SecurityTube Linux Assembly Expert (SLAE^{64})

SecurityTube Linux Assembly Expert

Training: http://www.SecurityTube-Training.com

Pentester Academy: http://www.PentesterAcademy.com

Vivek Ramachandran
SWSE, SMFE, SPSE, SGDE, SISE, SLAE^{32,64} Course Instructor

Module 1: 64-Bit ASM on Linux

5. Hello World in 64-bit Assembly

Vivek Ramachandran
SWSE, SMFE, SPSE, SGDE, SISE, SLAE^{64}, SLAE^{32} Course Instructor

http://SecurityTube-Training.com

Programming in Assembly

- NASM + LD for assembling and linking
- Executable in ELF format

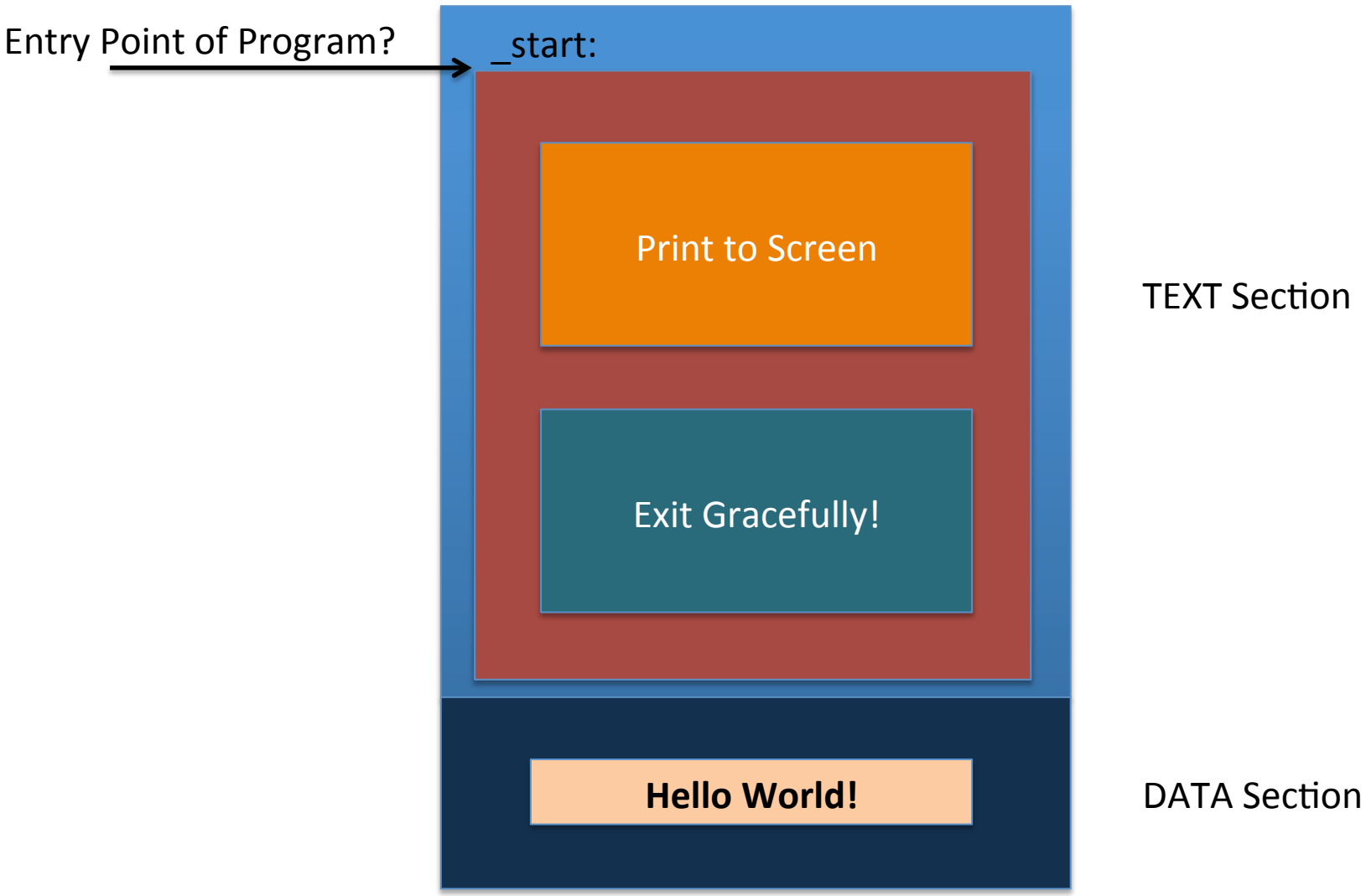
NASM Documentation:

http://nasm.us/

Assembly Language Syntax

- AT&T
- Intel

Hello World!



Why System Calls?

- Leverage OS for tasks
- Imagine if you had to write code from scratch to:
  - write to disk
  - print on screen
  - …
- System Calls provide a simple interface for user space programs to the Kernel

Where are these system calls defined?

```
#ifndef _ASM_X86_UNISTD_64_H
#define _ASM_X86_UNISTD_64_H

#ifndef __SYSCALL
#define __SYSCALL(a, b)
#endif

/*
 * 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
__SYSCALL(__NR_stat, sys_newstat)
#define __NR_fstat                              5
__SYSCALL(__NR_fstat, sys_newfstat)
#define __NR_lstat                              6
__SYSCALL(__NR_lstat, sys_newlstat)
#define __NR_poll                               7
__SYSCALL(__NR_poll, sys_poll)

#define __NR_lseek                              8
__SYSCALL(__NR_lseek, sys_lseek)
#define __NR_mmap                               9
__SYSCALL(__NR_mmap, sys_mmap)
#define __NR_mprotect                           10
__SYSCALL(__NR_mprotect, sys_mprotect)
#define __NR_munmap                             11
__SYSCALL(__NR_munmap, sys_munmap)
"/usr/include/x86_64-linux-gnu/asm/unistd_64.h" [readonly] 717L, 23049C
```

X86_64Mechanism to invoke System Call

- Syscall

- More Information:
  - http://blog.tinola.com/?e=5

write()

```
WRITE(2)                                                Linux Programmer's Manual

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 th
```

exit

```
_EXIT(2)                                                    Linux Pro

NAME
       _exit, _Exit - terminate the calling process

SYNOPSIS
       #include <unistd.h>

       void _exit(int status);
```

Invoking System Call with syscall

RAX	**System Call Number**	**Return Value in RAX**
RDI	1st Argument	
RSI	2nd Argument	
RDX	3rd Argument	
R10	4th Argument	
R8	5th Argument	
R9	6th Argument	

Calling Write

```
WRITE(2)                                                Linux Programmer's Manual

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 th
```

RAX = system call number

RDI = STDOUT

RSI = Pointer to “Hello World”

RDX = Length of “Hello World”

Calling Exit

```
_EXIT(2)                                                    Linux Pro

NAME
       _exit, _Exit - terminate the calling process

SYNOPSIS
       #include <unistd.h>

       void _exit(int status);
```

RAX = system call number

RDI = Status Code

Exercise: GDB

- Use GDB to step through the Hello World program and observe:
  - CPU Registers
  - Memory Location
  - …