به نام خدا

درس سیستم‌های عامل

نیم‌سال دوم ۰۰-۹۹

دانشکدهٔ مهندسی کامپیوتر

دانشگاه صنعتی شریف

مدرس **مهدی خرازی**

تمرین گروهی **دوم - آزمایشگاه زمان‌بندی**

موضوع **زمان‌بندی**

موعد تحویل کد و گزارش نهایی **ساعت ۲۳:۵۹ سه‌شنبه ۷ اردیبهشت ۱۴۰۰**

با سپاس از دستیاران آموزشی **حسین مقدس، حسین ذاکری‌نیا، حسین احمدزاده، ارشیا مقیمی، مجید گروسی و محمد امیدوار**

اقتباس شده از **CS162 در بهار ۲۰۲۰ در دانشگاه کالیفرنیا، برکلی**

۰ سرآغاز

این مستند به بیان سوالات آزمایش‌های آزمایشگاه زمان‌بندی می‌پردازد. این آزمایشگاه بخشی از تمرین گروهی دوم است. هدف از این آزمایش‌ها آشنایی با مفاهیم زمان‌بندی[۱] ، عدالت[۲] در زمان‌بندی، بهره‌وری[۳] و تاخیر[۴] در پاسخ‌گویی[۵] به برنامه‌ها است. در جریان پاسخ دادن به این سوالات و نوشتن کدهای خواسته شده، با جزییات پیاده‌سازی زمان‌بندها بیشتر آشنا می‌شوید و درک بیشتری از مفاهیم گفته شده به دست خواهید آورد.
کدهای خود را به دفترچه‌ی IPython که در مخزن handouts قرار دارد، اضافه کنید و به همراه کدهای دیگر و گزارش تمرین خود، دفترچه‌ی نهایی را تحویل دهید. هم‌چنین در یک گزارش جداگانه با قالب PDF پاسخ‌های خواسته شده در این آزمایشگاه را تحویل دهید. البته در قالب همان فایل IPython نیز می‌توانید گزارش را تکمیل کنید.

۱ آزمایش‌ها

۱.۱ آزمایش ۱: پیاده‌سازی‌کردن شبیه‌ساز زمان‌بندی

در هر دو بخش این سوال، غرض این است که زمان‌بند را در دفترچه‌ی IPython پیاده‌سازی کنید و سپس شبیه‌سازی را اجرا کنید تا CPU log را ببینید. خروجی شبیه‌سازی هر قسمت را در گزارش خود بنویسید.

(آ) زمان‌بند SRTF را پیاده‌سازی کنید. به منظور پاس شدن تست‌های موجود، لازم است که حالت‌های تساوی را با FIFO زمان‌بندی کنید. با در نظر گرفتن این که مدت زمان هر نوبت[۶] ۲ واحد است، زمان‌بندتان را روی `workload3` که در دفترچه وجود دارد، اجرا کنید و خروجی آن را گزارش کنید. آماده‌سازی‌های این کار در سلول شماره‌ی ۲۱ دفترچه انجام شده است.

(ب) زمان‌بند MLFQ را پیاده‌سازی کنید. زمان‌بند شما دو صف دارد. یک صف برای امور مربوط به تعامل با کاربر که اولویت بالایی دارند و یک صف برای اموری که پردازش زیادی نیاز دارند و اولویت پایینی دارند. در هر یک از صف‌ها، زمان‌بندی به شکل نوبتی[۷] انجام می‌شود. در ابتدا، همه‌ی فعالیت‌ها در صف اولویت بالا قرار دارند و اگر هر یک از این فعالیت‌ها نوبتش در حالی تمام شود که هنوز به پردازش بیشتر نیاز دارد، باید به صف اولویت پایین منتقل شود. مدت زمان هر نوبت را در صف اولویت بالاتر، ۲ واحد و مدت زمان هر نوبت را در صف اولویت پایین، ۴ واحد در نظر بگیرید و زمان‌بندتان را روی `workload3` اجرا کنید و خروجی آن را گزارش کنید. آماده‌سازی‌های این کار در سلول شماره‌ی ۲۷ دفترچه انجام شده است.

۲.۱ آزمایش ۲: به سوی بهره‌وری ۱۰۰ درصد!

دنباله‌ی فعالیت‌های B_i را برای پردازنده در نظر بگیرید. برای سادگی، طول هر یک از این فعالیت‌های پردازنده[۸] را عدد ثابت M در نظر بگیرید. اولین فعالیت پردازنده، B_0 ، در لحظه‌ی $t = 0$ می‌رسد.
برای $i \geq 1$ ، می‌دانیم که $ArrivalTime(B_i) = ArrivalTime(B_{i-1}) + X_i$ است و X_i ها متغیرهای تصادفی همانند و مستقلی $(i.i.d.)$ هستند که از توزیع احتمال نمایی با پارامتر λ پیروی می‌کنند.
برای این که پردازنده امکان این را داشته باشد که چند فعالیت را به طور هم‌زمان اجرا کند، این طور فرض می‌کنیم که هر یک از فعالیت‌های پردازنده مربوط به برنامه‌ی مستقلی است و به یک‌دیگر وابستگی ندارند.

(آ) این سامانه، سامانه‌ای باز است یا بسته؟ توضیح دهید.

(ب) چه مقداری برای λ باید انتخاب کنیم تا میانگین فاصله‌ی زمانی بین ورود دو فعالیت متوالی برابر با M شود؟ توضیح دهید.

1) Scheduling
2) Fairness
3) Utilization
4) Latency
5) Response (Time)
6) quantum
7) Round Robin
8) CPU Burst

(ج) چه مقداری برای λ باید انتخاب کنیم تا میانگین بهره‌وری سامانه برابر با ٪۵۰ شود؟ توضیح دهید.

اکنون یک شبیه‌سازی را در دفترچه‌ی IPython آماده کنید که این سامانه را با تعداد زیادی از فعالیت‌ها مدل کند. مقداری را برای M مشخص کنید و آن را ثابت نگه دارید. نرخ ورود فعالیت‌های پردازنده (یعنی همان λ) را تغییر دهید. از اعداد کوچک شروع کنید و مقدار آن را تا رسیدن به حدود مقداری که در قسمت (ب) به دست آورده‌اید، افزایش دهید. با توجه به ذات تصادفی زمان ورود فعالیت‌ها، بایستی شبیه‌سازی را برای هر نرخ چند بار تکرار کنید و دقت داشته باشید که مقادیری را برای λ انتخاب کنید که به مقدار قسمت (ب) به قدر کافی نزدیک باشند.

(د) همان‌طور که طبق چیزی که گفته شد، مقدار λ را تغییر می‌دهید، چه اتفاقی در مورد بهره‌وری CPU رخ می‌دهد؟ در یک نمودار خطی که محور افقی آن، نرخ ورود و محور عمودی آن، بهره‌وری باشد، نتایج به دست آمده را به تصویر بکشید.

(ه) همان‌طور که طبق چیزی که گفته شد، مقدار λ را تغییر می‌دهید، چه تغییری در مورد زمان پاسخ‌گویی (یعنی فاصله زمان ورود فعالیت تا به پایان رسیدن فعالیت) هر فعالیت رخ می‌دهد؟ در یک نمودار خطی که محور افقی آن، نرخ ورود و محور عمودی آن، زمان پاسخ‌گویی باشد، نتایج به دست آمده را به تصویر بکشید. دقت کنید که هم میانه‌ی[9] زمان پاسخ‌گویی را در نظر بگیرید و هم صدک[10] ۹۵ام زمان پاسخگویی را.

(و) آیا استفاده از یک زمان‌بند خاص، بر روی پاسخ قسمت (ه) تاثیر می‌گذارد؟ (راهنمایی: اگر فکر می‌کنید که استفاده از SRTF به شما کمکی خواهد کرد، به این بیاندیشید که حالات تساوی را چگونه باید مدیریت کرد و روش مدیریت شما چه تاثیری بر میانه‌ی زمان پاسخ‌گویی خواهد گذاشت.)

(ز) به طور کیفی توضیح دهید که چرا یک سامانه‌ی با بهره‌وری نزدیک به ۱۰۰ درصد، تاخیر زیادی در پاسخ‌گویی دارد.

۳.۱ آزمایش ۳: رعایت عدالت بین فعالیت‌ها

دو برنامه‌ی S و T را در نظر بگیرید که هر دو متناوبا -اما نه با دوره‌ی تناوبی معین- به پردازنده نیاز دارند. هر یک از این دو برنامه، به مجموعه‌ای از فعالیت‌های پردازنده نیاز دارد و هر یک از این دو برنامه، بعد از این که هر یک از فعالیت‌های پردازنده‌اش را تمام کرد، پردازنده را در اختیار زمان‌بند قرار می‌دهد. هیچ یک از این دو برنامه، بین بازه‌های فعالیت پردازنده به فعالیت ورودی/خروجی[11] نیاز ندارند.
اکنون S_i و T_i را متغیرهایی تصادفی در نظر بگیرید که به ترتیب بیان‌گر طول فعالیت iام پردازنده برای برنامه‌ی S و T هستند. **فرض کنید که طول فعالیت‌ها (یعنی S_i و T_i) همگی $i.i.d.$ هستند.**
بهنام که به درس سیستم‌های عامل خیلی علاقه‌مند شده است، تصمیم دارد که برنامه‌های S و T را بر روی یک پردازنده اجرا کند. با توجه به این که S و T هر دو دارای توزیع احتمالاتی یکسانی برای طول هر یک از فعالیت‌هایشان هستند، او با خودش استدلال می‌کند که حتی اگر زمان‌بند به دنبال برقراری برابری نباشد، این دو برنامه سهم یکسانی از پردازنده خواهند داشت. در نتیجه، او از یک زمان‌بند ساده‌ی FCFS استفاده می‌کند.

(آ) توضیح دهید که چرا طول صف در زمان‌بند FCFS ، هیچ‌گاه از ۲ بیشتر نمی‌شود.

(ب) مقدار $\Pr[S_1 < T_1]$ را معین کنید.

(ج) فرض کنید که S برای m بار از پردازنده استفاده کرده است و m عدد بزرگی است. با استفاده از قضیه‌ی حد مرکزی[12] ، نشان دهید که متغیر تصادفی مجموع زمان‌های پردازشی S، $\text{CPUTime}(S) = \sum_{i=1}^{m} S_i$ ، یک متغیر تصادفی نرمال است و آن را بر حسب $\mathbb{E}[S_i]$ ، $\text{Var}(S_i)$ و m پَرمایش[13] کنید.

(د) فرض کنید هر دو برنامه به دفعات زیادی اجرا شده‌اند (یعنی می‌توان از قضیه‌ی حد مرکزی استفاده کرد).
مقدار $\Pr[\alpha * \text{CPUTime}(S) < \text{CPUTime}(T)]$ را بر حسب تابعی از α به دست آورید. برای این کار، از پاسخ قسمت (ج) کمک بگیرید و پاسخ خود را بر حسب Φ ، تابع توزیع تجمعی نرمال استاندارد، بیان کنید.

9) Median
10) Percentile
11) IO Burst
12) Central Limit Theorem
13) Parameterize

احتمالی که در قسمت (د) مورد بحث قرار گرفت، بی‌عدالتی -که در این جا، بی‌عدالتی همان نابرابری است!- را به صورت کمّی بیان می‌کند. مثلاً، به ازای $\alpha = 1.05$ ، بیان می‌کند که برنامه‌ی T با چه احتمالی، **حداقل** به اندازه‌ی ٪۵ بیشتر از برنامه‌ی S از پردازنده استفاده می‌کند.

برای سادگی، در قسمت‌های بعد فرض کنید که $\mathbb{E}[S_i] = \sqrt{\mathrm{Var}(S_i)}$. البته این فرض، فرضی نامعقول هم نیست و مثلا به ازای توزیع نمایی، این فرض برقرار است.

(ه) با کمک نرم‌افزاری دلخواه (مثلاً Maple ، WolframAlpha ، Python ، ماشین‌حساب مهندسی و ...)، احتمال این پیش‌آمد را محاسبه کنید که یکی از برنامه‌ها به ازای $m = 100$ ،حداقل ٪۱۰ بیشتر از دیگری، از پردازنده استفاده کند. دقت کنید که هر دو حالت ممکن (T بیشتر از S و یا S بیشتر از T) را لحاظ کنید. همین محاسبه را به ازای $m = 10000$ تکرار کنید و روشن کنید که نتیجه چه تفاوتی می‌کند. آیا بهنام در مورد عدالتِ زمان‌بندِ FCFS درست فکر می‌کرد؟

(و) یک شبیه‌سازی در دفترچه‌ی IPython اجرا کنید تا نتایجی را که در قسمت (ه) به دست آورده‌اید، تایید کند. شیوه‌ی شبیه‌سازی خود را توضیح دهید و نموداری برای توضیح نتایج این قسمت رسم کنید.

(ز) (کنج‌کاوی اختیاری) اگر بهنام از یک زمان‌بندِ نوبتیِ بازدارانه [۱۴] با در نظر گرفتن مدت زمان کوتاهی برای هر نوبت استفاده کند، چه نتایجی به دست می‌آورد؟ در مورد خوبی(ها) و بدی(های) این روش تامل کنید.

14) Preemptive Round Robin