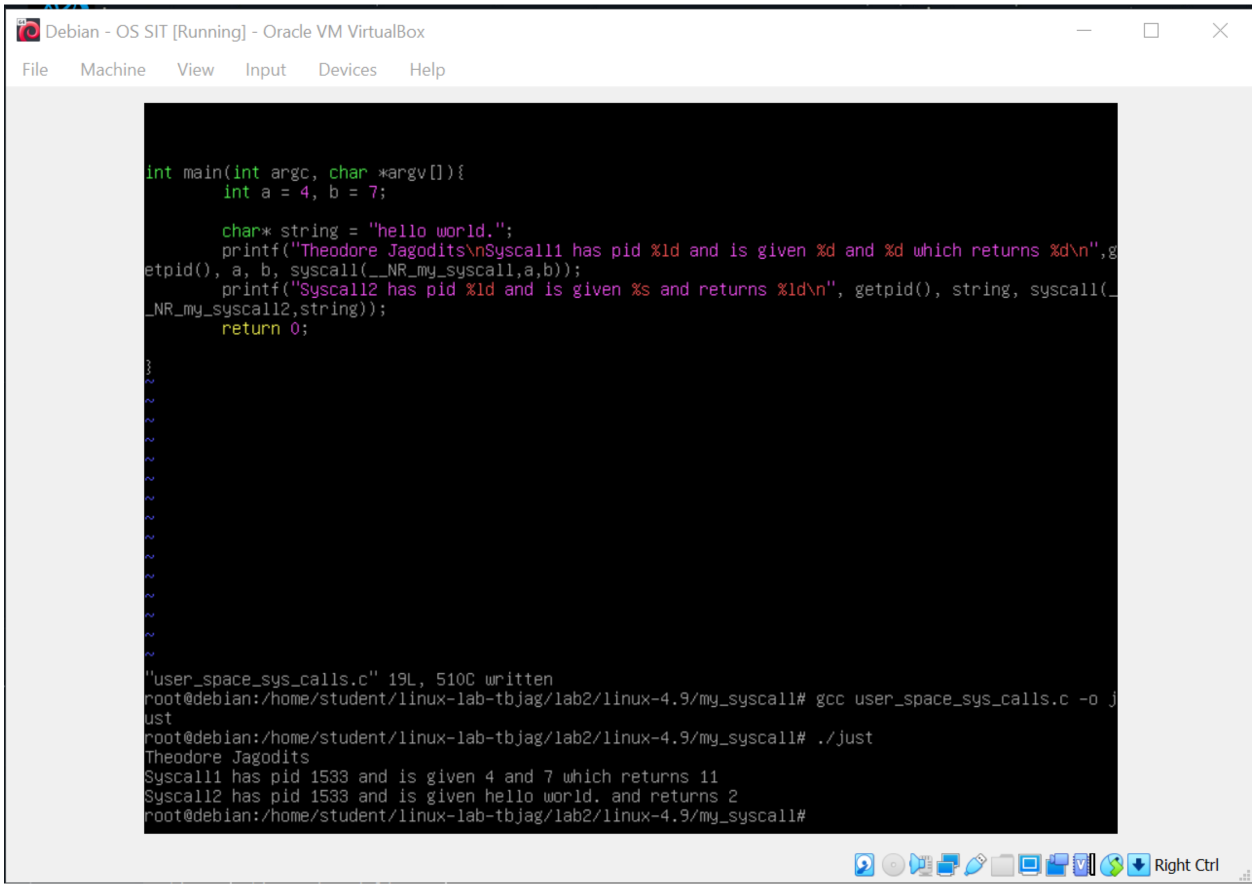


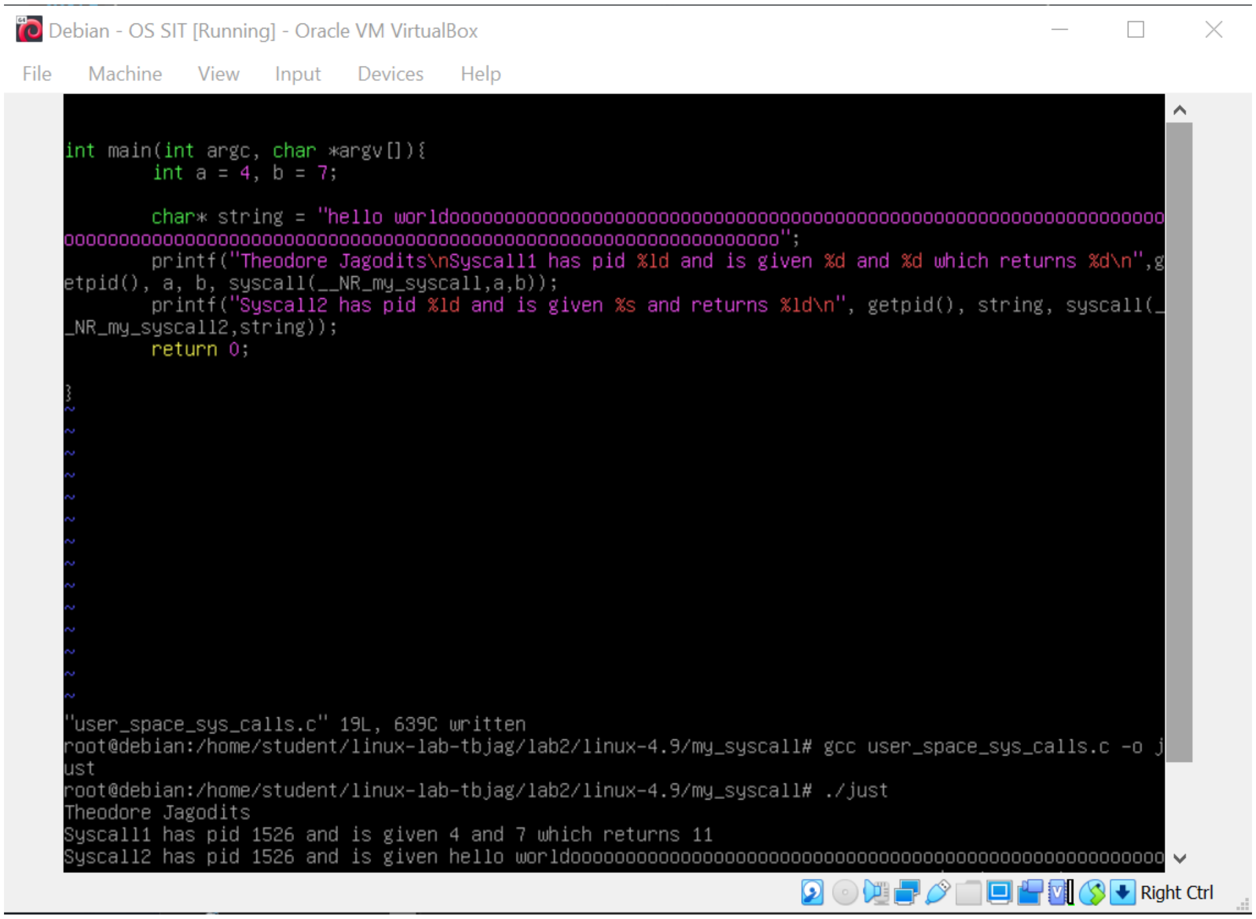
Debian - OS SIT [Running] - Oracle VM VirtualBox

```
int main(int argc, char *argv[]){
        int a = 4, b = 7;

        char* string = "hello world.";
        printf("Theodore Jagodits\nSyscall1 has pid %ld and is given %d and %d which returns %d\n",g
etpid(), a, b, syscall(__NR_my_syscall,a,b));
        printf("Syscall2 has pid %ld and is given %s and returns %ld\n", getpid(), string, syscall(_
_NR_my_syscall2,string));
        return 0;

}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"user_space_sys_calls.c" 19L, 510C written
root@debian:/home/student/linux-lab-tbjag/lab2/linux-4.9/my_syscall# gcc user_space_sys_calls.c -o j
ust
root@debian:/home/student/linux-lab-tbjag/lab2/linux-4.9/my_syscall# ./just
Theodore Jagodits
Syscall1 has pid 1533 and is given 4 and 7 which returns 11
Syscall2 has pid 1533 and is given hello world. and returns 2
root@debian:/home/student/linux-lab-tbjag/lab2/linux-4.9/my_syscall#
```

Less than 128 byte array

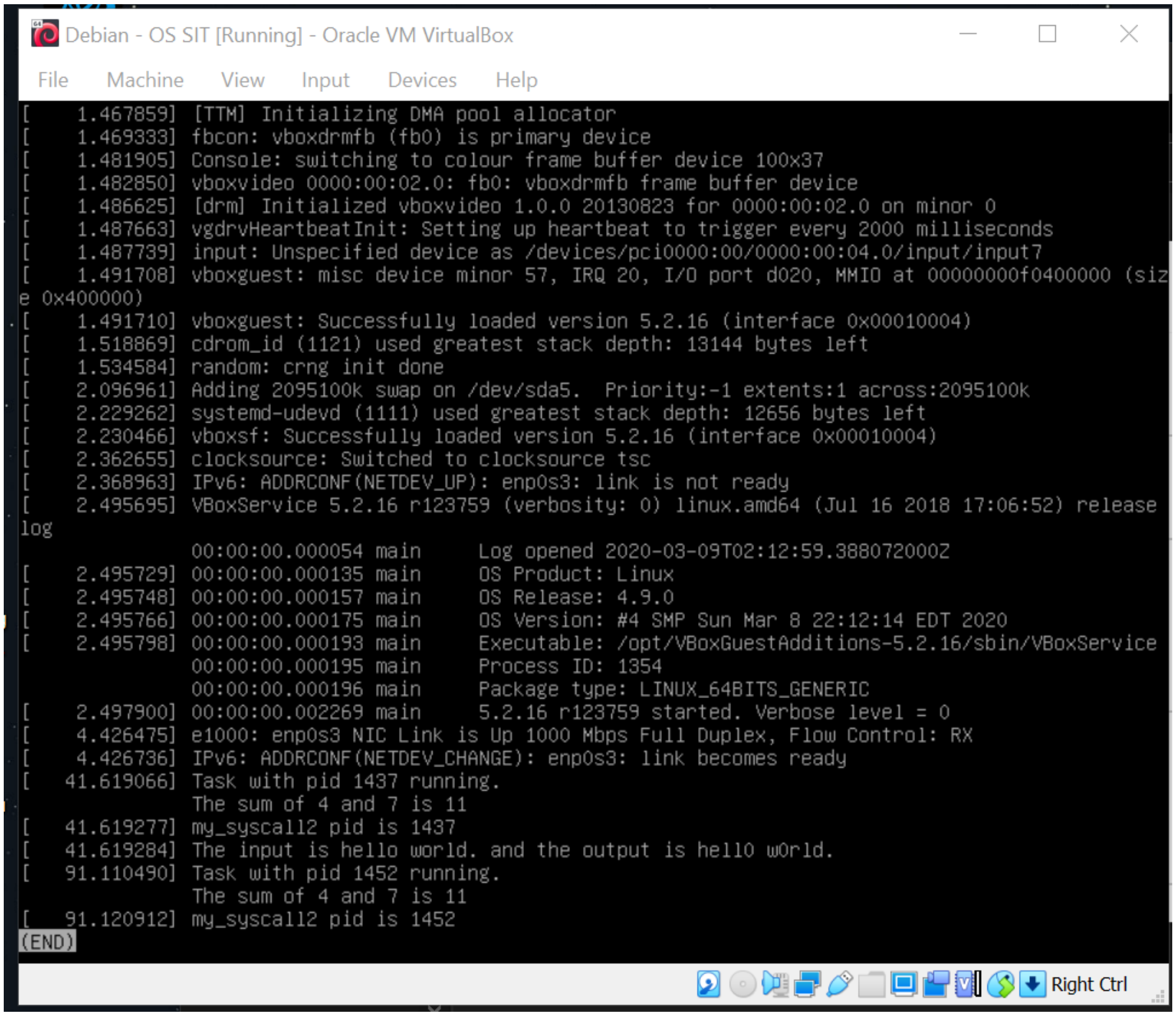

Debian - OS SIT [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
int main(int argc, char *argv[]){
        int a = 4, b = 7;

        char* string = "hello worldooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo";
        printf("Theodore Jagodits\nSyscall1 has pid %ld and is given %d and %d which returns %d\n",g
etpid(), a, b, syscall(__NR_my_syscall,a,b));
        printf("Syscall2 has pid %ld and is given %s and returns %ld\n", getpid(), string, syscall(_
_NR_my_syscall2,string));
        return 0;

}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"user_space_sys_calls.c" 19L, 639C written
root@debian:/home/student/linux-lab-tbjag/lab2/linux-4.9/my_syscall# gcc user_space_sys_calls.c -o j
ust
root@debian:/home/student/linux-lab-tbjag/lab2/linux-4.9/my_syscall# ./just
Theodore Jagodits
Syscall1 has pid 1526 and is given 4 and 7 which returns 11
Syscall2 has pid 1526 and is given hello worldoooooooooooooooooooooooooooooooooooooooooooooooooooooo
```

greater than 128 byte array


Debian - OS SIT [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
[    1.467859] [TTM] Initializing DMA pool allocator
[    1.469333] fbcon: vboxdrmfb (fb0) is primary device
[    1.481905] Console: switching to colour frame buffer device 100x37
[    1.482850] vboxvideo 0000:00:02.0: fb0: vboxdrmfb frame buffer device
[    1.486625] [drm] Initialized vboxvideo 1.0.0 20130823 for 0000:00:02.0 on minor 0
[    1.487663] vgdrvHeartbeatInit: Setting up heartbeat to trigger every 2000 milliseconds
[    1.487739] input: Unspecified device as /devices/pci0000:00/0000:00:04.0/input/input7
[    1.491708] vboxguest: misc device minor 57, IRQ 20, I/O port d020, MMIO at 00000000f0400000 (siz
e 0x400000)
[    1.491710] vboxguest: Successfully loaded version 5.2.16 (interface 0x00010004)
[    1.518869] cdrom_id (1121) used greatest stack depth: 13144 bytes left
[    1.534584] random: crng init done
[    2.096961] Adding 2095100k swap on /dev/sda5.  Priority:-1 extents:1 across:2095100k
[    2.229262] systemd-udevd (1111) used greatest stack depth: 12656 bytes left
[    2.230466] vboxsf: Successfully loaded version 5.2.16 (interface 0x00010004)
[    2.362655] clocksource: Switched to clocksource tsc
[    2.368963] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[    2.495695] VBoxService 5.2.16 r123759 (verbosity: 0) linux.amd64 (Jul 16 2018 17:06:52) release
log
               00:00:00.000054 main     Log opened 2020-03-09T02:12:59.388072000Z
[    2.495729] 00:00:00.000135 main     OS Product: Linux
[    2.495748] 00:00:00.000157 main     OS Release: 4.9.0
[    2.495766] 00:00:00.000175 main     OS Version: #4 SMP Sun Mar 8 22:12:14 EDT 2020
[    2.495798] 00:00:00.000193 main     Executable: /opt/VBoxGuestAdditions-5.2.16/sbin/VBoxService
               00:00:00.000195 main     Process ID: 1354
               00:00:00.000196 main     Package type: LINUX_64BITS_GENERIC
[    2.497900] 00:00:00.002269 main     5.2.16 r123759 started. Verbose level = 0
[    4.426475] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[    4.426736] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   41.619066] Task with pid 1437 running.
               The sum of 4 and 7 is 11
[   41.619277] my_syscall2 pid is 1437
[   41.619284] The input is hello world. and the output is hell0 w0rld.
[   91.110490] Task with pid 1452 running.
               The sum of 4 and 7 is 11
[   91.120912] my_syscall2 pid is 1452
(END)
```

Output of both runs in order, in the log

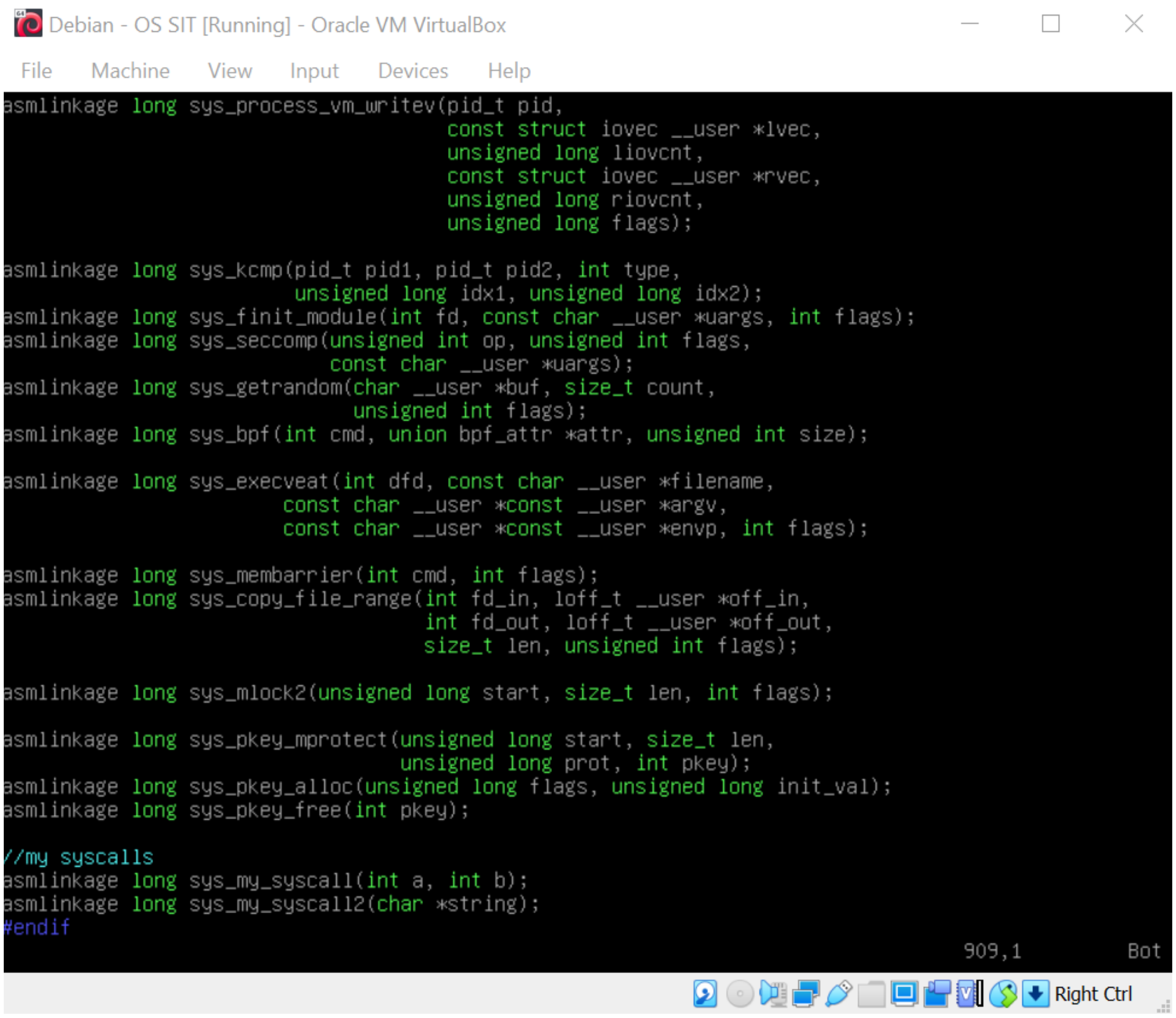
Report:

There were a few steps involved making the syscalls. First was creating the appropriate c file for the syscall. Using built in SYSCALL_DEFINE1/2 I created the functions and put them in my own folder. Then I changed the tables for arch/x86/entry/syscalls/syscall_64.tbl and added 2 entries for my two syscalls. Then I changed the include/linux/syscalls.h file by adding two function declarations.

Then I had to edit the kernel makefile and build my own include path by changing the core-y in the makefile and obj-y += my_syscall.o to include it in the system.

After that compile with the new kernel and run it.

NOTE: could not retrieve the /include/linux/syscalls.h providing screenshot, and is in my git



```
asmlinkage long sys_process_vm_writev(pid_t pid,
                                      const struct iovec __user *lvec,
                                      unsigned long liovcnt,
                                      const struct iovec __user *rvec,
                                      unsigned long riovcnt,
                                      unsigned long flags);

asmlinkage long sys_kcmp(pid_t pid1, pid_t pid2, int type,
                         unsigned long idx1, unsigned long idx2);
asmlinkage long sys_finit_module(int fd, const char __user *uargs, int flags);
asmlinkage long sys_seccomp(unsigned int op, unsigned int flags,
                            const char __user *uargs);
asmlinkage long sys_getrandom(char __user *buf, size_t count,
                              unsigned int flags);
asmlinkage long sys_bpf(int cmd, union bpf_attr *attr, unsigned int size);

asmlinkage long sys_execveat(int dfd, const char __user *filename,
                        const char __user *const __user *argv,
                        const char __user *const __user *envp, int flags);

asmlinkage long sys_membarrier(int cmd, int flags);
asmlinkage long sys_copy_file_range(int fd_in, loff_t __user *off_in,
                                    int fd_out, loff_t __user *off_out,
                                    size_t len, unsigned int flags);

asmlinkage long sys_mlock2(unsigned long start, size_t len, int flags);

asmlinkage long sys_pkey_mprotect(unsigned long start, size_t len,
                                  unsigned long prot, int pkey);
asmlinkage long sys_pkey_alloc(unsigned long flags, unsigned long init_val);
asmlinkage long sys_pkey_free(int pkey);

//my syscalls
asmlinkage long sys_my_syscall(int a, int b);
asmlinkage long sys_my_syscall2(char *string);
#endif
                                                                909,1         Bot
```