

1.新增list.h文件

这个文件定义了 `list_head_t` 结构体，表示链表的节点，使用 `prev` 和 `next` 指针连接形成双向链表。特别是它使用了循环链表（当链表为空时，头节点的 `next` 和 `prev` 指向自己）。这种结构对于高效的内存分配、回收和合并操作非常重要，尤其在伙伴系统中，内存块需要频繁地进行合并与拆分。

在伙伴系统中的作用

在伙伴系统的内存管理中，通常需要处理内存块的分配和回收，以及块的合并和拆分。链表的结构为这些操作提供了便捷的支持。以下是几个具体的应用场景：

内存块的管理：可以使用链表来管理不同大小的空闲内存块。每个空闲块的大小可以作为链表节点的一部分，而这些块按照大小或者其他标准组织成链表。每当有一个内存请求时，你可以快速地从链表中查找合适大小的空闲块。

内存块的合并与拆分：当释放一个内存块时，如果相邻的块也为空闲块，可以将它们合并为一个更大的块。通过链表操作，可以非常方便地进行这种合并。例如，通过 `list_del` 可以删除一个节点，`list_add` 可以将合并后的块重新加入链表。

内存分配与回收的高效性：使用循环链表来组织内存块，避免了对链表头尾的频繁操作，确保了内存分配和回收的效率。在伙伴系统中，链表为内存块的管理提供了灵活性和扩展性。

```
#ifndef QQ3_LIST_H
#define QQ3_LIST_H
/*author qq3小组
*/

/*the host structures will be organised into a circle link-list */

typedef struct list_head{
    struct list_head *prev;
    struct list_head *next;
}list_head_t;

#define INIT_LIST_HEAD(l)\
do{\
    (l)->prev = (l)->next = l;\
}while(0)

static inline void __list_add(list_head_t *new, list_head_t *prev,
                             list_head_t *next){
    new->next = next;
    next->prev = new;
    new->prev = prev;
    prev->next = new;
}
```

• 代码主要内容简介

1. 数据结构

list_head_t

- **描述**: 双向循环链表的节点结构。每个节点有两个指针, `prev` 和 `next`, 分别指向前一个和后一个节点, 链表是循环的, 即头节点的 `next` 指向头节点本身, 尾节点的 `prev` 指向尾节点本身。
- **作用**: 用于管理内存块或其他结构的链表。

2. 宏定义

INIT_LIST_HEAD()

- **描述**: 初始化链表头节点。
- **功能**: 将链表头的 `prev` 和 `next` 指向自己, 表示链表为空。

3. 链表操作函数

__list_add(list_head_t *new, list_head_t *prev, list_head_t *next)

- **描述**: 在链表中插入一个新的节点。
- **功能**: 将 `new` 节点插入到 `prev` 和 `next` 之间。

list_add(list_head_t *new, list_head_t *head)

- **描述**: 将一个新的节点插入到链表的头部。
- **功能**: 调用 `__list_add`, 将节点插入到链表头部, `head` 成为新节点的前驱。

list_add_tail(list_head_t *new, list_head_t *head)

- **描述**: 将一个新的节点插入到链表的尾部。
- **功能**: 调用 `__list_add`, 将节点插入到链表尾部, `head` 的 `prev` 成为新节点的后继。

__list_del(list_head_t *prev, list_head_t *next)

- **描述**: 从链表中删除一个节点。
- **功能**: 通过调整 `prev` 和 `next` 指针, 删除指定的节点。

list_del(list_head_t *entry)

- **描述**: 删除链表中的指定节点。
- **功能**: 调用 `__list_del`, 通过修改前后节点的指针将 `entry` 节点移出链表。

list_del_init(list_head_t *entry)

- **描述**: 删除链表中的指定节点并重新初始化它。
- **功能**: 删除节点后, 重新初始化该节点的 `prev` 和 `next` 指针, 使其变为一个空链表节点。

list_empty(list_head_t *entry)

- **描述**: 检查链表是否为空。
- **功能**: 判断链表头的 `next` 是否指向头节点自己, 若是, 则链表为空。

list_meet_tail(list_head_t *first, list_head_t *entry)

- **描述**: 检查指定节点是否是链表尾节点。
- **功能**: 判断节点的 `next` 是否指向链表头, 若是, 说明该节点是链表的尾节点。

4. 宏定义与辅助函数

LIST_FIND2(stru_t, mb_t, root, key, value, result)

- **描述**: 在链表中查找具有特定键值的节点。
- **功能**: 遍历链表, 查找与指定键值 `value` 匹配的节点, 将匹配节点指针存储在 `result` 中。

hashtable_add(list_head_t *hashtable, int hash, list_head_t *new)

- **描述**: 将一个新节点插入到哈希表中。
- **功能**: 根据哈希值将节点插入到对应的哈希表位置。

MB2STRU(stru_type, mb_addr, mb_name)

- **描述**: 通过链表节点指针反向查找并获取包含该节点的结构体指针。
- **功能**: 通过给定的成员名 `mb_name` 计算出结构体的基地址, 进而返回包含该节点的结构体指针。

container_of(head, stru, member)

- **描述**: 获取链表节点所对应的结构体指针。
- **功能**: 通过链表头节点 `head` 和成员名 `member` 计算出该节点对应的结构体实例。

list_for_each_safe(root, container, mbname)

- **描述**: 安全地遍历链表。
- **功能**: 遍历链表并对每个节点执行操作, 同时在遍历过程中安全地删除节点。

2.修改main.c

新增了对 `qq3_mm_init` 函数的调用, 它负责初始化伙伴系统内存管理, 替代了原本 `kinit2` 的单一物理内存分配功能。伙伴系统管理内存的分配和回收, 提高了内存分配的效率和灵活性。
具体的`qq3_mm_init`函数实现后续会介绍。

```

#include "types.h"
#include "defs.h"
#include "param.h"
#include "memlayout.h"
#include "mmu.h"
#include "proc.h"
#include "x86.h"

static void startothers(void);
static void mpmain(void) __attribute__((noreturn));
extern pde_t *kpgdir;
extern char end[]; // first address after kernel loaded from ELF file

// Bootstrap processor starts running C code here.
// Allocate a real stack and switch to it, first
// doing some setup required for memory allocator to work.
int
main(void)
{
    kinit1(end, P2V(4*1024*1024)); // phys page allocator

    kvmalloc(); // kernel page table
    mpinit(); // detect other processors
    lapicinit(); // interrupt controller
}

#include "types.h"
#include "defs.h"
#include "param.h"
#include "memlayout.h"
#include "mmu.h"
#include "proc.h"
#include "x86.h"
#include "qq3_mm.h"

static void startothers(void);
static void mpmain(void) __attribute__((noreturn));
extern pde_t *kpgdir;
extern char end[]; // first address after kernel loaded from ELF file

// Bootstrap processor starts running C code here.
// Allocate a real stack and switch to it, first
// doing some setup required for memory allocator to work.
int
main(void)
{
    kinit1(end, P2V(4*1024*1024)); // phys page allocator
    qq3_mm_init(V2P(end), 4*1024*1024, 4*1024*1024, PHYSTOP);
    kvmalloc(); // kernel page table
    mpinit(); // detect other processors
    lapicinit(); // interrupt controller
}

```

3.修改kalloc.c

修改后的 `kalloc.c` 文件内容通过集成伙伴系统的功能，改动了内存分配和回收机制，使其能够更高效地分配和管理物理内存块。

```

C kalloc.c
74- kmem.freelist = r;
75- if(kmem.use_lock)
76-     release(&kmem.lock);
77- }
78-
79- // Allocate one 4096-byte page of physical memory.
80- // Returns a pointer that the kernel can use.
81- // Returns 0 if the memory cannot be allocated.
82- char*
83- kalloc(void)
84- {
85-     struct run *r;
86-
87-     if(kmem.use_lock)
88-         acquire(&kmem.lock);
89-     r = kmem.freelist;
90-
91-     if(r)
92-         kmem.freelist = r->next;
93-     if(kmem.use_lock)
94-         release(&kmem.lock);
95-
96-     return (char*)r;
97- }
98-
99- // next - kpglist.head;
100- //
101- // free_page(v);
102- // if(kpglist.use_lock)
103- //     release(&kpglist.lock);
104- // }
105- // Allocate one 4096-byte page of physical memory.
106- // Returns a pointer that the kernel can use.
107- // Returns 0 if the memory cannot be allocated.
108- char*
109- kalloc(void)
110- {
111-     struct run *r;
112-
113-     if(kpglist.use_lock)
114-         acquire(&kpglist.lock);
115-     /*
116-      * r = kpglist.head;
117-      */
118-     if(r)
119-         kpglist.head = r->next;
120-     /*
121-      * r = __alloc_page();
122-      */
123-     //cprintf("hello:%x }\n", r);
124-     //panic("panic");
125-     if(kpglist.use_lock)
126-         release(&kpglist.lock);
127-     return (char*)r;
128- }

```

主要改动点与作用

1. 使用伙伴系统替代原有的链表管理机制

原代码： 使用一个简单的链表 `freelist` 来管理空闲页块。空闲页块以链表节点的形式连接，分配时从头部取一个节点，回收时将页块重新插入到链表头部。

```

struct {
    struct spinlock lock;
    int use_lock;
    struct run *freelist;
} kmem;

```

改动后： 重命名为 `kpglist`，但更重要的是，用伙伴系统的分配和回收方法取代了原本的链表操作。伙伴系统的分配和回收通过新增的 `__alloc_page` 和 `__free_page` 函数实现，而非直接操作链表。

```
struct {
    struct spinlock lock;
    int use_lock;
    struct run *head;
} kpglist;
```

作用：伙伴系统的内存管理具有更高效的内存块合并和拆分能力，可以减少内存碎片化问题，相比链表管理具有更高的性能和灵活性。

2. 修改 kinit1和kinit2 的初始化逻辑

原代码：在 `kinit1` 和 `kinit2` 中，通过 `freerange` 初始化物理内存块，并将其添加到空闲链表中。

```
void kinit1(void *vstart, void *vend) {
    initlock(&kmem.lock, "kmem");
    kmem.use_lock = 0;
    freerange(vstart, vend);
}

void kinit2(void *vstart, void *vend) {
    freerange(vstart, vend);
    kmem.use_lock = 1;
}
```

改动后：注释掉了 `freerange` 调用，改为通过伙伴系统管理内存块，同时增加了一个全局变量 `global_use_clock` 来控制伙伴系统的初始化阶段。

```
void kinit1(void *vstart, void *vend) {
    initlock(&kpglist.lock, "kpglist");
    kpglist.use_lock = 0;
    global_use_clock = 0;
}

void kinit2(void *vstart, void *vend) {
    kpglist.use_lock = 1;
    global_use_clock = 1;
}
```

作用：通过伙伴系统的初始化逻辑，替代了原有的 `freerange` 方法，优化了内存初始化过程。新增的全局变量 `global_use_clock` 可能用于全局同步伙伴系统的状态。

3. 内存回收 (**kfree**) 的改动

原代码：使用链表管理空闲内存页，通过将释放的页块插入到链表头部来实现回收。

```
r = (struct run*)v;
r->next = kmem.freelist;
kmem.freelist = r;
```

改动后：使用 `__free_page(v)` 函数处理内存回收，而不再直接操作链表。

```
__free_page(v);
```

作用： 将内存回收的逻辑交给伙伴系统管理，这样可以自动处理空闲块的合并操作，提高内存利用率，并减少碎片化。

4. 内存分配 (**kalloc**) 的改动

原代码： 从链表头部获取一个空闲页块。

```
r = kmem.freelist;
if(r)
    kmem.freelist = r->next;
```

改动后： 改为使用 `__alloc_page` 函数分配页块，并注释掉了原链表操作。

```
r = __alloc_page();
```

作用： 伙伴系统通过 `__alloc_page` 提供了更智能的分配机制，允许分配不同大小的内存块，同时保留空闲块的合并和拆分能力。

5. freerange 逻辑的保留

虽然 `freerange` 函数在 `kinit1` 和 `kinit2` 中被注释掉了，但其定义仍然存在，是为了兼容或者作为备用逻辑。

新增函数的作用

`__alloc_page`

- **描述：** 伙伴系统分配页块的核心函数。
- **作用：** 根据伙伴系统的规则，从内存池中分配一个页块。

`__free_page`

- **描述：** 伙伴系统释放页块的核心函数。
- **作用：** 将回收的页块重新加入伙伴系统的内存池中，并根据需要合并相邻空闲块。

```
C kalloc.c
74- kmem.freelist = r;
75- if(kmem.use_lock)
76-     release(&kmem.lock);
77 }
78
79 // Allocate one 4096-byte page of physical memory.
80 // Returns a pointer that the kernel can use.
81 // Returns 0 if the memory cannot be allocated.
82 char*
83 kalloc(void)
84 {
85     struct run *r;
86
87     if(kmem.use_lock)
88         acquire(&kmem.lock);
89     r = kmem.freelist;
90
91     if(r)
92         kmem.freelist = r->next;
93     if(kmem.use_lock)
94         release(&kmem.lock);
95
96     return (char*)r;
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
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530

```

4.新建.Bochsrc和bochsout并修改dot-bochsrc文件

1. .bochsrc 配置文件

`.bochsrc` 是 Bochs 模拟器的核心配置文件，它控制了整个虚拟机的行为和硬件模拟参数。这对我们演示伙伴系统的机制是非常必要的。

作用：

- 虚拟环境的基础配置：
 - 设置 CPU 参数（核心数量、IPS）。
 - 设置内存大小（如 `megs: 32`）。
 - 设置硬盘和启动顺序（如 `boot: disk`）。
- 伙伴算法运行的环境保障：
 - 确保模拟器能正确加载和运行操作系统（如加载 `xv6.img` 和 `fs.img`）。
- 定义外围设备（如 `floppya` 和 `ata0-master`）以确保文件系统和内核能够运行伙伴算法。

为什么需要它？

- `.bochsrc` 是运行伙伴算法所需虚拟环境的配置入口。如果没有正确配置，模拟器将无法正常运行伙伴算法所依赖的操作系统和相关程序。

```
# nas a selection of about 10 different display library implementations to
# different platforms.  If you run configure with multiple --with-* option
# the display_library command lets you choose which one you want to run with
# If you do not write a display_library line, Bochs will choose a default for
# you.
#
# The choices are:
#   x                use X windows interface, cross platform
#   win32            use native win32 libraries
#   carbon            use Carbon library (for MacOS X)
#   beos             use native BeOS libraries
#   macintosh        use MacOS pre-10
#   amigaos          use native AmigaOS libraries
#   sdl              use SDL library, cross platform
#   svga             use SVGALIB library for Linux, allows graphics without
#   term             text only, uses curses/ncurses library, cross platform
#   rfb              provides an interface to AT&T's VNC viewer, cross platform
#   wx               use wxWidgets library, cross platform
#   nogui            no display at all
#
# NOTE: if you use the "wx" configuration interface, you must also use
# the "wx" display library.
#
# Specific options:
# Some display libraries now support specific option to control their
# behaviour.  See the examples below for currently supported options
```


2. bochsout 日志文件

`bochsout` 是 Bochs 模拟器的输出日志文件，它记录了模拟器运行期间的所有信息，包括错误、警告和调试信息。

作用：

- 运行过程的追踪和诊断：
 - 如果伙伴算法运行失败，`bochsout` 中的日志可以帮助你定位错误是发生在操作系统加载阶段、文件系统访问阶段，还是伙伴算法本身。
- 性能监控和优化：
 - 日志中可能包含 CPU 指令执行情况和内存使用情况，这对优化伙伴算法的运行效率很有帮助。

为什么需要它？

- `bochsout` 是调试伙伴算法的重要工具。如果没有日志，你将无法了解虚拟机内部的运行情况，尤其是当算法因虚拟硬件配置问题或软件错误崩溃时，日志是唯一的线索。

```
00000000000i[ ] Bochs x86 Emulator 2.6.8
00000000000i[ ] Built from SVN snapshot on May 3, 2015
00000000000i[ ] Compiled on Dec 20 2024 at 06:03:53
00000000000i[ ] System configuration
00000000000i[ ] processors: 2 (cores=1, HT threads=1)
00000000000i[ ] A20 line support: yes
00000000000i[ ] IPS is set to 10000000
00000000000i[ ] CPU configuration
00000000000i[ ] SMP support: yes, quantum=16
00000000000i[ ] level: 6
00000000000i[ ] APIC support: xapic
00000000000i[ ] FPU support: yes
00000000000i[ ] MMX support: yes
00000000000i[ ] 3dnow! support: no
00000000000i[ ] SEP support: yes
00000000000i[ ] SIMD support: sse2
00000000000i[ ] XSAVE support: no
00000000000i[ ] AES support: no
00000000000i[ ] SHA support: no
00000000000i[ ] MOVBE support: no
00000000000i[ ] ADX support: no
00000000000i[ ] x86-64 support: yes
00000000000i[ ] 1G paging support: no
00000000000i[ ] MWAIT support: yes
00000000000i[ ] VMX support: 1
00000000000i[ ] Optimization configuration
00000000000i[ ] RepeatSpeedups support: no
00000000000i[ ] Fast function calls: no
00000000000i[ ] Handlers Chaining speedups: no
00000000000i[ ] Devices configuration
00000000000i[ ] NE2000 support: no
00000000000i[ ] PCI support: yes, enabled=yes
00000000000i[ ] SB16 support: no
00000000000i[ ] USB support: no
00000000000i[ ] VGA extension support: vbe
00000000000i[MEM0 ] allocated memory at 0x7fea607f5010. after alignment, vector=0x7fea607f
00000000000i[MEM0 ] 32.00MB
00000000000i[MEM0 ] mem block size = 0x00100000, blocks=32
00000000000i[MEM0 ] rom at 0xffe0000/131072 ('/usr/local/share/bochs/BIOS-bochs-latest')
00000000000i[PLUGIN] init dev of 'pci' plugin device by virtual method
00000000000i[DEV ] i440FX PMC present at device 0, function 0
00000000000i[PLUGIN] init dev of 'pci2isa' plugin device by virtual method
00000000000i[DEV ] PIIX3 PCI-to-ISA bridge present at device 1, function 0
```


3. dot-bochsrc 配置文件

`dot-bochsrc` 是另一个模拟器配置文件，它的作用类似于 `.bochsrc`，但针对的是一些补充性或特定场景的配置。

作用：

- 自定义补充配置：
 - `dot-bochsrc` 中可能存在一些未在 `.bochsrc` 中指定的额外选项（如显示模式、鼠标、键盘类型等），这些选项可以为伙伴系统提供更加灵活的支持。
- 快速切换配置：
 - 如果需要对伙伴系统进行不同硬件环境的测试，`dot-bochsrc` 提供了一种方便的方式来覆盖或补充默认配置。

为什么需要它？

- 在复杂的模拟环境中，可能需要为不同的测试场景提供不同的配置文件（如针对伙伴算法的显示优化、输入设备配置等），`dot-bochsrc` 就充当了一个额外的调试和测试工具。

```
# You may now use double quotes around pathnames, in case
# your pathname includes spaces.

=====
# CONFIG INTERFACE
#
# The configuration interface is a series of menus or dialog boxes that
# allows you to change all the settings that control Bochs's behavior.
# There are two choices of configuration interface: a text mode version
# called "textconfig" and a graphical version called "wx". The text
# mode version uses stdin/stdout and is always compiled in. The graphical
# version is only available when you use "--with-wx" on the configure
# command. If you do not write a config interface line, Bochs will
# choose a default for you.
#
# NOTE: if you use the "wx" configuration interface, you must also use
# the "wx" display library.
=====
#config interface: textconfig
#config interface: wx

=====
# DISPLAY LIBRARY
#
# The display library is the code that displays the Bochs VGA screen. Bochs
# has a selection of about 10 different display library implementations for
# different platforms. If you run configure with multiple --with-* options,
# the display library command lets you choose which one you want to run with.
# If you do not write a display library line, Bochs will choose a default for
# you.
#
# The choices are:
# x          use X windows interface, cross platform
# win32      use native win32 libraries
# carbon     use Carbon library (for MacOS X)
# beos       use native BeOS libraries
# macintosh  use MacOS pre-10
# amigaos    use native AmigaOS libraries
# sdl        use SDL library, cross platform
# svga       use SVGALIB library for Linux, allows graphics without X11
# term       text only, uses curses/ncurses library, cross platform
# rfb        provides an interface to AT&T's VNC viewer, cross platform
```

5.qq3_page.h- 页面描述和虚拟/物理地址映射

```
#ifndef QQ3_PAGE_H
#define QQ3_PAGE_H
/*author qq3小组
*/

#include "memlayout.h"

#define PAGE_SHIFT 12
#define PAGE_SIZE 0x1000
#define PAGE_MASK (~0xfff)
#define pa_idx(paddr) (((unsigned)paddr)>>PAGE_SHIFT)
#define pa_pg pa_idx

/*TODO cancell the following two macros, and 'vpg', 'ppg' is a good transfe
#define PG_H10(pg_id) (pg_id>>10)
#define PG_L10(pg_id) (pg_id&(0x400-1))

/* page table/directory entry ==> linear address of target page */

// >一个普通的整形变成了union，其实是反而隐藏了类型信息。但不妨一试。
// 毕竟pte是特殊的，常常知道pte是个unsigned。
#pragma pack(push)
#pragma pack(1)
union pte{
    int value;
    struct {
        unsigned present: 1;
        unsigned writable: 1;
        unsigned user: 1;
        unsigned PWT: 1;
        unsigned PCD: 1;
        unsigned accessed: 1;
        unsigned dirty: 1;
        unsigned : 2;
        unsigned avl: 3;
    };
};
```

主要结构和功能：

- **pte (Page Table Entry) :**
 - 这是一个联合体，包含了页面表项的标志位和物理页面的地址。
 - 页面表项用于描述页表中的每个页面，包含如是否存在、是否可写、是否是用户空间可访问、页面是否已经被访问等标志。具体来说，结构体中的每一位表示一个标志位。
 - **physical** 字段表示该页面所在的物理地址，配合其他标志一起使用。
- **linear_addr:**
 - 这是一个联合体，用于描述虚拟地址。通过将虚拟地址分成 **offset**、**tbl_idx**（表项索引）和 **dir_idx**（目录项索引）三个部分，帮助实现虚拟地址到物理地址的转换。

- `cr3` :
 - `cr3` 是 x86 架构下用于存储当前页目录基址的寄存器。它包含了一个物理地址，用于指向页目录。在这里，它用于表示页表的物理地址。
- `__pa` 和 `__va` 宏：
 - `__pa(vaddr)`：将虚拟地址 `vaddr` 转换为物理地址。
 - `__va(paddr)`：将物理地址 `paddr` 转换为虚拟地址。
 - 这些宏是操作系统中的关键部分，它们为伙伴算法的内存管理提供了物理地址和虚拟地址之间的转换接口。
- `pte2page()` 和 `__va2pte()` 函数：
 - `pte2page()` 用来将一个页表项转换为指向物理页面的指针。
 - `__va2pte()` 函数根据虚拟地址和页目录查找对应的页表项，用于访问内存。

这些定义和宏确保了操作系统能够通过页表操作虚拟内存和物理内存之间的转换，从而支持页面的分配与回收，这对于伙伴算法的实现至关重要。

6.qq3_mm.h - 内存管理的核心结构和函数声明

`qq3_mm.h` 文件定义了与内存分配和回收相关的核心数据结构，并声明了与伙伴算法实现相关的内存管理函数。

```

qq3_mm.h
1  #ifndef QQ3_MM_H
2  #define QQ3_MM_H
3  /* author qq3小组
4  **/
5
6  #include "types.h"
7  #include "defs.h"
8  #include "qq3_list.h"
9  #include "qq3_page.h"
0  #include "spinlock.h"
1
2
3  /*physical page descriptor
4  * 页描述暂不设锁，由上层路径加锁
5  */
6  typedef struct page{
7      struct list_head lru;    //用于buddy系统的链表头
8      int _count;
9      //代表该page作为buddy头的order,最大10, 约定32为特殊值表示已被分配
0      //约定负数表示作为buddy followee
1      char private;
2      char zone_id;
3  } page_t;
4
5
6  #define page_idx(page_t) ((unsigned)((page_t) - mem_map))
7  // #define pte_pfn(pte) ((pte)>>PAGE_SHIFT)
8  // #define pfn_page(pfn) (mem_map + (pfn))
9  // #define pte_page(pte) ( pfn_page( pte_pfn(pte) ) )
0  // #define page_va(page) __va( (page - mem_map) << PAGE_SHIFT)
1  // #define virt_to_page(vaddr) pfn_page( __pa(vaddr) >> PAGE_S
2
3  #define MAX_ORDER 10

```

主要结构和功能:

- `struct page`:
 - 该结构体代表每个物理页面。在伙伴算法中，每个 `struct page` 都对应系统中的一个物理页面。
 - 它可能包含一些元数据，如与其他页面的链接、空闲标志、页的大小等。在伙伴算法中，每个页面需要被标记和管理，以便进行有效的分配和释放。
- `struct free_area`:
 - `free_area` 用于管理不同大小的空闲块。在伙伴算法中，内存会被划分成若干大小的块，每个大小对应一个链表。`free_area` 包含一个链表，链表中的每个元素都是一个空闲内存块。每个内存块的大小是 2 的幂。
 - 在分配内存时，系统会在 `free_area` 中查找足够大的空闲块。如果没有找到合适的块，系统会递归地将更大的块拆分成小块。
- `alloc_pages()` 和 `free_pages()`:

- `alloc_pages()` 用于从 `free_area` 中分配一个或多个页面。当系统收到一个内存分配请求时，它会检查 `free_area` 中是否有足够大的空闲块。如果没有，系统会继续拆分更大的块，直到满足请求的大小。
- `free_pages()` 用于释放页面并将其归还给 `free_area`。在释放页面时，系统会检查相邻的块是否可以合并。如果可以，系统会将相邻的块合并成更大的块，这样有助于减少内存碎片。
- **zone 和 pageblock:**
 - `zone` 是系统中一个内存区域的定义，通常由多个 `free_area` 构成，代表某个物理区域的内存。
 - `pageblock` 是一组连续页面的集合，通常用于批量分配和回收。`zone` 内的页面被划分为若干 `pageblock`，并按大小维护空闲列表。

7. qq3_mm.c - 伙伴算法的实现

`qq3_mm.c` 文件实现了伙伴算法的核心逻辑，包括内存的分配、释放、拆分和合并等操作。

```

C qq3_mm.c
1  /* author qq3小组
2  */
3
4  #include "qq3_mm.h"
5  #include "qq3_page.h"
6  #include "qq3_global.h"
7
8
9  #define MEMBER_OFFSET(stru_type, member_name) \
10     ( (unsigned)&((stru_type *)0)->member_name) )
11
12 //演示free时的合并行为 屏幕打印 -表示发生一个合并 !表示不能继续合并, 入链
13 int debug_demo_free = 1;
14 //演示alloc时的拆分行行为 屏幕打印 %表示发生一个拆分 !表示order匹配终止拆
15 int debug_demo_alloc = 1;
16
17 int page_is_buddy(struct page *page, int order);
18 void init_free_area(int zone_id, int start_idx);
19 void __free_pages_bulk(struct page *page, zone_t *zone, int or
20 void cleave(free_area_t *free_area, int order);
21
22
23 struct spinlock page_big_lock;
24 #define PGNUM_MAX 1024*128
25 struct page mem_map[PGNUM_MAX]; //TODO 大数组, 过大会越过xv6的4M初
26 zone_t memzones[2];
27 void info_zone(zone_t *zone){
28     cprintf("\nbuddy total pages: %x \nbuddy quenes (4k-4M):",
29     free_area_t *area = zone->free_area;
30     for(int i = 0; i <= MAX_ORDER; i++){
31         cprintf("%x ", area[i].nr_free);
32     }
33     cprintf("\n");
34 }
35
36 void qq3_mm_init(unsigned va, unsigned end_va, unsigned va2, u
37     memset(mem_map, 0, PGNUM_MAX*sizeof(page_t));
38     initlock(&page_big_lock, "page_big_lock");

```

主要功能:

- `alloc_pages()`:
 - 该函数实现了内存的分配。它会遍历 `free_area` 中的空闲链表, 查找符合要求的空闲块。如果找到较大的块, 系统会继续将其拆分成更小的块, 直到找到满足请求的块。
 - 如果无法找到合适的块, 系统会继续向上查找更大的空闲块, 直到找到足够大的块或无法找到。
- `free_pages()`:
 - 该函数实现了内存的释放。在释放页面时, 系统会尝试将相邻的空闲块合并成一个更大的块, 以减少内存碎片。
 - 如果释放的块大小是 2 的幂次方, 它会检查相邻的块是否可以合并。如果可以, 系统会将其合并为更大的块, 并将合并后的块归还给 `free_area`。

- `cleave()` 和 `__free_pages_bulk()` :
 - `cleave()` 函数用于拆分较大的块，将其分为两个较小的块。它是实现伙伴算法的关键步骤之一。
 - `__free_pages_bulk()` 函数在释放多个页面时，会在释放时处理块的合并和拆分操作。
- 伙伴合并：
 - 在伙伴算法中，当释放的页面无法与邻近的页面合并时，它们会被加入到空闲列表中，并等待分配。通过合并相邻的块，可以有效减少内存碎片并提高内存利用率。

8.qq3_global.h - 全局变量和宏定义

`qq3_global.h` 主要定义了一些全局变量、常量和宏，用于内存管理和伙伴算法的配置。

```
C qq3_global.h
1  #ifndef QQ3_GLOBAL_H
2  #define QQ3_GLOBAL_H
3  //author qq3小组
4
5  #include "types.h"
6  #include "defs.h"
7
8  int global_use_clock;
9  void assert_func(char*exp,char*file,char*base_file,int i)
10
11  #define cassert(exp)
12      do{
13          if(!(exp)) assert_func(#exp,__FILE__,__BASE_
14      } while(0)
15
16  #endif
17
```

主要功能：

- `V2P` 和 `P2V` 宏：
 - 这两个宏在系统中起到了非常重要的作用，它们分别用于虚拟地址与物理地址之间的转换。伙伴算法依赖于物理页面的分配和释放，而这些宏提供了从虚拟地址到物理地址的转换接口。
- `ZONE_SIZE` 和 `MAX_ORDER` 宏：
 - `ZONE_SIZE` 定义了每个内存区域的大小，而 `MAX_ORDER` 则定义了支持的最大内存块大小（即 `order`），这些参数控制了伙伴算法中内存块的划分和管理范围。
- `init_zone()` 和 `init_free_area()` :
 - 这两个函数用于初始化内存区域和空闲块链表，为伙伴算法的运行做准备。

9.qq3_global.c - 全局内存管理初始化和信息输出

qq3_global.c 文件负责内存管理的初始化，确保在系统启动时正确配置内存区域、空闲链表等。

```
C qq3_global.c
1  #include"qq3_global.h"
2
3  void assert_func(char*exp,char*file,char*base_fi
4      |    cprintf("assert failure>>>exp:%s,file:%s,bas
5      |    panic("\nASSERT SPIN !!!");
6  }
7
```

主要功能：

- `init_zone()` :
 - 初始化内存区域，为后续的内存管理工作做准备。通过这个函数，系统会为每个内存区域设置初始状态，并为每个区域的 `free_area` 链表分配内存。
- `info_zone()` :
 - 打印系统当前内存区域的信息，便于调试和验证内存管理是否正常工作。通过输出内存区域的状态，可以帮助开发者检查系统在初始化过程中是否存在问题。

10.qq3_list.h - 链表操作

- `qq3_list.h` 提供了链表操作的工具函数，用于管理空闲页面链表。

```

C qq3_list.h
1  #ifndef QQ3_LIST_H
2  #define QQ3_LIST_H
3
4  /*the host structures will be organised into a circle link-list */
5
6  typedef struct list_head{
7      struct list_head *prev;
8      struct list_head *next;
9  }list_head_t;
10
11 #define INIT_LIST_HEAD(l)\
12     do{\
13         (l)->prev = (l)->next = l;\
14     } while(0)
15
16 static inline void __list_add(list_head_t *new, list_head_t *prev,
17                               list_head_t *next){
18     new->next = next;
19     next->prev = new;
20     new->prev = prev;
21     prev->next = new;
22 }
23
24 /**
25  * 1,assume only one entry in list, (new)'s 'next' pointer will point to
26  * 'head', and (head)'s prev will point to 'new'. So, the list is circle
27  */
28
29 static inline void list_add(list_head_t *new, list_head_t *head){
30     __list_add(new, head, head->next);
31 }
32
33 static inline void list_add_tail(list_head_t *new, list_head_t *head){

```

主要功能：

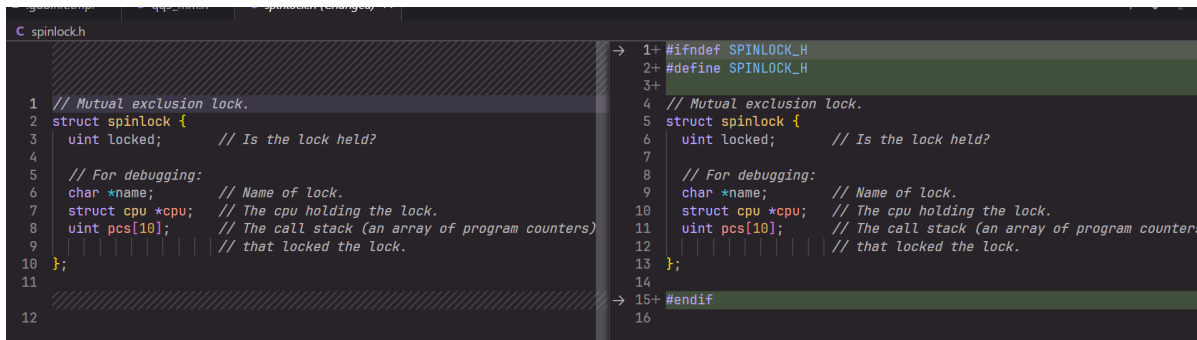
- `list_add()` 和 `list_del()` :
 - 这两个函数用于在链表中添加和删除元素。在伙伴算法中，当页面被分配或释放时，它们会被从 `free_area` 的链表中删除或添加。
- `list_for_each_safe()` :
 - 安全地遍历链表，在遍历过程中可以删除当前节点，而不会破坏链表的结构。对于内存的分配和释放，链表操作非常频繁，因此这些操作的安全性至关重要。

11.spinlock.h - 自旋锁的修改

自旋锁的修改主要为并发环境下的内存管理提供了保证。在多核系统中，多个处理器可能会同时访问和修改 `free_area` 链表，导致数据竞争问题。通过自旋锁保护对空闲链表的访问，可以确保每次只有一个 CPU 可以对内存块进行操作，避免了并发访问引发的错误。

****struct spinlock****：定义了自旋锁的结构，确保操作系统在进行内存分配和回收时能够同步地访问共享资源。

锁的保护：在 `alloc_pages()` 和 `free_pages()` 等内存管理函数中，通过加锁和解锁的方式确保内存块的分配、释放以及合并操作是线程安全的。



12. tags - 方便对项目代码进行管理

包含了代码标记 (tags) 文件的内容, 用于项目中的代码导航和搜索工具 (如ctags) 来帮助开发人员在大型代码库中快速找到相关函数、结构体或变量的定义和引用。

```
> Users > 32951 > Desktop > xv6 > xv6-public-master > tags
1  |!_TAG_FILE_FORMAT  2  /extended format; --format=1 will not append ;" to lin
2  !_TAG_FILE_SORTED  1  /0=unsorted, 1=sorted, 2=foldcase/
3  !_TAG_PROGRAM_AUTHOR  Darren Hiebert  /dhiebert@users.sourceforge.net/
4  !_TAG_PROGRAM_NAME  Exuberant Ctags //
5  !_TAG_PROGRAM_URL  http://ctags.sourceforge.net  /official site/
6  !_TAG_PROGRAM_VERSION  5.9~svn20110310 //
7  ALT kbd.h  11;"  d
8  AS  Makefile  /^AS = $(TOOLPREFIX)gas$/" m
9  ASFLAGS Makefile  /^ASFLAGS = -m32 -gdwarf-2 -Wa,-divide$/" m
10 ASSERT  lapic.c 25;"  d  file:
11 Align  umalloc.c /^typedef long Align;$/" t  file:
12 BACK  sh.c  12;"  d  file:
13 BACKSPACE  console.c 127;"  d  file:
14 BBLOCK  fs.h  48;"  d
15 BCAST  lapic.c 28;"  d  file:
16 BIG  ustests.c 1452;"  d  file:
17 BPB  fs.h  45;"  d
18 BSIZE  fs.h  6;" d
19 BUSY  lapic.c 29;"  d  file:
20 B_DIRTY buf.h  13;"  d
21 B_VALID buf.h  12;"  d
22 C  console.c 189;"  d  file:
23 C  kbd.h  32;"  d
24 CAPSLOCK  kbd.h  13;"  d
25 CC  Makefile  /^CC = $(TOOLPREFIX)gcc$/" m
26 CFLAGS Makefile  /^CFLAGS = -fno-pic -static -fno-builtin -fno-strict-alias
27 CMOS_PORT  lapic.c 123;"  d  file:
28 CMOS_RETURN lapic.c 124;"  d  file:
29 CMOS_STATA  lapic.c 163;"  d  file:
30 CMOS_STATA  lapic.c 164;"  d  file:
```