به نام خدا

گزارش پروژه درس اصول سیستم های عامل

استاد : دکتر خلیلیان

نام : محمدرضا معتبر

شماره دانشجویی : ۶۱۰۳۹۸۱۹۷

در این پروژه ۴ الگوریتم زمانبندی FCFS, SJF و RR ساده که مانند FCFSعمل میکند و RR بهبود یافته که مانند SJF عمل میکند با یک دیگر مقایسه شده اند.

شرح کوتاهی از الگوریتم ها:

- در SJF زمانی که CPU بی کار می شود از بین process های موجود آن prossec ی را شروع به پردازش می کند که زمان اجرای کمتری دارد یعنی در ready queue پردازش های موجود به ترتیب طول زمان اجرایشان نگه داری می شوند و هر بار عضو اول انتخاب میشود.
- در FCFS زمانی که CPU بی کار می شود از بین process های موجود آن prossec ی را شروع به پردازش می کند که زود تر درخواست پردازش داده باشد یعنی در ready queue پردازش های موجود به ترتیب زمان درخواست پردازششان نگه داری می شوند و هر بار عضو اول انتخاب میشود..
- در RR معمولی که مانند FCFS عمل میکند از بین process های موجود کسی را انتخاب می کند که زود تر درخواست پردازش داده باشد (در ready queue اول باشد) و بعد از حداکثر زمانی مشخص به اسم Quantum اجرای آن متوقف شده و به انتهای صف افزوده میشود.
- در RR بهبود یافته که مانند SJF عمل می کند به جای این که پردازش ها در صف به ترتیب زمان درخواست پردازششان مرتب شوند به ترتیب طول زمان باقی مانده از اجرایشان مرتب می شوند و عضو اول صف انتخاب می شود و حداکثر به اندازه زمان Quantum اجرا می شود و دوباره به صف اضافه میشود توجه کنید ممکن است به وست صف اضافه شود تا ترتیب در صف به هم نخورد.

شرح توابع و کلاس ها:

- در تابع random با توجه به mode ورودی و با استفاده از زمان سیستم و تابع درهمسازی sha256 و گرفتن باقی مانده مناسب یک عدد تصادفی بین l و r تولید میکند توجه کنید اگر mode برابر ۱ باشد از تابع داخلی پایتون استفاده میشود.
- تابع chisquare اعداد تصادفی حاصل از اجرای تابع random را مورد آزمایش قرار میدهد و طبق آزمایش های انجام شده توسط این تابع اعداد تولید شده توسط random بختی هستند.
- کلاس Task در واقع process را مشخص می کند یک Task دارای Arrival_Time (زمان درخواست پردازش), Burst_Time (طول زمان اجرا), Remaining_Time (طول زمان باقی مانده تا تمام شدن این Task) ,

Wait_Time (مجموع زمان انتظار این Task) و State (حالت Task) و CPU (CPU که این Task را دارد پردازش می کند) است.تابع reset برای یک Task آن را به حالت اولیه بر می گرداند این کار برای این است که Task ها قرار است توسط چند الگوریتم زمانبندی شوند و قبل از اجرای الگوریتم Taskها باید به حالت اولیه باشند.

- کلاس CPU برای مشخص کردن CPU ها است که هر CPU داری یک Task (Taskی که دارد آن را پردازش میکند), Wating_time (مجموع زمان های انتظار و بیکاری), Quantum (زمان Quantum آن که CPU ها در زمان اجرای دو الگوریتم FCFS و SJF فاقد Quantum هستند) تابع get_task نیز اگر CPU بی کار باشد از صف انتظار rq اولین Task را شروع به پردازش میکند و تابع CPU run را یک واحد زمانی جلو می برد یعنی اگر CPU دارای Taskی باشد آن را یک واحد زمانی به جلو می برد و اگر بی کار باشد به زمان بی کاری آن یکی اضافه می شود.
- کلاس Ready_Queue صف هایی از Task ها را مشخص می کند و برای مشخص کردن پردازش های موجود استفاده می شود در یک Ready_Queue یک لیست از Task ها وجود دارد وبا تابع pop_front یک Task از ابتدای آن برداشته می شود و با تابع push_back یک Task به انتهای آن اضافه میشود و با تابع insert_in_order با فرض این که Taskهای آن بر اساس زمان باقی مانده اجرایشان یا به عبارتی براساس طول زمان اجرایشان در الگوریتم SJF مرتب شده اند یک Task توسط الگوریتم باینری سرچ سر جای درست خود قرار می گیرد البته برای این که اجرای کد سریع تر شود می توان به جای این که Taskها را داخل یک لیست نگه داشت داخل set نگهداری کنیم.
- کلاس Dataset برای تولید یک مجموعه از Task ها با استفاده از تابع random استفاده می شود و تابع reset در آن تمام Task های موجود در آن را به حالت ابتدایی بر می گرداند.
- تابه FCFS با ورودی گرفتن یک Dataset و تعداد CPU ها الگوریتم زمانبندی FCFSرا روی Dataset انجام میدهد و میانگین زمان انتظار Taskها و میانگین انتظار CPUها را برمیگرداند.
- بقیه توابع نیز مانند تابع FCFS تابع زمانبندی متناظرشان را احرا می کنند و میانگین زمان انتظار Taskها و میانگین انتظار CPU ها را برمیگردانند.
- تابه std نیز انحراف معیار یک لیست را محاسبه می کند.

توجه کنید اگر در اجرای الگوریتم ها تنها یک CPU موجود باشند الگوریتم SJFبهترین عملکرد را دارد و RR ها بدترین و برای این که مقایسه بهتری بین الگوریتم ها انجام شود الگوریتم ها به چند CPU تعمیم داده شدند و در چند CPUهم مقایسه انجام شده است.

سپس ۲۰۰ بار مقایسه انجام شده است و هر بار Dataset تصادفی ایجاد شده و شماره مقایسه تعداد Task ها و حداکثر زمان درخواست پردازش و حداکثر طول زمان اجرای Task ها به همراه میانگین زمان انتظار Task ها و میانگین زمان انتظار CPU ها توسط اجرای هر یک از الگوریتم های زمانبندی و با ۱و ۲و ۵ CPU در فایل Result ذخیره شده است.

در فایل Final_Result میانگین انتظار کل Task ها و مجموع میانگین انتظار CPU ها و انحراف معیار میانگین انتظار Task ها و CPU ها در اجرای الگوریتم ها با تعداد CPU متفاوت نوشته شده است.

جمع بندی:

با توجه به داده های گرفاه شده از برنامه در زمانبندی در سیستم های با یک پردازش گر الگوریتم SJF بهترین عملکرد را دارد و الگوریتم های RR اصلا عملکرد خوبی ندارند ولی هر چه تعداد CPU ها بیشتر می شوند عملکرد شان بهتر می شود ولی در ۵ پردازش گر نیز همچنان SJF بهترین گزینه است و همچین در الگوریتم های RR تعداد Context Switch ها بسار زیاد است و این باعث بدتر شدن آن میشود ولی در این برنامه زمان Context Switch صفر در نظر گرفته شده است.

و همانطور که انتظار داشتم با توجه به این که SJF الگوریتم بهتری برای انتخاب Task نسبت به FCFS است اگر RR به جای این که بر اساس FCFS طراحی شود بر اساس SJF طراحی شود عملکرد بهتری خواهد داشت.

و با افزایش تعداد CPU ها در ابتدا بهبود چشمگیری در زمان انتظار ها دیده میشود ولی از جایی به بعد تاثیر کم و کم تر میشود به این صورت که با افزایش از یک CPU به دوCPU میانگین انتظار Task ها تقریبا یک سوم میشود ولی در افزایش از دو CPU به سه CPU این ضریب کاهش بیشتر بیشود (تقریبا یک دوم) و ...