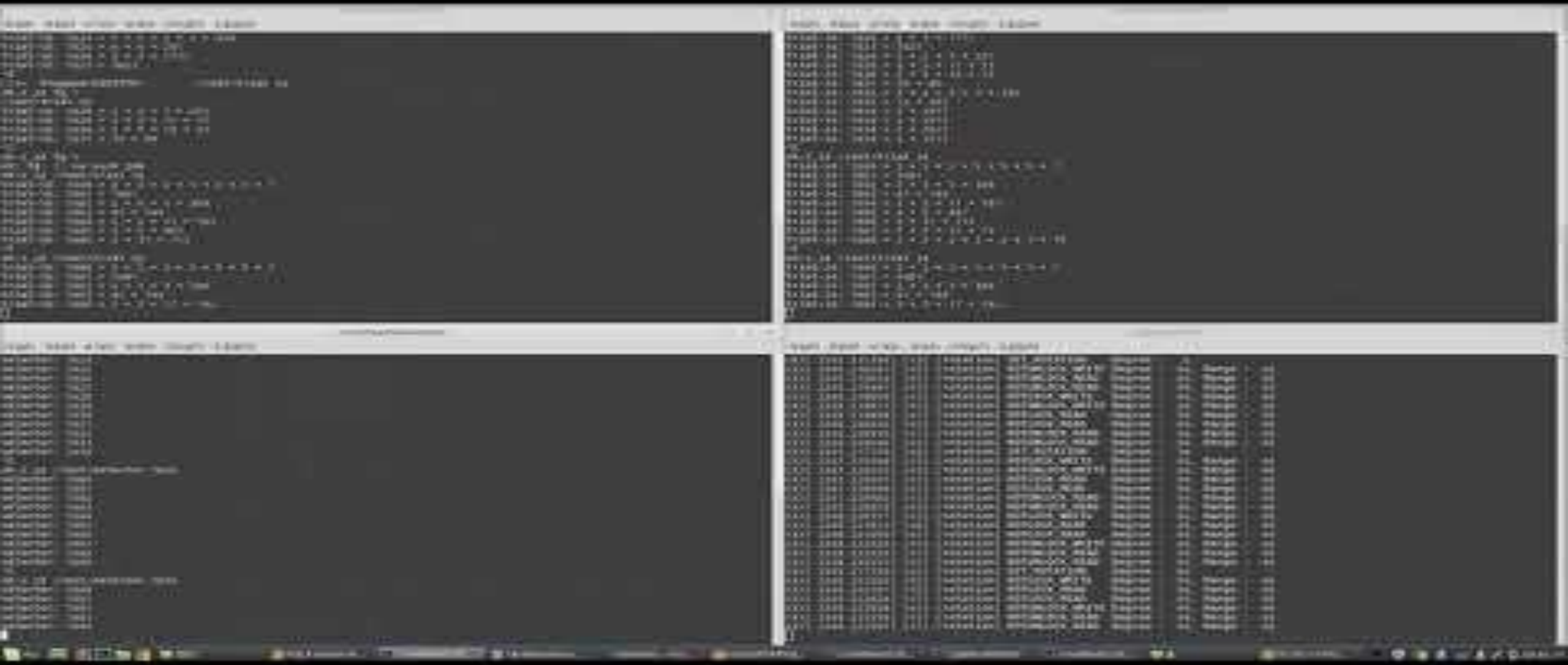
OS Project 2

OS Team 7



lock_info

```
31  struct lock_info{
32          int degree, range;
33          int mode; // 1 : read, 2 : write
34          struct completion comp;
35          struct list_head list;
36          struct task_struct *proc;
37  };
```

completion

- FIFO semaphore
- wait_for_completion_interruptible
  - -ERESTARTSYS when signal is pending
  - Wait for completion call
  - When completion is called, return 0
- Call wait_for_completion when the task wants to grab lock

Data structures

- Acquire check arrays
  - write_acq_chk[i] : # of acquired write lock contains degree i
  - read_acq_chk[i] : # of acquired read lock contains degree I
- wait_node_list_(read, write)
  - List of waiting locks
- acq_node_list_(read,write)
  - List of acquired locks

rotlock_mutex

- Mutex lock
  - Prevent simultaneous accessing of data structures(acquire check array, linked lists)
- Manage all data structures with one mutex lock

find_available

```
mutex.Lock()
chk = 0
for(lock_info in waitList_write)
        if(range(lock) contains degree)
                chk = 1
                if(task can acquire now)
                        complete()
                        mutex.unlock()
                        return
```

```
if(chk)return
for(lock_info in waitList_read)
        if(task can acquire now)
                complete()
mutex.unLock()
```

Lock and unlock

Lock

mutex.Lock()

new lock_info(new completion) T

waitList.push(T)

find_available()

return rotlock_wait()

unlock

mutex.Lock()

Find lock in acq_list match with given information

if no lock matched, return invalid

acq_chk_array[range(lock)]--

kfree()

exit_rotlock

- Always called when the process was terminated
  - Added in do_exit
- Find all lock_info matches with the process in wait lock & acquire lock lists and delete them
- If there is at least one deleted element, call find_available