مدیریت حافظه

عناصری از برنامه که در حین اجرا نیاز به حافظه دارند

۱- سگمنت کد برنامه

۲- برنامه‌های زمان اجرای سیستم

۳- ثوابت و ساختمان داده‌های تعریف شده توسط برنامه نویس

۴- نقاط برگشت زیر برنامه‌ها

۵- محیط‌های ارجاع

۶- حافظه‌های موقت برای ارزیابی عبارات

۷- حافظه موقت برای انتقال پارامتر

۸- بافرهای ورودی – خروجی

۹- داده‌های کمکی سیستم

علاوه بر داده‌ها و برنامه‌ها، عملیاتی نیز وجود دارند که نیاز به حافظه دارند. مانند:

۱- عملیات فراخوانی و برگشت از زیر برنامه

۲- علیات ایجاد و از بین بردن ساختمان داده‌ها

۳- عملیات درج و حذف عناصر ساختمان‌ها.

در بعضی از زبان‌ها مثل C و C++ برنامه نویس قادر است کنترل زیادی بر روی حافظه داشته باشد، در بعضی از زبان‌ها مثل Java ، کنترل حافظه را مستقیماً در اختیار برنامه نویس قرار نمی‌دهند و مدیریت حافظه توسط ویژگی‌های خود زبان صورت می‌گیرد.

امتیاز مدیریت حافظه توسط برنامه نویس این است که در هر زمانی که نیاز به حافظه است، حافظه تخصیص می‌یابد و در صورت عدم نیاز، بلافاصله حافظه آزاد می‌شود.

دیدگاه های بررسی مدیریت حافظه

مدیریت حافظه از سه دید گاه قابل بررسی است:

۱- از دیدگاه طراح زبان

۲- از دیدگاه پیاده ساز

۳- از دیدگاه برنامه نویس

مدیریت حافظه شامل سه مرحله است:

۱- تخصیص اولیه

۲ - بازیابی حافظه

۳- فشرده سازی و استفاده مجدد

در زبان‌هایی مثل C و C++ و Java، حافظه در زمان اجرا دارای سه بخش است:

بخش ایستا	حاوی مقادیری که میزان حافظه‌ی مورد نیاز آن‌ها قبل از زمان اجرا مشخص است و در طول اجرای برنامه ثابت می‌ماند.
بخش پشته زمان اجرا	رکوردهای فعالیت زیر برنامه‌ها را نگهداری می‌کند که شامل متغیرهای محلی، پارامترها و غیره است.
بخش heap	شامل مقادیری است که به طور پویا تخصیص می‌یابند و در حین اجرای برنامه سازمان‌دهی می‌شوند. مثل لیست‌های پیوندی.

روش های مدیریت حافظه

حافظه را می توان به روش های زیر مدیریت کرد:

۱- ایستا

در مدیریت حافظه ایستا، تخصیص حافظه در زمان ترجمه انجام می‌شود.

امتیاز تخصیص حافظه ایستا، کارآمد بودن و سهولت پیاده سازی آن است.

زیرا در زمان تخصیص حافظه در موقع اجرا، صرفه‌جویی می‌شود و روال مدیریت حافظه زمان اجرایی وجود ندارد که فضایی را اشغال کند.

مترجم می‌تواند آدرس تمام داده‌ها را مشخص کند.

۲- مدیریت حافظه پشته

ساده‌ترین تکنیک مدیریت حافظه زمان اجرا، پشته است.

رکورد فعالیت هر زیر برنامه در پشته ذخیره می‌شود.

این پشته در زمان اجرا ایجاد و دستکاری می‌شود، به همین علت به آن پشته زمان اجرا می‌گویند که کنترل آن توسط سیستم عامل انجام می‌شود و برنامه نویس در تخصیص حافظه و آزاد سازی آن نقشی ندارد.

۳- مدیریت حافظه heap

در این نوع مدیریت حافظه پویا، برنامه نویس در هر نقطه‌ای از برنامه می‌تواند حافظه‌ای را از سیستم دریافت کند و هر وقت به آن نیاز نداشت آن را به سیستم برمی‌گرداند.

تذکر: موارد ۲و۳ را مدیریت حافظه پویا نیز می نامند.

حافظه **heap** را می‌توان به دو صورت در نظر گرفت:

۱- با سلول‌های طول ثابت
۲- با سلول‌های طول متغیر

مدیریت حافظه heap با سلول‌های طول ثابت

در این روش، تمام تخصیص‌ها و آزاد سازی‌ها با سلول‌هایی با طول ثابت انجام می‌شوند. حافظه به دو روش می‌تواند به لیست فضای آزاد برگردانده شود:

۱- برگشت صریح (explicit return)
۲- جمع‌آوری حافظه مازاد (grarbage collection)

ساده‌ترین روش بازیابی حافظه heap، برگشت صریح است.
در C با تابع ( )free و در C++ با عملگر delete می‌توان این کار را انجام داد.
برگشت صریح منجر به دو مشکل می‌شود:
۱- ارجاع‌های معلق
۲- زباله

ارجاع معلق

ارجاع معلق، اشاره‌گری است که حاوی آدرس محلی از heap است که آن محل آزاد شده باشد.

در مثال زیر q ، ارجاع معلق است:

```
int    *p , *q;

p = new  int;

q = p;

delete  p;
```

روش‌های حل مسئله ارجاع معلق

۱- استفاده از سلول‌های حافظه واسطه
۲- قفل کردن و کلید (Looks – and – keys)
۳- به برنامه نویس اجازه داده نشود تا اشیای داده heap را آزاد کند.

سیستم زمان اجرا به طور خودکار، اشیای داده heap را که دیگر قابل استفاده نیستند، آزاد می‌کند که روش سوم، بهترین روش است.

در روش استفاده از سلول‌های حافظه واسطه، هر شیء داده پویای heap ، حاوی سلول خاصی به نام واسطه است که اشاره‌گری به شیء داده heap است.
متغیرهای اشاره‌گر به جای اشاره به شیء داده heap ، به این سلول واسطه اشاره می‌کنند.
وقتی شیءای آزاد شود، سلول واسطه برابر Null می‌شود.

هزینه استفاده از سلول‌های واسطه، از نظر زمان و فضای حافظه گران تمام می‌شود، چون سلول‌های واسطه هیچ‌وقت آزاد نمی‌شوند.

تذکر: طراحان زبان‌های معروف، از این روش استفاده نمی‌کنند.

روش‌های اصلی جمع‌آوری حافظه مازاد

۱- شمارش ارجاع
۲- نشانه گذاری و پیمایش
۳- جمع‌آوری کپی

از بین حافظه‌های مازاد و ارجاع معلق، ارجاع معلق مهم‌تر است.
جمع شدن حافظه‌های مازاد منجر به اتلاف حافظه می‌شود، ولی ارجاع معلق به دلیل تغییر تصادفی حافظه‌ای که در حال استفاده است، منجر به هرج و مرج می‌شود.

ارجاع معلق وقتی به وجود می‌آید که حافظه خیلی زود آزاد شود، ولی حافظه مازاد وقتی به وجود می‌آید که حافظه خیلی دیر آزاد شود.