

Write two LINUX programs that will exchange several messages and quit. Call these programs “Reader” and “Writer”. Allocate a shared memory segment using the last four digits of your social security number as the ID, just as you did in the two previous assignments. However, in this case the address returned by `shmat` will be treated as an integer array with 22 elements. (SegSize must be at least 88, or “sizeof(int)*22”.) This array will be used to implement a solution to the bounded buffer problem. The Writer program will insert 100 random integers (function `rand( )`) into the bounded buffer, and the Reader program will read the numbers and print them, one number per line. Element 0 of the shared array will be used to indicate the position at which the next element is to be inserted. Element 1 of the shared array will be used to indicate the position at which the next element is to be read. The remaining elements 2-21, will be used to implement a 20 element buffer. To simplify the routine, you might want to store the address of the array and positions doing something like the following:

```
int * MyAddr = shmat(MyShmid, NULL, 0);
int * WritePos = MyAddr;
int * ReadPos = MyAddr+1;
int * Array = MyAddr+2;
```

Remember that when you add an integer to a pointer it increments the pointer by that number of *units*, not that number of *bytes*. (A unit is the size of the thing pointed to.)

DO NOT use a buffer larger than 20!

Remember that to run your programs you must invoke them separately, and place an ampersand (&) after the first like this:

```
Prompt> Writer &
Prompt> Reader
```

It doesn't matter which you start first.