تمرین برنامه نویسی ۲

در این تمرین قرار است یک ماشین ساده طراحی کنید که کار مدیریت حافظه را بر پایه روش Buddy انجام می‌دهد. سیستم عامل یک ماجول بنام مدیر حافظه دارد که در آن چند لیست نگهداری می‌شود. این لیستها فضاهای آزاد و اشغال شده با اندازه‌های 32K, 64K, 128K, 256, 512K, 1024K را نگهداری می‌کنند. امکان تخصیص حافظه کوچکتر از 32K و بزرگتر از 1024K برای یک فرآیند میسر نیست (میتوانید دو لیست برای هر اندازه نگهداری نمایید یکی فضاهای آزاد و یکی فضاهای اشغال شده).
ماشین در ابتدا با یک حافظه N مگابایتی شروع به کار می‌کند. بسته به درخواست فرآیندها، مطابق آنچه در کتاب در مورد روش buddy گفته شده است حافظه به اندازه‌های توان ۲ شکسته می‌شود تا نیاز فرآیند پاسخ داده شود. البته باید دقت شود که کوچکترین اندازه 32K است.
فرآیندها با فراخوانی دستور زیر درخواست فضای حافظه را به ماجول مدیریت حافظه اعلام می‌کنند.

```
long address = allocate (pid, size)
```

درخواست اختصاص حافظه باندازه size در آدرس مجازی address
size = 1, 2, 4, 8, n
pid: شناسه فرآیند

اگر فرآیند هیچ بلاکی ندارد یک بلاک به اندازه لازم به آن اختصاص داده می‌شود. اما اگر در حال حاضر فرآیند دارای چند بلاک است و در همان بلاک جای خالی وجود دارد از همان بلاک فضا به فرآیند داده می‌شود و بلاک جدید در اختیار فرآیند قرار نمیگیرد.
بطور مثال با فراخوانی allocate(1000, 12K) یک بلاک جدید با اندازه 32K به فرآیند شماره 1000 اختصاص می‌یابد. دقت کنید که 20K از فضای بلاک فعلی خالی می‌ماند و باید این فضاهای خالی درون بلاکها نیز ثبت شود. درخواست بعدی allocate(1000, 19K) صادر می‌شود. از آنجا که در بلاک قبلی فرآیند، فضای آزاد به میزان خواسته شده وجود دارد بلاک جدید اختصاص نمی‌یابد. درخواست بعدی به صورت allocate(1000, 266K) ارسال شود. در این حالت باید یک بلاک 512K به فرآیند اختصاص داده می‌شود که 266K آن پر شده است.

فرآیندهای پس از مدت زمان تصادفی (در پیاده‌سازی شما) فضای آدرس به روش زیر را آزاد می‌کنند. مقدار address مقدار ابتدای آدرس اختصاص یافته است

```
deallocate (pid, address)
```

- زمانی که تمامی آدرسهای یک قطعه حافظه آزاد شد کل قطعه به لیست فضاهای آزاد افزوده می‌شود.
- اگر حافظه پر شد باید خطای پر بودن حافظه تولید شود
- هر ۵ ثانیه یک ریسمان در سیستم عامل وضعیت حافظه را گزارش میدهد.
  - میزان حافظه اشغال شده
  - میزان حافظه آزاد
  - میزان حافظه تکه‌تکه شده داخلی و خارجی
  - برای هر فرآیند
    - میزان حافظه مصرف شده
    - مدت زمان اجرا، زمان شروع اجرا و ...
- برای هر لیست باید یک lock جداگانه تعریف شود.

  - فضاهای آزاد شده دوباره با هم ترکیب شده و فضاهای آزاد بزرگتر را تشکیل می‌دهند

فرآیندها

فرآیندها توسط یک کلاس تعریف می‌شوند و در طول موجودیت خود به تعداد تصادفی و با فواصل زمانی تصادفی یکی از ۲ تابع فوق را فراخوانی می‌کنند. هر فراخوانی یکی از دستورات فوق به شکل موازی و با یک ریسمان جداگانه اجرا می‌کند (پیشنهاد می‌شود از ThreadPool استفاده کنید). زمانی که کار فرآیند تمام شد باید کل حافظه اختصاص داده شده به آن فرآیند آزاد شود.

- هر جا که می‌توانید از reader-writer lock استفاده کنید.
- از ساختمان داده‌هایی که خودشان thread-safe هستند و در زبان جاوا بیشتر در java.util.concurrent وجود دارند استفاده نکنید. در استفاده از ساختمان داده‌ها باید خود شما کار محافظت از داده‌های مشترک را انجام دهید.

بخش نمره اضافی

پس از مدتی فضاهای اختصاص یافته به فرآیندها دچار تکه‌تکه شدن داخلی می‌شود. در واقع برخی از آدرسها آزاد شده و در یک قطعه جاهای خالی وجود خواهد داشت.

۱- یک ریسمان دیگر در مدیریت حافظه هست که پس زمینه سعی می‌کند که اگر از بلاکهایی از حافظه‌های آزاد شده برای مدت معینی استفاده نشده‌اند آنها را به هم متصل کرده و بلاک بزرگتر میسازدو مثلا پس از ۱ ثانیه بلاکهای آزاد 32K دو بدو متصل شده و بلاک جدید 64K میسازند

۲- در صورت امکان فضاهای موجود یک فرآیند در بلاک‌های محدودتری قرار بگیرند. مثلا اگر یک فرآیند دارای دو بلاک 32K است که از آنها در یکی 10K و در دیگری 20K استفاده کرده آن دو فضا در یک بلاک قرار گرفته و یک بلاک آزاد شود.