

# کنترل سرعت در دستان شما

روش پردازش فایل انتخاب خواهد شد. حال اولین چیزی که به نظر می‌رسد آن است که داده‌ها را بابت به بابت از فایل خوانده و پس از پردازش مجدداً به فایل برگردانیم.

تجربه نشان داده است که این کار، حتی اگر تنها خواندن و نوشتن باشد و هیچ پردازش زمان‌گیری روی داده انجام نشود، باز هم کارایی و سرعت عمل برنامه بشدت کم خواهد بود. عملیات ورودی-خروجی نسبت به کارهای داخل CPU و حافظه اصلی از جمله کارهای زمان‌گیر به حساب می‌آید. حال اگر این کار با فایلی با حجم گفته شده بخواند به تعداد بایت‌های فایل انجام گیرد، افت شدید کارایی برنامه چیز عجیبی نخواهد بود. بنابراین بهتر است با هر بار عملیات ورودی / خروجی و پس از مراجعه به دیسک، حجم بیشتری از فایل را فراخوانی کنیم و پس از مقیم کردن آن مقدار در حافظه، باقی کار را با سرعت حافظه اصلی انجام دهیم. حال این سوال مطرح می‌شود که پس از هر بار مراجعه به دیسک چه مقدار از داده را به حافظه منتقل کنیم.

ممکن است برای اجرای یک برنامه آنقدر حافظه اصلی در اختیار داشته باشیم که برنامه‌نویس تصمیم بگیرد پس از هر مراجعه به دیسک مثلاً یک مگابایت از داده‌ها را به حافظه منتقل کند.

این کار دو اشکال می‌تواند داشته باشد. اول این که این برنامه نظر میزان از حافظه مصرفی، بهینه نخواهد بود و اگر قرار باشد برنامه در رایانه‌ای با حافظه کمتر اجرا شود، امکان مواجه شدن با کمبود حافظه و افت کارایی در آن وجود خواهد داشت. از طرف دیگر اگر نرم‌افزار ما یک برنامه محاوره‌ای باشد، کاربر در هنگام خوانده شدن هر یک مگابایت باید مدتی صبر کند تا بتواند با سایر بخش‌های برنامه کار کند که این مساله نیز یک ضعف برای نرم‌افزار به شمار می‌رود.

شاید بهترین حالت این باشد که پس از هر بار مراجعه به دیسک به میزان یک سکتور یا یک بلاک دیسک داده فراخوانی شود. یک بلاک دیسک معمولاً ۴ یا ۱۶ کیلوبایت است که البته کار برهنگام فرمت کردن دیسک می‌تواند این مقدار را تعیین کند. اما به طور عام این مقدار در نظر گرفته می‌شود. فراخوانی یک بلاک داده در هر بار عملیات I/O چند مزیت دارد. یکی آن که کمترین میزان حافظه برای بالا بردن کارایی برنامه مصرف می‌شود. از این مقدار کمتر حجم عملیات I/O اضافه می‌شود و از این بیشتر ممکن است بدون افزایش محسوس کارایی تنها حافظه بیشتری مصرف شود.

مزیت دیگر آن است که هنگام فراخوانی یک بلاک از دیسک، تضمین می‌شود که هر دیسک وقتی به محل مورد نظر رسید برای خواندن مقدار مورد نظر دیگر لازم نیست حرکت کند. چون یک بلاک داده در دیسک به صورت پشت سر هم نوشته می‌شود.

با توجه به این که زمان‌گیرترین عمل در یک دیسک سخت، حرکت هد آن است، کارایی بهینه خواهد بود و البته باز هم نسبت به شرایط و محیط اجرا می‌توان تدابیر دیگری اندیشید که این برعهده طراح نرم‌افزار و مخصوصاً برنامه‌نویس خواهد بود.



Photo: onestepback.com

و پس از اتمام پردازش روی آنها، نتیجه را در فایل بازنویسی کنیم. البته اغلب امکان پردازش مستقیم داده‌ها از روی فایل نیز وجود دارد که در صورت استفاده از این امکان حجم کمتری از حافظه اصلی اشغال خواهد شد. اما عملیات ورودی / خروجی (I/O) یعنی خواندن و نوشتن مکرر از / روی فایل زمان زیادی را از برنامه خواهد گرفت و ممکن است کارایی آن را بشدت کاهش دهد. هیچ قانونی برای استفاده از حافظه اصلی به جای فایل یا بالعکس برای برنامه‌نویسان وجود ندارد که بخواهند طبق آن عمل کنند. هر برنامه‌نویس با توجه به دانش و تجربه نسبی که دارد باید مناسب‌ترین روش را برای بهبود بازده برنامه خود اتخاذ کند.

## افزایش کارایی در ورودی / خروجی

مسلم است که اگر برنامه با حافظه اصلی کار می‌کند بیشترین سرعت ممکن را خواهد داشت. اما زمانی که تصمیم بر این شد که پردازش داده‌ها به طور مستقیم از روی فایل انجام شود، در به کارگیری آن روش نیز باید توجه لازم صورت گیرد. فرض کنید یک فایل حجیم صد مگابایتی داریم که قرار است روی تک‌تک بایت‌های آن پردازشی صورت گیرد و مجدداً در فایل نوشته شود. مسلماً مقیم کردن صد مگابایت داده در حافظه منطقی نخواهد بود و

بنابر این جدا از سرعت رایانه، به کارگیری یک الگوریتم مناسب در سرعت اجرای برنامه‌ها تأثیر مستقیم خواهد داشت.

## استفاده مناسب از حافظه اصلی و فایل

همان‌طور که می‌دانید سرعت عمل حافظه اصلی در مقایسه با حافظه جانبی بسیار بیشتر است. اما ظرفیت حافظه اصلی محدود بوده و قیمت آن نیز نسبت به دیسک سخت‌گران‌تر است. در نوشتن برنامه‌ها، برنامه‌نویس باید به این مساله توجه کافی داشته باشد که برنامه نوشته شده باید حتی‌الامکان میزان کمتری حافظه اصلی را اشغال کند.

یکی از معیارهای مهم در سنجش یک برنامه در کنار سرعت اجرای آن، میزان حافظه مصرفی آن است. اما استفاده صحیح و مناسب از آن باعث بهبود کارایی و افزایش سرعت برنامه خواهد شد. داده‌هایی که مقیم حافظه شده‌اند و روال‌ها و توابع برنامه به طور مستقیم از حافظه اصلی به آنها دسترسی دارند با حداکثر سرعت پردازش خواهند شد. بنابراین اگر قرار است روی داده‌هایی که قرار است کار زیادی با آنها داریم و درون فایل قرار دارند پردازشی کنیم، در صورتی که حجم آنها زیاد نیست بهتر است که فضایی را در حافظه اصلی برای آنها در نظر گرفته و آنها را به طریق در آنجا مقیم کنیم

## پاراساستوده‌نیا

در صحبت‌های قبل به سرعت برنامه‌ها و نقش برنامه‌نویس در آن اشاره کردیم و گفتیم یکی از مهم‌ترین عوامل در بهسازی سرعت برنامه، کاهش دستورات تکرار شونده است. درون رایانه هر کاری که قرار است انجام شود نیاز به زمان دارد. یک دستور if ساده حتی اگر شرط آن برقرار نباشد و کنترل برنامه به داخل بلاک if وارد نشود باز هم اگر این کار به تعداد زیاد انجام شود زمان‌گیر خواهد بود. حال می‌خواهیم ببینیم چه عوامل و نکات دیگری در سرعت برنامه موثر خواهد بود.

## الگوریتم مناسب

شاید بتوان گفت مهم‌ترین و تأثیرگذارترین مساله در تعیین کارایی و سرعت یک برنامه، الگوریتم یا روش مورد استفاده در آن برنامه است. از ابتدای به وجود آمدن رایانه، برنامه‌نویسان و متخصصان درصدد طراحی الگوریتمی بوده‌اند که بتواند با صرف کمترین زمان، هزینه و انرژی آنها را به هدف برساند.

ممکن است شخصی الگوریتمی طراحی کند که عمل بخصوصی را در مدت زمان مشخصی انجام دهد و شخص دیگری روشی ارائه دهد که بتواند همان کار را در زمان کمتری انجام دهد. مسلماً برنامه‌ای با ارزش‌تر است که بتواند کاری را در کمترین زمان ممکن انجام دهد.

به عنوان یک نمونه ساده فرض کنید برنامه‌ای قرار است یک سری داده را از ورودی به طور تصادفی دریافت کرده و آنها را مرتب کرده و به خروجی تحویل دهد. تاکنون روش‌های زیادی برای این کار مطرح شده است که هر یک به طریقی عمل مرتب‌سازی را انجام می‌دهد. هر کدام از آنها نیز نسبت به دیگری معایب و مزایایی دارد که آن روش را نسبت به بقیه متفاوت می‌سازد. یک برنامه‌نویس با توجه به نیازهای خود و شرایط موجود باید سعی کند بهترین روش را انتخاب کند و در صورتی که هیچ‌یک از روش‌های موجود پاسخگوی نیاز وی نبود خود اقدام به طراحی الگوریتمی کند که بتواند به نیازها پاسخ دهد.

همیشه یک الگوریتم نامناسب تنها موجب کندگی اجرا نخواهد شد. گاهی اگر یک الگوریتم زمان معقولی برای پاسخ‌دهی نداشته باشد، استفاده از آن عملاً غیرممکن خواهد بود. به عنوان یک مثال ساده، مساله‌ای به نام فروشنده دوره‌گرد (TSP) در علوم رایانه مطرح است که حل آن به روش معمول در مواردی ممکن است به هزار سال زمان نیاز داشته باشد!

این بدان معنی است که این روش عملاً قابل اجرا نیست. همین مساله با همان ابعاد با یک روش دیگر می‌تواند در ۵ دقیقه حل شود. ضمن این که روش جدید نیز که می‌تواند آن مساله را ۵ دقیقه‌ای حل کند اگر ابعاد مساله را بزرگ‌تر کنیم آن هم ممکن است برای پاسخ دهی به سال‌ها زمان نیاز داشته باشد.