

پروژه اول آزمایشگاه سیستم عامل

سید محمد امین اطمینانی (۸۱۰۱۹۸۵۵۹)

شایان شاه محمدی (۸۱۰۱۹۸۵۳۱)

مرتضی نوری (۸۱۰۱۹۸۴۸۱)

۱ - معماری سیستم عامل xv6 مبتنی بر پردازنده های x86 می باشد . با مطالعه فایل های traps.h ، asm.h ، mmu.h و x86.h مشخصاً به معماری x86 این سیستم عامل پی می بریم . در این فایل ها به المان های مختلف این پردازنده اشاره شده است .

۲ - یک پردازش در سیستم عامل xv6 از instruction ، data ، stack و وضعیت پیش پردازش که مخفی از هسته است تشکیل شده است . سیستم عامل xv6 بین پردازش های time-share برقرار می کند : سیستم عامل به دور از چشم کاربر (transparently) پردازنده ای به پردازش منتظر اجرا جهت اجرا اختصاص می دهد .

۴ -

- basic headers : بسیاری از ثابت ها در فایل های مختلف این دسته بندی قرار گرفته اند .
 - entering xv6 : فایل های این دسته بندی وظیفه initialize کردن سیستم عامل جهت شروع به کار را دارند .
 - locks : عملیات های مربوط به قفل کردن خانه هایی از حافظه در این دسته بندی مدیریت می شوند .
 - processes : در فایل های این بخش فرآیندهای مربوط به اختصاص حافظه و زمان بندی پردازش ها انجام می شود .
 - system calls : سیستم کال های سیستم عامل xv6 در این دسته بندی پیاده سازی شده اند .
 - file system : تعریف و پیاده سازی فایل سیستم های سیستم عامل xv6 در این بخش صورت گرفته است .
 - pipes : ساختار pipe در این بخش تعریف شده است . به طوری کلی یک pipe بافر کوچکی در کرنل است که قابل دسترسی توسط پردازش ها است .
 - string operations : برخی توابع برای کار کردن با رشته ها .
 - low-level hardware : در این بخش ارتباطات سخت افزاری جهت ورودی/خروجی و interrupt ها برقرار شده است .
 - user-level : ارتباط کاربر با سیستم عامل از طریق فایل های این دسته بندی انجام می شود . برای مثال رابط کاربر shell از این طریق در اختیار کاربر قرار می گیرد .
 - bootloader : ارتباط با BOIS و تحویل سیستم به xv6 در این بخش انجام می شود .
 - link : فایل موجود در این بخش یک لینکر به JOS می باشد .
- طبق صفحه گیت هاب لینوکس :

- ❖ فایل های هسته اصلی لینوکس در پوشه kernel قرار دارند .
- ❖ فایل های سرآیند لینوکس در پوشه include قرار دارند .
- ❖ فایل سیستم های لینوکس در پوشه fs قرار دارد .

۸ - با توجه خروجی دستور make -n و اینکه داده های مورد نیاز دیسک قابل بوت در سکتور 0x7X00 ذخیره می شود می توان فهمید در سکتور نخست xv6 چه داده ای ذخیره شده است :

```
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -fno-pic -O -nostdinc -l. -c bootmain.c
```

```
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -fno-pic -nostdinc -l. -c bootasm.S
```

```
ld -m elf_i386 -N -e start -Ttext 0x7C00 -o bootblock.o bootasm.o bootmain.o
```

```
gcc -m32 -gdwarf-2 -Wa,-divide -c -o vectors.o vectors.S
```

```
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o vm.o vm.c
```

```
gcc -m32 -gdwarf-2 -Wa,-divide -c -o entry.o entry.S
```

```
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -fno-pic -nostdinc -l. -c entryother.S
```

```
ld -m elf_i386 -N -e start -Ttext 0x7000 -o bootblockother.o entryother.o
```

۱۱ - فایل اسمبلی پردازنده را در حالت محافظت شده ۳۲ بیتی قرار می‌دهد و اختیار را به کد C واگذار می‌کند. کد C به تنهایی سرعت و دسترسی‌ها لازم را ندارد.

۱۲ -

- ثبات عام منظوره: ثبات ax که برای عملیات محاسباتی انجام می‌شود.
- ثبات قطعه: ثبات SS که برای اشاره به استک استفاده می‌شود.
- ثبات وضعیت: ثبات OF که وضعیت سرریز را مشخص می‌کند.
- ثبات کنترلی: ثبات PE که مشخص می‌کند آیا سیستم در Protected Mode هست یا خیر.

۱۴ -

- حالت محافظت شده: در این حالت به نرم‌افزار دسترسی به حافظه مجازی، مالتی تسکینگ جهت کنترل بیشتر روی نرم‌افزار و ... داده می‌شود.
- حالت حقیقی: در این حالت به نرم‌افزار دسترسی نامحدود به حافظه فیزیکی، ورودی/خروجی، و دستگاه‌های سخت-افزاری داده می‌شود.

۱۹ - زیرا ما می‌خواهیم از این جدول برای تبدیل آدرس‌های مجازی به آدرس‌های فیزیکی استفاده کنیم و اگر مجازی بود باید یک بخش اضافه فیزیکی نیز برای تبدیل این آدرس مجازی اضافه می‌کردیم.

۲۰ -

- kinit1: صفحه‌های خالی در حافظه فیزیکی به هسته اختصاص می‌دهد.
- Kvmalloc: صفحه نگاشت‌ها را در حافظه فیزیکی ایجاد می‌کند و آدرس آن را در ثبات کنترلی cr3 ذخیره می‌کند.
- mpinit: پردازنده‌های دیگر را تشخیص می‌دهد و اجازه می‌دهد با چند پردازنده کار کنیم.
- Lapicinit: باعث وقفه انداختن در پردازنده می‌شود.
- seginit: جدول ثبات‌ها را تعریف می‌کند.
- Picinit: وقفه ایجاد شده توسط lapicinit را متوقف می‌کند.
- loapicinit: باعث وقفه در پردازنده می‌شود. (برای کنترل ورودی و خروجی)
- Consoleinit: کنترل ورودی و خروجی.
- Uartinit: سریال پورت را برای وصل شدن سخت‌افزارها تعریف می‌کند.
- Pinit: جدول نگاشتی برای ذخیره کردن آدرس فیزیکی داده‌ها در پردازنده‌ها را تعریف می‌کند.
- Tvinit: وکتوری برای ذخیره آدرس پردازنده‌هایی که باعث trap شده‌اند تعریف می‌کند.
- Binit: بافر را تعریف می‌کند.

- Fileinit : جدولی برای ذخیره کردن فایل‌های در حال استفاده در هسته تعریف می‌کند.
- Ideinit : درایور دیسک را تعریف می‌کند.
- Startothers : پردازنده‌های دیگر را آماده اجرا می‌کند.
- Kinit2 : تعریف صفحه‌های خالی بیشتر در حافظه فیزیکی (باید بعد از startothers اجرا شود)
- userinit : اولین پردازش کاربر را ایجاد می‌کند.
- Mpmmain : آماده‌سازی اجرای هسته را تمام می‌کند و شروع به اجرای پردازش‌ها می‌کند.
- تابع معدل entry.s در linux :

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

۲۱ - فضای مجازی هسته شامل داده‌ها و اطلاعات کاربر و هسته است که خود فضای هسته نیز شامل BIOS ، داده‌های پردازش‌های هسته و اطلاعات و آدرس‌های دستگاه‌های ورودی خروجی است.

۲۲ - برای اینکه مشخص شود آن داده‌ها، داده‌های سطح کاربر هستند و اجازه دسترسی به هسته ندارند.

۲۳ -

- pgdir : آدرس جدول نگاشت پردازش‌ها را نگه می‌دارد.
- Sz : مقدار بایتی که توسط حافظه پردازش مصرف می‌شود را نگه می‌دارد.
- Tf : قاب تله را برای پردازش فعلی تعریف می‌کند تا در صورت لزوم استفاده کند.
- Cwd : مسیر فولدر فعلی.
- Name : نام پردازش را برای دیباگینگ مشخص می‌کند.
- State : وضعیت پردازش را مشخص می‌کند.
- Mem : آدرس شروع حافظه پردازش را ذخیره می‌کند.
- Kstack : پایین استک هسته برای این پردازش را مشخص می‌کند.
- Pid : آیدی پردازش.
- Parent : پدر پردازنده را مشخص می‌کند.
- Context : هنگامی که از یک پردازش به پردازش دیگر می‌رویم، مقدار برگشت را ذخیره می‌کند تا پردازش قبلی را از جای درست ادامه دهیم.
- Chan : اگر غیرصفر باشد، یعنی پردازش خوابیده و منتظر ورودی است.
- Killed : اگر غیرصفر باشد، یعنی پردازش کشته شده.
- Ofile : فایل‌های باز شده پردازش فعلی را نگه می‌دارد.
- ساختار معادل در لینوکس :

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

۲۵- kvmalloc نگاشتی برای آدرس‌دهی و مدیریت آدرس پردازش‌ها ایجاد می‌کند ولی setupkvm برای هر پردازش بطور جداگانه نگاشتی برای مدیریت آدرس داده‌های آن پردازنده در حال اجرا می‌سازد.

۲۷- پردازش در حال اجرا را با پردازش‌ای که به آن می‌دهیم جایگزین می‌کنیم و پردازش جدید را اجرا می‌کند.

اشکال زدایی

(۱) از دو دستور می‌توانیم استفاده کنیم :

```
info breakpoints [number 1] [number 2] ...[number N]
```

```
maint info breakpoints
```

نحوه استفاده از دستور دوم نیز همانند دستور اول است با این تفاوت که دستور دوم علاوه بر breakpoint هایی که کاربر صریحاً تعریف کرده است، آن‌هایی که GDB تعریف کرده است (با اهداف درونی) را نیز نمایش می‌دهد.

(۲) از دستورهای مختلفی می‌توانیم استفاده کنیم که مهم‌ترین آن‌ها عبارتند از:

- breakpoint ای را که روی یک تابع گذاشته شده است، حذف می‌کند.

```
clear function
```

```
clear filename:function
```

- breakpoint ای را که روی یک خط گذاشته شده است، حذف می‌کند.

```
clear linename
```

```
clear filename:linename
```

(۳) این دستور نشان دهنده این است که برنامه چطور به جایی که اکنون قرار دارد رسیده است. برای مثال در شکل زیر فریم ۰، فریم فعلی است، فریم ۱، جایی که فریم ۰ صدا زده شده است. (و به همین ترتیب)

```
morteza@morteza: ~/OS_xv6_Project/Source
File Edit View Search Terminal Help
(gdb) target remote tcp::26000
Remote debugging using tcp::26000
0x0000ffff in ?? ()
(gdb) b exec
Breakpoint 1 at 0x80100a10: file exec.c, line 12.
(gdb) c
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 1, exec (path=0x1c "/init", argv=0x8dffed0) at exec.c:12
12 {
(gdb) c
Continuing.

Thread 2 hit Breakpoint 1, exec (path=0x826 "sh", argv=0x8dfeed0) at exec.c:12
12 {
(gdb) c
Continuing.
[Switching to Thread 1]

Thread 1 hit Breakpoint 1, exec (path=0x1880 "ls", argv=0x8dfbeed0) at exec.c:12
12 {
(gdb) bt
#0 exec (path=0x1880 "ls", argv=0x8dfbeed0) at exec.c:12
#1 0x80105390 in sys_exec () at sysfile.c:419
#2 0x80104869 in syscall () at syscall.c:139
#3 0x80105825 in trap (tf=0x8dfbefb4) at trap.c:43
#4 0x8010563f in alltraps () at trapasm.S:20
#5 0x8dfbefb4 in ?? ()
```

(۴)

- تفاوت ۱:

- ❖ دستور p(int) مقدار ذخیره شده در یک متغیر را نشان می‌دهد.

- ❖ دستور x محتوای ذخیره شده در یک آدرس را نشان می‌دهد.

- تفاوت ۲:

- ❖ دستور p(int) یک عبارت (مثلاً نام متغیر) را می‌گیرد.

- ❖ دستور x آدرس خانه‌ای از حافظه را می‌گیرد.

با استفاده از دستور `info register name` می‌توان محتوای یک ثبات خاص را چاپ نمود. `Name`، نام ثبات مدنظر است.

```
(gdb) c
Continuing.

Thread 1 hit Breakpoint 1, exec (path=0x1880 "forktest", argv=0x8df23ed0) at exec.c:12
12 {
(gdb) p argv[0]
$1 = 0x1880 "forktest"
(gdb) x 0x1880
0x1880: 0x6b726f66
(gdb) info register eax
eax      0x8df23ed0      -1913504048
(gdb)
```

(۵) با استفاده از دستور `info registers` می‌توانیم وضعیت ثبات‌ها را نمایش بدهیم:

```
(gdb) info registers
eax      0x8dfbeed0      -1912869168
ecx      0x8dfbeed4      -1912869164
edx      0x0             0
ebx      0x2             2
esp      0x8dfbeeac      0x8dfbeeac
ebp      0x8dfbef68      0x8dfbef68
esi      0x8             8
edi      0x8dfbeecc      -1912869172
eip      0x80100c00      0x80100c00 <exec>
eflags   0x296          [ PF AF SF IF ]
cs       0x8             8
ss       0x10            16
ds       0x10            16
es       0x10            16
fs       0x0             0
gs       0x0             0
(gdb)
```

```
amin@SM2A: /mnt/c/Users/SM  x  amin@SM2A: /mnt/c/Users/SM  x  +  v
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
Seyed Mohammad Amin Atyabi, Shayan Shahmohammadi, Morteza Nouri
$ QEMU 4.2.1 monitor - type 'help' for more information
(qemu) info registers
EAX=fee00000 EBX=80112d20 ECX=00000000 EDX=00000001
ESI=00000000 EDI=80112784 EBP=8010b538 ESP=8010b52c
EIP=80102990 EFL=00000086 [--S--P--] CPL=0 II=0 A20=1 SMM=0 HLT=0
ES =0010 00000000 ffffffff 00cf9300 DPL=0 DS [-WA]
CS =0008 00000000 ffffffff 00cf9a00 DPL=0 CS32 [-R-]
SS =0010 00000000 ffffffff 00cf9300 DPL=0 DS [-WA]
DS =0010 00000000 ffffffff 00cf9300 DPL=0 DS [-WA]
FS =0000 00000000 00000000 00000000
GS =0000 00000000 00000000 00000000
LDT=0000 00000000 0000ffff 00008200 DPL=0 LDT
TR =0028 80112788 00000067 00408900 DPL=0 TSS32-avl
GDT= 801127f0 0000002f
IDT= 80114ca0 000007ff
CR0=80010011 CR2=00000000 CR3=003ff000 CR4=00000010
DR0=00000000 DR1=00000000 DR2=00000000 DR3=00000000
DR6=ffff0fff 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=00000000000000000000000000000000 XMM01=00000000000000000000000000000000
XMM02=00000000000000000000000000000000 XMM03=00000000000000000000000000000000
XMM04=00000000000000000000000000000000 XMM05=00000000000000000000000000000000
XMM06=00000000000000000000000000000000 XMM07=00000000000000000000000000000000
(qemu) |
```

با استفاده از دستور `info locals` می‌توانیم وضعیت متغیرهای محلی را مشاهده کنیم، همچنین با استفاده از دستور `info variables` می‌توانیم تمام متغیرهای `global/static` را در برنامه تحت اشکال زدایی مشاهده کنیم.

```
(gdb) info locals
s = <optimized out>
last = <optimized out>
i = <optimized out>
off = <optimized out>
argc = <optimized out>
sz = <optimized out>
sp = <optimized out>
ustack = {16843009 <repeats 14 times>, 2382097992, 2148546881, 2148609612, 0, 2382098008, 2148549845, 6272, 0, 2382098024,
2148547052, 16843009, 6272, 2382098040, 2148546881, 2148609612, 8, 2382098056, 2148549845, 49076, 8, 2382098072,
2148547052}
elf = {magic = 16843009, elf = '\001' <repeats 12 times>, type = 257, machine = 257, version = 16843009, entry = 16843009,
phoff = 16843009, shoff = 16843009, flags = 16843009, ehsize = 257, phentsize = 257, phnum = 257, shentsize = 257,
shnum = 257, shstrndx = 257}
ip = <optimized out>
ph = {type = 16843009, off = 16843009, vaddr = 16843009, paddr = 16843009, filesz = 16843009, memsz = 16843009,
flags = 16843009, align = 16843009}
pgdir = <optimized out>
oldpgdir = <optimized out>
curproc = <optimized out>
(gdb) █
```

All defined variables:

```
File bio.c:
struct {
    struct spinlock lock;
    struct buf buf[30];
    struct buf head;
} bcache;

File console.c:
struct {
    char buf[128];
    uint r;
    uint w;
    uint e;
} input;
static struct {
    struct spinlock lock;
    int locking;
} cons;
static ushort *crt;
static int panicked;

File file.c:
struct devsw devsw[];
struct {
    struct spinlock lock;
    struct file file[100];
} █
---Type <return> to continue, or q <return> to quit---
```

کاربرد های رجیستر edi :

رجیستر مکان مقصد است و برای عملیات مربوط به رشته ها، کپی کردن و مقداردهی آرایه حافظه و برای آدرس دهی اشاره گرهای دور از طریق رجیستر ES کاربرد دارد.

کاربرد های رجیستر esi :

رجیستر مکان مبدا: رجیستر مکان مبدا است و برای عملیات های مربوط به رشته ها و کپی کردن آرایه حافظه ای کاربرد دارد.

۶) به طور کلی این استراکت ورودی هایی که به کنسول داده می شود (هرچیزی که تایپ می شود) را نگه داری می کند این استراکت شامل متغیر های زیر است :

char buf[INPUT_BUF]: نشان دهنده سایز بافر است که در اینجا ۱۲۸ کاراکتر است.

uint r: نشان دهنده مکانی است که باید اطلاعات بافر را از آنجا بخوانیم.

uint w: نشان دهنده مکانی است که باید اطلاعات بافر را از آنجا بنویسیم.

uint e: نشان دهنده مکان بعدی cursor است.

نحوه تغییر متغیرهای درونی: برای مثال اگر یک کاراکتر را در کنسول بنویسیم، مقدار input.e یکی اضافه می‌شود (cursor به راست حرکت می‌کند) – اگر اینتر بزنیم مقدار input.r و input.w نیز یکی اضافه می‌شود به این معنی که کاراکتر وارد شده خوانده و نوشته شده است و منتظر ورودی‌های بعدی هستیم. در این حالت پایا، مقدار سه متغیر input.w، input.e و input.r برابر خواهند بود.

```
(gdb) p input.r
$1 = 0
(gdb) p input.w
$2 = 0
(gdb) p input.e
$3 = 0
(gdb) c
Continuing.

Thread 1 hit Breakpoint 1, consoleintr (getc=0x80102530 <kbdbgetc>) at console.c:193
193 {
(gdb) p input.r
$4 = 0
(gdb) p input.w
$5 = 0
(gdb) p input.e
$6 = 1
(gdb) c
Continuing.

Thread 1 hit Breakpoint 1, consoleintr (getc=0x80102530 <kbdbgetc>) at console.c:193
193 {
(gdb) p input.r
$7 = 2
(gdb) p input.w
$8 = 2
(gdb) p input.e
$9 = 2
```

(۷)

layout src : برنامه در حالت کد سورس (در اینجا c) در gdb نمایش داده می‌شود.

layout asm : برنامه در حالت کد اسمبلی در gdb نمایش داده می‌شود.

```
console.c
186     uint e; // Edit index
187 } input;
188
189 #define C(x) ((x)-'@') // Control-x
190
191 void
192 consoleintr(int (*getc)(void))
B+> 193 {
194     int c, doprocdump = 0;
195
196     acquire(&cons.lock);
197     while((c = getc()) >= 0){
198         switch(c){
199             case C('P'): // Process listing.
200                 // procdump() locks cons.lock indirectly; invoke later
201                 doprocdump = 1;
202         }
203     }
204 }

remote Thread 1 In: consoleintr L193 PC: 0x80100810
(gdb) layout src

B+> 0x801008dc <consoleintr+204> mov     %edi,%eax
0x801008de <consoleintr+206> call   0x80100410 <consputc>
0x801008e3 <consoleintr+211> cmp     $0xa,%edi
0x801008e6 <consoleintr+214> je      0x801009b1 <consoleintr+417>
0x801008ec <consoleintr+220> cmp     $0x4,%edi
0x801008ef <consoleintr+223> je      0x801009b1 <consoleintr+417>
0x801008f5 <consoleintr+229> mov     0x8010ffa0,%eax
0x801008fa <consoleintr+234> sub     $0xffffffff80,%eax
0x801008fd <consoleintr+237> cmp     %eax,0x8010ffa8
0x80100903 <consoleintr+243> jne     0x80100830 <consoleintr+32>
0x80100909 <consoleintr+249> sub     $0xc,%esp
0x8010090c <consoleintr+252> mov     %eax,0x8010ffa4
0x80100911 <consoleintr+257> push    $0x8010ffa0
0x80100916 <consoleintr+262> call    0x80103f30 <wakeup>
0x8010091b <consoleintr+267> add     $0x10,%esp
0x8010091e <consoleintr+270> jmp     0x80100830 <consoleintr+32>

remote Thread 1 In: consoleintr L220 PC: 0x801008dc
(gdb)
```


۸) می‌توان از دستور up (به منظور رفتن به استیک فریم یا فریم قبلی (بیرونی)) و دستور down (به منظور رفتن به استیک فریم یا فریم بعدی (درونی)) استفاده کرد.

```
(gdb) bt
#0  consoleintr (getc=0x80102530 <kbdgetc>) at console.c:229
#1  0x80102620 in kbdintr () at kbd.c:49
#2  0x8010587d in trap (tf=0x8010b508 <stack+3912>) at trap.c:67
#3  0x8010563f in alltraps () at trapasm.S:20
#4  0x8010b508 in stack ()
#5  0x80112784 in cpus ()
#6  0x80112780 in ?? ()
#7  0x80102e7f in mpmain () at main.c:57
#8  0x80102fbf in main () at main.c:37
(gdb) up
#1  0x80102620 in kbdintr () at kbd.c:49
49      consoleintr(kbdgetc);
(gdb) up
#2  0x8010587d in trap (tf=0x8010b508 <stack+3912>) at trap.c:67
67      kbdintr();
(gdb) up
#3  0x8010563f in alltraps () at trapasm.S:20
20      call trap
(gdb) up
#4  0x8010b508 in stack ()
(gdb) down
#3  0x8010563f in alltraps () at trapasm.S:20
20      call trap
(gdb) down
#2  0x8010587d in trap (tf=0x8010b508 <stack+3912>) at trap.c:67
67      kbdintr();
(gdb) □
```

اضافه کردن متن به بوت

```
QEMU

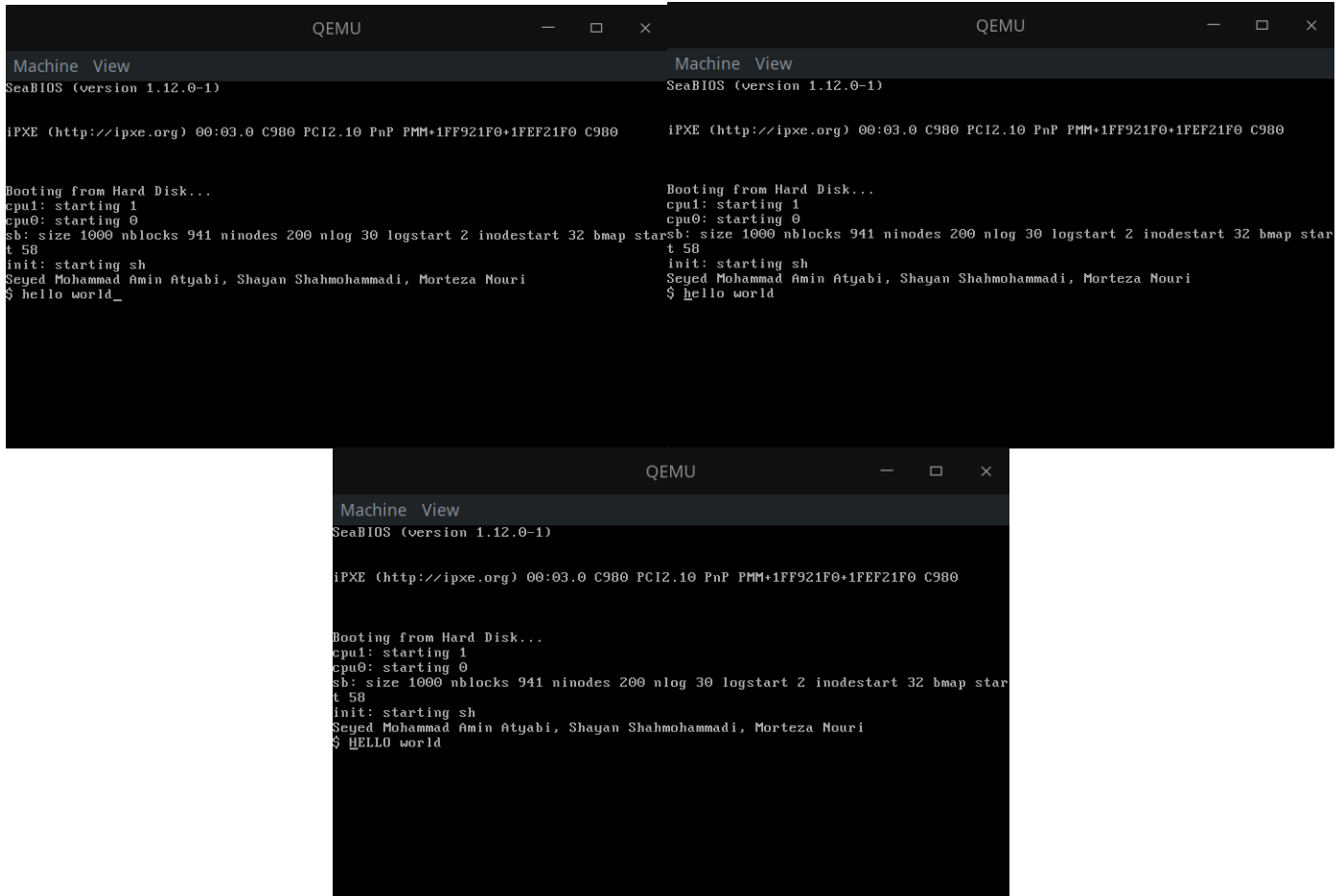
Machine View
SeaBIOS (version 1.12.0-1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF921F0+1FEF21F0 C980

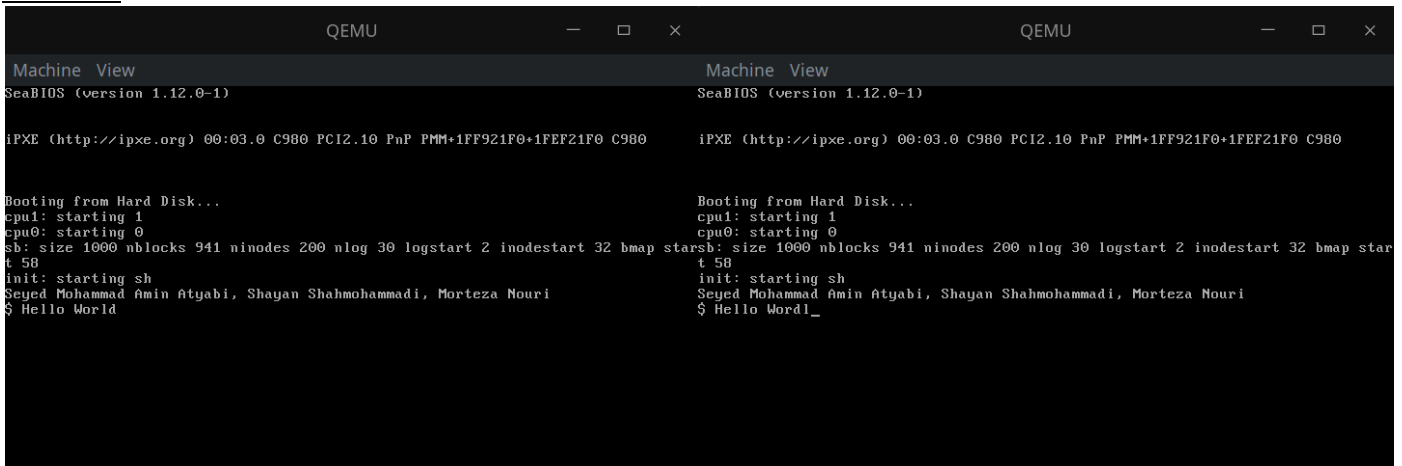
Booting from Hard Disk...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
Seyed Mohammad Amin Atyabi, Shayan Shahmohammadi, Morteza Nouri
$ _
```

اضافه کردن چند قابلیت به کنسول xv6

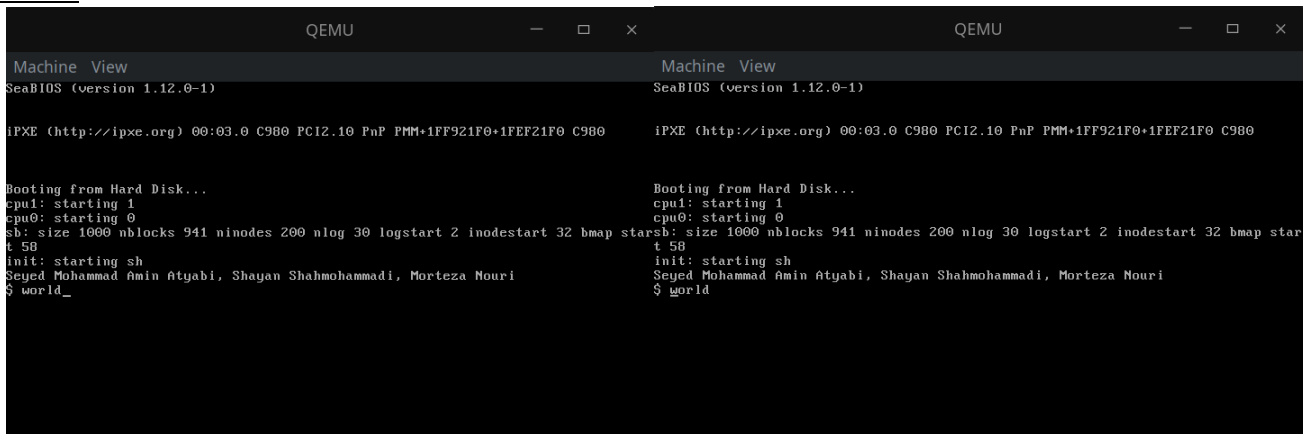
CTRL O :



CTRL T :



CTRL A :



```
QEMU

Machine View
SeaBIOS (version 1.12.0-1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF921F0+1FEF21F0 C980

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
Seyed Mohammad Amin Atyabi, Shayan Shahmohammadi, Morteza Nouri
$ hello world
```

اجرا برنامه سطح کاربر

```
QEMU

Machine View
SeaBIOS (version 1.12.0-1)

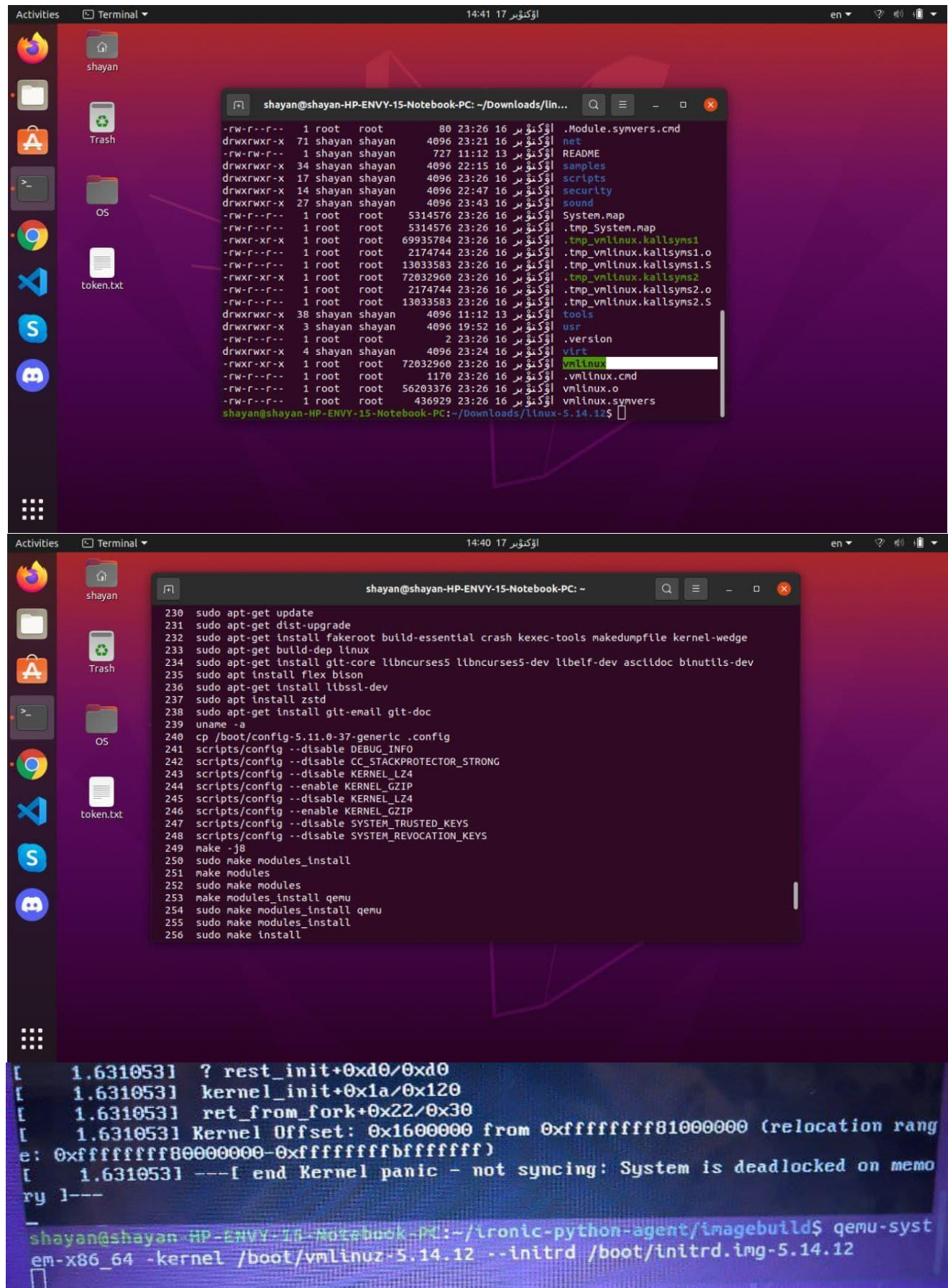
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF921F0+1FEF21F0 C980

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
Seyed Mohammad Amin Atyabi, Shayan Shahmohammadi, Morteza Nouri
$ factor 20
$ cat factor_result.txt
1 2 4 5 10 20

$ factor 224
$ cat factor_result.txt
1 2 4 7 8 14 16 28 32 56 112 224

$ _
```

ساختن هسته لینوکس



```
shayan@shayan-HP-ENVY-15-Notebook-PC: ~/Downloads/linux-5.14.12$ ls -la
-rw-r--r-- 1 root root      80 23:26 16 اؤكئؤ بر .Module.symvers.cnd
drwxrwxr-x 71 shayan shayan 4096 23:21 16 اؤكئؤ بر net
-rw-rw-r-- 1 shayan shayan 727 11:12 13 اؤكئؤ بر README
drwxrwxr-x 34 shayan shayan 4096 22:15 16 اؤكئؤ بر samples
drwxrwxr-x 17 shayan shayan 4096 23:26 16 اؤكئؤ بر scripts
drwxrwxr-x 14 shayan shayan 4096 22:47 16 اؤكئؤ بر security
drwxrwxr-x 27 shayan shayan 4096 23:43 16 اؤكئؤ بر sound
-rw-r--r-- 1 root root 5314576 23:26 16 اؤكئؤ بر System.map
-rw-r--r-- 1 root root 5314576 23:26 16 اؤكئؤ بر .tmp_System.map
-rwxr-xr-x 1 root root 69935784 23:26 16 اؤكئؤ بر .tmp_vmlinux.kallsyms1
-rw-r--r-- 1 root root 2174744 23:26 16 اؤكئؤ بر .tmp_vmlinux.kallsyms1.o
-rw-r--r-- 1 root root 13033583 23:26 16 اؤكئؤ بر .tmp_vmlinux.kallsyms1.S
-rwxr-xr-x 1 root root 72032960 23:26 16 اؤكئؤ بر .tmp_vmlinux.kallsyms2
-rw-r--r-- 1 root root 2174744 23:26 16 اؤكئؤ بر .tmp_vmlinux.kallsyms2.o
-rw-r--r-- 1 root root 13033583 23:26 16 اؤكئؤ بر .tmp_vmlinux.kallsyms2.S
drwxrwxr-x 38 shayan shayan 4096 11:12 13 اؤكئؤ بر tools
drwxrwxr-x 3 shayan shayan 4096 19:52 16 اؤكئؤ بر usr
-rw-r--r-- 1 root root 2 23:26 16 اؤكئؤ بر .version
drwxrwxr-x 4 shayan shayan 4096 23:24 16 اؤكئؤ بر virt
-rwxr-xr-x 1 root root 72032960 23:26 16 اؤكئؤ بر vmlinux
-rw-r--r-- 1 root root 1170 23:26 16 اؤكئؤ بر .vmlinux.cmd
-rw-r--r-- 1 root root 56203376 23:26 16 اؤكئؤ بر vmlinux.o
-rw-r--r-- 1 root root 436929 23:26 16 اؤكئؤ بر vmlinux.symvers
shayan@shayan-HP-ENVY-15-Notebook-PC: ~/Downloads/linux-5.14.12$
```

```
230 sudo apt-get update
231 sudo apt-get dist-upgrade
232 sudo apt-get install fakeroot build-essential crash kexec-tools makedumpfile kernel-wedge
233 sudo apt-get build-dep linux
234 sudo apt-get install git-core libncurses5 libncurses5-dev libelf-dev asciidoc binutils-dev
235 sudo apt install flex bison
236 sudo apt-get install libssl-dev
237 sudo apt install zstd
238 sudo apt-get install git-email git-doc
239 uname -a
240 cp /boot/config-5.11.0-37-generic .config
241 scripts/config --disable DEBUG_INFO
242 scripts/config --disable CC_STACKPROTECTOR_STRONG
243 scripts/config --disable KERNEL_LZ4
244 scripts/config --enable KERNEL_GZIP
245 scripts/config --disable KERNEL_LZ4
246 scripts/config --enable KERNEL_GZIP
247 scripts/config --disable SYSTEM_TRUSTED_KEYS
248 scripts/config --disable SYSTEM_REVOCATION_KEYS
249 make -j8
250 sudo make modules_install
251 make modules
252 sudo make modules
253 make modules_install qemu
254 sudo make modules_install qemu
255 sudo make modules_install
256 sudo make install
```

```
[ 1.6310531 ? rest_init+0xd0/0xd0
[ 1.6310531 kernel_init+0x1a/0x120
[ 1.6310531 ret_from_fork+0x22/0x30
[ 1.6310531 Kernel Offset: 0x1600000 from 0xffffffff81000000 (relocation rang
e: 0xffffffff80000000-0xffffffffbfffffff)
[ 1.6310531 ---[ end Kernel panic - not syncing: System is deadlocked on memo
ry 1---

shayan@shayan-HP-ENVY-15-Notebook-PC: ~/ironic-python-agent/imagebuild$ qemu-syst
em-x86_64 -kernel /boot/vmlinux-5.14.12 --initrd /boot/initrd.img-5.14.12
```