Wojciech Myszka

Laboratorium 14: Zaawansowane wejście/wyjście

2016-05-29 09:47:14 +0200

1. Zadanie

Napisz program czytający z pliku informacje i wyświetlający każdy bajt w kilku formatach. Na przykład tak: plik o poniższej zawartości:

```
#include <math.h>
#include <stdio.h>
int main()
{
    printf( "%lf\n", pow(2., 3.) );
    return 0;
}
```

jest drukowany tak:

```
0000000    #    i    n    c    l    u    d    e         <
         043  151  156  143  154  165  144  145  040  074
          35  105  110   99  108  117  100  101   32   60
0000010    m    a    t    h    .    h    >   \n    #    i
         155  141  164  150  056  150  076  012  043  151
         109   97  116  104   46  104   62   10   35  105
0000020    n    c    l    u    d    e         <    s    t
         156  143  154  165  144  145  040  074  163  164
         110   99  108  117  100  101   32   60  115  116
0000030    d    i    o    .    h    >   \n    i    n    t
         144  151  157  056  150  076  012  151  156  164
         100  105  111   46  104   62   10  105  110  116
0000040         m    a    i    n    (    )   \n    {   \n
         040  155  141  151  156  050  051  012  173  012
          32  109   97  105  110   40   41   10  123   10
0000050                        p    r    i    n    t    f
         040  040  040  040  160  162  151  156  164  146
          32   32   32   32  112  114  105  110  116  102
0000060    (         "    %    l    f    \    n    "    ,
```

```
          050  040  042  045  154  146  134  156  042  054
           40   32   34   37  108  102   92  110   34   44
0000070          p    o    w    (    2    .    ,         3
          040  160  157  167  050  062  056  054  040  063
           32  112  111  119   40   50   46   44   32   51
0000080    .    )         )    ;   \n
          056  051  040  051  073  012  040  040  040  040
           46   41   32   41   59   10   32   32   32   32
0000090    r    e    t    u    r    n         0    ;   \n
          162  145  164  165  162  156  040  060  073  012
          114  101  116  117  114  110   32   48   59   10
0000100    }   \n   \n
          175  012  012
          125   10   10
0000103
```

Powyższy efek osiągnięto używając systemowego programu od: od −Ad −cb −td1 −w10 a.c

Najpierw jest numer bajtu od początku pliku. Później zawartość tekstowa. kolejne trzy linie zawierają zawartość każdego baju wyświetlaną dziesiętnie, szesnastkowo i ósemkowo.

2. Podpowiedzi

1. Można przećwiczyć pytanie o nazwę pliku i otwieranie go w trybie dowolnym (tekstowym lub binarnym) oraz czytanie linia po linii lub bajt po bajcie. Ale czytanie w trybie tekstowym ogranicza możliwości programu!
2. Równie dobre rozwiązanie to takie w którym program czyta ze standardowego wejścia. Żeby móc czytać również pliki binarne — trzeba czytać je bajt po bajcie (znak po znaku) używając albo funkcji getc(stdin) albo funkcji scanf("%c", $znak).
3. Można również zaprogramować rozwiązanie w którym program po uruchomieniu sprawdza jak został wywołany i w zależności od liczby parametrów albo czyta ze standardowego wejścia (tylko nazwa programu) albo dodatkowy parametr traktuje jako nazwę pliku. Przykładowy program wygląda tak:

```
/*
 * czytanie.c
 *
 * Copyright 2016 wojciech myszka <myszka@norka>
 *
 * This program is free software; you can redistribute it and/or modi
 * it under the terms of the GNU General Public License as published b
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
```

```
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.
 *
 *
 */
#include <stdio.h>
int main(int argc, char **argv)
{
    FILE * plik;
    if ( argc == 1 )
    {
        printf("Tylko jeden argument (nazwa programu)" \
               " -- czytam z stdin\n");
        plik = stdin;
        /* Takie podstawienie można zrobić, i zmienna
         * plik będzie strukturą ze standardowym wejściem.
         */
    }
    else
    {
        printf("Argumentów wiecej, zakladam, że pierwszy" \
               " to nazwa pliku: %s\n", argv[1]);
        plik = fopen(argv[1], "r");
        if ( plik == NULL )
        {
            printf("Nie mogę otworzyć pliku!\n");
            return 2;
        }
    }
    /*
     * W tym miejscu mamy otworzony plik
     */
```

```
    char napis[100];
    int status;
    while ( ( status = fscanf(plik, "%99s", napis) ) > 0 )
        printf("status = %d : %s\n", status, napis);
    fclose(plik); // Zamykamy
    return 0;
}
```

4. Inna metoda polega na wczytaniu całego pliku do tablicy, a następnie wykonaniu wszystkich operacji na danych w pamięci.
   Tablica będzie typu **unsigned char**, co może dziwić, ale jest to najprostszy typ pozwalający efektywnie przechowywać zawartość pojedynczych bajtów.

```
/*
 * caly.c
 *
 * Copyright 2016 wojciech myszka <myszka@norka>
 *
 * This program is free software; you can redistribute it and/or modi
 * it under the terms of the GNU General Public License as published b
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.
 *
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void usage(char * name)
{
    printf("Uzycie: \n" \
           "        %s <nazwa pliku>\n" \
```

```
            "␣␣␣␣␣␣␣␣koncze␣prace!\n", name);
}

int main(int argc, char **argv)
{
    int status;
    int dlugosc;
    char polecenie[256] = {
        "ls␣-l␣"
    };
    FILE * plik;
    if ( argc == 1 )
    {
        usage(argv[0]);
        return 1;
    }
    else
    {
        plik = fopen(argv[1], "rb");
        if ( plik == NULL )
        {
            printf("Nie␣moge␣otworzyć␣pliku!\n");
            return 2;
        }
    }
    fseek(plik, 0L, SEEK_END);      // Przesuwamy wskaźnik na koniec p
    dlugosc = ftell(plik);          // Odczytujemy długość pliku
    printf("plik␣ma␣dlugosc␣%d␣bajtow\n", dlugosc);
    /*
     * Poniżej mały żarcik. Używam funkcji system do uruchomienia
     * programu ls, który wyświetla pozycję z katalogu związaną
     * z plikiem. Można tam odczytać długość pliku.
     */
    strcat(polecenie, argv[1]);
    system(polecenie);
    /*
     * Przydzielam tablicę odpowiedniej długości
     * Typ unsigned char wybrano, bo to najprostszy
     * typ, w którym łatwo przechowywać bajty.
     */
```

```
    unsigned char * bufor  = malloc(dlugosc);
    if ( bufor == NULL )
    {
        printf("Nie moge przdzielic pamieci\n");
        return 3;
    }
    rewind(plik);       //Przewijam plik na początek
    status = fread(bufor, 1, dlugosc, plik);
    if ( status != dlugosc )
    {
        printf("Cos poszlo zle!");
        free(bufor);
        fclose(plik);
        return 4;
    }
    /*
     * W tym miejscu mamy w tablicy bufor wczytany cały plik
     * możemy o przetwarzać w dowolny sposób.
     * Dostęp do i-tego bajtu pliku uzyskujemy przez
     * bufor[i]
     */
    free(bufor);
    fclose(plik);
    return 0;
}
```

3. Wersja PDF tego dokumentu...

...pod adresem.

Wersja: 55 z drobnymi modyfikacjami! data ostatniej modyfikacji 2016-05-29 09:47:14 +0200