Stack & Heap

مقدمة : على الرغم من توافر إمكانية ال Garbage Collector **(GC)** الموجودة فى الدوت نت الذي وظيفته إزالة جميع الكائنات التي لم تعد مستخدمة من قبل البرنامج في الذاكرة العشوائية إلا إننا يجب أن نحسن التعامل مع الذاكرة وذلك لتحسين آداء البرنامج ولذلك علينا ان نتعلم كيفية إدارة الذاكرة سواء فى الدوت نت او فى اى لغة خارج هذا النطاق .

ولذلك وضعت هذا الكتيب الذى يحتوى على بعض المعلومات التى يجب ان يعرفها معظم المبرمجين لكى نحصل على التميز فى إنتاج برامجنا وهذه المعلومات تساعدنا فى إنتاج برامج قوية وذات فاعلية أداء أعلى .

نظرة عامة :

إن معظم الثغارات الناتجة عن Buffer Overflow تكون بسبب بعض الاكواد الخبيثة الموجود فى البرامج , حيث أن هذه الاكواد تخزن فى ال Buffer فى مناطق مختلفة فى الذاكرة .

ولذلك سنشرح كيفية إدارة الذاكرة و كيفية عملها .

- نظرة عامة على العمليات فى الذاكرة :

عندما يبدأ تنفيذ البرنامج فإن عناصره المختلفة كالمتغيرات والدوال تخزن فى الذاكرة على شكل بناء منتظم .
حيث تنقسم الذاكرة الى :

مناطق عليا : وهى التى تكون بها بيئة العمليات كال **Arguments : env strings, arg strings,env pointer .**

وجزء اخر ينقسم الى عمليتين فرعيتين :

Stack and Heap

والتى يتم حجزهم **at run time**

- Stack : يستخدم فى حجز

Function arguments , local variables

والتى تعتمد على مبدأ LIFO(Last In ,First Out)
معنى ذلك انها يتم تفريغها أوتوماتيك .ثم يتم إنشاء Pointer يشير إلى أماكن ال Heap .

- الجزء الأخر من الذاكرة يحتوى على

The .bss and .data sections

والتى تحتوى على المتغيرات العامة Global variables

والتى تحجز عند وقت الترجمة Compilation time

1. **.data** : يحتوى على Static initialized data
2. **.bss** : تحتوى على Uninitialized data
3. **.text** : تحتوى على كود البرنامج وهى عبارة عن ذاكرة للقرائة فقط حيث لاتستطيع التعديل فى الكود .

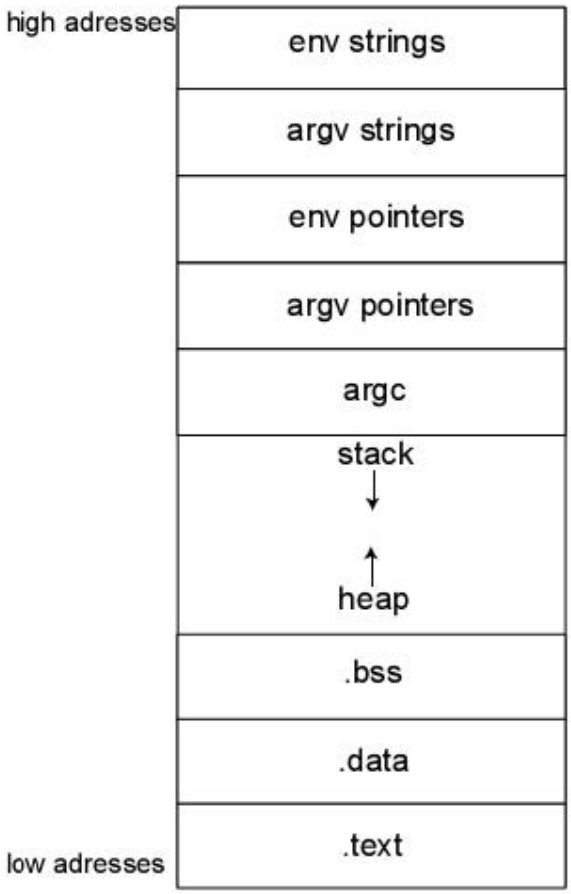

Figure 1.1: Process memory organization

- **Heap**

```
int main(){
char * tata = malloc(3);
...
}
```

tata points to an address which is in the heap.

- **.bss**

```
char global;
int main (){
...
}
```

Or

```
int main(){
static int bss_var;
```

...

}

global and bss_var will be in .bss

- **.data**

```
char global = ’a’;
int main(){
...
}
int main(){
static char data_var = ’a’;
...
}
```

global and data_var will be in .data.

هذا كله مثال على لغة مثل C++ والتى ليس بها GC ووضعت هذا الشرح كى ابين للمبرمج فائدة تعريف المتغيرات وكيفية إستغلال الذاكرة الاستغلال الامثل .

هذه مقارنة بسيطة وجدتها فى إحدى المقالات العربية .

Heap	Stack
جميع البرامج تستخدم (Heap) واحد فقط مثال ذلك (JVM Heap)	لكل برنامج أو بالتحديد لكل (Thread) لها (Stack) واحد
يتم فيه حفظ جميع (Reference Types) مثل كائنات المعرف (Class Objects)	يتم فيه حفظ تعليمات البرنامج
يتم فيه أيضاً حفظ متغيرات المعرف (Instance Variables)	يتم فيه حفظ جميع متغيرات (Primitive Types)
الكائنات الغير مستخدمة يتم إزالتها بواسطة (Garbage Collector)	يتم التخلص من المتغيرات حل الإنتهاء منها بحيث المتغيرات المعرفة في دالة ينتهي عمرها بمجرد الإنتهاء من تنفيذ تلك الدالة أو ما يسمي (Scoping)
ترتيب الكائنات غير مهم	الترتيب مهم لعمل البرنامج بشكل صحيح

م / محمد أسامة محمد خليل

E-mail: EngMohamed.Osama@Yahoo.com