Understanding and Using Linux System Calls

D. Thiebaut
Nov 2008

The information on the Linux system calls is taken from Randall Hyde's Linux System Call document (http://webster.cs.ucr.edu/Page_Linux/LinuxSysCalls.pdf), and is the basis for the examples in assembly language.

Hyde says: If you read the on-line documentation for the Linux system calls, you'll find that the API calls are specified using a "C" language syntax. However, it's very easy to convert the C examples to assembly language. Just load the associated system call constant into EAX and then load the 80x86 registers with the following values:
• 1st parameter: EBX
• 2nd parameter: ECX
• 3rd parameter: EDX
• 4th parameter: ESI
• 5th parameter: EDI

File I/O

This section describes the Linux system calls responsible for file I/O. The principal functions are open, close, creat, read, write, and llseek. Advanced users make want to use some of the other functions as well.

File Descriptors

Linux and applications refer to files through the use of a

file descriptor

or

file handle

This is a small unsigned integer value (held in a dword) that Linux uses as an index into internal file tables. When you open or create a file, Linux returns the file descriptor as the open/creat result. You pass this file handle to other functions to operate upon that file.
Under Linux, the file handle values zero, one, and two have special meaning. These correspond to the files associated with the standard input, standard output, and standard error devices. Theoretically, you should use constants for these values (e.g., stdin.handle, stdout.handle), but their values are so entrenched in modern UNIX programs that it would be very difficult for Linux kernel developers to change these values.
Generally, the first file a process opens is given the value three for its file handle and successively open files are given the next available (sequential) free handle value. However, your programs certainly should not count on this behavior. Someone may decide to add another "standard" device handle to the mix, or a kernel developer may decide it's better to start the handles with a larger value for some reason. In general, other than to satisfy your curiosity, you should never examine or modify the file handle value. You should treat the value as Linux's private data.

File operations

- Open/Create file (8)
- Open/Append to file (5)
- Read from file (3)
- Write to file (4)
- Close file (6)

Assembly Constants

The constants below are needed to define the mode of operation on the files (read, write, or both), and what permissions are given to the file (user-only access, or group, or world access).

```
%assign SYS_EXIT          1
%assign SYS_WRITE         4
%assign SYS_READ          3
%assign STDOUT            1
%assign SYS_OPEN          5
%assign SYS_CLOSE         6
%assign SYS_CREATE        8


%assign O_RDONLY          000000q    ; file is read-only
%assign O_WRONLY          000001q    ; file is write-only
%assign O_RDWR            000002q    ; read or write
%assign O_CREAT           000100q    ; create file or erase it

%assign S_IRUSR           00400q     ; user permission to read
%assign S_IWUSR           00200q     ; to write
%assign S_IXUSR           00100q     ; to execute
```

How to create a new file

We can use several macros to create the file, write a string to it, and close the file. The name of the file should be an ASCIIZ string (string terminated by a 0), and the data to be written should be in an array of bytes, and the length of the data should be known.
The macro createFile creates a file that is RWX by the user only. Group and Others will not have access to the file.

```
;;; createNewFile.asm
;;; D. Thiebaut
;;; Create a new file with name data.txt and stores a
;;; string in it.
;;;
%assign SYS_EXIT    1
%assign SYS_WRITE   4
%assign SYS_READ    3
%assign SYS_LSEEK   19
```

```
%assign SEEKSET     0
%assign STDOUT      1
%assign SYS_OPEN    5
%assign SYS_CLOSE   6
%assign SYS_CREATE  8


%assign O_RDONLY    000000q
%assign O_WRONLY    000001q
%assign O_RDWR      000002q
%assign O_CREAT     000100q

%assign S_IRUSR            00400q
%assign S_IWUSR            00200q
%assign S_IXUSR            00100q

;;; ; --- MACRO ------------------------------------------------
;;; ;     print           msg,length
%macro  print 2 ; %1 = address %2 = # of chars
        pushad            ; save all registers
        mov             edx,%2
        lea             ecx,[%1]
        mov             eax,SYS_WRITE
        mov             ebx,STDOUT
        int             0x80
        popad           ; restore all registers
%endmacro

;;; ; --- MACRO ------------------------------------------------
;;; ;     print2        "quoted string"
%macro  print2 1 ; %1 = immediate string,
        section .data
%%str   db              %1
%%strL  equ             $-%%str
        section .text
        print           %%str, %%strL
%endmacro

;;; --- MACRO ------------------------------------------------
;;;    createFile filename,  handle
%macro createFile 2
       mov    eax,SYS_CREATE
       mov    ebx,%1
       mov    ecx, S_IRUSR|S_IWUSR|S_IXUSR
       int    0x80

       test   eax,eax
       jns    %%createfile
       print2  "Could not open file"
       mov    eax,SYS_EXIT
       mov    ebx,0
       int    0x80         ; final system call

%%createfile:
       mov    %2,eax       ; save handle
%endmacro



;;; --- MACRO ------------------------------------------------
;;;    writeFile handle, buffer, noBytesToWrite
%macro  writeFile 3
       mov    eax,SYS_WRITE
```



```
       mov    ebx,%1
       mov    ecx,%2
       mov    edx,%3
       int    0x80
%endmacro


;;; --- MACRO ------------------------------------------------
;;;    close   handle
%macro close   1
       mov    eax,SYS_CLOSE
       mov    ebx,%1
       int    0x80
%endmacro



       ;; --------------------------
       ;; data segment
       ;; --------------------------

       section      .data
fileName db     "data.txt",0
handle   dd     0
buffer   db     "A fine romance with no kisses", 0x0a
         db     "A fine romance my friend this is.", 0x0a
LEN      equ    $-buffer


       ;; --------------------------
       ;; code area
       ;; --------------------------

       section      .text
       global _start

_start:
;;; write contents of buffer to file

        createFile      fileName, [handle]
        writeFile       [handle], buffer, LEN
        close           [handle]

       ;; exit()

       mov    eax,SYS_EXIT
       mov    ebx,0
       int    0x80         ; final system call
```

How to read an existing file

The program below illustrates how to read the contents of a file, put it in a buffer in memory, and display it on the screen.

Note that you will need a buffer to hold the data from the file. When you read the contents of the file, you must specify how much information you want to read, and

this information cannot be larger than the size of the buffer. The system call that reads data from the file returns the number of bytes read. This value is saved in a variable (noRead).

```
;;; readFile.asm
;;; D. Thiebaut
;;; Reads a file called data.txt and displays its
;;; contents on the screen.
;;;
%assign SYS_EXIT    1
%assign SYS_WRITE   4
%assign SYS_READ    3
%assign SYS_LSEEK   19
%assign SEEKSET     0
%assign STDOUT      1
%assign SYS_OPEN    5
%assign SYS_CLOSE   6
%assign SYS_CREATE  8


%assign O_RDONLY    000000q
%assign O_WRONLY    000001q
%assign O_RDWR      000002q
%assign O_CREAT     000100q

%assign S_IRUSR     00400q
%assign S_IWUSR     00200q
%assign S_IXUSR     00100q

;;; --- MACRO ------------------------------------------------
;;;     print           msg,length
%macro  print           2               ; %1 = address %2 = # of chars
        pushad                          ; save all registers
        mov             edx,%2
        lea             ecx,[%1]
        mov             eax,SYS_WRITE
        mov             ebx,STDOUT
        int             0x80
        popad                           ; restore all registers
%endmacro

;;; --- MACRO ------------------------------------------------
;;;     print2        "quoted string"
%macro  print2        1               ; %1 = immediate string,
        section .data
%%str   db              %1
%%strL  equ             $-%%str
        section .text
        print           %%str, %%strL
%endmacro

;;; --- MACRO ------------------------------------------------
;;;    openFile filename, handle
%macro openFile 2
       mov    eax,SYS_OPEN
       mov    ebx,%1
       mov    ecx, O_RDONLY
       mov    edx, S_IRUSR|S_IWUSR|S_IXUSR
       int    0x80

       test   eax,eax
       jns    %%readFile
       print2  "Could not open file"
```



```
       mov    eax,SYS_EXIT
       mov    ebx,0
       int    0x80         ; final system call

%%readFile:
       mov    %2, eax             ; save handle
%endmacro


;;; --- MACRO ------------------------------------------------
;;;    readFile handle, buffer, buffer-length, number-bytes-read
%macro  readFile 4
       mov    eax,SYS_READ
       mov    ebx,%1       ; file descriptor in bx
       mov    ecx,%2       ; buffer address
       mov    edx,%3       ; max number of bytes to read
       int    0x80
       mov    %4,eax       ; save number bytes read
%endmacro

;;; --- MACRO ------------------------------------------------
;;;    close   handle
%macro close   1
       mov    eax,SYS_CLOSE
       mov    ebx,%1
       int    0x80
%endmacro

       ;; -------------------------
       ;; data segment
       ;; -------------------------

       section      .data
fileName db     "data.txt",0
handle   dd     0
noRead   dd     0               ; to store # of chars read from file

        section .bss
MAXBUF  equ     100000
buffer  resb    MAXBUF          ; 100,000 bytes of storage

       ;; -------------------------
       ;; code area
       ;; -------------------------

       section      .text
       global _start

_start:
;;; open the file and put its contents in Buffer.
;;; keep track of # of bytes read in noRead

        openFile        fileName,  [handle]
        readFile        [handle], buffer, MAXBUF, [noRead]
        close           [handle]

        print           buffer, [noRead]

       ;; exit()

       mov    eax,SYS_EXIT
       mov    ebx,0
       int    0x80         ; final system call
```