Assembly Programming on Linux

Assembly Programming on Linux

Necessity

- OS Kernel Development
  - 80x86
  - Sparc
  - Alpha
- Some device drivers
  - SCSI Controller
  - Video Card
  - Network Card
  - Others that has ROM on board ..
- Compiler/Interpreter/Cross Assembler

Assembly Programming on Linux

Programming Environment

Platform
- 80x86 Linux (calvin.coe.p-net)

Assembler
- nasm (gas, as86, ...)

Debugger
- gdb (ddd, ???)

Linker
- ld (ld86, ???)

Binary format
- elf (a.out, coff, ...)

Editor
- whatever (pico, joe, vi, ...)

Assembly Programming on Linux

Basic Program Structure

edit prog1.asm

```
  ; line comment
global   _start ; define where the program start
  ; must be a label "_start"
segment   .text ; segment or section of code
_start:  ; starting point of program

segment   .data ; segment of predefined data

segment   .bss ; segment of uninitialize data
```

Assembly Programming on Linux

Assemble, Link and Run

- Assemble

```
$ nasm -f elf prog1.asm
```

--> prog1.o

- Link

```
$ ld -o prog1 prog1.o
```

(--> a.out)

- Run

```
$ ./prog1
```

Segmentation fault

Assembly Programming on Linux

Segmentation fault?

- Program must be terminated

C - function exit()
or ‘return’ from main() function

```
mov eax, 1
int 0x80
```

```
segment .text
        global _start
_start:
        mov eax, 1      ; exit function
        int 0x80
```

Assembly Programming on Linux

Hello World

```
segment  .text
        global _start
_start:
        mov eax, 4  ; write function
        mov ebx, 2  ; STDOUT
        mov ecx, hello ; hello message
        mov edx, msglen ; lenght
        int  0x80  ; system call

        mov eax, 1  ; exit function
        int  0x80  ; system call

segment  .data
hello  db "Hello World", 0xA, 0xD
msglen  equ $-hello
```

Assembly Programming on Linux

Linux System Call

/usr/src/linux/include/asm/unitstd.h

```
#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
#define __NR_mknod               14
#define __NR_chmod               15
```

Assembly Programming on Linux

Documentation of Linux System Call

$ echo "export MANPATH=/usr/share/man" >> ~/.bashrc
$ man 2 read

READ(2) Linux Programmer's Manual READ(2)

NAME
read - read from a file descriptor

SYNOPSIS
#include <unistd.h>

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

DESCRIPTION
read() attempts to read up to count bytes from file

Assembly Programming on Linux

Using of Linux System Call

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

Call read function

```
mov     eax, 3 ; function read
mov     ebx, stdin ; fd = stdin
mov     ecx, buf ; buf -> buf
mov     edx, count ; count = count
int     0x80  ; system call
```

Call write function

```
mov     eax, 4 ; function write
mov     ebx, stdout ; fd = stdout
mov     ecx, buf ; buf -> buf
mov     edx, count ; count = count
int     0x80  ; system call
```

Assembly Programming on Linux

Using of Linux System Call (cont.)

```
stdin   equ     1
stdout  equ     2
count   equ     1

        segment .text
        global  _start
_start:

        ... read system call ...
        ... write system call ...
        mov     eax, 1
        int     0x80

        segment .bss
buf     resb    80
```

Assembly Programming on Linux

Using of Linux System Call (cont.)

Assemble, Link and Run

```
$ nasm -f elf test1.asm
$ ld -o test1 test.o
$ ./test1
a
a$
```

Questions:

What happen when program waiting for input more than one character is entered?

Why?

Home work: