Qlibs

- and it's hunchbacked relatives -

Dr. Erwin Hoffmann

February 25, 2018

Qmail – 1995

When *Dan Bernstein* was a Phd Student at the University of Berkeley (LA)[1], he invented as replacement for the standard and error prone *'C'* `<stdio.h>` library functions, in particular

- **printf**
- **scanf**

save variants to together with a set of basic functions needed for network communications, including a IP(v4) address parser and some higher-level socket libraries together with a DNS stub resolver library, first as part of

- Qmail[2] (1.03), and
- ucspi-tcp[3] (0.88) and later
- djbdns[4] (1.05)

However, though the source code was freely available and modifiable, distribution (and modification) of those packages was restricted; which resulted in a condemn of *Bernstein* by the community.

[1] `https://de.wikipedia.org/wiki/Daniel_J._Bernstein`
[2] `http://cr.yp.to/qmail.html`
[3] `http://cr.yp.to/ucspi-tcp.html`
[4] http://cr.yp.to/djbdns.html

Fefe – 2001

Since distribution and modification of restricted, *Felix von Leiter* reimplemented the basic library functions as **libowfat**[5] in 2001.

In 2002, Felix added IPv6 capabilities into it, which now serves as a skeleton for a lot of other SW projects.

The **libowfat** library is still maintained and now available in version 0.31.

[5] https://www.fefe.de/libowfat/

Public domain – 2007

In 2007[6], Dan Bernstein – while not maintaining his SW any more – released all of this code into the *public domain*[7].

[6] http://cr.yp.to/qmail/dist.html

[7] https://www.heise.de/newsticker/meldung/Qmail-ist-Public-Domain-201769.html

qlibs – 2017

Kai Peter, developer & maintainer of **eQmail** and **OpenQmail**.

Kai and me agreed to work on a version of Qmail – **aqmail**[8]– based on the **qlibs** and picking up basic ideas of my package **s/qmail**[9].

[8]`http://aqmail.org`

[9]`http://www.fehcom.de/sqmail.html`

qlibs Content /1

The qlibs include DJB's data operators in 'C':

- **stralloc** – dynamical and save string operations
- **case** – case independent string manipulation/evaluation
- **scan** – string to integer conversion
- **fmt** – ASCII representation of strings and integers
- **byte** – byte manipulation/evaluation

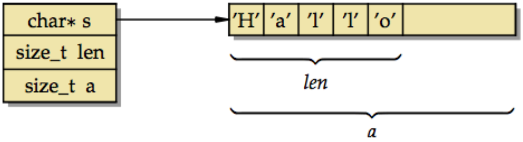


Figure: struct: stralloc[10]

↪ stralloc needs to be initialized: **stralloc ss = {0};**, len+1 = 'Z' if not 'terminated' **stralloc_0{&ss};**, ss may include '\0'.

[10] `http://www.mathematik.uni-ulm.de/sai/ws17/soft1/ss1-folien.pdf`

qlibs Content /2

A couple of data structures are supported:

- **tai** – Temps Atomic International
- **cdb** – constant data base, hash-based and machine independent

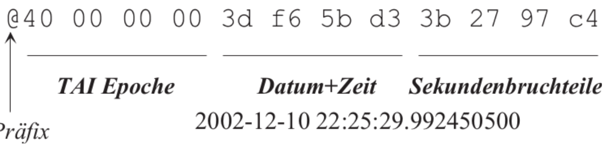


Figure: Explanation of the TAI data format

qlibs Content /3

Reading and writing is facilitated by only very few routines:

- **buffer** – reading data from FD 0, or writing to FD 1, FD 2 ...
- **getln** – read in one line of data
- **getoptb** – get option character from command line

Process support is very rudimentary only:

- **pathexec** – run a program within a given environment
- **env** – manage variables in the environment
- **fd** – duplicate or move a descriptor

qlibs Content /4

IPv4 and **IPv6** (parsing of IP addresses) was originally only given on a *label level*:

- inet **192.168.192.31**
- inet6 **2002:5b14:20cf:0:21e:90ff:fead:5a07**

CIDR support was not given: **10.0.0.0**/**17**.

↪ Within **ucspi-tcp6**[11] and **ucspi-ssl**[12], I realized CIDR support in particular for the **cdb** (containing IPv4/IPv6 addresses) based on a Bachelor thesis at the *Frankfurt University of Applied Sciences*.

[11] http://www.fehcom.de/ipnet/ucspi-tcp6.html
[12] http://www.fehcom.de/ipnet/ucspi-ssl.html

qlibs Content /5

The qlibs introduce an extended concept and usage for error codes within an applications:

```
                                          /* Comparison of error codes and constants:
                                               intern   Linux  FreeBSD  OmniOS         */
#define error_intr           EINTR        /*     -1        4        4       4   */
#define error_nomem          ENOMEM       /*     -2       12       12      12   */
#define error_noent          ENOENT       /*     -3        2        2       2   */
#define error_txtbsy         ETXTBSY      /*     -4       26       26      26   */
#define error_io             EIO          /*     -5        5        5       5   */
#define error_exist          EEXIST       /*     -6       17       17      17   */
#define error_timeout        ETIMEDOUT    /*     -7      110       60     145   */
#define error_inprogress     EINPROGRESS  /*     -8      115       36     160   */
#define error_wouldblock     EWOULDBLOCK  /*     -9   EAGAIN   EAGAIN  EAGAIN   */
#define error_again          EAGAIN       /*    -10       11       35      11   */
#define error_pipe           EPIPE        /*    -11       32       32      32   */
#define error_perm           EPERM        /*    -12        1        1       1   */
#define error_acces          EACCES       /*    -13       13       13      13   */
#define error_nodevice       ENODEV       /*    -14      (6)      (6)      19   */
#define error_proto          EPROTO       /*    -15       71       92      71   */
#define error_isdir          EISDIR       /*    -16       21       21      21   */
#define error_connrefused    ECONNREFUSED /*    -17      111       61     146   */
//extern int error_notdir;                /*    -18       20       20      20   */
#define error_rofs           EROFS        /*    -19       30       30      30   */
```

qlibs Content /6

The **socket** for TCP and UDP communication have been extended to IPv4 and IPv6 sockets with a common call.

- Based on the KAME[13] project, BSD (and Solaris) include IPv6 as 'super-set' of IPv6 and IPv4 within a common network stack.
- For this particular reason, IPv6-mapped IPv4-addresses were introduced: **::ffff:10.2.3.4**.
- The OpenBSD developer – however – considered those as 'dangerous'[14] and by today, IPv6-mapped IPv4-addresses are avoided by most Unix implementations.

```
    #include 'ip.h'
    #include 'socket_if.h'

     ipv4socket = ip6_isv4mapped(ip);
```

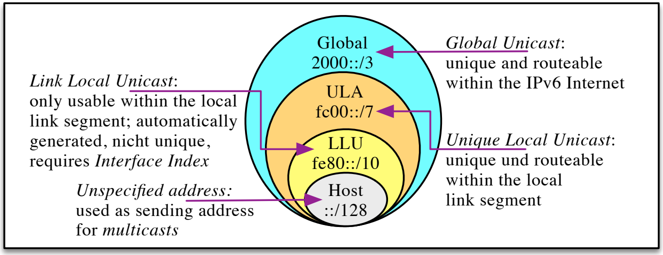
[13] http://www.kame.net

[14] https://stackoverflow.com/questions/32051957/ipv6-why-are-ipv4-mapped-addresses-a-security-risk

qlibs Content /7

In order to support not only route-able IPv6 addresses, but in addition so-called Link-Local Unicast (LLU) addresses, we need to understand the meaning of Interface Identifier, the (remote) LLU address is reachable thru:

- Unlike IPv4 – even with private addresses – IPv6 allows to define the very same address on different interfaces.
- In order to distinguish those, IPv6 has introduced the concept of an **Interface Identifier**, which in it's enumerated form is called the **Interface Index**.
- The IPv6 address hierarchy is strictly based on the very first bits given.


Global
2000::/3
ULA
fc00::/7
LLU
fe80::/10
Host
::/128
Global Unicast: unique and routeable within the IPv6 Internet
Link Local Unicast: only usable within the local link segment; automatically generated, nicht unique, requires *Interface Index*
Unique Local Unicast: unique und routeable within the local link segment
Unspecified address: used as sending address for *multicasts*

↪ Some consequence:

- **::1** is the 'unscoped' loopback address.
- **fe80::1%lo0** is the 'scoped' loopback address on the loopback interface.

qlibs Content /8

The qlibs include an enhanced **DNS stub resolver** library:

- Specification of up to 32 IPv4 and IPv6 Name Servers.
- Support for the application specific environment variable DNSCACHEIP along side with /etc/resolv.conf.
- 'Obfuscated' – stealth – Name Servers sitting in your *link-local* segment are addressable by means of the provided **Interface Identifier**: **fe80::53%eth0**.

```
DNSCACHEIP="10.0.1.53 fe80::1%lo0 ::1"
```

qlibs Questions?

Questions???

Let's install!!!

Projects based on DJB's lib

- s/qmail
- ucspi-tcp6
- ucspi-ssl
- djbdns/6/curve6
- tinydnssec (Peter Conrad)
- DNSCurve (Matthew Dempsky?)
- tinyldap (Fefe)
- gatling (Fefe)
- eQmail (Kai Peter)