akaedu

xwp

March 16, 2014

Contents

1 linux c 基础复习

1.1 基本命令

1.2 vim 操作

1.3 c 基础

1.3.1 基本类型

1.3.2 循环控制语句

1.3.3 数组

1.3.4 指针

1.3.5 结构体

2 Ubuntu 系统安装软件方法

2.1 Ubuntu 服务器软件源安装

- Ubuntu 软件中心 –》编辑 –》软件源
- 更新本地软件源列表 sudo apt-get update
- 搜索软件 sudo apt-cache search thunderbird
- 安装软件 sudo apt-get install thunderbird
- 卸载软件 sudo apt-get autoremove thunderbird

2.2 deb 包安装

- 网上下载相关软件 deb 包
- 安装 sudo dpkg -i filename.deb
- 卸载 sudo dpkg -r filename

2.3 源代码安装

- 网上下载相关软件源代码包（.tar.gz/.zip/.tar.bz2/.tar）
- 进入源码包，编译源代码包，并且安装
  - ./configure

  - make
  - sudo make install
- 卸载源代码包安装的软件, 进入到源代码包里执行
  - sudo make distclean

2.4 ubuntu 使用小技巧

- 显示桌面，Ctrl+win+d
- 切换工作区，Ctrl+Alt+up/down/left/right

3 git 使用教程

4 计算机组成原理

4.1 寄存器

4.2 CPU

4.3 内存

4.4 总线

4.5 MMU

5 X86 汇编

5.1 学习汇编的目的

学习汇编不是为了掌握用汇编去写程序，而是为了更好的透视理解C语言行为，理解处理器的工作方式，了解程序背后底层的东西，相当于习武中的内功心法，以便你日后写出高效的代码和调试诡异的BUG。

5.2 第一个汇编程序

```
#PURPOSE: Simple program that exits and returns a
#
status code back to the Linux kernel
#
#INPUT:
none
```

```
#
#OUTPUT: returns a status code. This can be viewed
#
by typing
#
#
echo $?
#
#
after running the program
#
#VARIABLES:
#
%eax holds the system call number
#
%ebx holds the return status
#
.section .data
.section .text
.globl _start
_start:
movl $1, %eax # this is the linux kernel command
#number (system call) for exiting
#a program
movl $4, %ebx # this is the status number we will
          # return to the operating system.
         # Change this around and it will
        # return different things to
       # echo $?
int $0x80 # this wakes up the
```

5.2.1 编译链接过程

.c –> .i –> .s –> .o –> a.out(ELF)

文件类型	文件属性	工具链	目标文件类型
.c	源代码	cpp 预处理器	.i
.i	预处理后的文件	gcc 编译器	.s
.s	编译后的文件	as 汇编器	.o
.o	汇编后的文件	ld 链接器	a.out
a.out	链接后的可执行文件	loader 加载器	进程

5.3 X86 寄存器

寄存器	功能
eax	通用
ebx	通用
ecx	通用
edx	通用
esi	通用
edi	通用
ebp	桢指针
esp	栈指针
eip	程序计数器
eflag	程序状态
浮点寄存器	浮点数运算

5.4 汇编语法

去我的 nfs 服务器上下载汇编电子教材

5.5 寻址方式

```
int a = 3, d = 5;
int b[5] = {1,2,3,4,5};
struct STU {
int id;
char name[20];
char sex;
}c;
int *p;
```

寻址方式	含义	对应 C 语法
直接寻址	movl ADDRESS, %eax	a = *p
变址寻址	movl $\text{data}_{\text{items}}$(,%edi,4), %eax	a = b[3’]’
间接寻址	movl(%eax), %ebx	a = *p
基址寻址	movl 4(%eax), %ebx	a = *(p+1)
立即数寻址	movl $12,%eax	a = 12
寄存器寻址	movl %eax, %ebx	a = d

5.6 E L F 文件格式

ELF 文件格式是一种开放的标准，unix/linux 可执行程序都采用 ELF 标准。

- 可重定位的目标文件 (Relocatable, 或者 Object File) 如.o 文件
- 可执行文件 (Executable)
- 共享库 (Shared Object, 或者 Shared Library)

5.6.1 绝对地址和相对地址辨析

- 类别绝对路径和相对路径
- 相对地址：内部偏移地址，两条指令间的相对距离
- 绝对地址：虚拟地址 4G 空间内的唯一地址

5.6.2 Section 和 Segment 辨析

- Section：站在链接器角度的链接单位，如：

段名	含义	解释
Section Headers	段符号表	保存了各个段的起始地址和大小
.text	代码段	存放程序执行的机器码指令
.ro.data	只读数据段	存放只读常量
.data	数据段	存放已初始化的全局变量
.Bss	0 数据段	存放未初始化全局变量
.shstrtab		保存着各 Section 的名字
.strtab		保存着程序中用到的符号的名字
.rel.text	重定位符号表	告诉链接器指令中的哪些地方需要做重定位
.symtab	符号表	保存了符号内部的相对地址

- Segment：站在加载器角度的加载单位，如：

PrProgram Headers:

Type	Offset	VirtAddr	PhysAddr	FileSize	MemSiz	Flg	Align
LOAD	0x000000	0x08048000	0x08048000	0x0009e	0x0009e	R E	0x1000
LOAD	0x0000a0	0x080490a0	0x080490a0	0x00038	0x00038	R W	0x1000

Section to Segment mapping:
Segment Sections...

segment 编号	segment 名
00	.text
01	.data

5.7 查看反汇编调试程序

- gcc -g test.c -o test
- objdump -dSsx test > file
- vim file
- 找到 main 函数的位置，然后开始分析

6 C 和汇编的关系

6.1 重点指令讲解

指令	示例	作用
push		
pop		
call		
leave		
ret		

6.1.1 栈的类型

栈的类型	工作方式
满递减	
满递增	
空递减	
空递增	

6.1.2 call

call 0x80a0000<foo>

- 把 call 的下一条指令的地址压到栈上保存
- 修改 eip，跳到 foo 函数的地址上去执行

6.1.3 ret

call 指令的逆向操作

- 把当前 esp 里保存的返回地址值给 eip
- 由于修改了 eip，所以跳向返回地址

6.1.4 leave

是函数开头的 push %ebp 和 mov %esp,%ebp 的逆操作

- 把 ebp 的值赋给 esp
- 现在 esp 所指向的栈顶保存着 foo 函数栈帧的 ebp , 把这个值恢复给 ebp , 同时 esp 增加 4

6.1.5 总结

- x86 平台函数调用，利用栈来传递参数
- 每个函数都有自己的桢指针，称为函数的栈桢，通过函数的栈桢向上可以取到调用函数的参数，向下可以取到函数定义的局部变量
- 函数调用时从右到左去压栈

6.2 变量存储布局

6.2.1 存储区域

存储区域	存储元素
栈	局部变量
data 段	已初始化全局变量，static 修饰并初始化的变量
bss 段	未初始化全局变量，static 修饰未初始的变量
rodata	只读数据，const 修饰的变量
text	存放对局部变量初始化的数据

6.2.2 存储的生命周期

存储区域	生命周期
栈	函数调用结束
data 段	整个程序结束前，都有效
bss 段	整个程序结束前，都有效
rodata	整个程序结束前，都有效
text	整个程序结束前，都有效

6.2.3 作用域

定义方式	作用域
在语句块{ 定义的变量，for（）{int a;}	本语句块内
在函数内定义的变量，fun{int a;}	本函数体内
在文件里, 其他函数外定义的 static int a;	本文件内
在文件里, 其他函数外定义的 int a;	所有文件

6.2.4 延长局部变量生命周期

```
  void foo(void)
{
static int a;
printf("%d\n", a);
a = 3;
printf("%d\n", a);
}
int main(void)
{
foo();
foo();
}
```

6.2.5 命名空间

6.2.6 链接属性

符号	作用域
globel/外部链接	全局
local/内部链接	局部
no link	不能用于链接

6.2.7 其他