

```
/* Source code for FNDFILE3.C */
```

```
#define      INCL_BASE
#define      INCL_ERRORS
#define      INCL_DOS
#define      INCL_DOSMISC
#define      INCL_DOSMODULEMGR

#include     <stdio.h>
#include     <stdlib.h>
#include     <os2.h>
#include     <dos.h>
#include     <string.h>

#define      FAIL_BUF_LEN 128
PSZ  mod_name[] = {"doscalls", "fndfile2"};
PSZ  routine_name[] = {"#64", "DO_LIST"};

USHORT      (pascal far *routine)(PSZ, PHDIR, USHORT, PFILEFINDBUF, USHORT,
PUSHORT, ULONG);

USHORT      far      _loadds pascal do_find(PSZ name_ptr, PHDIR
                                             hptr, USHORT attrb,
                                             PFILEFINDBUF buf_ptr,
                                             USHORT buf_len,
                                             PUSHORT num_ptr,
                                             ULONG reserved)

{
USHORT      stat;

    if(!(stat = routine(name_ptr, hptr, attrb, buf_ptr,
                        buf_len, num_ptr, reserved)))
    {
        printf("Good return. Files found = %d\n", *num_ptr);
    }
    else
    {
        switch(stat)
        {
            case  ERROR_BUFFER_OVERFLOW:
                printf("Buffer Overflow - Increase\ Buffer Size\n");
                break;

            case  ERROR_DRIVE_LOCKED:
                printf("Drive Locked\n");
                break;
        }
    }
}
```

```

        case ERROR_FILE_NOT_FOUND:
            printf("File: %s not found\n",\ name_ptr);
            break;

        case ERROR_INVALID_HANDLE:
            printf("Invalid handle: %d\n", *hptr);
            break;

        case ERROR_INVALID_PARAMETER:
            printf("Invalid Parameter\n");
            break;

        case ERROR_NO_MORE_FILES:
            printf("Ran out of files\n");
            break;

        case ERROR_NO_MORE_SEARCH_HANDLES:
            printf("Can;t allocate another Search\ Handle\n");
            break;

        case ERROR_NOT_DOS_DISK:
            printf("Not a DOS Disk\n");
            break;

        case ERROR_PATH_NOT_FOUND:
            printf("I can't locate that Path\n");
            break;

        default:
            printf("Unkown error in FindFirst:\ %d\n", stat);
            break;
    }
}
return(stat);
}

```

```

far _loadds pascal debug(USHORT debug_flag)
{
    USHORT      stat;
    CHAR        fail_buf[FAIL_BUF_LEN];
    static HMODULE  handle = 0;
    HMODULE     tmp_handle;
    CHAR        tmp_buf[128];

    printf("Debug is: %s\n", debug_flag ? "On" : "Off");

```

```

/* already a DLL loaded? */

if(handle)
{
    /* some DLL already loaded. Requested DLL? */

    stat = DosGetModHandle(mod_name[debug_flag],
                          &tmp_handle);

    /* if error, or a handle mismatch, then it isn't
    * the requested DLL */

    if(stat || tmp_handle != handle)
    {
        /* Get name of the DLL currently loaded */

        if((stat = DosGetModName(handle, 128,
                                tmp_buf)))
        {
            printf("Couldn't retrieve loaded DLL\ Name. Error code
is: %d\n", stat);
            return(FALSE);
        }
        else
            printf("Currently Loaded DLL is: %s\n",
                  tmp_buf);

        /* free the already loaded module, whatever
        * it is */

        DosFreeModule(handle);
    }
    else
    {
        /* current handle is for requested DLL.
        * Simply return */
        printf("DLL (%s) already loaded\n",
              mod_name[debug_flag]);
        return(TRUE);
    }
}
}

```

```

/* wrong DLL is now freed */
/* try to load the requested DLL, and get the entry
 * points */

if(stat = DosLoadModule(fail_buf, FAIL_BUF_LEN,
                        mod_name[debug_flag],
                        &handle))

{
    printf("Couldn't load: %s (stat is :%x)\n",
           mod_name[debug_flag], stat);
    printf("DLL problem was in: %s\n", fail_buf);
    return(FALSE);
}
else
    printf("Module handle is: %d\n", handle);

/* Now get the entry point for the requested routine */

if    (stat = DosGetProcAddress(handle,
                                routine_name[debug_flag], &routine))
{
    printf("Couldn't get routine: %s (stat is :%d)\n",
           routine_name[debug_flag], stat);
    return(FALSE);
}
else
    printf("Routine address is: %lx\n", routine);

/* module loaded, entry point returned, so we return */
return(TRUE);
}

```