



به نام خدا



دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس آزمایشگاه سیستم عامل

پروژه اول

دانیال سعیدی(810198571)
سروش صادقیان(810898048)
محمد قره حسنلو(810198461)

نام و نام خانوادگی

یکشنبه - ۲۲ اسفند ۱۴۰۰

تاریخ ارسال گزارش

<https://github.com/daniel-saeedi/OS-Lab>

ریپو گیتهاب

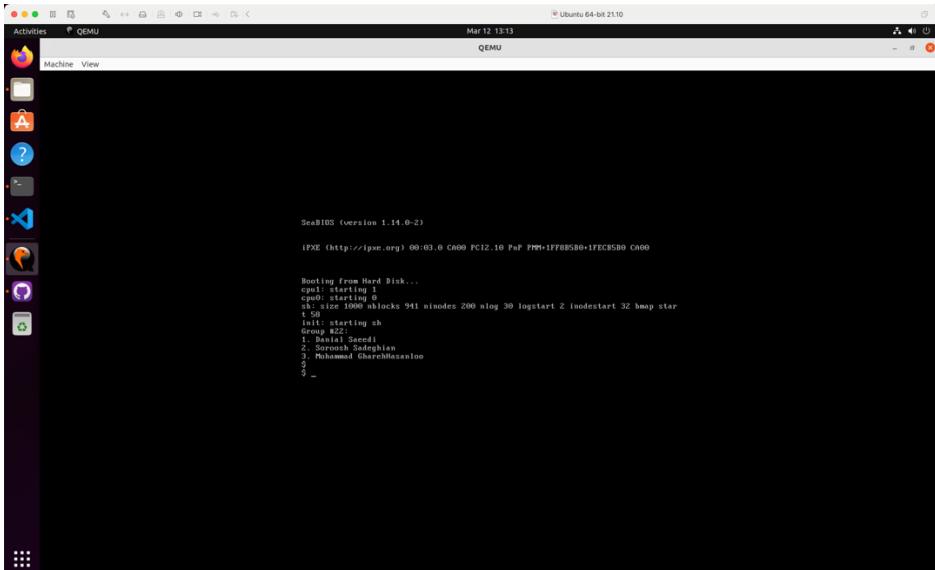
17915a566cea4b274c62582403503a9cada508a7

آخرین Commit ID

فهرست گزارش سوالات

ب	چاپ نام اعضای گروه
ب	Cursor & History
ت	اجرا و پیاده سازی برنامه سطح کاربر
4	سوال ۱
4	سوال ۲
4	سوال ۴
5	سوال ۸
5	سوال ۱۱
5	سوال ۱۲
5	سوال ۱۴
6	سوال ۱۸
7	سوال ۱۸
7	سوال ۲۳
7	سوال ۲۷
8	اشکال زدایی ۱
8	اشکال زدایی ۲
8	اشکال زدایی ۳
9	اشکال زدایی ۴
9	اشکال زدایی ۵
10	اشکال زدایی ۶
11	اشکال زدایی ۷
13	اشکال زدایی ۸

چاپ نام اعضای گروه

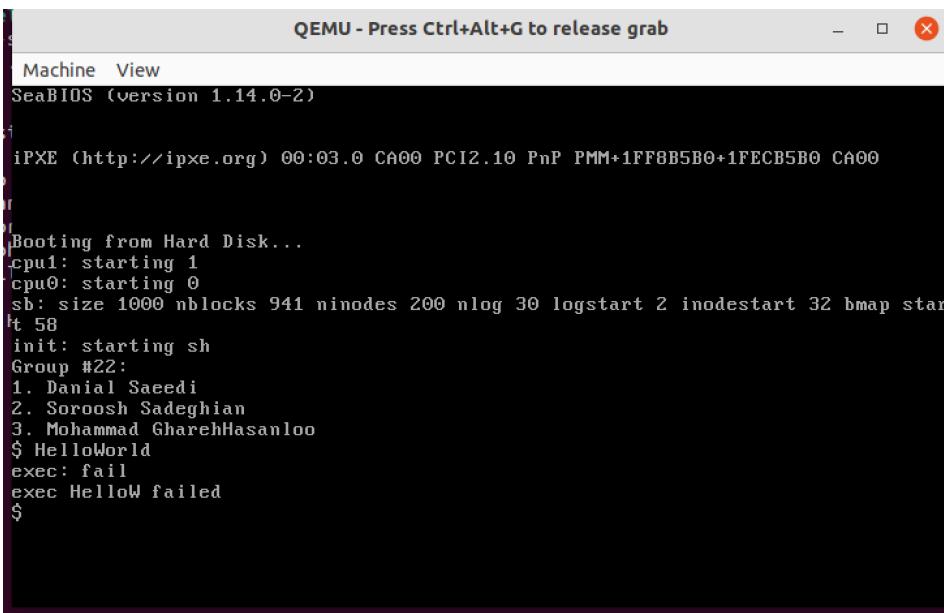


```
SeaBIOS (version 1.14.0-2)
IPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8B5B0+1FECB5B0 CA00

Booting from Hard Disk...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
Group #22:
1. Danial Saeedi
2. Soroosh Sadeghian
3. Mohammad GharehHasanloo
$
```

Cursor & History

پیاده سازی این قسمت در فایل `console.c` انجام شده است.

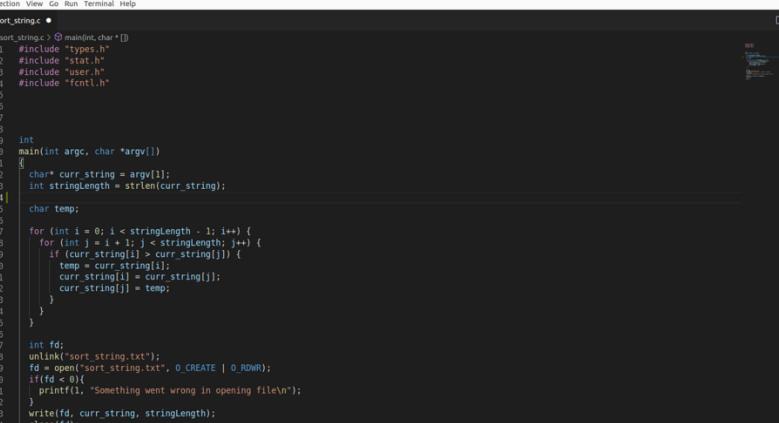


```
QEMU - Press Ctrl+Alt+G to release grab
Machine View
SeaBIOS (version 1.14.0-2)

IPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8B5B0+1FECB5B0 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 #22:
1. Danial Saeedi
2. Soroosh Sadeghian
3. Mohammad GharehHasanloo
$ HelloWorld
exec: fail
exec HelloWorld failed
$
```

اجرا و پیاده سازی برنامه سطح کاربر



A screenshot of the Visual Studio Code interface. The title bar shows "Ubuntu 64-bit 21.10" and the date "Mar 12 13:47". The status bar at the bottom indicates "Ln 14, Col 3" and "Spaces: 2". The code editor displays a C program named "sort_string.c" with the following content:

```
C sort_string.c
C sort_string.c > ① main(int argc[])
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <fcntl.h>
5
6
7
8
9     int
10    main(int argc, char *argv[])
11    {
12        char* curr_string = argv[1];
13        int stringLength = strlen(curr_string);
14
15        char temp;
16
17        for (int i = 0; i < stringLength - 1; i++) {
18            for (int j = i + 1; j < stringLength; j++) {
19                if (curr_string[i] > curr_string[j]) {
20                    temp = curr_string[i];
21                    curr_string[i] = curr_string[j];
22                    curr_string[j] = temp;
23                }
24            }
25        }
26
27        int fd;
28        unlink("sort_string.txt");
29        fd = open("sort_string.txt", O_CREAT | O_RDWR);
30        if (fd == -1) {
31            printf(1, "Something went wrong in opening file\n");
32        }
33        write(fd, curr_string, stringLength);
34        close(fd);
35
36        exit();
37    }
38
```

سوال ۱

سیستم عامل Unix ۷۶ بر اساس Unix ۷ پیاده سازی شده است. سیستم عامل Unix توسط Dennis Ritchie و Ken Thompson از ساختار Unix ۷۶ استفاده میکند ولی با ANSI C برای پردازنده های مبتنی بر x86 پیاده سازی شده است.

سوال ۲

به طور کلی یک پردازه در Unix ۷۶ شامل حافظه که این حافظه خود شامل دستورات، داده و پشته است و یک حالت برای هر پردازه که به صورت خصوصی در اختیار هسته است، میباشد. همچنین هر پردازه با یک شناسه برای هسته سیستم عامل قابل شناسایی است. به طور کلی، سیستم عامل از روش اشتراک گذاری زمان و زمان بندی (sharing-time) بین پردازه ها استفاده میکند، یعنی بطور مداوم و بدون اینکه کاربر متوجه شود (transparency)، پردازنده های موجود را بین پردازه های آماده اجرا شدن پخش میکند و زمانی که یک پردازه در حال اجرا نیست، سیستم عامل مقدار ثباتهای مربوط به پردازنده را ذخیره کرده تا برای اجرا مجدد آماده باشد.

سوال ۴

برای ایجاد یک پروسه از فراخوان سیستمی fork() استفاده میشود. یک child با همان محتوای حافظه ای پروسه parent ایجاد میکند. در پروسه child آیدی پروسه child را برمیگرداند و پروسه child صفر بر میگرداند. فراخواندن سیستمی exec() حافظه process فراخوان کننده را به صورت یک Memory Image جدید جایگزین میکند.

XV6 از exec و fork برای اجرای برنامه های طرف کاربر استفاده میکند. در حلقه اصلی با استفاده از getcmd یک خط از ورودی خوانده میشود. با فراخوانی fork یک کپی از پردازه (Process) ایجاد میگردد. پردازه پدر در حالی که پردازه فرزند در حال اجرا صبر می کند. در پردازه فرزند، runcmd با استفاده از فراخوان exec دستور را اجرا می کند. سپس از این پردازه خارج شده و به پردازه پدر باز میگردد.

سوال ۸

در UPROGS برنامه هایی که کاربر میتواند انها را اجرا کند ذخیره شده است مانند cat که محتوای یک فایل را به نمایش میگذارد یا rm و در ULIB هم اسم فایل هایی که آنها وابسته است نوشته شده است.

سوال ۱۱

دو فایل bootmain.c و bootasm.S که به ترتیب به زبان های اسembly و C هستند توسط کامپایلر کامپایل میشوند و Object File های این دو فایل که فایل اکستنشن آن ها .o هست ساخته میشود که فایل های باینری یا دودویی هستند. تفاوت فایل های boot با سایر فایل ها در قرارگیری first block on first track هستش که همیشه در sector اول است اما برای سایر فایل ها چنین نیست. همچنین کامند زیر فایل را به زبان اسembly تبدیل میکند:

```
Objdump -D bootmain.o
```

سوال ۱۲

این دستور برای کپی کردن یک فایل Object از یک پلتفرم مانند x86 به نوع دیگر پلتفرم مانند ARM استفاده میشود. این ابزار را افرادی استفاده می کنند که میخواهند یک فایل Object روی یک پلتفرم را به پلتفرم دیگر تبدیل کنند بدون آن که به کد های آن دسترسی داشته باشند.

سوال ۱۳

- **ثبتات عام منظوره:** xv6 شامل ۸ ثبات عام منظوره است. این ثبات ها عبارتند از eax, ebx, ecx, edx, edi, esi, ebp, esp Program Counter و یک eip. حرف e در این ثبات ها نشان دهنده extended است چون ۳۲ بیت هستند. این ثبات برای نگه داری بعضی اشاره گر ها، داده ها و برای نگه داری عملیات های ریاضی استفاده می شوند.
- **ثبتات قطعه:** آدرس استک، کد و داده در این ثباتها نگهداری میشود. برای مثال SS پوینتر به استک، CS پوینتر به کد و DS پوینتر به داده را نگه میدارد.
- **ثبتات وضعیت:** شامل اطلاعات راجع به وضعیت پردازنده است. EFLAGS در این بخش محسوب میشود و اطلاعات فلگ هایی نظیر zero, sign, carry و غیره را مشخص میکند.

- ثبات کنترلی: کنترل CPU یا دستگاههای دیجیتال دیگر را در دست دارد. cr4, %cr3, %cr2, %cr0 از این ثباتها هستند. این ثباتها وظیفه تغییر مدل آدرسدهی، کنترل paging و همپردازندگان را 1 دارند.

دستور :info registers

```
(gdb) info registers
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp          0xfffffd110      0xfffffd110
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip          0x10000c      0x10000c
eflags        0x10202      [ IF RF ]
cs           0x23         35
ss           0x2b         43
ds           0x2b         43
es           0x2b         43
fs           0x0          0
gs           0x0          0

(qemu) info registers
EAX=00000000 EBX=00000000 ECX=00000000 EDX=00000663
ESI=00000000 EDI=00000000 EBP=00000000 ESP=00000000
EIP=0000ffff EFL=00000002 [-----] CPL=0 II=0 A20=1 SMM=0 HLT=0
ES =0000 00000000 0000ffff 00009300
CS =f000 ffff0000 0000ffff 00009b00
SS =0000 00000000 0000ffff 00009300
DS =0000 00000000 0000ffff 00009300
FS =0000 00000000 0000ffff 00009300
GS =0000 00000000 0000ffff 00009300
LDT=0000 00000000 0000ffff 00008200
TR =0000 00000000 0000ffff 00008b00
GDT=    00000000 0000ffff
IDT=    00000000 0000ffff
CR0=600000010 CR2=00000000 CR3=00000000 CR4=00000000
DR0=00000000 DR1=00000000 DR2=00000000 DR3=00000000
DR6=ffff0ff0 DR7=00000400
EFER=0000000000000000
FCW=037f FSW=0000 [ST=0] FTW=00 MXCSR=00001f80
FPR0=0000000000000000 0000 FPR1=0000000000000000 0000
FPR2=0000000000000000 0000 FPR3=0000000000000000 0000
FPR4=0000000000000000 0000 FPR5=0000000000000000 0000
FPR6=0000000000000000 0000 FPR7=0000000000000000 0000
XMM00=0000000000000000 0000000000000000 XMM01=0000000000000000 0000000000000000
XMM02=0000000000000000 0000000000000000 XMM03=0000000000000000 0000000000000000
XMM04=0000000000000000 0000000000000000 XMM05=0000000000000000 0000000000000000
XMM06=0000000000000000 0000000000000000 XMM07=0000000000000000 0000000000000000
```

Figure 1 Info Register

سوال ۱۸

معادل S هسته لینوکس در لینک زیر است:

<https://github.com/torvalds/linux/blob/master/arch/arm64/kernel/entry.S>

سوال ۱۸

اگر این بخش را به صورت مجازی در نظر میگرفتیم باز باید یک بخش فیزیکی در نظر میگرفتیم تا این بخش مجازی را مشخص کند، یعنی در نهایت نیاز به بخش فیزیکی بود.

سوال ۲۳

معادل این استراکت در هسته لینکوس در لینک ذیل آمده است:

<https://github.com/torvalds/linux/blob/master/include/linux/sched.h>

- SZ: اندازه حافظه متعلق به پردازه به بایت
- pgdir: پوینتر به table page هست.
- kstack: پایین stack هسته برای این پردازه را مشخص میکند.
- state: وضعیت این پردازه را مشخص میکند.
- pid: عدد اختصاص داده شده به این پردازه را مشخص می کند.
- parent: پدر این پردازه یا به عبارت دیگر سازنده این پردازه را مشخص میکند.
- tf: چارچوب trap برای system call حال حاضر
- context: برای switching context نگهداری شده است.
- chan: اگر صفر نباشد به معنای خوابیدن پردازه است.
- killed: اگر غیر صفر باشد یعنی پردازه kill شده است.
- ofile: فایلهای باز شده توسط این پردازه.
- cwd: پوششی کنونی را مشخص میکند.
- name: نام این پردازه.

سوال ۲۷

اشکال زدایی ۱

با دستور زیر میتوان لیستی از Breakpoint ها را دریافت کرد:

maint info breakpoint

اشکال زدایی ۲

با دستور زیر میتوان یک Breakpoint را حذف کرد: (که line و filename به ترتیب نام فایل و خط breakpoint است)

clear filename:line

اشکال زدایی ۳

دستور (bt) سلسله توابع فراخوان شده و در استک اضافه شده اند را نشان میدهد. به عبارت دیگر این دستور نشان میدهد که برنامه چگونه به جایی که الان قرار دارد رسیده است.

```
(gdb) target remote tcp::26000
Remote debugging using tcp::26000
0x80103e05 in ?? ()
(gdb) b cat.c:12
Note: breakpoint 1 also set at pc 0x97.
Breakpoint 2 at 0x97: file cat.c, line 12.
(gdb) c
Continuing.

Thread 1 hit Breakpoint 1, cat (fd=3) at cat.c:12
12      while((n = read(fd, buf, sizeof(buf))) > 0) {
(gdb) bt
#0  cat (fd=3) at cat.c:12
#1  0x000000054 in main (argc=<optimized out>, argv=<optimized out>) at cat.c:39
(gdb) 
```

Figure 2 bt output

اشکال زدایی ۴

دستور `print` یک عبارت را دریافت و مقدار آن را نشان میدهد. اما دستور `x` براساس آدرس عمل می کند و مقدار را نمایش می دهد. شایان ذکر است که این دو دستور در نحوه نشان دادن اطلاعات با یکدیگر تفاوت دارند. با استفاده از دستور `register name info` می تولن یک register خاص را نشان داد، که در آن نام register به عنوان آرگومان داده می شود.

اُشکال زدایی ۵

پرای مشاهده وضعیت ریجسترها از دستور info registers استفاده میشود.

Figure 3 info registers

با دستور `locals info` میتوانیم وضعیت متغیر های local را ببینیم. برای مشاهده متغیرهایی که به صورت Global تعریف شده اند، از دستور `variables info` بهره می گیریم.

```
(gdb) info locals
n = <optimized out>
(gdb) info variables
All defined variables:

File cat.c:
5:     char buf[512];

File umalloc.c:
21:     static Header base;
22:     static Header *freep;

Non-debugging symbols:
0x000000878 digits
0x000000b5c __bss_start
0x000000b5c __edata
0x000000d80 __end
(gdb) █
```

Figure4 info variables & locals

برخی عملیات ها با استفاده از edi و esi قابل انجام اند. edi در این عملیات ها به معنای source و edi به معنای destination مکان است.

اشکال زدایی ۶

استراکت input شامل سه متغیر های r و w ابتدای خط ورودی را در buffer نشان میدهد. با استفاده از دستور print مقدار این دو متغیر را به دست آورديم. در فایل console.c روی خط 512 یک Breakpoint گذاشتیم بعد از آنکه cursor را به راست بردیم مقدار input.e یک **واحد افزایش یافته** است. مقدار input نشان دهنده مقدار انتهايی خط در حال تایپ شدن در buffer است.

The screenshot shows a terminal window titled "Terminal" on an Ubuntu 64-bit 21.10 desktop environment. The window title bar also displays "daniel@ubuntu: ~/Desktop/OS_Lab_Project/xv6-public-master". The terminal content is a GDB session:

```
(gdb) file kernel
[...]
(gdb) b console.c:511
Breakpoint 1 at 0x00100ee1: file console.c, line 512.
(gdb) print input.e
$2 = 0x80110b08
(gdb) continue
Continuing.

Thread 1 hit Breakpoint 1, consoleintr (getc=0x001032a0 <kbdbuf>) at console.c:512
$2      release cons lock;
(gdb) print input.e
$2 = 2
(gdb) continue
Continuing.

Thread 1 hit Breakpoint 1, consoleintr (getc=0x001032a0 <kbdbuf>) at console.c:512
$2      release cons lock;
(gdb) print input.e
$2 = 2
(gdb)
```

اشکال زدایی ۷

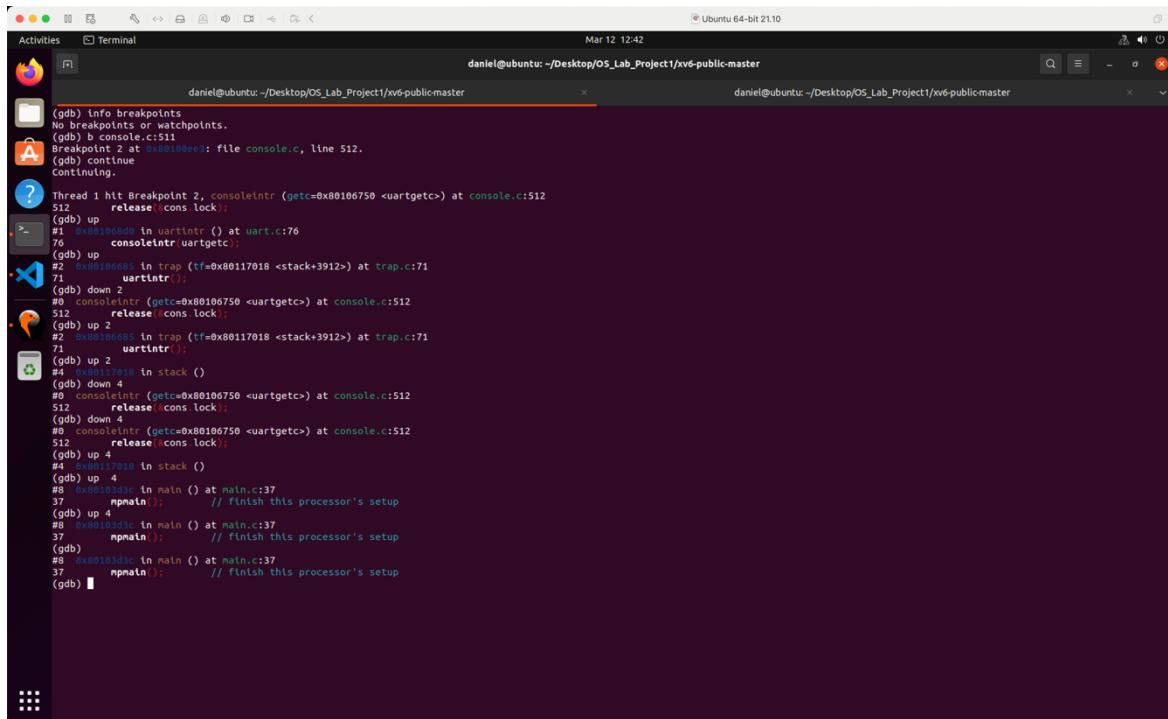
- با دستور **layout asm** میتوان برنامه را در حالت کد assembly دید.
- با دستور **layout src** میتوان برنامه را در حالت Source Code دید.

Figure 5 layout.asm

Figure 6 Layout src

اشکال زدایی ۸

با استفاده از دستورات up و down میتوان برای جابجایی میان توابع زنجیره ای فراخوانی جاری استفاده کرد. در ذیل چند نمونه آورده شده است:

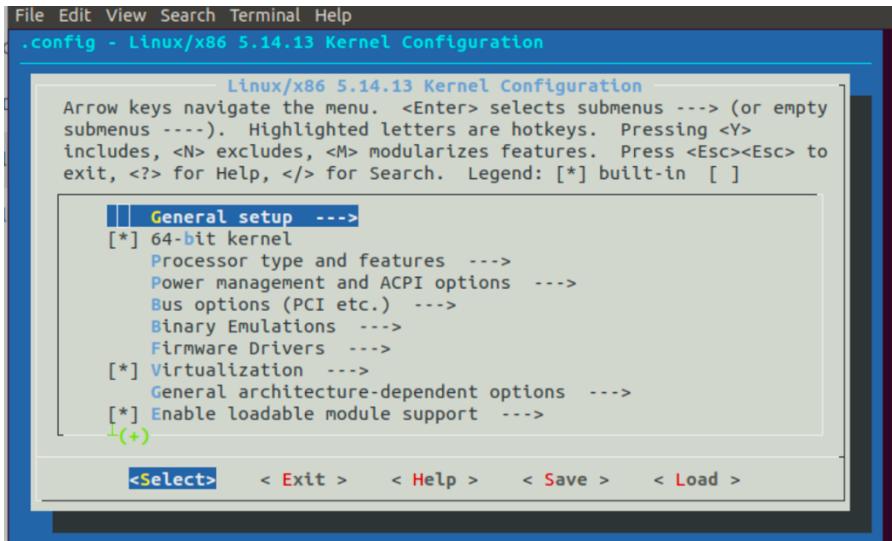


```
(gdb) info breakpoints  
No breakpoints or watchpoints.  
(gdb) b console.c:511  
Breakpoint 2 at 0x00100ee: file console.c, line 512.  
(gdb) continue  
Continuing.  
Thread 1 hit Breakpoint 2, consoleintr (getc=0x80106750 <uartgetc>) at console.c:512  
512     release(&cons.lock);  
(gdb) up  
#1 0x00106ed0 in uartintr () at uart.c:76  
76     consoleintr(uartgetc);  
(gdb) up  
#2 0x00106e55 in trap (@f=0x80117018 <stack+3912>) at trap.c:71  
71     uartintr();  
(gdb) down 2  
#0 0x00106e55 in trap (@f=0x80106750 <uartgetc>) at console.c:512  
512     release(&cons.lock);  
(gdb) up 2  
#2 0x00106e55 in trap (@f=0x80117018 <stack+3912>) at trap.c:71  
71     uartintr();  
(gdb) up 2  
#4 0x00117010 in stack ()  
(gdb) down 4  
#0 0x00106e55 in trap (@f=0x80106750 <uartgetc>) at console.c:512  
512     release(&cons.lock);  
(gdb) down 4  
#0 0x00106e55 in trap (@f=0x80106750 <uartgetc>) at console.c:512  
512     release(&cons.lock);  
(gdb) up 4  
#4 0x00117010 in stack ()  
(gdb) up  
#5 0x00106e5c in main () at main.c:37  
37     npmain(); // finish this processor's setup  
(gdb) up 4  
#8 0x0010303c in main () at main.c:37  
37     npmain(); // finish this processor's setup  
(gdb)  
#8 0x0010303c in main () at main.c:37  
37     npmain(); // finish this processor's setup  
(gdb)
```

قسمت امتیازی

طبق دستور العمل که کرنل linux را دانلود کردیم و سپس مراحل configure کردن کرنل را انجام دادیم.

دستور make menuconfig



```
dmesgout.txt
```

OPEN FILES dmesgout.txt

```

307 [ 0.882179] registered taskstats version 1
308 [ 0.882584] Loading compiled-in X.509 certificates
309 [ 0.883209] PM: Magic number: 9:271:338
310 [ 0.883606] tty ttv59: hash matches
311 [ 0.883972] printk: console [netcon0] enabled
312 [ 0.884388] netconsole: network logging started
313 [ 0.884868] cfg80211: Loading compiled-in X.509 certificates for regulatory e
314 [ 0.887238] modprobe (65) used greatest stack depth: 14376 bytes left
315 [ 0.889247] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47ae9cea7'
316 [ 0.889929] ALSA device list:
317 [ 0.890202] No soundcards found.
318 [ 0.890970] Freeing unused kernel image (initmem) memory: 1372K
319 [ 0.891555] platform regulatory.0: Direct firmware load for regulatory.db fa2
320 [ 0.892370] cfg80211: failed to load regulatory.db
321 [ 0.895850] Write protecting the kernel read-only data: 20480K
322 [ 0.897328] Freeing unused kernel image (text/rodata gap) memory: 2028K
323 [ 0.898042] Freeing unused kernel image (rodata/data gap) memory: 412K
324 [ 0.898649] Run /init as init process
325 [ 0.899003] with arguments:
326 [ 0.899003] /init
327 [ 0.899004] with environment:
328 [ 0.899004] HOME=/
329 [ 0.899005] TERM=linux
330 [ 1.008730] e1000 0000:00:03.0 enp0s3: renamed from eth0
331 [ 1.023790] udevadm (100) used greatest stack depth: 14352 bytes left
332 [ 1.052555] ata_id (110) used greatest stack depth: 13848 bytes left
333 [ 1.075631] random: fast init done
334 [ 1.448186] tsc: Refined TSC clocksource calibration: 2304.012 MHz
335 [ 1.450945] clocksource: tsc: mask: 0xffffffffffff max_cycles: 0x2136031s
336 [ 1.455042] clocksource: Switched to clocksource tsc
337 [ 1.498682] input: ImEXPS/2 Generic Explorer Mouse as /devices/platform/i8043
338 [ 36.885099] systemd-udevd (95) used greatest stack depth: 13336 bytes left
339 <12>[ 74.840991] group23:DanielSaiedi-MohammadGhareHasanloo-SorooshSadegian
340 [ 95.304565] process '/usr/bin/dmesg' started with executable stack
341
342

```

Line 15, Column 35 Tab Size: 4 Plain Text