Programming in C and C++ - Supervision 1

Nandor Licker

October 2019

1 Types, Variables, Expressions and Statements

Q1 Can `sizeof(long)` return 1? What other types must be 1 byte in size otherwise?

Q2 Why is `for (int i = 0; i < n; ++i)` problematic, if `n` was declared `unsigned`?

Q3 How do you compare a signed integer to an unsigned one safely?

Q4 Can you think of a 9 character C program which results in `SIGSEGV`?

2 Functions and the Preprocessor

Include-based programming is often used to deduplicate repetitive code. This pattern, hated by some, appreciated by others, relies on a header file invoking a macro with arguments to define various items:

```
    #define X(1)
    #define X(2)
    #define X(3)
```

Whenever needed, the macro is defined and the header is included:

```
    int a[] = {
    #define X(x) x
    #include "header.h"
    #undef X
};
```

For a more complete example, take a look at the following files from `clang`, located at `https://github.com/llvm/llvm-project`:

- `clang/include/clang/AST/OperationKinds.def`
- `clang/include/clang/AST/OperationKinds.h`

Imagine you are working on an exciting business application managing vegetables.

- Create a definition file enumerating vegetables and fruits, with separate macros for both vegetables and fruits. Enumerate at least 5 of each. Kudos of you teach me about new fruts or vegetables.
- Define an enum with fields for all vegetables and fruits.
- Define two methods: `is_fruit` and `is_vegetable` which check whether their parameter is a fruit or a vegetable.

3 Pointers and Structures

Q1 Instead of using a pointer and a dimension to refer to arrays or sections of arrays, a pair of pointers can be used. The number of elements can be computed using the pointer difference operator. Provide an example where the use of this operator results in undefined behaviour. Hint: use `https://godbolt.org` to try different architectures.

Q2 A neat trick with certain pointers is to exploit pointer alignment, which is of at least 8 bytes on 64-bit architectures: all pointers aligned to 8 bytes have their last 3 bits set to 0, allowing them to be used for good or evil. Mostly evil.

- Show how to pack a flag and a pointer into 8 bytes, as well as how to unpack them in C.
- Define a C++ template class which packs a pointer to some record type and a boolean flag into 8 bytes, with the relevant accessor methods. What property must the pointee type satisfy? How can it be checked?

Q3 Provide `typedef`s for the following types:

- A pointer to an array of ten functions which take an integer argument and return a pointer to functions returning pointers to read-only integers. You are allowed to hate me for this.
- A read-only pointer to an array of pointers to characters.
- A pointer to a function returning a function returning a function returning an integer.
- A pointer to a structure of two int fields. Do not define the structure separately. How can you use this?

Q4 Consider the following piece of code:

```
struct X {
    unsigned a : 1;
    unsigned b : 1;
};
int test(int p) {
    struct X x;
    (p < 0 ? x.a : x.b) = 1;
}
```

Explain what is happening. Kudos if you point to the C11 standard. Using `https://godbolt.org`, find as many compilers as you have patience for which fail to compile this code. Explain why some compilers fail more horribly than others.

4 Tips, Tricks and the Tick

Q1 Because I like to spread chaos in the world, I am going to introduce you to `alloca`:

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

 __attribute__((always_inline)) char *copy_string(const char *str) {
    size_t len = strlen(str) + 1;
    char *buffer = (char *)alloca(len);
    memcpy(buffer, str, len + 1);
    return buffer;
}

int main() {
    puts(copy_string("1234"));
}
```

Special thanks to JF Bastien for pointing out this use of alloca to me in a certain OS kernel.

Consult the alloca man page of any standard library/compiler. Does the use of `alloca` in the program shown result in undefined behaviour? Does the program leak memory? What is the difference between `alloca` and declaring `char buffer[len + 1]`? Why should `alloca` not be used arbitrarily? When would you use `alloca`?

Q2 Write a function which detects if the machine is little or big endian. Return a nice enum.

Q3 Write a method which swaps the endiannes of an integer.

Q4 Explain why it might not be a good idea to use Duff's device when building with a modern compiler.