Introducción a los Sistemas Operativos

Introducción – IV

Anexo llamadas al Sistema

Objetivo

Programar el llamado a una “System Call” de forma manual

Hello World!!

- Para programar el clasico “hello world” se necesitan mínimo realizar hacer 2 llamadas al sistema:
  - Una para escribir en pantalla el mensaje
  - Una terminar el proceso
- Por ello tendremos que hacer uso de las siguientes llamadas al sistema:
  - write (man 2 write)
  - exit (man exit)

Hello World en x86-32bit

- En x86-32bit las sistem calls tienen los siguientes números:
  - write → syscall número 4
  - exit → syscall número 1
- Linux-2.6.34.14/arch/x86/include/asm/unistd_32.h

```
/*
 * This file contains the system call numbers.
 */

#define __NR_restart_syscall      0
#define __NR_exit                 1
#define __NR_fork                 2
#define __NR_read                 3
#define __NR_write                4
#define __NR_open                 5
#define __NR_close                6
#define __NR_waitpid              7
#define __NR_creat                8
#define __NR_link                 9
#define __NR_unlink              10
#define __NR_execve              11
#define __NR_chdir               12
#define __NR_time                13
```

Hello World en x86-32bit (cont)

- Los manuales de las system calls permiten saber cuales son los parámetros

```
NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
       write()  writes  up  to  count bytes from the buffer pointed buf to the
       file referred to by the file descriptor fd.
```

```
NAME
       exit - cause normal process termination

SYNOPSIS
       #include <stdlib.h>

       void exit(int status);

DESCRIPTION
       The  exit() function causes normal process termination and the value of
       status & 0377 is returned to the parent (see wait(2)).
```

```
NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
       write()  writes  up  to  count byt
       file referred to by the file descr
```

```
/*
 * This file contains the system call numbers
 */

#define __NR_restart_syscall      0
#define __NR_exit                 1
#define __NR_fork                 2
#define __NR_read                 3
#define __NR_write                4
#define __NR_open                 5
#define __NR_close                6
#define __NR_waitpid              7
#define __NR_creat                8
#define __NR_link                 9
#define __NR_unlink              10
#define __NR_execve              11
#define __NR_chdir               12
#define __NR_time                13
```

```
NAME
       exit - cause normal process termin

SYNOPSIS
       #include <stdlib.h>

       void exit(int status);

DESCRIPTION
       The  exit() function causes normal process termination and the value of
       status & 0377 is returned to the parent (see wait(2)).
```

```
_start:

    ; sys_write(stdout, message, length)

    mov eax, 4          ; sys_write syscall
    mov ebx, 1          ; stdout
    mov ecx, message    ; message address
    mov edx, 14         ; message string length
    int 80h

    ; sys_exit(return_code)

    mov eax, 1          ; sys_exit syscall
    mov ebx, 0          ; return 0 (success)
    int 80h

section .data
    message: db 'Hello, world!',0x0A    ; message and newline
```

Hello World en x86-64bit

- En x86-64bit las sistem calls tienen los siguientes números:
  - write → syscall número 1
  - exit → syscall número 60
- Linux-2.6.34.14/arch/x86/include/asm/unistd_64.h

```
/*
 * This file contains the system call numbers.
 *
 * Note: holes are not allowed.
 */

/* at least 8 syscall per cacheline */
#define __NR_read                0
__SYSCALL(__NR_read, sys_read)
#define __NR_write               1
__SYSCALL(__NR_write, sys_write)
#define __NR_open                2
__SYSCALL(__NR_open, sys_open)
#define __NR_close               3
__SYSCALL(__NR_close, sys_close)
#define __NR_stat                4
```

```
#define __NR_vfork               58
__SYSCALL(__NR_vfork, stub_vfork)
#define __NR_execve              59
__SYSCALL(__NR_execve, stub_execve)
#define __NR_exit                60
__SYSCALL(__NR_exit, sys_exit)
#define __NR_wait4               61
__SYSCALL(__NR_wait4, sys_wait4)
#define __NR_kill                62
__SYSCALL(__NR_kill, sys_kill)
#define __NR_uname               63
```

Hello World en x86-64bit (cont)

- Se usan los mismos manuales de las system call que para el caso anterior
- Lo que cambia es el número de system call, el cual está en el kernel, no en el manual de la misma.
- Los procesadores X86-64 usan un esquema de registros diferentes.
- Se usa la instrucción **syscall** en lugar de la instrucción **int 80h**

```
NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
       write()  writes  up  to  count
       file referred to by the file de
```

- write → syscall número 1
- exit → syscall número 60

```
    ; sys_write(stdout, message, length)

    mov     rax, 1          ; sys_write
    mov     rdi, 1          ; stdout
    mov     rsi, message     ; message address
    mov     rdx, length     ; message string length
    syscall

    ; sys_exit(return_code)

    mov     rax, 60          ; sys_exit
    mov     rdi, 0          ; return 0 (success)
    syscall

section .data
    message: db 'Hello, world!',0x0A     ; message and newline
    length:    equ    $-message         ; NASM definition pseudo-instruction
```

```
NAME
       exit - cause normal process termination

SYNOPSIS
       #include <stdlib.h>

       void exit(int status);

DESCRIPTION
       The  exit() function causes normal process termination and the value of
       status & 0377 is returned to the parent (see wait(2)).
```

Referencias

Como programar un “hello world” en x86 32bit y 64bit

- http://shmaxgoods.blogspot.com.ar/2013/09/assembly-hello-world-in-linux.html
- http://stackoverflow.com/questions/19743373/linux-x86-64-hello-world-and-register-usage-for-parameters

Mas información sobre formas de pasar parametros a una syscall

- http://www.int80h.org/bsdasm/#system-calls