

C listing: DIRC2.C

```

/*****
/*
/*          D I R C 2
/*
/*-----*/
/*   Task           : Displays all files in any directory on the   */
/*                   screen, including subdirectories and volume     */
/*                   label names. File handling is performed by    */
/*                   predefined C functions.                        */
/*                   See also DIR1C.C.                             */
/*-----*/
/*   Author          : Michael Tischer                            */
/*   Developed on    : 10/15/91                                     */
/*   Last update     : 04/07/95                                     */
/*-----*/
/*   Memory model    : SMALL                                       */
*****/

```

```

/*== Add include files =====*/

```

```

#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <stdio.h>
#include <conio.h>

```

```

#ifdef __TURBOC__
    /* Turbo C compiler? */
    #include <dir.h>      /* Then add directory include */
#endif

```

```

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

```

```

typedef unsigned char BYTE;                                /* Create a byte */

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

#define TRUE      ( 0 == 0 )                               /* Constants make reading */
#define FALSE     ( 0 == 1 )                               /* program code easier   */

#define FENTS     14                                       /* Number of visible entries at a time */
#define OT        (20-FENTS >> 1)                         /* Top row of output window */
#define NOF       0x07                                     /* White text on black background */
#define INV       0x70                                     /* Black text on white background (inverse) */

#define FA_ALL    0x3F                                     /* All file types */

/*== Macros =====*/

#ifdef MK_FP                                               /* Macro MK_FP already defined? */
    #undef MK_FP                                           /* Yes --> Undefine macro */
#endif

#define MK_FP(seg,ofs) ((void far *) ((unsigned long) (seg)<<16|(ofs)))

/*-- Compiler-specific constants and macros for file searching -----*/
/*-- (for Microsoft C and Borland compilers) -----*/

#ifdef __TURBOC__                                         /* Turbo C compiler? */
    #define DIRSTRUCT          struct ffbldk
    #define FINDFIRST( path, buf, attr )    findfirst( path, buf, attr )
    #define FINDNEXT( buf )              findnext( buf )
    #define NAME                ff_name
    #define ATTRIBUTE            ff_attrib
    #define TIME                ff_ftime

```

```

#define DATE                ff_fdate
#define SIZE                ff_fsize
#else                       /* No --> Microsoft C */
#define DIRSTRUCT           struct find_t
#define FINDFIRST( path, buf, attr ) _dos_findfirst(path, attr, buf)
#define FINDNEXT( buf )     _dos_findnext( buf )
#define NAME               name
#define ATTRIBUTE          attrib
#define TIME               wr_time
#define DATE               wr_date
#define SIZE               size
#endif

/*****
/* PRINT : Similar to PRINTF, writes the string directly to          */
/*          video RAM.                                              */
/* Input   : COLUMN = Display column                                */
/*          SCROW   = Display row                                    */
/*          DCOLR   = Display color                                  */
/*          STRING  = Pointer to PRINTF string                      */
/*          ...     = other arguments                               */
/* Output  : None                                                  */
*****/

void Print( int Column, int ScRow, BYTE DColr, char * String, ...)
{
    struct vr {
        BYTE DisChar,                /* 2 bytes to a screen position */
        Attribute;                  /* ASCII code */
    } far * lptr;                  /* Corresponding attribute */
    va_list parameter;             /* Floating pointer to video RAM access */
    char Output[255],              /* Parameter list for VA... macros */
    /* Buffer for formatted string */

```

```

        *aptr = Output;                                /* For string execution */
static unsigned int vioseg = 0;
union REGS Register;    /* Register variables for interrupt call */

if ( vioseg == 0 )                                /* First call? */
{
    /* Yes --> Get segment address of video RAM */
    Register.h.ah = 0x0F;
    int86(0x10, &Register, &Register);
    vioseg = ( Register.h.al == 7 ? 0xb000 : 0xb800 );
}

va_start( parameter, String );                    /* Convert parameter */
vsprintf( Output, String, parameter );            /* Format */
lptr = (struct vr far *)
    MK_FP( vioseg, ( ScRow * 80 + Column ) << 1 );

for ( ; *aptr ; )                                /* Run string */
{
    lptr->DisChar = *aptr++;                        /* Character in video RAM */
    lptr++->Attribute = DColr;                      /* Set character attribute */
}

}

/*****
/* SCROLLUP: Moves a screen range up by one or more rows, or clears */
/*      rows. */
/* Input   : NUMROW    = Number of rows to be scrolled */
/*           DCOLR     = Color/attribute for blank lines */
/*           COLUMNUL  = Upper-left screen corner (column) */
/*           SCROWUL   = Upper-left screen corner (row) */
/*           COLUMNLR  = Lower-right screen corner (column) */
/*           SCROWLR   = Lower-right screen corner (row) */
*****/

```

```

/* Output   : None                                     */
/* Info     : If NumRow contains 0, the screen range fills with */
/*           blank spaces.                               */
/*****

```

```

void ScrollUp( BYTE NumRow, BYTE DColr, BYTE ColumnOL,
               BYTE ScRowOL, BYTE ColumnUR, BYTE ScRowUR)

```

```

{
    union REGS Register;          /* Register variables for interrupt call */

    Register.h.ah = 6;             /* Function number */
    Register.h.al = NumRow;        /* Number of rows */
    Register.h.bh = DColr;         /* Color or blank lines) */
    Register.h.ch = ScRowOL;       /* Coordinates for */
    Register.h.cl = ColumnOL;      /* scrolling or clearing */
    Register.h.dh = ScRowUR;       /* Specify screen window */
    Register.h.dl = ColumnUR;
    int86(0x10, &Register, &Register); /* Call interrupt 10H */
}

```

```

/*****
/* SETPOS: Specifies a cursor position in the current screen page. */
/* Input   : COLUMN = New cursor column */
/*          SCROW  = New cursor row */
/* Output  : None */
/*****

```

```

void SetPos( BYTE Column, BYTE ScRow)

```

```

{
    union REGS Register;          /* Register variables for interrupt call */

    Register.h.ah = 2;             /* Function number */

```

```

Register.h.bh = 0;                                /* Screen page */
Register.h.dh = ScRow;                            /* Screen row */
Register.h.dl = Column;                          /* Screen column */
int86(0x10, &Register, &Register);              /* Call interrupt 10H */
}

```

```

/*****
/* CLS: Clears current screen page and places cursor in upper-left */
/*      corner.                                                    */
/* Input   : None                                                  */
/* Output  : None                                                  */
*****/

```

```
void Cls( void )
```

```

{
    ScrollUp(0, NOF, 0, 0, 79, 24);                /* Clear screen */
    SetPos(0, 0);                                  /* Place cursor */
}

```

```

/*****
/* SCREENDESIGN : Prepares screen for directory display.          */
/* Input   : None                                                  */
/* Output  : None                                                  */
*****/

```

```
void ScreenDesign( void )
```

```

{
    BYTE i;                                         /* Loop counter */

    Cls();                                         /* Clear screen */
    Print( 14, OT, NOF,

```

```

        "+-----+");
Print( 14, OT+1, NOF,
      "|  Filename  | Size  |      Date      | Time  | RHSVD |");
Print( 14, OT+2, NOF,
      "|-----+-----+-----+-----+-----|");
for (i = OT+3; i < OT+3+FENTS; i++)
    Print( 14, i, NOF,
          "|          |          |          |          |");
Print( 14, OT+FENTS+3, NOF,
      "+-----+");
}

/*****
/* PRINTDATA: Displays entry information.
/* Input   : P2ENTRY = Pointer to a DirStruct with file information
/*          SCROW   = Screen row of entry
/* Output  : None
*****/

void PrintData(DIRSTRUCT *P2Entry, BYTE ScRow)
{
    BYTE i, j;
    static char *Month[] =          /* Loop counter */
                                /* Vector with pointers to month name */
    {
        "Jan", "Feb", "Mar", "Apr", "May", "Jun",
        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
    };

    /*-- Display miscellaneous file information -----*/

    Print(15, ScRow, NOF, "%s", P2Entry->NAME);

```

```

Print( 28, ScRow, NOF, "%7lu", P2Entry->SIZE);
Print( 36, ScRow, NOF, "%s %2d %4d", Month[(P2Entry->DATE >> 5 & 15) - 1],
      P2Entry->DATE & 31,
      (P2Entry->DATE >> 9) + 1980);
Print(51, ScRow, NOF, "%02d:%02d", P2Entry->TIME >> 11,
      P2Entry->TIME >> 5 & 63 );

for (i = j = 1; i <= 16; i <= 1, ++j) /* Display file attributes */
  Print( 58+j, ScRow, NOF, "%c", (P2Entry->ATTRIBUTE & i) ? 'X' : ' ');
}

/*****
/* DIR: Controls directory reading and output.
/* Input   : SPath      = Pointer to search path with file pattern
/*          : Attribute = Search attribute
/* Output  : None
*****/

void Dir( char *SPath, BYTE Attribute )
{
  int      NumOfEntries,          /* Total number of entries found */
          NumInScrn;             /* Number of entries per screen */
  DIRSTRUCT P2Entry;             /* Pointer to a directory entry */

  ScreenDesign();                /* Prepare screen for directory display */

  NumInScrn = NumOfEntries = 0;   /* No more entries to display */
                                  /* No more entries found */
  if ( !FINDFIRST(SPath, &P2Entry, Attribute) ) /* Search: 1st entry */
  {
    /* One file found */
    do
      /* Display file and use GetNext() to find next one */
    {

```



```

PrintData(&P2Entry, OT+FENTS+2);          /* Display entries */
if (++NumInScrn == FENTS )                /* Is the window full? */
{
    NumInScrn = 0;                          /* Fill the window */
    Print(14, OT+4+FENTS, INV,
        "                Please press a key                ");
    getch();                                /* Wait for a key */
    Print(14, OT+4+FENTS, NOF,
        "                ");
}
ScrollUp(1, NOF, 15, OT+3, 63, OT+2+FENTS);
Print(15, OT+2+FENTS, NOF,
    "                |                |                |                ");
++NumOfEntries;
}
while ( !FINDNEXT( &P2Entry ) );          /* End loop when no */
}                                           /* more files exist */

SetPos(14, OT+4+FENTS);
switch (NumOfEntries)
{
    case 0 : printf("No files found");
              break;
    case 1 : printf("One file found");
              break;
    default : printf("%d files found", NumOfEntries);
              break;
}
}

/*****
MAIN PROGRAM
*****/

```

```
/* **** */
```

```
void main( int NumRow, char *Argumente[] )
{
    switch ( NumRow )
    {
        case 1 : Dir( "*.*", FA_ALL ); /* React to parameters */
            break; /* as necessary */
        case 2 : Dir( Argumente[1], FA_ALL ); /* Display all files in */
            break; /* current directory */
        default : printf("Invalid number of parameters\n"); /* specified directory */
    }
}
```