C++

Kopie zapasowe

Sławomir Białek

Wielu użytkowników komputerów przechowuje swoje dane i programy na dyskietkach. Niejednemu spośród nich zdarzyło się, że nie mógł odczytać z dyskietki kilku sektorów. Wówczas najczęściej nie można już uruchomić programu znajdującego się na takiej dyskietce.

Dobrym zabezpieczeniem przed takimi wypadkami jest wykonanie kopii zapasowych, jednak pochłania to o 100 proc. więcej dyskietek. Moje rozwiązanie pozwala zmniejszyć tę liczbę. Wykorzystuję w nim metodę strippingu stosowanego w macierzach dyskowych RAID.

Polega ona na generowaniu parzystości danych za pomocą operatora XOR. Na przykład, jeżeli chcemy zrobić kopię zapasową trzech dyskietek, należy wykonać operację XOR na odpowiednich bitach tych dyskietek i wynik zapisać na czwartej dyskietce - będzie ona żądaną kopią bezpieczeństwa.

	Dysk1	Dysk2	Dysk3	Dysk4	
Bit 1	0	1	1	0	(0 XOR 1 XOR 1)
Bit 2	1	0	0	1	(1 XOR 0 XOR 0)

Jeśli nie możemy odczytać któregoś z bitów, wykonujemy operację XOR na pozostałych trzech bitach (kolejność nie jest wymagana) i otrzymujemy czwarty, brakujący bit. Opisana metoda działa pod warunkiem, że „zginie" tylko jeden bit z czterech.

Opisane wyżej działanie możemy również wykonać na większej liczbie bitów, np. 10, ale wzrasta wtedy prawdopodobieństwo, że z tej jedenastki (10+1) „zginą" dwa lub więcej bitów.

Po uruchomieniu programu (w którym zastosowana została opisana metoda) do wyboru są dwie możliwości. W pierwszej po odczytaniu podanej liczby dyskietek (ich kolejność jest bez znaczenia) zostanie utworzona dyskietka zawierająca parzystość danych. Powinna ona być sformatowana tak samo, jak dyskietki, z których robiona jest kopia.

UWAGA: dyskietka taka wygląda tak, jakby była niesformatowana lub uszkodzona, należy ją zatem dobrze opisać, aby jej np. nie sformatować. Należy też zaznaczyć, z których dyskietek została utworzona, ponieważ podczas późniejszego odtwarzania muszą zostać odczytane te same dyskietki.

Opcji pierwszej można również użyć, gdy zginęła nam cała dyskietka. Po odczytaniu przez program pozostałych dyskietek i tej z parzystością danych, na następnej zostanie nagrana zawartość brakującej dyskietki.

W przypadku, gdy z którejś dyskietki nie można odczytać niektórych sektorów, należy wybrać drugą możliwość programu. W podanej liczbie dyskietek należy uwzględnić i tę, która zawiera parzystość danych. Jako pierwszą wkłada się dyskietkę przeznaczoną do naprawy, potem kolejność już nie jest ważna. Jej zawartość, ale z odzyskanymi straconymi sektorami, zostanie nagrana na końcu.

Program działa pod warunkiem, że nie nastąpi uszkodzenie innych sektorów leżących w tym samym miejscu (strona, ścieżka, sektor) na innych dyskietkach.

Program zakłada na dysku C, w katalogu głównym, plik tymczasowy o pojemności „obrabianej" dyskietki. Stąd do działania programu na dysku twardym potrzebne jest tyle wolnego miejsca, ile ma go dyskietka. □

```
#include <sys\stat.h>
#include <fcntl.h>
#include <bios.h>
#include <io.h>
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
int il_cyl,il_stron,sekt_sciez,rozm_sekt;
char *bufor1,*bufor2;
int bajt_sciez;
int sciezka,glowica,sektor;
int deskr;
int ile_dysk,naped;
struct pozycja {int sciezka; int strona; int sektor;};
enum { plik, dyskietka };
enum { czytanie=2, pisanie=3 };
int biosdisk_(int operacja, int ile, void *bufor)
{
 int blad,bis=0;
 do
 {
  bis++;
  blad = biosdisk(operacja, naped, glowica,
       sciezka, sektor, ile,bufor);
 } while ( (blad) && (bis != 5) );
 return (!blad);
}
int od_zap(int skad, int co, int ile, char *bufor)
{
 int status,off;
 off = (sektor-1) * rozm_sekt;
 if (skad==dyskietka)
    return (status=biosdisk_(co, ile, bufor+off));
 else if (skad==plik)
 {
  long poz =((long)sciezka * il_stron * sekt_sciez
 + (long)glowica * sekt_sciez + sektor)
 * rozm_sekt-rozm_sekt; //wyznaczenie pozycji w pliku
  lseek(deskr,poz,SEEK_SET);
  if (co==czytanie)
 status=read(deskr, bufor+off, ile * rozm_sekt);
  else if (co==pisanie)
 status=write(deskr, bufor+off, ile * rozm_sekt);
  return (status == -1 ? 0 : 1);
 }
}
void error(int nr)
{
 char *msg[] = { „Nieprawidłowy wybór",          // 0
       „Kłopoty z alokacją pamięci",   // 1
       „Kłopoty przy zapisie na HDD",  // 2
       „Kłopoty przy odczycie z HDD",  // 3
       „Kłopoty przy zapisie na FDD",  // 4
       „Kłopoty przy odczycie z FDD" }; // 5
 cout << endl << msg[nr];
 close(deskr);
 remove(„C:\\temp.$$$");
 exit(1);
}
void plik2dysk()  // skopiowanie zawartosci pliku
{       // tymczasowego na dyskietkę
 sektor=1;
 for (sciezka = 0; sciezka < il_cyl; sciezka++)
  for (glowica = 0; glowica < il_stron; glowica++)
   {
  if (!od_zap(plik,czytanie,sekt_sciez,bufor1))
          error(3);
   if (!od_zap(dyskietka,pisanie,sekt_sciez,bufor1))
          error(4);
   }
}
```

```
void kopia()
{
 cout << „\nWłóż pierwszą dyskietkę do"
      << „ stacji i naciśnij klawisz ENTER";
 while((getch()) != '\r');
 sektor=1;
 for (sciezka = 0; sciezka < il_cyl; sciezka++)
   for (glowica = 0; glowica < il_stron; glowica++)
    {
    if (!od_zap(dyskietka,czytanie,sekt_sciez,bufor1))
              error(5);
    if (!od_zap(plik,pisanie,sekt_sciez,bufor1))
              error(2);
    }
 for (int powt = 1; powt < ile_dysk; powt++)
 {
  cout << „\nWłóż " << powt+1 << „ dyskietkę do stacji"
 << „ i naciśnij klawisz ENTER.";
  while((getch()) !='\r');
  for (sciezka = 0; sciezka < il_cyl; sciezka++)
    for (glowica = 0; glowica < il_stron; glowica++)
     {
 if (!od_zap(dyskietka,czytanie,sekt_sciez,bufor1))
      error(5);
 if (!od_zap(plik,czytanie,sekt_sciez,bufor2))
      error(3);
 // tutaj następuje XOR-owanie
 for (int i = 0; i < bajt_sciez; i++)
      bufor1[i] ^= bufor2[i];
 if (!od_zap(plik,pisanie,sekt_sciez,bufor1))
      error(2);
     }
 }
 cout << „\nWłóż dyskietkę, na ktorą ma zostać nagrana"
      << „ parzystość danych";
 while((getch()) !='\r');
 plik2dysk();
}
void odzysk()
{
  cout << „\nWłóż dyskietkę, która ma"
 << „ zostać naprawiona i naciśnij klawisz ENTER";
 while((getch()) != '\r');
 int i,j;
 int licz_bl = 0;  // liczba nieodczytanych sektorów
 int rozm_bl = il_stron * il_cyl * sekt_sciez;
    //rozmiar tablicy z pozycjami błędnych sektorów
 pozycja *bledy;
 if (!(bledy = new pozycja[rozm_bl])) error(1);
 for(i = 0; i < rozm_bl; i++)  // wyczyszczenie tablicy
 {                 // z pozycjami błedów
  bledy[i].sciezka=0;bledy[i].strona=0;
   bledy[i].sektor =0;
 }
 for (sciezka = 0; sciezka < il_cyl; sciezka++)
   for (glowica = 0; glowica < il_stron; glowica++)
    {
     for (sektor = 1; sektor <= sekt_sciez; sektor++)
 if (!od_zap(dyskietka,czytanie,1,bufor1))
//    if (!biosdisk__(czytanie,1,bufor1+(sektor-1)*rozm_sekt))
  {
    licz_bl++;
   bledy[licz_bl-1].sciezka=sciezka;
   bledy[licz_bl-1].strona=glowica;
   bledy[licz_bl-1].sektor=sektor;
    for(i = 0; i < rozm_sekt; i++)
       bufor1[(sektor - 1) * rozm_sekt + i] = 0;
 }
     sektor=1;
   if (!od_zap(plik,pisanie,sekt_sciez,bufor1)) error(2);
    }
 for (int powt = 1; powt < ile_dysk; powt++)
 {
  cout << „\nWłóż " << powt+1 << „ dyskietkę do stacji"
 << „ i naciśnij klawisz ENTER.";
  while((getch()) !='\r');
  for(i=0; i<licz_bl; i++)
   {
    glowica=bledy[i].strona;
    sciezka=bledy[i].sciezka;
    sektor =bledy[i].sektor;
   if (!od_zap(dyskietka,czytanie,1,bufor1)) error(5);
   if (!od_zap(plik,czytanie,1,bufor2)) error(3);
    // XOR-owanie
    for (j=0; j < bajt_sciez; j++) bufor1[j] ^= bufor2[j];
   if (!od_zap(plik,pisanie,1,bufor1)) error(2);
   }
 }
 cout << „\nWłóż dyskietkę na której zostanie nagrana"
      << „ zawartość odzyskanej dyskietki i naciśnij"
      << „ klawisz ENTER";
 while((getch()) !='\r');
 plik2dysk();
 delete[rozm_bl] bledy;
}
int main()
{
 if ((deskr = open(„C:\\temp.$$$", O_CREAT | O_TRUNC |
      O_BINARY | O_RDWR , S_IWRITE)) == -1) error(2);
 clrscr();
 cout << „\n1.Tworzenie dyskietki z parzystością danych";
 cout << „\n2.Odtwarzanie uszkodzonej dyskietki";
 cout << „\nTwój wybór? ";
 int wybor;
 cin >> wybor;
 if (wybor<1 || wybor>2) error(0);
 cout << „\nKtóra stacja dysków A , B ?";
 char znak;
 cin >> znak;
 if (!isalpha(znak)) error(0);
 naped = znak - (isupper(znak) ? 'A' : 'a');
 cout << „\nPodaj rodzaj dyskietki: 1-360, 2-720, 3-1.2,"
      << „ 4-1.44 ?";
 cin >> znak;
 il_stron=2;
 rozm_sekt=512;
 switch(znak)
 { case '1': il_cyl = 40; sekt_sciez = 9;  break;
   case '2': il_cyl = 80; sekt_sciez = 9;  break;
   case '3': il_cyl = 80; sekt_sciez = 15; break;
   case '4': il_cyl = 80; sekt_sciez = 18; break;
    default: error(0);                     break;
 }
 cout << „\nIle dyskietek?";
 cin >> ile_dysk;
 if (ile_dysk<1) error(0);
 bajt_sciez = rozm_sekt*sekt_sciez;
 if (!(bufor1 = new char[bajt_sciez])) error(1);
 if (!(bufor2 = new char[bajt_sciez])) error(1);
 switch(wybor)
 { case 1: kopia();  break;
   case 2: odzysk(); break;
 }
 delete[bajt_sciez] bufor1;
 delete[bajt_sciez] bufor2;
 close(deskr);
 remove(„C:\\temp.$$$");
 return 0;
}
```