Leszifr3

(JPW 2018)

zastosowanie biblioteki

Biblioteka leszifr3lib.dll (lub statyczna leszifr3lib.a) pozwala na bezpośrednie łączenie kodu z przykładem w pliku leszifr3cmd.c oraz z innymi programami. Biblioteka jest skompilowana w GCC z source code w C.

Uwaga!

Nazwy plików po szyfrowaniu mają zamienione polskie znaki diakrytyczne na odpowiadające litery w ACII np. litera "ą" w nazwie pliku \*.archi będzie zastąpiona literą "a". Ponadto wszystkie inne znaki w nazwie poza literami małymi i dużymi, cyframi oraz znakami "."   
i "-" (minus) zostaną zastąpione podkreślnikiem "\_". Zmiana nazwy natywnej zawierająca polskie znaki diakrytyczne oraz usunięcie innych znaków z tabeli ASCI poza wymienionymi jest spowodowana zgodnością z wymogami REPO. Na przykład:

nazwa pliku natywnego: "moje zdjęcie z wakacji [2016].jpg" po szyfrowaniu będzie wyglądać następująco: : "moje\_zdjecie\_z\_wakacji\_\_2016\_.jpg.archi" . Należy również zwrócić uwagę, że nazwy plików nie mogą być dłuższe niż 255 znaków (liczone w kodowaniu utf-8). Kodowanie cp1250 (WIndows) używa drugiej strony kodowej ASCII (kodowanie 1-bajtowe) znaków diakrytycznych, natomiast utf-8 koduje polskie znaki diakrytyczne za pomocą 2 bajtów. To powoduje, że nazwa pliku po kodowaniu utf-8 może być dłuższa jeśli nazwa pliku w kodowaniu cp1250 zawiera polskie znaki diakrytyczne. Leszif3 zgłosi to jako błąd (filaneme too long). Jeśli wbudowana wewnętrzna metryczka pliku szyfrowanego będzie kodowana przekroczy 3000 znaków wówczas zostanie automatycznie skrócona. Leszifr3 nie poinformuje o błędzie. Przy kodowaniu utf-8 liczba znaków zmniejszy się o dwubajtowe kody znaków diakrytycznych. Wówczas dostępna długość tekstu zmniejszy się i leszifr3 skróci metryczkę do 3000 bajtów.

Funkcje nagłówkowe w pliku leszifr3.h:

1. HELP

**void print\_help(void);**

Wywołanie tej funkcji w programie wyświetla standardowe USAGE.

Funkcja jest bezparametrowa.

2. Włącznik kodowania utf-8 znaków diakrytycznych

Kodowanie znaków w Windows jest ustawione w PL na cp1250.   
 Aby zachować zgodność z OSami Linux i MacOS należy ustawić  
 argument w przełączniku na sw=1. Jeśli przełącznik będzie   
 ustawiony na 0 (lub inną wartość) wówczas kodowanie znaków  
 diakrytycznych będzie w cp1250. Leszifr3 załączoną metryczkę  
 oraz natywne nazwy kodowanych plików (przy ustawionym SW = 1)

przekonwertuje automatycznie z utf-8 na cp1250. Z punktu widzenia  
 użytkownika dane w metryczce oraz nazwy plików będą zawsze   
 zawierać właściwe znaki diakrytyczne w kodowaniu cp1250.

Funkcję:

**void switch\_utf\_converting(int sw);**

sw = 1 włącza konwersję z cp1250 na utf-8

sw = 0 wyłącza konwersję i zachowuje kodowanie cp1250

należy sw zadeklarować na początku głównego programu.   
 Domyślnie parametr jest ustawiony na konwersję cp1250 -> utf-8.

3. Kodowane (szyfrowanie)

//z wyłączonym windows progress bar

**int leszifr3code(const char\*fname, const char\*passwd, int print);**

//z włączonym windows progress bar

**int leszifr3code\_g(const char\*fname, const char\*passwd, int print);**

funkcja ma 3 argumenty:

fname - nazwa pliku przeznaczonego do szyfrowania. Nazwę należy

podać wraz ze ścieżka jeśli leszifr znajduje sie w innym

katalogu niż plik przeznaczony do szyfrowania.

Nazwa pliku (bez ścieżki) może mieć max 256 znaków,

inaczej nastąpi obcięcie nazwy i funkcja nie będzie

działać.

password - hasło musi posiadać długość co najmniej 4 znaki

max do 40 znaków.

print - Wpisanie 1 spowoduje wyświetlenie błędów.

Podanie innej liczby (najlepiej 0) zablokuje wypisanie

błędów na ekranie.

Funkcja zwraca nr bledu oraz 0 (brak bledu).

Wszystkie bledy maja wartości ujemne co pozwala na sterowanie

wynikami działania programu. Tabela błędów

jest zawarta w leszifr3lib.o i opis bledu zostanie podany jeśli

argument print=1.

3. Dekodowanie (de-szyfrowanie)

//z wyłączonym windows progress bar

**int leszifr3decode(const char\*fname, const char\*passwd, int print);**

//z włączonym windows progress bar

**int leszifr3decode\_g(const char\*fname, const char\*passwd, int print);**

Opis parametrów jest taki sam jak w punkcie 2 (szyfrowanie).

Różnica dotyczy nazwy pliku, ponieważ plik musi być to plik

zaszyfrowany przez leszifr3. Zazwyczaj będzie to plik

z rozszerzeniem ".archi".

4. Odczyt dodanego opisu pliku

**int leszifr3desc(const char\*fname, const char\*passwd, char\*out, int print);**

Opis argumentów jest taki sam jak w punkcie 2 z uwaga w punkcie 3.

Argument out jest tablica o rozmiarze 3000 i wymagana jest deklaracja

tablicy w programie np. char out[3000] lub char\*out = (char\*)malloc(3000);

Uwaga, tekst w out zachowuje znaki sterujące/formatujące. Należy pamiętać  
 o tym, że tekst metryczki może zawierać do 3000 bajtów. Z tego powodu   
 deklarowana tablica musi mieć rozmiar 3000 bajtów. Inaczej program może  
 działać awaryjnie ponieważ przy mniejszej tablicy może wystąpić przekroczenie  
 zakresu deklarowanej tablicy.

5. Odczyt wersji pliku lub wersji generatora

**void leszifr3ver(const char\*ver);**

Funkcja wyświetla wersje leszifr3 lub wersje i typ generatora

szumu bitowego.

Jeśli argument ver jest stringem "-v" (minus mała litera v)

wówczas zostanie wyświetlona wersja leszifr3.

Wpisanie do var stringu "-vg" spowoduje wyświetlenie wersji

generatora szumu bitowego. Podanie innych wpisów nie spowoduje

żadnej odpowiedzi.

Kompilacja

Przykładowa kompilacja znajduje się w pliku Make.bat i zawiera następujące linie:

dla kompilacji z biblioteką statyczną:

gcc -Wall -o leszifr3.exe leszifr3cmd.c leszifr3lib.a

(wykomentowana)

oraz dla kompilacji z biblioteką dynamiczną:

gcc -Wall -o leszifr3.exe leszifr3cmd.c leszifr3lib.dll