## 实验目的

* 建立对系统调用接口的深入认识；
* 掌握系统调用的基本过程；
* 能完成系统调用的全面控制；
* 为后续实验做准备。

## 实验内容

|  |
| --- |
| 此次实验的基本内容是：在 Linux 0.11 上添加两个系统调用，并编写两个简单的应用程序测试它们。 （1）iam() 第一个系统调用是 iam()，其原型为：  int iam(const char \* name);  完成的功能是将字符串参数 name 的内容拷贝到内核中保存下来。要求 name 的长度不能超过 23 个字符。返回值是拷贝的字符数。如果 name 的字符个数超过了 23，则返回 “-1”，并置 errno 为 EINVAL。  在 kernal/who.c 中实现此系统调用。 |
| （2）whoami() 第二个系统调用是 whoami()，其原型为：  int whoami(char\* name, unsigned int size);  它将内核中由 iam() 保存的名字拷贝到 name 指向的用户地址空间中，同时确保不会对 name 越界访存（name 的大小由 size 说明）。返回值是拷贝的字符数。如果 size 小于需要的空间，则返回“-1”，并置 errno 为 EINVAL。  也是在 kernal/who.c 中实现。 |
| （3）测试程序 运行添加过新系统调用的 Linux 0.11，在其环境下编写两个测试程序 iam.c 和 whoami.c。 |

## 实验报告

|  |
| --- |
| * 从 Linux 0.11 现在的机制看，它的系统调用最多能传递几个参数？你能想出办法来扩大这个限制吗？ * 用文字简要描述向 Linux 0.11 添加一个系统调用 foo() 的步骤。 |
| 最多三个参数，也就是通过寄存器ebx，ecx，edx，可以通过寄存器传递参数位置，也就是寄存器间接寻址来传递数组来实现多个参数的传递。 |
| 在include/unistd.h中添加用户系统调用接口和系统调用号  在kernel/system\_call.s中修改调用总数  修改 include/linux/sys.h的函数调用表  在foo.c编写函数实现foo系统调用  修改makefile  编译  替换内核头文件  通过系统调用的宏编写用户接口 |

## 实验步骤

1. 在include/unistd.h中添加用户系统调用接口和系统调用号

|  |
| --- |
| 修改文件 |
| 2020-05-04 21-30-14 的屏幕截图 |
| 2020-05-04 21-31-37 的屏幕截图 |

1. 在kernel/system\_call.s中修改调用总数

|  |
| --- |
| 2020-05-04 21-32-50 的屏幕截图 |

1. 修改 include/linux/sys.h的函数调用表

|  |
| --- |
| 2020-05-04 21-34-18 的屏幕截图 |
| 2020-05-04 21-34-23 的屏幕截图 |

1. 编写who.c实现系统调用

|  |
| --- |
| #define \_\_LIBRARY\_\_  #include "unistd.h"  #include "asm/segment.h"  #include "errno.h"  #define MAX\_SIZE 23  char user\_name[MAX\_SIZE+1];  int sys\_iam(const char \*name)  {  int i;  for(i=0; get\_fs\_byte(name+i)&&i<=MAX\_SIZE; i++);  if(i>MAX\_SIZE)  {  //错误做法 return errno=EINVAL,-1  //EINVAL是个宏定义,置errno并返回-1下同  return -EINVAL;  }  for(i=0; user\_name[i]=get\_fs\_byte(name+i); i++);  return i;  }  int sys\_whoami(char \*name, unsigned int size)  {  int i;  //需要复制'\0'在判断中进行  for(i=0; put\_fs\_byte(user\_name[i], name+i), user\_name[i]&&i<=MAX\_SIZE; i++);  return i>size? -EINVAL: i;  } |

1. 修改makefile

|  |
| --- |
| 2020-05-04 21-36-09 的屏幕截图2020-05-04 21-36-14 的屏幕截图 |

1. 编写iam.c whoami.c

|  |
| --- |
| #define \_\_LIBRARY\_\_  #include "unistd.h"  #include "stdio.h"  \_syscall2(int, whoami,char\*,name,unsigned int,size);  int main(int argc, char\*\* argv)  {  char buf[32];  return whoami(buf,24)>0?puts(buf),0:-1;  } |
| #define \_\_LIBRARY\_\_  #include "unistd.h"  #include "stdio.h"  \_syscall1(int, iam, const char\*, name);  int main(int argc, char\*\* argv)  {  return argc==2&&iam(argv[1])>0?0:-1;  } |

1. 挂载硬盘，替换头文件

|  |
| --- |
| 2020-05-04 09-47-04 的屏幕截图 |

1. 运行测试

|  |  |
| --- | --- |
| 测试一 | 2020-05-04 21-41-24 的屏幕截图 |
| 测试二 | 2020-05-04 21-41-50 的屏幕截图 |
| 测试三 | 2020-05-04 21-42-00 的屏幕截图 |