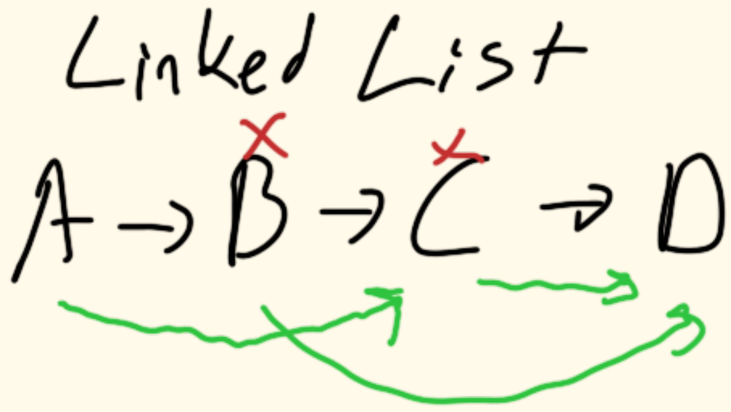
1)

a) Two deletes executed simultaneously

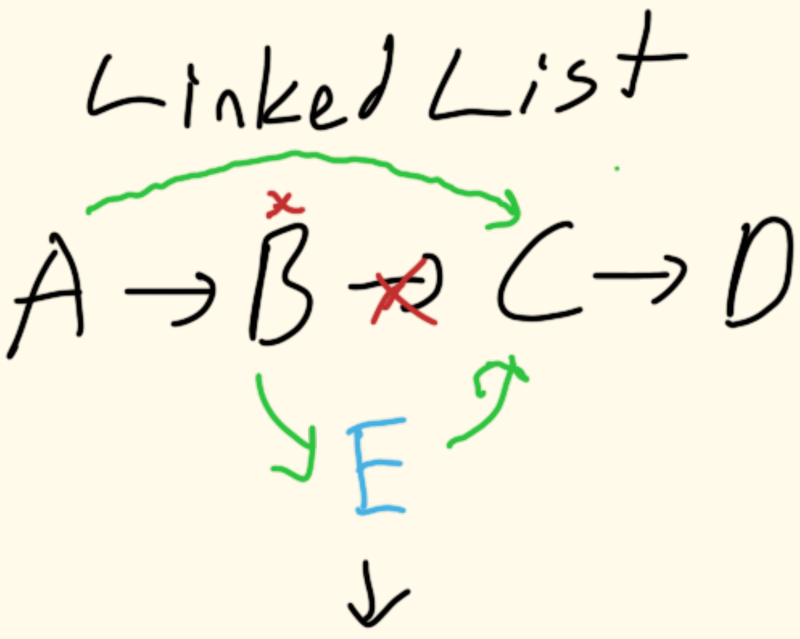


How it occur:
-Thread1 wants to delete B which is A's next
-Thread2 wants to delete C which is B's next
-Thread1 updates A.next as C (A.next.next)
-Thread2 updates B.next as D (B.next.next)

What would be the consequence:
Node C is disappeared from list even though Thread2 want to delete it. Node B is also not in the list because Thread1 removed it. Since thread2 was operating while assumşng that NodeB still existed NodeC become unreachable.

This causes memory leak. Allocad memory is no longer usable.

b) An insert and a delete executed simultaneously

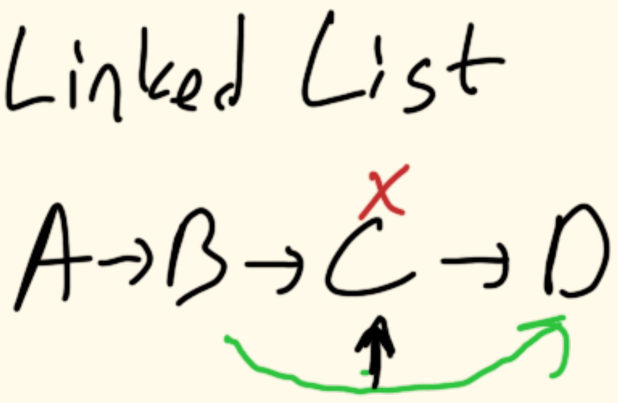


How it occur:
-Thread1 wants to insert E between B and C.
-Thread2 wants to delete B.
-Thread1 sets E.next to C (B.next).
-Thread2 sets A.next to C (B.next).
-Thread1 sets B.next to E.(There is no B)

What would be the consequence:
In this scenario removal of B causes problem by making thread1 fail to insert nodeE between B and C.

Traversals may crash. List will have a corrupted structure.

c) A member and a delete executed simultaneously

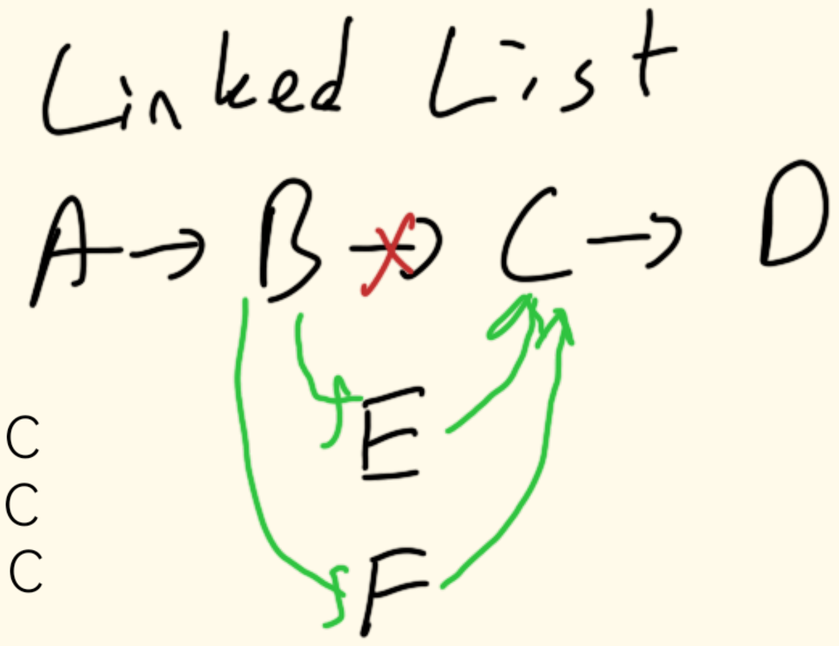


How it occur:
-Thread1 searches and finds C
-Thread2 proceed to delete C
-Thread1 attempt to reach data on C (There is no C)

What would be the consequence:
Thread1 tries to access freed memory which will cause a segmenataion fault.

Program probably will crash

d) Two inserts executed simultaneously

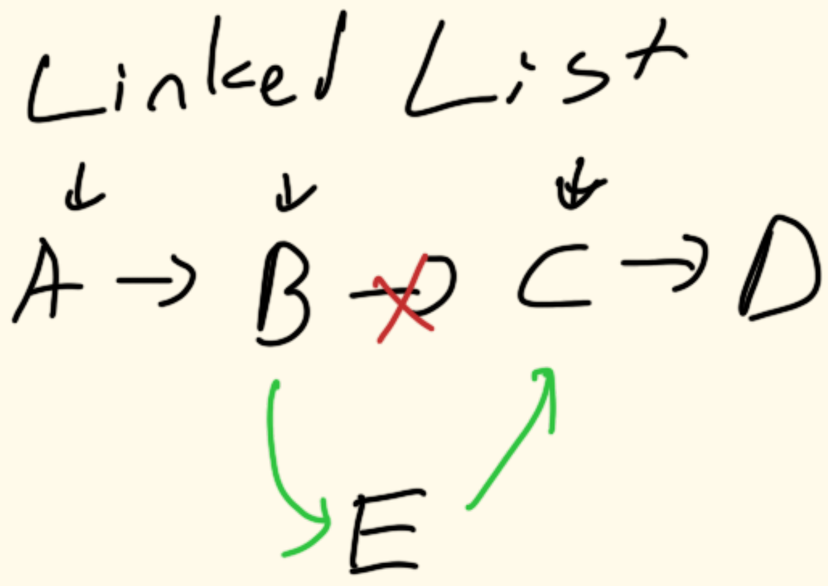


How it occur:
-Thread1 wants to insert E between B and C
-Thread2 wants to insert F between B and C
-Both threads set their new nodes' next to C
-Thread1 set B.next to E
-Thread2 set B.next to F

What would be the consequence:
One of the new nodes, either E or F, will be lost. Will cause a memory leak. List becomes corrupted.

This causes memory leak and unpredictable consequences.

e) An insert and a member executed simultaneously



How it occur:
-Thread1 wants to insert E between B and C
-Thread2 searches for C traversing from the head.
-Thread1 sets E.next to C
-Thread1 sets B.next to E.

What would be the consequence:
-It can miss the insertion of NodeE, still finds NodeC. But in another scenario if it also searches the newly inserted NodeE, it can totally miss the NodeE.
-If it traverse the NodeB while Thread1 was modifying it's next pointer traversal may be broke. Potential segmentaion fault.

Member function could return incorrectinconsistent results.

2) i)

```
omero@omero-fedora:~/Education/Multicore_Programming/Homeworks/Hw2$ ./pth_ll_rwl 8
How many keys should be inserted in the main thread?
100000
How many ops total should be executed?
1000000
Percent of ops that should be searches? (between 0 and 1)
0.99
Percent of ops that should be inserts? (between 0 and 1)
0.005
Inserted 100000 keys in empty list
Elapsed time = 6.561672e+02 seconds
Total ops = 1000000
member ops = 990131
insert ops = 5069
delete ops = 4800
```

```
omero@omero-fedora:~/Education/Multicore_Programming/Homeworks/Hw2$ ./rwlocks_imp 8
How many keys should be inserted in the main thread?
100000
How many ops total should be executed?
1000000
Percent of ops that should be searches? (between 0 and 1)
0.99
Percent of ops that should be inserts? (between 0 and 1)
0.005
Inserted 100000 keys in empty list
Elapsed time = 2.116298e+02 seconds
Total ops = 1000000
member ops = 990131
insert ops = 5069
delete ops = 4800
```

ìi)

```
omero@omero-fedora:~/Education/Multicore_Programming/Homeworks/Hw2$ ./pth_ll_rwl 8
How many keys should be inserted in the main thread?
100000
How many ops total should be executed?
1000000
Percent of ops that should be searches? (between 0 and 1)
0.9
Percent of ops that should be inserts? (between 0 and 1)
0.05
Inserted 100000 keys in empty list
Elapsed time = 1.073676e+03 seconds
Total ops = 1000000
member ops = 899573
insert ops = 50414
delete ops = 50013
```

```
omero@omero-fedora:~/Education/Multicore_Programming/Homeworks/Hw2$ ./rwlocks_imp 8
How many keys should be inserted in the main thread?
100000
How many ops total should be executed?
1000000
Percent of ops that should be searches? (between 0 and 1)
0.9
Percent of ops that should be inserts? (between 0 and 1)
0.05
Inserted 100000 keys in empty list
Elapsed time = 3.900447e+02 seconds
Total ops = 1000000
member ops = 899573
insert ops = 50414
delete ops = 50013
```

ì ìi)

```
omero@omero-fedora:~/Education/Multicore_Programming/Homeworks/Hw2$ ./pth_ll_rwl 8
How many keys should be inserted in the main thread?
100000
How many ops total should be executed?
1000000
Percent of ops that should be searches? (between 0 and 1)
0.5
Percent of ops that should be inserts? (between 0 and 1)
0.25
Inserted 100000 keys in empty list
Elapsed time = 5.169086e+03 seconds
Total ops = 1000000
member ops = 500468
insert ops = 249684
delete ops = 249848
```

```
omero@omero-fedora:~/Education/Multicore_Programming/Homeworks/Hw2$ ./rwlocks_imp 8
How many keys should be inserted in the main thread?
100000
How many ops total should be executed?
1000000
Percent of ops that should be searches? (between 0 and 1)
0.5
Percent of ops that should be inserts? (between 0 and 1)
0.25
Inserted 100000 keys in empty list
Elapsed time = 1.490028e+03 seconds
Total ops = 1000000
member ops = 500468
insert ops = 249684
delete ops = 249848
```

Elapsed times

i) base: 656s my_rw: 211s
ii) base: 1073s my_rw: 390s
iii)base: 5169s my_rw: 1490s

As can be seen from the elapsed times our implementation makes the process noticably faster.