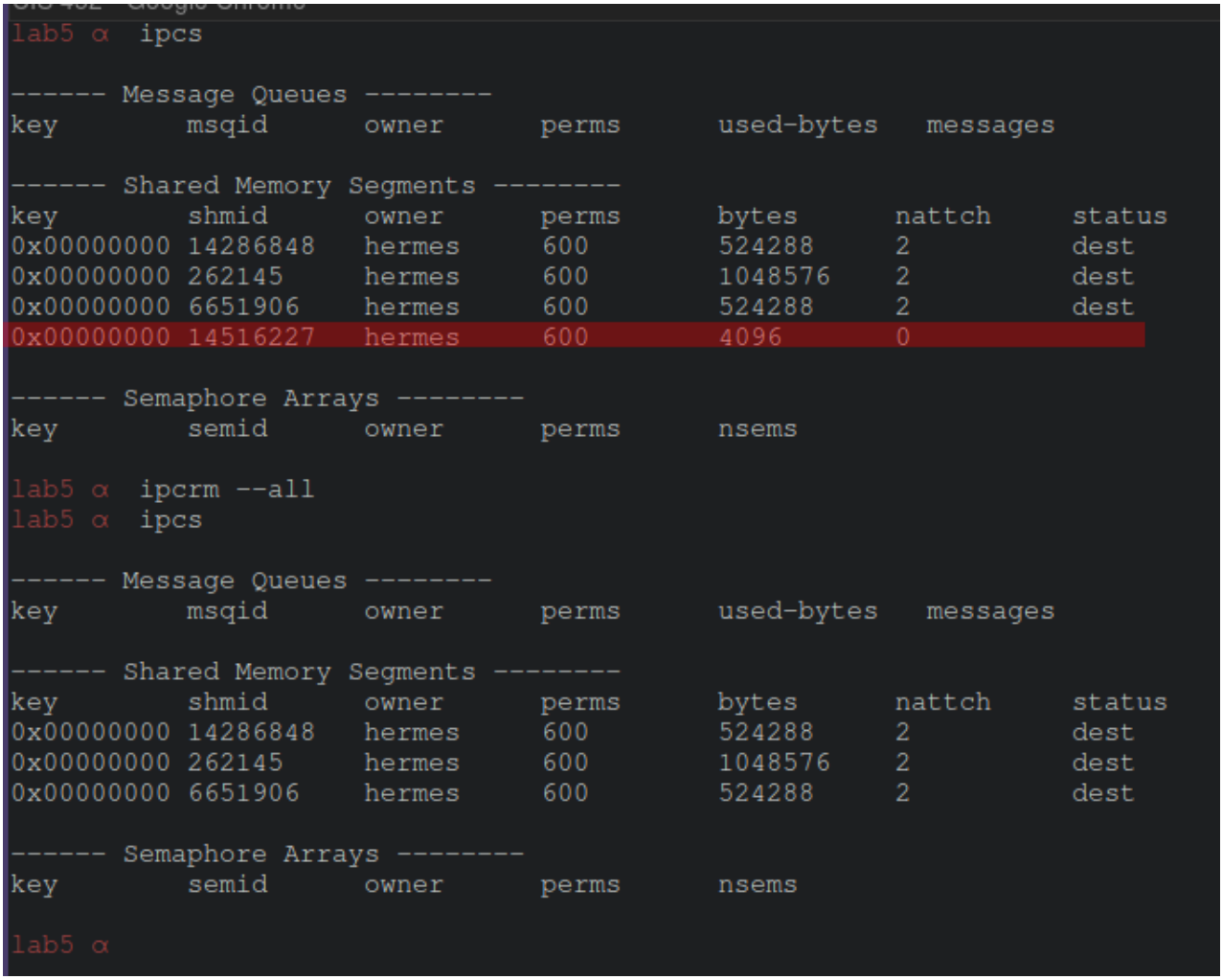
Lab 5

Jarred Parr, Alexander Fountain

1. The program first gets the shared memory id by allocating 4 bytes of space. Then, it creates a pointer by attaching to the newly created shared memory by id. Afterwards, it prints out the shared memory pointer at its originally allocated location and then the location plus an additional 4 bytes (the size of `FOO`). The additional 4 bytes point to the end location of the memory in this allocation.
2. The shmget function call takes 3 aguments, the key, the size, and the flag.
    1. The key keeps track of the memory location of the data and points to it. This can be shared to different variables.
    2. The size argument takes the size in bytes for the memory segment and allocates for it.
    3. The flags do different things.
        1. `IPC_CREAT` will create a new memory segment.
        2. `S_IRUSR` specifies the access level permission of this shared data, same with `S_IWUSR`.
        3. `IPC_EXCL` is used to ensure that the `IPC_CREAT` call creates the memory segment, if the segment already exits, then the call will fail
        4. `SHM_HUGETLB` Allocates a segment using "huge pages"
        5. `SHM_HUGE_2MB`, `SHM_HUGE_1GB` - Used in conjunction with `SHM_HUGETBL` to select alternative `hugetlb` page sizes.
3. The shmctl function performs commands onto the existing shared memory allocation defined by shmget. Two examples of this would be the `IPC_RMID` like we see in the example program. This command will specify that we are going to remove the shared memory identifier fro the system and deallocate the shared memory. Another use is the `IPC_INFO` flag. This retuns information about the system wide memory limits into a struct. With this we can determine shared memory limits and parameters, this gives the programmer more knowledge when it comes to finally performing the allocation, or just working with shared memory in general.
4. When running sample1part4.c we can see that the shared memory is 4096 bytes.
5.



Project Code

read.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <signal.h>
#include <string.h>
#define BIT_SIZE 4096

void sig_handler(int);

int shmid;
char* shm_ptr;
key_t key;

int main() {
  signal(SIGINT, sig_handler);
```

```
  key = ftok("f", 3);

  if ((shmid = shmget (key, BIT_SIZE, IPC_CREAT|S_IRUSR|S_IWUSR)) < 0) {
    perror("Failed to make shared memory");
    return EXIT_FAILURE;
  }

  if ((shm_ptr = shmat(shmid, 0, 0)) == (void*) -1) {
    perror("Can't attach");
    return EXIT_FAILURE;
  }

  for (;;) {
    if (*shm_ptr != '#') {
      printf("%s\n", shm_ptr);
      *shm_ptr = '#';
    }
  }

  return EXIT_SUCCESS;
}

void sig_handler(int i) {
  printf("Interrupted");

  if (shmctl(shmid, IPC_RMID, NULL) < 0) {
    perror("What??? We can't deallocate?!?! RUN, RUN NOW!!!");
    exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```

write.c

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/types.h>
#define BIT_SIZE 4096

void sig_handler(int);

int shmid;
char* shm_ptr;
key_t key;

int main() {
```

```
  signal(SIGINT, sig_handler);

  key = ftok("f", 3);

  if ((shmid = shmget(key, BIT_SIZE, 0666|IPC_CREAT)) < 0) {
    perror("Failed to make shared memory");
    exit(EXIT_FAILURE);
  }

  if ((shm_ptr = shmat(shmid, 0, 0)) == (void*) -1) {
    perror("Failed to attach");
    exit(EXIT_FAILURE);
  }

  for (;;) {
    if (*shm_ptr == '#') {
      printf("Whatcha wanna say? ");
      scanf("%s", shm_ptr);
      printf("\n");
    }
  }

  return EXIT_SUCCESS;
}

void sig_handler(int i) {
  printf("Interrupt called");

  if (i == SIGINT) {
    if (shmdt(shm_ptr) < 0) {
      perror("Failed to let go\n");
    }
  }

  exit(EXIT_SUCCESS);
}
```

Screenshot of output