9. hét

Az óra témája

Az óra célja, hogy a hallgatók megismerjék az alacsony szintű fájlkezelés módszereit, képesek legyenek egyszerű írási és olvasási feladatok megvalósítására alacsony szintű fájlkezelés segítségével.

Mintafeladat

A következő program bemutatja, teljes ASCII karaktertábla kiíratását és visszaolvasását egy a program által létrehozott fájlból.

```
#include <conio.h>
#include <stdio.h>

#include <io.h>
#include <fcntl.h>
#include <sys/stat.h>

int mywrite ( void );
int myread ( void );

const char fnev[] = "adat.dat";

// ------------------------------------------------------------------
int main ( void )
// ------------------------------------------------------------------
{
    mywrite();
    myread();
}

// ------------------------------------------------------------------
int mywrite ( void )
// ------------------------------------------------------------------
{
int fhandler;
int f;
unsigned char i = 0;

    fhandler = open ( fnev, O_RDWR | O_CREAT |
                O_TRUNC | O_BINARY, S_IWRITE );

    if ( fhandler < 0 )
    {
        printf ( "File hiba: %s\n", fnev );
        return -1;
    }

    for ( f=0; f<=255; f++ )
```

```
    {
        write ( fhandler, &i, sizeof(i));
        i++;
    }

    close ( fhandler );
}

// ------------------------------------------------------------
int myread( void )
// ------------------------------------------------------------
{
int fhandler;
long hossz;
char buf[1024];
int i, f;

    fhandler = open ( fnev, O_RDONLY |
                O_BINARY, S_IREAD );

    hossz = lseek ( fhandler, 0, SEEK_END );  // Ugrás a file végére,
    printf ( "Hossz: %u\n", hossz );          // hogy megállapítsuk a hosszát
    lseek ( fhandler, 0, SEEK_SET );          // Vissza a file elejére
                                              // az olvasásoz
    f = read ( fhandler, buf, 1024 );         // 1024 byte-ot próbál olvasni,
    printf ( "%u byte beolvasva.\n", f );     // de csak annyit fog, amilyen
                                              // hosszu a fájl

    buf[0] = 32;                              // A [0]-dik pozición 0 volt,
                                              // átírjuk Space-re
    buf[f] = '\0';                            // 'End of string' a végére,
                                              // hogy lezárja a sztringet
    printf ( ". . . . . .\n" );
    printf ( "%s\n", buf );                   // Kiírja sztringként
                                              // a char tömböt.
    close ( fhandler );
}
```

Mintafeladat

A következő program példát mutat több integer egyenkénti kiíratására, valamint egy sztring és egy int tömb egylépésbeni kiiratására, alacsonyszintű fájlkezelés segítségével. A visszaolvasó függvény valamennyi adatot vosszaolvassa és kiírja a képernyőre.

```
#include <conio.h>
#include <stdio.h>

#include <io.h>
#include <fcntl.h>
#include <sys/stat.h>
// #include <conio.h>
```

```
void mywrite ( void );
void myread ( void );

const char fnev[] = "adat.dat";
const char sep[] = { 9 };


int main ( void )
{

    // clrscr();
    mywrite();
    myread();
    getch();

}

void mywrite ( void )
{
int fhandler;
const unsigned char szoveg[] = "Probaszoveg a file-ban.";
unsigned int itomb[10];

int i, f;

    printf ( ">>> Trace: void mywrite ( void )\n" );

    fhandler = open ( fnev, O_RDWR | O_CREAT |
                O_TRUNC | O_BINARY, S_IWRITE );

    if ( fhandler < 0 )
    {
        printf ( "File hiba: %s\n", fnev );
        return;
    }

    for ( i=0; i<256; i++ )
    {
        write ( fhandler, &i, sizeof(i));
        // write ( fhandler, sep, 1 );
    }
    printf ( "0-255, int kiirva.\n" );

    i = write ( fhandler, szoveg, sizeof(szoveg));
    printf ( "%u byte kiirva.\n", i );

    for ( i=0; i<10; i++ ) itomb[i]=65;
    i = write ( fhandler, itomb, sizeof(itomb));
    printf ( "%u byte kiirva.\n", i );

    printf ( "Kesz.\n" );
    close ( fhandler );
}


void myread( void )
```

```
{
int fhandler;
long hossz;
char buf[1024];
int i, f;

    printf ( ">>> Trace: void myread ( void )\n" );

    fhandler = open ( fnev, O_RDONLY |
                O_BINARY, S_IREAD );

    hossz = lseek ( fhandler, 0, SEEK_END );
    printf ( "Hossz: %u\n", hossz );

    i = tell ( fhandler );
    printf ( "File kurzor: %u\n", i );

    lseek ( fhandler, 0, SEEK_SET );
    i = tell ( fhandler );
    printf ( "File kurzor: %u\n", i );

    f = read ( fhandler, buf, 1024 );
    if ( f<0 )
    {
        printf ( "Olvasasi hiba.\n" );
        return;
    } else
    {
        printf ( "%u byte beolvasva.\n", f );
    }

    for ( i=0; i<f; i++)
        if ( buf[i]==0 ) buf[i]=32;
    buf[f] = 0;

    printf ( "A file tartalma: \n%s\n", buf );

    close ( fhandler );

}
```

Magyarázat

A program a fájlkezelés mellett bemutatja a trace-elés nagyon egyszerű formáját, valamennyi függvény hívásakor kiírja a nevét és a paraméterlistáját a képernyőre.