```
# -*- coding: iso-latin-1 -*-

# Il solito esempio di reader e write realizzato questa volta
# utilizzando la classe Queue dell'omonimo modulo della libreria
# standard di Python.  Come si può notare tutta la parte di semafori e
# di sincronizzazione è scomparsa.  Le parti più "difficili" sono ora
# la gestione delle opzioni e il subclassing di Queue per poter fare
# "logging".

import threading
import random
import time
import sys
import Queue

DO_SLEEP = 1
VERBOSE = 0

def show(s, file=sys.stdout):
    if VERBOSE:
        file.write(s + "\n")

class Counter:

    # La classe Counter permette di creare degli oggetti iterabili che
    # restituiscono degli interi decrescenti da value - 1 fino a 0.
    # Il metodo __ITER__ deve restituire un oggetto con un metodo NEXT
    # che restituisce i valori voluti e solleva una eccezione
    # StopIteration per terminare l'iterazione.

    def __init__(self, value):
        assert value >= 0
        self.value = value
        self.lock = threading.Lock()

    def __iter__(self):
        return self

    def next(self):
        with self.lock:
            if self.value == 0:
                raise StopIteration
            self.value -= 1
            return self.value

class DataQueue(Queue.Queue):

    # La classe DataQueue eredita dalla classe Queue del modulo
    # Queue e (ri)definisce i due metodi privati _PUT e _GET
    # semplicemente per fare logging visto che lo storage è realizzato
    # con i metodo APPEND e POPLEFT della classe originale.

    def __init__(self, maxsize=0):
        Queue.Queue.__init__(self, maxsize)

    def _put(self, item):
        id = threading.current_thread().name
        show("Writer %s put %s" % (id, item))
        self.queue.append(item)

    def _get(self):
        id = threading.current_thread().name
        item = self.queue.popleft()
        show("Reader %s got %s" % (id, item))
        return item

def sleep_between(min, max):
    diff = max - min
    time.sleep(float(min))
    time.sleep(diff * random.random())

def make_item():
    if DO_SLEEP: sleep_between(0.1, 0.2)
    return random.randint(0,100)

def use_item(i):
    if DO_SLEEP: sleep_between(0.5, 0.6)

def writer(id, queue, counter):
    for c in counter:
        i = make_item()
        queue.put(i, id)
    show("writer %3d leaving" % (id))

def reader(id, queue):
    while 1:
        try:
            c = queue.get(id)
            if not c:
                show("reader %3d leaving" % (id))
                break
        except Queue.Empty:
            show("empty queue: reader %3d leaving" % (id))
            break

def main(items_count, readers_count, writers_count):

    queue = DataQueue(items_count)
    counter = Counter(items_count)

    rr = [threading.Thread(target=reader, name="r_%0d" % id,
            args=(id, queue))
            for id in range(readers_count)]
    ww = [threading.Thread(target=writer, name="w_%0d" % id,
            args=(id, queue, counter))
            for id in range(writers_count)]

    for r in rr: r.start()
    for w in ww: w.start()
    for w in ww: w.join()
    for r in rr:
        queue.put(None, -1)
    for r in rr: r.join()
```

```
def usage():
    print("""\
Usage: multi-read-write [-i|--items-count   items_count]
                        [-w|--writers-count writers_count]
                        [-r|--readers-count readers_count]
                        [-v|--verbose]
                        [-h|--help]
    """)

if __name__ == '__main__':

    items_count = 40
    writers_count = 2
    readers_count = 5

    from getopt import gnu_getopt as get_opt

    optlist, args = get_opt(
        sys.argv,
        "hi:b:w:r:v", [
            "help",
            "item-count=",
            "writers-count=",
            "readers-count=",
            "verbose"])

    for k,v in optlist:
        if k in ["-h", "--help"]:
            usage()
            sys.exit(0)
        if k in ["-i", "--items-count"]:
            items_count = int(v)
        if k in ["-w", "--writers-count"]:
            writers_count = int(v)
        if k in ["-r", "--readers-count"]:
            readers_count = int(v)
        if k in ["-v", "--verbose"]:
            global VERBOSE
            VERBOSE = True

    show("""
    multi-read-write
        items_count   %3d
        writers_count %3d
        readers_count %3d
        """ % (
            items_count,
            writers_count,
            readers_count),
        sys.stderr)

    main(items_count, readers_count, writers_count)
```