```
make kernel/mysys.o ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-

make kernel/sys.o ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-

make drivers/char/mtest/sample.o ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
```

kernel/mysys.c:-

```
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(testcall)
{
  printk("This is a test call\n");
  return 0;
}
```

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- kernel/mysys.o
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- zImage
```

```
//arch/arm/tools/syscall.tbl
398  common mytestcall sys_mytestcall
```

2) take two integer arguments, return an integer may be sum of two numbers

```
SYSCALL_DEFINE2(testcall, int, x, int , y) {

}
==> long sys_testcall(int x,int y) {
}
```

3) receive a string and log inside system call

```
SYSCALL_DEFINE2(test, const char*, buf, size_t,nbytes)  {
//      long sys_testcall(const char* buf, size_t nbytes)  {
        char tbuf[64];
        ret=copy_from_user(tbuf, ubuf, nbytes);
        printk("%s",tbuf);
}
```

4) echo back string, testcall(s1, s2);

```
long sys_testcall(const char* src, const char* dest, size_t nbytes) {
        char tbuf[64];
        ret=copy_from_user(tbuf, src, nbytes);
        ret=copy_to_user(dest,tbuf,nbytes);
        return nbytes;
}
```

How to browse kernel source:-

offset of
container of

```
sudo apt install cscope
# go to KSRC
make cscope
```

Example Search:-

copy_from_user	kfifo.h
copy_to_user	list.h
printk	
container_of	
alloc_chrdev_region	

```
struct task_struct {
struct file_operations {
struct inode_operations {
struct cdev {
```

Online LXR Tools ==> "Linux LXR"

lxr.linux.no
elixir.bootlin.com

Additional:- Linux Kernel Driver Database

```
struct sample {
  int x;
  int y;
  int z;
}
```

* system call to pass structure from user space to kernel space
* system call to retrieve structure from kernel space to user space

```
struct sample s1;
s1.x=10; s1.y=20; s1.z=30;
testcall(&s1);

//SYSCALL_DEFINE1(testcall, const struct sample *, ptr)
long sys_testcall(const sample* ptr) {
        struct sample temp;
        ret=copy_from_user(&temp, ptr, sizeof(struct sample));
        if(ret) {                    }
        //print temp.x, temp.y, temp.z
}
```

```
struct sample s1;
testcall(&s1);

//system call will fill structure member
```

All process attributes:-	struct task_struct in sched.h current macro, adddress of struct task_struct instance of active process
Process Traversal Hints:-	for_each_process, init_task, next_task

----

* Simple system call
* Integer params, return integer
* Passing strings
* Filling back string , e.g. echo back
* Passing structures
* Filling structures (like return)
* Return pid of calling process
* Log various process atrributes
* Traverse process list, log (print) few attributes