Harold Molina-Bulla DTSC-UC3M

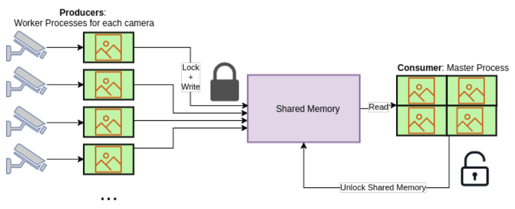
SESSION 6

SHARED MEMORY WITH LOCKS

Harold Molina-Bulla DTSC-UC3M

SHARED MEMORY

- Shared Memory is a reserved memory area, where several independent processes can read and write simultaneously.
- Advantages:
  - Fastest access from each parallel process
  - No need to exchange data between processes
- Disadvantages
  - Hazzards



SHARED MEOMORY IN MULTIPROCESSING MODULE

- There are 2 kind of shared memory classes:
  - Value: Reserve a memory to store just one possible value of data type in first parameter
    - `multiprocessing.sharedctypes.Value(`*`typecode_or_type`*`, `*`*args`*`, `*`lock=True`*`)`
  - Array: Reserve a memory area to store an array of data type defined in the third parameter
    - `multiprocessing.sharedctypes.Array(`*`typecode_or_type`*`, `*`size_or_initializer`*`, `*`*`*`, `*`lock=True`*`)`

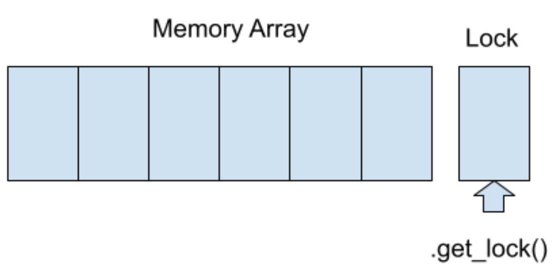
C TYPES

ctypes	sharedctypes using type	sharedctypes using typecode
c_double(2.4)	Value(c_double, 2.4)	Value('d', 2.4)
MyStruct(4, 6)	Value(MyStruct, 4, 6)	
(c_short * 7)()	Array(c_short, 7)	Array('h', 7)
(c_int * 3)(9, 2, 8)	Array(c_int, (9, 2, 8))	Array('i', (9, 2, 8))

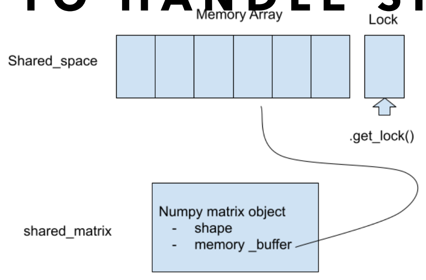
LOCK CLASS

- Lock Class is an object class that allows lock or release access to a Shared Memory position.
- Each Shared Memory object has his own lock property, and we can set on/off locally or assign an external Lock object, to be handled from other processes.
  - `lock = Lock()`
  - `x = Value(c_double, 1.0/3.0, lock=False)`
  - `s = Array('c', b'hello world', lock=lock)`

SHARED MEMORY OBJECTS



USE NUMPY TO HANDLE SHARED ARRAY



```
• def tonumpyarray(shared_space):
•     #mp_array is a shared memory array with lock
•     return np.frombuffer(mp_arr.get_obj(),dtype=np.uint8)
```

- Shared_space is an object with get_lock() property.
  - This variable reference allows lock and release access to shared memory area
- Shared_matrix is a NumPY array, with all the nparray methods.
  - Allows us to change the shape and get a matrix instead a simple vector (or even a cube of data)
- We should take care about the basic data size, and define the correct data type in both structures

RISKS

- All the three hazards
  - Read after Write (RW)
  - Write after Read (WR)
  - Write after Write (WW)
- Race Condition
  - The program runs without control, destroying the previous data stored in the shared memory area.
- Solution: locks
  - Problem with locks: should be used in the correct place