沈阳航空航天大学

计算机学院

**实验报告**

|  |  |
| --- | --- |
| 课程名称 | 操作系统 |
| 专 业 | 物联网工程 |
| 班 级 | 2234280102 |
| 学 号 | 223428010210 |
| 学生姓名 | 陈梓欣 |
| 指导教师 | 宗传玉 |
| 实验时间 | 周五9-10节 |
| 实验地点 | 工训楼310 |

# 实验名称

系统调用

# 二、实验目的

1. 掌握Linux系统调用工作原理和Linux系统调用的执行过程。
2. 掌握增加系统调用及添加内核函数的编程方法。

# 三、实验内容和要求

1、在Linux0.11内核中添加一个新的系统调用函数max函数，函数原型：

int max(int a, int b, int c)

该函数实现比较三个参数的大小，并将最大值作为函数返回值；

2、在Linux 0.11内核中添加两个系统调用函数Iam和Whoami，函数原型如下：

（1）int Iam(const char\* name)；将字符串name的内容保存到内核中，返回值是拷贝的字符数，如果name长度大于32，则返回-1，并置errno为EINVAL。

（2）int Whoami(char\* name, int size)；将Iam保存到内核中的字符串拷贝到数据缓冲区name中，size为数据缓冲区name的长度，返回值是拷贝的字符数。如果size小于所需空间，则返回-1，并置errno为EINVAL。

编写两个应用程序。其中，Iam应用程序调用Iam函数，并使用命令行中的第一个参数作为Iam函数的参数；Whoami应用程序调用Whoami函数，并打印输出获取的字符串。测试效果如下图：



**图1 测试结果**

# 四、实验设计

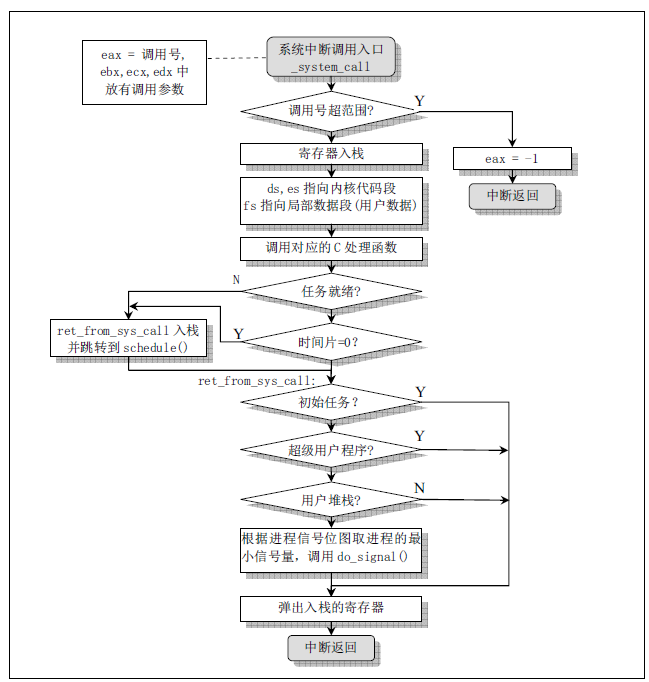
系统调用是操作系统为应用程序提供的与内核进行交互的一组接口。通过这些接口， 用户态应用程序的进程可以切换到内核态，由系统调用对应的内核函数代表该进程继续运行，从而可以访问操作系统维护的各种资源，实现应用程序与内核的交互。

系统调用通过中断机制向内核提交请求，系统调用的功能由内核函数实现，进入内核后不同的系统调用会有其各自对应的内核函数，这些内核函数就是系统调用的“服务例程”。

Linux 内核为了区分不同的系统调用，为每个系统调用分配了唯一的系统调用号。这些系统调用号定义在文件 include/unistd.h 中，系统调用号实际上对应于include/linux/sys.h 中定义的系统调用函数指针表数组 sys\_call\_table[]中项的索引值（也就是数组的下标）。对于所有系统调用的实现函数，内核把它们安照系统调用功能号顺序排列成一张函数指针表（在include/linux/sys.h文件中）。然后在中断int0x80的处理过程中，根据用户提供的功能号调用对应系统调用函数进行处理。

在 Linux 中规定 int 0x80 指令是系统调用的总入口，系统调用号存放在 EAX 寄存器中，应用程序通过系统调用传递给内核函数的参数依次存放于 EBX、ECX 和 EDX 这三个寄存器中（即系统调用最多只能有 3 个参数）。

Linux 系统调用的执行过程为：首先，应用程序准备好系统调用需要的参数，并直接或间接（通过库函数）发出一个调用请求，即调用 int 0x80 指令；然后，该指令通过陷阱处理机制（中断处理机制的一种），使该系统调用进入内核的入口点\_system\_call 函数（kernel/system\_call.s 文件中第 101 行）；最后，由\_system\_call 函数找到系统调用对应的内核函数，该内核函数将被执行并通过 EAX 寄存器返回结果。系统调用处理过程的整个流程见下图。



**图2 系统调用流程图**

# 五、实验步骤及实验结果

1. 准备实验

1）启动Engintime Linux Lab。

2）新建一个Linux011 Kernel实验项目。

1. 添加新的系统调用函数

为Linux 0.11添加5个新的系统调用函数，分别是：

1. maxALL——该函数实现比较三个参数的大小并将较大值返回的功能。
2. Iam——将输入的字符串内容保存到内核中，返回值是拷贝的字符数（限制字符串长度最长为32）。
3. Whoami：将Iam保存到内核中的字符串拷贝到数据缓冲区中（限制字符串长度最长为32）。
4. TIam——将输入的字符串内容保存到内核中，返回值是拷贝的字符数（限制字符串长度最长为12）。
5. TWhoami——将Iam保存到内核中的字符串拷贝到数据缓冲区中（限制字符串长度最长为12）。
6. 步骤如下：

1）首先要为新系统调用分配一个唯一的系统调用号，以原来的最大系统调用号为基础加1作为新的系统调用号。在“项目管理器”窗口中双击打开include/unistd.h文件，在第163-167行添加新的系统调用号，如图2所示：



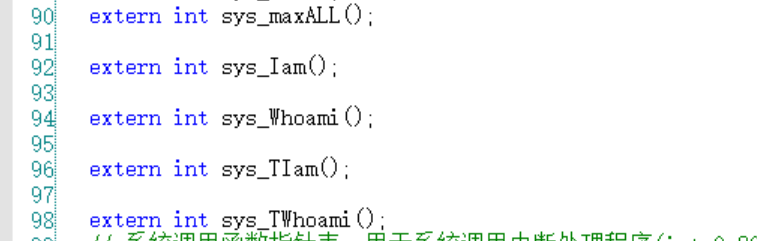
**图3 添加新的系统调用号**

2）添加新系统调用号的同时，也要使系统调用总数在原来的基础上增加1。打开kernel/system\_call.s文件，修改在第73行定义的系统调用总数，如下图：



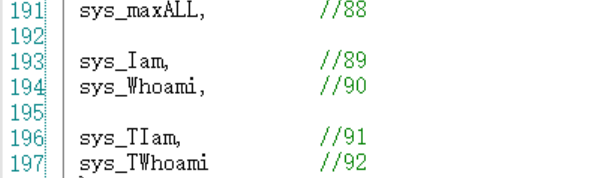
**图4 修改Linux内核的系统调用总数**

3）在include/linux/sys.h文件中的第90-98行使用C语言声明内核函数的原型，如图4所示。注意，这里定义的函数原型并不需要与函数的定义完全一致，只需要定义函数的返回值为int类型，参数为空，能够让标识符代表一个函数名称即可。



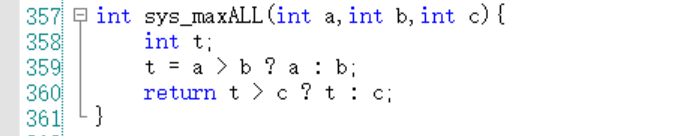
**图5 使用C语言声明内核函数的原型**

在此文件的最后，向系统调用函数指针表sys\_call\_table[]中添加新系统调用函数的指针（注意，系统调用号必须与系统调用内核函数指针在系统调用函数表中的索引一一对应），如图5所示：

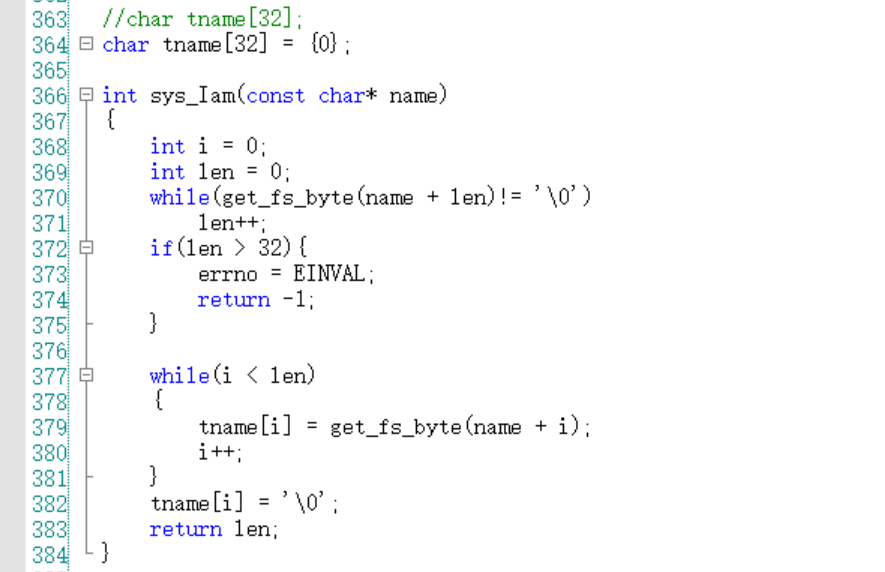


**图6 在系统调用函数指针表中增加内核函数的指针**

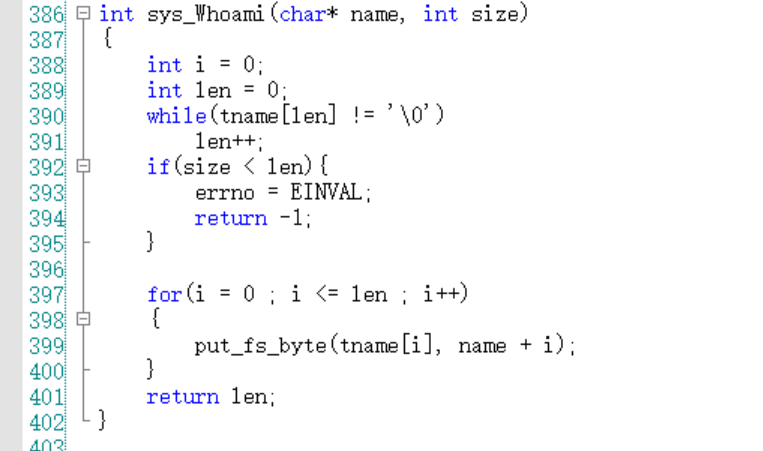
4）在kernel/sys.c文件的最后编写代码，实现新系统调用对应的内核函数，如下图所示：



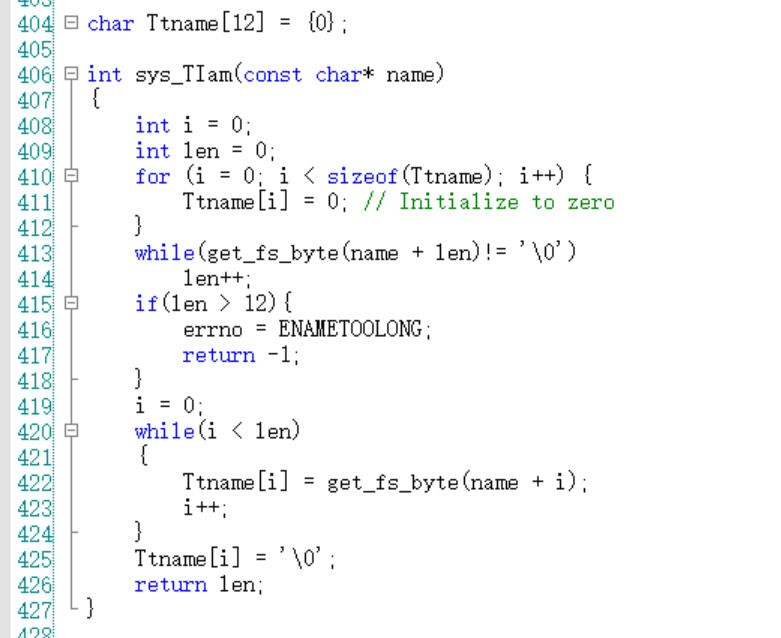
**图7 系统调用内核函数maxALL**

****

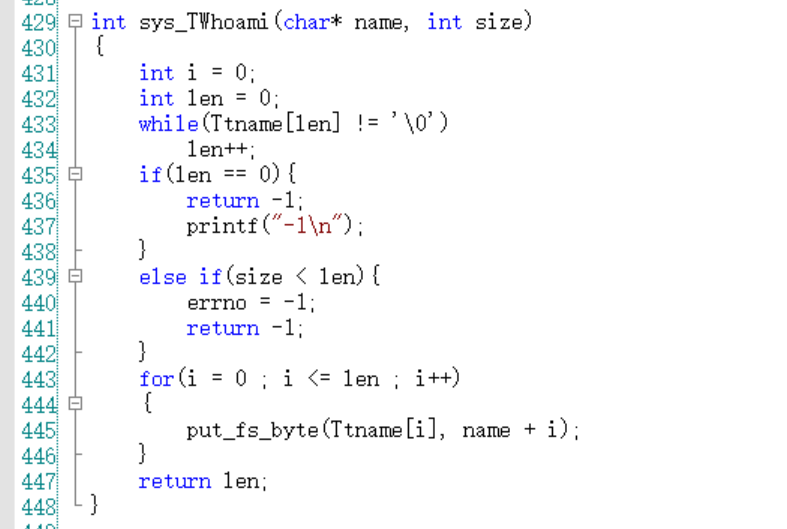
**图8 系统调用内核函数Iam**

****

**图9 系统调用内核函数Whoami**

****

**图10 系统调用内核函数TIam**

****

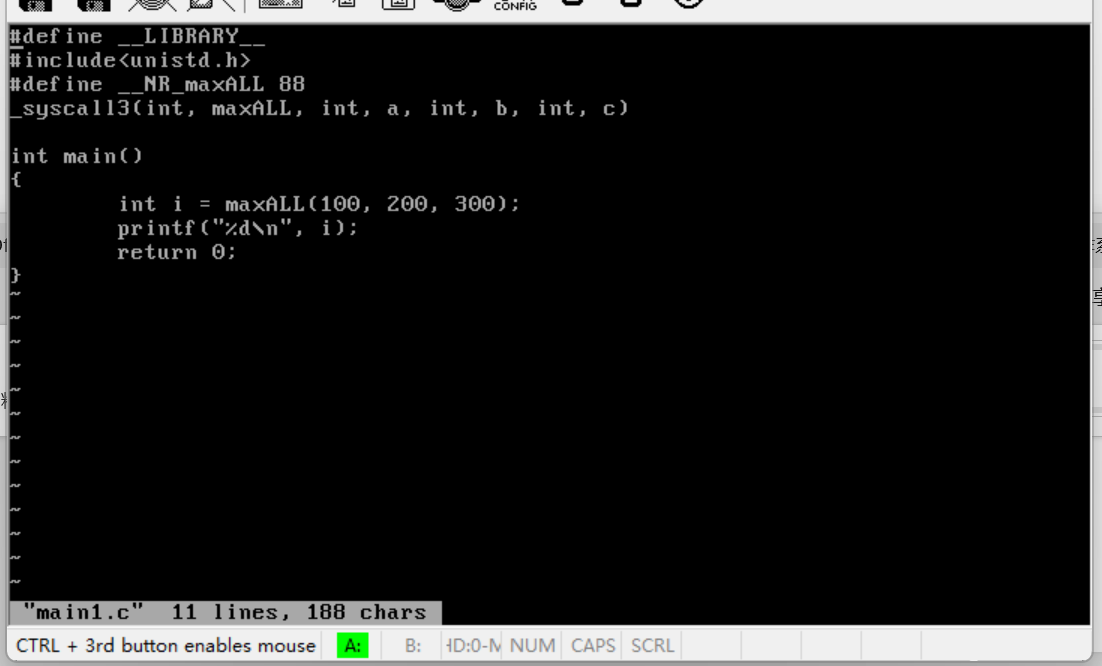
**图11 系统调用内核函数TWhoami**

5）按F7生成Linux 0.11内核，修改语法错误直到生成成功。接下来需要在Linux 0.11中分别编写应用程序来测试新系统调用是否添加成功。

1. 在应用程序中测试新系统调用

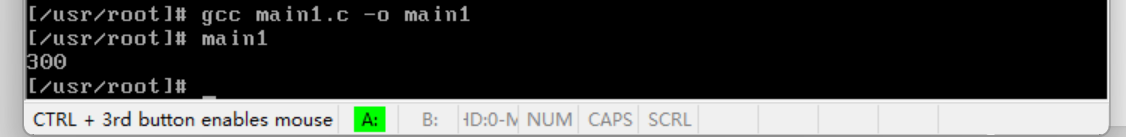
1）按F5启动调试。

2）待Linux 0.11启动后，使用vi编辑器新建一个main1.c文件。编辑main1.c文件中的源代码（如图7所示）。其中，需要定义 \_\_LIBRARY\_\_ 宏及包含unistd.h头文件，还需要再次定义 \_\_NR\_max 宏，并使用 \_syscall2宏对系统调用函数进行定义，当该宏展开时，就会在源代码文件中使用C语言添加maxALL函数的完整实现。



**图12 编辑main1.c文件**

3）使用命令gcc main.c -o main生成可执行文件main1。



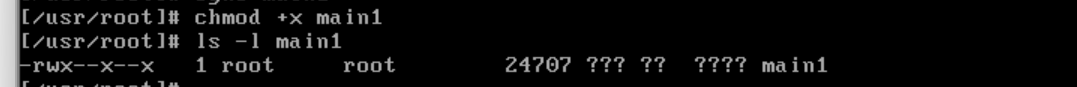
4）执行sync命令，将文件保存到磁盘。





**图13 将文件保存到磁盘**

5）执行 chmod +x main 命令为main文件添加可执行权限。



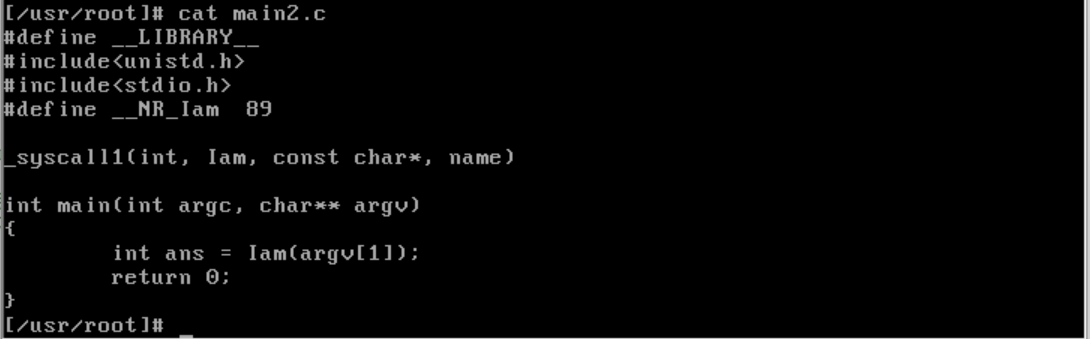
**图14 添加权限**

6）运行main1。如果main1能输出正确的结果，则说明新系统调用添加成功。

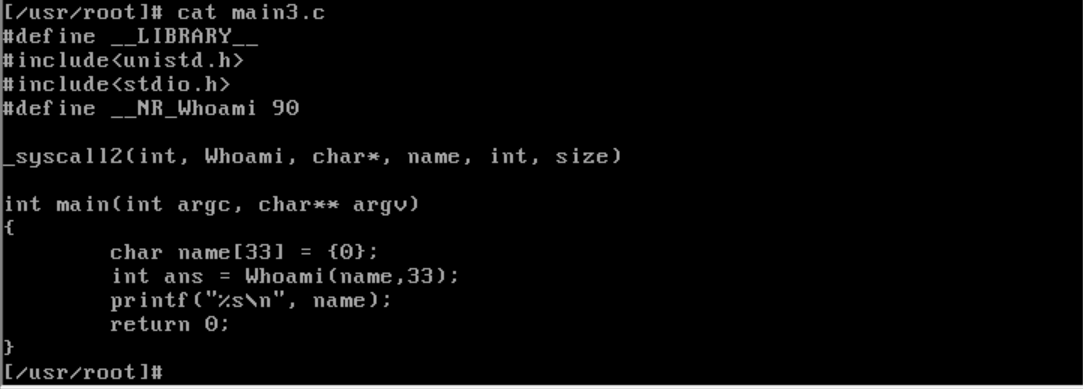
屏幕截图 2024-10-24 190747

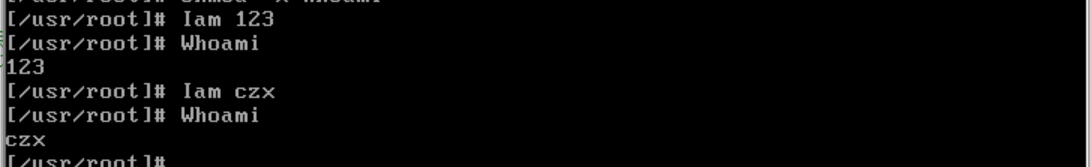
**图15 正确返回最大值**

1. 仿照上述2）-6）步骤，分别测试新系统调用Iam与Whoami。



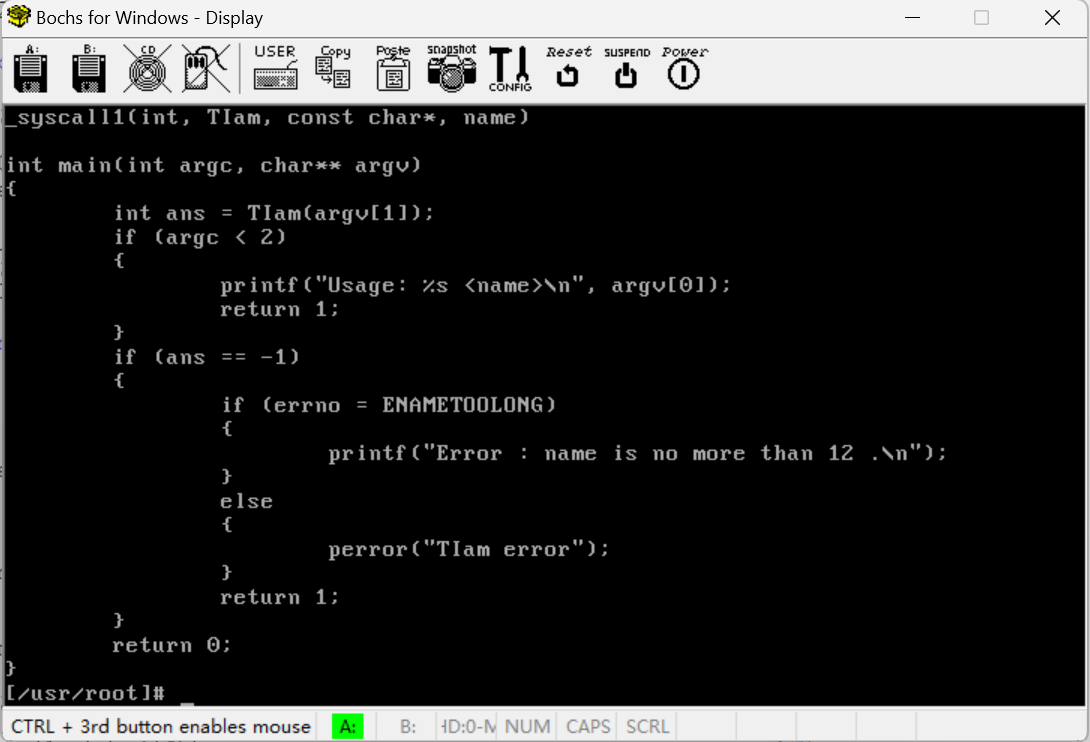
**图16 编辑main2.c文件**

**图17 编辑main3.c文件**

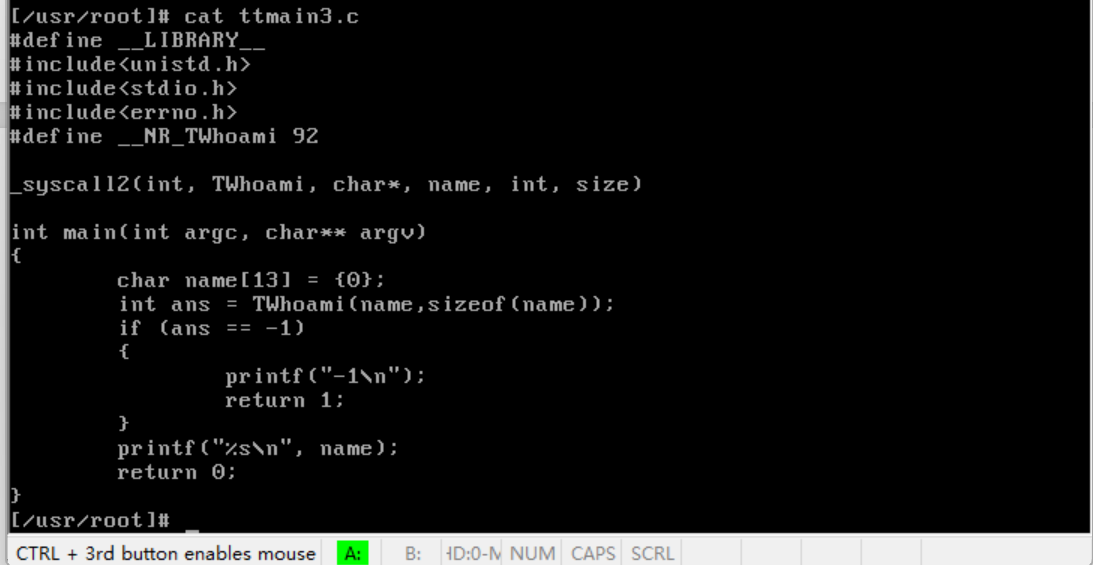


**图18 正确返回输入的字符串**

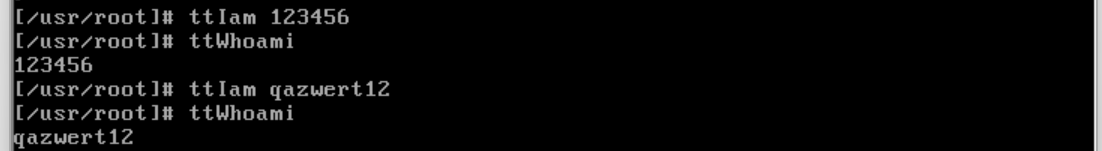
1. 在Linux 0.11内核中添加两个系统调用函数TIam和TWhoami。

要求：名字最大长度是12， 如果用户输入字符串长度大于12，TIam程序提示： name is no more than 12。 此时运行TWhoami程序，提示 -1

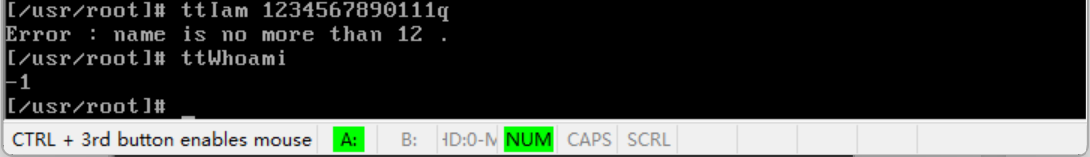
**图19 编辑ttmain2.c文件**



**图20 编辑ttmain3.c文件**



**图21 输入字符串符合要求则正确返回**



**图22 字符串长度大于12则按要求报错**

# 六、实验出现的问题及解决方法

问题：运行调用TIam函数编写的程序，当输入字符串长度大于12时，所输入的字符串无法被正确存入内核，则当前程序无法覆盖掉内核中已有的数据，导致接下来运行调用TWhoami函数的程序无法返回预期的-1。

解决方法：在TWoami函数中添加一个ofr循环语句，实现：在每一次调用TWhami函数从用户空间读取数据前，先清空上一次程序运行时的数据残留，避免当输入字符串长度大于12时误读残留数据导致程序无法正确返回-1。

# 七、结论

本次实验通过向Linux 0.11内核添加新的系统调用函数，深入理解了Linux系统调用的工作原理及其实现过程。实验首先通过学习Linux内核的系统调用机制，掌握了如何通过中断（int 0x80）进入内核，调用系统调用函数并传递参数。实验中，我们添加了一个`max`函数，该函数实现比较三个参数的大小并返回最大值。此外，还实现了两个新的系统调用：`Iam`用于将字符串保存到内核中，`Whoami`则从内核中读取该字符串并返回。

在实现过程中，我们熟悉了Linux内核中系统调用的注册与调度机制。通过修改内核源代码，如`unistd.h`、`sys.h`、`sys\_call\_table`等文件，成功为内核添加了新的系统调用。实验中还通过调试工具分析了系统调用执行的过程，掌握了如何通过寄存器传递参数和返回值，如何调试系统调用，并确保在内核中正确保存和读取数据。

通过本次实验，我对Linux内核的系统调用机制、内核函数的实现和调试过程有了更加深入的理解，也提高了在Linux环境下进行系统级编程和调试的能力。