Tutorial 6
Dynamic Memory Allocation

Oct. 22, 2024

Fang Zihao (USTF)

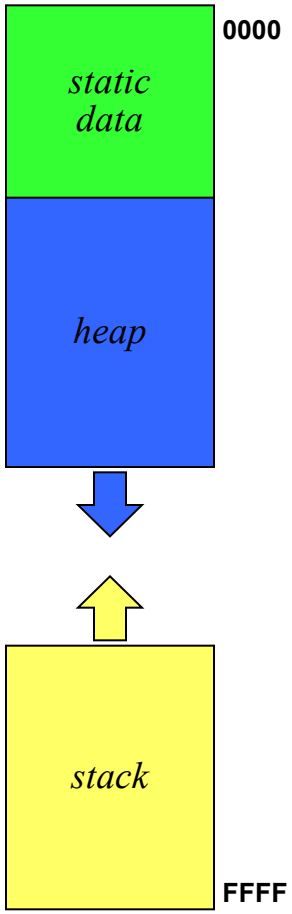
(SDS, 122090106@link.cuhk.edu.cn)

Objectives today

1. Concepts clarification
2. Memory Management Mechanisms
3. C++ Dynamic Memory Allocation
4. Example: Linked-List
5. Supplemental Materials (Not Required)

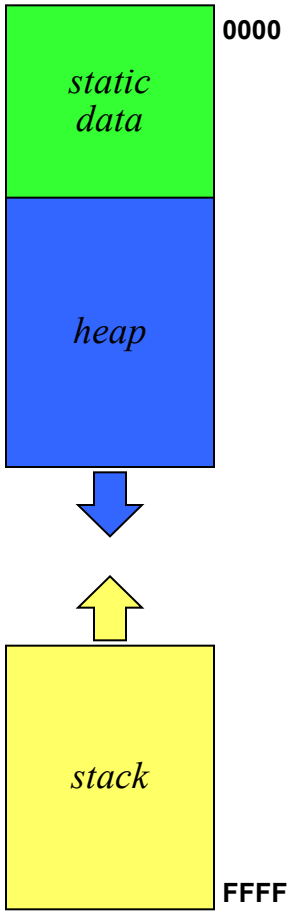
1. Concepts

- Static Data
  - Program codes
  - Global variables/constants
  - Persist throughout the lifetime of the program
- Question: Are we allowed to modify static data?
  - Yes, and No.
  - On Von Neumann Architectures: Yes
  - On Havard Architectures: Only global variables.



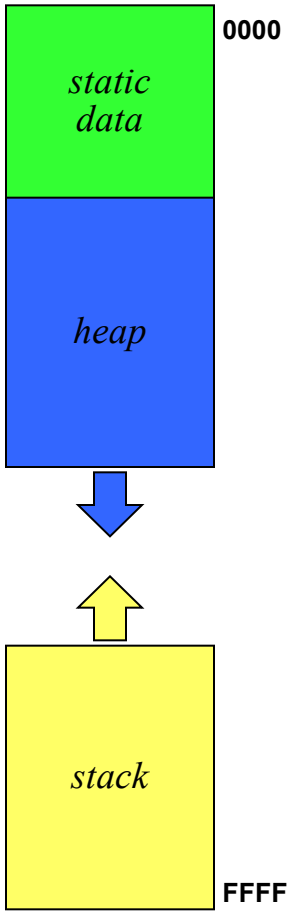
1. Concepts

- Heap
  - Manually allocated and freed.
  - Controlled by programmers
  - In C++, we will use `new` and `delete`
  - In C, we will use `malloc` and `free`



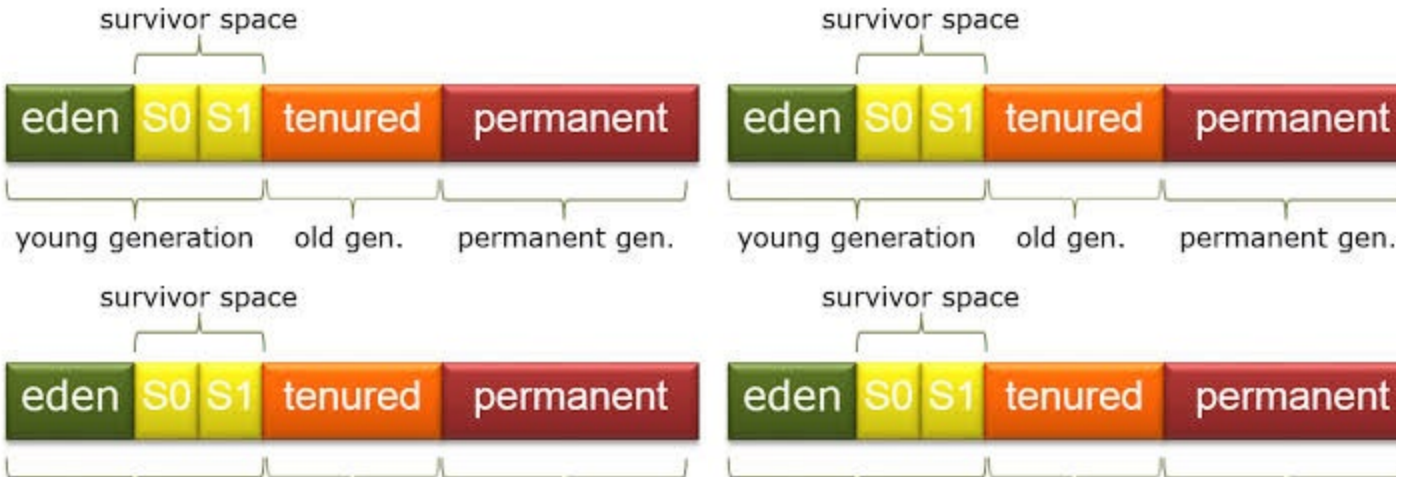
1. Concepts

- Stack
  - Allocated for procedural calls (stack frame)
  - Controlled by programs
  - For local variables



2. Memory Management Mechanisms

- Garbage Collection
  - Programming Languages: Java, Python, C#, etc.
  - Easier for programmers
  - Worse performance
  - Need a runtime to perform garbage collection



2. Memory Management Mechanisms

- Manual Memory Allocation
  - Programming Languages: C, C++, Rust, etc.
  - For better performance
  - Can be used on bare-bone systems.
  - Error-prone for large and complex data management
  - Programs fail to release allocated memory will lead to memory leak.

3. C++ Dynamic Memory Allocation

- Allocation
  - `type *name = new type; // Single variable`
  - `type *name = new type[N]; // Array`
- Allocation with Initial Values
  - `type *name = new type(value);`
  - `type *name = new type(args...); // Object`
  - `type *name = new type{args...}; // The same`
  - `type *name = new type[N](); // Array with default initial values`
  - `type *name = new type[N]{1, 2, 3, ...}; Array with specified initial values;`

3. C++ Dynamic Memory Allocation

- Deallocation
  - `delete name; // Single variable`
  - `delete[] name; // Array`
- Question: What if I delete an array using `delete`?
  - No difference if you want to delete an array of primitive data type!
  - Makes difference if you want to delete an array of objects.

4. Example: Linked-List

```
// Definition

struct int_list {
    int value;
    int_list *next;
};
```

4. Example: Linked-List

```
auto arr = {1, 3, 5, 7, 9};
auto head = new int_list{0, nullptr};
auto it = head;
// Appending values
for (int i : arr) {
    it->next = new int_list{i, nullptr};
    it = it->next;
}
```

4. Example: Linked-List

```
// Inserting values
// Insert "10" as the third node
int pos = 3, value = 10;
it = head;
for (int i = 0; i < pos - 1; ++i) {
    it = it->next;
}
it->next = new int_list{value, it->next};
```

4. Example: Linked-List

```
// Deleting values
// Delete the second node
pos = 2;
it = head;
for (int i = 0; i < pos - 1; ++i) {
    it = it->next;
}
auto tmp = it->next->next;
delete it->next;
it->next = tmp;
```

4. Example: Linked-List

```
// Printing values
it = head;
while (it != nullptr) {
    cout << it->value << endl;
    it = it->next;
}
```

4. Example: Linked-List

```
// Clearing
it = head;
while (it != nullptr) {
    auto tmp = it;
    it = it->next;
    delete tmp;
}
```

5. C Dynamic Memory Allocation

- Allocation
  - `type *name = (type*)malloc(sizeof type); // Single variable`
  - `type *name = (type*)malloc(N * sizeof type); // Array`
- No way to allocate with initial values!
- Deallocation
  - `free(name); // Free a single variable or array.`

5. Smart Pointers

- Smart Pointers:

Pointer categories

unique_ptr (C++11)	smart pointer with unique object ownership semantics (class template)
shared_ptr (C++11)	smart pointer with shared object ownership semantics (class template)
weak_ptr (C++11)	weak reference to an object managed by `std::shared_ptr` (class template)

- Use `get` method to get the raw pointer.

For more, please visit https://en.cppreference.com/w/cpp/memory#Smart_pointers

5. Smart Pointers

- Question: How to construct a linked-list using unique_ptr?
- See linked_list_unique.cpp!

7. Q & A time

- Thank you for your listening!

- Fang Zihao (USTF)
- (SDS, 122090106@link.cuhk.edu.cn)