

Write a LINUX program that will create a shared memory segment of size 408, attach it to the current process. An array of integers will be created in this segment. This array will be used to create a solution to the bounded buffer problem.

This function creates the segment.

```
int MyLastFour=1111,SegSize=408;  
int MyShmid = shmget(MyLastFour,SegSize,IPC_CREAT|0x1c0);
```

MyLastFour must contain the last four digits of your social security number, SegSize is the size of the memory segment you must create, and IPC_CREAT|0x1c0 is just something you need.

The following function attaches the segment and gives you access to it by returning its address.

```
int * Buffer = shmat(MyShmid,NULL,0);
```

MyShmid is the return value from shmget. NULL and 0 are constants.

The last two elements of the array will be SendPos and RecPos respectively, so use the following definitions:

```
int *SendPos = Buffer+100;  
int *RecPos = Buffer+104;
```

The following function detaches the segment.

```
shmdt(MyAddr);
```

The following function destroys the memory segment. If you forget this the memory segment won't go away. It will hang around until you finally destroy it. (It's a benign problem.)

```
shmctl(MyShmid,IPC_RMID,NULL);
```

MyShmid is the return value from shmget. IPC_RMID and NULL are constants.

You need to put the following includes at the beginning of your program. (In addition to the usual stuff.) (This comes after the usual stuff.)

```
#include <sys/ipc.h>  
#include <sys/shm.h>  
#include <errno.h>  
#include <unistd.h>
```

Create two processes. Your program is the sender. The receiver must be created using fork. The sender should generate 200 random integers and the receiver should print them, 10 per line.