

Listing: FLOCKC.C

```

/*****
/*
/*          F L O C K C . C
/*
/*-----*/
/*   Task           : Demonstration of the file locking functions on a
/*                   : network. Uses the module NETFILEC.
/*-----*/
/*   Memory model   : SMALL
/*-----*/
/*   Author          : Michael Tischer
/*   developed on    : 02/10/1992
/*   last Update     : 04/07/1995
*****/

```

```

#include <stdio.h>
#include "netfilec.c"                                /* include network routine*/

/*== Constants =====*/

#define TFileName "flockc.dat"                       /* filename for test file*/

/*== Type definitions =====*/

typedef char Test[ 5 ];                               /* Test for file*/
typedef unsigned char BYTE;

/*== display routines for Microsoft C =====*/

#ifndef __TURBOC__                                    /* Microsoft C?*/

#define clrscr() clearwindow( 1, 1, 80, 25 )

```

```

/*****
/* Gotoxy          : position the Cursor                               */
/* Input           : coordinates of the Cursor                       */
/* Output          : none                                             */
*****/

void gotoxy( int x, int y )
{
    regs.h.ah = 0x02;          /* Function number for the Interrupt call*/
    regs.h.bh = 0;              /* screen page*/
    regs.h.dh = y - 1;
    regs.h.dl = x - 1;
    int86( 0x10, &regs, &regs );          /* Call Interrupt*/
}
#endif

/*****
/* clearwindow     : clear a portion of the screen                   */
/* Input           : s.u.                                           */
/* Output          : none                                             */
*****/

void clearwindow( int x1, int y1, int x2, int y2 )
{
    regs.h.ah = 0x07;          /* Function number for the Interrupt call*/
    regs.h.al = 0x00;
    regs.h.bh = 0x07;
    regs.h.ch = y1 - 1;
    regs.h.cl = x1 - 1;
    regs.h.dh = y2 - 1;
    regs.h.dl = x2 - 1;
    int86( 0x10, &regs, &regs );          /* Call Interrupt*/
}

```

```

gotoxy( x1, y1 );                                /* position cursor*/
}

/*****
/* FiMode      : Create file mode from access type and locking      */
/* Input       : s.u.                                                */
/* Output      : file mode                                           */
*****/

int FiMode( int AxsType,                          /* access type of the file*/
            int LockMd )                          /* locking mode of the file*/
{
    static BYTE AxsType_Ary[ 3 ] = { FM_R, FM_W, FM_RW };
    static BYTE LockMd_Ary[ 5 ]  = { SM_COMP, SM_RW, SM_R,
                                     SM_W, SM_NO };

    return AxsType_Ary[ AxsType-1 ] | LockMd_Ary[ LockMd-1 ];
}

/*****
/* DFileTest    : Demonstrates access conflicts or file locks with   */
/*                and without file locking.                          */
/* Input         : s.u.                                                */
/* Output        : none                                               */
*****/

void DFileTest( int AxsTypeA,                      /* access type - file A*/
                int LockMdA,                      /* lock mode - file A*/
                int AxsTypeB,                      /* access type - file B*/
                int LockMdB )                      /* lock mode - file B*/
{
    Test  TestAOut = "AAAA\0";                    /* Test data records*/

```

```

Test  TestBOut = "BBBB\0";

Test  TestAInp;
Test  TestBInp;
NFILE TFileA;          /* Data records for read test*/
NFILE TFileB;          /* Test files for normal access*/
char  SDummy[ 50 ];    /* Status of the network function*/

clearwindow( 1, 11, 80, 25 );
printf( "File A: Name = %s Access type = %2i Lock mode = %2i\n",
        TFileName, AxsTypeA, LockMdB );
printf( "File B: Name = %s Access type = %2i Lock mode = %2i\n\n",
        TFileName, AxsTypeB, LockMdB );

/*-- Open files -----*/

printf( "Opening file A:  " );
NetReset( TFileName, FiMode( AxsTypeA, LockMdB ),
          sizeof( Test ), &TFileA );
if ( NetError == NE_FileNotFound )
    NetRewrite( TFileName, FiMode( AxsTypeA, LockMdB ),
               sizeof( Test ), &TFileA );
NetErrorMsg( NetError, SDummy );
printf( "Status %2u = %s\n", NetError, SDummy );

printf( "Opening file B:  " );
NetReset( TFileName, FiMode( AxsTypeB, LockMdB ),
          sizeof( Test ), &TFileB );
NetErrorMsg( NetError, SDummy );
printf( "Status %2u = %s\n\n", NetError, SDummy );

/*-- Write files -----*/

```

```

printf( "Writing to file A:" );
if ( Is_NetWriteOk( &TFileA ) )                                /* write enabled?*/
{
    NetWrite( &TFileA, TestAOut );
    printf( " Record '%s' written\n", TestAOut );
}
else
    printf( " File not open for writing\n" );

printf( "Writing to file B:" );
if ( Is_NetWriteOk( &TFileB ) )                                /* write enabled?*/
{
    NetWrite( &TFileB, TestBOut );
    printf( " Record '%s' written\n\n", TestBOut );
}
else
    printf( " File not open for writing\n\n" );

/*-- File pointers for both file moved to beginning -----*/

if ( Is_NetOpen( &TFileA ) )                                    /* file open?*/
    NetSeek( &TFileA, 0L );
if ( Is_NetOpen( &TFileB ) )                                    /* file open?*/
    NetSeek( &TFileB, 0L );

/*-- Read files -----*/

printf( "Reading file A:" );
if ( Is_NetReadOk( &TFileA ) )                                  /* read enabled?*/
{
    NetRead( &TFileA, TestAInp );

```

```

    printf( " Record '%s' read\n", TestAInp );
}
else
    printf( " File not open for reading\n" );

printf( "Reading file B:" );
if ( Is_NetReadOk( &TFileB ) )                /* read enabled?*/
{
    NetRead( &TFileB, TestBInp );
    printf( " Record '%s' read\n\n", TestBInp );
}
else
    printf( " File not open for reading\n\n" );

/*-- Close files -----*/

NetClose( &TFileA );
NetClose( &TFileB );
}

/*****
/*          M A I N   P R O G R A M          */
*****/

void main( void )
{
    int AxsTypeA;                /* file access modes*/
    int AxsTypeB;
    int LockMdA;                /* file lock modes*/
    int LockMdB;

    clrscr();

```

```

gotoxy( 1, 1 );
printf( "Demonstration of DOS File Locking Functions          " \
        "(C) 1992 by Michael Tischer\n" );
printf( "===== " \
        "=====\\n\\n" );

if ( ShareInst() )                                /* SHARE installed?*/
{
/*-- Select file mode -----*/

printf( "Available access types:          Available lock types:\\n" );
printf( " 1: Read-only                      " ),
printf( " 1: Compatibility mode (no locking)\\n" );
printf( " 2: Write-only                        " );
printf( " 2: Prohibit other file accesses generally\\n" );
printf( " 3: Read and Write                      " );
printf( " 3: Read access enabled only   \\n" );
printf( "                                " );
printf( " 4: Write access enabled only  \\n" );
printf( "                                " );
printf( " 5: All enabled (record locking)\\n" );

printf( "\\nAccess type Test file A: " );
scanf( "%i", &AxsTypeA );
printf( "Lock Mode Test file A: " );
scanf( "%i", &LockMdA );
printf( "Access type Test file B: " );
scanf( "%i", &AxsTypeB );
printf( "Lock mode Test file B: " );
scanf( "%i", &LockMdB );

DFileTest( AxsTypeA, LockMdA, AxsTypeB, LockMdB );

```

```
}  
else  
    printf( "\nPlease install SHARE before running this program.\n" );  
}
```