Testing the Syscalls

To test the syscalls you can use the following code:

```
char str[80], buffer[512];

syscall_printString("Enter a line: ");
syscall_readString(str);
syscall_printString("\r\n");
syscall_printString(str);
syscall_printString("\r\n");
syscall_printString("Reading sector 30 from floppy\r\n");
syscall_readSector(buffer, 30);
syscall_printString(buffer);
```

Save this code into a file called ***test.c***. For this program to work you'll have to write something into sector 30 of the floppy.

Building the test program

To build the test program you have to follow these steps:

1. Compile *test.c*

   ```
   bcc -ansi -c -o test.o test.c
   ```

2. Compile *os-api.asm*

   ```
   as86 -o os-api.o os-api.asm
   ```

3. Link the test program

   ```
   ld86 -o syscall-test -d test.o os-api.o
   ```

4. Copy the test program into the **sector 11** of the floppy

   ```
   dd if=syscall-test of=floppya.img bs=512 count=1 seek=11 conv=notrunc
   ```

Notes:
Here we're assuming that the API is implemented in the file *os-api.asm*

To load the test program from sector 11 into the memory you'll have to use the following kernel function:

```
; Load a program from sector 11 into segment 0x20000
_loadProgram:
        mov ax, #0x2000
        mov ds, ax
        mov ss, ax
        mov es, ax

        ;let's have the stack start at 0x2000:fff0
        mov ax, #0xfff0
        mov sp, ax
        mov bp, ax

        ; Read the program from the floppy
        mov     cl, #12         ;cl holds sector number
        mov     dh, #0          ;dh holds head number - 0
        mov     ch, #0          ;ch holds track number - 0
        mov     ah, #2          ;absolute disk read
        mov     al, #1          ;read 1 sector
        mov     dl, #0          ;read from floppy disk A
        mov     bx, #0          ;read into offset 0 (in the segment)
        int     #0x13           ;call BIOS disk read function

        ; Switch to program
        jmp #0x2000:#0
```

You'll have to change the file *kernel.asm* to add this function. Remember to add the .*export* directive at the beginning of the file, so you can call the function from C code.
In your kernel main function you'll have to call the following functions:

```
   makeInterrupt21();
   loadProgram();
```