Lab 3. Shared-memory Segments

Most UNIX and Linux systems provide the **ipcs** command. This command lists the status of various POSIX inter-process communication mechanisms, including shared-memory segments. Much of the information for the command comes from the data structure `struct shmid_ds`, which is available in the `/usr/include/sys/shm.h` file. Some of the fields in this structure include:

- `int shm_segsz` — size of the shared-memory segment
- `short shm_nattch` — number of attaches to the shared-memory segment
- `struct ipc_perm shm_perm` — permission structure of the shared-memory segment

The `struct ipc_perm` data structure (which is available in the file `/usr/include/sys/ipc.h`) contains the fields:

- `unsigned short uid` —identifier of the user of the shared-memory segment
- `unsigned short mode` —permission modes
- `key_t key` (on Linux systems, `__key`) —user-specified key identifier

(For more information, read http://linux.die.net/man/2/shmctl)

The permission modes are set according to how the shared-memory segment is established with the `shmget()` system call. Permissions (https://en.wikipedia.org/wiki/File_system_permissions) are identified according to the following:

mode	meaning
0400	Read permission of owner.
0200	Write permission of owner.
0100	Execute permission of owner.
0040	Read permission of group.
0020	Write permission of group.
0010	Execute permission of group.
0004	Read permission of world.
0002	Write permission of world.
0001	Execute permission of world.

Permissions can be accessed by using the bitwise `AND` operator `&`. For example, if the statement `mode & 0400` evaluates to "true," the permission mode gives read permission to the owner of the shared-memory segment.

A shared-memory segment can be identified according to a user-specified key or according to the integer value returned from the `shmget()` system call, which represents the integer identifier of the shared-memory segment created. The `shmid_ds` structure for a given integer segment identifier can be obtained with the `shmctl()` system call. If successful, `shmctl()` returns 0; otherwise, it returns -1 indicating an error condition (the global variable `errno` can be accessed to determine the error condition.)

```
#include <stdio.h>
#include <unistd.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char *argv[]) {
        int segment_id;                 // identifier of the shared memory segment
        unsigned short mode;            // permissions of the segment
        struct shmid_ds shmbuffer;      // the shared memory segment

// Step 1: Create a new shared memory segment using shmget

        [Your code goes here. Check course slides to see how to use shmget.]

// Step 2: Retrieve the information of the segment
        if (shmctl(segment_id, IPC_STAT, &shmbuffer) == -1) {
                printf("Unable to access segment %d\n",segment_id);
                return 0;
        }
// Step 3: output information about the segment in the required format

        [Your code that outputs Segment ID, Key, Owner UID, size, Number of
        Attaches goes here.]

       // Output mode in the right format
        mode = shmbuffer.shm_perm.mode;

        /** OWNER */
        if (mode & 0400)
                printf("r");
        else
                printf("-");
        if (mode & 0200)
                printf("w");
        else
                printf("-");
        if (mode & 0100)
                printf("a");
        else
                printf("-");

        /** GROUP */
        if (mode & 0040)
                printf("r");
        else
                printf("-");
        if (mode & 0020)
                printf("w");
        else
                printf("-");
        if (mode & 0010)
                printf("a");
        else
                printf("-");

        /** WORLD */
        if (mode & 0004)
```

```
                printf("r");
        else
                printf("-");
        if (mode & 0002)
                printf("w");
        else
                printf("-");
        if (mode & 0001)
                printf("a");
        else
                printf("-");
        …

// Step 4: Create a new process using fork

        [Your code goes here.]

// Step 5: The child process sends a message to the parent process via the
//         shared memory segment created in Step 1 and the parent prints out
//         the message it received from the child process

        [Your code goes here.]
```

Complete the above C program to create a shared-memory segment and use it for inter-process communication. Your code will also call `shmctl()` to obtain its `shmid_ds` structure and output the following values of the shared-memory segment in the format shown in Sample Output:

- Segment ID
- Key
- Mode
- Owner UID
- Size
- Number of attaches

Submissions

- (80%) Well-commented source code
- (20%) Write a report that
    - describes how to compile and run your program
    - includes screenshots of your running program

Sample Output: