

Wstęp do programowania w języku C

Marek Piotrów - Wykład 9
Operacje na plikach oraz unie i pola bitowe

9 grudnia 2021

System plików - operacje na plikach

- System plików jest częścią systemu operacyjnego. System plików umożliwia przechowywanie informacji na różnych urządzeniach (pamięciach zewnętrznych) w jednolity sposób.
- Różne systemy operacyjne mają różne systemy plików. W większości pliki umieszczane są w hierarchicznej strukturze katalogów i są opisane nazwą i innymi atrybutami (np. czasami utworzenia i ostatniej modyfikacji, prawami użytkowników do dostępu do zawartości pliku).
- Biblioteka standardowa C pozwala na (w miarę) jednolity dostęp do systemu plików niezależnie od systemu operacyjnego.

System plików - otwarcie i zamknięcie pliku

- Każdy dostęp do nazwanego pliku z programu w C musi rozpocząć się od operacji *otwarcia pliku*:
`FILE *fopen(const char *path, const char *mode)`
- Otwarcie pliku może zakończyć się sukcesem lub porażką.
- Jeśli kończy się sukcesem, to `fopen` zwraca wskaźnik do standardowej struktury `FILE` opisującej otwarty plik.
- Jeśli kończy się porażką (plik nie istnieje lub nie można go utworzyć, nie mamy dostatecznych praw dostępu do pliku lub katalogu, itp.), to `fopen` zwraca `NULL`.
- Każdy otwarty plik musi być zamknięty operacją *zamknięcia pliku* `int fclose(FILE *fp)` Funkcja ta zwraca 0 po poprawnym zamknięciu pliku.

System plików - operacje pisania i czytania

- Wskaźnik zwracany przez `fopen` jest potrzebny do operacji czytania pliku i pisania do pliku. Jest on pierwszym (lub ostatnim) argumentem poszczególnych operacji.
- Do sformatowanego czytania z plików tekstowych mamy funkcję: `int fscanf(FILE *stream, const char *format, ...)`
- Do sformatowanego pisania do plików tekstowych mamy funkcję: `int fprintf(FILE *stream, const char *format, ...)`
- Pojedyncze znaki możemy czytać używając `int getc(FILE *stream)` oraz pisać za pomocą `int putc(int c, FILE *stream)`.

Kopiowanie pliku I

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

/***** KOPIOWANIE PLIKU *****/

void kopiuuj(FILE *zfp, FILE *dofp);

int main(int argc, char *argv[])
{
    FILE *zfp, *dofp;
    char *nazwa_z=NULL;
    char *nazwa_do=NULL;
    int i, binarnie=0;

    if (argc == 1) {
        fprintf(stderr, "\nUzycie: %s <plik zrodlowy> <plik docelowy> [-b]\n",
                argv[0]);
        fputs("\n\t-b oznacza kopiowanie pliku binarnego\n\n", stderr);
        exit(1);
    }
}
```

Kopiowanie pliku II

```
for (i=1; i < argc; ++i)
    if (argv[i][0] == '-' )
        if (tolower(argv[i][1]) == 'b' && argv[1][2] == '\\0')
            binarnie=1;
        else {
            fprintf(stderr, "%s: nieznana opcja: %s\n", argv[0], argv[i]);
            exit(2);
        }
    else if (nazwa_z == NULL)
        nazwa_z=argv[i];
    else if (nazwa_do == NULL)
        nazwa_do=argv[i];

if (nazwa_z == NULL)
    zfp=stdin;
else if ((zfp=fopen(nazwa_z, (binarnie? "rb" : "r"))) == NULL) {
    fprintf(stderr, "%s: blad otwarcia pliku: %s\n", argv[0], nazwa_z);
    exit(2);
}
if (nazwa_do == NULL)
    dofp=stdout;
else if ((dofp=fopen(nazwa_do, (binarnie? "wb" : "w"))) == NULL) {
    fprintf(stderr, "%s: blad otwarcia pliku: %s\n", argv[0], nazwa_do);
    exit(2);
}
kopiuj(zfp, dofp);
```

Kopiowanie pliku III

```
if (ferror(zfp)) {  
    fprintf(stderr,"%s: blad odczytu pliku: %s\n",argv[0],  
            (nazwa_z != NULL ? nazwa_z : "standardowe wejście"));  
    exit(2);  
}  
if (ferror(dofp)) {  
    fprintf(stderr,"%s: blad zapisu pliku: %s\n",argv[0],  
            (nazwa_do != NULL ? nazwa_do : "standardowe wyjście"));  
    exit(2);  
}  
fclose(zfp);  
fclose(dofp);  
return 0;  
}  
  
void kopiuj(FILE *zfp,FILE *dofp)  
{  
    int c;  
  
    while ((c=getc(zfp)) != EOF)  
        putc(c,dofp);  
}
```

System plików - operacje na plikach binarnych

- W plikach są też przechowywane dane binarne np. pliki spakowane, obrazy, itp.

- Do odczytu danych binarnych używamy najczęściej operacji:

```
size_t fread(void *ptr, size_t size, size_t  
nmemb, FILE *stream)
```

- Do zapisu danych binarnych używamy najczęściej operacji:

```
size_t fwrite(const void *ptr, size_t size,  
size_t nmemb, FILE *stream)
```

- Odczytać pozycje w pliku binarnym pozwala funkcja: `long ftell(FILE *stream)` a powrócić do ustalonej pozycji: `int fseek(FILE *stream, long offset, int whence)`

Operacje na plikach binarnych - prosta baza danych I

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NAZWA_BAZY "karty.dat"

typedef struct karta {
    long numer;
    char imie[20];
    char nazwisko[30];
    double kwota;
} Karta;

typedef enum oper {nowa=1,zmiana,stan} Operacja;

FILE *otworz_baze(void);
int czytaj_karte(FILE *baza,long numer,Karta *wsk);
int zapisz_karte(FILE *baza,long numer,Karta *wsk);
```

Operacje na plikach binarnych - prosta baza danych II

```
int main(int argc, char *argv[])
{
    Karta pierwsza, aktualna;
    double zmiana;
    FILE *baza;

    if (argc == 1) {
        fprintf(stderr, "Uzycie: %s <operacja> <argumenty>\n", argv[0]);
        fputs("dozwolone operacje:\n", stderr);
        fputs("    -nowa <imie> <nazwisko> <kwota>\n", stderr);
        fputs("    -zmiana <nr-karty> <kwota>\n", stderr);
        fputs("    -stan <nr-karty>\n\n", stderr);
        exit(1);
    }
    if ((baza = otworz_baze()) == NULL) {
        fprintf(stderr, "%s: blad otwarcia bazy\n", argv[0]);
        exit(2);
    }
    czytaj_karte(baza, 0L, &pierwsza);
    if (strcmp(argv[1], "-nowa") == 0 && argc == 5) {
        aktualna.numer = ++pierwsza.numer;
        strncpy(aktualna.imie, argv[2], 20);
        strncpy(aktualna.nazwisko, argv[3], sizeof(aktualna.nazwisko));
        aktualna.kwota = atof(argv[4]);
        zapisz_karte(baza, 0L, &pierwsza);
        zapisz_karte(baza, aktualna.numer, &aktualna);
    } else
```

Operacje na plikach binarnych - prosta baza danych III

```
if (strcmp(argv[1], "-zmiana") == 0 && argc == 4) {
    aktualna.numer=atoi(argv[2]);
    zmiana=atof(argv[3]);
    if (0 < aktualna.numer && aktualna.numer <= pierwsza.numer) {
        czytaj_karte(baza, aktualna.numer, &aktualna);
        aktualna.kwota+=zmiana;
        zapisz_karte(baza, aktualna.numer, &aktualna);
    } else {
        fprintf(stderr, "%s: zly numer karty\n", argv[0]);
        exit(2);
    }
} else
if (strcmp(argv[1], "-stan") == 0 && argc == 3) {
    aktualna.numer=atoi(argv[2]);
    if (0 < aktualna.numer && aktualna.numer <= pierwsza.numer)
        czytaj_karte(baza, aktualna.numer, &aktualna);
    else {
        fprintf(stderr, "%s: zly numer karty\n", argv[0]);
        exit(2);
    }
} else {
    fprintf(stderr, "%s: zla operacja lub liczba jej argumentow\n", argv[0]);
    exit(2);
}

printf("Karta: %ld\nImie: %s\nNazwisko: %s\nKwota: %.2lf\n",
    aktualna.numer, aktualna.imie, aktualna.nazwisko, aktualna.kwota);
```

Operacje na plikach binarnych - prosta baza danych IV

```
fclose(baza);  
return 0 ;  
}  
  
FILE *otworz_baze(void)  
{  
    FILE *fp;  
    Karta pierwsza;  
  
    if ((fp=fopen(NAZWA_BAZY,"r+b")) == NULL) { /* baza nie istnieje */  
        fp=fopen(NAZWA_BAZY,"w+b");  
        pierwsza.numer=0L;  
        fwrite(&pierwsza,sizeof(Karta),1,fp);  
        fclose(fp);  
    }  
    return fopen(NAZWA_BAZY,"r+b");  
}
```

Operacje na plikach binarnych - prosta baza danych V

```
int czytaj_karte(FILE *baza, long numer, Karta *wsk)
{
    fseek(baza, numer*sizeof(Karta), SEEK_SET);
    return fread(wsk, sizeof(Karta), 1, baza);
}

int zapisz_karte(FILE *baza, long numer, Karta *wsk)
{
    fseek(baza, numer*sizeof(Karta), SEEK_SET);
    return fwrite(wsk, sizeof(Karta), 1, baza);
}
```

Prosta baza danych 2 - telefony brydżystów I

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define NAZWA_BAZY "osoby.dat"
#define PIERWSZA 27011

typedef struct osoba {
    int numer;
    char imie[20];
    char nazwisko[30];
    char telefon[20];
    struct {
        unsigned int plec : 1; // 0-mezczyzna, 1-kobieta
        unsigned int stan_cyw : 1; // 0-wolny, 1-zwiazany
        unsigned int bridge : 1; // 0-nie gra, 1-gra
    } cecha;
    union {
        char konwencja[20];
        int kiedy_chce_sie_nauczyc; // 0-nigdy, >0 -rok
    } bdana;
} Osoba;

typedef enum oper {nowy=1,zmiana,dane} Operacja;
```

Prosta baza danych 2 - telefony brydżystów II

```
FILE *otworz_baze(void);
unsigned int hash(char *napis);
int czytaj_osobe(FILE *baza, char *nazwisko, Osoba *wsk);
int zapisz_osobe(FILE *baza, Osoba *wsk);
int wczytaj_dane(Osoba *wsk);
void wypisz_dane(Osoba os);

int main(int argc, char *argv[])
{
    Osoba pierwsza, aktualna;
    char *nazwisko;
    FILE *baza;

    if (argc <= 2) {
        fprintf(stderr, "Uzycie: %s <operacja> <nazwisko>\n", argv[0]);
        fputs("dozwolone operacje:\n", stderr);
        fputs("        -nowy \n", stderr);
        fputs("        -zmiana \n", stderr);
        fputs("        -dane \n\n", stderr);
        exit(1);
    }
    if ((baza=otworz_baze()) == NULL) {
        fprintf(stderr, "%s: blad otwarcia bazy\n", argv[0]);
        exit(2);
    }
    nazwisko=argv[2];
    fread(&pierwsza, sizeof(Osoba), 1, baza);
```

Prosta baza danych 2 - telefony brydżystów III

```
if (strcmp(argv[1], "-nowy") == 0)
if (czytaj_osobe(baza, nazwisko, &aktualna) == 0) {
    fprintf(stderr, "%s: sa juz dane dla tego nazwiska\n", argv[0]);
    wypisz_dane(aktualna);
    exit(2);
} else {
    aktualna.numer = ++pierwsza.numer;
    strncpy(aktualna.nazwisko, nazwisko, sizeof(aktualna.nazwisko));
    if (wczytaj_dane(&aktualna) == 0) {
        fseek(baza, 0L, SEEK_SET);
        fwrite(&pierwsza, sizeof(Osoba), 1, baza);
        zapisz_osobe(baza, &aktualna);
    }
}
else
if (strcmp(argv[1], "-zmiana") == 0)
if (czytaj_osobe(baza, nazwisko, &aktualna) == 0) {
    wypisz_dane(aktualna);
    if (wczytaj_dane(&aktualna) == 0)
        zapisz_osobe(baza, &aktualna);
} else {
    fprintf(stderr, "%s: brak danych dla tego nazwiska\n", argv[0]);
    exit(2);
}
else
if (strcmp(argv[1], "-dane") == 0)
```


Prosta baza danych 2 - telefony brydżystów IV

```
if (czytaj_osobe(baza,nazwisko,&aktualna) == 0)
    wypisz_dane(aktualna);
else {
    fprintf(stderr,"%s: brak danych dla tego nazwiska\n",argv[0]);
    exit(2);
}
else {
    fprintf(stderr,"%s: zla operacja lub nazwisko\n",argv[0]);
    exit(2);
}
fclose(baza);
return 0 ;
}
```

Prosta baza danych 2 - telefony brydżystów V

```
FILE *otworz_baze(void)
{
    FILE *fp;
    Osoba pierwsza;
    int i;

    if ((fp=fopen(NAZWA_BAZY,"r+b")) == NULL) { /* baza nie istnieje */
        fp=fopen(NAZWA_BAZY,"w+b");
        pierwsza.numer=0;
        for (i=0; i <= PIERWSZA; ++i)
            fwrite(&pierwsza,sizeof(Osoba),1,fp);
        fclose(fp);
    }
    return fopen(NAZWA_BAZY,"r+b");
}
```

Prosta baza danych 2 - telefony brydżystów VI

```
unsigned hash(char *nazwa)
{
    char *p;
    unsigned long g,h=0;

    for (p=nazwa; *p != '\0'; ++p) {
        h=(h << 4) + *p;
        if ((g=(h & 0xF0000000)) != 0)
            h=h^(g | (g >> 24));
    }
    return h % PIERWSZA + 1;
}
```

Prosta baza danych 2 - telefony brydżystów VII

```
int czytaj_osobe(FILE *baza, char *nazwisko, Osoba *wsk)
{
    unsigned int i, h;

    i = h = hash(nazwisko);
    fseek(baza, h * sizeof(Osoba), SEEK_SET);
    do {
        if (fread(wsk, sizeof(Osoba), 1, baza) != 1)
            break;
        if (strcmp(wsk->nazwisko, nazwisko) == 0)
            return 0;
        i = (i % PIERWSZA) + 1;
        if (i == 1)
            fseek(baza, sizeof(Osoba), SEEK_SET);
    } while (wsk->numer > 0 && i != h);
    return 1;
}
```

Prosta baza danych 2 - telefony brydżystów VIII

```
int zapisz_osobe(FILE *baza, Osoba *wsk)
{
    unsigned int h,i;
    Osoba os;

    i=h=hash(wsk->nazwisko);
    fseek(baza,h*sizeof(Osoba),SEEK_SET);
    do {
        if (fread(&os,sizeof(Osoba),1,baza) != 1)
            break;
        if (os.number == 0 || strcmp(os.nazwisko,wsk->nazwisko) == 0) {
            fseek(baza,-1*sizeof(Osoba),SEEK_CUR);
            return (fwrite(wsk,sizeof(Osoba),1,baza)==1 ? 0 : 1);
        }
        i=(i % PIERWSZA)+1;
        if (i == 1)
            fseek(baza,sizeof(Osoba),SEEK_SET);
    } while (i != h);
    return 1;
}
```

Prosta baza danych 2 - telefony brydżystów IX

```
int wczytaj_dane(Osoba *wsk)
{
    char dana[100];

    printf("Numer: %d\n",wsk->numer);
    printf("Nazwisko: %s\n",wsk->nazwisko);
    printf("Imie: "); scanf("%s",wsk->imie);
    printf("Numer telefonu: "); scanf("%s",wsk->telefon);
    printf("Plec (k-kobieta,m-mezczyzna): "); scanf("%s",dana);
    wsk->cecha.plec=(tolower(dana[0])=='k' ? 1 : 0);
    printf("Stan cywilny (z-zajety,w-wolny): "); scanf("%s",dana);
    wsk->cecha.stan_cyw=(tolower(dana[0])=='z' ? 1 : 0);
    printf("Czy gra w bridge (t-tak,n-nie): "); scanf("%s",dana);
    wsk->cecha.bridge=(tolower(dana[0])=='t' ? 1 : 0);
    if (wsk->cecha.bridge) {
        printf("Uzywa konwencje: "); scanf("%s",wsk->bdana.konwencja);
    } else {
        printf("Chce sie nauczyc w roku (0-nie chce): ");
        scanf("%d",&wsk->bdana.kiedy_chce_sie_nauczyc);
    }
    return 0;
}
```

Prosta baza danych 2 - telefony brydżystów X

```
void wypisz_dane(Osoba os)
{
    printf("Numer: %d\n",os.numer);
    printf("Nazwisko: %s\n",os.nazwisko);
    printf("Imie: %s\n",os.imie);
    printf("Numer telefonu: %s\n",os.telefon);
    printf("Plec: %s\n",(os.cecha.plec ? "kobieta" : "mężczyzna"));
    printf("Stan cywilny: %s\n",(os.cecha.stan_cyw ? "zajęty" : "wolny"));
    printf("Czy gra w bridge: %s\n",(os.cecha.bridge ? "tak" : "nie"));
    if (os.cecha.bridge)
        printf("Używa konwencji: %s\n",os.bdana.konwencja);
    else
        printf("Chce się nauczyć w roku: %d\n",os.bdana.kiedy_chce_sie_nauczyc);
}
```