

## آزمایشگاه سیستم عامل

### گزارش پروژه یک

- آرین باستانی - 810100088
- مهدیار هرندی - 810199596
- محمدرضا نعمتی - 810100226

- <https://github.com/mmd-nemati/OS-Lab-Projects>
- Last commit hash: a5cc9a44777e5762bbc119ae6066b981b6ee2def

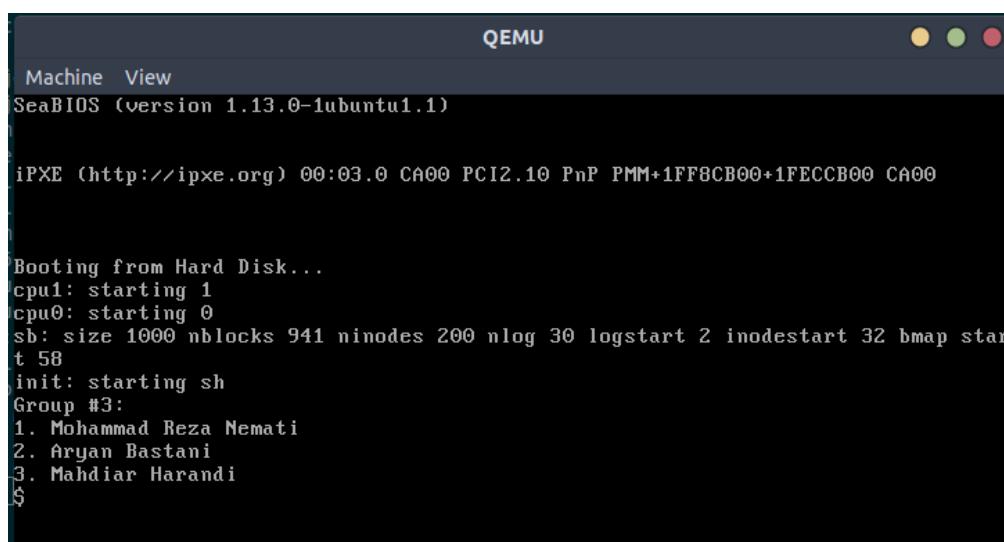
برای پرینت شدن اسمی گروه در ابتدای ران شدن xv6, کافی است فایل init.c را بصورت زیر ادیت

کنیم (خط 24).

```

15     if(open("console", O_RDWR) < 0){
16         mknod("console", 1, 1);
17         open("console", O_RDWR);
18     }
19     dup(0); // stdout
20     dup(0); // stderr
21
22     for(;;){
23         printf(1, "init: starting sh\n");
24 |         printf(1, "Group #3:\n1. Mohammad Reza Nemati\n2. Aryan Bastani\n3. Mahdiar Harandi\n");
25         pid = fork();
26         if(pid < 0){
27             printf(1, "init: fork failed\n");
28             exit();
29         }
30         if(pid == 0){
```

با شبیه سازی دوباره سیستم عامل می بینیم که اسمی اعضا در ابتدای استارت کنسول پرینت شده است.



## قابلیت های اضافه شده به کنسول:

**Ctrl + B**: برای پیاده سازی این دستور باید یک پوینتر دیگر به input اضافه کنیم تا علاوه بر معلوم بودن انتها و ابتدای بافر اینپوت برای نوشتن، جایی که در آن در حال نوشتن هستیم نیز معلوم باشد. اسم این پوینتر را cursor می گذاریم:

```
struct {
    char buf[INPUT_BUF];
    uint r; // Read index
    uint w; // Write index
    uint e; // Edit index
    uint c; // Cursor index
} input;
```

و با هر بار کم یا اضافه کردن input.r باید input.e را نیز تغییر دهیم چرا که اگر Ctrl + B زده نشده باشد باید Edit همواره برابر با cursor باشد.

و همچنین برای نوشتن و یا پاک کردن در جایی بجز ته نوشته های قبلی (مثلا نوشتن بعد از زدن چند B (Ctrl + B) باید کarakتر های داخل crt را نیز شیفت داد. بنابراین وقتی با زدن B متغیر pos را تغییر من دهیم با استفاده از متغیر گلوبال به نام distance می توانیم به ته نوشته ها دسترسی داشته باشیم:

```
+ static int distance = 0;
+
```

و باید با هر بار دادن ورودی جدید به کنسول (مثلًا پس از '\n') متغیر pos را دوباره به ته کاراکترها برگردانده و distance را صفر کنیم:

```

if(c == '\n'){
    pos += distance;
    distance = 0;
    input.c = input.e;
    pos += 80 - pos%80;
}

```

بنابراین با هر بار زدن Ctrl + B کافیست متغیرهای cursor و pos را - و متغیر distance را ++ کنیم.  
 قبلش چک می‌کنیم که به خط قبلی نرویم و یا اینکه pos برابر با صفر نباشد)

```

else if(c == CONTROL_B && pos > 0 &&
        input.buf[(input.c-1) % INPUT_BUF] != '\n'){
    if(input.c != input.w){
        input.c--;
        pos--;
        distance++;
    }
}

```

و سپس برای نوشتمن در کنسول، هر دفعه چک می‌کنیم اگر distance بزرگتر از صفر بود (یا input.c از input.e کوچکتر بود) بنابراین در حال نوشتمن در بین کاراکترهای دیگر هستیم و باید با نوشتمن هر کاراکتر جدید، کاراکترهای سمت راست را در input و crt یکی به راست شیفت دهیم.

: input buffer شیفت دادن در

```

else{
    ushort next_elmnt, curr_elmnt = input.buf[input.c % INPUT_BUF];
    input.e++;
    for(int i = input.c; i < input.e; i++){
        next_elmnt = input.buf[(i + 1) % INPUT_BUF];
        input.buf[(i + 1) % INPUT_BUF] = curr_elmnt;
        curr_elmnt = next_elmnt;
    }
    input.buf[input.c++ % INPUT_BUF] = c;
}

```

crt شیفت دادن در : crt

```

if(distance > 0){
    ushort next_elmnt, curr_elmnt = crt[pos];
    for(int i = pos; i <= pos + distance; i++){
        next_elmnt = crt[i + 1];
        crt[i + 1] = curr_elmnt;
        curr_elmnt = next_elmnt;
    }
}

```

و برای backspace در بین کاراکتر های دیگر پس از زدن Ctrl + B نیز همین کار را تکرار می کنیم ولی اینبار

کاراکتر ها را به چپ شیفت می دهیم:

```
    case C('H'): case '\x7f': // Backspace
        if(input.c != input.w && input.buf[(input.c-1) % INPUT_BUF] != '\n'){
            if(input.c != input.e){
                ushort per_elmnt, curr_elmnt = input.buf[input.e % INPUT_BUF];
                for(int i = input.e; i >= input.c; i--){
                    per_elmnt = input.buf[(i - 1) % INPUT_BUF];
                    input.buf[(i - 1) % INPUT_BUF] = curr_elmnt;
                    curr_elmnt = per_elmnt;
                }
            }
            input.c--;
            input.e--;
            consputc(BACKSPACE);
        }
        break;
```

```
if(pos > 0){
    if(distance > 0){
        ushort per_elmnt, curr_elmnt = crt[pos + distance];
        for(int i = pos + distance; i >= pos; i--){
            per_elmnt = crt[i - 1];
            crt[i - 1] = curr_elmnt;
            curr_elmnt = per_elmnt;
        }
    }
    pos--;
}
```

اجرای این دستور:

```
'init: starting sh
Group #3:
1. Mohammad Reza Nemati
2. Aryan Bastani
3. Mahdiar Harandi
$ Hello_World
```

```
'init: starting sh
Group #3:
1. Mohammad Reza Nemati
2. Aryan Bastani
3. Mahdiar Harandi
$ Hello!!!!_World
```

```
'init: starting sh
Group #3:
1. Mohammad Reza Nemati
2. Aryan Bastani
3. Mahdiar Harandi
$ !!!! World
```

: مانند دستور قبلی عمل می کنیم ولی اینبار با هر بار اجرای این دستور باید متغیر های **Ctrl+F** را - - و متغیر های **pos** و **cursor** را یکی زیاد کنیم ( قبلش چک می کنیم که به خط بعدی نزولیم):

```
        else if(c == CONTROL_F){
            input.c++;
            pos++;
            distance--;
        }
```

و چون در قسمت قبل نوشتن و پاک کردن در بین کاراکتر های دیگر را هندل کردیم نیازی به کار دیگری در این قسمت نیست.

و اجرای این دستور:

```
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $ cat /etc/group | grep ^pi
pi:x:100:pi
pi@raspberrypi:~ $ cat /etc/group | grep ^pi > /etc/group.pi
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $ cat /etc/group.pi
pi:x:100:pi
pi@raspberrypi:~ $ rm /etc/group.pi
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $
```

```
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $ cat /etc/group | grep ^pi
pi:x:100:pi
pi@raspberrypi:~ $ cat /etc/group | grep ^pi > /etc/group.pi
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $ cat /etc/group.pi
pi:x:100:pi
pi@raspberrypi:~ $ rm /etc/group.pi
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $
```

```
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $ cat /etc/group | grep ^pi
pi:x:100:pi
pi@raspberrypi:~ $ cat /etc/group | grep ^pi > /etc/group.pi
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $ cat /etc/group.pi
pi:x:100:pi
pi@raspberrypi:~ $ rm /etc/group.pi
pi@raspberrypi:~ $ ls -l
total 0
pi@raspberrypi:~ $
```

**Ctrl + L**: با اضافه کردن کیس مشخص شده این دستور را اضافه می کنیم. ابتدا کل بافر اینپوت را صفر می کنیم.

بعد از مقداردهی به ایندکس های مربوط به اینپوت تابع `cgaputc` که وظیفه هندل کردن محیط گرافیکی کنسول را دارد با آرگومان مشخص شده کال می کنیم.

```

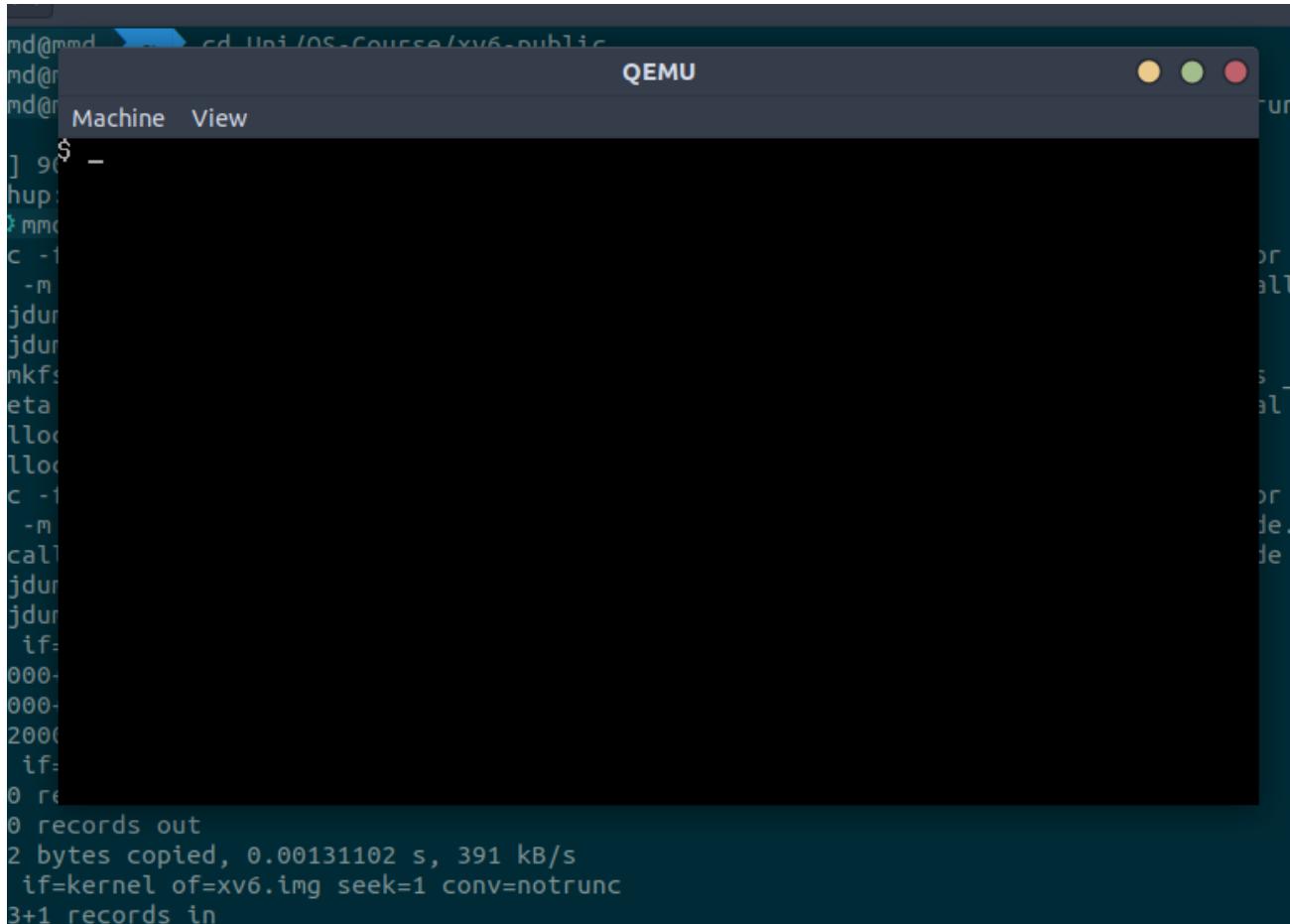
    }
    break;
  case C('L'):
    memset(input.buf, 0, 128);
    input.e = input.w;
    input.c = input.e;
    cgaputc(CONTROL_L);
    break;
  case C('R'):
  
```

در تابع `cgaputc` ابتدا محل `cursor` که متغیر `pos` است را به انتهای خط می بریم. سپس آرایه `crt` را که محتویات صفحه را شامل می شود را به طور کامل `blank` می کنیم. (عبارت `bitwise` مشخص شده برابر `blank` است). بعد از آن محتویات خانه اول و دوم `crt` را برابر کارکترهای گفته شده قرار می دهیم.

```

95  }
96  else if(c == CONTROL_L){      You, 3 days ago • Implement Ctrl+L
97    pos += distance;
98    distance = 0;
99    while(pos > 0)
100      crt[pos--] = ' ' | 0x0700;
101    crt[0] = '$' | 0x0700;
102    crt[1] = ' ' | 0x0700;
103    pos = 2;
104  }
105  else{
  
```

اجرای دستور Ctrl + L



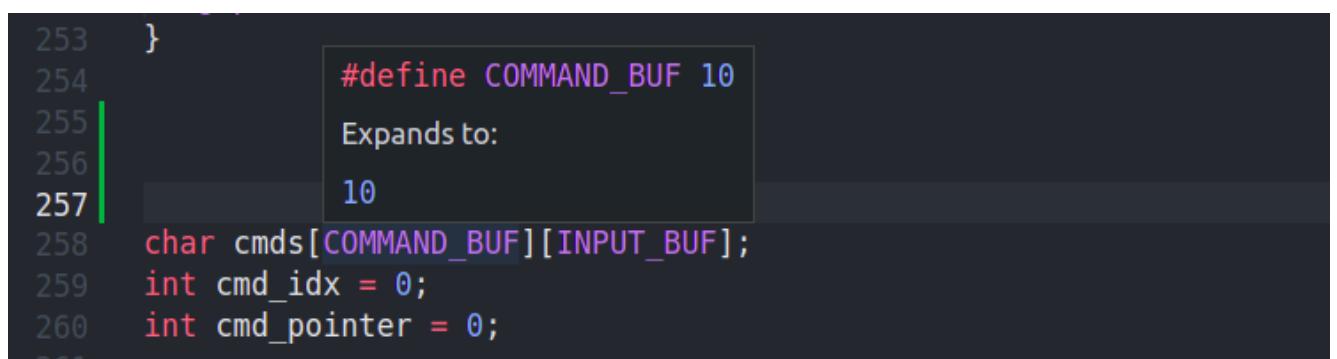
```

md@md: ~ cd Uni/OS-Course/xv6-public
QEMU
Machine View
$ -
hup:
> mmc
c -1
-m
jdu
jdu
mkfs
eta
lloc
lloc
c -1
-m
call
jdu
jdu
if=
000-
000-
2000
if=
0 re
0 records out
2 bytes copied, 0.00131102 s, 391 kB/s
if=kernel of=xv6.img seek=1 conv=notrunc
3+1 records in

```

: ابتدا متغیر های مورد نیاز برای ذخیره کردن کامندهای قبلی و نمایش آنها براساس تعداد دکمه

های زده شده به صورت گلوبال تعریف می کنیم.



```

253 }
254
255
256
257 #define COMMAND_BUF 10
258     char cmds[COMMAND_BUF][INPUT_BUF];
259     int cmd_idx = 0;
260     int cmd_pointer = 0;

```

#define COMMAND\_BUF 10  
Expands to:  
10

سپس در حالت default سوییچ تغییرات لازم برای ذخیره سازی کامند زده شده با اینتر را پیاده می‌کنیم. به این صورت که اگر کامندهای زده شده به 10 نرسیده باشند صرفاً به خانه بعدی اضافه شوند و ایندکس آخرین کامند هم اضافه شود. اما اگر کامندهای قبلی بیشتر از 10 بوده باشند باید همه کامندها یک عدد شیفت بخورند تا خانه ای برای کامند جدید خالی شود. برای شیفت دادن باید همه خانه‌های آرایه را خالی کرد که هیچ کاراکتری از کامند قبلی در آن باقی نماند. سپس کامند جدید را به آخرین خانه اضافه می‌کنیم. همچنانی پس از 10 کامند دیگر cmd\_idx تغییر نمی‌کند و همواره خانه آخر را نشان می‌دهد. همچنانی هر موقع که کامندی وارد شد، cmd\_pointer را برابر آخرین دستور وارد شده یعنی cmd\_idx قرار می‌دهیم.

```

367     consputc(c);
368     if(c == '\n' || c == C('D') || input.e == input.r+INPUT_BUF){
369         if(c == '\n'){
370             if(cmd_idx == COMMAND_BUF){ // clear all elements before shift all commands up
371                 for(i = 0; i < COMMAND_BUF - 1; i++){
372                     for(n = 0; n < INPUT_BUF; n++)
373                         cmd[cmd_idx][n] = '\0';
374                     for(j = 0; cmd[cmd_idx + 1][j] != '\0'; j++)
375                         cmd[cmd_idx + 1][j] = cmd[i][j];
376                 }
377                 for(n = 0; n < INPUT_BUF; n++)
378                     cmd[COMMAND_BUF - 1][n] = '\0';
379                 cmd_idx--;
380             }
381
382             for(i = input.w; i < input.e - 1; i++) // insert new command
383                 cmd[cmd_idx][i - input.w] = input.buf[i % INPUT_BUF];
384
385             cmd_idx = (cmd_idx == COMMAND_BUF)? COMMAND_BUF : cmd_idx + 1;
386             cmd_pointer = cmd_idx;
387         }
388         input.w = input.e;
389         wakeup(&input.r);
390     }
391     // input.end++;
392 }
393 break;

```

در سوییچ کیس هم کدهای ASCII مطابق up و down arrow را قرار می دهیم. کد های این قسمت تقریباً یکسان هستند. در هر دو کل کارکتر های خط پاک می شوند و دستور موجود در ایندکس cmd\_pointer در بافر نوشته می شود و ایندکس های input.e و input.c متناسب با آنها تغییر می کنند. صرفاً در up arrow اگر متغیر cmd\_pointer به اول آرایه دستورات نرسیده بود و در down arrow هم اگر به آخرین دستور وارد شده نرسیده بود تغییرات اعمال می شوند.

```

314     break;
315     case 0xE2: // UP key
316         if(cmd_pointer > 0){
317             cmd_pointer--;
318             while(input.e != input.w && input.buf[(input.e-1) % INPUT_BUF] != '\n'){
319                 input.e--;
320                 input.c--;
321                 consputc(BACKSPACE);
322             }
323             input.w = input.e;
324             for(i = 0; cmd[cmd_pointer][i] != '\0'; i++){
325                 input.buf[(input.e) % INPUT_BUF] = cmd[cmd_pointer][i];
326                 consputc(cmd[cmd_pointer][i] & 0xff);
327                 input.e++;
328                 input.c++;
329             }
330         }
331     break;
332     case 0xE3: // DOWN key
333         if(cmd_pointer != cmd_idx){
334             cmd_pointer++;
335             while(input.e != input.w && input.buf[(input.e-1) % INPUT_BUF] != '\n'){
336                 input.e--;
337                 input.c++;
338                 consputc(BACKSPACE);
339             }
340             input.w = input.e;
341             for(i = 0; cmd[cmd_pointer][i] != '\0'; i++){
342                 input.buf[(input.e) % INPUT_BUF] = cmd[cmd_pointer][i];
343                 consputc(cmd[cmd_pointer][i] & 0xff);
344                 input.e++;
345                 input.c++;
346             }
347         }
348     break;
349 }
```

:arrow up جرای

```

@mmdd ~ cd Uni/OS-Course/xv6-public
QEMU
Machine View
SeaBIOS (version 1.13.0-1ubuntu1.1)

IPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CB00+1FECCB00 CA00

Booting from Hard Disk...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
-Group #3:
1. Mohammad Reza Nemati
2. Aryan Bastani
3. Mahdiyar Harandi
$ echo abc
abc
$ echo bcd
bcd
$ echo abc_
0(
f:
r:
records out

```

:arrow down جرای

```

@mmdd ~ cd Uni/OS-Course/xv6-public
QEMU
Machine View
SeaBIOS (version 1.13.0-1ubuntu1.1)

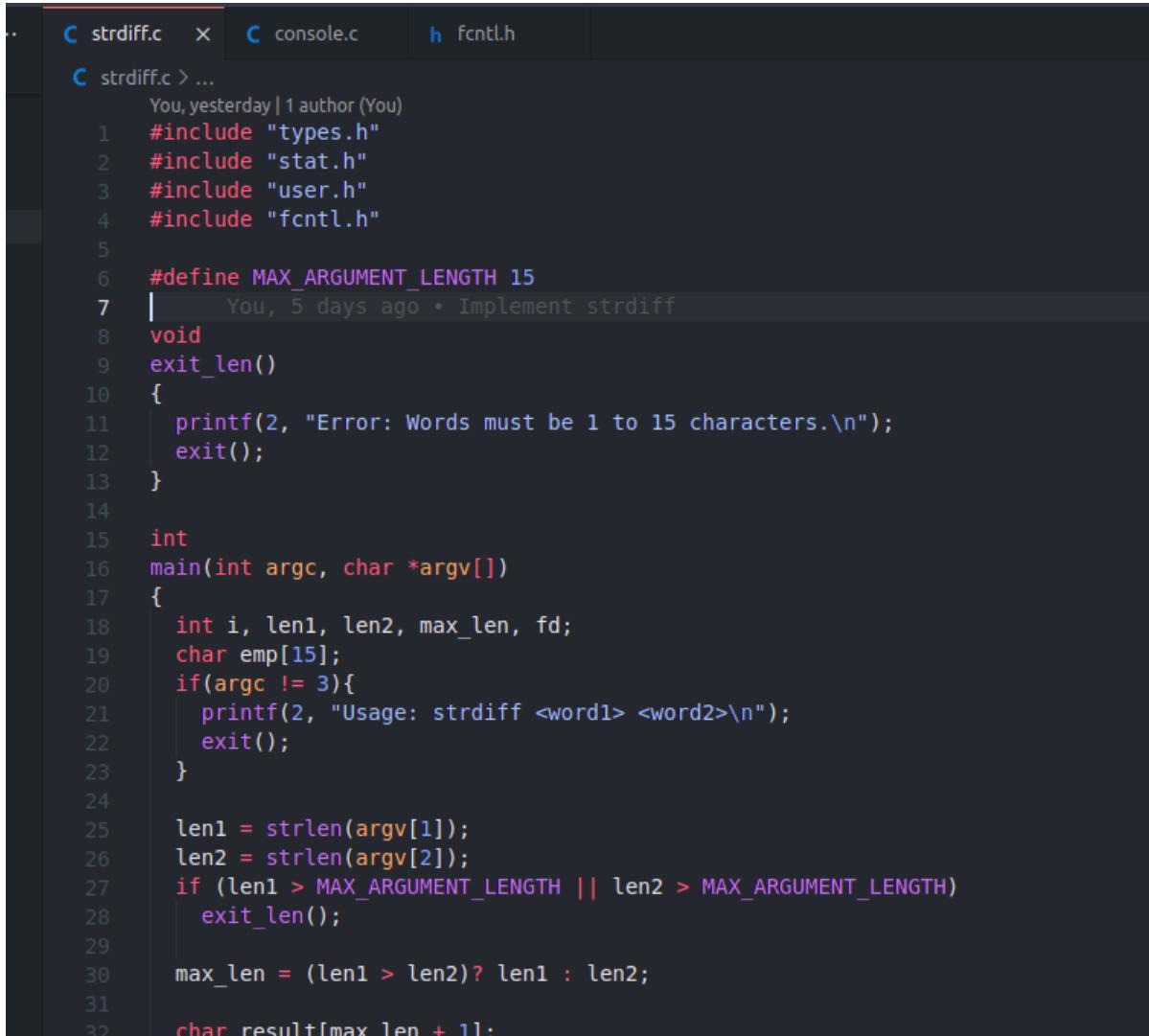
IPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CB00+1FECCB00 CA00

Booting from Hard Disk...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
-Group #3:
1. Mohammad Reza Nemati
2. Aryan Bastani
3. Mahdiyar Harandi
$ echo abc
abc
$ echo bcd
bcd
$ echo abc_
0(
f:
r:
records out
2 bytes copied, 0.00131102 s, 391 kB/s

```

## برنامه سطح کاربر :strcmp

برنامه گفته شده با منطق گفته شده به همراه ارور هندلینگ های مختلف در صورت پروژه پیاده سازی کرده ایم. در آخر جواب بدست آمده را در strdiff\_result.txt ذخیره می کنیم. فایل را با مود O\_CREAT | O\_RDWR باز می کنیم که در صورت موجود نبود فایل از قبل آن را ایجاد کند.



```

strdiff.c  X  console.c  fcntl.h
C strdiff.c > ...
You, yesterday | 1 author (You)
1 #include "types.h"
2 #include "stat.h"
3 #include "user.h"
4 #include "fcntl.h"
5
6 #define MAX_ARGUMENT_LENGTH 15
7 | You, 5 days ago • Implement strcmp
8 void
9 exit_len()
10 {
11     printf(2, "Error: Words must be 1 to 15 characters.\n");
12     exit();
13 }
14
15 int
16 main(int argc, char *argv[])
17 {
18     int i, len1, len2, max_len, fd;
19     char emp[15];
20     if(argc != 3){
21         printf(2, "Usage: strdiff <word1> <word2>\n");
22         exit();
23     }
24
25     len1 = strlen(argv[1]);
26     len2 = strlen(argv[2]);
27     if (len1 > MAX_ARGUMENT_LENGTH || len2 > MAX_ARGUMENT_LENGTH)
28         exit_len();
29
30     max_len = (len1 > len2)? len1 : len2;
31
32     char result[max_len + 1];

```

```

30     max_len = (len1 > len2)? len1 : len2;
31
32     char result[max_len + 1];
33     for(i = 0; i < max_len; i++)
34         if(i + 1 > len1)
35             result[i] = '1';
36
37         else if(i + 1 > len2)
38             result[i] = '0';
39
40         else if (argv[1][i] >= argv[2][i])
41             result[i] = '0';
42
43         else if (argv[1][i] < argv[2][i])
44             result[i] = '1';
45
46     if((fd = open("strdiff_result.txt", O_CREATE | O_RDWR)) >= 0){
47         write(fd, result, strlen(result));
48         write(fd, emp, MAX_ARGUMENT_LENGTH - strlen(result));
49         write(fd, (char *)"\n", 1);
50         close(fd);
51     }
52     else
53         printf(2, "Error: create strdiff_result.txt file failed.\n");
54
55     exit();
56 }
```

همچنین باید در Makefile هم تغییراتی را اعمال کنیم که برنامه به صورت یک user program توسط سیستم عامل شناخته شود و همچنین فایل strdiff\_result.txt هم توسط سیستم عامل شناسایی شود.

```

107 UPROGS=\
108     _cat\
109     _echo\
110     _forktest\
111     _grep\
112     _init\
113     _kill\
114     _ln\
115     _ls\
116     _mkdir\
117     _rm\
118     _sh\
119     _stressfs\
120     _usertests\
121     _wc\
122     _zombie\
123     _strdiff\
124
125
126 fs.img: mkfs README strdiff_result.txt $(UPROGS)
127     ./mkfs fs.img README strdiff_result.txt $(UPROGS)
128
129     -include *.d
130
131 ## Rename it to rev-01-2012-07-30-01 and check.
132
133 # check in that version.
134
135
136 EXTRA=\
137     mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
138     ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\
139     printf.c umalloc.c strdiff.c      You, 5 days ago • Init project, Add group members to init.c
140     README strdiff_result.txt dot-bochssrc *.pl toc.* runoff runoff.list\
141     .gdbinit.tpl gdbutil\
142
143
144 dist:
145     rm -rf dist
146     cd ..
```

## اجرای برنامه :strdiff

```

d@r:~$ strdiff apple banana
d@r:~$ cat strdiff_result.txt
100011
mm: 1111100
d@r:~$ cat strdiff_result.txt
d@r:~$ records out
bytes copied 0.00131102 s 391 kB/s

```

## سوالات:

1. سیستم عامل xv6 یک باز طراحی مدرن از Unix version 6 است که تقریباً معماری و ساختار آن را دارد ولی به زبان C ANSI و برای x86 multiprocessor های اینتل نوشته شده است. این موارد در داکیومنت سیستم عامل توضیح داده شده است.

2. هر پراسس دارای user-space memory (که شامل دستورها، دیتاهای و استک است) و یک state برای هر پراسس است که این state به طور خصوصی در اختیار کرنل قرار دارد. Xv6 برای time-share کردن پراسس ها، در هر زمانی که CPU یک پراسس را اجرا نمی کند، بین پراسس های مختلف سوییچ می کند. در واقع رجیسترها مورد نیاز آن را ذخیره می کند، به سراغ پراسس بعدی می رود و هر موقع که به این پراسس برگشت، رجیسترها آن را بازیابی می کند و به ادامه اجرای آن می پردازد.

4. سیستم کال fork یک پراسس جدید به اسم child process می سازد که محتوای مموری کاملاً یکسانی با parent process خود دارد. سیستم کال exec به جای مموری پراسسی که از آن صدا زده شده است را memory image لود شده از یک فایل در فایل سیستم قرار می دهد و آن را اجرا می کند.

8. متغیر UPROGS لیست user program های تعریف شده در سیستم را دارد. برای مثال برای اجرای strdiff به عنوان برنامه سطح کاربر، آن را به این لیست اضافه کردیم. متغیر LIBLIS میستی از user library های زبان C است که در کامپایل xv6 استفاده می شوند. همچنین در اجرا و کامپایل user program هایی که در UPROGS قرار دارند هم مورد نیاز هستند.

11. فایل های باینری آبجکت xv6 دارای فرمت ELF اند که از سکشن های آن می توان به data. و .bss. اشاره کرد. با اجرای دستور objdump -h bootblock.o می توان نوع فایل باینری و سکشن های ELF را مشاهده کرد. بعد از لود شدن بوت لودر توسط پردازنده در آدرس 0x7c00 کرنل اجرا می شود. بخش اصلی فایل bootblock.o در مقایسه با بقیه object file ها فقط .text است و قادر بخش های دیگر است. Bootblock.o از آدرس خاصی شروع به اجرا شدن می کند؛ در نتیجه در هنگام ساخته شدن از فلگ Ttext 0x7C00- استفاده می شود که آدرس بخش .text. فایل خروجی را مشخص می کند. فلگ e start - نقطه شروع لیل start در تولید فایل bootblock با استفاده از دستور زیر میسر می شود:

```
objcopy -S -O binary -j .text bootblock.o bootblack
```

این دستور محتويات بخش .text. را به صورت raw binary در bootblock می ریزد. نتیجه می گیریم که ELF از فرمت header هم ندارد؛ پس نوع فایل دودویی

مربوط به بوت raw binary است و با سایر فایل های باینری از این جهت که به فرمت ELF نیست تفاوت دارد. دلیل عدم استفاده از فرمت ELF در bootblock اول به شناسایی نشدن آن توسط CPU؛ و دوم به کم کردن حجم فایل در حدود ۵۱۰ بایت بر می‌گردد.

دستور لازم برای تبدیل bootblock به اسembly به شرح زیر خواهد بود:

```
Objdump -D -b binary -m i386 -M addr16,data16 bootblock
```

من توانیم با استفاده از این فلگ آدرس شروع قرار گرفتن اسembly خروجی در حافظه را به نوعی تغییر دهیم که شبیه به واقعیت (`-adjust-vma=0x7C00`) شروع شود:

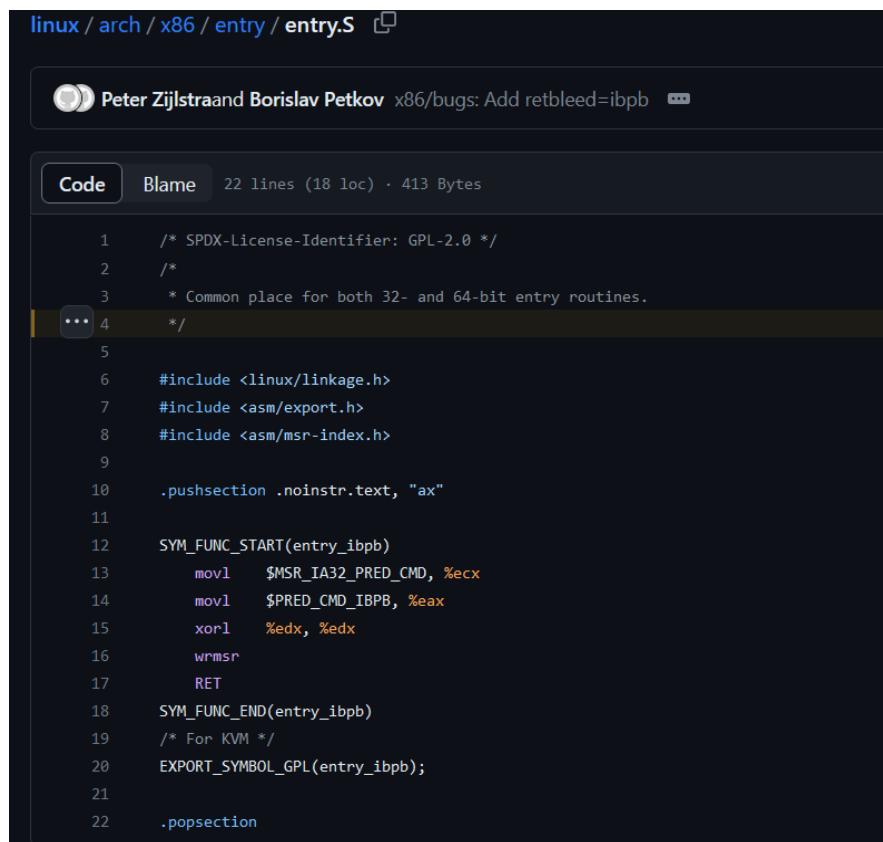
```
bootblock:      file format binary

Disassembly of section .data:
000007c00 <.data>:
7c00:    fa          cli
7c01:    31 c0        xor  %ax,%ax
7c03:    8e d8        mov  %ax,%ds
7c05:    8e c0        mov  %ax,%es
7c07:    8e d0        mov  %ax,%ss
7c09:    e4 64        in   $0x64,%al
7c0b:    a8 02        test $0x2,%al
7c0d:    75 fa        jne  0x7c09
7c0f:    b0 d1        mov  $0xd1,%al
7c11:    e6 64        out  %al,$0x64
7c13:    e4 64        in   $0x64,%al
7c15:    a8 02        test $0x2,%al
7c17:    75 fa        jne  0x7c13
7c19:    b0 df        mov  $0xdf,%al
7c1b:    e6 60        out  %al,$0x60
7c1d:    0f 01 16 78 7c lgdtw 0x7c78
7c22:    0f 20 c0        mov  %cr0,%eax
7c25:    66 83 c8 01    or   $0x1,%eax
7c29:    0f 22 c0        mov  %eax,%cr0
7c2c:    ea 31 7c 08 00 ljmp $0x8,$0x7c31
7c31:    66 b8 10 00 8e d8 mov  $0xd88e0010,%eax
7c37:    8e c0        mov  %ax,%es
7c39:    8e d0        mov  %ax,%ss
7c3b:    66 b8 00 00 8e e0 mov  $0xe08e0000,%eax
7c41:    8e e8        mov  %ax,%gs
7c43:    bc 00 7c        mov  $0x7c00,%sp
7c46:    00 00        add  %al,(%bx,%si)
7c48:    e8 f0 00        call 0x7d3b
```

12. با استفاده از این دستور می‌توان محتویات یک فایل object که با فلگ `-o`- ایجاد می‌شود را درون یک فایل object دیگر کپی کرد. همچنین از آن برای ایجاد raw binary file از خروجی فایل های object هم استفاده می‌شود.

- رجیستر عام منظوره ESP: برای ذخیره کردن top استک فریم کنونی
- رجیستر قطعه CS: ذخیره segment code به memory address pointer کنونی
- رجیستر کنترلی CR3: ذخیره آدرس page table پراسس در حال اجرای کنونی
- رجیستر وضعیت SIE: حاوی بیتها این برای فعال یا غیرفعال کردن برخی interrupt های مشخص

18. با جستجو در [ربازبینی گیت‌هاب لینوکس](#) می‌توانیم فایل entry.S را بیابیم. البته این فایل با توجه به کامنت ابتدای آن، فقط بخش مشترک entry routine سیستم 32 بیتی و 64 بیتی را حاوی هست و در همان لینک بالا می‌توان entry های کامل این دو سیستم را مشاهده کرد که از هم جدا هستند.



The screenshot shows a GitHub commit page for the file `entry.S` in the `linux/arch/x86/entry` directory. The commit is authored by [Peter Zijlstra and Borislav Petkov](#) and is titled "x86/bugs: Add retbleed=ibpb". The commit message includes the following text:

```

/*
 * Common place for both 32- and 64-bit entry routines.
 */
#include <linux/linkage.h>
#include <asm/export.h>
#include <asm/msr-index.h>

.pushsection .noinstr.text, "ax"
SYM_FUNC_START(entry_ibpb)
    movl    $MSR_IA32_PRED_CMD, %ecx
    movl    $PRED_CMD_IBPB, %eax
    xorl    %edx, %edx
    wrmsr
    RET
SYM_FUNC_END(entry_ibpb)
/* For KVM */
EXPORT_SYMBOL_GPL(entry_ibpb);
.popsection

```

The commit has 22 lines (18 loc) and 413 Bytes.

19. در واقع ما برای ترجمه آدرس های مجازی به آدرس های فیزیکی به این table نیاز داریم. اگر آدرس خود این table هم به صورت مجازی باشد، دوباره باید آن را به آدرس فیزیکی تبدیل کنیم که نیاز همین table می شود و عملاً امکان پذیر نیست. به همین دلیل آدرس فیزیکی page table را ذخیره می کنیم.

22. تمام قطعه های هسته و کاربر یک بخش از حافظه را در اختیار دارند. هر یک از این قطعه ها، با یک دسکریپتور در GDT مشخص شده است که این دیسکریپتور شامل اطلاعاتی مانند آدرس شروع قطعه، اندازه قطعه و سطح دسترسی قطعه می باشد. برای خواندن یک دستور، ابتدا قطعه آن از طریق دسکریپتورش یافت می شود؛ سپس صفحه مربوط به آن پیدا می شود و بعد از تبدیل آدرس منطقی به فیزیکی، دستور از حافظه خوانده شده و اجرا می شود. سطح دسترسی مورد نیاز یک دستور از روی سطح دسترسی دسکریپتور یا همان DPL مشخص می شود -به گونه ای که از طریق دسکریپتورهای متفاوت، سطح دسترسی فعلی دستورهای تعیین می شود- حتی اگر آن ها قطعات یکسانی از حافظه را تعیین کنند. بنابراین، با وجود اینکه هر دو بخش کاربر و هسته به قطعات یکسانی دسترسی دارند، اما سطح دسترسی شان متفاوت است و کاربر نمی تواند هر دستورالعملی را اجرا کند.

```

enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };

// Per-process state
▼ struct proc {
    uint sz;                                // size of process memory (bytes)
    pde_t* pgdir;                            // Page table
    char *kstack;                            // Bottom of kernel stack for this process
    enum procstate state;                  // Process state
    int pid;                                 // Process ID
    struct proc *parent;                   // Parent process
    struct trapframe *tf;                  // Trap frame for current syscall
    struct context *context;                // swtch() here to run process
    void *chan;                             // If non-zero, sleeping on chan
    int killed;                            // If non-zero, have been killed
    struct file *ofile[NFILE];             // Open files
    struct inode *cwd;                     // Current directory
    char name[16];                          // Process name (debugging)
};


```

متغیر `sz`: حجم یا همان تعداد بایت‌های مموری پراسس را ذخیره می‌کند.

متغیر `pgdir`: یک پوینتر به `page table` مورد نیاز پراسس برای ترجمه آدرس مجازی است.

متغیر `kstack`: یک پوینتر از تایپ `*char` است که از آن برای ران کردن `system call` استفاده می‌شود.

متغیر `state`: استیت کنونی پراسس را ذخیره می‌کند که تایپ آن `enum procstate` است که در تصویر مشخص است.

متغیر `pid`: برای ذخیره سازی `process ID` ای که کرنل به این پراسس اختصاص داده است.

متغیر `parent`: پوینتر از همین تایپ استراکت به پراسس پدر ذخیره می‌کند.

متغیر `tf`: پوینتر از تایپ `*struct trapframe` که برای استیت برنامه را در حین ران شدن `system call` ذخیره می‌کند.

متغیر context: پوینتر از تایپ struct context که دیتاها رجیسترهاي برنامه را بخاطر عملیات ذخیره می کند.

متغیر chan: در صورتی که مقدار آن صفر نباشد، نشان می دهد که پراسس برای انجام کاری در حال wait کردن است. chan مخفف channel است.

متغیر killed: در صورتی که مقدار آن صفر نباشد، نشان می دهد که پراسس توسط سیستم عامل kill شده است.

متغیر ofile: آرایه ای با سایز 16 (NOFILE) که پوینترهایی که به فایل های open شده توسط پراسس اشاره می کنند را ذخیره می کند.

متغیر cwd: مخفف Current Working Directory است و آن را ذخیره می کند.

متغیر name: آرایه ای با سایز 16 از کارکترها برای ذخیره اسم پراسس که صرفا برای debugging است. استراکت مشابه آن در لینوکس task\_struct است که در این [لینک](#) قابل مشاهده است و همانطور که مشخص است، فیلد های بسیار بیشتری نسبت به proc در xv6 دارد.

27. بخش های از آماده سازی سیستم که در تمامی هسته های پردازنده مشترک هستند مشتمل از Switchkvm، seginit، lapiinet، mpmain کمک تابع switchkvm آدرس page table را که توسط آدرس پردازنده اول ایجاد شده است را در رجیستر خود ذخیره می کنند. و دیگر اینکه تمام پردازنده ها توسط تابع mpmain آماده اجرای برنامه ها می شوند؛ به همین دلیل است که این توابع باید بین تمام پردازنده ها مشترک باشند.

توابع switchkvm و mpmain از بخش های مشترک هسته های پردازنده هستند؛ همه پردازنده ها به کمک تابع switchkvm آدرس page table را که توسط آدرس پردازنده اول ایجاد شده است را در رجیستر خود ذخیره می کنند. و دیگر اینکه تمام پردازنده ها توسط تابع mpmain آماده اجرای برنامه ها می شوند؛ به همین دلیل است که این توابع باید بین تمام پردازنده ها مشترک باشند.

از بخش های اختصاصی هسته اول هم می توان به تابع `ideinit` اشاره کرد که پردازنده به کمک این تابع `startothers` را شناسایی می کند. همچنین، این پردازنده به کمک تابع `startothers` سایر پردازنده ها را می کند؛ در هر دو مورد، دلیل این است که نیازی نیست هر پردازنده این کار را انجام دهد. زمان بند توسط تابع `scheduler` انجام می پذیرد و بین تمامی هسته ها مشترک است؛ چرا که هر پردازنده باید `scheduler` مربوط به خود را داشته باشد.

## اشکال زدایی

1. دستور `'info break'` یا `'info breakpoints'`. شماره بریک پوینت را از دستور های بالا می توان یافت.
2. دستور `'del <number of breakpoint>'`. به طور کلی نشان می دهد که برنامه چگونه به این نقطه رسیده است. در واقع `call` را نمایش می دهد. در خط اول تابع کنونی و در هر کدام از خط های بعدی `caller` مورد قبلی را چاپ می کند. `'bt'` یک alias برای دستور `'backtrace'` است.
3. دستور `'print'` مقدار متغیر ورودی را نشان می دهد ولی دستور `'x'` محتوای آدرس مموری ورودی را نشان می دهد. دستور `'print'` به طور پیش فرض در نظر می گیرد که مقدار ورودی یک `machine word` است. اما دستور `'x'` آخرین اندازه ای که با آن کار شده است را ذخیره می کند و در ادامه هم از همان استفاده می کند.
- برای نشان داده محتوای یک رجیستر از دستور `'r <register name>'` یا `'info register <register name>'` استفاده می کنیم.

5. برای نمایش وضعیت همه رجیسترها از دستور `'r'` یا `'info registers'` استفاده من کنیم. برای

نمایش متغیرهای لوکال دستور `locals` کاربرد دارد.

```
(gdb) list
256
257     #define C(x) ((x)-'@') // Control-x
258
259     void
260     consoleintr(int (*getc)(void))
261     {
262         int c, doprocdump = 0;
263         int i, n, j;
264         acquire(&cons.lock);
265         while((c = getc()) >= 0){
(gdb) info locals
c = <optimized out>
doprocdump = <optimized out>
i = <optimized out>
n = <optimized out>
j = <optimized out>
(gdb) █
```

رجیستر edi یا Destination Index معمولاً در عملیات های استرینگ یا عملیات هایی که دیتا در آنها

کپ یا جا به جا من شود کاربرد دارد. این رجیستر محل اینکه دیتا کجا ذخیره شود را نگه من دارد.

رجیستر esi یا Source Index هم در همان عملیات های edi کاربرد دارد. محل اینکه دیتای مورد نیاز از

کجا دریافت شود را نگه من دارد.

6. اینپوت یک استراکت برای نگهداری وضعیت دستور های تایپ شده در کنسول است.

فیلد buf خود کاراکترهای وارد شده را نگه من دارد. فیلد w اندیس ابتدای دستور تایپ

شده، فیلد e اندیسی که در این زمان با وارد کردن ورودی در آنجا ادیت انجام من شود و

فیلد r اندیس ابتدای خط است که از آنجا دستور خوانده من شود اما دیرتر از w اپدیت

من شود تا هنگام اجرای دستور از اندیس r تا w بررسی شود.

- با زدن ایتر و وارد شدن به خط بعدی برای وارد کردن دستور بعدی، مقدار فیلدهای w و

r اضافه من شوند. با هر کارکتر جدید مثل backspace یا در کل هر تغییری در ورودی،

فیلد e کم یا زیاد می شود. خود کارکترهای موجود در خط دستور در buf ذخیره می شود.

7. دستور `layout asm` سورس زبان C نقطه توقف را نشان می دهد. دستور `layout src` سورس assembly نقطه توقف را نشان می دهد که کامپایلر از کدهای C تولید کرده است.
8. ابتدا با دستور `bt` تمام function stack call تا جایی که به نقطه توقف رسیدیم را می بینیم. سپس با دستور `frame <number of frame to inspect>` می توان بین توابع استک جابه جا شد.

## پیکربندی و ساختن هسته لینوکس (بخش امتیازی)

بخشی از فایل config به همراه ادیت هایی برای نصب در Virtualbox:

```
# CONFIG_KERNEL_LZO is not set
# CONFIG_KERNEL_LZ4 is not set
CONFIG_KERNEL_ZSTD=y
CONFIG_DEFAULT_INIT=""
CONFIG_DEFAULT_HOSTNAME="(none)"
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ head .config
#
# Automatically generated file; DO NOT EDIT.
# Linux/x86 6.1.59 Kernel Configuration
#
CONFIG_CC_VERSION_TEXT="gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0"
CONFIG_CC_IS_GCC=y
CONFIG_GCC_VERSION=90400
CONFIG_CLANG_VERSION=0
CONFIG_AS_IS_GNU=y
CONFIG_AS_VERSION=23400
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ scripts/config --disable SYSTEM_TRUSTED_KEYS
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ scripts/config --disable SYSTEM_REVOCATION_KEYS
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ scripts/config --set-str CONFIG_SYSTEM_TRUSTED_KEYS ""
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ scripts/config --set-str CONFIG_SYSTEM_REVOCATION_KEYS ""
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ 
```

نتیجه اجرای دستور fakeroot make -j6

در صورت استفاده نکرده از fakeroot، پراسس ممکن است بدون دادن اروری به اتمام برسد و متوجه خطاهای نشویم. (البته در این بخش به دلیل نصب نشدن درست برخی پکیج‌ها نیاز شد چندین بار فرایند make کردن متوقف شود که در زیر فقط تصویر نهایی آورده شده.)

```
LD [M] 1s/voodoo/voodoo.ko
LD [M] kernel/configs.ko
LD [M] lib/reed_solomon/reed_solomon.ko
LD [M] net/ipv4/netfilter/ip_tables.ko
LD [M] net/netfilter/x_tables.ko
LD [M] net/netfilter/xt_tcpudp.ko
LD [M] net/sched/sch_fq_codel.ko
LD [M] sound/ac97_bus.ko
LD [M] sound/core/seq/snd-seq.ko
LD [M] sound/core/snd-pcm.ko
LD [M] sound/core/snd-seq-device.ko
LD [M] sound/core/snd-timer.ko
LD [M] sound/core/snd.ko
LD [M] sound/pci/ac97/snd-ac97-codec.ko
LD [M] sound/soundcore.ko
LD [M] sound/pci/snd-intel8x0.ko
MKPIGGY arch/x86/boot/compressed/piggy.S
AS      arch/x86/boot/compressed/piggy.o
LD      arch/x86/boot/compressed/vmlinux
ZOFFSET arch/x86/boot/zoffset.h
OBJCOPY arch/x86/boot/vmlinux.bin
AS      arch/x86/boot/header.o
LD      arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready (#4)
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ sudo make modules_install -j8
```

اجرای دستور sudo make modules\_install -j8

```

SIGN  /lib/modules/6.1.59/kernel/sound/core/seq/snd-seq.ko
SIGN  /lib/modules/6.1.59/kernel/sound/ac97_bus.ko
INSTALL /lib/modules/6.1.59/kernel/sound/core/snd-pcm.ko
INSTALL /lib/modules/6.1.59/kernel/sound/core/snd-seq-device.ko
INSTALL /lib/modules/6.1.59/kernel/sound/core/snd-timer.ko
INSTALL /lib/modules/6.1.59/kernel/sound/core/snd_ko
INSTALL /lib/modules/6.1.59/kernel/sound/pci/ac97/snd-ac97-codec.ko
INSTALL /lib/modules/6.1.59/kernel/sound/pci/snd-intel8x0.ko
SIGN  /lib/modules/6.1.59/kernel/sound/core/snd-seq-device.ko
INSTALL /lib/modules/6.1.59/kernel/sound/soundcore.ko
SIGN  /lib/modules/6.1.59/kernel/sound/core/snd-timer.ko
SIGN  /lib/modules/6.1.59/kernel/sound/core/snd_ko
SIGN  /lib/modules/6.1.59/kernel/sound/core/snd-pcm.ko
SIGN  /lib/modules/6.1.59/kernel/sound/pci/ac97/snd-ac97-codec.ko
SIGN  /lib/modules/6.1.59/kernel/sound/pci/snd-intel8x0.ko
SIGN  /lib/modules/6.1.59/kernel/sound/soundcore.ko
DEPMOD /lib/modules/6.1.59
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ []

```

اجرای دستور sudo make install برای نصب نهایی کرنل:

```

- Original module
- No original module exists within this kernel
- Installation
- Installing to /lib/modules/6.1.59/updates/dkms/

vboxnetflt.ko:
Running module version sanity check.
- Original module
- No original module exists within this kernel
- Installation
- Installing to /lib/modules/6.1.59/updates/dkms/

depmod...

DKMS: install completed.

run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.1.59 /boot/vmlinuz-6.1.59
update-initramfs: Generating /boot/initrd.img-6.1.59
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 6.1.59 /boot/vmlinuz-6.1.59
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.1.59 /boot/vmlinuz-6.1.59
run-parts: executing /etc/kernel/postinst.d/vboxadd 6.1.59 /boot/vmlinuz-6.1.59
VirtualBox Guest Additions: Building the modules for kernel 6.1.59.

VirtualBox Guest Additions: Look at /var/log/vboxadd-setup.log to find out what
went wrong
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 6.1.59 /boot/vmlinuz-6.1.59
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-86-generic
I: /boot/initrd.img is now a symlink to initrd.img-6.1.59
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.1.59 /boot/vmlinuz-6.1.59
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.59
Found initrd image: /boot/initrd.img-6.1.59
Found linux image: /boot/vmlinuz-5.15.0-86-generic
Found initrd image: /boot/initrd.img-5.15.0-86-generic
Found linux image: /boot/vmlinuz-5.15.0-73-generic
Found initrd image: /boot/initrd.img-5.15.0-73-generic
Found linux image: /boot/vmlinuz-5.13.0-30-generic
Found initrd image: /boot/initrd.img-5.13.0-30-generic
Found linux image: /boot/vmlinuz-5.11.0-41-generic
Found initrd image: /boot/initrd.img-5.11.0-41-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ echo $?
0
mmd@mmd-VirtualBox:~/ker/linux-6.1.59$ []

```

اثبات نصب کامل کرنل جدید:

```
mmd@mmd-VirtualBox:~$ uname -rs
Linux 6.1.59
mmd@mmd-VirtualBox:~$ 
```

همچنین با دستورات و کدهای زیر، اسم اعضای گروه در دستور dmesg پرینت می شود.

EXPLORER

- OS-COURSE
  - > xv6-public
  - .Module.symvers.cmd
  - .modules.order.cmd
  - .OSLab1\_bonus.ko.cmd
  - .OSLab1\_bonus.mod.cmd
  - .OSLab1\_bonus.mod.o.cmd
  - .OSLab1\_bonus.o.cmd
  - Makefile**
  - Module.symvers
  - modules.order
  - OSLab1\_bonus.c** 2
  - OSLab1\_bonus.ko
  - OSLab1\_bonus.mod
  - C OSLab1\_bonus.mod.c
  - OSLab1\_bonus.mod.o
  - OSLab1\_bonus.o

Makefile

```
1 obj-m += OSLab1_bonus.o
2
3 all:
4 | make -C /lib/modules/6.1.59/build M=$(PWD) modules
```

OSLab1\_bonus.c 2

```
#include <linux/module.h>
#include <linux/kernel.h>
MODULE_LICENSE();
int init_module(void)
{
    printk(KERN_INFO "Group #3:\n Mohammad Reza Nemati\nMahdiar Harandi\nAryan Bastani\n");
    return 0;
}
void cleanup_module(void)
```

mmd@mmd-VirtualBox:~/uni/OS-Course\$ make
make -C /lib/modules/6.1.59/build M=/home/mmd/uni/OS-Course modules
make[1]: Entering directory '/home/mmd/ker/linux-6.1.59'
make[1]: Leaving directory '/home/mmd/ker/linux-6.1.59'
mmd@mmd-VirtualBox:~/uni/OS-Course\$ sudo insmod OSLab1\_bonus.ko
insmod: ERROR: could not insert module OSLab1\_bonus.ko: File exists
mmd@mmd-VirtualBox:~/uni/OS-Course\$ dmesg | tail
[ 54.389434] audit: type=1400 audit(1697836631.292:46): apparmor="DENIED" operation="capable" profile="/snap/snapd/20290/usr/lib/snapd/snap-confine" pid=2148 comm="snap-confine" capability=4 capname="fsetid"
[ 58.694594] audit: type=1400 audit(1697836635.600:47): apparmor="DENIED" operation="open" profile="snap.snap-store.ubuntu-software" name="/etc/appstream.conf" pid=2148 comm="snap-store" requested\_mask="r" denied\_mask="r" fsuid=1000 ouid=0
[ 170.128024] [drm:vmw\_msg\_ioctl [vmwgfx]] \*ERROR\* Failed to open channel.
[ 170.128049] [drm:vmw\_msg\_ioctl [vmwgfx]] \*ERROR\* Failed to open channel.
[ 419.798656] OSLab1\_bonus: module license '' taints kernel.
[ 419.798661] Disabling lock debugging due to kernel taint
[ 419.798994] Group #3:
 Mohammad Reza Nemati
 Mahdiar Harandi
 Aryan Bastani

منبع نصب کرنل:

<https://davidaugustat.com/linux/how-to-compile-linux-kernel-on-ubuntu>