * fileName,
            const char * text )
```

Funkcja dodająca tekst do wybranego pliku.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst
in	*text*	Tekst który dodajemy do pliku

5.22.1.2 updateTextFileParam()

```
void updateTextFileParam (
            const char * fileName,
            const char * text,
            const char * param )
```

Funkcja dodająca tekst do wybranego pliku z użyciem zmiennej.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst
in	*text*	Tekst który dodajemy do pliku
in	*param*	Zmienna przechowywująca nazwy plików lub katalogów

5.22.1.3 updateTextFileRecursive()

```
void updateTextFileRecursive (
            const char * fileName,
            const char * text,
            int iter )
```

Funkcja dodająca tekst rekurencyjnie do wybranego pliku.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst
in	*text*	Tekst który dodajemy do pliku
in	*iter*	służy do robienia wcięć w tekście @

5.22.1.4 updateTextFileRecursiveParam()

```
void updateTextFileRecursiveParam (
            const char * fileName,
            const char * text,
            int iter,
            const char * param )
```

Funkcja dodająca tekst rekurencyjnie do wybranego pliku z użyciem zmiennej.

Parameters

in	*fileName*	nazwa pliku do którego dodajemy tekst

Parameters

in	*text*	Tekst który dodajemy do pliku
in	*param*	Zmienna przechowywująca nazwy plików lub katalogów
in	*iter*	służy do robienia wcięć w tekście

5.23 updateTextFile.h

Go to the documentation of this file.

```
00001 void updateTextFile(const char *fileName,const char *text);
00002 void updateTextFileRecursive(const char * fileName,const char *text, int iter);
00003 void updateTextFileParam(const char *fileName,const char *text,const char *param);
00004 void updateTextFileRecursiveParam(const char * fileName,const char *text, int iter, const char*
      param);
```

Index