بسم الله الرحمن الرحیم

گرد آورنده : پیمان شیخ سلطان

موضوع : پیاده سازی الگوریتم های FCFS, SJF, SRT, RR, MCPU در ویندوز فرم به زبان سی شارپ

مقدمه :

در سیستم عامل های ما کار های متفاوتی صورت می پذیرد که مسئول اجرای این کار ها سی پی یو (CPU) است به هر عدد از این کار ها پردازه (Process) گفته می شود هر پردازه دارای مشخصاتی است که با این مشخصات در درس سیستم عامل آشنا شده ایم و اما آنچه که در این سند ذکر می گردد پیاده سازی یا بهتر است بگوییم شبیه سازی این رفتار ورود و خروج پردازه به سیستم و پردازه و اجرای آنها توسط واحد پردازش و کنترل یا همان CPU می باشد

پردازه ها هنگامی شکل میگیرند که قرار است عملی مانند باز شدن درب سی دی رام صورت بپذیرد در این هنگام سیستم عامل مجموع دستوراتی را باید ایجاد کند تا این عمل انجام گردد اما نکته ای مد نظر است اگر هر پردازه که قرار است در سیستم اجرا گردد فورا به CPU برود و برای اجرا آماده شود به قول خودمان سنگ روی سنگ بند نخواهد آمد نکته ای که مطرح است این است که باید بدانیم بعضی اعمال برای کامپیوتر بسیار ضرروری تر هستند از گوش دادن به یک موسیقی یا نوشتن یک فایل ورود باید در نظر گرفت اجرای متداوم سرویس هایی سیستم عامل باعث نمایش این صفحه شکیل می شود

این موضوع درست است که CPU قادر به اجرای خیلی سریع پردازه هاست اما اگر سیل عظیمی از پردازه های ایجاد شده توسط نرم افزار ها به سمت CPU هدایت شوند و هیچ مدیری را برای کنترل ورود و خروج آنها در نظر نگیریم در این هنگام CPU جان سالم به در نخواهد برد ☺ پس برای جلوگیری از وضع موجود افرادی پیدا و تصمیماتی برای کنترل ورود و خروج پردازه در نظر گرفتند که به آنها الگوریتم های اجرای پردازه در سیستم عامل گفته می شود

در این داکیومنت ما با شبیه سازی بعضی از الگوریتم ها به درک چگونگی اجرا پی خواهیم برد امید است بتوانیم در آینده ای نزدیک الگوریتم های بسیار برتر و بالا تری ارائه دهیم تا شاهد نسل جدید الگوریتم های سریع تر و برتر باشیم در ادامه با نحوه اجرا آشنا خواهیم شد

با ما همراه باشید!

برای شبیه سازی الگوریتم ها ابتدا با توجه به اصول شی گرایی در زبان محبوب سی شارپ یک شی به نام Process ایجاد گردید.ویژگی های این شی

* ProcessName : نام پردازه مورد نظر
* ProcessArriveTime : زمان رسیدن پردازه به صف
* ProcessServiceTime : زمان مورد نیاز پردازه برای اجرا در CPU
* ProcessRemainingTime : زمان باقی مانده پردازه یعنی پردازه به CPU رفته و از حالت اجرا خارج شده
* ProcessGetQTime : در الگوریتم روند رابین بررسی می کند که آیا پردازه زمان مورد نیاز کوانتوم زمانی را دریافت کرده است یا خیر فرض کنید کوانتوم 2 می باشد و پردازه وارد صف موقت شده و یک کوانتوم دریافت می کند و تایم لاین یک ثانیه جلو می رود و باید یک ثانیه کوانتوم دیگر دریافت کند این متغییر نقش نگه دارنده مقدار کوانتوم های دریافت شده را دارد اگر متغییر به خاطر آمدن پردازه جدید جا به جا شود این مقدار 0 می شود تا به انتهای صف رفته و هنگامی که نوبتش شد کوانتوم دریافت نماید
* ProcessWaitingTime : زمان انتظار پردازه یعنی پردازه برای وارد شدن به CPU باید مقدار زمانی منتظر بماند
* ProcessTurnaroundTime : زمان سرویس هر پردازه و زمان انتظار محاسبه شده بعد از اجرا ی الگوریتم در این قسمت ذخیره می شود.
* ProcessIsComplete : کامل شدن پردازه یعنی اینکه پردازه زمان باقی مانده اش صفر (0) شده است و از صف خارج می شود
* ProcessEnterTimeFromQueue : لیستی از زمان هایی که پردازه برای اجرا وارد CPU می شود
* ProcessExitTimeFromQueue : لیستی از زمان هایی که پردازه از حالت اجرا در CPU خارج می شود
* is\_null : آیا این پردازه پردازه ای خالی است یعنی اینکه اگر پردازه نام نداشته باشد خالی در نظر گرفته می شود
* Process : این یک کانستراکتور یا تابع سازنده پردازه است و زمانی که برای ایجاد شی جدید از آن استفاده می کنیم در **حالت اول** سه آرگومان یا ورودی دریافت می کند و یک پردازه برای ما ایجاد می نماید
  + ProcessName : مقدار اول ورودی از تابع را در این قسمت وارد می کند این مقدار pn نام دارد
  + ProcessArriveTime : مقدار دوم ورودی از تابع را در این قسمت وارد می کند این مقدار pa نام دارد
  + ProcessServiceTime : مقدار سوم ورودی از تابع را در این قسمت وارد می کند این مقدار ps نام دارد
  + ProcessWaitingTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessTurnaroundTime: در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessRemainingTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessGetQTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessIsComplete : در ابتدای ساخت شی این مقدار برابر با خیر (false) خواهد بود
  + ProcessEnterTimeFromQueue : در ابتدای ساخت شی این مقدار برابر با یک لیست خالی خواهد بود
  + ProcessExitTimeFromQueue : در ابتدای ساخت شی این مقدار برابر با یک لیست خالی خواهد بود
* Process : این یک کانستراکتور یا تابع سازنده پردازه است و زمانی که برای ایجاد شی جدید از آن استفاده می کنیم در **حالت دوم** هیچ ورودی دریافت نمی کند و یک پردازه برای ما ایجاد می نماید
  + ProcessName : در ابتدای ساخت شی این مقدار برابر با خالی یا “” خواهد بود
  + ProcessArriveTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessServiceTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessWaitingTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessTurnaroundTime: در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessRemainingTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessGetQTime : در ابتدای ساخت شی این مقدار برابر با صفر (0) خواهد بود
  + ProcessIsComplete : در ابتدای ساخت شی این مقدار برابر با خیر (false) خواهد بود
  + ProcessEnterTimeFromQueue : در ابتدای ساخت شی این مقدار برابر با یک لیست خالی خواهد بود
  + ProcessExitTimeFromQueue : در ابتدای ساخت شی این مقدار برابر با یک لیست خالی خواهد بود

در فرم ابتدایی برنامه به نام Form1 5 دکمه وجود دارد که عبارتند از F.C.F.S , S.J.F , S.R.T , R.R , M.CPU

هر کدام از دکمه ها با رویداد کلیک به فرم دیگری به نام excuteForm هدایت می شوند و هنگام انتقال مقدار ویژگی یا پراپرتی form\_type را با توجه به نوع مقدار دهی میکنند

برای F.C.F.S مقدار 1

برای S.J.F مقدار 2

برای S.R.T مقدار 3

برای R.R مقدار 4

برای M.CPU مقدار 5

به فرم excuteForm ارسال میگردد و باید توجه کرد این فرم جدید به صورت ShowDialog باز می شود یعنی فقط یکی از آنها باز می شود و منتظر می ماند تا به فرم پدر پاسخ یا DialogResult ارسال گردد یا فرم بسته شود تا پاسخ بستن یا DialogResult.Cancel به فرم پدر ارسال شود.

در فرم اجرا یا excuteForm متغییر هایی تعریف شده که جدا از form\_type که کاربرد آن را دیدیم باقی موارد عبارتند از

* qtime : این مقدار همان زمان کوانتوم برای الگوریتم روند رابین است
* cn\_switch : این مقدار همان زمان مورد نیازی است که پردازه ای که توسط الگوریتم روند رابین در حال اجرا بوده از سی پی یو خارج می شود و مدیر صف می رود تا اطاعات پردازه فعلی را در کنترل بلاک آن وارد کند و پردازه بعدی را می آورد تا آن را به حالت اجرا در بیاورد و این زمانی را اشغال می کند که به آن زمان تعویض متن می گویند
* process\_list : این لیست خالی برای این است که پردازه ها را در آن وارد کنیم تا بتوانیم روی آنها تغییراتی را اعمال نماییم

معرفی توابع برنامه :

* BindGrid : کاربرد این تابع برای این است که در فرم اجرا بتوانیم پردازه هایی که توسط تابع addProcess\_btn\_Click اضافه می شوند را در دیتا گرید ویو DataGridView نمایش دهد همچنین متن دو عدد لیبل یا برچسبی که داخل فرم اجرا وجود دارد را برابر با تعداد عناصر لیست پردازه ها قرار میدهد
* addProcess\_btn\_Click : کاربرد این تابع این است که فرم AddProcess را در حالت showDialog به حالت اجرا در خواهد آورد و فرم AddProcess مقدار هایی مثل نام پردازه زمان مورد نیاز برای اجرا در CPU و زمان رسید پردازه به صف را دریافت می کند و به فرم اجرا می فرستد در نهایت تابع BindGrid را صدا خواهد زد
* excuteForm\_Load : این تابع در زمانی که فرم باز می شود اجرا می گردد و کار آن این است که ابتدا تابع BindGrid را اجرا نموده و سپس با تو جه به form\_type نام سر برگ فرم اجرا را یکی از موارد
  + First Come First Service
  + Shortest Remaining Time
  + Shortest Job First
  + Round Robin
  + Multi CPU

قرار می دهد همچنین متن لیبل یا برچسب های فرم را برابر با مقدار 0 قرار می دهد

* refreesh\_btn\_Click : این تابع تابع BindGrid را صدا میزند
* deleteProcess\_btn\_Click : وظیفه این تابع حذف پردازه از لیست پردازه ها یا process\_list هم چنین صدا زدن تابع BindGrid برای اپدیت دیتا گرید ویو می باشد باید توجه کرد اگر لیست پردازه ها خالی باشد این دکمه نباید فشرده شود زیرا باعث خارج شدن برنامه از حالت اجرا یا کرش Crash کردن برنامه می گردد برای همین آن را غیر فعال می نماییم تا کاربر در صورت خالی بودن لیست نتواند این دکمه را کلیک نماید
* calculate\_btn\_Click : این تابع ابتدا وضعیت دکمه اجرای الگوریتم را غیر فعال می نماید تا کاربر تا زمانی که پردازه ای را اضافه ننموده نتواند دوبار این دکمه را بفشارد تا در عملکرد محاسبه اشکالی وارد گردد سپس باتوجه به form\_type یکی از توابع FCFS SJF SRT RR MCPU را به حالت اجرا در خواهد آورد تا محاسبات لازمه صورت پذیرد
* FCFS : این تابع لیست پردازه ها را دریافت و بر روی آن پردازشی انجام می دهد تا زمان انتظار کل و زمان کل کار محاسبه و نمایش داده شود این تابع الگوریتم FCFS را اجرا می کند
* SJF : این تابع لیست پردازه ها را دریافت و بر روی آن پردازشی انجام می دهد تا زمان انتظار کل و زمان کل کار محاسبه و نمایش داده شود این تابع الگوریتم SJF را اجرا می کند
* SRT : این تابع لیست پردازه ها را دریافت و بر روی آن پردازشی انجام می دهد تا زمان انتظار کل و زمان کل کار محاسبه و نمایش داده شود این تابع الگوریتم SRT را اجرا می کند
* RR : این تابع لیست پردازه ها را دریافت و بر روی آن پردازشی انجام می دهد تا زمان انتظار کل و زمان کل کار محاسبه و نمایش داده شود این تابع الگوریتم RR را اجرا می کند
* MCPU : این تابع لیست پردازه ها را دریافت و بر روی آن پردازشی انجام می دهد تا زمان انتظار کل و زمان کل کار محاسبه و نمایش داده شود این تابع الگوریتم MCPU را اجرا می کند
* sort\_by\_alg\_btn\_Click : این تابع با استفاده از form\_type و دستورات Linq لیست پردازه های نمایش داده شده در دیتا گرید ویو را مرتب می نماید

نحوه عملکرد تابع FCFS :

در این الگوریتم پردازه ها به این صورت مرتب می شود که اولویت با زمان رسیدن و نام پردازه در صورت تکراری بودن زمان رسیدن است

سپس یک برهه زمانی یا TimeLine را در نظر می گیریم که پردازه ها به آن وارد می شود و اجرا می شود و تا زمان پایان اجرا که ProcessIsComplete برابر با true شود ادامه خواهد یافت و سپس پردازه بعدی در دست گرفته می شود

در نهایت مقدار total\_waiting\_time برابر با مقدار کل زمان انتظار پراسس ها که از فرمول

current\_time - item.ProcessArriveTime

بدست آماده و داخل پردازه ذخیره شده قرار خواهد گرفت

بدین معنی که پردازه ای که داریم روی آن حلقه می زنیم و حلقه به آن رسیده است زمانی رسیدن به صف دارد و زمان فعلی تایم لاین یا current\_time که از آن کم شود زمان انتظار بدست خواهد آمد با یک مثال عددی برای درک همراه باشید

فرض کنید پردازه ای در زمان 0 وارد صف شده و 3 ثانیه برای اجرا نیاز دارد پس current\_time برابر با 0 خواهد بود در همین حین پردازه ای در زمان 2 وارد صف شده و 5 ثانیه برای اجرا نیاز دارد

حال ثانیه به ثانیه تایم لاین یا برهه زمانی را بررسی می نماییم در ثانیه 0 تا 1 پردازه از زمان باقی مانده اش یکی کم می گردد و می شود 2 هنوز پردازه دیگری وارد نشده و الگوریتم اجازه خارج شدن این پردازه از CPU را هم نمی دهد پس به ثانیه بعدی می رویم باید توجه کرد این پردازه که اول از همه آمده اصلا منتظر دیگر پردازه ها نمانده و زمان انتظار آن 0 است

حال به ثانیه 1 به 2 میرویم و در این زمان پردازه دیگری وارد سیستم شده زمان باقی مانده این پردازه از 2 به 1 تغییر می کند و current\_time هم برابر با 2 خواهد بود

حال به ثانیه 2 به 3 میرویم پردازه دوم 1 ثانیه منتظر مانده و زمان باقی مانده این پردازه هم 0 شده پس باید تکمیل و از صف و لیست پردازه ها temp\_list خارج شود و نوبت اجرا به پردازه بعدی که 1 ثانیه هم منتظر مانده بدهد توجه کنید current\_time برابر با 3 شده است

زمان رسید پردازه جدید 2 و current\_time برابر با 3 است و اگر این دو را از هم کم کنیم مقدار زمان انتظار پردازه دوم بدست می آید

می شودItem.total\_service\_time در نهایت هنگامی که حلقه بر روی همه پردازه ها صورت پذیرفت برابر با مجموع همه زمان ها اجرا یا

مقدار دو برچسب برابر با total\_waiting\_time و total\_service\_time می شود و الگوریتم با موفقیت اجرا شده است

نحوه عملکرد تابع SJF :

اجرای این الگوریتم به این صورت است که پردازه ها به اولویت زمان رسیدن و زمان سرویس مرتب می شوند و دقیقا اجرایی برابر با FCFS دارند با این تفاوت که دیگر نام پردازه دلیل بر اجرای آن نیست بلکه همان مرتب سازی اولیه است که این الگوریتم را به وجود می آورد

نحوه عملکرد تابع SRT :

این الگوریتم به این صورت اجرا شده که پراسس ها با توجه به زمان رسیدن و زمان سرویس و در نهایت بر اساس نام مرتب می شوند

برای پیاده سازی تایمم لاین یک متفییر به نام total\_runtime ایجاد گردیده و با توجه به ورود و خروج پردازه ها به صف و زمان اجرا شدن زیاد می شود مثلا یک پردازه زمان سرویس 10 دارد و این مقدار total\_runtime در ابتدا 0 است و بعد از اجرا برابر با 10 می شود یعنی اینکه حالا ما در ثانیه 10 هستیم و این دقیقا همان شبیه سازی است که از آن سخن گفته شد یک لیست temp\_process\_time\_line هم در نظر می گیریم که وظیفه آن این است که پردازه هایی که می خواهد از CPU خارج شوند در به صورت موقت در خود نگه دارد تا اجرای آن کامل شود حال یک تایم لاین را در نظر بگیرید که پردازه ها در آن وارد و از آن خارج می شوند پردازه هایی که وارد تایم لاین می شوند از زمان باقی مانده آنها کم می شود و هنگام وارد شدن به لیست زمان ورود ProcessEnterTimeFromQueue پردازه زمان مورد نظر اضافه می شود و هنگام خارج شدن پردازه از CPU هم به لیست زمان خروج ProcessExitTimeFromQueue پردازه زمان خروج اضافه می شود

با توجه به الگوریتم SRT که انقطاع پذیر است پردازه ها وارد CPU می شوند و به حالت اجرا می روند زمانی که پردازه دیگری وارد صف temp\_process\_time\_line شود مدیر زمان باقی مانده پردازه ای را کم می کند که زمان باقی مانده کمتری دارد و هنگامی هم که پردازه تکمیل شد آن را از صف حذف می کند

در نهایت با توجه به ورود و خروج پردازه ها در ProcessEnterTimeFromQueue و ProcessExitTimeFromQueue در صورتی که پردازه بیش از دو بار به صف آمده و از صف رفته زمان خروج را از زمان ورود کم و به زمان انتظار پردازه اضافه می نماید زیر به صورت منطقی اگر پردازه ای یک بار وارد صف شود یعنی هیچ منتظر نمانده اما اگر چند بار به صف بیاید و از صف خارج شود یعنی زمانی منتظر مانده و ما آن زمان را محاسبه می کنیم

باقی موارد مانند زمان سرویس و مجموع زمان انتظار هم مانند الگوریتم های دیگر اجرا می شود

نحوه عملکرد تابع RR :

اجرای این الگوریتم بسیار مشابه SRT است در ابتدا کار به وسیله فرم q\_time دو مقدار دریافت می کند یکی زمان تعویض متن و دیگری زمان کوانتوم سی پی یو CPU که آنها را با زدن دکمه (ادامه) آنها را به فرم بعدی یا excuteForm ارسال می کند و اما بعد پردازه ها هنگام ورود شان به صف موقت به نام temp\_process\_time\_line\_list با توجه به مقدار کوانتوم زمانی یا qtime اجرا و از حالت اجرا خارج می شوند و جای خود را به پردازه دیگر می دهند هنگامی که بخواهند جای خود را به پردازه دیگر بدهند زمانی برای نوشتن اطلاعات پردازه بر روی بلاک کنترل می گیرد که این زمان زمان تعویض متن نام دارد و باید به زمان انتظار همه پردازه های موجود در temp\_process\_time\_line\_list اضافه شود در حیت اضافه شدن پردازه به صف زمان افزوده شدن آن از متغییر تایم لاین یعنی time\_line خوانده و به صف ProcessExitTimeFromQueue یا ProcessEnterTimeFromQueue اضافه می شود

باید توجه کرد در این الگوریتم به محض ورود پردازه به تایم لاین باید آن پردازه اجرا و پردازه در حال اجرا از جالت اجرا خارج شود و یک زمان تعویض متن به زمان انتظار آن اضافه شود

در نهایت زمانی که هیچ پردازه ای داخل temp\_list و temp\_process\_time\_line\_list باقی نماند یعنی اجرای الگوریتم با موفقیت کامل شده است و وقت محاسبه زمان انتظار کل می باشد در این جا هم لیست زمان های ورود و خروج پردازه از صف مطرح است و با توجه به اینکه اگر پردازه یک بار به صف آمده و یک بار خارج شده یعنی منتظر نمانده و زمان انتظارش همان مجموع زمان های context\_switch یا تعویض متن است در غیر این صورت زمان خروج پردازه از زمان ورودش پردازه کم و زمان انتظار بدست می آید

دیگر خروجی ها مثل مقدار لیبل ها همانند دیگر پردازه ها قرار خواهند گرفت

نحوه عملکرد تابع MCPU :

این الگوریتم بدین صورت اجزا می شود که در ابتدا توسط فرم CpuCounter تعداد سی پی یو CPU های موجود وارد می شود سپس به تعداد سی پی یو های وارد شده لیست موقت ایجاد می کند و پردازه هایی که در این الگوریتم وارد شده اند را به ترتیب زمان رسیدن و زمان نام پردازه مرتب می کند

حالا اولین پردازه که زمان رسیدن اش از زمان فعلی شبیه ساز زمان یا همان متغییر timeline کمتر باشد را به سی پی یو ایکس ام اضافه می کند

البته باید به این نکته هم توجه کرد پردازه در صورتی اضافه می شود که داخل سی پی یو پردازه ای وجود نداشته باشد یا در واقع cpus\_list[i].Count == 0 یعنی تعداد پردازه داخل آن سی پی یو برابر با 0 باشد در همین حال باقی پردازه هایی که وارد صف شده اند ولی داخل سی پی یو نرفته اند باید به زمان انتظار آنها یک واحد اضافه شود

foreach (var item in temp\_list) برای همه پردازه های وارد شده

{

if (item.ProcessArriveTime <= timeline) اگر از تایم زمانی کمتر بودند پس وارد شده و اجرا نشده اند

{

item.ProcessWaitingTime += 1; در نتیجه باید برای وارد سی پی یو شدن منتظر بمانند تا ثانیه بررسی شوند

}

}

و اما شرط پایانی این است که داخل لیست موقت یا temp\_list هیچ پردازه ای باقی نماند و همزمان داخل سی پی یو هم پردازه ای در حال اجرا نباشد و گرنه باید تایم لاین را همچنین زیاد کنیم تا همه پردازه های داخل سی پی یو اجرا شوند

در نهایت مقدار های بدست آمده را ثبت می کنیم و داخل برچسب ها قرار می دهیم