株式会社 ITS MORE
2020年4月設立

Digitally signed by 佐藤 豊
Date: 2020.08.13 19:24:31 +09'00'

2020年8月13日 投稿者: SATOXITS

続々々々々続GShell — 動的リンク

開発：さて今日は、GShellの動的リンク機能にチャレンジしたいと思います。

社長：タイムスリッパでやってたのがちょうど一ヶ月前ですね。

開発：それがもう、恐ろしいくらいに何も覚えてないのですが、ブログと、残されたコードを頼りにして行きたいと思います。

基盤：一ヶ月前というとあの血染めの帰宅の頃ですが、あの時脳震盪で記憶が飛んだとか？

開発：それでまず、Goプログラム自体を動的リンクモードで生成するという話が気になっていたのでこれを…

```
MacMini% go build -linkshared hello.go
-linkshared not supported on darwin/amd64
```

開発：そうですか。ではLinuxで。

```
jp1$ gsh
!1! go build -o hello-st hello.go
--I-- Aug 13 07:39:05(0.071806989s)
!2! go build -o hello-dy -linkshared hello.go
--I-- Aug 13 07:39:26(2.518295855s)
!3! ./hello-st
Go-Hello GShell!
--I-- Aug 13 07:39:59(0.001106968s)
!4! ./hello-st
Go-Hello GShell!
--I-- Aug 13 07:40:01(0.001145272s)
!5! ./hello-dy
Go-Hello GShell!
--I-- Aug 13 07:40:09(0.022653034s)
!6! ./hello-dy
Go-Hello GShell!
--I-- Aug 13 07:40:10(0.022031093s)
!7! hi -l
!0   Aug 13 07:39:05 71.881054ms/t 0.053845s/u 0.014769s/s go build -o he
!1   Aug 13 07:39:23 2.518336593s/t 2.000691s/u 0.293146s/s go build -o h
!2   Aug 13 07:39:59  1.139488ms/t 0.000820s/u 0.000174s/s ./hello-st
!3   Aug 13 07:40:01  1.172903ms/t 0.000000s/u 0.001025s/s ./hello-st
!4   Aug 13 07:40:09 22.683319ms/t 0.010582s/u 0.010768s/s ./hello-dy
!5   Aug 13 07:40:10 22.060834ms/t 0.016688s/u 0.004349s/s ./hello-dy
--I-- Aug 13 07:40:12(0.000086621s)
```

社長：尋常では無い遅さですね。

開発：gshell自体に1度だけ動的リンクしようという話と方向違いですからね。毎回動的リンクをやってるわけで。まあでも、macOSだと3ミリ秒のところが、Linuxなら1ミリ秒で起動するようです。

開発：さてそれで、発掘しましたタイムスリッパ。7月13日05:50のタイムスタンプのあるlibgasket.so.1 です。これをかませてやると1970年にタイムスリップするはず…

```
MacMini% gsh
!1! date
Thu Aug 13 08:07:31 JST 2020
--I-- Aug 13 08:07:31(0.002259270s)
!2! quit
--I-- Aug 13 08:07:32(0.000009611s)
MacMini%
MacMini% DYLD_INSERT_LIBRARIES=./libgasket.so.1 gsh
!1! date
Thu Aug 13 08:07:48 JST 2020
--I-- Jan  1 09:10:44(0.003175317s)
```

開発：あれ？

基盤：しないですね。

開発：そういえばmacOSの場合には、ビルド時になにかやる必要があったような…

社長：いや、Jan 1 09:10:44 (1970) にスリップしてるんでは。

開発：あそうか…　/bin/dateは動的ライブラリでスリップしてくれないという話でしたね。というか、さっきから DYLD_ 設定しなくても勝手にスリップするんですが…かと思うと他の端末ではそういう事はない…

社長：動的ライブラリのキャッシュっぽいですね。

開発：確かに。タイムスリッパ版 gettimeofday がキャッシュされていると思われるような症状です…　そういえば DYLD_ほにゃららにキャッシュを無効にするというのがあったような…　man dyld。まずはトレースですね。DYLD_PRINT_LIBRARIES= DYLD_PRINT_BINDINGS= gsh ・・・

```
MacMini% DYLD_PRINT_LIBRARIES= DYLD_PRINT_BINDINGS= gsh
dyld: loaded: /Users/ysato/gsh/gasket/mac/gasket1.5/tmp/gsh
dyld: loaded: <C912577C-4968-39DB-BB13-4A44A51ED218> ./libgasket.so.1
dyld: loaded: <001B3B7F-D02C-31D3-B961-1ED445D5A266> /usr/lib/libSystem.B.dylib
dyld: loaded: <C0D70026-EDBE-3CBD-B317-367CF4F1C92F> /System/Library/Frameworks/CoreFoundation.fra
dyld: loaded: <EB9E9E2A-B53B-36EE-B9CE-EEB99B603CB1> /System/Library/Frameworks/Security.framework
dyld: loaded: <5940876E-AC8A-3BE0-80B3-DE3FB14E257A> /usr/lib/system/libcache.dylib
dyld: loaded: <C095BD55-1D27-337F-9B02-885E1C7FF87A> /usr/lib/system/libcommonCrypto.dylib
dyld: loaded: <6E80AC11-A277-31FA-AEEF-E5A528274C77> /usr/lib/system/libcompiler_rt.dylib
dyld: loaded: <EB5E0BC8-873D-3546-A40E-C36DC46FA8F6> /usr/lib/system/libcopyfile.dylib
dyld: loaded: <0B6C52DB-5A50-3FCD-8B5E-C0C2F35857E3> /usr/lib/system/libcorecrypto.dylib
dyld: loaded: <EAD535EE-1270-39A9-A254-95CF117FF3B0> /usr/lib/system/libdispatch.dylib
dyld: loaded: <24C41E8B-6B33-30C7-94C9-02D2BD051D66> /usr/lib/system/libdyld.dylib
dyld: loaded: <6F582FDB-EB1A-3ED2-A989-B750643E2647> /usr/lib/system/libkeymgr.dylib
dyld: loaded: <AFBCBDD3-0B55-3ECD-8E04-A73A3A57356B> /usr/lib/system/liblaunch.dylib
dyld: loaded: <1B0296B5-3FD0-342C-BCC2-9886351A4391> /usr/lib/system/libmacho.dylib
```

基盤：うわわわ…

開発：発見しました。

```
dyld: bind indirect sym: gsh:_gettimeofday$non_lazy_ptr = libgasket.so.1:_gettimeofday, *0x015fe21
```

社長：なるほど。macOSでは動的ライブラリが細分されているんですごい数をロードする、それで起動に3ミリ秒もかかるということですかね。

開発：ちょっと邪魔なのでdyldの出力は別ファイルに出します。DYLD_PRINT_TO_FILE=dylog。うーむ、起動するまでに120ファイルの .dyld をロードしてます。1ファイル25マイクロ秒でロードできちゃってると…　まあ、オンメモリだからなんでしょうけど。

基盤：macos dyld cache clear で検索。どうも実体は /private/var/db/dyld の下にありますね。1.4GBもあるんだがどうしたらいいんだみたいな質問が。うちのMacMiniはどうかと言うと…

```
MacMini% ls -l
total 4109528
-r--r--r--  1 root  wheel  2103640064 Jul 23 23:04 dyld_shared_cache_x86_64h
-rw-r--r--  1 root  wheel      435627 Jul 23 23:04 dyld_shared_cache_x86_64h.map
```

一同：（苦笑）

基盤：でもこの中にgasketは無いんですよね。これはシステムのdylib専用のキャッシ

ュなんですかね？こっちは3ヶ月使ってきたMacMiniですが、まだあんまり活動してない iMacはというと。

```
iMac% ls -l
total 4106216
-r--r--r--  1 root  wheel  2101944320 Jul 22 21:16 dyld_shared_cache_x86_64h
-rw-r--r--  1 root  wheel      435157 Jul 22 21:16 dyld_shared_cache_x86_64h.map
```

開発：タイムスタンプも大きさもほぼ同じですね。要するに Catalina 10.15.6 に共通ということですかね。何が違うんでしょう？

基盤：.map のdiffを取ると、まあアドレスは連鎖的に変わるでしょうけど、大きな違いは2つdylibが加わったということのようです。

```
*** 472,473 ****
--- 472,474 ----
  /System/Library/PrivateFrameworks/AirPlaySync.framework/Versions/A/AirPlaySync
+ /System/Library/PrivateFrameworks/AirTrafficHost.framework/Versions/A/AirTrafficHost
  /System/Library/PrivateFrameworks/AmbientDisplay.framework/Versions/A/AmbientDisplay
***************
*** 713,714 ****
--- 714,716 ----
  /System/Library/PrivateFrameworks/DeviceIdentity.framework/Versions/A/DeviceIdentity
+ /System/Library/PrivateFrameworks/DeviceLink.framework/Versions/A/DeviceLink
  /System/Library/PrivateFrameworks/DeviceManagement.framework/Versions/A/DeviceManagement
```

社長：で、プライベートなキャッシュはどこにあるんですかね。端末というか、セッションごとにあるように見えますが。

基盤：ところでこのdyldって、なんて読むんですかね。ディル…　って読むとヤバいと思うんですがw

開発：単にダイエルディーって読んでましたが。

開発：ああそれで、dyldinfo -bind gsh で、動的にリンクされているシンボルを眺めたのですが、やはり差し替えたら面白いと思うのは、ファイル名を解釈する部分、つまりopenとexecveかなと思います。ホスト名についてはgetaddrinfoとgetpeername。これで名前空間をいじって簡易マウントする。それから、出力をteeしちゃうという意味ではwriteとsend。入力をteeするという意味では read と recvfrom。

社長：端末入出力の記録がわかりやすいので、read / write から行きましょう。

開発：了解。ではまず、gettimeofdayのラッピングは外して…

基盤：今日に戻りました(^-^)

開発：ああ、どうもGoで区切り記号を省略する癖がついてCでもやっちゃいますねｗ。むむ、read をラップして __read を呼ぶという戦法は Macのリンカーには通じないのかな…　とりあえず read としてエラーを返すことにします。こんな感じ。

```
#include <errno.h>
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t nbyte){
        fprintf(stderr,"read(%d,%d)(^-^;\n",fd,(int)nbyte);
        errno = EBADF;
        return -1;
}
```

開発：で、これを動的ライブラリにしてgshにリンクさせると。

```
MacMini% gsh
read(3,8)(^-^;
read(3,128)(^-^;
read(4,8)(^-^;
!1! date
--F-- getline process error (fork/exec /Users/ysato/gsh/gsh-getline
: bad file descriptor)
read(4,4096)(^-^;
--I-- Aug 13 01:50:14(0.000008003s)
```

基盤：入れ替わってますね。

開発：さてでは、どうやって本物に中継するかですが。システムコールなら syscall でイケばy位とは思いますが。

基盤：7/12のタイムスリッパでも、Linux は func を入れ替えて本物は __funcを呼ぶでイケる、macOSは不明なのでペンディングということになってますね。

開発：うーん、dyldinfo を見ても、__ を前置するという流儀のように見えるんですが…

```
MacMini% dyldinfo -bind gsh
binding information (from external relocations and indirect symbol table):
segment      section          address        type   weak  addend dylib            symbol
__DATA       __nl_symbol_ptr  0x015FE020    pointer           0 flat-namespace   ___error
__DATA       __nl_symbol_ptr  0x015FE028    pointer           0 flat-namespace   _getaddrinfo
__DATA       __nl_symbol_ptr  0x015FE030    pointer           0 flat-namespace   _freeaddrinfo
__DATA       __nl_symbol_ptr  0x015FE038    pointer           0 flat-namespace   _gai_strerror
__DATA       __nl_symbol_ptr  0x015FE040    pointer           0 flat-namespace   ___stack_chk_guard
```

基盤：なんとなく、macOS固有の拡張シンボルにつけてるようにも見えますが。

社長：なんにしても、動的ライブラリがシンボルとしてエクスポートしてないとどうにもならないですね。

開発：Linux GCCではどうなっているかと言うと…

```
jp1$ ldd gasket
        linux-vdso.so.1 =>  (0x00007ffdc1b89000)
        libgasket.so.1 => not found
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f1fc44bc000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f1fc40ee000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f1fc46c0000)
jp1$ nm -D /lib64/libc.so.6|grep gettime
0000000000114800 T __clock_gettime
00000000000b5650 i __gettimeofday
00000000003c8a40 B __vdso_clock_gettime
0000000000114800 W clock_gettime
00000000000b5650 i gettimeofday
00000000000c0bb0 T ntp_gettime
00000000000c0c00 T ntp_gettimex
00000000000ff3e0 T timerfd_gettime
```

開発：この i というのは GNU の拡張で indirection とかの意味だそうです。あるいは、read 関係。

```
jp1$ nm /lib64/libc.so.6|grep read'$'|grep -v __GI|grep -v _IO
00000000000ef1d0 t __exit_thread
00000000000ee090 t __libc_pread
00000000000ef9b0 t __libc_read
00000000000ee090 t __pread
00000000000ef9b0 W __read
00000000000b6d00 t __tzfile_read
00000000000febd0 T eventfd_read
0000000000079130 t fmemopen_read
000000000006f0f0 W fread
0000000000170ad0 t freeres_libptread
00000000000f33f0 T fts_read
00000000000ee090 W pread
00000000000ef9b0 W read
```

社長：もともと Weak になってたりするんですね。というか、fread は差し替えたら自作しなさいってことですかね？？

開発：ポリシーが全くわからないですね。そもそもどういう意図でこういう2種類を提供しているのかも。

開発：どうも macOSでも同じような感じですね。タイムスリッパのgettimeofdayがうまく行ったのはこうなっていたから。

```
MacMini% nm /usr/lib/system/libsystem_kernel.dylib|grep gettimeofday
00000000000025f9 T ___commpage_gettimeofday
0000000000002600 t ___commpage_gettimeofday_internal
0000000000003e01 T ___gettimeofday
0000000000003dec t ___gettimeofday_with_mach
MacMini% nm /usr/lib/system/libsystem_c.dylib|grep gettimeofday
                 U ___commpage_gettimeofday
                 U ___gettimeofday
00000000000108b6 T _gettimeofday
000000000008dee8 b _gettimeofday.cached_tz.0
000000000008deec b _gettimeofday.cached_tz.1
000000000008dee4 b _gettimeofday.validtz
```

開発：一方で read はこんな感じなのでダメ。

```
MacMini% nm /usr/lib/system/libsystem_c.dylib|grep read'$'
000000000003a6d5 T ___fread
000000000004027d T ___sread
00000000000402bb T __sread
000000000004f212 t _eofread
000000000004f71b t _eofread
00000000000590f0 t _fmemopen_read
000000000003a68e T _fread
000000000007b0a0 T _fts_read
MacMini% nm /usr/lib/system/libsystem_kernel.dylib|grep read'$'
00000000000018b18 T __kernelrpc_mach_vm_read
00000000000007d60 T __kernelrpc_vm_read
00000000000021424 T _aio_read
000000000001f1a5 T _etap_trace_thread
00000000000011933 T _mach_vm_read
000000000000227c T _pread
00000000000001814 T _read
000000000000b97e T _semaphore_signal_thread
00000000000007d11 T _vm_read
```

開発：あれ？でも libsystem_c.dylib にはやっぱり、___symbol シリーズがありますね。ほとんどのシステムコールに…　なぜ read には無いのか？あ、これですかね？

```
00000000000007584 T ___ptrace
00000000000210b4 T ___pwrite_nocancel
00000000000002694 T ___read_nocancel
00000000000210cc T ___readv_nocancel
00000000000210e4 T ___reboot
000000000000439c T ___recvfrom
```

社長：その noncancel 版の仕様が公開されているかですが…　というか、readv を呼ぶのが一番簡単では。

開発：そうですね。これでどうかな。

```
#include <errno.h>
#include <unistd.h>
#include <sys/uio.h>
ssize_t read(int fd, void *buf, size_t nbyte){
        int rcc;
        struct iovec iov = { buf, nbyte };
        rcc = readv(fd,&iov,1);
        fprintf(stderr,"readv(%d,%d)=%d(^-^;\n",fd,rcc,(int)nbyte);
        return rcc;
}
```

開発：でもってgshを起動…

```
MacMini% gsh
readv(10,1)=1(^-^;
readv(10,1)=1(^-^;
readv(10,1)=1(^-^;
readv(10,1)=1(^-^;
readv(3,8)=8(^-^;
readv(3,128)=128(^-^;
readv(4,0)=8(^-^;
!1! date
readv(3,4)=8192(^-^;
readv(3,0)=8(^-^;
Thu Aug 13 12:08:38 JST 2020
readv(3,292)=4096(^-^;
readv(3,0)=4096(^-^;
--I-- Aug 13 12:08:38(0.002810319s)
readv(4,0)=8(^-^;
!2!
```

基盤：大成功！

開発：明るい未来が見えた気がした。

社長：いけそうですね。

開発：どうもmacOSの動的ライブラリのキャッシュの挙動がわからないですが、原理的には問題ないようですね。

＊　＊　＊

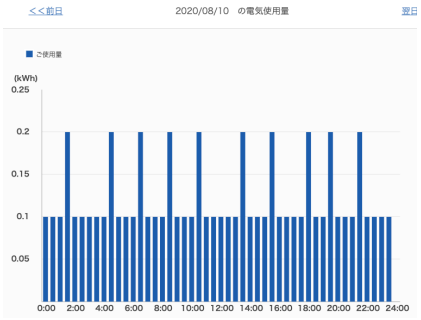
社長：帰りました。外は今日は普通に夏日よりでした。

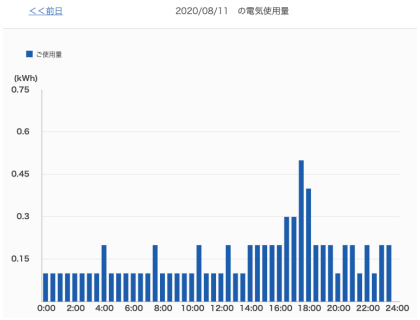
社長：きょうはひや麦でした。あれは、カップ麺では実現できない世界ですね。

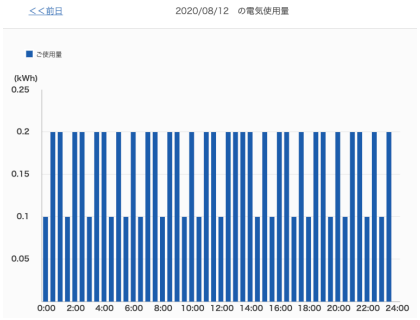
基盤：室内も34.4度にとどまっています。

社長：ところてんを買ってきました。冷蔵庫で冷やして食べましょう。

経理：その冷蔵庫なんですが、少し深刻な事態に。これを見て下さい。







経理：昨日までの3日間ですが、一昨日に冷蔵庫の電源を入れる前は200W＋、一昨日は洗濯機の稼働もありましたが、昨日の様相を見ると300Wには達していると見られます。つまり、冷蔵庫が100W消費しているのではないかと。

開発：まあ100Wで生活が豊かになるならそれはそれで良いかなとも思いますけどね。

基盤：ところでさきほどベランダに打ち水というか、バケツ一杯の水をぶちまけたので

すが、0.1度ばかり室温が下がったように見えます。

＊　＊　＊

社長：さてそれでは、動的ライブラリをリンクして呼び出すのをやってみましょうかね。

開発：それが、syscall に dlopen が無いようなのです…

社長：そういえば、たしかに関数名や変数名が動的にリンクできたとしても、相手とスタックの構造が違うとどうなるんだという事はあるかもですね。

開発：ですが、これまでやったように、gettimeofdayやらreadやらのCのライブラリは普通に動的リンクできて呼び出せちゃうわけなんですが…

社長：つまりわれわれとしては、何かのCの標準関数のふりをして繋がる作戦ですか。

開発：標準的には pluginというパッケージがあるようではあります。これだと、通常のgoプログラムが取り込める。で、ソースファイルに書いてある例のとおり にやったのですが、なぜか関数名をリゾルブしてくれないんです。go build -buildmode=plugin hello.go とすると hello.so という動的ライブラリが出来るので、nm hello.soで見てみるわけですが、その関数名が出てこない。

関数：それでもういちど例をじっと眺めたのですが、例では関数名が大文字なんですね。まさかと思って大文字にしたら、通りましたw。つまり、Goでは大文字で始まる名前だけがエクスポートされるっていう流儀なんだと思います。で、こんなふうになりました。

```
MacMini% gsh
!1! plugin hello aa bb cc
Go-Hello GShell!([hello aa bb cc])
--I-- Aug 13 18:14:15(0.002955741s)
!2! plugin hello aa bb cc
Go-Hello GShell!([hello aa bb cc])
--I-- Aug 13 18:14:16(0.000078243s)
!3! plugin hello aa bb cc
Go-Hello GShell!([hello aa bb cc])
--I-- Aug 13 18:14:18(0.000084711s)
!4! hi -l
!0   Aug 13 18:14:15   3.06786ms/t 0.002304s/u 0.000859s/s plugin hello aa bb cc
!1   Aug 13 18:14:16     91.8µs/t 0.000038s/u 0.000056s/s plugin hello aa bb cc
!2   Aug 13 18:14:18    97.496µs/t 0.000047s/u 0.000067s/s plugin hello aa bb cc
```

開発：一度目の呼び出しはいつものように3ミリ秒かかっていますが、2度目以降は100マイクロ秒以下で終わっています。

基盤：やった！**30倍速**！

開発：まあ、2回目以降はいきなりプロセス内部での呼び出しですからね。

開発：ちなみに中身はこのようなものです。

```
MacMini% cat hello.go
package main
import (
        "fmt"
        "os"
)
func main(){
        fmt.Printf("Go-Hello GShell!(%v)\n",os.Args)
}
func Main(){
        main()
}
```

```
MacMini% go build -buildmode=plugin hello.go
MacMini% ls -l hello.so
-rw-r--r--  1 ysato  staff  3381400 Aug 13 18:20 hello.so
MacMini% nm hello.so|grep Main
00000000000bc280 T _plugin/unnamed-1af01d1d26fd00b6262fc89250cfc597edf423c3.Main
0000000000116a80 S _plugin/unnamed-1af01d1d26fd00b6262fc89250cfc597edf423c3.Main.stkobj
```

社長：素晴らしい。これなら、内部コマンド同様に使えますね。内部コマンドでも30usくらいかかりますからね。100マイクロ秒なら、大粒の内部関数としても使える

レベルでしょう。うーむ素晴らしい。お祝いにスイカを食べましょう。

開発：まあ、内部の関数と同様に仕様を記述して普通に呼び出せればもっと良いのですけどね。

＊　＊　＊

基盤：今朝方WordPressを5.5に上げたのですが、こんな事になっちゃいましたね。

開発：青の時代が終わった。

社長：いやそれで、前々からインライン画像の配置のデフォルトが中央寄せで無いのが嫌だったのですが、5.5から中央寄せになりましたね。ところが私はこういう端末画面の貼り付けは左詰めにしようと最近決めたのです。ところが、左詰めに変更すると次の段落が右の隙間に入り込むようになっちゃって、なんというかWordPress本体による破壊テロかと。

基盤：まあ、短い段落にインライン画像を時々はさむだけの編集機能でしたら、別にWordPressでなくても、なんでもいいっちゃいいんですよね。

— 2020-0813 SatoxITS