Shared Memory

http://www.makelinux.net/alp/035
http://www.csl.mtu.edu/cs4411.ck/www/NOTES/process/shm/shmat.html

Semaphores:
http://csit.udc.edu/~byu/UDC3529315/Lecture6.pdf
http://stackoverflow.com/questions/6328706/semaphores-for-process-synchronization

- One of the simplest inter-process communication (IPC) methods

- Pros: Shared memory is fast to read and write
  Shared memory is the fastest form of inter-process communication because all processes share the same piece of memory. Access to this shared memory is as fast as accessing a process's non-shared memory, and it does not require a system call or entry to the kernel. It also avoids copying data unnecessarily.

- Cons: You must provide process synchronization to access shared memory
  Use semaphores. Also, for multiple processes to use a shared segment, they must make arrangements to use the same key.

- Flow: allocate →attach → detach → de-allocate the shared memory
  To use a shared memory segment, one process must allocate the segment. Then each process desiring to access the segment must attach the segment. After finishing its use of the segment, each process detaches the segment. At some point, one process must de-allocate the segment.

- **same segment identifier, different attached address spaces**

- Allocation of shared memory: shmget ("SHared Memory GET")

- Attachment of shared memory: shmat ("SHared Memory ATtach")
  Children processes created by calls to fork inherit attached shared segments; they can detach the shared memory segments, if desired.

- Detachment of shared memory: shmdt (“SHared Memory DeTach”)

- Controlling and De-allocating Shared Memory: shmctl (“Shared Memory ConTroL”)
  shmctl call returns information about a shared memory segment and can modify it

  To obtain information about a shared memory segment, pass IPC_STAT as the second argument and a pointer to a struct shmid_ds.

  To remove a segment, pass IPC_RMID as the second argument, and pass NULL as the third argument. The segment is removed when the last process that has attached it finally detaches it.

Listing 5.1 (*shm.c*) Exercise Shared Memory

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

int main ()
```

```
{
  int segment_id;                          //the identifier of the segment id
 char* shared_memory;                //the pointer of type char* that points to the shared memory
 struct shmid_ds    shmbuffer;    //a structure for obtaining information about a shared memory segment by calling shmctl()
 int segment_size;

 //0x6400 bytes, integral multiples of Linux system’s page size(getpagesize→4KB)
 const int shared_segment_size = 0x6400;

  /* Allocate a shared memory segment.   */
 //return value: a segment identifier
 segment_id = shmget (IPC_PRIVATE,            //Its first parameter is an integer key that specifies which segment to create. Using the special constant IPC_PRIVATE as the
key value guarantees that a brand new memory segment is created
                                       shared_segment_size,    //specify the size of shared memory
                                       IPC_CREAT | IPC_EXCL | S_IRUSR | S_IWUSR);    //IPC_CREAT indicates that a new segment should be created;    IPC_EXCL is always used
with IPC_CREAT, causes shmget to fail if a segment key is specified that already exists; S_IRUSR and S_IWUSR specify read and write permissions for the owner of the
shared memory segment


 /* Attach the shared memory segment.   */
//same segment identifier, different attached address spaces
 //return value: the address of the attached shared memory segment.
 shared_memory = (char*) shmat (segment_id, //specify the identifier of shared memory segment, which is returned by shmget()
                                                         0,    // a pointer that specifies where in your process's address space you want to map the shared memory; if you specify NULL, Linux
will choose an available address.
```

```
                                        0)    //If the flag is SHM_RDONLY, this shared memory is attached as a read-only memory; otherwise like 0, it is readable and writable.
printf ("shared memory attached at address %p\n", shared_memory);

/* Determine the segment's size. */
shmctl (segment_id,    //the identifier of the shared memory segment
        IPC_STAT,      //To obtain information about a shared memory segment, pass IPC_STAT as the second argument and a pointer to a struct shmid_ds.
        &shmbuffer); //a structure for obtaining information about a shared memory segment

segment_size   =            shmbuffer.shm_segsz; //obtain the segment size from the structure
printf ("segment size: %d\n", segment_size);

/* Write a string to the shared memory segment.   */
sprintf (shared_memory, "Hello, world.");

/* Detach the shared memory segment.   */
shmdt (shared_memory);    //pass the address of the shared memory segment

 /* Reattach the shared memory segment, at a different address.   */
//same segment identifier, different attached address spaces
shared_memory = (char*) shmat ( segment_id,            //specify the identifier of shared memory segment
                                (void*) 0x5000000,    // a pointer that specifies where in your process's address space you want to map the shared memory; if you
specify NULL, Linux will choose an available address.
                                0);                   //0, it is readable and writable.

printf ("shared memory reattached at address %p\n", shared_memory);
```

```
  /* Print out the string from shared memory.   */
  printf ("%s\n", shared_memory);

  /* Detach the shared memory segment.   */
  shmdt (shared_memory);    //pass the address of the shared memory segment

   /* Deallocate the shared memory segment.   */
  shmctl (segment_id,
              IPC_RMID,        //To remove a segment, pass IPC_RMID as the second argument, and pass NULL as the third argument.
              0);

   return 0;
}
```

Debugging

The ipcs command provides information on interprocess communication facilities, including shared segments. Use the -m flag to obtain information about shared memory. For example, this code illustrates that one shared memory segment, numbered 1627649, is in use:

```
% ipcs -m
------ Shared Memory Segments -------
key          shmid     owner     perms     bytes     nattch     status
0x00000000    1627649    user     640       25600         0
```

If this memory segment was erroneously left behind by a program, you can use the ipcrm command to remove it.

```
% ipcrm shm 1627649
```