DOI:10.1145/1897852.1897870

Article development led by acmqueue
queue.acm.org

Why can't we all use standard libraries for commonly needed algorithms?

BY POUL-HENNING KAMP

B.Y.O.C (1,342 Times and Counting)

ALTHOUGH SELDOM ARTICULATED clearly, or even at all, one of the bedrock ideas of good software engineering is reuse of code libraries holding easily accessible implementations of common algorithms and facilities. The reason for this reticence is probably because there is no way to state it succinctly, without sounding like a cheap parody of Occam's razor: *Frustra fit per plura quod potest fieri per pauciora* (it is pointless to do with several where few will suffice).

Obviously, choice of programming language means that "few" will never be "a single one," and until somebody releases a competent implementation under an open source license, we may have several more versions floating around than are strictly necessary, for legal rather than technological reasons.

It also never hurts to have a few competing implementations to inspire improvement; in fact, there seems to be a distinct lack of improvement where a single implementation becomes too "golden."

So much for theory. Does any of this hold in practice?

One of the nice side effects of the "software tools" concept is that programs are data, too. We can apply data mining methods to program source code, allowing us to investigate such questions.

Cryptography algorithms provide a good example because they are easier to identify than other algorithms. Magic numbers in crypto algorithms make for good oracular answers to their presence: you are not likely to encounter both 0xc76c51a3 and 0xd192e819 anywhere other than an implementation of SHA-2. Creating an oracle to detect sorting algorithms in source code with ($p>0.9$) would be a good student project (albeit, likely impossible).

For data mining FOSS (free and open source software) programs, the FreeBSD operating system ships with a handy facility called the Ports Collection, containing strategic metadata for 22.003 pieces of FOSS. A small number of these "ports" are successive versions of the same software (Perl 5.8, Perl 5.10, among others), but the vast majority are independent pieces of software, ranging from trivialities such as XLogo to monsters such as Firefox and OpenOffice.

A simple command downloads and extracts the source code to as many ports as possible into an easily navigated directory tree:

```
cd /usr/ports ; make -k extract
```

You will obviously need both sufficient disk space and patience. (Using `cd /usr/ports ; make -k -j 10 extract` will do 10 pieces of software in parallel, but will be a bandwidth hog.)

The results are worse. I had not expected to see 1,342, as shown in the accompanying table.[a] I expect that these numbers will trisect my readers into three somewhat flippantly labeled seg-

a Sorry, I forgot to include the DES algorithm in the search.

Crypto algorithms search results.

Cryptographic Implementations

Algorithm	Detected
MD2	6
MD4	49
MD5	920
SHA-1	136
SHA-2	192
AES	39
Total	**1,342**

ments: “huh?,” “sigh,” and “aghast.”

The “huh?” segment wonders what the big deal is: the absence of a standardized system library with these functions means that you have to “Bring Your Own Crypto” if you want some.

The “sigh” segment thinks this is the least of our troubles.

The “aghast” segment will see this as a total failure of good software engineering practices, a call to arms for better education, and reason for a stake through the heart of the Open Zombie Group.

And they are all correct, of course, each from its own vantage point.

Fortunately, what this is not, is The Next Big Security Issue, even though I would not be surprised if one or more “security researchers” would claim so from their parents’ basement.[b] If these had been independent implementations, then there would be reason to worry about the security implications, but they are not.

In a few cases, optimized or license-sanitized versions have been written, but overwhelmingly this is just pointless copy-and-paste of identical source code in blatant disregard of Occam’s three-quarters-millennia-old advice.

I am a card-carrying member of the “aghast” segment. My membership card is a FreeBSD commit message shown in the figure here.

My libmd, which is as unencumbered by copyright issues as it can be, later grew more cryptographic hash algorithms, such as RIPEMD-160 and the SHA family, and it has been adopted by some other operating systems.

I am also in the “sigh” segment, because not all mainstream operating systems have adopted libmd, despite having 16 years to do so, and if they have, they do not agree what should be in it. For example, Solaris seems to leave MD2 out (see http://hub.opensolaris.org/bin/view/Project+crypto/libmd), which begs the question: Which part of “software portability” don’t they understand?

I am, sadly, also in the “huh?” segment, because there seems to be no hope. The rational thing to expect would be that somebody from The Open Group reads this article, reproduces my statistics, and decides that yes, there is indeed demand for a “libstdcrypto” filled with the usual bunch of crypto algorithms. That, I am told, is impossible. The Open Group does not write new standards; they just bicker over the usability of `${.CURDIR}` in `make(1)` and probably also the potential market demand for fire that can be applied nasally.

The other possible avenue of hope is that the ISO-C standardization group would address this embarrassing situation. Before getting your hopes too high, bear in mind they have still not managed to provide for specification of integer endianness, even though CPUs can do it and hardware and protocols have needed it since the days of the ARPANET.

If the ISO-C crew decided to do it, their process for doing so would undoubtedly consume 5–10 years before a document came out at the other end, by which time SHA-3 would likely be ready, rendering the standard instantly obsolete.

But it is all a pipe dream, if ISO is still allergic to standards with ITAR restrictions. And you can forget everything about a benevolent dictator laying down the wise word as law: Linus doesn’t do userland.

To be honest, what I have identified here is probably the absolutely worst-case example.

First, if you need SHA-2, you need SHA-2, and it has to do the right and correct thing for SHA-2. There is little or no room for creativity or improvements, apart from performance.

Second, crypto algorithms are everywhere these days. Practically all communication methods, from good old email over VPNs (virtual private networks) and torrent sites to VoIP (voice over IP), offers strong crypto.

But aren’t those exactly the same two reasons why we should not be in this mess to begin with?

b The fact that MD5 seems to be more in demand—yes, I may indeed be to blame for that myself, but that is a story for another day; search for “md5crypt” if you cannot wait—than its quality warrants is a matter of choice of algorithm, not a matter of implementation of the algorithm chosen.

A card-carrying member of the “aghast” segment.

```
src/lib/libmd/Makefile:

r1802 | phk | 1994-07-24 03:29:56 +0000 (Sun, 24 Jul 1994)


Imported libmd. This library contains MD2, MD4, and MD5.

These three boggers pop up all over the place all of the time, so I

decided we needed a library with them. In general, they are used for

security checks, so if you use them you want to link them static.
```

Related articles on queue.acm.org

Languages, Levels, Libraries, and Longevity
John R. Mashey
http://queue.acm.org/detail.cfm?id=1039532

Gardening Tips
Kode Vicious
http://queue.acm.org/detail.cfm?id=1870147

Poul-Henning Kamp (phk@FreeBSD.org) has programmed computers for 26 years and is the inspiration behind bikeshed.org. His software has been widely adopted as “under the hood” building blocks in both open source and commercial products. His most recent project is the Varnish HTTP accelerator, which is used to speed up large Web sites such as Facebook.