

C listing: DUMPC.C

```

/*****
/*
/*-----
/* Task : A filter, which reads in characters from the
/* standard input device and outputs them as a
/* hex and ASCII dump on the standard output device
/*-----
/* Author : MICHAEL TISCHER
/* Developed on : 08/14/87
/* Last update : 04/07/95
*****/
#include <stdio.h> /* Include header files */
#include <dos.h>

/*== Type definitions =====*/
typedef unsigned char BYTE; /* Handle as a byte */

/*== Constants =====*/
#define NUL 0 /* ASCII code for NULL character */
#define BEL 7 /* ASCII code for Bell */
#define BS 8 /* ASCII code for Backspace */
#define TAB 9 /* ASCII code for Tab */
#define LF 10 /* ASCII code for Linefeed */
#define CR 13 /* ASCII code for Carriage Return */
#define ESC 27 /* ASCII code for Escape */

/*== Macros =====*/
#define tohex(c) ( ((c)<10) ? ((c) | 48) : ((c) + 'A' - 10) )

/*****
/* GETSTDIN: Reads a certain number of characters from the standard */

```

```

/*          input device and places them in a buffer          */
/* Input    : BUFFER = Pointer to buffer receiving characters */
/*          MAXCHAR = Maximum number of characters read at a time */
/* Output   : Number of characters read                        */
/*****
int GetStdIn(char *Buffer, int MaxChar)
{
    union REGS Register;          /* Register variable for interrupt call */
    struct SREGS Segment;         /* Accepts the segment register */

    segread(&Segment);            /* Read contents of segment register */
    Register.h.ah = 0x3F;          /* Function number */
    Register.x.bx = 0;             /* Standard input device is handle 0 */
    Register.x.cx = MaxChar;       /* Number of bytes to be read */
    Register.x.dx = (unsigned int) Buffer; /* Offset address of buffer */
    intdosx(&Register, &Register, &Segment); /* Call Interrupt 21H */
    return(Register.x.ax);         /* Number of bytes read to caller */
}

/*****
/* STRAP : Attach character to string */
/* Input : STRING = Pointer to string to be appended */
/*          TEXTPOINTER = Pointer to string with additional text */
/* Output : Pointer behind the last added character */
/*****
char *Strap(char *String, char *Textpointer)
{
    while (*Textpointer)           /* Repeat until '\0' detected */
        *String++ = *Textpointer++; /* Transmit character */
    return(String);                /* Pass pointer to calling function */
}

```

```

/*****
/* DODUMP : Reads the characters in and outputs them as dump          */
/* Input   : None                                                    */
/* Output  : None                                                    */
*****/

```

```

void DoDump( void )

```

```

{
    char NineBytes[9],                /* Accepts the characters read */
        DumpBuf[80],                /* Accepts a line of DUMP */
        *NextAscii; /* Points to next ASCII character in the buffer */
    BYTE i,                          /* Loop counter */
        Readbytes; /* Number of bytes read */

```

```

    DumpBuf[30] = 219; /* Set separator between hex and ASCII */
    while((Readbytes = GetStdIn(NineBytes, 9)) != 0)
        /* as long as characters are available */

```

```

    {
        for (i = 0; i < 30; DumpBuf[i++] = ' ');
        /* Fill buffer with spaces */
        NextAscii = &DumpBuf[31]; /* ASCII characters start here */
        for (i = 0; i < Readbytes; i++)
            /* Process all characters read */

```

```

    {
        DumpBuf[i*3] = tohex((BYTE) NineBytes[i] >> 4);
        /* Convert code to hex */
        DumpBuf[i*3+1] = tohex((BYTE) NineBytes[i] & 15);
        switch (NineBytes[i]) /* Read ASCII code */
        {
            case NUL : NextAscii = Strap(NextAscii, "<NUL>");
                        break;
            case BEL : NextAscii = Strap(NextAscii, "<BEL>");

```

```

        break;
    case BS : NextAscii = Strap(NextAscii, "<BS>");
        break;
    case TAB : NextAscii = Strap(NextAscii, "<TAB>");
        break;
    case LF : NextAscii = Strap(NextAscii, "<LF>");
        break;
    case CR : NextAscii = Strap(NextAscii, "<CR>");
        break;
    case ESC : NextAscii = Strap(NextAscii, "<ESC>");
        break;
    case EOF : NextAscii = Strap(NextAscii, "<EOF>");
        break;
    default : *NextAscii++ = NineBytes[i];
}
}
*NextAscii = 219; /* End character for ASCII representation */
*(NextAscii+1) = '\r'; /* Carriage Return to end of buffer */
*(NextAscii+2) = '\0'; /* NULL converted to LF on output */
puts(DumpBuf); /* Write string to standard output device */
}
}

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

void main()
{
    DoDump(); /* Character input/output */
}

```

