

Lab 3 – Shared Memory Segments

```
Activities Terminal
melissa@ubuntu: ~/Operating_Systems/lab3$ ltps
----- Message Queues -----
key      msqid    owner    perms    used-bytes  messages
----- Shared Memory Segments -----
key      shmid    owner    perms    bytes       nattch     status
0x00000000 327680    melissa  600      524288      2          dest
0x00000000 163841    melissa  600      16777216    2          dest
0x00000000 425986    melissa  600      524288      2          dest
0x00000000 458755    melissa  600      16777216    2          dest
0x00000000 491324    melissa  600      524288      2          dest
0x00000000 524293    melissa  600      524288      2          dest
0x00000000 622598    melissa  600      524288      2          dest
0x00000000 786439    melissa  600      524288      2          dest
0x00000000 819208    melissa  600      524288      2          dest
0x00000000 851977    melissa  600      4194304     2          dest
0x00010bd9 1146890   melissa  0        30          0          -
0x00000000 1179659   melissa  600      400         0          -
0x00000000 1212428   melissa  600      400         0          -
0x00000000 1245197   melissa  600      400         0          -
----- Semaphore Arrays -----
key      semid     owner    perms    nsens
melissa@ubuntu:~/Operating_Systems/lab3$ ./shm 327680
ID: 327680
KEY: 0
MODE: rw-----
OWNER_UID: 2004
SIZE: 524288
ATTACHES: 2
melissa@ubuntu:~/Operating_Systems/lab3$ ./shm 425986
ID: 425986
KEY: 0
MODE: rw-----
OWNER_UID: 2163
SIZE: 524288
ATTACHES: 2
melissa@ubuntu:~/Operating_Systems/lab3$ ./shm
ID: 1277966
KEY: 0
MODE: rw-----
OWNER_UID: 5017
SIZE: 400
ATTACHES: 0
melissa@ubuntu:~/Operating_Systems/lab3$
```

In order to run and compile my program, first run “make” to check if any updates to the C file have been made. This will compile the program and create an executable to be run on command. Since this program finds the ID, key, mode, owner_uid, size, and attaches, we want to choose a segment to pull these values from. In order to choose a shared memory segment, run the command “ipcs” to view shared memory segments running on your computer and take the shmid of one segment. We can now execute the program that was written using “./shm” followed by the chosen shmid. As seen above, when inputting the shmid, the program finds the wanted information from the id provided. If no shmid is provided, the program generates a random one and executes the program normally.

This program finds the id by either taking user input or generating its own id by using the function shmget(), which allocates a System V shared memory segment. Assuming the id exists, using the function shmctl(), which checks if the id is a valid segment, the mode is then found by accessing the data structures shm_id_ds and ipc_perm. Shm_id_ds contains the variables struct ipc_perm shm_perm, size_t shm_segsz, time_t shm_atime, time_t shm_dtime, time_t shm_ctime, pid_t shm_cpid, pid_t shm_lpid, and shmatt_t shm_nattch. In ipc_perm, which is contained in shm_id_ds, the variables key_t __key, uid_t uid, gid_t gid, uid_t cuid, gid_t cgid, unsigned short mode, and unsigned short __seq are accessible. When getting the value of mode from ipc_perm, the number obtained is then converted to a readable character such as “r” for “read,” “w” for “write,” or “rw” for “read-write.” Finally, the program then gets the remaining values wanted from their respected variables in the shm_id_ds and ipc_perm data structures and displays them to the user.