دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

# پروژه‌ی اول

دکتر جوادی

سیستم‌های عامل

# پروژه‌ی اول

## مقدمه

همانطور که در کلاس درس بیان شد، پروژه درس سیستم‌های عامل مربوط به شناخت کامل سیستم عامل آموزشی xv6 و افزودن قابلیت‌های جدید به آن است. با انجام دقیق این پروژه و تطبیق مفاهیمی که در کلاس درس معرفی شده است به یادگیری عمیق و دائمی شما کمک خواهد کرد. این نیز مهم است که برنامه‌نویسی در سطح سیستم عامل به شما کمک می‌کند تا تجربه‌ای منحصر به فرد در برنامه‌نویسی سیستمی کسب کنید.

هدف از دو فاز ابتدایی این پروژه آشنایی شما دانشجویان با این سیستم عامل آزمایشی است و روند کار به این صورت است که در فاز اول به صورت فردی و در فاز دوم به صورت گروهی باید تغییراتی در این سیستم عامل ایجاد کنید تا بتوانید درک خوبی از این سیستم عامل را بدست آورید. در فاز سوم نیز از شما درخواست خواهیم کرد که به گروه‌های دو نفره تقسیم شوید و تغییراتی که در این فاز انجام خواهید داد، بسیار مهم و جدی خواهند بود. موضوع این قسمت در زمان مناسب به شما اعلام خواهد شد. در فاز اول پروژه، بایستی درک مناسبی از برخی مفاهیم از جمله پردازه ها، سیستم کال‌ها و عملکرد کلی سیستم عامل را پیدا کنید و در نهایت دو سیستم کال ساده را به سیستم عامل خود اضافه کنید.

در این ترم، ما با آخرین نسخه از سیستم عامل xv6 که بر اساس معماری پردازنده RISC-V توسعه یافته است کار خواهیم کرد. برای آشنایی با این سیستم عامل، می‌توانید از لینک زیر استفاده کنید:

https://pdos.csail.mit.edu/6.828/2022/xv6.html

برای اجرای این سیستم عامل می توانید مراحل و پیش نیازهای build کردن را از لینک زیر دریافت کنید.

https://pdos.csail.mit.edu/6.828/2022/tools.html

توصیه ما استفاده از docker container میباشد به این صورت که image محیط اجرایی پردازنده xv6 را در قالب یک container اجرا میکنیم و در ادامه هم نحوه اجرای آن را توضیح دادیم ولی در صورتی که به هر علتی مایل به استفاده از این روش نیستید. میتوانید این سیستم عامل را با پیروی از دستورات بالا راه اندازی کنید.

# پروژه‌ی اول

مراحل اجرای xv6 image از طریق Docker

۱. نصب و اجرای Docker

۱-۱. با مراجعه به این آدرس برنامه Docker Desktop را برای سیستم عامل خود نصب کنید.

۱-۱-۱ . در صورتی که هنگام مراجعه به وب سایت و یا مراحل دانلود با خطاهای مرتبط با تحریم برخورد کردید از شکن برای عبور از تحریم استفاده کنید. در صورتی که همچنان در این مراحل مشکل داشتید با تیم تدریس یاری در میان بگذارید.

۱-۲ . برنامه Docker Desktop را اجرا و از فعال شدن Docker Engine داخل آن اطمینان حاصل کنید.

۲. دریافت Docker Image

۲-۱. با استفاده از دستور docker pull wtakuo/xv6-env، image مربوطه را دانلود کنید

۲-۱-۱ . در صورتی که در اجرای دستور بالا با خطاهای مربوط به تحریم روبه رو شدید به جای دستور بالا از این دستور استفاده کنید:

m.docker-registry.ir/wtakuo/xv6-env

۳. اجرای محیط Container جهت کار با سیستم عامل

۳-۱ . کد های مربوط به سیستم را در یک پوشه فرضا ((xv6-riscv ذخیره کنید. داخل این پوشه دستور زیر را اجرا کنید.

*docker run -it --rm -v $(pwd):/home/xv6/xv6-riscv wtakuo/xv6-env*

چنانچه بر روی ویندوز کار می کنید از این دستور استفاده کنید:

wtakuo/xv6-env "cd%:/home/xv6/xv6-riscv/." docker run -it --rm -v

در صورتی که همه مراحل را به درستی طی کرده باشید با پیغام زیر مواجه می شوید:

۳-۲ . در این محیط میتوانید دستورات لازم جهت بیلد و اجرای سیستم عامل را فراخوانی کنید.

با استفاده از دستور make clean فایل های کامپایل شده را پاک کنید.

با استفاده از دستور make فایل های موجود را کامپایل کنید.

با استفاده از دستور make qemu فایل های کامپایل شده را اجرا کنید تا سیستم عامل مربوط به کدی که نوشته اید اجرا شود.

## فاز اول

### بررسی فرایند اجرای سیستم عامل

در ابتدا می خواهیم نحوه عملکرد و اجرای سیستم عامل و لود شدن اولین پردازه را بررسی کنیم. برای این منظور ابتدا فایل main.c سیستم عامل را بررسی کنید. سپس مراحلی که برای ساخته شدن (نه اجرا) اولین پردازه سیستم عامل طی میشود را بررسی کنید. برای راحتی و درک بهتر. میتوانید از ابزار GDB و قرار دادن Breakpoint برای تابع userinit استفاده کنید. (میتوانید به ویدیوهای آموزشی قرار داده شده مراجعه کنید)

برای توضیحات تکمیلی و مطالعه بیشتر در مورد کدها و آشنایی بیشتر با قسمت های مختلف سیستم عامل می توانید از لینک های زیر استفاده کنید:

- XV6 BOOK
- Learn OS with Me
- Google

# پروژه‌ی اول

پیاده سازی دو سیستم کال

در این قسمت می خواهیم دو سیستم کال به سیستم عامل XV6 اضافه کنیم:

## 1- int history(int historyId):

تاریخچه دستورات گذشته به کاربران ترمینال اجازه می‌دهد که به سرعت چندین درخواست را ارزیابی کنند، بدون اینکه کل دستور را نوشته باشند. در این بخش از وظایف، شما باید ویژگی تاریخچه و قابلیت به‌روزرسانی آسان کنسول را برای تاریخچه مورد نیاز پیاده‌سازی کنید. در سیستم‌های عامل مدرن، تاریخچه در شل پیاده‌سازی می‌شود. به منظور پیاده‌سازی ساده‌تر، شما باید تاریخچه را در کرنل پیاده‌سازی کنید.

پیاده‌سازی شما باید حداکثر ۱۶ دستور را پشتیبانی کند. برای انجام این کار،

"define MAX_HISTORY 16" را در فایل console.h اضافه کنید. پس از پیاده‌سازی تاریخچه، نیاز داریم که یک روش برای دسترسی به تاریخچه داشته باشیم  از این رو از سیستم کال history استفاده میکنیم. ورودی historyId است که  کد دستوری است که اجرا شده.

برای پیاده سازی راحت تر از historyBufferArray struct استفاده کنید و این struct را به فایل c.console اضافه کنید:

```
struct {
  char bufferArr[MAX_HISTORY][INPUT_BUF]; // holds the actual command strings -
  uint lengthsArr[MAX_HISTORY]; // this will hold the length of each command string
  uint lastCommandIndex;  // the index of the last command entered to history
  int numOfCommmandsInMem; // number of history commands in mem
  int currentHistory; // holds the current history view -> displacement from the last command index
} historyBufferArray;
```

این Struct اطلاعات لازم برای پیدا کردن  کامند های اجرا شده و نمایش  آن ها در کنسول را در اختیار شما قرار می دهد و با استفاده از آن می توانید فراخوانی سیستمی را پیاده سازی کنید.

# پروژه‌ی اول

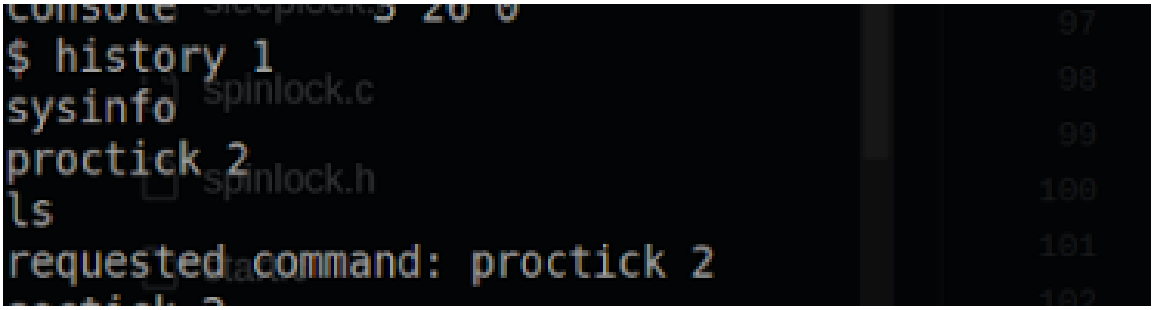
برای پیاده سازی ابتدا با کمک historyId چک کنید که کامندی که از historyBufferArray خوانده می شود valid باشد و در محدوده تعریف شده برای کامند ها باشد. سیستم عامل xv6 تا ۱۶ کامند را میتواند در بافر ذخیره کند.

همچنین با استفاده از این شناسه و اطلاعاتی که در استراکتی که تعریف کرده اید، می توانید کامند را از historyBufferArray پیدا کنید لازم است که در ابتدای کار buffer خالی باشد و دیتایی از قبل در آن نباشد تا از ناسازگاری کامند ها جلوگیری شود. بعد از پیدا کردن کامند ها آن ها را از historyBufferArray بخوانید و چاپ کنید.

در نتیجه شما باید یک فراخوانی سیستمی پیاده سازی کنید که تاریخچه ای از دستورات اجرا شده را در کنسول نمایش دهد و در هر بار اجرای این فراخوانی لیستی که در کنسول نمایش داده می شود بروز رسانی شود.

دقت کنید که خود دستور history نباید در تاریخچه ذخیره شود. برای پیاده سازی لازم است یک تابع برای ذخیره کامند وارد شده بنویسید و آن را در جای مناسب در تابع consoleintr صدا کنید. ذخیره سازی در تاریخچه از خانه ی شماره ی صفر شروع می شود و با پر شدن آرایه به صورت چرخشی آرایه پر می شود.

نمونه ای از خروجی :



# پروژه‌ی اول

## بخش امتیازی:

در این بخش باید برای دسترسی به تاریخچه از کلید های ↑ / ↓ استفاده کنید. دقت کنید با فشار دادن کلید بالا به آخرین کامند وارد شده دسترسی پیدا می شود و کامند نوشته شده در کنسول پاک می شود و بافر ورودی خالی می شود. سپس کامند خوانده شده از تاریخچه در کنسول برای نمایش چاپ می شود و بافر ورودی با این کامند پر می شود. کلید بالا برای حرکت به عقب در آرایه است و کلید پایین برای حرکت به جلو در آرایه استفاده می شود.

## 2- top command:

دستور top یک دستور پایه ای یونیکس است که با استفاده از این دستور می توان لیستی از فرایندهای فعال را که در حال حاضر در حال اجرا هستند را نمایش داد. هدف از این قسمت گنجاندن دستور top در محیط XV6، مشابه آنچه در لینوکس است، می باشد. این کار با توسعه یک برنامه کاربر که شامل فراخوانی های سیستمی برای نمایش لیست فرآیندها، کاربران منابع سیستم و اطلاعات خلاصه سیستم است، انجام می شود.

حال در این قسمت می خواهیم نمونه ی ساده شده ای از این دستور را در سیستم عامل xv6 پیاده سازی کنیم.

پیاده سازی اصلی این فراخوانی باید در فایل proc.c گنجانده شود. این فراخوانی روی جدول فرایندها یعنی ptable پیمایش میکند و شناسه، نام فرآیند، و وضعیت فرآیندها را دریافت می کند.

برای پیاده سازی این بخش لازم است یک استراکت جدید به نام top تعریف کنید.

سیستم کال ما باید یک پوینتر به یک struct top بگیرد، اطلاعات درون این ساختار را کامل کند (مشخصا این قسمت به صورت call by reference است) و در صورت عدم خطا مقدار صفر و در صورت وجود خطا مقدار منفی یک را برگرداند. ساختار top struct به صورت زیر است و شما باید آن را در فایل مناسب تعریف کنید.

# پروژه‌ی اول

```
struct proc_info {
  char name[16];
  int pid;
  int ppid;
  enum procstate state;
};
```

```
struct top {
  long uptime;
  int total_process;
  int running_process;
  int sleeping_process;
  struct proc_info p_list[NPROC];
};
```

با استفاده از این فراخوانی اطلاعات زیر با موفقیت نمایش داده می شود:

- تعداد کل پردازه ها
- تعداد کل پردازه های در حال اجرا.
- تعداد کل پردازه های sleeping
- لیستی از اطلاعات پردازه ها به صورت زیر:

شناسه (pid)، شناسه ی پردازه مادر، وضعیت فرآیند، نام فرآیند مربوطه.

وضعیت فرآیندها (running, sleeping, runnable, unused)

```
$ testTop
uptime:48 seconds
total process:35
running process:3
sleeping process:2
process data:
name     PID     PPID     state
init     1     0     sleeping
sh     2     1     sleeping
testTop     36     2     running
testProcinfo     4     1     runnable
```

# پروژه‌ی اول

## مراحل اضافه کردن یک سیستم کال در سیستم عامل xv6

بعد از پیاده سازی فراخوانی های سیستمی گفته شده لازم است که آن ها را به xv6 اضافه کنید که در ادامه مراحل اضافه کردن یک فراخوانی سیستمی را توضیح داده ایم :

- در صورت نیاز اضافه کردن struct جدید در هدر فایل مربوطه
- در صورت تعریف یک تایپ جدید یا یک تابع جدید که قرار است در فایل های دیگر مورد استفاده قرار گیرد، تعریف آن را در فایل defs.h مانند نمونه های قبلی قرار دهید. در کل هر تایپ یا تابعی که در فایل دیگری تعریف شده و لازم است در فایل های دیگر مورد استفاده قرار گیرد باید پروتوتایپ آن در این فایل قرار بگیرد.
- در فایل syscall.h نام سیستم کال جدید را با یک شماره ی جدید اضافه کنید.
- در فایل syscall.c نام تابع سیستم کال مورد نظر را به نامی که در فایل syscall.h قرار دادید مپ کنید.
- پروتوتایپ تابع مورد نظر را نیز در ابتدای فایل syscall.c اضافه کنید. ("extern " function definition)
- پیاده سازی تابع سیستم کال مورد نظر را با نامی که در مپ قرار دادید در فایل sysproc.c قرار دهید. در این بخش اصولا بخش انتقال و تبدیل پارامتر ها انجام می شود و در فایل proc.c پیاده سازی اصلی سیستم کال انجام می شود.
- برای اینکه برنامه های کاربر بتوانند از این سیستم کال ها استفاده کنند لازم است در فایل usys.pl سیستم کال مربوط را اضافه کنید.
- پروتوتایپ تابع مورد نظر را در فایل user.h اضافه کنید.

در نهایت Makefile را تغییر دهید و اسم فایل برنامه ی کاربری که نوشتید را در بخش UPROGS

اضافه کنید.


# پروژه‌ی اول

از شما درخواست داریم که یک **private repository** در گیت هاب درست کنید و تغییرات کد خود را مرحله به مرحله **Commit** کنید و در صورت تمایل می توانید هر یک از تدریس یاران را به پروژه ی خود اضافه کنید. دقت کنید که شما نبایستی برنام ههای خود را با دیگر دانشجویان به اشتراک بگذارید.

## توضیحات

- پروژه شما تحویل آنلاین خواهد داشت بنابراین از استفاده از کدهای یکدیگر یا کدهای موجود در وب که قادر به توضیح دادن عملکرد آنها نیستید، بپرهیزید.
- ابهامات خود را در گروه درس در تلگرام مطرح کنید و ما در سریعترین زمان ممکن به آنها پاسخ خواهیم داد .

## آنچه که باید ارسال کنید:

یک فایل زیپ با نام OS_P1_Sid.zip (که Sid را با شماره دانشجویی خود جایگزین کنید) که شامل دو مورد زیر است:

- پوشه ای که در آن کدهای شما وجود دارد. دقت کنید که تنها و تنها فایل هایی را که تغییر داده اید یا اضافه کرده اید را برای ما بفرستید.


# پروژه‌ی اول

موفق باشید

تیم تدریس یاری درس سیستم های عامل