

Barrier: synchronizes threads in stages of execution so that no thread can begin a stage until all threads finish the previous stage

```
1 operation: cross           if (not last to cross)
                                     sleep
else
                                     wakeup all
```

Semaphore: synchronized counter

```
Z ops: up (V)                 down (P)
      if (waiting)             if (value == 0)
        wakeup                  sleep()
      else                       else
        value++                 value--
```

Buffer b

Semaphore empty = b.size
fill = 0
mutex = 1

producer

```
while (true)
1) item = produce_item()
2) if (b.count == b.size)
3) sleep()
4) mutex.down()
   b.insert_item(item);
   mutex.up()
5) b.count ++
6) if (b.count == 1) consumer.wakeup()
   fill.up()
```

consumer

```
while (true)
if (b.count == 0)
sleep()
mutex.down()
item = b.remove()
mutex.up()
b.count --
if (b.count == b.size - 1)
producer.wakeup()
empty.up()
```

problem if interrupted here

fill.down()

empty.up()