Demon

Wygenerowano przez Doxygen 1.9.6

Rozdział 1

Strona główna dokumentacji Demona

1.1 Dokumentacja

Wydział Informatyki Politechniki Białostockiej Przedmiot: Systemy Operacyjne	Data oddania: 28.04.2023 r.
Pracownia specjalistyczna nr 2 Zespół: 1. Mateusz Kondraciuk 2. Jakub Franciszek Modzelewski	Prowadzący: dr inż. Wojciech Kwedlo Ocena:

1.2 Realizacja zadania

W celu sprawdzenia treści zadania proszę udać się pod Treść Zadania.
W celu sprawdzenia wykonanych podpunktów proszę udać się pod Wykonane podpunkty.
W celu sprawdzenia instrukcji obsługi proszę udać się pod Instrukcja obsługi.

Rozdział 2

Treść Zadania

Temat 2 - Demon synchronizujący dwa podkatalogi
[12p.] Program który otrzymuje co najmniej dwa argumenty: ścieżkę źródłową oraz ścieżkę docelową . Jeżeli któraś ze ścieżek nie jest katalogiem program powraca natychmiast z komunikatem błędu. W przeciwnym wypadku staje się demonem. Demon wykonuje następujące czynności: śpi przez pięć minut (czas spania można zmieniać przy pomocy dodatkowego opcjonalnego argumentu), po czym po obudzeniu się porównuje katalog źródłowy z katalogiem docelowym. Pozycje które nie są zwykłymi plikami są ignorowane (np. katalogi i dowiązania symboliczne). Jeżeli demon (a) napotka na nowy plik w katalogu źródłowym, i tego pliku brak w katalogu docelowym lub (b) plik w katalogu źrodłowym ma późniejszą datę ostatniej modyfikacji demon wykonuje kopię pliku z katalogu źródłowego do katalogu docelowego - ustawiając w katalogu docelowym datę modyfikacji tak aby przy kolejnym obudzeniu nie trzeba było wykonać kopii (chyba ze plik w katalogu źródłowym zostanie ponownie zmieniony). Jeżeli zaś odnajdzie plik w katalogu docelowym, którego nie ma w katalogu źródłowym to usuwa ten plik z katalogu docelowego. Możliwe jest również natychmiastowe obudzenie się demona poprzez wysłanie mu sygnału SIGUSR1. Wyczerpująca informacja o każdej akcji typu uśpienie/obudzenie się demona (naturalne lub w wyniku sygnału), wykonanie kopii lub usunięcie pliku jest przesłana do logu systemowego. Informacja ta powinna zawierać aktualną datę.
a) [10p.] Dodatkowa opcja -R pozwalająca na rekurencyjną synchronizację katalogów (teraz pozycje będące katalogami nie są ignorowane). W szczególności jeżeli demon stwierdzi w katalogu docelowym
podkatalog którego brak w katalogu źródłowym powinien usunąć go wraz z zawartością.
b) [12p.] W zależności od rozmiaru plików dla małych plików wykonywane jest kopiowanie przy pomocy read/write a w przypadku dużych przy pomocy mmap/write (plik źródłowy) zostaje zamapowany w całości w pamięci. Próg dzielący pliki małe od dużych może być przekazywany jako opcjonalny argument.
Uwagi: (a) Wszelkie operacje na plikach i tworzenie demona należy wykonywać przy pomocy API Linuksa a nie standardowej biblioteki języka C (b) kopiowanie za każdym obudzeniem całego drzewa katalogów zostanie potraktowane jako poważny błąd (c) podobnie jak przerzucenie części zadań na shell systemowy (funkcja system). 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.
W celu sprawdzenia wykonanych podpunktów proszę udać się pod Wykonane podpunkty.

Rozdział 3

Wykonane podpunkty

3.1 Podpunkty

3.1.1 Główne zadanie

Program otrzymuje dwa katalogi. Co 5 minut sprawdza ich stan, dąży do tego aby katalog który został podany jako drugi argument posiadał te same pliki co katalog pierwszy. Kopiuje pliki i je usuwa.
Dodatkowo możliwe jest natychmiastowe wybudzenie demona wysyłając sygnał SIGUSR1.

3.1.2 Rekurencyjna synchronizacja katalogów

Dodatkowa opcja -R pozwala na rekurencyjną synchronizację katalogów.

3.1.3 Rozmiar pliku

Inna metoda kopiowania pliku w zależności od jego rozmiaru. Próg dzielący pliki na małe i duże jest przekazywany jako argument.

Rozdział 4

Instrukcja obsługi

4.1 Parametry podawane przy uruchamianiu programu

4.1.1 Folder źródłowy

Pierwszym parametrem do podania jest folder źródłowy- folder którego pliki i katalogi będziemy kopiować.

4.1.2 Folder docelowy

Drugim parametrem do podania jest folder docelowy- folder do którego pliki i katalogi będą kopiowane.

4.1.3 Rozmiar pliku

Kolejnym parametrem jest wielkość która będzie dzieliła pliki na "małe" i "duże". Wielkość ta jest podawana w b.

4.1.4 Rekurencyjna synchronizacja katalogów -R (opcjonaly)

Ostatnim parametrem jest opcja -R pozwalająca na rekurencyjną synchronizację katalogów.

4.2 Przykłady

4.2.1 ./program source dest 1000000

Wywołujemy program, jako folder źródłowy został wybrany katalog source, jako docelowy dest, 1000000b jest wielkością dzielącą pliki na małe i duże.

4.2.2 ./program source dest 1000000 -R

To samo co w poprzednim z tą różnicą, że dodatkowo została użyta opcja -R, czyli nasz program będzie rekurencyjnie synchronizował katalogi.

Rozdział 5

Indeks struktur danych

5.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

Rozdział 6

Indeks plików

6.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 7

Dokumentacja struktur danych

7.1 Dokumentacja struktury doesItExists

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

Pola danych

- 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.

7.1.1 Opis szczegółowy

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

Dokumentacja dla tej struktury została wygenerowana z pliku:

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

Rozdział 8

Dokumentacja plików

8.1 Dokumentacja pliku C:/Users/user/Desktop/daemon/copy.c

Plik obsługujący kopiowanie.

```
#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>
```

Definicje

- #define **PACKAGE** "mmap"

  W informatyce mmap jest uniksowym wywołaniem systemowym zgodnym z POSIX, które odwzorowuje pliki lub urządzenia w pamięci.

Funkcje

- 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.

8.1.1 Opis szczegółowy

Plik obsługujący kopiowanie.

Plik którego funkcje dzielą pliki po rozmiarze a następnie je kopiują.

8.1.2 Dokumentacja funkcji

8.1.2.1 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.

Parametry

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

Zwracane wartości

0	jest zwracane w przypadku sukcesu

8.1.2.2 copyFile()

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

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

Parametry

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

Zwracane wartości

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

8.1.2.3 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.

Parametry

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

Zwracane wartości

0	zwaracane jako informacja, że pliku nie należy skopiować
1	zwaracane jako informacja, że plik należy skopiować

8.1.2.4 copySmallFile()

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

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

Parametry

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

Zwracane wartości

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

8.2 copy.h

```
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);
```

8.3 Dokumentacja pliku C:/Users/user/Desktop/daemon/copyRecursive.c

Kopiowanie rekurencyjne.

```
#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"
```

Funkcje

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

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

8.3.1 Opis szczegółowy

Kopiowanie rekurencyjne.

Plik posiadający jedną funkcję, której zadaniem jest kopiowanie rekurencyjne.

8.3.2 Dokumentacja funkcji

8.3.2.1 copyRecursiveDir()

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

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

Parametry

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

Zwracane wartości

1	Zwraca błąd, jego szczegóły można sprawdzić w pliku errors.txt
checkFlag	zmienna przechowywująca informacje czy wystąpił błąd

8.4 copyRecursive.h

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

8.5 Dokumentacja pliku C:/Users/user/Desktop/daemon/currentTime.c

Aktualna godzina i data.

```
#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>
```

Funkcje

- struct tm * currentTime ()

  Funkcja służąca do pobierania aktualnego czasu.

8.5.1 Opis szczegółowy

Aktualna godzina i data.

Plik posiadający jedną funkcję, której zadaniem jest zwracanie aktualnego czasu.

8.5.2 Dokumentacja funkcji

8.5.2.1 currentTime()

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

Funkcja służąca do pobierania aktualnego czasu.

Zwracane wartości

timeinfo	aktualny czas

8.6 currentTime.h

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

8.7 Dokumentacja pliku C:/Users/user/Desktop/daemon/daemon.c

Tworzenie demona.

```
#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>
```

Funkcje

- void **create_deamon** ()

  Funkcja tworząca demona- tworzony jest proces potomny, następnie zabijany proces "rodzic". "Osierocony" proces staje się demonem.

8.7.1 Opis szczegółowy

Tworzenie demona.

Plik posiadający funkcję która służy do tworzenia demona.

8.8 daemon.h

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

8.9 Dokumentacja pliku C:/Users/user/Desktop/daemon/daemon_at_work.c

Demon.

```
#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"
```

Funkcje

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

  Funkcja wywyływana przez demona, służy do sprawdzanie potrzeby kopiowania czy usuwania plików.

8.9.1 Opis szczegółowy

Demon.

Plik posiadający jedną funkcję, która jest wywoływana przez demona, służy do sprawdzanie potrzeby kopiowania czy usuwania plików.

8.9.2 Dokumentacja funkcji

8.9.2.1 daemon_at_work()

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

Funkcja wywyływana przez demona, służy do sprawdzanie potrzeby kopiowania czy usuwania plików.

Parametry

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

Zwracane wartości

"	Zwraca błąd, jego szczegóły można sprawdzić w pliku errors.txt

8.10 daemon_at_work.h

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

8.11 Dokumentacja pliku C:/Users/user/Desktop/daemon/deleteNotExisting.c

Usuwanie plików.

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

Struktury danych

- struct doesItExists

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

Definicje typów

- typedef struct doesItExists **doesItExists_t**

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

Funkcje

- 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.

8.11.1 Opis szczegółowy

Usuwanie plików.

Plik sprawdzający czy podany plik lub katalog istnieje w folderze źródłowym, jeśli nie, jest usuwany.

8.11.2 Dokumentacja funkcji

8.11.2.1 addToList()

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

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

Parametry

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

Zwracane wartości

"	Pusty return nie informuje o błędzie

8.11.2.2 deleteList()

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

Wyczyszczenie listy.

Parametry

in	*start*	początek listy

8.11.2.3 deleteRecursive()

```
int deleteRecursive (
```

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

Funkcja usuwająca rekurencyjnie.

Parametry

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

Zwracane wartości

1	Zwraca błąd, jego szczegóły można sprawdzić w pliku errors.txt
checkFlag	zmienna przechowywująca informacje czy wystąpił błąd

8.11.2.4 isThereThatFile()

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

Sprawdza czy podany plik lub katalog jest na liście.

Parametry

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

Zwracane wartości

1	Plik nie został znaleziony
0	Plik został znaleziony

8.12 Dokumentacja pliku C:/Users/user/Desktop/daemon/main.c

Główny plik.

```
#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"
```

Funkcje

- int main (int argc, char ∗argv[ ])

  Funkcja main wywoływana na początku działania programu, po odebraniu parametrów wejściowych wywołuje demona.

8.12.1 Opis szczegółowy

Główny plik.

Plik rozpoczynający pracę całego programu.
To tu sprawdzane jest wywołanie sygnału SIGUSR1.
W tym pliku możemy zmienić czas "snu" demona.

8.12.2 Dokumentacja funkcji

8.12.2.1 main()

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

Funkcja main wywoływana na początku działania programu, po odebraniu parametrów wejściowych wywołuje demona.

Parametry

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

Zwracane wartości

1	Zwraca błąd, jego szczegóły można sprawdzić w pliku errors.txt

Zwracane wartości

0	Kończy działanie programu

8.13 Dokumentacja pliku C:/Users/user/Desktop/daemon/sigusr1.c

Sygnał

```
#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"
```

Funkcje

- void funkcja_obslugujaca_sigusr1 (int numer)

  Funkcja informująca iż sygnał sigusr1 został przechwycony i zostanie obsłużony.

8.13.1 Opis szczegółowy

Sygnał

Plik obsługujący wywołanie sygnału SIGUSR1

8.13.2 Dokumentacja funkcji

8.13.2.1 funkcja_obslugujaca_sigusr1()

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

Funkcja informująca iż sygnał sigusr1 został przechwycony i zostanie obsłużony.

Parametry

in	*numer*	zmnienna potrzebna do prawidłowego działania kodu

8.14 sigusr1.h

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

8.15 Dokumentacja pliku C:/Users/user/Desktop/daemon/updateTextFile.c

Aktualizowanie logs.txt i errors.txt.

```
#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"
```

Funkcje

- 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.

8.15.1 Opis szczegółowy

Aktualizowanie logs.txt i errors.txt.

Plik służący do aktualizacji plików zawierających logi i błędy o nowe wartości.

8.15.2 Dokumentacja funkcji

8.15.2.1 updateTextFile()

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

Funkcja dodająca tekst do wybranego pliku.

Parametry

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

8.15.2.2 updateTextFileParam()

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

Funkcja dodająca tekst do wybranego pliku z użyciem zmiennej.

Parametry

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

8.15.2.3 updateTextFileRecursive()

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

Funkcja dodająca tekst rekurencyjnie do wybranego pliku.

Parametry

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

8.15.2.4 updateTextFileRecursiveParam()

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

Funkcja dodająca tekst rekurencyjnie do wybranego pliku z użyciem zmiennej.

Parametry

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

8.16 updateTextFile.h

```
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);
```

Skorowidz