

# 操作系统实验6

邓人嘉 21301032

## 一、实验步骤

### 1.1 在内核中支持动态内存分配

- Cargo.toml中增加依赖 (os/Cargo.toml)

```
8 [dependencies]
9 riscv = { git = "https://github.com/rcore-os/riscv", features = ["inline-asm"] }
10 lazy_static = { version = "1.4.0", features = ["spin_no_std"] }
11 buddy_system_allocator = "0.6"
12 spin = "0.7.0"
13 bitflags = "1.2.1"
```

- 在main.rs中引入alloc库的依赖 (os/src/main.rs)

```
@3d686a71b33d:/mnt/os/src × + v
1 #![no_std]
2 #![no_main]
3 #![feature(panic_info_message)]
4 #![feature(alloc_error_handler)]
5
6 use core::arch::global_asm;
7
8 extern crate alloc;
```

- 根据alloc留好的结构提供全局动态内存分配器，并处理动态内存分配失败的情况 (os/src/mm/heap\_allocator.rs)

```
[root@94131a7dd3b2 src]# mkdir mm
[root@94131a7dd3b2 src]# cd mm
[root@94131a7dd3b2 mm]# vim heap_allocator.rs
```

```
@94131a7dd3b2:/mnt/os/src/ × + v
1 use buddy_system_allocator::LockedHeap;
2 use crate::config::KERNEL_HEAP_SIZE;
3
4 #[global_allocator]
5 static HEAP_ALLOCATOR: LockedHeap = LockedHeap::empty();
6
7 static mut HEAP_SPACE: [u8; KERNEL_HEAP_SIZE] = [0; KERNEL_HEAP_SIZE];
8
9 pub fn init_heap() {
10     unsafe {
11         HEAP_ALLOCATOR
12             .lock()
13             .init(HEAP_SPACE.as_ptr() as usize, KERNEL_HEAP_SIZE);
14     }
15 }
```

```

12     .lock();
13     .init(HEAP_SPACE.as_ptr() as usize, KERNEL_HEAP_SIZE);
14 }
15 }
16
17 #[allow(unused)]
18 pub fn heap_test() {
19     use alloc::boxed::Box;
20     use alloc::vec::Vec;
21     extern "C" {
22         fn sbss();
23         fn ebss();
24     }
25     let bss_range = sbss as usize..ebss as usize;
26     let a = Box::new(5);
27     assert_eq!(*a, 5);
28     assert!(bss_range.contains(&(a.as_ref() as *const _ as usize)));
29     drop(a);
30     let mut v: Vec<usize> = Vec::new();
31     for i in 0..500 {
32         v.push(i);
33     }
34     for i in 0..500 {
35         assert_eq!(v[i], i);
36     }
37     assert!(bss_range.contains(&(v.as_ptr() as usize)));
38     drop(v);
39     println!("heap_test passed!");
40 }

```

- 在main.rs中增加处理动态内存分配失败的情况 (os/src/main.rs)

```

1 #![no_std]
2 #![no_main]
3 #![feature(panic_info_message)]
4 #![feature(alloc_error_handler)]

```

- 实现测试动态内存分配 (os/src/mm/heap\_allocator.rs)

```

37     assert!(bss_range.contains(&(v.as_ptr() as usize)));
38     drop(v);
39     println!("heap_test passed!");
40 }
41
42 #[allow(unused)]
43 pub fn heap_test() {
44     use alloc::boxed::Box;
45     use alloc::vec::Vec;
46     extern "C" {
47         fn sbss();
48         fn ebss();
49     }
50     let bss_range = sbss as usize..ebss as usize;
51     let a = Box::new(5);
52     assert_eq!(*a, 5);
53     assert!(bss_range.contains(&(a.as_ref() as *const _ as usize)));
54     drop(a);
55     let mut v: Vec<usize> = Vec::new();
56     for i in 0..500 {
57         v.push(i);
58     }
59     for i in 0..500 {
60         assert_eq!(v[i], i);
61     }
62     assert!(bss_range.contains(&(v.as_ptr() as usize)));
63     drop(v);
64     println!("heap_test passed!");
65 }

```

- 封装heap\_allocator (os/src/mm/mod.rs)

```

1 mod heap_allocator;
2
3 pub fn init() {
4     heap_allocator::init_heap();
5     heap_allocator::heap_test();
6 }

```

- 在main.rs中对mm进行初始化

```
@94131a7dd3b2:/mnt/os/src
17 mod config;
18 mod task;
19 mod timer;
20 mod mm;
21
22 global_asm!(include_str!("entry.asm"));
23 global_asm!(include_str!("link_app.S"));
24
25 fn clear_bss() {
26     extern "C" {
27         fn sbss();
28         fn ebss();
29     }
30     (sbss as usize..ebss as usize).for_each(|a| {
31         unsafe { (a as *mut u8).write_volatile(0) }
32     });
33 }
34
35 #[no_mangle]
36 pub fn rust_main() -> ! {
37     clear_bss();
38     mm::init();
39     trap::enable_timer_interrupt();
40     timer::set_next_trigger();
41     println!("[kernel] Hello, world!");
42     trap::init();
43     loader::load_apps();
44     task::run_first_task();
45     panic!("Unreachable in rust_main!");
-- INSERT --
38,16 94%
```

- 修改config.rs, 添加:

```
@94131a7dd3b2:/mnt/os/src
1 pub const KERNEL_HEAP_SIZE: usize = 0x30_0000;
```

## 1.2 实现虚拟地址与物理地址的基本定义

- 定义所需要的基本数据结构, 包括物理地址、虚拟地址、物理页号、虚拟页号 (os/src/mm/address.rs)

```
@94131a7dd3b2:/mnt/os/src/
1 #[repr(C)]
2 #[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
3 pub struct PhysAddr(pub usize);
4
5 #[repr(C)]
6 #[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
7 pub struct VirtAddr(pub usize);
8
9 #[repr(C)]
10 #[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
11 pub struct PhysPageNum(pub usize);
12
13 #[repr(C)]
14 #[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
15 pub struct VirtPageNum(pub usize);
```

- 实现上述这些类型和usize之间的相互转换。

```
@94131a7dd3b2/mnt/os/src/ x + v
16
17 /// T: {PhysAddr, VirtAddr, PhysPageNum, VirtPageNum}
18 /// T -> usize: T.0
19 /// usize -> T: usize.into()
20
21 impl From<usize> for PhysAddr {
22     fn from(v: usize) -> Self { Self(v) }
23 }
24 impl From<usize> for PhysPageNum {
25     fn from(v: usize) -> Self { Self(v) }
26 }
27 impl From<usize> for VirtAddr {
28     fn from(v: usize) -> Self { Self(v) }
29 }
30 impl From<usize> for VirtPageNum {
31     fn from(v: usize) -> Self { Self(v) }
32 }
33 impl From<PhysAddr> for usize {
34     fn from(v: PhysAddr) -> Self { v.0 }
35 }
36 impl From<PhysPageNum> for usize {
37     fn from(v: PhysPageNum) -> Self { v.0 }
38 }
39 impl From<VirtAddr> for usize {
40     fn from(v: VirtAddr) -> Self { v.0 }
41 }
42 impl From<VirtPageNum> for usize {
43     fn from(v: VirtPageNum) -> Self { v.0 }
44 }
-- INSERT --
44,2 93%
```

- 实现地址和页号之间的相互转换。

```
@94131a7dd3b2/mnt/os/src/ x + v
46 impl VirtAddr {
47     pub fn floor(&self) -> VirtPageNum { VirtPageNum(self.0 / PAGE_SIZE) }
48     pub fn ceil(&self) -> VirtPageNum { VirtPageNum((self.0 - 1 + PAGE_SIZE) / PAGE_SIZE) }
49     pub fn page_offset(&self) -> usize { self.0 & (PAGE_SIZE - 1) }
50     pub fn aligned(&self) -> bool { self.page_offset() == 0 }
51 }
52 impl From<VirtAddr> for VirtPageNum {
53     fn from(v: VirtAddr) -> Self {
54         assert_eq!(v.page_offset(), 0);
55         v.floor()
56     }
57 }
58 impl From<VirtPageNum> for VirtAddr {
59     fn from(v: VirtPageNum) -> Self { Self(v.0 << PAGE_SIZE_BITS) }
60 }
61 impl PhysAddr {
62     pub fn floor(&self) -> PhysPageNum { PhysPageNum(self.0 / PAGE_SIZE) }
63     pub fn ceil(&self) -> PhysPageNum { PhysPageNum((self.0 - 1 + PAGE_SIZE) / PAGE_SIZE) }
64     pub fn page_offset(&self) -> usize { self.0 & (PAGE_SIZE - 1) }
65     pub fn aligned(&self) -> bool { self.page_offset() == 0 }
66 }
67 impl From<PhysAddr> for PhysPageNum {
68     fn from(v: PhysAddr) -> Self {
69         assert_eq!(v.page_offset(), 0);
70         v.floor()
71     }
72 }
73 impl From<PhysPageNum> for PhysAddr {
74     fn from(v: PhysPageNum) -> Self { Self(v.0 << PAGE_SIZE_BITS) }
75 }
-- INSERT --
75,2 97%
```

- 实现查询索引等其他内容。

```
@94131a7dd3b2/mnt/os/src/ x + v
166 type IntoIter = SimpleRangeIterator<T>;
167 fn into_iter(self) -> Self::IntoIter {
168     SimpleRangeIterator::new(self.l, self.r)
169 }
170 }
171 pub struct SimpleRangeIterator<T> where
172     T: StepByOne + Copy + PartialEq + PartialOrd + Debug, {
173     current: T,
174     end: T,
175 }
176 impl<T> SimpleRangeIterator<T> where
177     T: StepByOne + Copy + PartialEq + PartialOrd + Debug, {
178     pub fn new(l: T, r: T) -> Self {
179         Self { current: l, end: r, }
180     }
181 }
182 impl<T> Iterator for SimpleRangeIterator<T> where
183     T: StepByOne + Copy + PartialEq + PartialOrd + Debug, {
184     type Item = T;
185     fn next(&mut self) -> Option<Self::Item> {
186         if self.current == self.end {
187             None
188         } else {
189             let t = self.current;
190             self.current.step();
191             Some(t)
192         }
193     }
194 }
195 pub type VPNRange = SimpleRange<VirtPageNum>;
-- INSERT --
195,46 98%
```

### 1.3 定义页表项数据结构

- 实现页表项中的标志位PTElags

- os/src/main.rs

```
10 #[macro_use]
11 extern crate bitflags;
```

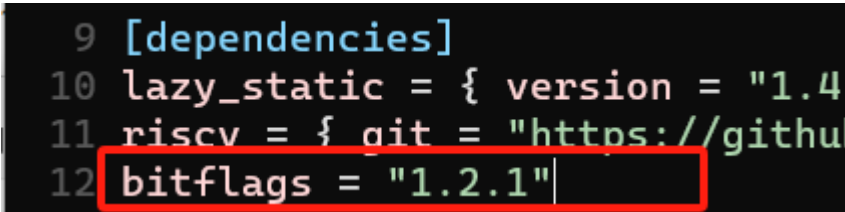
- os/src/mm/page\_table.rs



A screenshot of a code editor window with a dark theme. The title bar shows the file path @94131a7dd3b2:/mnt/os/src/. The code is in Rust and defines a public struct PTEFlags of type u8. It includes several constant bit flags: V (bit 0), R (bit 1), W (bit 2), X (bit 3), U (bit 4), G (bit 5), A (bit 6), and D (bit 7). The code is as follows:

```
1 use bitflags::*;
2
3 bitflags! {
4     pub struct PTEFlags: u8 {
5         const V = 1 << 0;
6         const R = 1 << 1;
7         const W = 1 << 2;
8         const X = 1 << 3;
9         const U = 1 << 4;
10        const G = 1 << 5;
11        const A = 1 << 6;
12        const D = 1 << 7;
13    }
14 }
```

- 在配置文件中增加bitflgs的依赖。(os/Cargo.toml)



A screenshot of a code editor window showing the Cargo.toml file. The file is in TOML format and lists dependencies. The 'bitflags' dependency is highlighted with a red rectangle. The code is as follows:

```
9 [dependencies]
10 lazy_static = { version = "1.4"
11 riscv = { git = "https://github.com/riscv/riscv"
12 bitflags = "1.2.1"
```

- 实现PageTableEntry。 (os/src/mm/page\_table.rs)

```

16 use super::{frame_alloc, PhysPageNum, FrameTracker, VirtPageNum, VirtAddr, StepByOne};
17
18 #[derive(Copy, Clone)]
19 #[repr(C)]
20 pub struct PageTableEntry {
21     pub bits: usize,
22 }
23
24 impl PageTableEntry {
25     pub fn new(ppn: PhysPageNum, flags: PTEFlags) -> Self {
26         PageTableEntry {
27             bits: ppn.0 << 10 | flags.bits as usize,
28         }
29     }
30     pub fn empty() -> Self {
31         PageTableEntry {
32             bits: 0,
33         }
34     }
35     pub fn ppn(&self) -> PhysPageNum {
36         (self.bits >> 10 & ((1usize << 44) - 1)).into()
37     }
38     pub fn flags(&self) -> PTEFlags {
39         PTEFlags::from_bits(self.bits as u8).unwrap()
40     }
41     pub fn is_valid(&self) -> bool {
42         (self.flags() & PTEFlags::V) != PTEFlags::empty()
43     }
44     pub fn readable(&self) -> bool {
45         (self.flags() & PTEFlags::R) != PTEFlags::empty()
46     }
47     pub fn writable(&self) -> bool {
48         (self.flags() & PTEFlags::W) != PTEFlags::empty()
49     }
50     pub fn executable(&self) -> bool {
51         (self.flags() & PTEFlags::X) != PTEFlags::empty()
52     }
53 }

```

- 修改src/mm/mod.rs

```

1 mod heap_allocator;
2 mod page_table;
3
4 use page_table::{PTEFlags};
5 pub use page_table::{PageTableEntry};
6

```

## 1.4 实现物理帧的管理和分配

- 设置物理内存的终止地址。 (os/src/config.rs)

```

1 pub const MEMORY_END: usize = 0x80800000;

```

- 实现物理帧管理。 (os/src/mm/frame\_allocator.rs)

```

106 FRAME_ALLOCATOR
107     .exclusive_access()
108     .alloc()
109     .map(|ppn| FrameTracker::new(ppn))
110 }
111
112 fn frame_dealloc(ppn: PhysPageNum) {
113     FRAME_ALLOCATOR
114     .exclusive_access()
115     .dealloc(ppn);
116 }
117
118 #[allow(unused)]
119 pub fn frame_allocator_test() {
120     let mut v: Vec<FrameTracker> = Vec::new();
121     for i in 0..5 {
122         let frame = frame_alloc().unwrap();
123         println!("{}", frame);
124         v.push(frame);
125     }
126     v.clear();
127     for i in 0..5 {
128         let frame = frame_alloc().unwrap();
129         println!("{}", frame);
130         v.push(frame);
131     }
132     drop(v);
133     println!("frame_allocator_test passed!");
134 }
135
-- INSERT --

```

- 增加sync模块

- os/src/sync/up.rs

```

1 use core::cell::{RefCell, RefMut};
2
3 /// Wrap a static data structure inside it so that we are
4 /// able to access it without any 'unsafe'.
5 ///
6 /// We should only use it in uniprocessor.
7 ///
8 /// In order to get mutable reference of inner data, call
9 /// 'exclusive_access'.
10 pub struct UPSafeCell<T> {
11     /// inner data
12     inner: RefCell<T>,
13 }
14
15 unsafe impl<T> Sync for UPSafeCell<T> {}
16
17 impl<T> UPSafeCell<T> {
18     /// User is responsible to guarantee that inner struct is only used in
19     /// uniprocessor.
20     pub unsafe fn new(value: T) -> Self {
21         Self { inner: RefCell::new(value) }
22     }
23     /// Panic if the data has been borrowed.
24     pub fn exclusive_access(&self) -> RefMut<'_, T> {
25         self.inner.borrow_mut()
26     }
27 }
28

```

- os/src/sync/mod.rs

```

1 mod up;
2
3 pub use up::UPSafeCell;

```

- /os/src/main.rs增加sync模块。

```

14 #[macro_use]
15 mod console;
16 mod lang_items;
17 mod sbi;
18 mod syscall;
19 mod trap;
20 mod loader;
21 mod config;
22 mod task;
23 mod timer;
24 mod mm;
25 mod sync;

```

- 物理帧管理测试

- os/src/config.rs

```

1 pub const PAGE_SIZE: usize = 0x1000;
2 pub const PAGE_SIZE_BITS: usize = 0xc;

```

- o os/src/mm/mod.rs

```
@94131a7dd3b2/mnt/os/src/ x + v
1 mod heap_allocator;
2 mod page_table;
3 mod address;
4 mod frame_allocator;
5
6 use page_table::{PTEFlags};
7 pub use page_table::{PageTableEntry};
8
9 use address::{VPNRange, StepByOne};
10 pub use address::{PhysAddr, VirtAddr, PhysPageNum, VirtPageNum};
11 pub use frame_allocator::{FrameTracker, frame_alloc};
12
13 pub fn init() {
14     heap_allocator::init_heap();
15     heap_allocator::heap_test();
16     frame_allocator::init_frame_allocator();
17     frame_allocator::frame_allocator_test();
18 }
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
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
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
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
26
```



### ○ 测试结果

```
[Finished release [optimized] target(s) in 2.02s  
[rustsbi] RustSBI version 0.2.0-alpha.6  
  
[RustSBI]  
  
[rustsbi] Implementation: RustSBI-QEMU Version 0.0.2  
[rustsbi-dtb] Hart count: cluster0 with 1 cores  
[rustsbi] misa: RV64ACDFIMSU  
[rustsbi] mideleg: ssoft, stimer, sext (0x222)  
[rustsbi] medeleg: ima, ia, bkpt, la, sa, uecall, ipage, lpage, spage (0xb1ab)  
[rustsbi] pmp0: 0x100000000 ..= 0x10001ffff (rwX)  
[rustsbi] pmp1: 0x800000000 ..= 0x8fffffffx (rwX)
```

```
[rustsbi] pmp1: 0x800000000 .:= 0x8fffffff (rwx)
[rustsbi] pmp2: 0x0 .:= 0xffffffffffffff (---)
qemu-system-riscv64: clint: invalid write: 00000004
[rustsbi] enter supervisor 0x80200000
heap_test passed!
last 721 Physical Frames.
FrameTracker: PPN=0x8052f
FrameTracker: PPN=0x80530
FrameTracker: PPN=0x80531
FrameTracker: PPN=0x80532
FrameTracker: PPN=0x80533
FrameTracker: PPN=0x80533
FrameTracker: PPN=0x80532
FrameTracker: PPN=0x80531
FrameTracker: PPN=0x80530
FrameTracker: PPN=0x8052f
frame_allocator_test passed!
[kernel] Hello, world!
power_3 [10000/200000]
power_5 [10000/200000]
power_5 [20000/200000]
power_5 [30000/200000]
power_5 [40000/200000]
power_5 [50000/200000]
power_5 [60000/200000]
power_5 [70000/200000]
power_5 [80000/200000]
power_5 [90000/200000]
power_5 [100000/200000]
power_5 [power_7 [10000/200000]
power_7 [20000/200000]
```

```
power_7 [20000/200000]
power_7 [30000/200000]
power_7 [40000/200000]
power_7 [power_3 [20000/200000]
power_7 [30000/200000]
power_7 [40000/200000]
power_7 [50000/200000]
power_7 [60000/200000]
power_7 [70000/200000]
110000/200000]
power_5 [120000/200000]
power_5 [130000/200000]
power_5 [140000/200000]
power_5 [150000/200000]
power_5 [160000/200000]
power_5 [170000/200000]
power_5 [180000/200000]
power_5 [190000/200000]
power_5 [200000/200000]
5*200000 = 670295496
Test power_5 OK!
[kernel] Application exited with code 0
50000/200000]
power_3 [80000/200000]
power_3 [90000/200000]
power_3 [100000/200000]
power_3 [110000/200000]
power_3 [120000/200000]
power_3 [130000/200000]
power_3 [140000/200000]
power_3 [150000/200000]
```

```

@94131a7dd3b2/mnt/os
power_3 [150000/200000]
power_3 [160000/200000]
power_3 [170000/200000]
power_3 [180000/200000]power_7 [60000/200000]
power_7 [70000/200000]
power_7 [80000/200000]
power_7 [90000/200000]
power_7 [100000/200000]
power_7 [110000/200000]
power_3 [190000/200000]
power_3 [200000/200000]
3^200000 = 87108973
Test power_3 OK!
[kernel] Application exited with code 0
]
power_7 [120000/200000]
power_7 [130000/200000]
power_7 [140000/200000]
power_7 [150000/200000]
power_7 [160000/200000]
power_7 [170000/200000]
power_7 [180000/200000]
power_7 [190000/200000]
power_7 [200000/200000]
7^200000 = 277895943
Test power_7 OK!
[kernel] Application exited with code 0
Test sleep OK!
[kernel] Application exited with code 0
Panicked at src/task/mod.rs:98 All applications completed!
[root@94131a7dd3b2 os]#

```

## 1.5 多级页表管理

- 实现页表的基本数据结构。 (os/src/mm/page\_table.rs)

```

55 use alloc::vec::Vec;
56 use alloc::vec;
57
58
59 pub struct PageTable {
60     root_ppn: PhysPageNum,
61     frames: Vec<FrameTracker>,
62 }
63
64 impl PageTable {
65     pub fn new() -> Self {
66         let frame = frame_alloc().unwrap();
67         PageTable {
68             root_ppn: frame.ppn,
69             frames: vec![frame],
70         }
71     }
72 }

```

- 实现建立和拆除虚实地址之间的映射关系。 (os/src/mm/page\_table.rs)

```

73
74 impl PageTable {
75     fn find_pte_create(&mut self, vpn: VirtPageNum) -> Option<&mut PageTableEntry> {
76         let idxs = vpn.indexes();
77         let mut ppn = self.root_ppn;
78         let mut result: Option<&mut PageTableEntry> = None;
79         for i in 0..3 {
80             let pte = &mut ppn.get_pte_array()[idxs[i]];
81             if i == 2 {
82                 result = Some(pte);
83                 break;
84             }
85             if !pte.is_valid() {
86                 let frame = frame_alloc().unwrap();
87                 *pte = PageTableEntry::new(frame.ppn, PTEFlags::V);
88                 self.frames.push(frame);
89             }
90             ppn = pte.ppn();
91         }
92         result
93     }
94
95     #[allow(unused)]
96     pub fn map(&mut self, vpn: VirtPageNum, ppn: PhysPageNum, flags: PTEFlags) {
97         let pte = self.find_pte_create(vpn).unwrap();
98         assert!(!pte.is_valid(), "vpn {:?} is mapped before mapping", vpn);
99         *pte = PageTableEntry::new(ppn, flags | PTEFlags::V);
100     }
101
102     #[allow(unused)]
103     pub fn unmap(&mut self, vpn: VirtPageNum) {
104         let pte = self.find_pte_create(vpn).unwrap();
105         assert!(pte.is_valid(), "vpn {:?} is invalid before unmapping", vpn);
106         *pte = PageTableEntry::empty();
107     }
108 }
-- INSERT --

```

108,2 Bot

- 实现手动查询页表的方法。 (os/src/mm/page\_table.rs)

```

108 }
109 impl PageTable {
110     /// Temporarily used to get arguments from user space.
111     pub fn from_token(satp: usize) -> Self {
112         Self {
113             root_ppn: PhysPageNum::from(satp & ((1usize << 44) - 1)),
114             frames: Vec::new(),
115         }
116     }
117
118     fn find_pte(&self, vpn: VirtPageNum) -> Option<PageTableEntry> {
119         let idxs = vpn.indexes();
120         let mut ppn = self.root_ppn;
121         let mut result: Option<PageTableEntry> = None;
122         for i in 0..3 {
123             let pte = &ppn.get_pte_array()[idxs[i]];
124             if i == 2 {
125                 result = Some(pte);
126                 break;
127             }
128             if !pte.is_valid() {
129                 return None;
130             }
131             ppn = pte.ppn();
132         }
133         result
134     }
135
136     pub fn translate(&self, vpn: VirtPageNum) -> Option<PageTableEntry> {
137         self.find_pte(vpn)
138             .map(|pte| {pte.clone()})
139     }
140     pub fn token(&self) -> usize {
141         8usize << 60 | self.root_ppn.0
142     }
143 }
-- INSERT --

```

## 1.6 实现地址空间抽象

- 实现地址空间抽象
  - 首先，以逻辑段MapArea描述一段连续地址的虚拟内存。 (os/src/mm/memory\_set.rs)

```

1 pub struct MapArea {
2     vpn_range: VPNRange,
3     data_frames: BTreeMap<VirtPageNum, FrameTracker>,
4     map_type: MapType,
5     map_perm: MapPermission,
6 }

```

- 接着，用MapType描述逻辑段内所有虚拟页号映射到物理页帧的方式。  
(os/src/mm/memory\_set.rs)

```

8 #[derive(Copy, Clone, PartialEq, Debug)]
9 pub enum MapType {
10     Identical,
11     Framed,
12 }

```

- 利用MapPermission控制逻辑段的访问方式，其是页表项标志位PTEFlags的子集  
(os/src/mm/memory\_set.rs)

```

14 bitflags! {
15     pub struct MapPermission: u8 {
16         const R = 1 << 1;
17         const W = 1 << 2;
18         const X = 1 << 3;
19         const U = 1 << 4;
20     }
21 }

```

- 实现地址空间，也就是一系列有关联的逻辑段 (os/src/mm/memory\_set.rs)

```

23 pub struct MemorySet {
24     page_table: PageTable,
25     areas: Vec<MapArea>,
26 }

```

- 实现MemorySet的方法 (os/src/mm/memory\_set.rs)

```

28 impl MemorySet {
29     pub fn new_base() -> Self {
30         Self {
31             page_table: PageTable::new(),
32             areas: Vec::new(),
33         }
34     }
35     pub fn token(&self) -> usize {
36         self.page_table.token()
37     }
38     /// Assume that no conflicts.
39     pub fn insert_framed_area(&mut self, start_va: VirtAddr, end_va: VirtAddr, permission: MapPermission) {
40         self.push(MapArea::new(
41             start_va,
42             end_va,
43             MapType::Framed,
44             permission,
45             None);
46     }
47     fn push(&mut self, mut map_area: MapArea, data: Option<&[u8]>) {
48         map_area.map(&mut self.page_table);
49         if let Some(data) = data {
50             map_area.copy_data(&mut self.page_table, data);
51         }
52         self.areas.push(map_area);
53     }
54     /// Mention that trampoline is not collected by areas.
55     fn map_trampoline(&mut self) {
56         self.page_table.map(
57             VirtAddr::from(TRAMPOLINE).into(),
58             PhysAddr::from(strampoline as usize).into(),
59             PTEFlags::R | PTEFlags::X,
60         );
61     }
62     pub fn activate(&self) {
63         let satp = self.page_table.token();
64         unsafe {
65             satp.write(satp);
66             asm!("sfence.vma");
67         }
68     }
69     pub fn translate(&self, vpn: VirtPageNum) -> Option<PageTableEntry> {
70         self.page_table.translate(vpn)
71     }
72 }
-- INSERT --

```

- 实现MapArea (os/src/mm/memory\_set.rs)

```

194 pub struct MapArea {
195     vpn_range: VPNRange,
196     data_frames: BTreeMap<VirtPageNum, FrameTracker>,
197     map_type: MapType,
198     map_perm: MapPermission,
199 }
200
201 impl MapArea {
202     pub fn new(
203         start_va: VirtAddr,
204         end_va: VirtAddr,
205         map_type: MapType,
206         map_perm: MapPermission
207     ) -> Self {
208         let start_vpn: VirtPageNum = start_va.floor();
209         let end_vpn: VirtPageNum = end_va.ceil();
210         Self {
211             vpn_range: VPNRange::new(start_vpn, end_vpn),
212             data_frames: BTreeMap::new(),
213             map_type,
214             map_perm,
215         }
216     }
217     pub fn map_one(&mut self, page_table: &mut PageTable, vpn: VirtPageNum) {
218         let ppn: PhysPageNum;
219         match self.map_type {
220             MapType::Identical => {
221                 ppn = PhysPageNum(vpn.0);
222             }
223             MapType::Framed => {

```

- 实现创建内核地址空间的方法new\_kernel. (os/src/mm/memory\_set.rs)

此处容器id更换了是因为我重装了docker，之前的容器丢失了。所以重新导入镜像，创建了新的容器。

```
20 extern "C" {
21     fn stext();
22     fn etext();
23     fn srodata();
24     fn erodata();
25     fn sdata();
26     fn edata();
27     fn sbss_with_stack();
28     fn ebss();
29     fn ekernel();
30     fn strampoline();
31 }

39 pub struct MemorySet {
40     page_table: PageTable,
41     areas: Vec<MapArea>,
42 }
43
44 impl MemorySet {
45
46     pub fn new_kernel() -> Self {
47         let mut memory_set = Self::new_base();
48         memory_set.map_trampoline();
49         println!(".text [{}:{}_{}]", stext as usize, etext as usize);
50         println!(".rodata [{}:{}_{}]", srodata as usize, erodata as usize);
51         println!(".data [{}:{}_{}]", sdata as usize, edata as usize);
52         println!(".bss [{}:{}_{}]", sbss_with_stack as usize, ebss as usize);
53         println!("mapping .text section");
54         memory_set.push(MapArea::new(
55             (stext as usize).into(),
56             (etext as usize).into(),
57             MapType::Identical,
58             MapPermission::R | MapPermission::X,
59         ), None);
60         println!("mapping .rodata section");
61         memory_set.push(MapArea::new(
62             (srodata as usize).into(),
63             (erodata as usize).into(),
64             MapType::Identical,
65             MapPermission::R,
66         ), None);
67         println!("mapping .data section");
68         memory_set.push(MapArea::new(
69             (sdata as usize).into(),
```

- 实现应用地址空间
  - 所有应用使用同一个链接脚本(user/src/linker.ld)

```
1 OUTPUT_ARCH(riscv)
2 ENTRY(_start)
3
4 BASE_ADDRESS = 0x0;
5
6 SECTIONS
7 {
8     . = BASE_ADDRESS;
9     .text : {
10         *(.text.entry)
11         *(.text .text.*)
12     }
13     . = ALIGN(4K);
14     .rodata : {
15         *(.rodata .rodata.*)
16         *(.srodata .srodata.*)
17     }
18     . = ALIGN(4K);
19     .data : {
20         *(.data .data.*)
21         *(.sdata .sdata.*)
22     }
23     .bss : {
24         *(.bss .bss.*)
25         *(.sbss .sbss.*)
26     }
27     /DISCARD/ : {
28         *(.eh_frame)
29         *(.debug*)
30     }
31 }

"linker.ld" [dos] 31L, 512B
```

- 修改精简loader子模块(os/src/loader.rs)

```
@3d686a71b33d:/mnt/os/src  ×  +  v
1 pub fn get_num_app() -> usize {
2     extern "C" { fn _num_app(); }
3     unsafe { (_num_app as usize as *const usize).read_volatile() }
4 }
5
6 pub fn get_app_data(app_id: usize) -> &'static [u8] {
7     extern "C" { fn _num_app(); }
8     let num_app_ptr = _num_app as usize as *const usize;
9     let num_app = get_num_app();
10    let app_start = unsafe {
11        core::slice::from_raw_parts(num_app_ptr.add(1), num_app + 1)
12    };
13    assert!(app_id < num_app);
14    unsafe {
15        core::slice::from_raw_parts(
16            app_start[app_id] as *const u8,
17            app_start[app_id + 1] - app_start[app_id]
18        )
19    }
20 }
```

- 还需要解析ELF格式的数据，从而得到一个完整的应用地址空间(os/src/mm/memory\_set.rs)

```
@3d686a71b33d:/mnt/os/src/  ×  +  v
123
124 pub fn from_elf(elf_data: &[u8]) -> (Self, usize, usize) {
125     let mut memory_set = Self::new_bare();
126     memory_set.map_trampoline();
127     let elf = xmas_elf::ElfFile::new(elf_data).unwrap();
128     let elf_header = elf.header;
129     let magic = elf_header.pt1.magic;
130     assert_eq!(magic, [0x7f, 0x45, 0x4c, 0x46], "invalid elf!");
131     let ph_count = elf_header.pt2.ph_count();
132     let mut max_end_vpn = VirtPageNum(0);
133     for i in 0..ph_count {
134         let ph = elf.program_header(i).unwrap();
135         if ph.get_type().unwrap() == xmas_elf::program::Type::Load {
136             let start_va: VirtAddr = (ph.virtual_addr() as usize).into();
137             let end_va: VirtAddr = ((ph.virtual_addr() + ph.mem_size()) as usize).into();
138             let mut map_perm = MapPermission::U;
139             let ph_flags = ph.flags();
140             if ph_flags.is_read() { map_perm |= MapPermission::R; }
141             if ph_flags.is_write() { map_perm |= MapPermission::W; }
142             if ph_flags.is_execute() { map_perm |= MapPermission::X; }
143             let map_area = MapArea::new(
144                 start_va,
145                 end_va,
146                 MapType::Framed,
147                 map_perm,
148             );
149             max_end_vpn = map_area.vpn_range.get_end();
150             memory_set.push(
151                 map_area,
152                 Some(&elf.input[ph.offset() as usize..(ph.offset() + ph.file_size()) as usize])
153             );
154         }
155     }
156     -- INSERT --
157     153,1 44%
```

- 在配置文件Cargo.toml增加依赖(os/Cargo.toml)

```
@3d686a71b33d:/mnt/os  ×  +  v
1 [package]
2 name = "os"
3 version = "0.1.0"
4 edition = "2021"
5
6 # See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
7
8 [dependencies]
9 riscv = { git = "https://github.com/rcore-os/riscv", features = ["inline-asm"] }
10 lazy_static = { version = "1.4.0", features = ["spin_no_std"] }
11 buddy_system_allocator = "0.6"
12 spin = "0.7.0"
13 bitflags = "1.2.1"
14 xmas-elf = "0.7.0"
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
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
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
2
```

- 实现memory\_set子模块增加代码(os/src/mm/memory\_set.rs)

```
1 use core::arch::asm;
2 use super::{PageTable, PageTableEntry, PTEFlags};
3 use super::{VirtPageNum, VirtAddr, PhysPageNum, PhysAddr};
4 use super::{FrameTracker, frame_alloc};
5 use super::{VPNRange, StepByOne};
6 use alloc::collections::BTreeMap;
7 use alloc::vec::Vec;
8 use riscv::register::satp;
9 use alloc::sync::Arc;
10 use lazy_static::*;
11 use crate::sync::UPSafeCell;
12 use crate::config::{
13     MEMORY_END,
14     PAGE_SIZE,
15     TRAMPOLINE,
16     TRAP_CONTEXT,
17     USER_STACK_SIZE
18 };
```

- os/src/config.rs

```
9 pub const TRAMPOLINE: usize = usize::MAX - PAGE_SIZE + 1;
10 pub const TRAP_CONTEXT: usize = TRAMPOLINE - PAGE_SIZE;
```

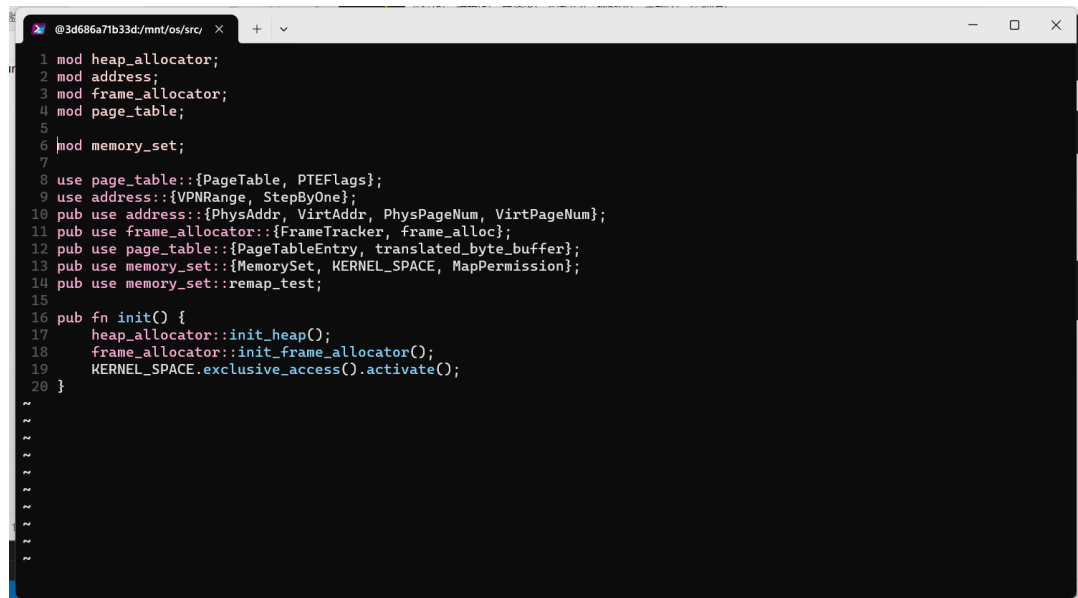
## 1.7 实现基于地址空间的分时多任务

- 建立基于分页模式的虚拟地址空间

- 创建内核地址空间。(os/src/mm/memory\_set.rs)

```
38 lazy_static! {
39     pub static ref KERNEL_SPACE: Arc<UPSafeCell<MemorySet>> = Arc::new(unsafe {
40         UPSafeCell::new(MemorySet::new_kernel())
41     });
42 }
```

- 在rust\_main中进行内存管理子系统的初始化(os/src/mm/mod.rs)



```
1 mod heap_allocator;
2 mod address;
3 mod frame_allocator;
4 mod page_table;
5
6 mod memory_set;
7
8 use page_table::{PageTable, PTEFlags};
9 use address::{VPNRange, StepByOne};
10 pub use address::{PhysAddr, VirtAddr, PhysPageNum, VirtPageNum};
11 pub use frame_allocator::{FrameTracker, frame_alloc};
12 pub use page_table::{PageTableEntry, translated_byte_buffer};
13 pub use memory_set::{MemorySet, KERNEL_SPACE, MapPermission};
14 pub use memory_set::remap_test;
15
16 pub fn init() {
17     heap_allocator::init_heap();
18     frame_allocator::init_frame_allocator();
19     KERNEL_SPACE.exclusive_access().activate();
20 }
21
22 ~~~~~
23 ~~~~~
24 ~~~~~
25 ~~~~~
26 ~~~~~
27 ~~~~~
28 ~~~~~
29 ~~~~~
30 ~~~~~
31 ~~~~~
32 ~~~~~
33 ~~~~~
34 ~~~~~
35 ~~~~~
36 ~~~~~
37 ~~~~~
38 ~~~~~
39 ~~~~~
40 ~~~~~
41 ~~~~~
42 ~~~~~
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 ~~~~~
```

- 检查内核地址空间的多级页表设置(os/src/mm/memory\_set.rs)

```
289 #[allow(unused)]
290 pub fn remap_test() {
291     let mut kernel_space = KERNEL_SPACE.exclusive_access();
292     let mid_text: VirtAddr = ((stext as usize + etext as usize) / 2).into();
293     let mid_rodata: VirtAddr = ((srodata as usize + erodata as usize) / 2).into();
294     let mid_data: VirtAddr = ((sdata as usize + edata as usize) / 2).into();
295     assert_eq!(
296         kernel_space.page_table.translate(mid_text.floor()).unwrap().writable(),
297         false
298     );
299     assert_eq!(
300         kernel_space.page_table.translate(mid_rodata.floor()).unwrap().writable(),
301         false,
302     );
303     assert_eq!(
304         kernel_space.page_table.translate(mid_data.floor()).unwrap().executable(),
305         false,
306     );
307     println!("remap_test passed!");
308 }
-- INSERT --
```

- 实现跳板机制

- 扩展Trap上下文(os/src/trap/context.rs)

```

1 use riscv::register::sstatus::{Sstatus, self, SPP};
2 #[repr(C)]
3 pub struct TrapContext {
4     pub x: [usize; 32],
5     pub sstatus: Sstatus,
6     pub sepc: usize,
7     pub kernel_satp: usize,
8     pub kernel_sp: usize,
9     pub trap_handler: usize,
10 }
11 impl TrapContext {
12     pub fn set_sp(&mut self, sp: usize) { self.x[2] = sp; }
13     pub fn app_init_context(
14         entry: usize,
15         sp: usize,
16         kernel_satp: usize,
17         kernel_sp: usize,
18         trap_handler: usize,
19     ) -> Self {
20         let mut sstatus = sstatus::read();
21         sstatus.set_spp(SPP::User);
22         let mut cx = Self {
23             x: [0; 32],
24             sstatus,
25             sepc: entry,
26             kernel_satp,
27             kernel_sp,
28             trap_handler,
29         };
30         cx.set_sp(sp);
31     }
32 }
33 -- INSERT --

```

- 实现地址空间的切换(os/src/trap/trap.S)

```

1 .altmacro
2 .macro SAVE_GP n
3     sd x\n, \n*8(sp)
4 .endm
5 .macro LOAD_GP n
6     ld x\n, \n*8(sp)
7 .endm
8 .section .text.trampoline
9 .globl __alltraps
10 .globl __restore
11 .align 2
12 __alltraps:
13     csrrw sp, sscratch, sp
14     # now sp->*TrapContext in user space, sscratch->user stack
15     # save other general purpose registers
16     sd x1, 1*8(sp)
17     # skip sp(x2), we will save it later
18     sd x3, 3*8(sp)
19     # skip tp(x4), application does not use it
20     # save x5~x31
21     .set n, 5
22     .rept 27
23         SAVE_GP %n
24     .set n, n+1
25 .endr
26 # we can use t0/t1/t2 freely, because they have been saved in TrapContext
27 csrr t0, sstatus
28 csrr t1, sepc
29 sd t0, 32*8(sp)
30 sd t1, 33*8(sp)
31 "trap.S" [dos] 69L, 1709B

```

- 将 trap.S 中的整段汇编代码放置在 .text.trampoline 段，并在调整内存布局的时候将它对齐到代码段的一个页面中。(os/src/linker.ld)

```

1 OUTPUT_ARCH(riscv)
2 ENTRY(_start)
3 BASE_ADDRESS = 0x80200000;
4
5 SECTIONS
6 {
7     . = BASE_ADDRESS;
8     skernel = .;
9
10    stext = .;
11    .text : {
12        *(.text.entry)
13        . = ALIGN(4K);
14        strampoline = .;
15        *(.text.trampoline);
16        . = ALIGN(4K);
17        *(.text .text.*)
18    }
19
20    . = ALIGN(4K);
21    etext = .;
22    srodata = .;
23    .rodata : {
24        *(.rodata .rodata.*)
25        *(.srodata .srodata.*)
26    }
27
28    . = ALIGN(4K);
29    erodata = .;
30    sdata = .;

```

- 加载和执行应用程序



- 修改任务子模块，并更新任务控制块的管理。(os/src/task/task.rs)

```
@3d686a71b33d/mnt/os/src x + v
1 use crate::mm::{MemorySet, MapPermission, PhysPageNum, KERNEL_SPACE, VirtAddr};
2 use crate::trap::{TrapContext, trap_handler};
3 use crate::config::{TRAP_CONTEXT, kernel_stack_position};
4 use super::TaskContext;
5
6 pub struct TaskControlBlock {
7     pub task_status: TaskStatus,
8     pub task_cx: TaskContext,
9     pub memory_set: MemorySet,
10    pub trap_cx_ppn: PhysPageNum,
11    pub base_size: usize,
12 }
13
14 impl TaskControlBlock {
15     pub fn get_trap_cx(&self) -> &'static mut TrapContext {
16         self.trap_cx_ppn.get_mut()
17     }
18     pub fn get_user_token(&self) -> usize {
19         self.memory_set.token()
20     }
21     pub fn new(elf_data: &[u8], app_id: usize) -> Self {
22         // memory_set with elf program headers/trampoline/trap context/user stack
23         let (memory_set, user_sp, entry_point) = MemorySet::from_elf(elf_data);
24         let trap_cx_ppn = memory_set
25             .translate(VirtAddr::from(TRAP_CONTEXT).into())
26             .unwrap()
27             .ppn();
28         let task_status = TaskStatus::Ready;
29         // map a kernel-stack in kernel space
30         let (kernel_stack_bottom, kernel_stack_top) = kernel_stack_position(app_id);
31     }
32 }
33
34 "task.rs" [dos] 63L, 2122B 1,1 Top
```

- os/src/config.rs

```
@3d686a71b33d/mnt/os/src x + v
1 pub const USER_STACK_SIZE: usize = 4096 * 2;
2 pub const KERNEL_STACK_SIZE: usize = 4096 * 2;
3 pub const KERNEL_HEAP_SIZE: usize = 0x30_0000;
4 pub const MEMORY_END: usize = 0x80800000;
5 pub const PAGE_SIZE: usize = 0x1000;
6
7 pub const PAGE_SIZE_BITS: usize = 0xc;
8
9 pub const TRAMPOLINE: usize = usize::MAX - PAGE_SIZE + 1;
10 pub const TRAP_CONTEXT: usize = TRAMPOLINE - PAGE_SIZE;
11
12
13 /// Return (bottom, top) of a kernel stack in kernel space.
14 pub fn kernel_stack_position(app_id: usize) -> (usize, usize) {
15     let top = TRAMPOLINE - app_id * (KERNEL_STACK_SIZE + PAGE_SIZE);
16     let bottom = top - KERNEL_STACK_SIZE;
17     (bottom, top)
18 }
19
20 pub const CLOCK_FREQ: usize = 12500000;
21
22 ~~~~~
23
24 "config.rs" [dos] 20L, 691B 1,1 ALL
```

- 在内核初始化的时候，需要将所有的应用程序加载到全局应用管理器中。同时，也要修改TaskManager的实现。(os/src/task/mod.rs)

```
@3d686a71b33d/mnt/os/src x + v
1 mod context;
2 mod switch;
3 mod task;
4
5 use crate::loader::{get_num_app, get_app_data};
6 use crate::trap::TrapContext;
7 use crate::sync::UPSafeCell;
8 use lazy_static::*;
9 use switch::_switch;
10 use task::{TaskControlBlock, TaskStatus};
11 use alloc::vec::Vec;
12
13 pub use context::TaskContext;
14
15 pub struct TaskManager {
16     num_app: usize,
17     inner: UPSafeCell<TaskManagerInner>,
18 }
19
20 struct TaskManagerInner {
21     tasks: Vec<TaskControlBlock>,
22     current_task: usize,
23 }
24
25 lazy_static! {
26     pub static ref TASK_MANAGER: TaskManager = {
27         println!("init TASK_MANAGER");
28         let num_app = get_num_app();
29         println!("num_app = {}", num_app);
30         let mut tasks: Vec<TaskControlBlock> = Vec::new();
31     }
32 }
33
34 "mod.rs" [dos] 152L, 4391B 1,1 Top
```

- 修改os/src/task/switch.S

```

1 |.altmacro
2 |.macro SAVE_SN n
3 |    sd s\@n, (\@n+2)*8(a0)
4 |.endm
5 |.macro LOAD_SN n
6 |    ld s\@n, (\@n+2)*8(a1)
7 |.endm
8 |.section .text
9 |.globl __switch
10 |__switch:
11 |    # __switch(
12 |    #     current_task_cx_ptr: *mut TaskContext,
13 |    #     next_task_cx_ptr: *const TaskContext
14 |    # )
15 |    # save kernel stack of current task
16 |    sd sp, 8(a0)
17 |    # save ra & s0~s11 of current execution
18 |    sd ra, 0(a0)
19 |    .set n, 0
20 |    .rept 12
21 |        SAVE_SN %n
22 |        .set n, n + 1
23 |    .endr
24 |    # restore ra & s0~s11 of next execution
25 |    ld ra, 0(a1)
26 |    .set n, 0
27 |    .rept 12
28 |        LOAD_SN %n
29 |        .set n, n + 1
30 |    .endr

```

- 修改switch.rs

```

1 |use core::arch::global_asm;
2 |
3 |global_asm!(include_str!("switch.S"));
4 |
5 |use super::TaskContext;
6 |
7 |extern "C" {
8 |    pub fn __switch(
9 |        current_task_cx_ptr: *mut TaskContext,
10 |        next_task_cx_ptr: *const TaskContext
11 |    );
12 |}

```

- 改进Trap的处理

- 首先修改init函数。然后在trap\_handler的开头增加set\_kernel\_trap\_entry。同时，在处理完trap后还要调用trap\_return 返回用户态。(os/src/trap/mod.rs)

```

18 |
19 |
20 |use crate::task::{
21 |    exit_current_and_run_next,
22 |    suspend_current_and_run_next,
23 |    current_user_token,
24 |    current_trap_cx,
25 |};
26 |use crate::timer::set_next_trigger;
27 |use crate::config::{TRAP_CONTEXT, TRAMPOLINE};
28 |
29 |global_asm!(include_str!("trap.S"));
30 |
31 |pub fn init() {
32 |    set_kernel_trap_entry();
33 |}
34 |
35 |fn set_kernel_trap_entry() {
36 |    unsafe {
37 |        stvec::write(trap_from_kernel as usize, TrapMode::Direct);
38 |    }
39 |}
40 |
41 |fn set_user_trap_entry() {
42 |    unsafe {
43 |        stvec::write(TRAMPOLINE as usize, TrapMode::Direct);
44 |    }
45 |}
46 |
47 |pub fn enable_timer_interrupt() {
-- INSERT --

```

- 在每一个应用程序第一次获得CPU权限时，内核栈顶放置在内核加载应用的时候构造的一个任务上下文。(os/src/task/context.rs)

```

1 use crate::trap::trap_return;
2
3 #[repr(C)]
4 pub struct TaskContext {
5     ra: usize,
6     sp: usize,
7     s: [usize; 12],
8 }
9
10 impl TaskContext {
11     pub fn zero_init() -> Self {
12         Self {
13             ra: 0,
14             sp: 0,
15             s: [0; 12],
16         }
17     }
18     pub fn goto_trap_return(kstack_ptr: usize) -> Self {
19         Self {
20             ra: trap_return as usize,
21             sp: kstack_ptr,
22             s: [0; 12],
23         }
24     }
25 }

```

"context.rs" [dos] 25L, 470B 1,1 All

- 改进sys\_write的实现
  - 由于地址空间的隔离，sys\_write无法直接方法应用空间的数据。为此，页表page\_table提供一个将应用地址空间的缓冲区转化为内核地址空间可以直接访问的辅助函数。

(os/src/mm/page\_table.rs)

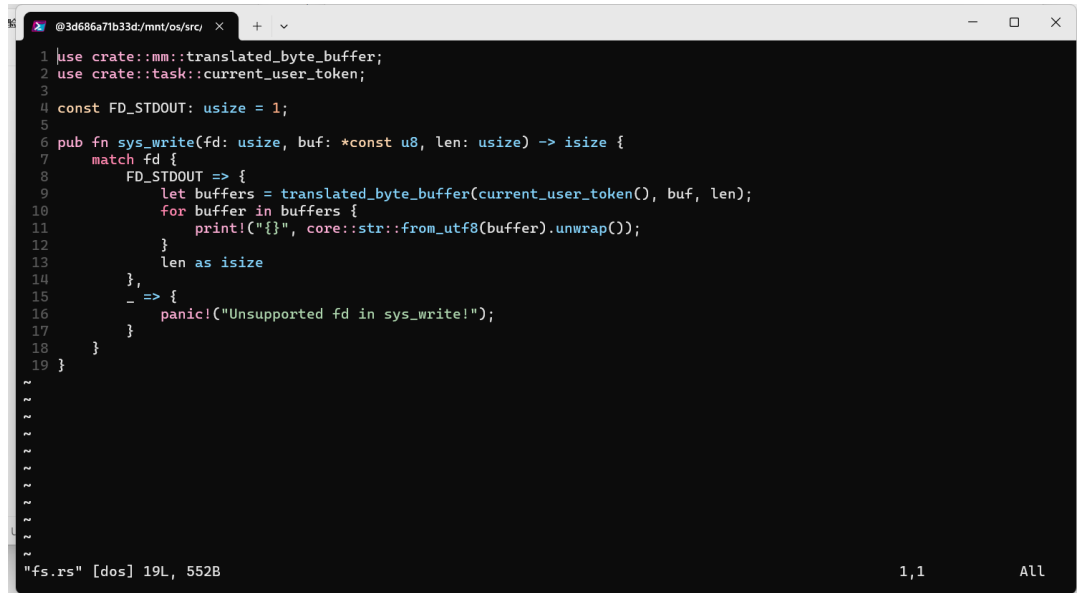
```

128 }
129 pub fn token(&self) -> usize {
130     8usize << 60 | self.root_ppn.0
131 }
132 }
133
134 pub fn translated_byte_buffer(token: usize, ptr: *const u8, len: usize) -> Vec<&'static mut [u8]> {
135     let page_table = PageTable::from_token(token);
136     let mut start = ptr as usize;
137     let end = start + len;
138     let mut v = Vec::new();
139     while start < end {
140         let start_va = VirtAddr::from(start);
141         let mut vpn = start_va.floor();
142         let ppn = page_table
143             .translate(vpn)
144             .unwrap()
145             .ppn();
146         vpn.step();
147         let mut end_va: VirtAddr = vpn.into();
148         end_va = end_va.min(VirtAddr::from(end));
149         if end_va.page_offset() == 0 {
150             v.push(&mut ppn.get_bytes_array()[start_va.page_offset()..]);
151         } else {
152             v.push(&mut ppn.get_bytes_array()[start_va.page_offset()..end_va.page_offset()]);
153         }
154         start = end_va.into();
155     }
156     v
157 }

```

157,1 Bot

- 修改sys\_write系统调用。(os/src/syscall/fs.rs)

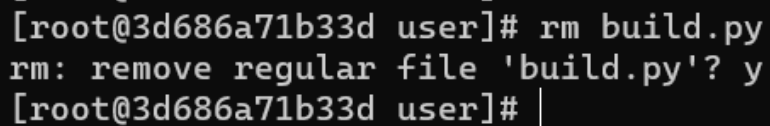


```
1 use crate::mm::translated_byte_buffer;
2 use crate::task::current_user_token;
3
4 const FD_STDOUT: usize = 1;
5
6 pub fn sys_write(fd: usize, buf: *const u8, len: usize) -> isize {
7     match fd {
8         FD_STDOUT => {
9             let buffers = translated_byte_buffer(current_user_token(), buf, len);
10             for buffer in buffers {
11                 print!("{}", core::str::from_utf8(buffer).unwrap());
12             }
13             len as isize
14         },
15         _ => {
16             panic!("Unsupported fd in sys_write!");
17         }
18     }
19 }
```

"fs.rs" [dos] 19L, 552B 1,1 All

## 1.8 修改应用程序

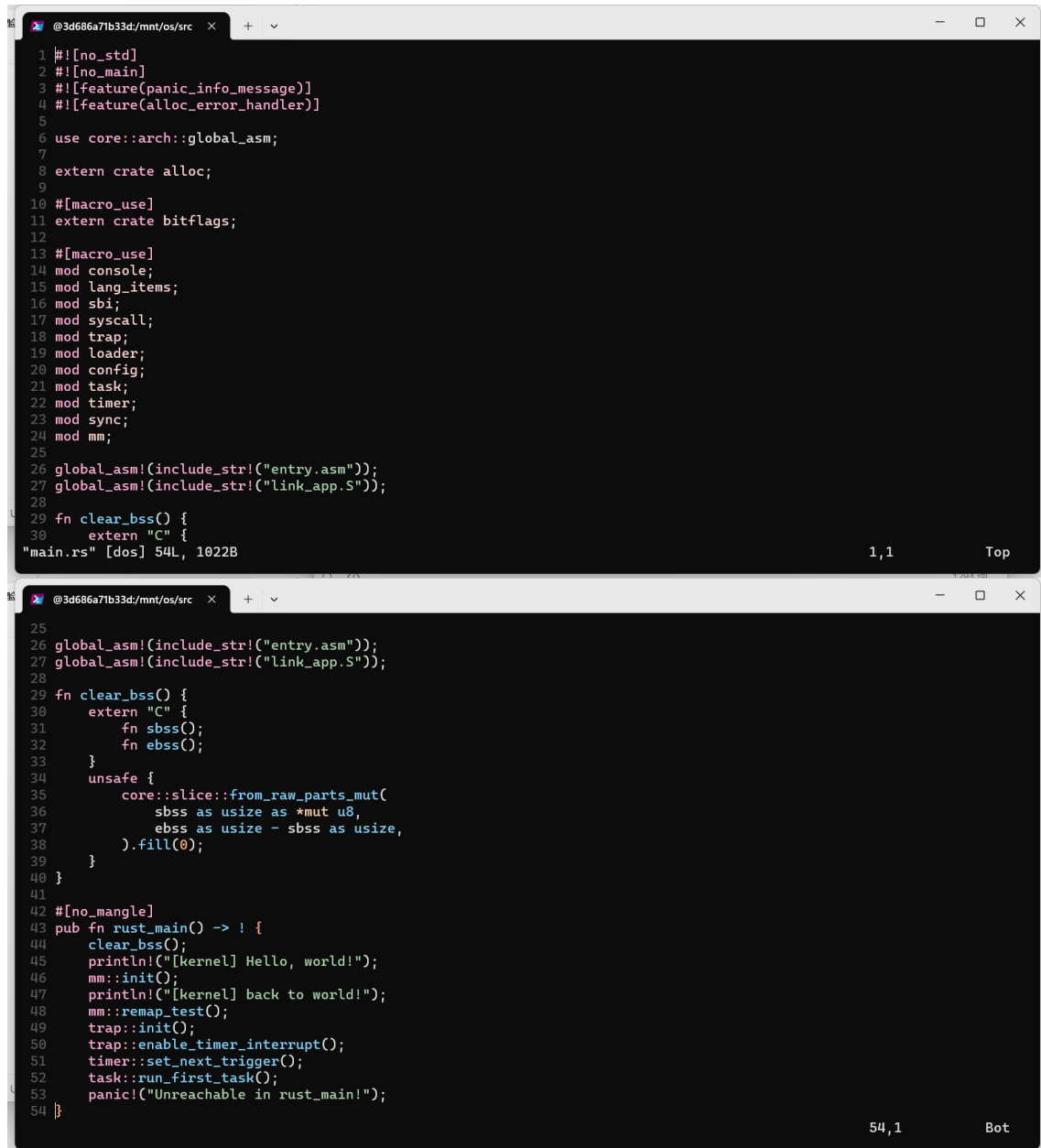
- 删除user/src/lib.rs中的clear\_bss()
- 删除build.py



```
[root@3d686a71b33d user]# rm build.py
rm: remove regular file 'build.py'? y
[root@3d686a71b33d user]# |
```

## 1.9 修改main.rs

- os/src/main.rs



```
1  #![no_std]
2  #![no_main]
3  #![feature(panic_info_message)]
4  #![feature(alloc_error_handler)]
5
6  use core::arch::global_asm;
7
8  extern crate alloc;
9
10 #[macro_use]
11 extern crate bitflags;
12
13 #[macro_use]
14 mod console;
15 mod lang_items;
16 mod sbi;
17 mod syscall;
18 mod trap;
19 mod loader;
20 mod config;
21 mod task;
22 mod timer;
23 mod sync;
24 mod mm;
25
26 global_asm!(include_str!("entry.asm"));
27 global_asm!(include_str!("link_app.S"));
28
29 fn clear_bss() {
30     extern "C" {
31         fn sbss();
32         fn ebss();
33     }
34     unsafe {
35         core::slice::from_raw_parts_mut(
36             sbss as usize as *mut u8,
37             ebss as usize - sbss as usize,
38         ).fill(0);
39     }
40 }
41
42 #[no_mangle]
43 pub fn rust_main() -> ! {
44     clear_bss();
45     println!("[kernel] Hello, world!");
46     mm::init();
47     println!("[kernel] back to world!");
48     mm::remap_test();
49     trap::init();
50     trap::enable_timer_interrupt();
51     timer::set_next_trigger();
52     task::run_first_task();
53     panic!("Unreachable in rust_main!");
54 }
```

- 修改os/build.rs

```
@3d686a71b33d/mnt/os
1 use std::io::{Result, Write};
2 use std::fs::{File, read_dir};
3
4 fn main() {
5     println!("cargo:rerun-if-changed=../user/src/");
6     println!("cargo:rerun-if-changed={}", TARGET_PATH);
7     insert_app_data().unwrap();
8 }
9
10 static TARGET_PATH: &str = "../user/target/riscv64gc-unknown-none-elf/release/";
11
12 fn insert_app_data() -> Result<> {
13     let mut f = File::create("src/link_app.S").unwrap();
14     let mut apps: Vec<_> = read_dir("../user/src/bin")
15         .unwrap()
16         .into_iter()
17         .map(|dir_entry| {
18             let mut name_with_ext = dir_entry.unwrap().file_name().into_string().unwrap();
19             name_with_ext.drain(name_with_ext.find('.').unwrap()..name_with_ext.len());
20             name_with_ext
21         })
22         .collect();
23     apps.sort();
24
25     writeln!(f, r#"
26     .align 3
27     .section .data
28     .global _num_app
29     _num_app:
30     .quad {}"#, apps.len())?;
31
32     for i in 0..apps.len() {
33         writeln!(f, r#"
34         .quad app_{}_start"#, i)?;
35     }
36     writeln!(f, r#"
37         .quad app_{}_end"#, apps.len() - 1)?;
38
39     for (idx, app) in apps.iter().enumerate() {
40         println!("app_{}: {}", idx, app);
41         writeln!(f, r#"
42         .section .data
43         .global app_{}_start
44         .global app_{}_end
45         .align 3
46         app_{}_start:
47         .incbin "{2}{1}"
48         app_{}_end: "#, idx, app, TARGET_PATH)?;
49     }
50     Ok(())
51 }
```

9,0-1 Top

```
21 }
22 .collect();
23 apps.sort();
24
25 writeln!(f, r#"
26 .align 3
27 .section .data
28 .global _num_app
29 _num_app:
30 .quad {}"#, apps.len())?;
31
32 for i in 0..apps.len() {
33     writeln!(f, r#"
34     .quad app_{}_start"#, i)?;
35 }
36 writeln!(f, r#"
37     .quad app_{}_end"#, apps.len() - 1)?;
38
39 for (idx, app) in apps.iter().enumerate() {
40     println!("app_{}: {}", idx, app);
41     writeln!(f, r#"
42     .section .data
43     .global app_{}_start
44     .global app_{}_end
45     .align 3
46     app_{}_start:
47     .incbin "{2}{1}"
48     app_{}_end: "#, idx, app, TARGET_PATH)?;
49 }
50 }
```

50,0-1 Bot

```
@3d686a71b33d:/mnt/os x v
cargo install cargo-binutils
    Updating `ustc` index
    Ignored package `cargo-binutils v0.3.6` is already installed, use --force to override
rustup component add rust-src
info: component 'rust-src' is up to date
rustup component add llvm-tools-preview
info: component 'llvm-tools' for target 'x86_64-unknown-linux-gnu' is up to date
make[1]: Entering directory '/mnt/user'
    Finished release [optimized] target(s) in 0.27s
rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unknown-none-elf/release/00power_3 --strip-all -O binary ta
rget/riscv64gc-unknown-none-elf/release/00power_3.bin; rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unkn
own-none-elf/release/01power_5 --strip-all -O binary target/riscv64gc-unknown-none-elf/release/01power_5.bin; rust-obj
copy --binary-architecture=riscv64 target/riscv64gc-unknown-none-elf/release/02power_7 --strip-all -O binary target/ris
cv64gc-unknown-none-elf/release/02power_7.bin; rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unknown-none
-elf/release/03sleep --strip-all -O binary target/riscv64gc-unknown-none-elf/release/03sleep.bin;
make[1]: Leaving directory '/mnt/user'
    Compiling os v0.1.0 (/mnt/os)
    Finished release [optimized] target(s) in 4.43s
[rustsbi] RustSBI version 0.2.0-alpha.6

┌───────────┐ ┌───────────┐ ┌───────────┐ ┌───────────┐ ┌───────────┐
│  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │
│  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │
└───────────┘ └───────────┘ └───────────┘ └───────────┘ └───────────┘

[rustsbi] Implementation: RustSBI-QEMU Version 0.0.2
[rustsbi-dtb] Hart count: cluster0 with 1 cores
[rustsbi] misa: RV64ACDFIMSU
[rustsbi] mideleg: ssoft, stimer, sext (0x222)
[rustsbi] medeleg: ima, ia, bkpt, la, sa, uecall, ipage, lpage, spage (0xb1ab)
[rustsbi] pmp0: 0x100000000 ..= 0x10001fff (rwx)
[rustsbi] pmp1: 0x800000000 ..= 0x8fffffff (rwx)
[rustsbi] pmp2: 0x0 ..= 0xfffffffffffff (---)
qemu-system-riscv64: clint: invalid write: 00000004
[rustsbi] enter supervisor 0x80200000
[kernel] Hello, world!
last 607 Physical Frames.
.text [0x80200000, 0x8020b000)
.rodata [0x8020b000, 0x8020f000)
.data [0x8020f000, 0x80290000)
.bss [0x80290000, 0x805a1000)
mapping .text section
mapping .rodata section
mapping .data section
mapping .bss section

@3d686a71b33d:/mnt/os x v
mapping .text section
mapping .rodata section
mapping .data section
mapping .bss section
mapping physical memory
[kernel] back to world!
remap_test passed!
init TASK_MANAGER
num_app = 4
power_3 [10000/300000]
power_3 [20000/300000]
power_3 [30000/300000]
power_3 [40000/300000]
power_3 [50000/300000]
power_3 [60000/300000]
power_3 [70000/300000]
power_3 [80000/300000]
power_3 [power_5 [10000/210000]
power_5 [20000/210000]
power_5 [30000/210000]
power_5 [40000/210000]
power_5 [50000/210000]
power_5 [60000/210000]
power_5 [70000/210000]
power_5 [80000/210000]
power_5 [90000/210000]
power_5 [100000/210000]
power_5 [power_7 [10000/240000]
power_7 [20000/240000]
power_7 [30000/240000]
power_7 [40000/240000]
power_7 [50000/240000]
power_7 [60000/240000]
power_7 [70000/240000]
power_7 [80000/240000]
power_7 [90000/240000]
power_7 [100000/240000]
power_7 [110000/240000]
power_7 [120000/240000]
power_7 [130000/240000]
power_7 [140000900000/300000]
power_3 [100000/300000]
power_3 [110000/300000]
power_3 [120000/300000]
power_3 [130000/300000]
power_3 [140000/300000]
```

```
@3d686a71b33d:/mnt/os
power_3 [200000/300000]
power_3 [210000/300000]
110000/210000]
power_5 [120000/210000]
power_5 [130000/210000]
power_5 [140000/210000]
power_5 [150000/210000]
power_5 [160000/210000]
power_5 [170000/210000]
power_5 [180000/210000]
power_5 [190000/210000]
power_5 [200000/210000]
power_5 [210000/210000]
5^210000 = 527227302(MOD 998244353)
Test power_5 OK!
[kernel] Application exited with code 0
power_3 [220000/300000]
power_3 [230000/300000]
power_3 [240000/300000]
power_3 [250000/300000]
power_3 [260000/300000]
power_3 [270000/300000]
power_3 [280000/300000]
power_3 [290000/300000]
power_3 [300000/300000]
3^300000 = 612461288(MOD 998244353)
Test power_3 OK!
[kernel] Application exited with code 0
/240000]
power_7 [150000/240000]
power_7 [160000/240000]
power_7 [170000/240000]
power_7 [180000/240000]
power_7 [190000/240000]
power_7 [200000/240000]
power_7 [210000/240000]
power_7 [220000/240000]
power_7 [230000/240000]
power_7 [240000/240000]
7^240000 = 304164893(MOD 998244353)
Test power_7 OK!
[kernel] Application exited with code 0
Test sleep OK!
[kernel] Application exited with code 0
[kernel] Panicked at src/task/mod.rs:115 All applications completed!
[root@3d686a71b33d os]#
```

## 二、思考问题

### 2.1 分析虚拟地址和物理地址的设计与实现；

- 虚拟地址和物理地址的数据结构的基本定义在os/src/mm/address.rs。
- 在os/src/mm/address.rs中实现和usize之间的相互转换、地址和页号之间的相互转换。
- 在os/src/mm/page\_table.rs中通过map方法建立虚实地址之间的映射关系，通过unmap方式实现了拆除映射关系。

### 2.2 分析物理帧是如何管理与分配的；

- 在os/src/mm/frame\_allocator.rs中，FrameTracker记录了物理帧的信息，包括页号，实现了FrameAllocator，用于分配和回收物理帧(alloc和dealloc)。
- 利用StackFrameAllocator类型的FrameAllocatorImpl实现栈管理物理页。实现frame\_alloc和frame\_dealloc分配和回收物理帧。
- 物理帧的分配先通过recycled检查回收的物理帧，优先从已经回收的列表中取出，如果没有回收的物理帧，就将当前位置设为新的物理帧。
- 物理帧的回收，首先检测物理帧是否合法，并放入recycled中。

### 2.3 分析内核的地址空间以及应用程序的地址空间是如何实现的；

- 实现在os/src/mm/memory\_set子模块
- 定义MapArea结构体，存储连续空间的虚拟页的范围、映射方式、映射权限。
- 实现memory\_set方法管理操作系统的地址空间。
- 内核地址空间



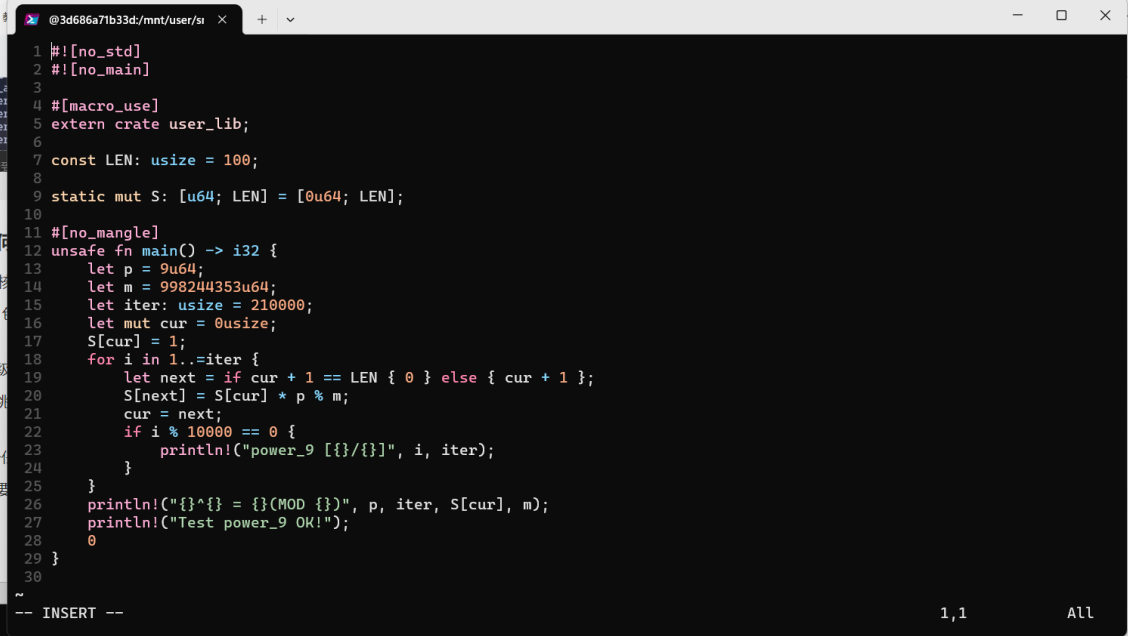
- 实现new\_kernel方法，用于创建内核地址空间。
- 定义一个memory\_set用于管理内核地址空间。
- 将多个MapArea 放进memory\_set，实现内核地址空间的分配和管理。
- 应用地址空间
  - user/src/linker.ld 中定义定义应用程序在内存中的布局，基础地址BASE\_ADDRESS设为0。
  - 在MemorySet中定义from\_elf方法，用于读取elf格式的可执行文件。解析页的大小和起始终止位置，放入map\_area中。将map\_area放入memory\_set中管理。

## 2.4 分析基于地址空间的分时多任务是如何实现的；

- 在os/src/mm/memory\_set.rs中创建统一的内核地址空间。
- 在rust\_main中进行内存管理子系统的初始化，包括heap\_allocator和frame\_allocator并激活内核空间。
- 实现remap\_test方法，检查内核地址空间的多级页表设置
- 扩展Trap上下文，实现地址空间的切换，建立跳板页面。在调整内存布局的时候将trap对齐到代码段的一个页面中。
- 通过 TaskControlBlock 任务管理器来管理每个任务的信息，并实行上下文切换。
- 实现TaskManager，在内核初始化的时候，需要将所有的应用程序加载到全局应用管理器中。

## 2.5 编写新的应用程序并测试验证结果。

- 实现了04power\_9.rs，计算9的210000次方mod998244353



```

1  #![no_std]
2  #![no_main]
3
4  #[macro_use]
5  extern crate user_lib;
6
7  const LEN: usize = 100;
8
9  static mut S: [u64; LEN] = [0u64; LEN];
10
11 #[no_mangle]
12 unsafe fn main() -> i32 {
13     let p = 9u64;
14     let m = 998244353u64;
15     let iter: usize = 210000;
16     let mut cur = 0usize;
17     S[cur] = 1;
18     for i in 1..=iter {
19         let next = if cur + 1 == LEN { 0 } else { cur + 1 };
20         S[next] = S[cur] * p % m;
21         cur = next;
22         if i % 10000 == 0 {
23             println!("power_9 [{} / {}]", i, iter);
24         }
25     }
26     println!("{}", p ^ iter % m);
27     println!("Test power_9 OK!");
28     0
29 }
30
-- INSERT --
1,1 All

```

- 实现了05\_fac.rs, 计算300000的阶层

```
@3d686a71b33d:/mnt/user/si x + v
1
2
3 #![no_std]
4 #![no_main]
5
6 #[macro_use]
7 extern crate user_lib;
8
9 const LEN: usize = 100;
10
11 static mut S: [u64; LEN] = [0u64; LEN];
12
13 #[no_mangle]
14 unsafe fn main() -> i32 {
15     let m = 1000000007u64;
16     let iter: usize = 300000;
17     let mut cur = 0usize;
18     S[cur] = 1;
19     for i in 1..=iter {
20         let next = if cur + 1 == LEN { 0 } else { cur + 1 };
21         S[next] = S[cur] * i as u64 % m;
22         cur = next;
23         if i % 10000 == 0 {
24             println!("fac [{} / {}]", i, iter);
25         }
26     }
27     println!("{!} = {}(MOD {})", iter, S[cur], m);
28     println!("Test fac OK!");
29     0
30 }
~
-- INSERT --
```

14,26 All

- 运行结果

```

@3d686a71b33d:/mnt/os
[rustsbi] enter supervisor 0x80200000
[kernel] Hello, world!
last 543 Physical Frames.
.text [0x80200000, 0x8020b000)
.rodata [0x8020b000, 0x8020f000)
.data [0x8020f000, 0x802d0000)
.bss [0x802d0000, 0x805e1000)
mapping .text section
mapping .rodata section
mapping .data section
mapping .bss section
mapping physical memory
[kernel] back to world!
remap_test passed!
init TASK_MANAGER
num_app = 6
power_3 [10000/300000]
power_3 [20000/300000]
power_3 [30000/300000]
power_3 [40000/300000]
power_3 [50000/300000]
power_3 [60000/300000]
power_3 [70000/300000]power_5 [10000/210000]
power_5 [20000/210000]
power_5 [30000/210000]
power_5 [40000/210000]
power_5 [50000/210000]
power_5 [60000/210000]
power_5 [70000/210000]
power_5 [80000/210000]
power_5 [90000/210000]
power_7 [10000/240000]

power_5 [60000/210000]
power_5 [70000/210000]
power_5 [80000/210000]
power_5 [90000/210000]
power_7 [10000/240000]
power_7 [20000/240000]
power_7 [30000/240000]
power_7 [40000/240000]
power_7 [50000/240000]
power_7 [60000/240000]
power_7 [70000/240000]
power_7 [80000/240000]
power_7 [90000/240000]
power_7 [100000/240000]power_9 [10000/210000]
power_9 [20000/210000]
power_9 [30000/210000]
power_9 [40000/210000]
power_9 [50000/210000]
power_9 [60000/210000]
power_9 [70000/210000]
power_9 [80000/210000]
power_9 [90000/210000]
power_9 [100000/210000]fac [10000/300000]
fac [20000/300000]
fac [30000/300000]
fac [40000/300000]
fac [50000/300000]
fac [60000/300000]
fac [70000/300000]
fac [80000/300000]
fac [90000/300000]
]

power_3 [90000/300000]
power_3 [100000/300000]
power_3 [110000/300000]
power_3 [120000/300000]
power_3 [130000/300000]
power_3 [140000/300000]
power_3 [150000/300000]
power_3 [160000/300000]
power_3 [170000/300000]
power_3 [180000/300000]
power_3 [power_5 [100000/210000]
power_5 [110000/210000]
power_5 [120000/210000]
power_5 [130000/210000]
power_5 [140000/210000]
power_5 [150000/210000]
power_5 [160000/210000]
power_5 [170000/210000]
]
power_7 [110000/240000]
power_7 [120000/240000]
power_7 [130000/240000]
power_7 [140000/240000]
power_7 [150000/240000]
power_7 [160000/240000]
power_7 [170000/240000]
power_7 [180000/240000]
power_7 [190000/240000]
power_7 [210000]
power_9 [110000/210000]
power_9 [120000/210000]
power_9 [130000/210000]

```

```
@3d686a71b33d:/mnt/os
power_9 [150000/210000]
power_9 [160000/210000]
power_9 [170000/210000]
power_9 [180000/210000]
power_9 [190000/210000]
power_9 [200000fac [100000/300000]
fac [110000/300000]
fac [120000/300000]
fac [130000/300000]
fac [140000/300000]
fac [150000/300000]
fac [160000/300000]
fac [170000/300000]
fac [180000/300000]
fac [190000/300000]
fac [200000190000/300000]
power_3 [200000/300000]
power_3 [210000/300000]
power_3 [220000/300000]
power_3 [230000/300000]
power_3 [240000/300000]
power_3 [250000/300000]
power_3 [260000/300000]
power_3 [270000/300000]
power_3 [280000/300000]
power_5 [180000/210000]
power_5 [190000/210000]
power_5 [200000/210000]
power_5 [210000/210000]
5^210000 = 527227302(MOD 998244353)
Test power_5 OK!
[kernel] Application exited with code 0

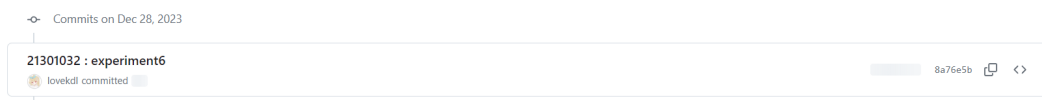
Test power_5 OK!
[kernel] Application exited with code 0
200000/240000]
power_7 [210000/240000]
power_7 [220000/240000]
power_7 [230000/240000]
power_9 [210000/210000]
9^210000 = 933220523(MOD 998244353)
Test power_9 OK!
[kernel] Application exited with code 0
/300000]
fac [210000/300000]
fac [220000/300000]
fac [230000/300000]
fac [240000/300000]
fac [power_3 [290000/300000]
power_3 [300000/300000]
3^300000 = 612461288(MOD 998244353)
Test power_3 OK!
[kernel] Application exited with code 0
230000/240000]
power_7 [240000/240000]
7^240000 = 304164893(MOD 998244353)
Test power_7 OK!
[kernel] Application exited with code 0
250000/300000]
fac [260000/300000]
fac [270000/300000]
fac [280000/300000]
fac [290000/300000]
fac [300000/300000]
300000! = 373281933(MOD 1000000007)

Test fac OK!
[kernel] Application exited with code 0
Test sleep OK!
[kernel] Application exited with code 0
[kernel] Panicked at src/task/mod.rs:115 All applications completed!
[root@3d686a71b33d os]#
```

- 验证成功

### 三、git截图

- git截图(<https://github.com/lovekdl/GardenerOS>)



### 四、其它说明

- 中途容器id更换是因为重装了docker，之前的容器丢失了。所以重新导入镜像，创建了新的容器。