LOBACHEVSKY UNIVERSITY

Advanced C++

4. Task: 3Sum.

Compiling, linking and execution (14.11.2017)

Sidnev A.A.

3Sum

LOBACHEVSKY UNIVERSITY

Task: Given an array *S* of *n* integers, are there elements *a*, *b*, *c* in *S* such that $a + b + c = 0$?

Find all unique triplets in the array which gives the sum of zero.

Example: given array S = [-1, 0, 1, 2, -1, -4],

A solution set is:

[

[-1, 0, 1],

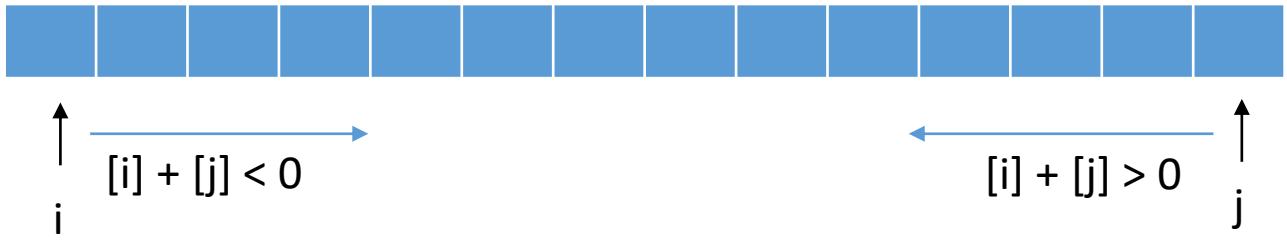
[-1, -1, 2]

]

3Sum: Reduce to 2Sum

- Time complexity: O(n^2).
- Sort the array and reduce task to 2Sum problem. Move first pointer k from left to right and solve 2Sum problem for other elements: [i] + [j] = -[k].

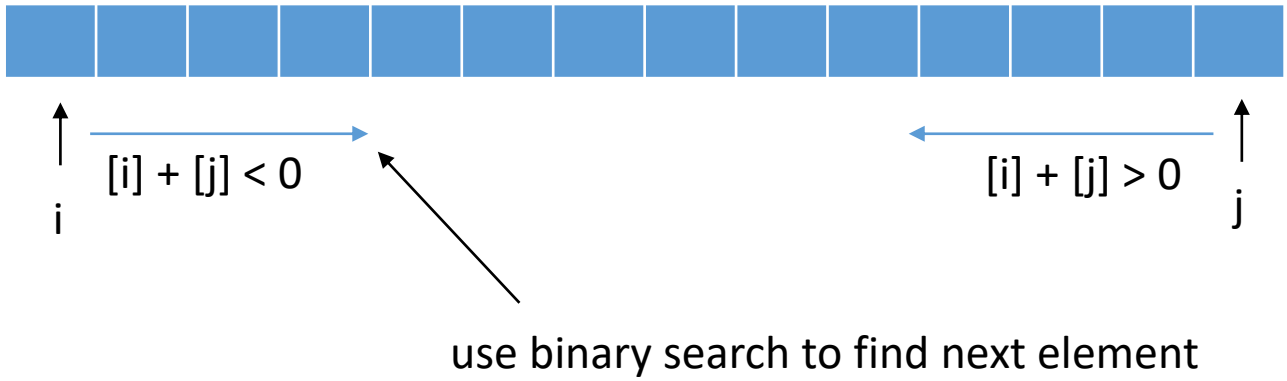
- 2Sum problem with 2 pointers (array already sorted).



3Sum: Reduce to 2Sum + binary search

- Time complexity: O(n^2 logn).
- Sort the array and reduce task to 2Sum problem. Move first pointer k from left to right and solve 2Sum problem for other elements: [i] + [j] = -[k].

- 2Sum problem with 2 pointers and binary search (array already sorted).



3Sum: hash tables

LOBACHEVSKY UNIVERSITY

- Time complexity: O(n^3).
- Use unorderd_map to store elements and their quantity.
- Two loops iterate over elements i and j. The third element -[i] -[j] is found in the unordered_map.

3Sum: permutations

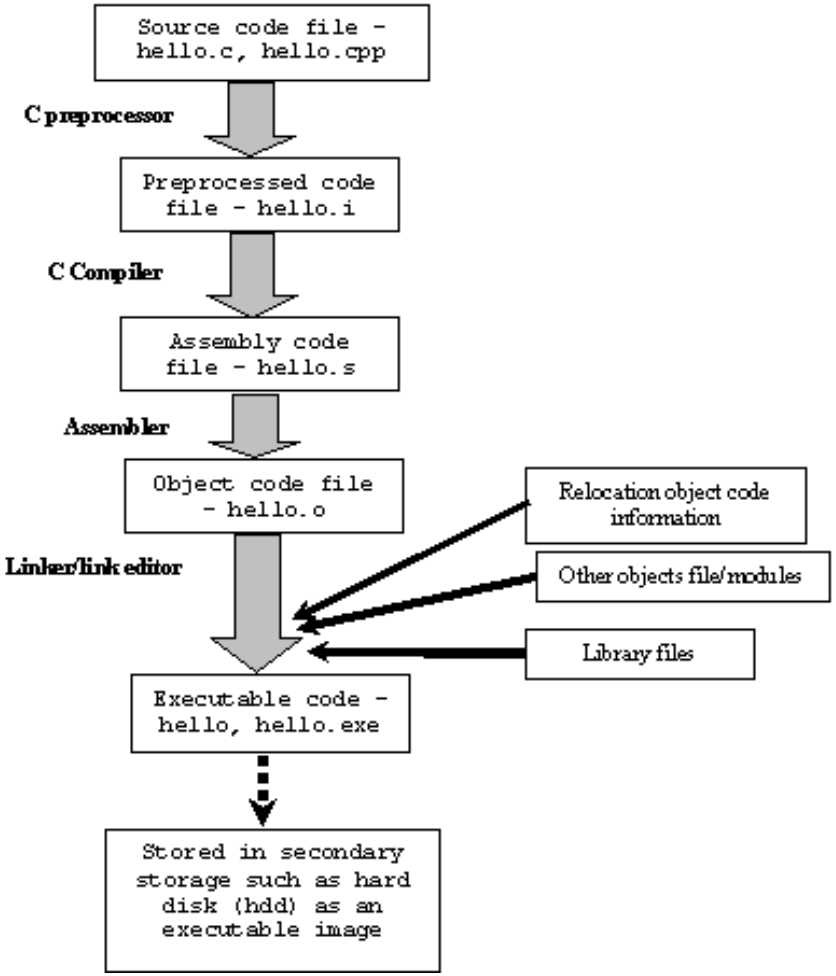
LOBACHEVSKY UNIVERSITY

- Time complexity: O(n!).

```
vector<vector<int>> threeSum(vector<int>& nums) {
  auto nsize = nums.size();
  if (nsize < 3)
    return vector<vector<int>>();
  vector<vector<int>> output;
  sort(nums.begin(), nums.end());
  do {
    if (nums[nsize - 1] + nums[nsize - 2] + nums[nsize - 3] == 0) {
      vector<int> v = { nums[nsize - 1], nums[nsize - 2], nums[nsize - 3] };
      sort(v.begin(), v.end());
      output.emplace_back(v);
    }
  } while (next_permutation(nums.begin(), nums.end()));
  sort(output.begin(), output.end());
  output.erase(unique(output.begin(), output.end()), output.end());
  return output;
}
```

Compilers, Assemblers and Linkers

1. Preprocessor
   - include-files
   - conditional compilation instructions
   - macros
2. Compiler
   - const/constexpr
3. Assembler
4. Linker



http://www.tenouk.com/ModuleW.html

Object file

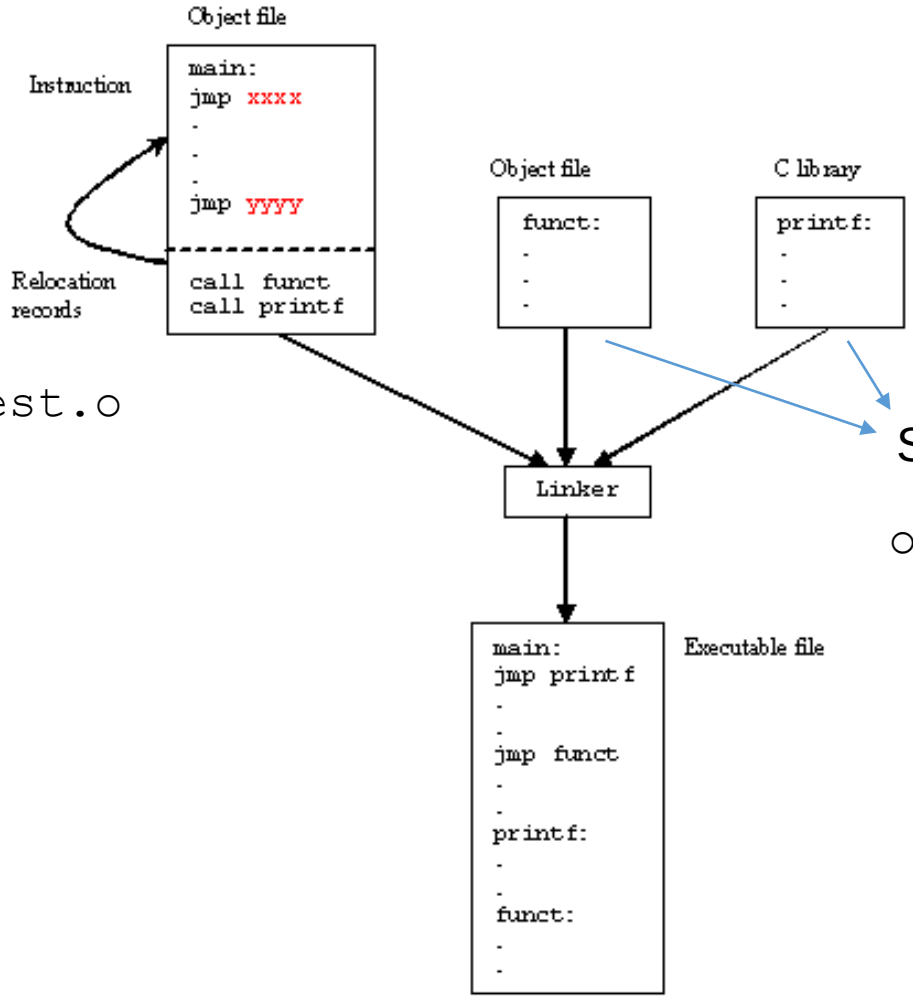
Section	Short description
.text	Executable instruction codes, shared among every process running the same binary. READ/EXECUTE.
.bss	Block Started by Symbol. It holds un-initialized global and static variables. Doesn't take up any actual space in the object file.
.data	Contains the initialized global and static variables and their values. It is usually the largest part of the executable. READ/WRITE.
.rdata	Also known as .rodata (read-only data) section. This contains constants and string literals.
.reloc	Stores the information required for relocating the image while loading.
Symbol table	Symbol table holds information needed to locate and relocate a program's symbolic definitions and references.
Relocation records	Relocation records are information used by the linker to adjust section contents.

http://www.tenouk.com/ModuleW.html

Linker

- Most assembly instructions are easily translated into machine code using a one-to-one correspondence
- But in our program we declared **labels** for addresses
  - Addresses in the .bss and the .data segments
  - Addresses in the .text segments (for jumps)
- **Question**: How should the assembler translate instructions that use these labels into machine code?
  - E.g., add [L], ax
  - E.g., call my_function
- **Answer**: it cannot do the full job without knowing the "whole" program so as to determine addresses

Relocation records



`objdump -r test.o`

Symbol tables

`objdump -t test.o`

http://www.tenouk.com/ModuleW.html

objdump (old example)

LOBACHEVSKY UNIVERSITY

```cpp
// A.cpp -> A.obj
#include <iostream>
using std::cout;
using std::endl;

namespace A {
int x1;
int x2 = 1;
extern int x3;
extern int x4;
extern const int x5;

int f() {
  cout << x1 << endl;
  cout << x2 << endl;
  cout << x3 << endl;
  cout << x4 << endl;
  cout << x5 << endl;
  return 5;
}
}  // namespace A
```

```cpp
// B.cpp -> B.obj
float x1;

namespace A {
int x3 = 2;
int x4;
extern const int x5 = 1;

int f();
}  // namespace A

int main() {
  x1 = A::f();
}
```

objdump (symbol table)

// A.o

0000000000000000 g O .bss
0000000000000004 _ZN1A2**x1**E

0000000000000000 g O .data
0000000000000004 _ZN1A2**x2**E

0000000000000000 g F .text
00000000000000a6 _ZN1A1**f**Ev

// B.o

0000000000000000 g O .bss
0000000000000004 **x1**

0000000000000000 g O .data
0000000000000004 _ZN1A2**x3**E

0000000000000004 g O .bss
0000000000000004 _ZN1A2**x4**E

0000000000000000 g O .rodata
0000000000000004 _ZN1A2**x5**E

0000000000000000 g F .text
0000000000000020 **main**

0000000000000000 ***UND***
0000000000000000 _ZN1A1**f**Ev

objdump (relocation records)

// A.o

OFFSET TYPE VALUE

0000000000000006 R_X86_64_PC32 _ZN1A2**x1**E-0x0000000000000004

000000000000000d R_X86_64_32 _ZSt4**cout**

0000000000000025 R_X86_64_PC32 _ZN1A2**x2**E-0x0000000000000004

000000000000002c R_X86_64_32 _ZSt4**cout**

…

// B.o

OFFSET TYPE VALUE

0000000000000005 R_X86_64_PC32 _ZN1A1**f**Ev-0x0000000000000004

0000000000000015 R_X86_64_PC32 **x1**-0x0000000000000004

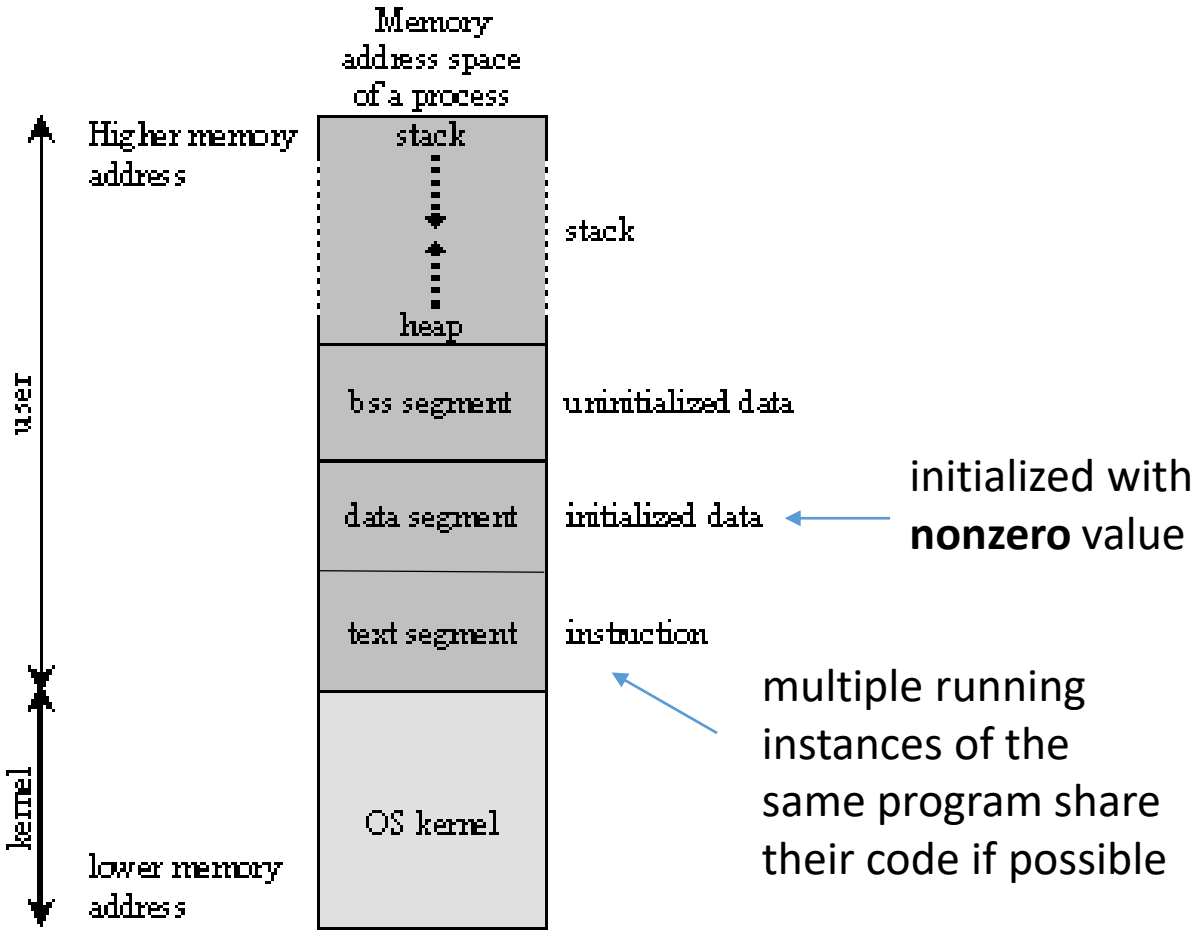
Linker

- Step 1: concatenate all the text segments from all the .o files
- Step 2: concatenate all the data/bss segments from all the .o files
- Step 3: Resolve references
  - “symbol not found”
  - “multiply defined”

The loader

- Read the executable file's header to find out the size of the text and data segments
- Creates a new address space for the program that is large enough to hold the text and data segments, and to hold the stack (within some bounds)
- Copies the text and data segments into the address space
- Copies arguments passed to the program on the stack. Initializes the registers
  - Clear most of them, set ESP to the top of the stack
- Jump to a standard "start up routine", which sets the PC and calls the exit() system call when the program terminates

Loading executable

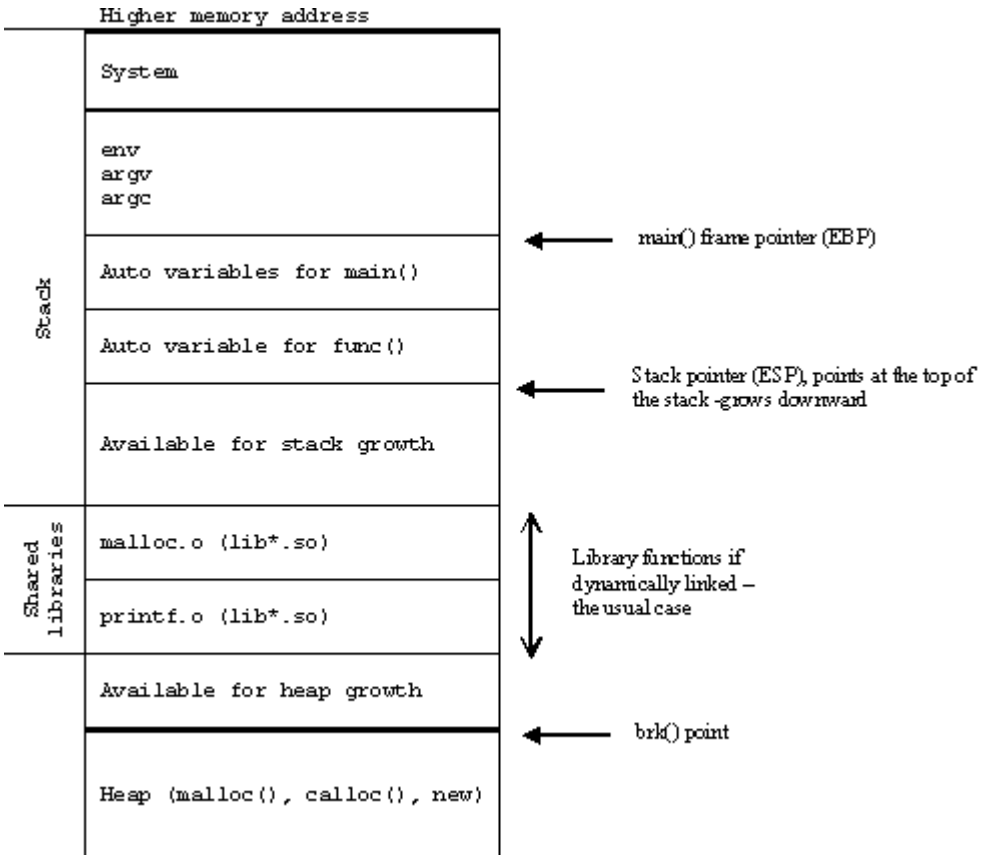


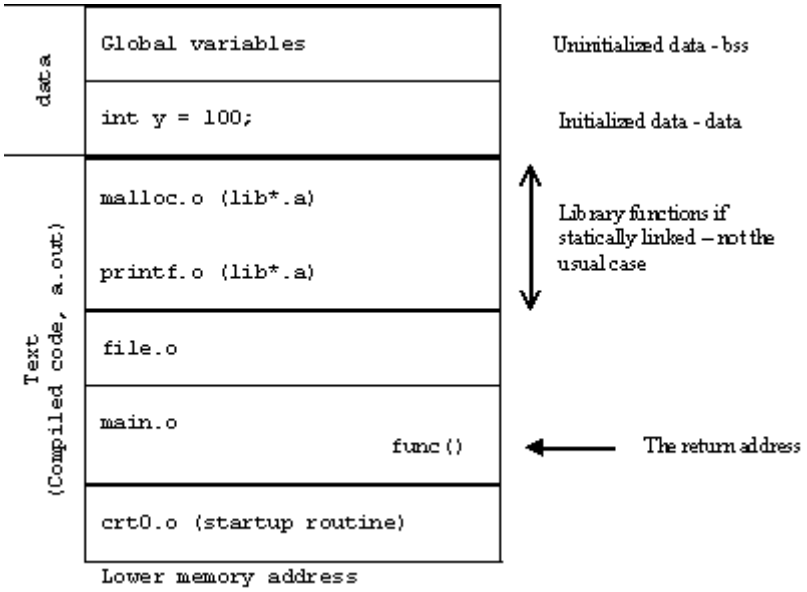
initialized with **nonzero** value

multiple running instances of the same program share their code if possible

http://www.tenouk.com/ModuleW.html

Execution





http://www.tenouk.com/ModuleW.html