

OS Lab2 Report

1. 个人信息

姓名：余帅杰

学号：181860077

邮箱：3121416933@qq.com

2. 实验进度

我完成了所有的内容，

实验内容

1. 补全文件系统
2. 修改中断向量表
3. 实现键盘按键的串口回显
4. 实现printf的处理例程
5. 完善格式化输出

最终的结果：

1. cp指令正常运行
2. 键盘按键成功输出
3. 通过测试样例printf，多种格式正常输出

代码修改位置

1. func.c文件里的cp函数
2. idt.c里的initIdt函数
3. irqHandle.c里的syscallPrint函数
4. syscall.c里的printf函数
5. irqHandle.c里的keyboardHandle函数
6. 修改main函数加入样例

3. 实验思路

文件系统

文件系统是补全cp指令

先阅读linux文件系统的概论，随后看到上下文有mkdir和touch指令，然后举一反三写出cp指令

基本思路就是

1. 读取超级块内容
2. 先解析源路径
3. 然后读取inode，fatherNode等等的的数据
4. 为目的文件分配inode块
5. 打开源路径文件后拷贝数据

键盘按键回显

阅读serial.c, 里面提供了显示的接口

阅读keyboard.c里面提供了读取键盘数据的接口

思路如下:

1. 先去idt.c里加上对应的中断号和处理函数
2. 利用keyboard.c里的getKeyCode获取键码
3. 利用keyboard.c里的getChar对上述的键码进行处理
4. 利用serial.c里的putChar进行输出

printf处理例程

阅读代码可以知道int80指令最后会一路落到syscallPrint函数

且利用手册中的代码块可以实现对对应位置的字符打印, 则处理换行和滚屏问题

思路如下

1. 得到数据字符
2. 判断是否为换行, 如果是就换行
3. 不是换行就对应的打印到位置
4. 打印完后纵坐标加1, 判断是否满了换行
5. 最后判断是否需要滚屏

printf格式化输出

printf有四种格式需要注意

阅读代码发现框架代码提供了三种函数

分析后发现三种函数是分别在buffer中插入格式化符号的对应数据。那么就利用起来。还有就是代码里的paralist, 应该是参数的指针

思路如下

1. 对paralist+=4指向第一个参数
2. 开始遍历format数组
3. 如果是%则获取下一个字符并对应的移动指针
4. 对上述获取的字符进行分析, 获取参数并调用对应的函数
5. 如果不是就直接拷贝到buffer同时移动指针即可
6. 最后判断count, 输出所有buffer里的数据

paralist是指针指向栈里的数据, 由于是按4对齐, 每一次读取后就是移动4, 然后按照格式符的种类调整类型读取即可

4. 实验结果

下面是文件系统的运行结果

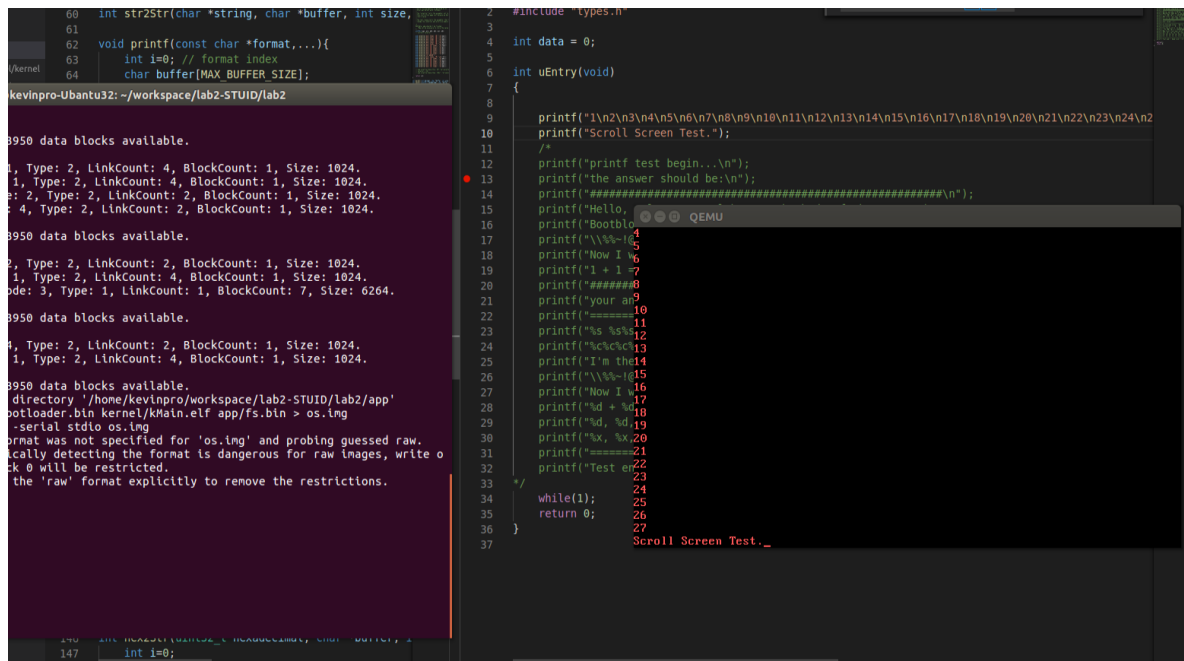
可以看到最后的一次ls和手册一致

```

FORMAT success.
1023 inodes and 3959 data blocks available.
mkdir /boot
MKDIR success.
1022 inodes and 3958 data blocks available.
cp /boot/initrd
cp success.
1021 inodes and 3950 data blocks available.
mkdir /usr
MKDIR success.
1020 inodes and 3949 data blocks available.
ls /
Name: ., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
Name: .., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
Name: boot, Inode: 2, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
Name: usr, Inode: 4, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
LS success.
1020 inodes and 3949 data blocks available.
ls /boot
Name: ., Inode: 2, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
Name: .., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
Name: initrd, Inode: 3, Type: 1, LinkCount: 1, BlockCount: 8, Size: 7380.
LS success.
1020 inodes and 3949 data blocks available.
ls /usr
Name: ., Inode: 4, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
Name: .., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
LS success.
1020 inodes and 3949 data blocks available.
make[1]: Leaving directory '/home/kevinpro/workspace/lab2-STUID/lab2/app'
cat bootloader/bootloader.bin kernel/kMain.elf app/fs.bin > os.img
qemu-system-i386 -serial stdio os.img
WARNING: Image format was not specified for 'os.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
8192

```

下面的是测试滚屏的，可以看到成功的显示



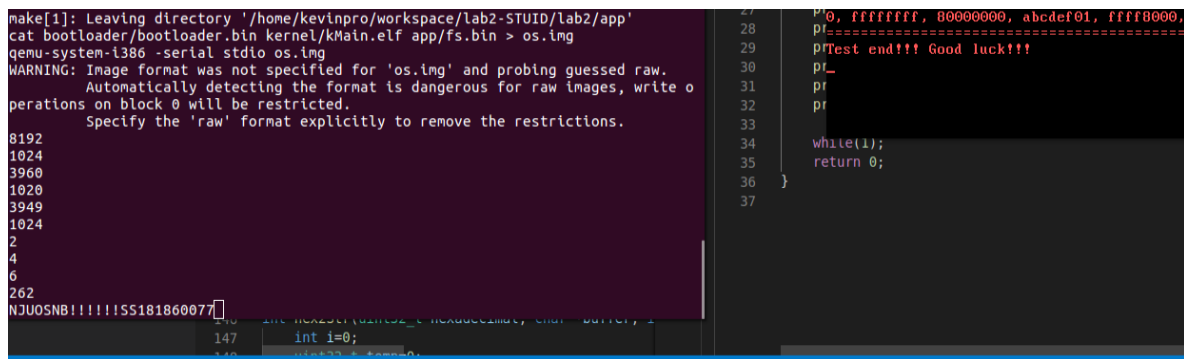
```

60 int str2Str(char *string, char *buffer, int size,
61
62 void printf(const char *format,...){
63     int i=0; // format index
64     char buffer[MAX_BUFFER_SIZE];
65
66 kevinpro-Ubuntu32: ~/workspace/lab2-STUID/lab2
67
68 3950 data blocks available.
69
70 ., Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
71 .., Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
72 boot, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
73 usr, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
74
75 3950 data blocks available.
76
77 ., Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
78 .., Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
79 boot, Type: 1, LinkCount: 1, BlockCount: 7, Size: 6264.
80
81 3950 data blocks available.
82
83 ., Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
84 .., Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
85
86 3950 data blocks available.
87
88 directory '/home/kevinpro/workspace/lab2-STUID/lab2/app'
89 cat bootloader/bootloader.bin kernel/kMain.elf app/fs.bin > os.img
90 -serial stdio os.img
91 format was not specified for 'os.img' and probing guessed raw.
92 Automatically detecting the format is dangerous for raw images, write o
93 perations on block 0 will be restricted.
94 Specify the 'raw' format explicitly to remove the restrictions.
95
96 8192
97 1024
98 3960
99 1020
100 3949
101 1024
102 2
103 4
104 6
105 262
106 NJUOSNB!!!!!!SS181860077
107
108 147 int i=0;
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

```

下面是键盘输出的结果

随便在qemu中打字，在teminal中成功的输出



```

make[1]: Leaving directory '/home/kevinpro/workspace/lab2-STUID/lab2/app'
cat bootloader/bootloader.bin kernel/kMain.elf app/fs.bin > os.img
qemu-system-i386 -serial stdio os.img
WARNING: Image format was not specified for 'os.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
8192
1024
3960
1020
3949
1024
2
4
6
262
NJUOSNB!!!!!!SS181860077

```

```

name: .. Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
name: boot, Inode: 1, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
name: usr, Inode: 4, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
S success.
920 Inodes and 3949 data blocks available.
ls /boot
name: .. Inode: 2, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
name: .. Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
name: initrd, Inode: 3, Type: 1, LinkCount: 1, BlockCount: 8, Size: 7380.
S success.
920 Inodes and 3949 data blocks available.
ls /usr
name: .. Inode: 4, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
name: .. Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
S success.
920 Inodes and 3949 data blocks available.
make[1]: Leaving directory '/home/kevinpro/workspace/lab2-STUID/lab2/app'
at bootloader/bootloader.bin kernel/Main.elf app/fs.bin > os.img
menu-system-1360 -serial stdio os.img
WARNING: Image format was not specified for 'os.img' and probing guessed raw.
        Automatically detecting the format is dangerous for raw images, write o
erations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
```

问题1

由于只是在外层到内层（低特权级到高特权级）切换时，新堆栈才会从TSS中取得，所以TSS 并没有位于最外层 ring3 的堆栈信息；同时在R3进入R0的时候会把信息压入堆栈（同时自动加载ring0的esp和ss（在TSS里）），出来的时候只要从堆栈恢复就可以。

问题2

块大小	1KB	2KB	4KB
一个文件最大大小	16GB	256GB	2TB
文件系统最大大小	4TB	8TB	16TB

每一个block可以记录 $1000/4=256$ 个号码

单层 $256 \times 1K = 256K$

双层 $256 \times 256 \times 256 \times 1K = 16777216k$

单文件最大总量 = $(12 + 256 + 65536 + 16777216) / (1024 * 1024) = 16.06\text{G}$

同理计算有2KB的

2000/4=512则有最大总量256.50G

同理4KB的有4.00T

PS: 当block单位容量为4K时, 由于文件系统本身的限制(2T)和原结果有写不同

参考: <https://www.cnblogs.com/justmine/p/9128730.html>

问题3

我们在使用eax, ecx, edx, ebx, esi, edi前将寄存器的值保存到了栈中, 如果去掉保存和恢复的步骤, 从内核返回之后会不会产生不可恢复的错误?

如果内核用到了这些寄存器, 那么值就被改变, 没有备份则可能出错