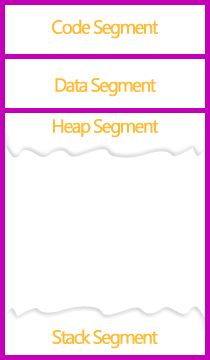
Biến là một thể hiện trừu tượng của vùng nhớ, dữ liệu và cả hàm cũng được lưu trữ trong bộ nhớ. Để có thể làm việc hiệu quả hơn với bộ nhớ, cần nắm bắt những đặc điểm của chúng.

Tổng quan

Khi một chương trình C++ thực thi, có rất nhiều vùng nhớ có thể được sử dụng, nhưng trong đó có 4 vùng nhớ cần lưu ý:

Memory segment của chương trình

Code Segment

Ngôn ngữ lập trình như C/C++, Java và ngôn ngữ máy tính là 2 ngôn ngữ hoàn toàn khác nhau. Khi một chương trình được biên dịch, nó sẽ được chuyển thành các mã máy chỉ bao gồm mã nhị phân 0 và 1.

Đây là vùng nhớ chứa mã máy để thực thi chương trình. Cũng có thể tưởng tượng rằng đây là vùng nhớ lưu trữ câu chuyện kể về chương trình cho máy tính hiểu và người dịch truyện là trình biên dịch. Tuy nhiên, đây là vùng nhớ mang thuộc tính read only nhằm ngăn chặn việc vô tình thay đổi những chỉ thị trong vùng nhớ này khi chương trình thực thi.

Data Segment

Đây là nơi lưu trữ những dữ liệu tĩnh: biến có phạm vi global hoặc các biến static. Thường được chia làm 2 vùng nhỏ hơn là **Initialized Data Segment** và **Uninitialized Data Segment**.

**Initialized Data Segment** chứa giá trị của những biến global và static và được khởi tạo giá trị sớm. Vùng nhớ này có thể mang thuộc tính read only, hoặc read-write tùy thuộc vào cách mà bạn khởi tạo biến này. Chẳng hạn tôi có đoạn chương trình sau:

const char\* \_string = "DataSegment\_stdio\_vn";

int main()

{

return 0;

}

Lúc này, biến \_string sẽ được lưu trong **Initialized Data Segment** với thuộc tính read-write, còn chuỗi DataSegment\_stdio\_vn sẽ được lưu trong vùng nhớ đó nhưng với thuộc tính read only.

**Uninitialized Data Segment** (hay còn gọi là BSS Segment) là vùng nhớ lưu trữ những biến global và static được khởi tạo với giá trị bằng 0 hay chưa được khởi tạo. Tôi có ví dụ sau:

int global\_var\_1;

int main()

{

static int static\_var\_2;

return 0;

}

Khi này, biến global\_var\_1 và static\_var\_2 sẽ được lưu trong **Uninitialized Data Segment**và có giá trị là 0*.*

Stack Segment

Stack có thể được biết đến như một cấu trúc **First In – Last Out** (FILO). Trong bộ nhớ máy tính, Stack là một phân khúc lưu trữ những biến tạm thời được tạo ở trong phương thức (kể cả hàm main()). Phân khúc này được CPU quản lý và tối ưu khá chặt chẽ. Khi một biến được khai báo trong hàm, nó sẽ được push vào Stack, khi hàm đó kết thúc, toàn bộ những biến đã được đẩy vào trong stack sẽ được pop và giải phóng. Vì đặc tính này cho nên biến lưu trên Stack mang tính chất local.

Khi sử dụng biến lưu trên Stack, không cần quan tâm đến việc cấp phát và thu hồi biến đó. Ngoài ra, truy cập ở phân khúc Stack sẽ nhanh hơn một chút so với Heap nhưng bù lại phân khúc này lại có kích thước có giới hạn tùy vào cấu hình với trình biên dịch.

Heap Segment

Heap là phân khúc bộ nhớ cấp phát tự do. Tuy nhiên nó lại không được quản lý tự động, phải tự quản lý toàn bộ những vùng nhớ đã cấp phát trên Heap, nếu không còn sử dụng nữa mà không tự thu hồi sẽ gây ra hiện tượng rò rỉ bộ nhớ - memory leak. Để cấp phát vùng nhớ trên Heap, có thể dùng malloc() – tức là memory allocation hoặc new. Để thu hồi vùng nhớ bạn có thể dùng free() – tức là memory deallocation hoặc delete. Dữ liệu trên Heap do lập trình viên tự quản lý, do đó, nó có thể được sử dụng giữa các phương thức với nhau.

Lời kết

Hiểu về [tổ chức bộ nhớ của chương trình](https://www.stdio.vn/article/to-chuc-bo-nho-chuong-trinh-23BLW1) rất ý nghĩa trong việc nắm rõ cách thức chương trình cấp phát và thu hồi bộ nhớ, giúp có định hướng tối ưu bộ nhớ của chương trình nên đây là kiến thức cần tìm hiểu chuyên sâu.