株式会社 ITS MORE
2020年4月設立

Digitally signed by 佐藤 豊
Date: 2020.08.15 23:42:20 +09'00'

2020年8月15日 投稿者: SATOXITS

続GShell/0.1.0 — Go1.15と内蔵find

基盤：8月11日にGo 1.15 がリリースされてたので更新しました。

基盤：以下がハイライト。

Some of the highlights include:

- Substantial improvements to the Go linker
- Improved allocation for small objects at high core counts
- X.509 CommonName deprecation
- GOPROXY supports skipping proxies that return errors
- New embedded tzdata package
- A number of Core Library improvements

開発：リンカーが高速化って動的リンカーのことですかね。まずはビルド。これまでの1.14.3では…

```
MacMini% go version
go version go1.14.3 darwin/amd64
MacMini% time go build gsh.go
go build gsh.go  0.12s user 0.12s system 150% cpu 0.163 total
MacMini% time go build gsh.go
go build gsh.go  0.13s user 0.13s system 149% cpu 0.168 total
MacMini% ls -l gsh
-rwxr-xr-x  1 ysato  staff  12882028 Aug 15 11:33 gsh
MacMini% size gsh
__TEXT   __DATA   __OBJC  others  dec     hex
7446528 475136  0       69771264        77692928        4a18000
```

開発：新作1.5では…

```
iMac% go version
go version go1.15 darwin/amd64
iMac% time go build gsh.go
go build gsh.go  0.13s user 0.13s system 176% cpu 0.149 total
iMac% time go build gsh.go
go build gsh.go  0.13s user 0.13s system 171% cpu 0.153 total
iMac% ls -l gsh
-rwxr-xr-x  1 ysato  staff  11631116 Aug 15 11:32 gsh
iMac% size gsh
__TEXT   __DATA   __OBJC  others  dec     hex
6475776 499712  0       69558272        76533760        48fd000
```

社長：コードが1MBくらい小さくなりましたね。

開発：この1.14.3はMacMini 3.6GHz i3x4で、1.15はiMac 3.7GHz i5x6で動かしているという違いがあります。gshのnopコマンドで比較すると、それなりの違いは見られます。

```
MacMini% go version
go version go1.14.3 darwin/amd64
MacMini% gsh
!1! nop
--I-- Aug 15 12:15:17(0.000008910s)
!2! nop
--I-- Aug 15 12:15:18(0.000008513s)
!3! nop
--I-- Aug 15 12:15:18(0.000006222s)
!4! nop
--I-- Aug 15 12:15:19(0.000006092s)
!5! hi -l
!0   Aug 15 12:15:17     141.1µs/t 0.000055s/u 0.000072s/s nop
!1   Aug 15 12:15:18    33.175µs/t 0.000019s/u 0.000014s/s nop
!2   Aug 15 12:15:18    37.435µs/t 0.000023s/u 0.000024s/s nop
!3   Aug 15 12:15:19    26.839µs/t 0.000015s/u 0.000011s/s nop
```

```
iMac% go version
go version go1.15 darwin/amd64
iMac% gsh
!1! nop
--I-- Aug 15 12:11:41(0.000009085s)
!2! nop
--I-- Aug 15 12:11:42(0.000006388s)
!3! nop
--I-- Aug 15 12:11:43(0.000006027s)
!4! nop
--I-- Aug 15 12:11:43(0.000006584s)
!5! hi -l
!0   Aug 15 12:11:41    105.29µs/t 0.000042s/u 0.000053s/s nop
!1   Aug 15 12:11:42    25.416µs/t 0.000015s/u 0.000009s/s nop
!2   Aug 15 12:11:43    23.923µs/t 0.000014s/u 0.000008s/s nop
!3   Aug 15 12:11:43    25.128µs/t 0.000015s/u 0.000009s/s nop
```

基盤：最速で比べると 26.8us : 23.9us だから、10%程度iMacのほうが速いですね。

開発：まだiMacは暇人ですしね。そのわりには値がぶれますが…　まあ1コアだけ使う分にはほぼ同等と考えて良いようです。で、hello.goはというと。

```
!13! go version
go version go1.14.3 darwin/amd64
--I-- Aug 15 11:53:44(0.013793680s)
!14! ls -l hello.go hello-go hello-c hello-so.so
-rwxr-xr-x  1 ysato  staff    12556 Aug 14 10:23 hello-c
-rwxr-xr-x  1 ysato  staff  2182536 Aug 14 10:23 hello-go
-rw-r--r--  1 ysato  staff  3381400 Aug 14 10:23 hello-so.so
-rwxr-x---  1 ysato  staff      121 Aug 13 17:47 hello.go
--I-- Aug 15 11:54:12(0.004689374s)
!15! hi -l
!0   Aug 15 11:53:13 796.663341ms/t 0.187190s/u 0.204425s/s hello.go
!1   Aug 15 11:53:15 245.031972ms/t 0.162907s/u 0.101579s/s hello.go
!2   Aug 15 11:53:18 238.623913ms/t 0.167532s/u 0.101416s/s hello.go
!3   Aug 15 11:53:22  15.81026ms/t 0.001797s/u 0.003787s/s hello-go
!4   Aug 15 11:53:23  3.553783ms/t 0.001420s/u 0.001661s/s hello-go
!5   Aug 15 11:53:24  3.558388ms/t 0.001402s/u 0.001673s/s hello-go
!6   Aug 15 11:53:30  4.045346ms/t 0.000915s/u 0.001555s/s hello-c
!7   Aug 15 11:53:31  2.555854ms/t 0.000838s/u 0.001204s/s hello-c
!8   Aug 15 11:53:32  2.581881ms/t 0.000847s/u 0.001213s/s hello-c
!9   Aug 15 11:53:38  4.765978ms/t 0.002285s/u 0.000762s/s plugin hello-so
!10  Aug 15 11:53:39     43.87µs/t 0.000027s/u 0.000017s/s plugin hello-so
!11  Aug 15 11:53:40     63.19µs/t 0.000040s/u 0.000036s/s plugin hello-so
!12  Aug 15 11:53:44 13.835503ms/t 0.004480s/u 0.005156s/s go version
```

```
!14! go version
go version go1.15 darwin/amd64
--I-- Aug 15 11:50:45(0.007991011s)
!15! ls -l hello.go hello-go hello-c hello-so.so
-rwxr-xr-x  1 ysato  staff    12556 Aug 15 11:46 hello-c
-rwxr-xr-x  1 ysato  staff  2159480 Aug 15 11:46 hello-go
-rw-r--r--  1 ysato  staff  3265152 Aug 15 11:46 hello-so.so
-rwxr-x---  1 ysato  staff      121 Aug 13 17:47 hello.go
--I-- Aug 15 11:51:47(0.003467510s)
!16! hi -l
!0   Aug 15 11:48:57 422.908825ms/t 0.164544s/u 0.125355s/s hello.go
!1   Aug 15 11:49:00 193.691975ms/t 0.145846s/u 0.106837s/s hello.go
!2   Aug 15 11:49:09 193.624972ms/t 0.147212s/u 0.104415s/s hello.go
!3   Aug 15 11:49:15  6.067502ms/t 0.002526s/u 0.002457s/s hello-go
!4   Aug 15 11:49:17  6.725759ms/t 0.002642s/u 0.002725s/s hello-go
!5   Aug 15 11:49:18  6.972684ms/t 0.002755s/u 0.002700s/s hello-go
!6   Aug 15 11:49:22  2.153638ms/t 0.000777s/u 0.001041s/s hello-c
!7   Aug 15 11:49:23  2.112038ms/t 0.000774s/u 0.000987s/s hello-c
!8   Aug 15 11:49:24  2.101191ms/t 0.000759s/u 0.000985s/s hello-c
!9   Aug 15 11:49:30  2.157406ms/t 0.001695s/u 0.000562s/s plugin hello-so
!10  Aug 15 11:49:31    37.228µs/t 0.000025s/u 0.000013s/s plugin hello-so
!11  Aug 15 11:49:32    37.872µs/t 0.000027s/u 0.000011s/s plugin hello-so
```

開発：go run は239msが194msに19%高速化します。この理由はホストの10%の性能差には収まらないような。比較のためにCのバイナリは2.6msから2.1msへ。あーこれは20%ですね。CPU以外の足回りも含めると、 MacMini より 20%くらいiMac27のほうが速いみたいな。

基盤：大金をはたいた甲斐があったようですね。

開発：ところが、ビルドされたバイナリの起動は3ms台から6ms台に、2倍くらい遅くなります。なんなんでしょう？プラグインの動的リンクは速くなったように見えますが、これは再現性が無い。

社長：なんにしても、GShellにとってはSubstantial というような改善には見えないですね…　まあ、利用法によるんでしょうけど。

開発：かといって特に新たな問題もないようですから、これからは Goは1.15で行こうと思います。

＊　＊　＊

開発：さて、当面取り組みたいのは2つ、ひとつは内蔵find + 内蔵grep の検索結果で得たファイル名を vi などの外部コマンドに渡す機能。これはたとえば検索結果を環境変数に入れるとか、which や history の拡張のように！とかでファイルを呼び出せるようにする。

基盤：その機能はめっちゃ欲しいですね。

開発：もうひとつはfindの高速化。どうもGoのディレクトリスキャンはCで実現した場合より2倍遅いんです。これは os パッケージなりでディレクトリエントリを返す時に、エントリにあるファイルをGoが勝手に開いてソートして返しているせいではないかと昨日は格闘して疲れました。勝手に開くくせに、返してくるデータの中にはGoが stat システムコールで得ているはずのファイルのブロックサイズとかデバイス情報とかが含まれていないんです。これでは find の -ls とか du 互換の機能を作るのに使い物になりません。内蔵tarも作れません。で syscall パッケージでトライしたのですが、それでもまだダメっぽい。なので、これはこの際、ディレクトリのスキャンを並列化しようかと思います。

社長：ある意味面白い方向へ後押しされてますね。

開発：それにしても彼らがそういう「抽象化」とか「OS独立性」を実現したい気持ちはわかるのですが、ディレクトリのスキャンとかファイルの属性とかは、とうの昔にOSを超えて共通のものになっています。なのに今この時代にそういうところで引っかかってのってなんだろうって思うんです。

社長：10年前には片付いてた問題のように思いますけどね。そういうOSの太古の脳の部分にあれからまた新しい非互換性が生じたとは思えないんですが。

基盤：ひょっとしてAndroidに無いものは削りたいとか…

社長：Goでshellを書くとかシステム管理コマンドを作るとかは想定してないのかもですね。

社長：なんだかしらす丼が食べたくなったので食事に行ってきます。

＊　＊　＊

社長：帰りました。そとは今日も夏でした。

社長：ところで私は、うちの玄関を出た時に見えるこの景色が結構好きです。

開発：電線の目障りがなければもっと良いのですが。

基盤：肉眼で感じる高さ感が消失しているのが面白いですね。

社長：それで、なんか夏らしさを写真に取ろうかと思ったのがこれ。

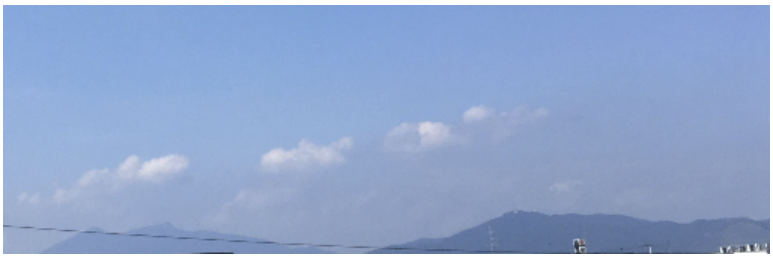
社長：アスファルトと夏草と短く暗い影。

基盤：残念ながらまったく伝わってこないですね。

社長：夏の撮影者。

基盤：これは少し感じるものがあります。

社長：電線と民家がなければなーと思いますね。



開発：北のベランダからポールを伸ばしてその先にカメラをつけるという計画です。

＊　＊　＊

開発：さてさて、find 結果のファイルのリストを後続のコマンドで利用する機能にですが。

社長：できましたか。

開発：機能そのものはすぐに出来たのですが、ユーザにそれをどう引用してもらうかが問題でした。結果にユーザが名前を付けるのも良いのですが、ちょっとかったるい。で考えたのは、N番目のコマンドの実行の結果得られたファイルのリストを !Nf というふうに参照する、というものです。こんな感じです。

```
iMac% gsh
!1! ver
gsh/0.1.0 (2020-0815a)
--I-- Aug 15 23:04:52(0.000027763s)
!2! -grep Hello
./hello.c:2:    printf("C-Hello GShell!!\n");
./hello.go:6:   fmt.Printf("Go-Hello GShell!(%v)\n",os.Args)
./Makefile:14:          "Hello World" \
./Hello World:0: Hello World!
./hello.sh:2: echo "SH-Hello Gshell!"
./arc/work/hello.c:2:   printf("C-Hello GShell!!\n");
./arc/work/hello.go:6:  fmt.Printf("Go-Hello GShell!(%v)\n",os.Args)
./arc/work/hello.sh:2: echo "SH-Hello Gshell!"
--I-- Aug 15 23:05:10(0.008664951s)
!3! wc !2f
       4       9      81 ./hello.c
      11      16     121 ./hello.go
      49     165    1100 ./Makefile
       1       2      13 ./Hello World
       3       4      35 ./hello.sh
       4       9      81 ./arc/work/hello.c
      11      16     121 ./arc/work/hello.go
       3       4      35 ./arc/work/hello.sh
      86     225    1587 total
--I-- Aug 15 23:05:27(0.001901389s)
!4! vi !2f
8 files to edit
--I-- Aug 15 23:05:37(4.083833178s)
!5! echo !2f
./hello.c ./hello.go ./Makefile ./Hello World ./hello.sh ./arc/work/hello.c
 ./arc/work/hello.go ./arc/work/hello.sh
--I-- Aug 15 23:05:45(0.000039255s)
!6! file !2f
./hello.c:           c program text, ASCII text
./hello.go:          c program text, ASCII text
./Makefile:          makefile script text, ASCII text
./Hello World:       ASCII text
./hello.sh:          POSIX shell script text executable, ASCII text
./arc/work/hello.c:  c program text, ASCII text
./arc/work/hello.go: c program text, ASCII text
./arc/work/hello.sh: POSIX shell script text executable, ASCII text
--I-- Aug 15 23:06:09(0.021188775s)
```

開発：行頭から -grep xxx で始まるのは、find . -exec grep xxx “{}” “;” の短縮形です。

社長：ファイルの名前による探索と中身による探索が一体化しているわけですね。

基盤：これは普通に便利。

社長：思うに従来のshellって、コマンドを起動するところまでは世話をするけど、コマンドの実行結果をどう他のコマンドに引き継ぐのかという所がほとんど無い用に思いますね。

開発：shellが直接関与している結果としてはコマンドの終了コードくらいですかね。あとは、コマンドの出力ストリームを他のコマンドにパイプで渡してやるくらいな。

社長：ファイルの中身の構造に関与しないところがUnix的な良さではあったと思うのですが、それがコマンドの実装に負担をかけていたことはあると思います。コマンドの実行の結果ファイルのリストが得られるなら、それを標準メタ情報出力的なファイルに出してもらって中継するとかが良いですかね。

開発：名前の付いたファイルの状態を保存する、ファイルの名前で他のコマンドにそれを伝える、というのは自然だとは思います。ただ、メタ情報については、名前と値の組とか、あるいはもっと構造的データ型式を定めてあげるのが良いかなとは思いますね。

基盤：今的にはJSONとかXMLでしょうか。

開発：それは嫌だなー…　まあ環境変数的に、名前と値の組。値の内部構造としては…まあJSONでもいいのかな。

社長：上意下達が環境変数なので、下からの結果の戻しも環境変数と同じ構造になってれば良いですね。

開発：まあ、プロセス間でメモリマップしてやりとりすれば、ファイルであることを意識しなくても済みますしね。ファイルだと、第三者や時間を超えて情報を共有できるのが魅力です。

社長：Go言語間であれば、データのバイナリ表現であっても良いですよね。テキスト型式である必要は無い。

開発：いわゆる .conf ｔか .rc とかの初期設定ファイルにしても、そのための特殊な文法を導入する必要はなくて、Goで書いてコンパイルして動的リンクして組み込めば良いと思うんですよね。ユーザはGoだけ知ってればよいし、設定記述の構文的な間違いはコンパイラが見つけてくれるし、起動も速い。

社長：プラグインはGoの核心ですね。

開発：並列findの件は明日取り組みたいと思います。

— 2020-0815 SatoxITS

GShell-0.1.0-by-SatoxITS ダウンロード