

# Project1 report

رضا چهرقانی 810101401

امیر نداف فهمیده 810101540

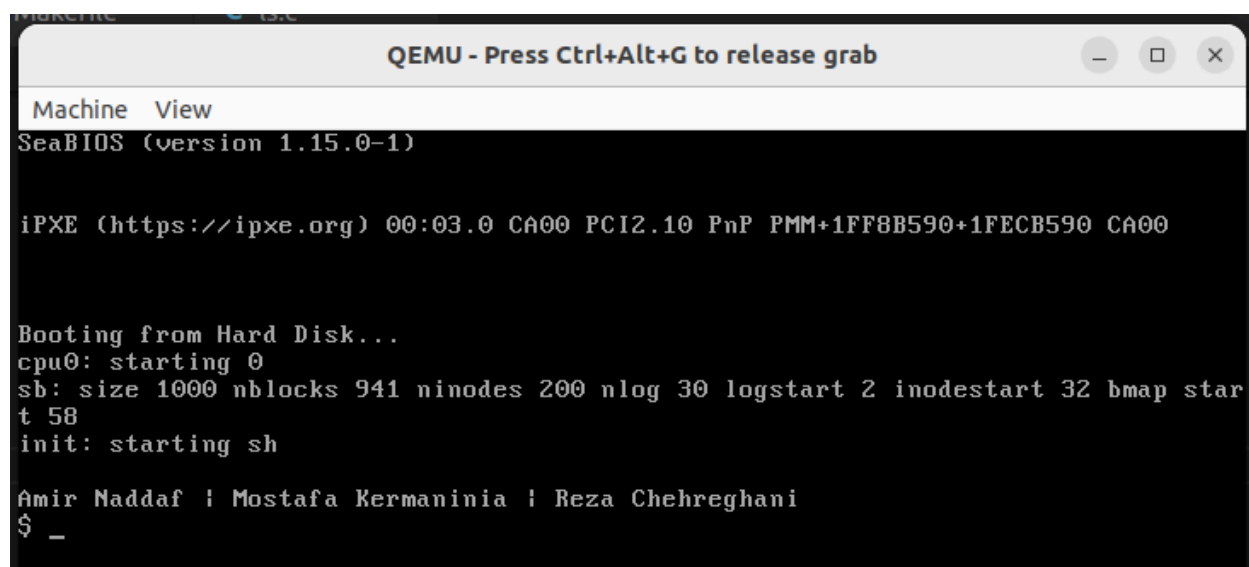
مصطفی کرمانی نیا 810101575

مخزن گیتهاب این پروژه:

<https://github.com/reza-chehreghani/OS-Lab-xv6>

## ● اضافه کردن یک متن به Message Boot

نام اعضای گروه پس از بوت شدن سیستم عامل روی ماشین مجازی Qemu، در انتهای پیام های نمایش داده شده در کنسول نشان داده می شود:



```
QEMU - Press Ctrl+Alt+G to release grab
Machine View
SeaBIOS (version 1.15.0-1)

iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8B590+1FECB590 CA00

Booting from Hard Disk...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh

Amir Maddaf | Mostafa Kermaninia | Reza Chehreghani
$ _
```

---

- اضافه کردن قابلیت های جدید به کنسول

1. جلو و عقب رفتن curse

2. نمایش history

3. عملیات های copy & paste

4. عملیات های ریاضی

- برنامه سطح کاربر

- مقدمه ای درباره سیستم عامل و xv6

سوال 1) سه وظیفه اصلی سیستم عامل را نام ببرید.

1. مدیریت و ساده سازی دسترسی به سخت افزار: سیستم عامل وظیفه دارد که سخت افزار کامپیوتر را کنترل و مدیریت کند و جزئیات فنی پیچیده آن را از دید برنامه ها پنهان سازد. به این ترتیب، برنامه ها بدون نیاز به درک مستقیم از سخت افزار، می توانند از منابع مختلف سیستم استفاده کنند.

- 
2. اشتراک‌گذاری منابع بین برنامه‌ها: سیستم عامل منابع سخت‌افزاری مانند CPU، حافظه و دستگاه‌های ورودی/خروجی را به‌طور منظم بین برنامه‌های مختلف به اشتراک می‌گذارد، تا به نظر برسد که این برنامه‌ها به صورت همزمان اجرا می‌شوند، حتی اگر در واقعیت اینگونه نباشد.
3. فراهم کردن بستری امن برای ارتباط و همکاری بین برنامه‌ها: سیستم عامل راه‌های ایمن و کنترل‌شده‌ای را برای تعامل و تبادل داده میان برنامه‌های مختلف ارائه می‌دهد تا بتوانند به صورت مؤثر با هم کار کنند، بدون اینکه امنیت و ثبات سیستم به خطر بیافتد.

**سوال 2) فایل‌های اصلی سیستم‌عامل xv6 در صفحه یک کتاب xv6 لیست شده‌اند. به طور مختصر هر گروه را توضیح دهید. نام پوشه اصلی فایل‌های هسته سیستم‌عامل، فایل‌های سرایند و فایل سیستم در سیستم‌عامل لینوکس چیست؟ در مورد محتویات آن مختصراً توضیح دهید.**

#### 1. Basic Headers (فایل‌های سرایند اصلی)

فایل‌های سرایند این دسته شامل تعاریفات پایه‌ای و اساسی سیستم‌عامل هستند. آن‌ها ساختارهای داده، انواع داده‌های سفارشی، پارامترهای اصلی سیستم (مانند تعداد ماکسیمم فرآیندها)، ساختار چیدمان حافظه و آدرس قسمت‌های مختلف در حافظه و توابع مهمی را تعریف می‌کنند که در بخش‌های مختلف سیستم‌عامل استفاده می‌شوند. همچنین شامل تعاریفات مرتبط با معماری x86 و اطلاعات مربوط به واحد مدیریت حافظه (MMU) و فرمت اجرایی فایل‌های باینری (ELF) و ساختارهای مربوط به تاریخ و زمان سیستم می‌شوند. این فایل‌ها پایه‌ای برای ارتباط سیستم‌عامل با سخت‌افزار و نرم‌افزار هستند.

#### 2. Entering xv6 (ورود به xv6)

این دسته از فایل‌ها مربوط به راه‌اندازی اولیه سیستم‌عامل است. فایل‌های اسمبلی و C در این دسته وظیفه دارند که پس از روشن شدن سیستم، پردازنده را آماده‌سازی کنند و سپس هسته سیستم‌عامل را به اجرا درآورند. این مرحله شامل تنظیمات اولیه CPU، آماده‌سازی حافظه، و شروع اجرای فرآیندهای پایه‌ای هسته است. در سیستم‌های چندپردازنده، فایل‌های مخصوص به راه‌اندازی پردازنده‌های اضافی نیز در این دسته قرار دارند.

#### 3. Locks (قفل‌ها)

فایل‌های این دسته به منظور همگام‌سازی (synchronization) بین بخش‌های مختلف سیستم عامل طراحی شده‌اند. قفل‌های چرخشی (spinlocks) و قفل‌های خواب (sleeplocks) به عنوان ابزارهای اصلی برای جلوگیری از دسترسی همزمان به منابع مشترک استفاده می‌شوند. این فایل‌ها ساختارهای داده و توابعی را فراهم می‌کنند که به هسته اجازه می‌دهد تا فرآیندهای چندگانه را به صورت همزمان اما کنترل‌شده مدیریت کند.

#### 4. Processes (فرآیندها)

این دسته از فایل‌ها مسئول مدیریت فرآیندها در سیستم عامل هستند. فایل‌های این دسته شامل تعریف و پیاده‌سازی ساختار فرآیندها، تخصیص و آزادسازی حافظه برای آن‌ها، و همچنین مدیریت وضعیت فرآیندها و سوییچ کردن بین فرآیندهای مختلف است. همچنین، این دسته شامل کدهای مربوط به زمان‌بندی (scheduling) فرآیندها و تعاملات آن‌ها با سیستم عامل است.

#### 5. System Calls (فراخوانی‌های سیستمی)

این دسته شامل فایل‌هایی است که پیاده‌سازی فراخوانی‌های سیستمی را بر عهده دارند. سیستم عامل از طریق این فراخوانی‌ها خدماتی مانند ایجاد فرآیندها، مدیریت حافظه، ارتباطات بین فرآیندها و مدیریت فایل‌ها را در اختیار برنامه‌های کاربری قرار می‌دهد. همچنین شامل مدیریت وقفه‌ها و استثنائات (interrupts & exceptions) و پیاده‌سازی توابعی است که به برنامه‌های سطح کاربر اجازه می‌دهد تا از خدمات هسته استفاده کنند.

#### 6. File System (سیستم فایل)

این فایل‌ها مربوط به پیاده‌سازی سیستم فایل در xv6 هستند. سیستم فایل شامل مدیریت دیسک، فایل‌ها و دایرکتوری‌ها، بافرهای دیسک، و sleeplock برای کنترل همزمانی در عملیات فایل است. علاوه بر این، این دسته شامل پیاده‌سازی عملیات خواندن و نوشتن فایل‌ها، مدیریت فایل‌ها در سطح هسته و اجرای برنامه‌ها از طریق فایل‌های اجرایی است.

#### 7. Pipes (لوله‌ها)

این دسته شامل پیاده‌سازی مکانیزم لوله‌ها (pipes) است که یک روش ارتباط بین فرآیندها (IPC) است. لوله‌ها به فرآیندها این امکان را می‌دهند که به طور همزمان از طریق یک جریان داده با هم ارتباط برقرار کنند. این فایل‌ها ارتباط داده‌ای بین فرآیندها را در سیستم مدیریت می‌کنند.

---

## 8. String Operations (عملیات‌های رشته‌ای)

این فایل‌ها شامل توابعی برای مدیریت و پردازش رشته‌ها (strings) هستند. این توابع شامل عملیات‌هایی مانند کپی کردن، مقایسه، و محاسبه طول رشته‌ها می‌شوند و در بخش‌های مختلف سیستم‌عامل استفاده می‌شوند.

## 9. Low-Level Hardware (سخت‌افزار سطح پایین)

این دسته شامل فایل‌هایی است که به سیستم اجازه می‌دهند تا با سخت‌افزار سطح پایین، مانند پردازنده‌های چندگانه و کنترل‌کننده‌های وقفه (APIC)، تعامل کند. این فایل‌ها مسئول پیاده‌سازی تعامل مستقیم با سخت‌افزار و مدیریت وقفه‌ها و منابع سخت‌افزاری هستند که برای عملکرد صحیح سیستم حیاتی‌اند.

## 10. User-Level (سطح کاربر)

این دسته شامل فایل‌هایی است که به تعامل بین هسته و برنامه‌های سطح کاربر مرتبط است. این فایل‌ها شامل پیاده‌سازی فراخوانی‌های سیستمی سطح کاربر، اجرای شل ساده (که دستورات کاربر را دریافت و اجرا می‌کند)، و کدهای اولیه‌ی سطح کاربر برای شروع فرآیندهای جدید است.

## 11. Bootloader (بوت‌لودر)

فایل‌های بوت‌لودر مسئول بارگذاری هسته سیستم‌عامل از دیسک به حافظه و آغاز اجرای آن هستند. این فایل‌ها شامل کدهای اسمبلی و C برای اجرای فرآیندهای اولیه‌ی بوت‌لودر و شروع به کار سیستم‌عامل بعد از روشن شدن سیستم می‌شوند.

## 12. Link (لینک‌دهی)

این فایل شامل اسکریپت لینک‌دهی است که ترتیب و نحوه‌ی چیدمان قسمت‌های مختلف حافظه در فایل اجرایی هسته را تعیین می‌کند. این فایل مشخص می‌کند که هر بخش از کد و داده‌ها در چه آدرسی قرار گیرد.

**پوشه‌های مشابه در لینوکس:**

## 1. فایل‌های هسته سیستم‌عامل (Kernel Core)

- مسیر: `/usr/src/linux/kernel/`

- محتویات: این بخش شامل کدهای اصلی مدیریت هسته سیستم‌عامل است. اعمال مهمی مانند مدیریت

فرآیندها، زمان‌بندی، مدیریت فراخوانی‌های سیستمی، و مدیریت حافظه در این دایرکتوری پیاده‌سازی

---

شده‌اند. فایل‌های موجود در این مسیر به نحوه‌ی عملکرد کلی هسته و تعاملات آن با سخت‌افزار سیستم و برنامه‌های کاربر مربوط می‌شوند.

## 2. فایل‌های سر ایند (Header Files)

- مسیر: `/usr/src/linux/include/`

- محتویات: این دایرکتوری شامل فایل‌های سر ایند است که تعاریفات ساختارهای داده، ثابت‌ها و پروتوتایپ توابع مورد استفاده در هسته لینوکس را فراهم می‌کنند. این فایل‌ها به توسعه‌دهندگان هسته اجازه می‌دهند تا کدهای خود را مطابق با استانداردهای سیستم عامل بنویسند. همچنین این پوشه شامل فایل‌های مربوط به معماری‌های مختلف برای CPUهای مختلف است که از طریق زیرشاخه‌های خاص در دسترس هستند.

## 3. فایل‌های سیستم‌فایل (File System)

- مسیر: `/usr/src/linux/fs/`

- محتویات: این دایرکتوری شامل کدهای مرتبط با سیستم‌فایل لینوکس است. در این دایرکتوری توابعی برای مدیریت فایل‌ها (مانند باز کردن، خواندن، نوشتن و بستن فایل‌ها) و داده‌ساختارهای ضروری برای مدیریت فایل‌ها تعریف شده‌اند. همچنین سیستم‌های فایل مختلف (مانند NFS، ext4، و FAT) در این مسیر پیاده‌سازی شده‌اند.

## ● کامپایل سیستم عامل xv6

**سوال 3) دستور `make -n` را اجرا نمایید. کدام دستور، فایل نهایی هسته را می‌سازد؟**

دستور `make -n` در سیستم‌های یونیکس و لینوکس، دستوری است که اجازه می‌دهد تا بررسی کنید که چه فرمان‌هایی قرار است توسط `make` اجرا شوند، بدون آنکه واقعاً آن‌ها را اجرا کند. این دستور نشان می‌دهد که اجرای `make` چه دستوراتی را در نظر دارد انجام دهد، اما آن دستورات را اجرا نمی‌کند. پس ما با زدن این دستور و خواندن `makefile` به این نتیجه رسیدیم که این بخش مسئول ساختن فایل نهایی هسته است:

```
ld -m elf_i386 -T kernel.ld -o kernel entry.o bio.o console.o exec.o file.o fs.o ide.o ioapic.o kalloc.o kbd.o lapic.o log.o main.o mp.o picirq.o pipe.o proc.o sleeplock.o spinlock.o string.o swtch.o syscall.o sysfile.o sysproc.o trapasm.o trap.o uart.o vectors.o vm.o -binary initcode entryother
```

این دستور هسته‌ی سیستم‌عامل (kernel) را می‌سازد زیرا از ابزار ld برای لینک کردن تمام فایل‌های o. مربوط به اجزای هسته (که قبلاً از طریق کامپایلر به صورت جداگانه کامپایل شده‌اند) استفاده می‌کند و خروجی نهایی آن یک فایل اجرایی به نام kernel است. فایل‌های o. شامل کدهای مختلف سیستم‌عامل هستند که با استفاده از اسکریپت kernel.ld ترکیب می‌شوند. خروجی این فرآیند، هسته‌ی نهایی است که توسط سیستم اجرا می‌شود.

**سوال 4) در Makefile متغیرهایی به نام‌های UPROGS و ULIB تعریف شده است. کاربرد آنها چیست؟**

UPROGS: این متغیر شامل لیستی از برنامه‌های فضای کاربر (user programs) است که در محیط کاربر اجرا می‌شوند. این برنامه‌ها از طریق کامپایل شدن به فایل‌های اجرایی تبدیل می‌شوند و به عنوان بخشی از سیستم در اختیار کاربر قرار می‌گیرند. در فایل Makefile، این برنامه‌ها معمولاً به صورت زیر تعریف شده‌اند:

```
UPROGS=\
_cat\
_echo\
_forktest\
_grep\
_init\
_kill\
_ln\
_ls\
_mkdir\
_rm\
_sh\
_stressfs\
_usertests\
_wc\
_zombie\
_history\
_encode\
_decode\
```

هر یک از این برنامه‌ها، به صورت یک فایل اجرایی در فضای کاربر (مانند cat, echo, sh) ایجاد می‌شود. علامت \_ در ابتدا، نشان‌دهنده‌ی فایل‌های میانی است که در نهایت به نام اصلی خود (بدون \_) تغییر نام می‌دهند.

ULIB: این متغیر به کتابخانه‌های فضای کاربر (user libraries) اشاره دارد که برای لینک کردن برنامه‌های کاربری از آن‌ها استفاده می‌شود. در Makefile، کتابخانه‌های کاربری مانند ulib.o, usys.o و سایر فایل‌های مورد نیاز برای اجرای برنامه‌های کاربر در این متغیر تعریف شده‌اند که بعداً توسط ld به فایل‌ها اضافه شده و در کدهای کرنل از این کتابخانه‌ها استفاده می‌شود:

```
ULIB = ulib.o usys.o printf.o umalloc.o
```

پس این object file شامل توابع کتابخانه‌ای و system call‌هایی هستند که برنامه‌های کاربر به آن‌ها نیاز دارند تا بتوانند با هسته تعامل داشته باشند.

## ● اجرا بر روی شبیه‌ساز QEMU

سوال (5) دستور make qemu -n را اجرا نمایید. دو دیسک به عنوان ورودی به شبیه‌ساز داده شده است. محتوای آن‌ها چیست؟ (راهنمایی: این دیسک‌ها حاوی سه خروجی اصلی فرایند بیلد هستند.)

## ● اجرای بوت‌لودر

سوال (8) علت استفاده از دستور objcopy در حین اجرای عملیات make چیست؟

سوال (13) کد bootmain.c هسته را با شروع از سکتور بعد از سکتور بوت خوانده و در آدرس 0x100000 قرار می‌دهد. علت انتخاب این آدرس چیست؟

## ● اجرای هسته xv6

سوال (18) علاوه بر صفحه‌بندی در حد ابتدایی از قطعه‌بندی به منظور حفاظت هسته استفاده خواهد شد. این عملیات توسط seginit() انجام می‌گردد. همانطور که ذکر شد، ترجمه قطعه تأثیری بر ترجمه آدرس



---

منطقی نمی‌گذارد. زیرا تمامی قطعه‌ها اعم از کد و داده روی یکدیگر می‌افتند. با این حال برای کد و داده‌های سطح کاربر پرچم SEG\_USER تنظیم شده است. چرا؟ (راهنمایی: علت مربوط به ماهیت دستورالعمل‌ها و نه آدرس است.)

● اجرای نخستین برنامه سطح کاربر

سوال 19) جهت نگهداری اطلاعات مدیریتی برنامه‌های سطح کاربر ساختاری تحت عنوان `struct proc` (خط ۲۳۳۶) ارائه شده است. اجزای آن را توضیح داده و ساختار معادل آن در سیستم عامل لینوکس را بیابید.

سوال 23) کدام بخش از آماده‌سازی سیستم، بین تمامی هسته‌های پردازنده مشترک و کدام بخش اختصاصی است؟ (از هر کدام یک مورد را با ذکر دلیل توضیح دهید.) زمان‌بند روی کدام هسته اجرا می‌شود؟

---