The story of a **bluetooth connection**

Ferenc Fabian
infinite loop

Ferenc Fabian

PumpkinSeed

Gopher, Rustacean, Hobby Hacker

Edit profile

96 followers · **175** following

@infiniteloopcloud
Amsterdam
13:49 (UTC +02:00)
qwer.kocka@gmail.com
https://medium.com/@ferencfbin

Pinned

Customize your pins

golang/go Public
The Go programming language
Go ☆ 113k ⑂ 16.8k

zgendao/ethane Public
Forked from th4s/ethane
Ethane is an alternative web3 implementation with the aim of being slim and simple.
Rust ☆ 49 ⑂ 6

cosmos/cosmos-sdk Public
⛓ A Framework for Building High Value Public Blockchains ✨
Go ☆ 5.3k ⑂ 2.9k

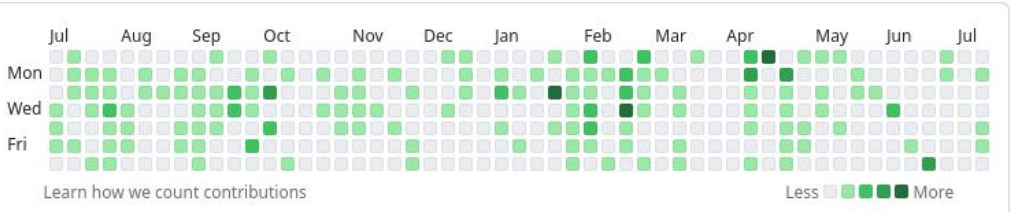
fakeit Public
Fake data generator library with 130+ functions written in Rust
Rust ☆ 43 ⑂ 6

sqlfuzz Public
Simple SQL table fuzzing
Go ☆ 152 ⑂ 16

simplerand Public
Simple and fast random number generator based on linear congruential generators
Rust ☆ 7 ⑂ 2

578 contributions in the last year

Contribution settings



2023
2022
2021
2020
2019

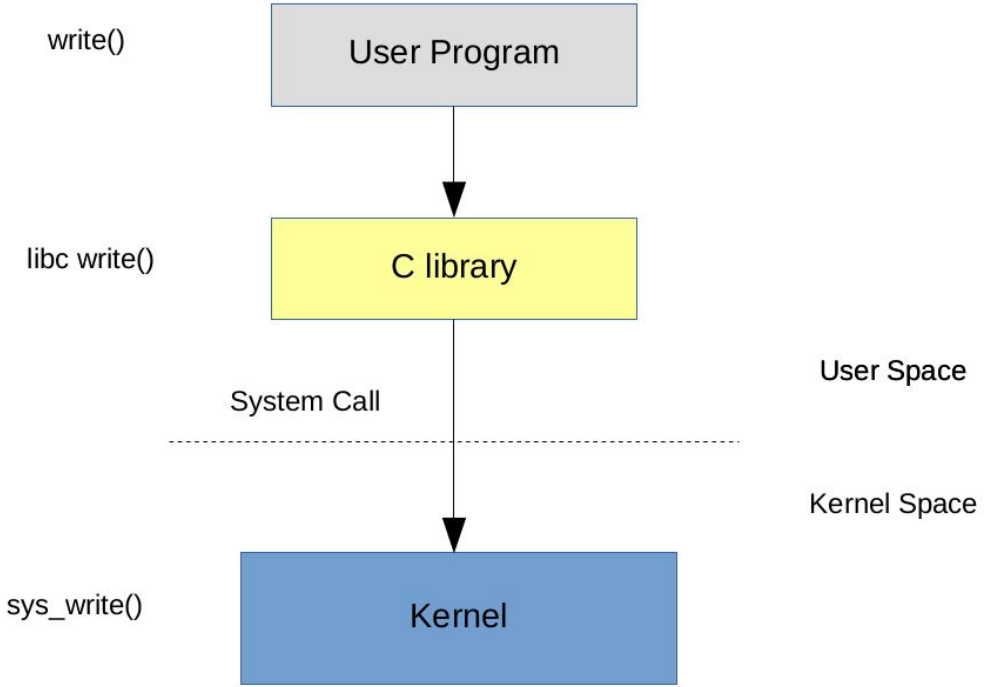
Agenda

- Defining the problem
- Solving the problem
- Solving the problem for everyone

Defining the problem

- Pax D135
- No documentation
- Only a barely working APK

Syscalls


write()
User Program
libc write()
C library
User Space
System Call
Kernel Space
sys_write()
Kernel

Syscalls - Linux

zsyscall_linux_amd64.go

```
425 }
426
427 // THIS FILE IS GENERATED BY THE COMMAND AT THE TOP; DO NOT EDIT
428
429 func connect(s int, addr unsafe.Pointer, addrlen _Socklen) (err error) {
430   _, _, e1 := Syscall(SYS_CONNECT, uintptr(s), uintptr(addr), uintptr(addrlen))
431   if e1 != 0 {
432     err = errnoErr(e1)
433   }
434   return
435 }
```

Syscalls - Linux

```
func Connect(params Params) (Communicator, error) {
        fd, err := unix.Socket(unix.AF_BLUETOOTH, unix.SOCK_STREAM, unix.BTPROTO_RFCOMM)
        if err != nil {
                return &bluetooth{
                        log:            params.Log,
                        FileDescriptor: fd,
                        Addr:           params.Address,
                }, err
        }
        params.Log.Print("unix socket returned a file descriptor: ", fd)
        socketAddr := &unix.SockaddrRFCOMM{Addr: addressToByteArray(params.Address), Channel: 6}
        if err := unix.Connect(fd, socketAddr); err != nil {
                return &bluetooth{
                        log:            params.Log,
                        FileDescriptor: fd,
                        SocketAddr:     socketAddr,
                        Addr:           params.Address,
                }, err
        }
        params.Log.Print("unix socket linked with an RFCOMM")
```

Syscalls - Windows

```
fd, err := windows.Socket(windows.AF_BTH, windows.SOCK_STREAM, windows.BTHPROTO_RFCOMM)
if err != nil {
        return &bluetooth{
                log:    params.Log,
                Handle: fd,
                Addr:   params.Address,
        }, err
}
```

```
s := &windows.SockaddrBth{
        BtAddr: addressUint64,
        Port:   6,
}
if err := windows.Connect(fd, s); err != nil {
        return &bluetooth{
                log:    params.Log,
                Handle: fd,
                Addr:   params.Address,
        }, err
}
params.Log.Print("unix socket linked with an RFCOMM")
```

Scanning - Linux

```
func (scanner) Scan() ([]Device, error) {
        out, err := exec.Command("hcitool", "scan").Output()
        if err != nil {
                return nil, fmt.Errorf("hcitool scan: %s", err.Error())
        }
```

Scanning - Windows

PumpkinSeed commented on Aug 3, 2022 Tip ...

Brief overview

Adding the following structs and functions:

- WSAQUERYSET The parameter of the following functions. It has other functions in this function which is not linked individually.
- WSALookupServiceBegin
- WSALookupServiceNext
- WSALookupServiceEnd

WSAQUERYSET

```
1   typedef struct _WSAQuerySetA {
2       DWORD dwSize;
3       LPSTR lpszServiceInstanceName;
4       LPGUID lpServiceClassId;
5       LPWSAVERSION lpVersion;
6       LPSTR lpszComment;
7       DWORD dwNameSpace;
8       LPGUID lpNSProviderId;
9       LPSTR lpszContext;
10      DWORD dwNumberOfProtocols;
11      LPAFPROTOCOLS lpafpProtocols;
12      LPSTR lpszQueryString;
13      DWORD dwNumberOfCsAddrs;
14      LPCSADDR_INFO lpcsaBuffer;
15      DWORD dwOutputFlags;
16      LPBLOB lpBlob;
17  } WSAQUERYSETA,*PWSAQUERYSETA,*LPWSAQUERYSETA;
```

Go memory layout

```
1  type T1 struct {
2      a int8
3
4      // On 64-bit architectures, to make field b
5      // 8-byte aligned, 7 bytes need to be padded
6      // here. On 32-bit architectures, to make
7      // field b 4-byte aligned, 3 bytes need to be
8      // padded here.
9
10     b int64
11     c int16
12
13     // To make the size of type T1 be a multiple
14     // of the alignment guarantee of T1, on 64-bit
15     // architectures, 6 bytes need to be padded
16     // here, and on 32-bit architectures, 2 bytes
17     // need to be padded here.
18 }
```

C memory layout

`#pragma pack` specifically is used to indicate that the struct being packed should not have its members aligned. It's useful when you have a memory mapped interface to a piece of hardware and need to be able to control exactly where the different struct members point. It is notably not a good speed optimization, since most machines are much faster at dealing with aligned data.

Go solution for memory alignment

1. Raw byte array
2. Struct with byte arrays
3. Struct with pointers and byte arrays
4. Struct using the types from sys/windows

Testing with unsafe.Pointer

```
3311    type WSAQUERYSET struct {
3312            Size                uint32
3313            ServiceInstanceName *uint16
3314            ServiceClassId      *GUID
3315            Version             *WSAVersion
3316            Comment             *uint16
3317            NameSpace           uint32
3318            NSProviderId        *GUID
3319            Context             *uint16
3320            NumberOfProtocols   uint32
3321            AfpProtocols        *AFProtocols
3322            QueryString         *uint16
3323            NumberOfCsAddrs     uint32
3324            SaBuffer            *CSAddrInfo
3325            OutputFlags         uint32
3326            Blob                *BLOB
3327    }
```

Syscalls - MacOS

- Not supported in our case
- It’s way different than Linux and Windows
- Implementation of CBCentralManagerDelegate

Our own bluetooth-go

bluetooth-go Public

This is a raw bluetooth library which connects to a MAC address by using syscall on Linux and Windows.

Go 2 MIT 0 0 0 Updated on Feb 20

Golang Pull Request on Gerrit

Future plans for the bluetooth-go

- Change the hcitool
- Implement the MacOS part

Thanks! Questions?

- github.com/PumpkinSeed
- linkedin.com/in/ferencfabian

- go-review.googlesource.com/q/owner:qwer.kocka@gmail.com
- github.com/infiniteloopcloud/bluetooth-go