When to use C and when to use Python

Justus Perlwitz

2017-07-05

Contents

I've now solved the first 15 Project Euler challenges in C. But then, I've hit a road block. Let me explain. Let's say you want to find the sum of digits for the number `3^300`. Would you use C or Python?

1 Python

In Python, it is relatively easy:

```
sum(map(int, str(3**300)))
```

2 C

To achieve the same thing in C you would need:

1. A bigint library
2. A bigint -> string conversion function
3. Use manual memory allocation
4. Worry about printing the result correctly

All this without ever having the chance to express on a high level:

> Calculate `3^300`, find the digits, convert them to integers, sum them.

Instead, C would require you to say:

> Initialize my bigint library, start the calculation. Check whether the calculation did not run out of memory. Then, carefully execute a loop that uses the bigint -> string conversion routine to retrieve single bigint characters. Then, convert these characters back to integers, and then finally sum the integers.

While you can always create a nice high level abstraction inside of C, the drawback is flexibility. While you carefully crafted all this code within an hour or so, using Python allows you to solve 20 similar problems in the same time, while providing superior memory safety, readability and maintainability.

This comes with some very obvious drawbacks:

1. The Python code is much slower. It requires tons of Python object meta-data and requires garbage collection.
2. The C code can be optimized much more easily by using the correct compiler flags.
3. The C code can easily interface with assembly instructions that might be offered on your CPU to offer further speedups.

The Compromise

A great solution that a lot of Python projects find is to have the high level API run in Python, while the low-level hot loops are written in C. One of the most famous projects that do it this way would be NumPy. It contains C code for extremely efficient array packing, but still exposes a powerful Python interface to achieve a lot in a relatively small amount of LOC.