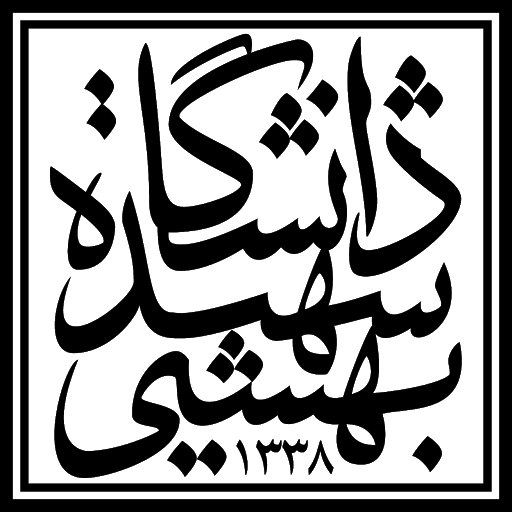
****

**گزارش اول آزمایشگاه سیستم عامل**

استاد : دکتر شهاب‌الدین نبوی

اعضای گروه :

محمد خدام

حامد خادمی خالدی

امیر حلاجی بیدگلی

متین زیودار

بهار ۱۴۰۰

[مقدمه](#_Toc69878648)

[حافظه thread](#_Toc69878649)

[بخش‌های heap](#_Toc69878650)

[نکات اولیه](#_Toc69878651)

[مکان حافظه داینامیک](#_Toc69878652)

[دیتا استراکچر حافظه داینامیک](#_Toc69878653)

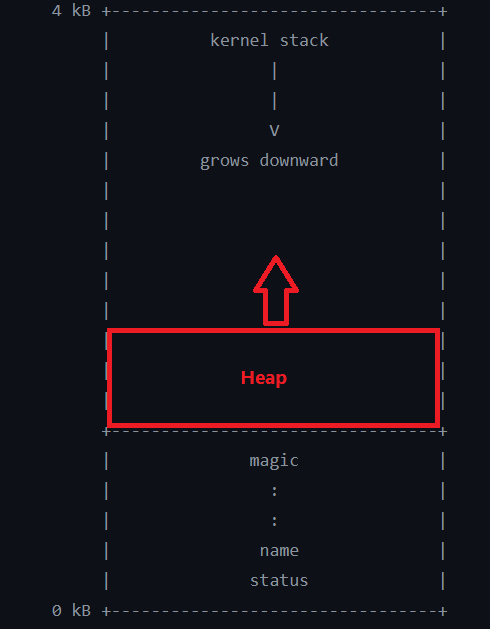
[توابع](#_Toc69878654)

[منابع](#_Toc69878655)

# مقدمه

هدف از این پروژه پیاده سازی حافظه پویا برای هر ترد در سیستم عامل pintos می‌باشد. برای پیاده سازی این نیاز به توابع مختلفی نیاز است که در ادامه به بررسی کد تک تک ‌آن ها پرداخته و آن‌ها را شرح می‌دهیم

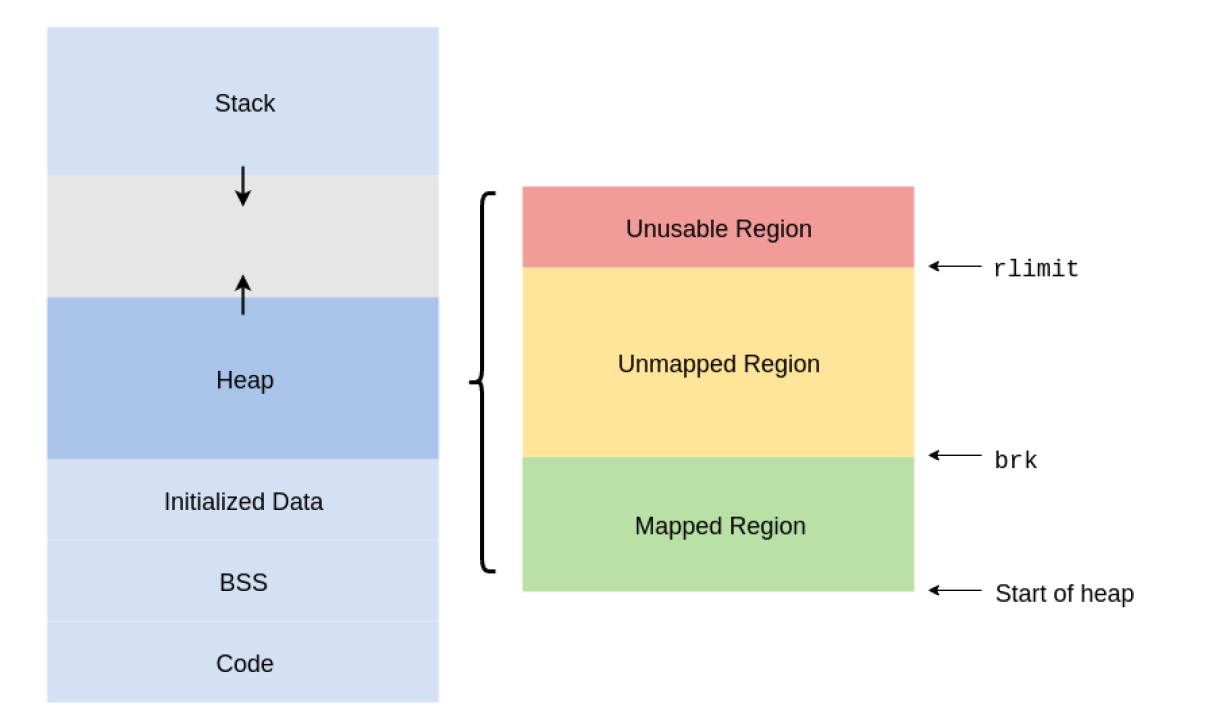
# حافظه thread

هر thread یا process در پینتوس حافظه اختصاصی خود را دارد که شکل آن به صورت زیر است ( هدف ما پیاده سازی بخش Heap است)

# بخش‌های heap

با توجه به شکل زیر بخش heap از سه بخش تشکیل شده :

* بخش سبز : بخشی ک در اختیار heap است
* بخش زرد : heap برای دسترسی به آن باید از سیستم عامل اجازه بگیرد ( پوینتر brk می تواند جابجا شود)
* بخش قرمز : این بخش برای زمانی است که نمی‌خواهیم heap از آن بالاتر رود ( در این پروژه این بخش پیاده سازی نشده است)



# نکات اولیه

## مکان حافظه داینامیک

در مورد شروع حافظه داینامیک لازم است این بخش با بخش دیگری تداخل نداشته باشد و همان طور که قبلا نشان داده شد باید بالای اطلاعات هر ترد قرار گیرد (دقیقا بالای magic)

همچنین این حافظه تا جایی پیش می‌رود که با استک تداخلی نداشته باشد (این فرض برای استک نیز باید چک شود)

## دیتا استراکچر حافظه داینامیک

هر بلاک از حافظه پویا شامل دو بخش دیتا و اطلاعات آن بلاک است :

* + بلاک دیتا :
    - شامل دیتای آن بلاک است
  + اطلاعات (metadata)
    - size : سایز بلاک را مشخص می‌کند
    - Free :خالی یا پر بودن بلاک را مشخص می‌کند
    - Next : یک پوینتر به خانه بعدی می‌باشد
    - Prev : یک پوینتر به خانه قبلی می باشد

\* همچنین یک پوینتر اشاره کننده به بلاک نیز نگه می‌داریم

# توابع

* تابع extend heap :  
  این تابع در صورت امکان یک بلاک به انتهای heap اضافه می‌کند و در غیر این صورت null بر می‌گرداند  
  برای پیاده سازی این تابع از سیستم کال sbrk استفاده می‌شود تا چک کند که آیا می‌توان heap را گسترش داد یا خیر
* تابع get block :  
  این تابع یک آدرس گرفته و اگر این آدرس به ابتدای یک بلاک اشاره کند آن بلاک را بر‌می‌گرداند و در غیر این صورت null باز می‌گرداند
* تابع split block :  
  یک اشاره گر به یک بلاک و یک سایز از ورودی گرفته و بلاک مد نظر را به اندازه آن تقسیم می‌کند   
  لازم اندازه این سایز یک حداقلی داشته باشد و همچنان از اندازه بلاک بزرگتر نباشد و بعد از چک کردن به سراغ درست کردن ارتباطات بین بلاک‌های جدید می‌رویم
* تابع Fusion :  
  یک اشاره‌گر به یک بلاک را گرفته و اگر بتوان آن را با هریک از همسایه‌ها ادغام کرد آن را ادغام می‌کند (‌لازم است ارتباط بین بلاک ‌ها - next and prev - نیز درست شوند)
* تابع memcpy:

این تابع، دو آرگومان دارد که محتویات بلاک قدیمی را در بلاک جدید، کپی می‌کند. در زمان‌هایی که بخواهیم از حافظه بزرگتری استفاده کنیم و آن را نداشته باشیم، پس از ایجاد کردن حافظه جدید، محتویات آن را با استفاده از این تابع، کپی می‌کنیم.

* تابع realloc:

این تابع، دو پارامتر دارد که یکی از آن‌ها، اشاره‌گری به یک بلاک و دیگری، اندازه بلاک است. کاری که می‌کند این است که سایز بلاک را به اندازه سایز دلخواه تغییر بدهد. یک نمونه استفاده از تابع memcpy، میتواند این باشد که ابتدا با تابع realloc، یک حافظه بزرگتر بخواهیم و سپس memcpy کنیم.

* تابع malloc:

این تابع در ورودی یک سایز می‌گیرد و به دنبال یافتن اولین خانه‌ در حافظه است که خالی باشد و حداقل اندازه مورد نظر را نیز داشته باشد.

* تابع memfree:

همانطور که از اسم این تابع مشخص است، این تابع برا آزادسازی است. اینگونه است که یک اشاره‌گر در ورودی می‌گیرد و بلاکی که این اشاره‌گر به آن اشاره می‌کند را، آزادسازی می‌کند. لازم به ذکر است علاوه بر اینکه اشاره‌گر های next و prev را null می‌کنیم، باید تابع fusion را نیز استفاده کنیم که نسبت به ادغام دو فضایی که اکنون از هم جدا هستند، اقدام کند.

* تابع new\_op:

این تابع، یک سایز به عنوان ورودی می‌گیرد و به اندازه آن، malloc می‌کند و در نهایت آدرس آن را بر می‌گرداند.

* تابع delete\_op:

این تابع، یک اشاره‌گر به عنوان ورودی می‌گیرد و memfree را روی آن صدا می‌زند.

## 

# منابع

<https://inst.eecs.berkeley.edu/~cs162/fa19/static/hw/hw5.pdf>

<https://github.com/thecuongthehieu/CS-162-Operating-Systems-and-System-Programming-Homework>