

PA2 实验报告

191300073 AI 杨斯凡 191300073@smail.nju.edu.cn

1. 归并排序:

因为上个学期算法课我写过归并排序, 我对上个学期的代码进行了微改

```
void Merge_Sort(void* arg)
{
    int *argu = (int*)arg;
    int left = argu[0];
    int right = argu[1];

    if (left < right)
    {
        pthread_t pid1;
        pthread_t pid2;

        int arg1[2];
        int arg2[2];

        int middle;
        middle = (left + (right - 1)) / 2;
        arg1[0] = left;
        arg1[1] = middle;
        arg2[0] = middle + 1;
        arg2[1] = right;

        {
            pthread_create(&pid1, NULL, Merge_Sort, arg1);
            pthread_create(&pid2, NULL, Merge_Sort, arg2);
        }

        pthread_join(pid1, NULL);
        pthread_join(pid2, NULL);

        merge(left, middle, right);
    }
}
```

将数组起始位置参数设置成数组传入, 而进行 mergesort 的时候, 不再递归调用, 而是创建新的线程去执行, 只需等待线程完成即可。

而为了比较时间, 我在单线程的时候使用了递归调用。

可以看出, 当数组数较少的时候, 递归次数少, 线程少, 两者时间相差无几。

```
ysf@ysf0411: ~/Desktop/PA
ysf@ysf0411:~/Desktop/PA$ time ./merge
0 1 5 8 9 11 11 14 17 18
real    0m0.003s
user    0m0.003s
sys     0m0.000s
ysf@ysf0411:~/Desktop/PA$ gcc merge.c -o merge -pthread
merge.c: In function 'Merge_Sort':
merge.c:53:37: warning: passing argument 3 of 'pthread_create' from incompatible
pointer type [-Wincompatible-pointer-types]
pthread_create(&pid1, NULL, Merge_Sort, arg1);

In file included from merge.c:2:0:
/usr/include/pthread.h:198:12: note: expected 'void * (*)(void *)' but argument
is of type 'void (*)(void *)'
extern int pthread_create (pthread_t *__restrict __newthread,

merge.c:54:37: warning: passing argument 3 of 'pthread_create' from incompatible
pointer type [-Wincompatible-pointer-types]
pthread_create(&pid2, NULL, Merge_Sort, arg2);

In file included from merge.c:2:0:
/usr/include/pthread.h:198:12: note: expected 'void * (*)(void *)' but argument
is of type 'void (*)(void *)'
```

```
pointer type [-Wincompatible-pointer-types]
pthread_create(&pid2, NULL, Merge_Sort, arg2);

In file included from merge.c:2:0:
/usr/include/pthread.h:198:12: note: expected 'void * (*)(void *)' but argument
is of type 'void (*)(void *)'
extern int pthread_create (pthread_t *__restrict __newthread,

merge.c: In function 'main':
merge.c:74:32: warning: passing argument 3 of 'pthread_create' from incompatible
pointer type [-Wincompatible-pointer-types]
pthread_create(&pid, NULL, Merge_Sort, arg);

In file included from merge.c:2:0:
/usr/include/pthread.h:198:12: note: expected 'void * (*)(void *)' but argument
is of type 'void (*)(void *)'
extern int pthread_create (pthread_t *__restrict __newthread,

ysf@ysf0411:~/Desktop/PA$ time ./merge
0 1 3 3 3 6 7 10 10 10
real    0m0.005s
user    0m0.001s
sys     0m0.008s
ysf@ysf0411:~/Desktop/PA$
```

而当数组大小很大的时候，多线程在线程调度上就得花费很多时间

```
ysf@ysf0411: ~/Desktop/PA
16017 16045 16055 16056 16058 16088 16109 16110 16130 16140 16162 16185 16214 1
6225 16227 16239 16257 16266 16276 16295 16329 16363 16405 16415 16439 164
70 16499 16511 16528 16556 16605 16617 16641 16646 16676 16703 16708 16713 16741
16832 16850 16880 16886 16910 16921 16924 16926 16977 16991 16997 17011 17040 1
7042 17079 17126 17162 17166 17193 17200 17208 17278 17298 17345 17367 17383 173
97 17418 17427 17465 17475 17508 17516 17569 17592 17608 17658 17665 17666 17705
17712 17721 17727 17734 17744 17799 17812 17820 17837 17848 17898 17921 17930 1
7975 17979 17983 18013 18016 18072 18084 18093 18095 18109 18118 18118 18189 181
93 18214 18222 18222 18227 18247 18270 18282 18298 18301 18312 18314 18341 18381
18431 18444 18487 18537 18538 18548 18551 18588 18606 18607 18619 18623 18635 1
8651 18663 18680 18686 18689 18705 18742 18768 18772 18795 18807 18808 18868 189
07 18941 18951 18954 18956 18976 18977 18987 19045 19045 19045 19047 19190 19196
19213 19222 19223 19227 19258 19268 19306 19319 19322 19375 19387 19436 19453 1
9488 19501 19518 19533 19558 19564 19571 19576 19605 19636 19646 19658 19701 197
68 19779 19787 19789 19803 19821 19832 19833 19855 19855 19885 19889 19915 19916
19943
real    0m0.003s
user    0m0.003s
sys     0m0.000s
ysf@ysf0411:~/Desktop/PA$ gcc merge.c -o merge -pthread
merge.c: In function 'Merge_Sort':
merge.c:53:37: warning: passing argument 3 of 'pthread_create' from incompatible
pointer type [-Wincompatible-pointer-types]
pthread_create(&pid1, NULL, Merge_Sort, arg1);
```

单线程

```
ysf@ysf0411: ~/Desktop/PA
15130 15136 15141 15148 15185 15186 15203 15227 15240 15243 15270 15283 15348 1
5372 15393 15413 15426 15472 15506 15548 15552 15554 15631 15674 15690 15695 157
11 15744 15772 15772 15778 15785 15815 15819 15834 15845 15861 15886 15909 15933
15934 15937 15963 15985 15995 16001 16009 16020 16073 16086 16162 16180 16202 1
6213 16237 16246 16264 16277 16298 16302 16324 16331 16337 16339 16351 16352 163
64 16366 16377 16425 16427 16456 16486 16487 16528 16538 16547 16563 16597 16601
16626 16637 16682 16688 16705 16707 16745 16755 16763 16789 16823 16857 16896 1
6904 16918 16935 16940 16958 16997 17002 17006 17023 17058 17061 17069 17117 171
49 17158 17175 17219 17228 17245 17290 17308 17330 17337 17351 17403 17494 17519
17532 17557 17595 17610 17711 17733 17742 17783 17798 17804 17821 17845 17859 1
7862 17877 17884 17891 17895 17900 17912 17915 17924 17975 18002 18028 18054 180
64 18095 18105 18109 18123 18132 18136 18139 18148 18192 18196 18244 18269 18279
18366 18368 18370 18382 18384 18393 18417 18444 18499 18511 18667 18669 18673 1
8673 18694 18708 18709 18712 18722 18725 18775 18788 18843 18844 18845 18846 188
60 18865 18885 18903 18909 18920 18946 18952 18963 18976 18980 19001 19012 19045
19060 19103 19194 19195 19198 19209 19233 19266 19267 19358 19359 19363 19365 1
9388 19398 19401 19422 19432 19436 19459 19467 19468 19470 19480 19496 19521 195
23 19557 19562 19577 19578 19597 19598 19622 19675 19686 19697 19703 19770 19779
19785 19791 19802 19803 19812 19850 19855 19865 19874 19877 19881 19893 19905 1
9913 19940 19944 19974 19990 19993
real    0m0.028s
user    0m0.017s
sys      0m0.114s
ysf@ysf0411:~/Desktop/PA$
```

多线程

2. 对于读者写者，我按照课本上的进行 PV 操作，得以正常实现，而使用读写锁，更加容易得实现了读者写者问题。

而两者在同步上也有着区别。为了使得结果更加明显，我使用了 printf 和 sleep 来使得读写请求的间隔更大，最后结果反映两者都是读优先。

```
Q2.c
41
42 int main()
43 {
44     //读写操作
45     // while(1){
46     // pthread_t read;
47     // pthread_t write;
48
49     // sem_init(&mutex,0,1);//初始化为1
50     // sem_init(&writeblock,0,1);
51
52     // pthread_create(&read,NULL,reader,NULL);
53     // pthread_create(&write,NULL,writer,NULL);
54
55     // pthread_join(read,NULL);
56     // pthread_join(write,NULL);
57
58     // }
59
60     //TEST1
61     sem_init(&mutex,0,1);//初始化为1
62     sem_init(&writeblock,0,1);
63     pthread_t write;
64     pthread_t read[2];
65     pthread_create(&read[0],NULL,reader,NULL);
66     pthread_create(&write,NULL,writer,NULL);
67     pthread_create(&read[1],NULL,reader,NULL);
68
69     pthread_join(read[0],NULL);
70     pthread_join(write,NULL);
71     pthread_join(read[1],NULL);
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
230
```

```

8 void *reader()
9 {
10     while(1){
11         if( pthread_rwlock_tryrdlock(&lock) != 0) //申请失败
12         {
13             printf("Reader failed to get the lock\n");
14             continue; //继续申请锁
15         }
16     }
17     else
18     {
19         printf("read:%d\n",file);
20         sleep(1);
21         pthread_rwlock_unlock(&lock); //释放锁
22     }
23     printf("read over\n");
24     break;
25 }
26 }
27 }
28 }
29 }
30 void *writer()
31 {
32     while(1){
33         if(pthread_rwlock_trywrlock(&lock) != 0)
34         {
35             // printf("Writer failed to get the lock\n");
36             continue;
37         }
38         else
39         {
40             file++;
41             printf("Write the file!\n");
42             pthread_rwlock_unlock(&lock);
43         }
44     }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

```

ysf@ysf0411: ~/Desktop/PA
Writer failed to get the lock
Writer failed to get the lock
Writer failed to get the lock
Writer failed to get the lock
Writer failed to get the lock
Write the file!
write over
read over
read over
ysf@ysf0411:~/Desktop/PA$ gcc Q22.c -o Q22 -pthread
Q22.c: In function 'reader':
Q22.c:20:4: warning: implicit declaration of function 'sleep' [-Wimplicit-declaration]
    sleep(1);
    ^~~~~
ysf@ysf0411:~/Desktop/PA$ ./Q22
read:0
read:0
read over
read over
Write the file!
write over
ysf@ysf0411:~/Desktop/PA$ ./Q22
read:0
read:0
read over
read over
Write the file!
write over
ysf@ysf0411:~/Desktop/PA$

```

(读写锁)

3.

我定义了一个类，然后使用构造函数对缓冲区的大小初始化，然后初始化信号量，put, get 函数我对照课本上的读者写者问题实现，使用了 PV 操作来实现。而后面发现在多线程中无法调用类中的函数，我对类进行了修改。

```

class CircleBuffer
{
public:
    void put(int value);
    int get(void);
    CircleBuffer(int k); //构造函数
    int getsize();
private:
    int size;
    int in;
    int out;
    int *arr;
    sem_t empty, full, mutex;
};

CircleBuffer::CircleBuffer(int k)
{
    size=k;
    in=0;
    out=0;
    arr=new int[size];
    sem_init(&mutex,0,1);
    sem_init(&empty,0,k);
    sem_init(&full,0,0);
}

int CircleBuffer::getsize(){
    return size;
}

```

```

class CircleBuffer
{
public:
    void put(int value);
    void get();
    CircleBuffer(int k); //构造函数
    int getsize();
    static void * thread_run1(void* tmp)
    {
        CircleBuffer *p = (CircleBuffer *)tmp;
        p->put(5);
    }
    static void * thread_run2(void* tmp)
    {
        CircleBuffer *p = (CircleBuffer *)tmp;
        p->get();
    }
private:
    int size;
    int in;
    int out;
    int *arr;
    sem_t empty, full, mutex;
};

```

```

void CircleBuffer::put(int value){
    sem_wait(&empty);
    sem_wait(&mutex);
    arr[in]=value;
    printf("index= %d put %d\n",in,value);
    in=(in+1)%size;

    sem_post(&mutex);
    sem_post(&full);
}

void CircleBuffer::get(){
    int ret;
    sem_wait(&full);
    sem_wait(&mutex);
    ret=arr[out];
    printf("index= %d get %d\n",out,ret);
    out=(out+1)%size;

    sem_post(&mutex);
    sem_post(&empty);
}

```

(修改后的函数)

```

int main(){
    cout<<circlebuffer->getsize()<<"\n";

    for(int i=0;i<10;i++){
        circlebuffer->put(3-i);
        if(i==5) circlebuffer->get();
    }
    pthread_t producer[2];
    pthread_t customer[2];

    pthread_create(&customer[0],NULL,CircleBuffer::thread_r
ysf@ysf0411:~/Desktop$ g++ CircleBuffer.c
ysf@ysf0411:~/Desktop$ ./CircleBuffer
10
index= 0 put 3
index= 1 put 2
index= 2 put 1
index= 3 put 0
index= 4 put -1
index= 5 put -2
index= 0 get 3
index= 6 put -3
index= 7 put -4
index= 8 put -5
index= 9 put -6
index= 0 put 5
index= 1 get 2
index= 1 put 5
index= 2 get 1
ysf@ysf0411:~/Desktop$

```

(实验结果：可以正确同步)

4.死锁，我仿照了书上的哲学家问题，只不过我把人数降到了 2，并且每次对一个信号量进行 P 操作后，都 sleep 挂起当前线程，执行另一个线程，从而提高发生死锁概率，最后成功实现了死锁。

```
void *writer_1()
{
    while(1){
        sem_wait(&writeblock_1);
        sleep(2);
        sem_wait(&writeblock_2);
        printf("write file\n");
        file++;
        sem_post(&writeblock_1);
        sem_post(&writeblock_2);
    }
}

void *writer_2()
{
    while(1){
        sem_wait(&writeblock_2);
        sleep(2);
        sem_wait(&writeblock_1);
        printf("write file\n");
        file++;
        sem_post(&writeblock_2);
        sem_post(&writeblock_1);
    }
}
```