

به نام هستی بخش
سیستم های عامل
نیم سال اول ۱۴۰۴-۱۴۰۵



مدرس: دکتر ابراهیمی مقدم
تاریخ تحویل:

دانشکده‌ی مهندسی و علوم کامپیوتر
پروژه‌ی پایانی
طراحان: سید صدرا موسوی، فرزین حاج‌غلامرضایی بهادرانی،
مهدی رضایی

در این پروژه قصد داریم تاثیر اندازه‌ی پیج را بر روی تعداد پیج‌فالت‌ها (page fault) و همچنین نسخه‌ی ساده‌سازی شده‌ای از ۲ الگوریتم LRU و Second chance را بررسی کنیم. برای انجام این کار ابتدا باید نحوه‌ی کلی کارکرد سیستم را بررسی کنیم و بعد فایل‌های مربوط به پروژه‌ی اولیه را بررسی کنیم و درنهایت خواسته‌های اصلی پروژه را بررسی کنیم.

نحوه‌ی کارکرد کلی سیستم

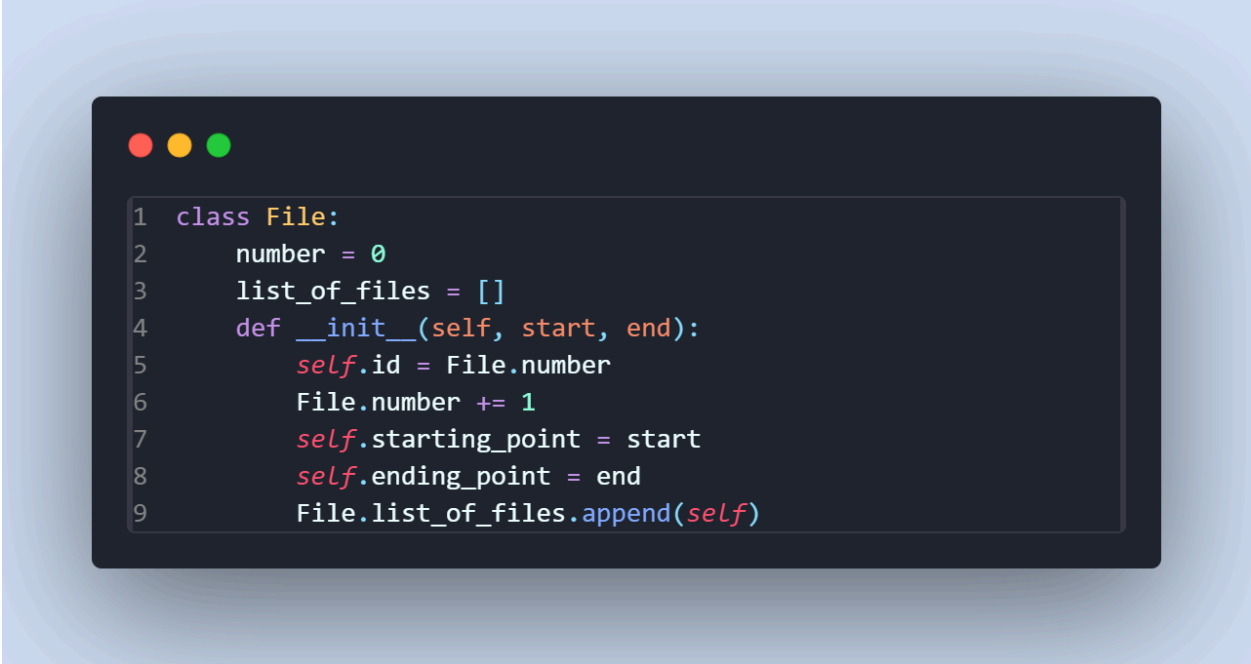
به صورت کلی page fault زمانی رخ می‌دهد که یک پراسس بخواهد به پیجی دسترسی پیدا کند که در حال حاضر در حافظه‌ی ما حضور ندارد. در سیستم‌عامل‌های امروزی در هنگام پیج‌فالت، علاوه‌بر پیجی که مورد نظر است تعدادی از پیج‌های نزدیک به آن پیج هم به حافظه انتقال داده می‌شوند؛ زیرا در ادامه ممکن است پراسس مورد نظر به آن پیج‌ها هم نیاز داشته باشد و بخواهد با آن پیج‌ها نیز کار کند. همچنین در اکثریت سیستم‌عامل‌های امروز اندازه‌ی پیج‌های ما 4096 بایت یا 4KB است. این یعنی اگر فرض کنیم یک سیستم‌عامل بعد اتفاق افتادن یک page fault، علاوه بر پیج مورد نظر پیج‌های بعدی و قبلی این پراسس را هم به حافظه منتقل کند، ما در هر بار انتقال داده‌ها از دیسک به حافظه (یا به اصطلاح swap in کردن) حدود 12KB را به حافظه‌ی اصلی خودمان منتقل می‌کنیم. کاری که در نظر داریم در این پروژه انجام دهیم این است که چه اتفاقی می‌افتد اگر اندازه‌ی پیج‌های ما بجای 4KB، بیشتر یا کمتر باشد. حال که با کلیت پروژه آشنا شدیم به سراغ فایل‌های پروژه‌ی اولیه خواهیم رفت.

بررسی فایل‌های پروژه‌ی اولیه

برای اینکه شما بهتر بتوانید با پروژه ارتباط برقرار کنید و نیاز به پیاده‌سازی بعضی موارد نداشته باشید، پروژه‌ی اولیه‌ای در اختیار شما قرار گرفته است که در آن بخش‌هایی از سیستم پیاده‌سازی شده است که در ادامه به بررسی این پیاده‌سازی‌ها می‌پردازیم:

[File.py](#):

در این فایل، کلاس File قرار دارد. فایل‌ها در این سیستم نماد داده‌هایی هستند که در دیسک قرار دارند. این کلاس ۲ فیلد استاتیک دارد که یکی برای شماره‌گذاری فایل‌هاست و دیگری لیستی از همه‌ی فایل‌هاست. کد این بخش در پایین قرار داده شده است.



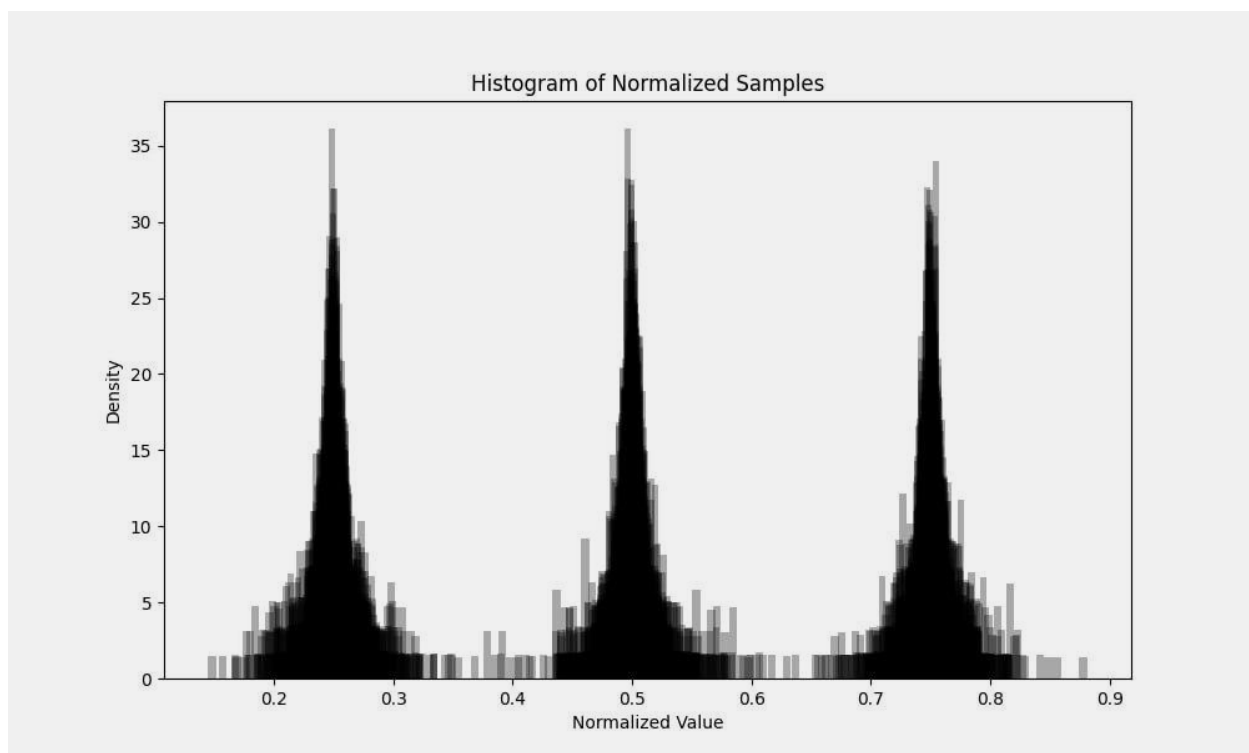
```
1 class File:
2     number = 0
3     list_of_files = []
4     def __init__(self, start, end):
5         self.id = File.number
6         File.number += 1
7         self.starting_point = start
8         self.ending_point = end
9         File.list_of_files.append(self)
```

[CPU.py](#):

در این فایل، کلاس CPU قرار دارد. این کلاس 1 فیلد دارد که در آن لیست تردهایی نگهداری می‌شود که در این سی‌پی‌یو قرار دارد. همچنین ۳ تابع که AddThread برای اضافه کردن تردها به سی‌پی‌یو است و start شبیه‌سازی‌ای از اجرا شدن تردها در سیستم را پیاده‌سازی می‌کند.

این متد پیاده‌سازی‌ای شبیه به الگوریتم Round Robin دارد. با یک اندازه‌ی پیچ مشخص، ۵۰ بار همه‌ی تردهای موجود در این پردازنده را اجرا می‌کند. همچنین reset هم برای بازگرداندن وضعیت تردهای این سی‌پی‌یو به وضعیت اولیه است. این متد هر بار که به آخرین مرحله‌ی اجرا رسیدیم باید فراخوانی شود (منظور از آخرین مرحله‌ی اجرا زمانی است که اجراهای شما با یک اندازه‌ی پیچ‌سایز مشخص تمام شده است و می‌خواهید به سراغ پیچ‌سایز بعدی بروید). کد این بخش هم این پایین برای شما قرار داده شده است.

```
1 from Thread import Thread
2 from MMU import MMU
3 import time
4
5 class CPU:
6     def __init__(self):
7         self.list_thread = []
8
9     def AddThread(self, Thread):
10        self.list_thread.append(Thread)
11
12    def reset(self):
13        for t in self.list_thread:
14            t.reset()
15
16    def run(self):
17        for i in range(50):
18            for i in range(len(self.list_thread)):
19                selected_thread = self.list_thread[i]
20
21                MMU.handle_request(selected_thread.start())
```



[:Thread.py](#)

در این فایل، کلاس Thread پیاده‌سازی شده است. این کلاس از ۳ فیلد غیر استاتیک:

- (1) id: مشخص‌کننده‌ی ترد مورد نظر است.
- (2) files: همه‌ی فایل‌هایی که این ترد از آن‌ها استفاده می‌کند. استفاده کردن در اینجا به معنای استفاده‌ی واقعی نیست. صرفاً به این معنی است که قرار است پیچ‌های این فایل‌ها دسترسی زده شود.
- (3) file_weights: علت حضور این فیلد جلوتر توضیح داده شده است.

و از ۲ فیلد استاتیک:

- (1) number: برای شماره‌گذاری تردها.
- (2) MAXIMUM_NUMBER_OF_FILES: مشخص می‌کند هر ترد حداکثر چند فایل می‌تواند داشته باشد.

تشکیل شده است. در ابتدای ساخته شدن هر ترد، تعداد فایل‌هایی که هر ترد استفاده می‌کند به صورت یک عدد رندوم از بازه‌ی ۱ تا MAXIMUM_NUMBER_OF_FILES انتخاب می‌شود. سپس مشخص می‌شود که هر ترد دقیقاً از کدام فایل‌ها استفاده می‌کند و با انتخاب یک فایل، آن فایل از لیست کلی فایل‌ها در کلاس فایل حذف می‌شود که این یعنی **بین فایل‌هایی که هر دو ترد مختلف استفاده می‌کنند اشتراکی وجود ندارد**. کد این بخش از کلاس ترد پایین قرار داده شده است.

```
1 import random
2 from File import File
3 import numpy as np
4 from MMU import MMU
5 import math
6 import time
7 class Thread:
8     number = 0
9     MAXIMUM_NUMBER_OF_FILES = 4
10    def __init__(self):
11        self.id = Thread.number
12        Thread.number += 1
13        self.files = []
14        self.file_weights = []
15        temp = random.randrange(1, Thread.MAXIMUM_NUMBER_OF_FILES)
16        for i in range(temp):
17            selected = random.randint(0, len(File.list_of_files))
18            chosen = File.list_of_files.pop(selected)
19            self.files.append(chosen)
20            self.file_weights.append(1)
21            File.list_of_files.remove(File.list_of_files[selected])
22
```

همچنین در کلاس ترد، تابعی با نام start وجود دارد. این تابع نماد اجرا شدن هر ترد است. در اولین باری که ترد اجرا می‌شود وزن همه‌ی فایل‌ها با هم برابر است و یکی از فایل‌ها به صورت رندوم با شانس برابر انتخاب می‌شود. پس از انتخاب شدن این فایل، وزن این فایل بیشتر می‌شود و از دفعات بعد این فایل با احتمال بیشتری ممکن است انتخاب شود که این یعنی به صورت نمادین نشان‌دهنده‌ی این است که پراسس شما در مرحله‌ی بعد با احتمال بیشتری ممکن است کار قبلی خود را انجام بدهد. برای انتخاب یک پوزیشن از فایل انتخاب شده اصل لوکالیتی باید رعایت شود و این اصل با استفاده از تابع generate_random_file_access انجام شده است و نیازی به پیاده‌سازی آن توسط شما وجود ندارد. اصل لوکالیتی با استفاده از توزیع نرمال پیاده‌سازی حول چارک‌ها پیاده‌سازی شده است. خروجی این تابع به صورت یک تاپل خواهد بود که عنصر اول آن خود فایل و عنصر دوم آن پوزیشن مورد نظر از آن فایل است.

```

1  def Start(self):
2      if not self.files:
3          return
4
5      # Choose a file index based on weights
6      selected_index = random.choices(range(len(self.files)), weights=self.file_weights, k=1)[0]
7      file = self.files[selected_index]
8
9      # Increase weight to favor future selections
10     self.file_weights[selected_index] += 1
11
12     return (file, Thread.generate_random_file_access(file.starting_point, file.ending_point))
13
14 def generate_random_file_access(a, b):
15     mean = np.random.choice([1, 2, 3]) * (b - a) / 4 + a
16     std_dev = math.log2(b - a)
17     return int(np.clip(np.random.normal(mean, std_dev), a, b))
18

```

[:MMU.py](#)

در این فایل، کلاس MMU قرار دارد. هسته‌ی اصلی پروژه‌ی شما حول این کلاس خواهد بود. این کلاس در حال حاضر از ۳ فیلد استاتیک:

(1) page_faults: این فیلد در حال حاضر تعداد پیچ‌فالت‌ها را نگه داری می‌کند.

(2) page_size: این فیلد اندازه‌ی پیچ‌ها را مشخص می‌کند.

(3) mem_size: اندازه‌ی مموری کلی را مشخص می‌کند.

و ۲ فیلد غیر استاتیک:

(1) mmap: مموری مپ. در واقع ما به صورت مستقیم با مموری کار نمی‌کنیم بلکه با

پیچ‌هایی که در مموری هستند کار می‌کنیم و با استفاده از این مموری مپ، متوجه

می‌شویم کدام پیچ‌ها پر و کدام پیچ‌ها خالی هستند.

(2) free_places: تعداد مکان‌هایی که در حال حاضر خالی هستند.

تشکیل شده است. این کلاس به صورت کلی باید توسط شما پیاده‌سازی و تکمیل شود. اضافه کردن هر نوع تابع و فیلد به این کلاس بلامانع است. همچنین امضای برخی توابع که ممکن

است نیاز باشد که پیاده‌سازی کنید قرار داده شده است. در پایین کد این بخش قرار داده شده است.

```
1 class MMU:
2     page_faults = 0
3     page_size = 4
4     mem_size = 4000
5     def __init__(self):
6         self.mmap = [(-1, -1)] * (MMU.mem_size // MMU.page_size)
7         self.free_places = (MMU.mem_size // MMU.page_size)
8
9     def reset(self):
10        # TO DO
11        pass
12
13    def handle_request(self, request):
14        # TO DO
15        pass
16
17
18    def eviction_system(self):
19        # TO DO
20        pass
21
22    def search(self, request):
23        # TO DO
24        pass
25
```

مواردی که باید پیاده‌سازی کنید.

1) MMU: کلاس MMU را به گونه‌ای پیاده‌سازی کنید تا تمام فرآیندهای مربوط به حافظه را به صورت کلی انجام دهد. هرگونه پیاده‌سازی مواردی که مربوط به حافظه می‌شود در جایی غیر از کلاس MMU نمره‌ی منفی به همراه خواهد داشت. همچنین توجه داشته باشید که در هنگام پیچ‌فالت خوردن، علاوه بر پیچی که منجر به پیچ‌فالت شده، پیچ‌های قبل و بعد آن پیچ (که از همان فایل هستند) هم باید به حافظه منتقل شود. توجه داشته باشید که برای پیدا کردن قربانی از بین پیچ‌های درون حافظه باید دو الگوریتم LRU و Second Chance پیاده‌سازی شده باشد.

(2) Main: در حال حاضر تابع مینی نوشته نشده است که بتوان با استفاده از آن پروژه را اجرا کرد. وظیفه‌ی شما این است که کد این بخش را مطابق با کدی که نوشته‌اید بنویسید. در پایان اجرای این فایل شما باید ۱۱ ردیف پلات که هر ردیف دارای ۳ ستون است داشته باشید. ردیف‌های ۱ تا ۱۰ ستون اول مربوط به تعداد پیچ‌فالت‌ها به ازای اندازه‌ی پیچ‌های متفاوت است که برای انتخاب قربانی از الگوریتم LRU استفاده شده است. ستون دوم نیز به همین صورت است فقط برای انتخاب قربانی از الگوریتم second chance استفاده شده است و ستون سوم هم همانند ستون اول است فقط با حافظه‌ی ۲ برابر. ردیف یازدهم مربوط به میانگین تعداد پیچ‌فالت‌های ۱۰ ردیف قبلی است به ازای اندازه‌ی پیچ‌های مختلف. **برای پلات کردن خروجی‌های خود باید از هیستوگرام استفاده کنید.**

نکات تکمیلی

- برای پیاده‌سازی این پروژه می‌توانید از زبان‌های C و JAVA و Python و Golang استفاده کنید اما در صورت استفاده از زبانی غیر از زبان پایتون، پیاده‌سازی موارد پیاده‌سازی شده بر عهده‌ی خودتان خواهد بود.
- استفاده از هوش مصنوعی برای پیاده‌سازی پروژه بلامانع است اما در زمان تحویل پروژه سوالاتی از مباحث مربوط به پروژه از شما پرسیده خواهد شد تا تسلط شما بر پروژه مشخص شود. توجه داشته باشید که نمره‌ی تسلط شما به صورت **ضریبی از نمره‌ی کل شما** در نظر گرفته خواهد شد.
- در صورت پیدا کردن باگ در کدهای پروژه‌ی اولیه به ازای هر باگ، ۵ نمره‌ی امتیازی برای شما در نظر گرفته خواهد شد.
- تا حد امکان تغییرات خود را فقط در فایل مربوط به MMU انجام دهید و باقی فایل‌ها را تغییر ندهید.
- کد کلی خود را چند بار ران کنید و خروجی‌های این چند بار ران شدن را در قالب یک فایل گزارش (صرفاً تصویر خروجی‌ها) در فایل آپلودی خود قرار دهید.

نکات پایانی

- سوالات و ابهامات خود را در دیسکاشن کانال یا در پیوی @sadramousavi77 و @farzinium و @E_M_R_82 مطرح کنید.

- فرمت نامگذاری پروژه به صورت OS-PROJECT-[Student ID]-[Student Name] باشد.
- در صورت مشاهدهی هرگونه تقلب، نمرهی صفر برای افراد خاطی درنظر گرفته خواهد شد.

موفق باشید