Directory Structure

- **Assignment_4**
    - main.c
    - P.c
    - C.c
    - patch
    - makefile
    - readme.pdf

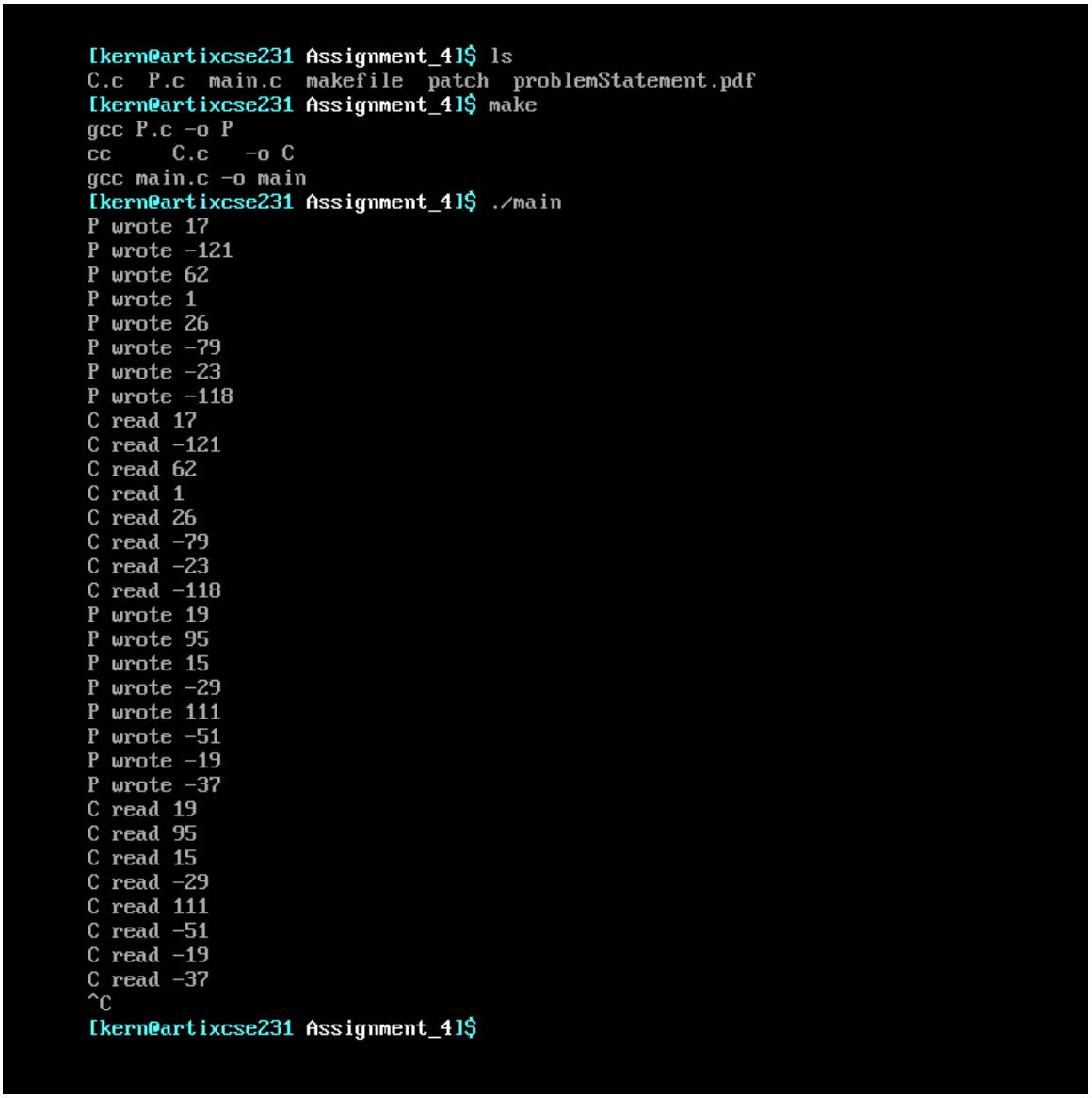
Input

No input needs to be given for running the program.

Output

The bytes wrote by P.c and then the same read by C.c using writer and reader syscalls implemented by us.

Expected Output



System Calls Used

- **queueMake -** This custom system call is used to create a queue in the kernel space. This makes use of a struct Queue defined by us.
- **writer** - This system call enqueues given 8 bytes of data to the queue made earlier in the previous step. It also takes care of synchronization between threads by using mutex implemented as struct mutex.
- **reader -** This system call dequeue the 8 bytes present in the kernel if present and then copies the same data to the specified character array in the userspace.

Compiling the Kernel

- First, go to Artix Home Directory and then mkdir Linux-Kernel.
- Then give the command or use the supplied linux-5.14.3 tarball file lynx https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.14.3.tar.xz
- Then extract tarball: tar -xvf linux-5.14.3.tar.xz
- Now put the patch file insider Linux-Kernel directory.
- Now go to the linux source by cd linux-5.14.3
- Now apply patch by the following command
- patch -p1 -R <../patch
- The modification for the custom system call has been added to the kernel source.
- Then paste the supplied .config inside the extracted directory.
- Now we need to compile the kernel using following commands:
- make
- sudo make modules_install
- sudo cp arch/x86_64/boot/bzImage /boot/vmlinuz-linux-a4
- sudo cp System.map System.map-5.14.3
- sudo mkinitcpio -k 5.14.3 -g /boot/initramfs-linux-a4.img
- sudo grub-mkconfig -o /boot/grub/grub.cfg
- sudo reboot

Compiling the Program

- After compiling and configuring the linux kernel do the following
- For compiling the program inside the Assignment_4 folder run make
- Then run the demo executable using ./main

Source Files modified

- sys.c file located at linux-5.14.3/kernel/sys.c - This file is used to add our custom system call.
- syscall_64.tbl file located at linux-5.14.3/arch/x86/entry/syscalls/syscall_64.tbl. This file is used to add our system call to the system call table.

Data Structures Used

- struct mutex - This is the structure for mutex in the kernel space included in mutex.h. Using this structure we make a mutex for synchronizing the thread behavior.
- struct Queue - This is our custom struct for creating the queue. It is defined as follows -

```
struct Queue {
        int size;
        int capacity;
        char *array;
        int front;
        int rear;
};
```

Description of how the program works

- Using syscall(450) we call our queueMake system call to make the queue in which data would be read into.
- First inside the P.c we open the "/dev/urandom" file in P.c using fopen in read mode.
- Then using fread we read 8 characters into our character array.
- We then pass this character array using writer() system call to our kernel space which then enqueues the data to the queue created earlier using syscall(448,myData) where myData is the pointer to character array
- Now the writer system call enqueues the data to the queue.
- Now the C.c using syscall(449,myData) dequeues the data to myData character array using reader() system call and prints it.