Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki
Katedra Elektrotechniki Przemysłowej i Automatyki
Zakład Urządzeń i Systemów Automatyki

Programowanie komputerów 2

Operacje na plikach w języku C

Dostęp niskopoziomowy

Instrukcja laboratoryjna
(wersja robocza)

Paweł Strączyński
2016

W języku C dostęp do plików możliwy jest na dwa sposoby:

- wysokopoziomowy – wykorzystujący tzw. strumienie,
- niskopoziomowy – z wykorzystaniem tzw. deskryptorów plików.

Tematem zajęć jest dostęp do plików z wykorzystaniem deskryptorów (sposób niskopoziomowy).

Niskopoziomowy dostęp do pliku

Niskopoziomowy dostęp do pliku w języku możliwy jest z wykorzystaniem tzw. deskryptora pliku. Deskryptorem pliku nazywa się identyfikator (liczba całkowita - typu int) używany przez system operacyjny do obsługi operacji we/wy. Podobnie jak w przypadku dostępu wysokopoziomowego wykonanie operacji na pliku powinno składać się z następujących etapów:

- otwarcie pliku – funkcja ***open()***
- operacje na pliku np. odczyt, zapis - ***read()***, ***write()***.
- zamkniecie pliku– funkcja ***close()***

1. Otwarcie pliku

Deklarację funkcji ***open()*** przedstawiono na listingu 1. Funkcja pobiera dwa argumenty:

- nazwę pliku lub bezwzględną ścieżkę do pliku,
- tryb otwarcia – atrybuty przedstawiono w tabeli 1.

Funkcja zwraca deskryptor pliku (nieujemna liczba całkowita). W przypadku gdy nie można wykonać operacji otwarcia (brak pliku itp.) funkcja zwraca wartość mniejszą od zera.

Listing 1. Deklaracja funkcji open()

```
int open(const char *filename, const char *mode );
```

Tabela 1. Tryby otwarcia pliku

Tryb	Znaczenie
O_RDONLY	tryb tylko do odczytu
O_WRONLY	tryb tylko do zapisu
O_RDWR	tryb zapisu i odczytu

Ponadto istnieje możliwość otwarcia pliku z dodatkowym parametrem – tabela 2. Aby użyć parametru należy zsumować bitowo wartość parametru z wartością trybu otwarcia. Przykładowe użycie przedstawiono na listingu poniżej.

Listing 2. Przykładowe użycie funckji open()

```
fd = open("plik.bin", O_WRONLY |O_CREAT | S_IRWXU  | S_IRWXG | S_IRWXO  );
```

Tabela 2. Parametry otwarcia pliku

Parametr	Znaczenie
O_CREAT	utwórz plik jeśli nie istnieje
O_EXCL	zwróć błąd jeśli plik który ma być utworzony już istnieje
O_APPEND	tryb dopisywania, wskaźnik pozycji na koniec pliku

2. Odczyt danych z pliku

Do odczytu danych z pliku służy funkcja ***read()***. Jej deklarację przedstawiono na listingu 3. Funkcja ta pobiera trzy argumenty:

- deskryptor odczytywanego pliku,
- bufor w którym mają być przechowywane odczytane dane,
- liczba bajtów do odczytu.

Funkcja zwraca liczbę odczytanych bajtów. Gdy napotkany zostanie koniec pliku (*EOF*) funkcja zwraca 0, natomiast w przypadku błędu zwraca -1.

Listing 3. Deklaracja funkcji read()

```
int read(int file_deskr, char *buff, int nbyte );
```

3. Zapis danych do pliku

Zapis danych do pliku możliwy jest z wykorzystaniem funkcji ***write()***. Deklarację funkcji przedstawiono na listingu 4. Funkcja ta podobnie jak funkcja ***read()*** pobiera argumenty:

- deskryptor pliku do zapisu,
- bufor z danymi do zapisu,
- liczba bajtów do zapisu.

Funkcja zwraca liczbę zapisanych bajtów. W przypadku błędu zwracana jest liczba -1.

Listing 4. Deklaracja funkcji write()

```
int write(int file_deskr, char *buff, int nbyte );
```

4. Zamknięcie pliku

Do zamknięcia pliku służy funkcja ***close()***. Jako argument pobiera deskryptor do pliku który ma zostać zamknięty. W przypadku powodzenia funkcja zwraca 0, natomiast w przypadku błędu zwracana jest wartość -1.

Listing 5. Deklaracja funkcji close()

```
int close( int file_deskr);
```

Zadania do wykonania

Zadanie 1

Napisać program zliczający i wyświetlający na ekranie dla wskazanego pliku tekstowego:

- liczbę linii,
- liczbę znaków,
- liczbę spacji.

Wykorzystać niskopoziomowy dostęp do pliku.

Zadanie 2

Zmodyfikować program z zadania 1 tak aby dane nie były drukowane na standardowym wyjściu, a dopisywane na końcu pliku.

Zadanie 3

Napisać program zamieniający małe litery w pliku tekstowym na duże. Przekonwertowany tekst powinien być zapisywany w nowym pliku – zawartość pliku pierwotnego pozostawić bez zmian. Na listingach poniżej przedstawiono przykładową zawartość pliku przed i po konwersji. Operacje na pliku przeprowadzić z wykorzystaniem deskryptora pliku.

Listing 6. Przykładowa zawartość pliku przed konwersją

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incidi-
dunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercita-
tion ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id
est laborum.
```

Listing 7. Przykładowa zawartość pliku po konwersji

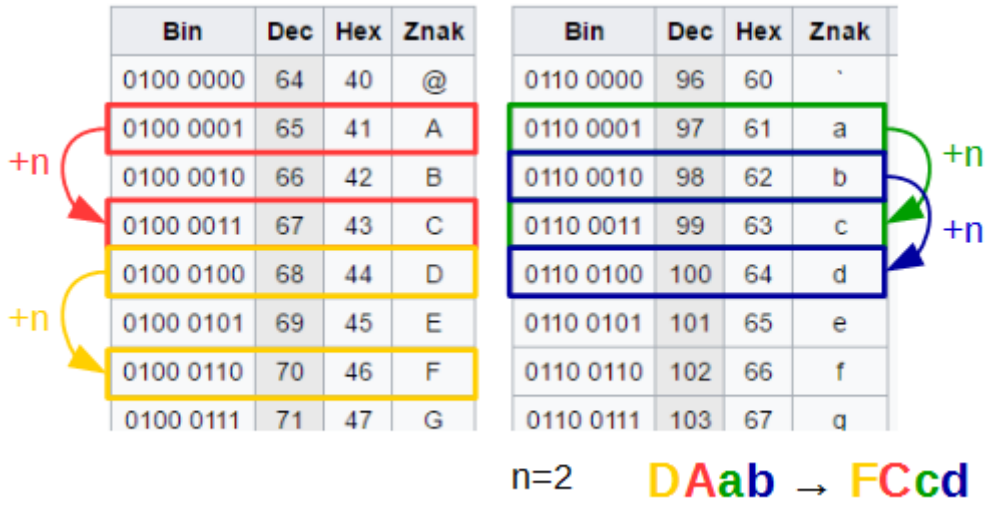
```
LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO EIUSMOD TEMPOR INCIDI-
DUNT UT LABORE ET DOLORE MAGNA ALIQUA. UT ENIM AD MINIM VENIAM, QUIS NOSTRUD EXERCITA-
TION ULLAMCO LABORIS NISI UT ALIQUIP EX EA COMMODO CONSEQUAT. DUIS AUTE IRURE DOLOR IN
REPREHENDERIT IN VOLUPTATE VELIT ESSE CILLUM DOLORE EU FUGIAT NULLA PARIATUR. EXCEPTEUR
SINT OCCAECAT CUPIDATAT NON PROIDENT, SUNT IN CULPA QUI OFFICIA DESERUNT MOLLIT ANIM ID
EST LABORUM.
```

Zadanie 4

Napisać program szyfrujący zawartość pliku tekstowego. Ideę algorytmu szyfrującego jaki należy zastosować przedstawiono na rysunku 1. Nazwa pliku do zaszyfrowania oraz wartość przesunięcia powinny być przekazywane jako argumenty programu. Plik po konwersji powinien przyjmować nazwę według zależności przedstawionej na listingu 7. Zawartość pliku pierwotnego powinna zostać bez zmian. Utworzyć drugi program służący do rozszyfrowywania pliku. Wykorzystać niskopoziomowy dostęp do pliku.

Listing 7. Nazwa pliku po konwersji

```
plik1.txt - > plik1_conv.txt
```



Rysunek 1. Idea systemu szyfrowania pliku tekstowego