

```

1  /*<html>
2  <span id="gsh">
3  <span id="gsh-digest" data-target-id="gsh" data-crc32u="2307934617" data-length="194595" data-lines="6938" data-time="1599783893220"></span>
4  <meta charset="UTF-8">
5  <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  <link rel="icon" id="GshFaviconURL" href=""><!-- place holder -->
7  <span hidden="" id="GshVersion">gsh-0.3.7--2020-09-10--SatoxITS</span>
8  <title>GShell-0.3.7 by SatoxITS</title>
9  <header id="GshBanner" height="100px" onclick="shiftBG();" style="">
10 <div align="right"><note><a href="http://archive.gshell.org">GShell/<a> version 0.3.7 // 2020-09-10 // SatoxITS</note></div>
11 </header>
12 <h2>GShell // a General purpose Shell built on the top of Golang</h2>
13 <p>
14 <note>
15 It is a shell for myself, by myself, of myself. --SatoxITS(^-^)
16 </note>
17 </p>
18 <span id="gsh-WinId" onclick="win_jump('0.1');">0</span>
19 <span id="GshMenu">
20 | <span id="gsh-menu-exit" onclick="html_close();"></span>
21 | <span id="gsh-menu-fork" onclick="html_fork();">Fork</span>
22 | <span id="GshMenuStop" onclick="html_stop(this,true);">Stop</span>
23 | <span id="GshMenuFold" onclick="html_fold(this);">Unfold</span>
24 | <span id="gsh-menu-cksum" onclick="html_digest();">Digest</span>
25 | <span id="GshMenuSign" onclick="html_sign(this);">Source</span>
26 |<!-- | <span id="gsh-menu-pure" onclick="html_pure(this);">Pure</span> -->
27 |</span>
28 */
29 /*
30 <details id="GshStatement" class="gsh-document"><summary>Statement</summary>
31 <h3>Fun to create a shell</h3>
32 <p>For a programmer, it must be far easy and fun to create his own simple shell
33 rightly fitting to his favor and necessities, than learning existing shells with
34 complex full features that he never use.
35 I, as one of programmers, am writing this tiny shell for my own real needs,
36 totally from scratch, with fun.
37 </p><p>
38 For a programmer, it is fun to learn new computer languages. For long years before
39 writing this software, I had been specialized to C and early HTML2 :-).
40 Now writing this software, I'm learning Go language, HTML5, JavaScript and CSS
41 on demand as a novice of these, with fun.
42 </p><p>
43 This single file "gsh.go", that is executable by Go, contains all of the code written
44 in Go. Also it can be displayed as "gsh.go.html" by browsers. It is a standalone
45 HTML file that works as the viewer of the code of itself, and as the "home page" of
46 this software.
47 </p><p>
48 Because this HTML file is a Go program, you may run it as a real shell program
49 on your computer.
50 But you must be aware that this program is written under situation like above.
51 Needless to say, there is no warranty for this program in any means.
52 </p>
53 <address>Aug 2020, SatoxITS (sato@its-more.jp)</address>
54 </details>
55 */
56 /*
57 <details id="GshFeatures" class="gsh-document"><summary>Features</summary><p>
58 </p>
59 <h3>Vi compatible command line editor</h3>
60 <p>
61 The command line of GShell can be edited with commands compatible with
62 <a href="https://www.washington.edu/computing/unix/vi.html"><b>vi</b></a>.
63 As in vi, you can enter <i><b>command mode</b></i> by <b>ESC</b> key,
64 then move around in the history by <b><code>j k ? n N</code></b>,
65 or within the current line by <b><code>l h f w b o $ %</code></b> or so.
66 </p>
67 </details>
68 */
69 /*
70 <details id="gsh-gindex">
71 <summary>Index</summary><div class="gsh-src">
72 Documents
73 <span class="gsh-link" onclick="jumpto_JavaScriptView();">Command summary</span>
74 Go lang part<span class="gsh-src" onclick="document.getElementById('gsh-gocode').open=true; ">
75 Package structures
76 <a href="#import">import</a>
77 <a href="#struct">struct</a>
78 Main functions
79 <a href="#comexpansion">str-expansion</a> // macro processor
80 <a href="#finder">finder</a> // builtin find + du
81 <a href="#grep">grep</a> // builtin grep + wc + cksum + ...
82 <a href="#plugin">plugin</a> // plugin commands
83 <a href="#ex-commands">system</a> // external commands
84 <a href="#builtin">builtin</a> // builtin commands
85 <a href="#network">network</a> // socket handler
86 <a href="#remote-sh">remote-sh</a> // remote shell
87 <a href="#redirect">redirect</a> // StdIn/Out redirecton
88 <a href="#history">history</a> // command history
89 <a href="#rusage">rusage</a> // resouce usage
90 <a href="#encode">encode</a> // encode / decode
91 <a href="#IME">IME</a> // command line IME
92 <a href="#getline">getline</a> // line editor
93 <a href="#scanf">scanf</a> // string decomposer
94 <a href="#interpreter">interpreter</a> // command interpreter
95 <a href="#main">main</a>
96 </span>
97 JavaScript part
98 <a href="#script-src-view" class="gsh-link" onclick="jumpto_JavaScriptView();">Source</a>
99 <a href="#gsh-data-frame" class="gsh-link" onclick="jumpto_DataView();">Builtin data</a>
100 CSS part
101 <a href="#style-src-view" class="gsh-link" onclick="jumpto_StyleView();">Source</a>
102 References
103 <a href="#" class="gsh-link" onclick="jumpto_WholeView();">Internal</a>
104 <a href="#gsh-reference" class="gsh-link" onclick="jumpto_ReferenceView();">External</a>
105 Whole parts
106 <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Source</a>
107 <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Download</a>
108 <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Dump</a>
109
110 </div>
111 </details>
112 */
113 /*<details id="gsh-gocode">
114 <summary>Go Source</summary><div class="gsh-src" onclick="document.getElementById('gsh-gocode').open=false; ">
115 // gsh - Go lang based Shell
116 // (c) 2020 ITS more Co., Ltd.
117 // 2020-0807 created by SatoxITS (sato@its-more.jp)
118
119 package main // gsh main
120
121 // <a name="import">Imported packages</a> // <a href="https://golang.org/pkg/">Packages</a>
122 import (
123     "fmt" // <a href="https://golang.org/pkg/fmt/">fmt</a>
124     "strings" // <a href="https://golang.org/pkg/strings/">strings</a>

```

```

125 "strconv" // <a href="https://golang.org/pkg/strconv/">strconv</a>
126 "sort" // <a href="https://golang.org/pkg/sort/">sort</a>
127 "time" // <a href="https://golang.org/pkg/time/">time</a>
128 "bufio" // <a href="https://golang.org/pkg/bufio/">bufio</a>
129 "io/ioutil" // <a href="https://golang.org/pkg/io/ioutil/">ioutil</a>
130 "os" // <a href="https://golang.org/pkg/os/">os</a>
131 "syscall" // <a href="https://golang.org/pkg/syscall/">syscall</a>
132 "plugin" // <a href="https://golang.org/pkg/plugin/">plugin</a>
133 "net" // <a href="https://golang.org/pkg/net/">net</a>
134 "net/http" // <a href="https://golang.org/pkg/net/http/">http</a>
135 // "html" // <a href="https://golang.org/pkg/html/">html</a>
136 "path/filepath" // <a href="https://golang.org/pkg/path/filepath/">filepath</a>
137 "go/types" // <a href="https://golang.org/pkg/go/types/">types</a>
138 "go/token" // <a href="https://golang.org/pkg/go/token/">token</a>
139 "encoding/base64" // <a href="https://golang.org/pkg/encoding/base64/">base64</a>
140 "unicode/utf8" // <a href="https://golang.org/pkg/unicode/utf8/">utf8</a>
141 // "gshdata" // gshell's logo and source code
142 "hash/crc32" // <a href="https://golang.org/pkg/unicode/hash/crc32/">crc32</a>
143 )
144
145 // // 2020-0906 added,
146 // // <a href="https://golang.org/cmd/cgo/">CGO</a>
147 // #include <poll.h> // </poll.h> to be closed as HTML tag :-p
148 // typedef struct { struct pollfd fdv[8]; } pollFdv;
149 // int pollx(pollFdv *fdv, int nfdv, int timeout){
150 //     return poll(fdv->fdv,nfdv,timeout);
151 // }
152 import "C"
153
154 // // 2020-0906 added,
155 func CFPollInl(fp*os.File, timeoutUs int)(ready uintptr){
156     var fdv = C.pollFdv{}
157     var nfdv = 1
158     var timeout = timeoutUs/1000
159
160     fdv.fdv[0].fd = C.int(fp.Fd())
161     fdv.fdv[0].events = C.POLLIN
162     if( 0 < EventRecvFd ){
163         fdv.fdv[1].fd = C.int(EventRecvFd)
164         fdv.fdv[1].events = C.POLLIN
165         nfdv += 1
166     }
167     r := C.pollx(&fdv,C.int(nfdv),C.int(timeout))
168     if( r <= 0 ){
169         return 0
170     }
171     if (int(fdv.fdv[1].revents) & int(C.POLLIN)) != 0 {
172         //fprintf(stderr,"--De-- got Event\n");
173         return uintptr(EventFdOffset + fdv.fdv[1].fd)
174     }
175     if (int(fdv.fdv[0].revents) & int(C.POLLIN)) != 0 {
176         return uintptr(NormalFdOffset + fdv.fdv[0].fd)
177     }
178     return 0
179 }
180
181 const (
182     NAME = "gsh"
183     VERSION = "0.3.7"
184     DATE = "2020-09-10"
185     AUTHOR = "SatoxITS(^-^)"
186 )
187 var (
188     GSH_HOME = ".gsh" // under home directory
189     GSH_PORT = 9999
190     MaxStreamSize = int64(128*1024*1024*1024) // 128GiB is too large?
191     PROMPT = "> "
192     LINESIZE = (8*1024)
193     PATHSEP = ":" // should be ";" in Windows
194     DIRSEP = "/" // canbe \ in Windows
195 )
196
197 // -xX logging control
198 // --A-- all
199 // --I-- info.
200 // --D-- debug
201 // --T-- time and resource usage
202 // --W-- warning
203 // --E-- error
204 // --F-- fatal error
205 // --Xn- network
206
207 // <a name="struct">Structures</a>
208 type GCommandHistory struct {
209     StartAt time.Time // command line execution started at
210     EndAt time.Time // command line execution ended at
211     ResCode int // exit code of (external command)
212     CmdError error // error string
213     OutData *os.File // output of the command
214     FoundFile []string // output - result of unfind
215     Rusagev [2]syscall.Rusage // Resource consumption, CPU time or so
216     CmdId int // maybe with identified with arguments or impact
217     // redirection commands should not be the CmdId
218     WorkDir string // working directory at start
219     WorkDirX int // index in ChdirHistory
220     CmdLine string // command line
221 }
222 type GChdirHistory struct {
223     Dir string
224     MovedAt time.Time
225     CmdIndex int
226 }
227 type CmdMode struct {
228     Background bool
229 }
230 type Event struct {
231     when time.Time
232     event int
233     evarg int64
234     CmdIndex int
235 }
236 var CmdIndex int
237 var Events []Event
238 type PluginInfo struct {
239     Spec *plugin.Plugin
240     Addr plugin.Symbol
241     Name string // maybe relative
242     Path string // this is in Plugin but hidden
243 }
244 type GServer struct {
245     host string
246     port string
247 }
248
249 // <a href="https://tools.ietf.org/html/rfc3230">Digest</a>

```

```

250 const ( // SumType
251     SUM_ITEMS = 0x000001 // items count
252     SUM_SIZE  = 0x000002 // data length (simply added)
253     SUM_SIZEHASH = 0x000004 // data length (hashed sequence)
254     SUM_DATEHASH = 0x000008 // date of data (hashed sequence)
255     // also envelope attributes like time stamp can be a part of digest
256     // hashed value of sizes or mod-date of files will be useful to detect changes
257
258     SUM_WORDS = 0x000010 // word count is a kind of digest
259     SUM_LINES = 0x000020 // line count is a kind of digest
260     SUM_SUM64 = 0x000040 // simple add of bytes, useful for human too
261
262     SUM_SUM32_BITS = 0x000100 // the number of true bits
263     SUM_SUM32_2BYTE = 0x000200 // 16bits words
264     SUM_SUM32_4BYTE = 0x000400 // 32bits words
265     SUM_SUM32_8BYTE = 0x000800 // 64bits words
266
267     SUM_SUM16_BSD = 0x001000 // UNIXsum -sum -bsd
268     SUM_SUM16_SYSV = 0x002000 // UNIXsum -sum -sysv
269     SUM_UNIXFILE = 0x004000
270     SUM_CRCIEEE = 0x008000
271 )
272 type CheckSum struct {
273     Files      int64 // the number of files (or data)
274     Size       int64 // content size
275     Words      int64 // word count
276     Lines      int64 // line count
277     SumType    int
278     Sum64      uint64
279     Crc32Table crc32.Table
280     Crc32Val   uint32
281     Sum16      int
282     Ctime      time.Time
283     Atime      time.Time
284     Mtime      time.Time
285     Start      time.Time
286     Done       time.Time
287     RusgAtStart [2]syscall.Rusage
288     RusgAtEnd   [2]syscall.Rusage
289 }
290 type ValueStack [][]string
291 type GshContext struct {
292     StartDir string // the current directory at the start
293     GetLine  string // gsh-getline command as a input line editor
294     ChdirHistory []GChdirHistory // the 1st entry is wd at the start
295     gshPA      syscall.ProcAttr
296     CommandHistory []GCommandHistory
297     CmdCurrent   GCommandHistory
298     BackGround   bool
299     BackGroundJobs []int
300     LastRusage   syscall.Rusage
301     GshHomeDir   string
302     TerminalId   int
303     CmdTrace     bool // should be [map]
304     CmdTime      bool // should be [map]
305     PluginFuncs []PluginInfo
306     iValues      []string
307     iDelimiter   string // field separator of print out
308     iFormat      string // default print format (of integer)
309     iValStack    ValueStack
310     LastServer   GServer
311     RSERVER      string // [gsh://]host[:port]
312     RWD          string // remote (target, there) working directory
313     lastCheckSum CheckSum
314 }
315
316 func nsleep(ns time.Duration){
317     time.Sleep(ns)
318 }
319 func usleep(ns time.Duration){
320     nsleep(ns*1000)
321 }
322 func msleep(ns time.Duration){
323     nsleep(ns*1000000)
324 }
325 func sleep(ns time.Duration){
326     nsleep(ns*1000000000)
327 }
328
329 func strBegins(str, pat string)(bool){
330     if len(pat) <= len(str){
331         yes := str[0:len(pat)] == pat
332         //fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat,yes)
333         return yes
334     }
335     //fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat,false)
336     return false
337 }
338 func isin(what string, list []string) bool {
339     for _, v := range list {
340         if v == what {
341             return true
342         }
343     }
344     return false
345 }
346 func isinX(what string,list[]string)(int){
347     for i,v := range list {
348         if v == what {
349             return i
350         }
351     }
352     return -1
353 }
354
355 func env(opts []string) {
356     env := os.Environ()
357     if isin("-s", opts){
358         sort.Slice(env, func(i,j int) bool {
359             return env[i] < env[j]
360         })
361     }
362     for _, v := range env {
363         fmt.Printf("%v\n",v)
364     }
365 }
366
367 // - rewriting should be context dependent
368 // - should postpone until the real point of evaluation
369 // - should rewrite only known notation of symbol
370 func scanInt(str string)(val int,leng int){
371     leng = -1
372     for i,ch := range str {
373         if '0' <= ch && ch <= '9' {
374             leng = i+1

```

```

375     }else{
376         break
377     }
378 }
379 if 0 < leng {
380     ival,_ := strconv.Atoi(str[0:leng])
381     return ival,leng
382 }else{
383     return 0,0
384 }
385 }
386 func substHistory(gshCtx *GshContext,str string,i int,rstr string)(leng int,rst string){
387     if len(str[i+1:]) == 0 {
388         return 0,rstr
389     }
390     hi := 0
391     histlen := len(gshCtx.CommandHistory)
392     if str[i+1] == '!' {
393         hi = histlen - 1
394         leng = 1
395     }else{
396         hi,leng = scanInt(str[i+1:])
397         if leng == 0 {
398             return 0,rstr
399         }
400         if hi < 0 {
401             hi = histlen + hi
402         }
403     }
404     if 0 <= hi && hi < histlen {
405         var ext byte
406         if 1 < len(str[i+leng:]) {
407             ext = str[i+leng:][1]
408         }
409         //fmt.Printf("--D-- %v(%c)\n",str[i+leng:],str[i+leng])
410         if ext == 'f' {
411             leng += 1
412             xlist := []string{}
413             list := gshCtx.CommandHistory[hi].FoundFile
414             for _,v := range list {
415                 //list[i] = escapeWhiteSP(v)
416                 xlist = append(xlist,escapeWhiteSP(v))
417             }
418             //rstr += strings.Join(list," ")
419             rstr += strings.Join(xlist," ")
420         }else
421         if ext == '@' || ext == 'd' {
422             // IN@ .. workdir at the start of the command
423             leng += 1
424             rstr += gshCtx.CommandHistory[hi].WorkDir
425         }else{
426             rstr += gshCtx.CommandHistory[hi].CmdLine
427         }
428     }else{
429         leng = 0
430     }
431     return leng,rstr
432 }
433 func escapeWhiteSP(str string)(string){
434     if len(str) == 0 {
435         return "\\z" // empty, to be ignored
436     }
437     rstr := ""
438     for _,ch := range str {
439         switch ch {
440             case '\\': rstr += "\\\\"
441             case ' ': rstr += "\\s"
442             case '\t': rstr += "\\t"
443             case '\r': rstr += "\\r"
444             case '\n': rstr += "\\n"
445             default: rstr += string(ch)
446         }
447     }
448     return rstr
449 }
450 func unescapeWhiteSP(str string)(string){ // strip original escapes
451     rstr := ""
452     for i := 0; i < len(str); i++ {
453         ch := str[i]
454         if ch == '\\' {
455             if i+1 < len(str) {
456                 switch str[i+1] {
457                     case 'z':
458                         continue;
459                 }
460             }
461         }
462         rstr += string(ch)
463     }
464     return rstr
465 }
466 func unescapeWhiteSPV(strv []string)([]string){ // strip original escapes
467     ustrv := []string{}
468     for _,v := range strv {
469         ustrv = append(ustrv,unescapeWhiteSP(v))
470     }
471     return ustrv
472 }
473
474 // <a name="comexpansion">str-expansion</a>
475 // - this should be a macro processor
476 func strsubst(gshCtx *GshContext,str string,histonly bool) string {
477     rbuff := []byte{}
478     if false {
479         //@@@ Unicode should be cared as a character
480         return str
481     }
482     //rstr := ""
483     inEsc := 0 // escape characer mode
484     for i := 0; i < len(str); i++ {
485         //fmt.Printf("--D--Subst %v:%v\n",i,str[i:])
486         ch := str[i]
487         if inEsc == 0 {
488             if ch == '!' {
489                 //leng,xrstr := substHistory(gshCtx,str,i,rstr)
490                 leng,rs := substHistory(gshCtx,str,i,"")
491                 if 0 < leng {
492                     //_,rs := substHistory(gshCtx,str,i,"")
493                     rbuff = append(rbuff,[]byte(rs)...)
494                     i += leng
495                     //rstr = xrstr
496                     continue
497                 }
498             }
499             switch ch {

```

```

500         case '\\': inEsc = '\\'; continue
501         //case '%': inEsc = '%'; continue
502         case '$':
503     }
504 }
505 switch inEsc {
506 case '\\':
507     switch ch {
508         case '\\': ch = '\\'
509         case 's': ch = ' '
510         case 't': ch = '\t'
511         case 'r': ch = '\r'
512         case 'n': ch = '\n'
513         case 'z': inEsc = 0; continue // empty, to be ignored
514     }
515     inEsc = 0
516 case '$':
517     switch {
518         case ch == '%': ch = '%'
519         case ch == 'T':
520             //rstr = rstr + time.Now().Format(time.Stamp)
521             rs := time.Now().Format(time.Stamp)
522             rbuff = append(rbuff,[]byte(rs)...)
523             inEsc = 0
524             continue;
525         default:
526             // postpone the interpretation
527             //rstr = rstr + "%" + string(ch)
528             rbuff = append(rbuff,ch)
529             inEsc = 0
530             continue;
531     }
532     inEsc = 0
533 }
534 //rstr = rstr + string(ch)
535 rbuff = append(rbuff,ch)
536 }
537 //fmt.Printf("--D--subst(%s)(%s)\n",str,string(rbuff))
538 return string(rbuff)
539 //return rstr
540 }
541 func showFileInfo(path string, opts []string) {
542     if isin("-l",opts) || isin("-ls",opts) {
543         fi, err := os.Stat(path)
544         if err != nil {
545             fmt.Printf("----- ((%v))",err)
546         }else{
547             mod := fi.ModTime()
548             date := mod.Format(time.Stamp)
549             fmt.Printf("%v %v %s ",fi.Mode(),fi.Size(),date)
550         }
551     }
552     fmt.Printf("%s",path)
553     if isin("-sp",opts) {
554         fmt.Printf(" ")
555     }else
556     if ! isin("-n",opts) {
557         fmt.Printf("\n")
558     }
559 }
560 func userHomeDir()(string,bool){
561     /*
562     homedir,_ = os.UserHomeDir() // not implemented in older Golang
563     */
564     homedir,found := os.LookupEnv("HOME")
565     //fmt.Printf("--I-- HOME=%v(%v)\n",homedir,found)
566     if !found {
567         return "/tmp",found
568     }
569     return homedir,found
570 }
571 }
572 func toFullpath(path string) (fullpath string) {
573     if path[0] == '/' {
574         return path
575     }
576     pathv := strings.Split(path,DIRSEP)
577     switch {
578     case pathv[0] == ".":
579         pathv[0],_ = os.Getwd()
580     case pathv[0] == "..": // all ones should be interpreted
581         cwd,_ := os.Getwd()
582         ppathv := strings.Split(cwd,DIRSEP)
583         pathv[0] = strings.Join(ppathv,DIRSEP)
584     case pathv[0] == "-":
585         pathv[0],_ = userHomeDir()
586     default:
587         cwd,_ := os.Getwd()
588         pathv[0] = cwd + DIRSEP + pathv[0]
589     }
590     return strings.Join(pathv,DIRSEP)
591 }
592 }
593 func IsRegFile(path string)(bool){
594     fi, err := os.Stat(path)
595     if err == nil {
596         fm := fi.Mode()
597         return fm.IsRegular();
598     }
599     return false
600 }
601 }
602 // <a name="encode">Encode / Decode</a>
603 // <a href="https://golang.org/pkg/encoding/base64/#example_NewEncoder">Encoder</a>
604 func (gshCtx *GshContext)Enc(argv[]string){
605     file := os.Stdin
606     buff := make([]byte,LINESIZE)
607     li := 0
608     encoder := base64.NewEncoder(base64.StdEncoding,os.Stdout)
609     for li = 0; ; li++ {
610         count, err := file.Read(buff)
611         if count <= 0 {
612             break
613         }
614         if err != nil {
615             break
616         }
617         encoder.Write(buff[0:count])
618     }
619     encoder.Close()
620 }
621 func (gshCtx *GshContext)Dec(argv[]string){
622     decoder := base64.NewDecoder(base64.StdEncoding,os.Stdin)
623     li := 0
624     buff := make([]byte,LINESIZE)

```

```

625 for li = 0; ; li++ {
626     count, err := decoder.Read(buff)
627     if count <= 0 {
628         break
629     }
630     if err != nil {
631         break
632     }
633     os.Stdout.Write(buff[0:count])
634 }
635 }
636 // lnspl [N] [-crflf][C \\\
637 func (gshCtx *GshContext)SplitLine(argv[]string){
638     strRep := isin("-str",argv) // "..."+
639     reader := bufio.NewReaderSize(os.Stdin,64*1024)
640     ni := 0
641     toi := 0
642     for ni = 0; ; ni++ {
643         line, err := reader.ReadString('\n')
644         if len(line) <= 0 {
645             if err != nil {
646                 fmt.Fprintf(os.Stderr,"--I-- lnspl %d to %d (%v)\n",ni,toi,err)
647                 break
648             }
649         }
650         off := 0
651         ilen := len(line)
652         remlen := len(line)
653         if strRep { os.Stdout.Write([]byte("\n")) }
654         for oi := 0; 0 < remlen; oi++ {
655             olen := remlen
656             addnl := false
657             if 72 < olen {
658                 olen = 72
659                 addnl = true
660             }
661             fmt.Fprintf(os.Stderr,"--D-- write %d [%d.%d] %d %d/%d/%d\n",
662                 toi,ni,oi,off,olen,remlen,ilen)
663             toi += 1
664             os.Stdout.Write([]byte(line[0:olen]))
665             if addnl {
666                 if strRep {
667                     os.Stdout.Write([]byte("\n\n"))
668                 }else{
669                     //os.Stdout.Write([]byte("\r\n"))
670                     os.Stdout.Write([]byte("\n"))
671                     os.Stdout.Write([]byte("\n"))
672                 }
673             }
674             line = line[olen:]
675             off += olen
676             remlen -= olen
677         }
678         if strRep { os.Stdout.Write([]byte("\n")) }
679     }
680     fmt.Fprintf(os.Stderr,"--I-- lnspl %d to %d\n",ni,toi)
681 }
682 }
683 // CRC32 <a href="http://golang.jp/pkg/hash-crc32">crc32</a>
684 // 1 0000 0100 1100 0001 0001 1101 1011 0111
685 var CRC32UNIX uint32 = uint32(0x04C11DB7) // Unix cksum
686 var CRC32IEEE uint32 = uint32(0xEDB88320)
687 func byteCRC32add(crc uint32,str[]byte,len uint64)(uint32){
688     var oi uint64
689     for oi = 0; oi < len; oi++ {
690         var oct = str[oi]
691         for bi := 0; bi < 8; bi++ {
692             //fprintf(stderr,"--CRC32 %d %X (%d.%d)\n",crc,oct,oi,bi)
693             ovf1 := (crc & 0x80000000) != 0
694             ovf2 := (oct & 0x80) != 0
695             ovf := (ovf1 && !ovf2) || (!ovf1 && ovf2)
696             oct <<= 1
697             crc <<= 1
698             if ovf { crc ^= CRC32UNIX }
699         }
700     }
701     //fprintf(stderr,"--CRC32 return %d %d\n",crc,len)
702     return crc;
703 }
704 func byteCRC32end(crc uint32, len uint64)(uint32){
705     var slen = make([]byte,4)
706     var li = 0
707     for li = 0; li < 4; {
708         slen[li] = byte(len)
709         li += 1
710         len >>= 8
711         if( len == 0 ){
712             break
713         }
714     }
715     crc = byteCRC32add(crc,slen,uint64(li))
716     crc ^= 0xFFFFFFFF
717     return crc
718 }
719 func strCRC32(str string,len uint64)(crc uint32){
720     crc = byteCRC32add(0,[]byte(str),len)
721     crc = byteCRC32end(crc,len)
722     //fprintf(stderr,"--CRC32 %d %d\n",crc,len)
723     return crc
724 }
725 func CRC32Finish(crc uint32, table *crc32.Table, len uint64)(uint32){
726     var slen = make([]byte,4)
727     var li = 0
728     for li = 0; li < 4; {
729         slen[li] = byte(len & 0xFF)
730         li += 1
731         len >>= 8
732         if( len == 0 ){
733             break
734         }
735     }
736     crc = crc32.Update(crc,table,slen)
737     crc ^= 0xFFFFFFFF
738     return crc
739 }
740 }
741 func (gsh*GshContext)xChecksum(path string,argv[]string, sum*Checksum)(int64){
742     if isin("-type/f",argv) && !IsRegFile(path){
743         return 0
744     }
745     if isin("-type/d",argv) && IsRegFile(path){
746         return 0
747     }
748     file, err := os.OpenFile(path,os.O_RDONLY,0)
749     if err != nil {

```

```

750     fmt.Printf("--E-- cksum %v (%v)\n",path,err)
751     return -1
752 }
753 defer file.Close()
754 if gsh.CmdTrace { fmt.Printf("--I-- cksum %v %v\n",path,argv) }
755
756 bi := 0
757 var buff = make([]byte,32*1024)
758 var total int64 = 0
759 var initTime = time.Time{}
760 if sum.Start == initTime {
761     sum.Start = time.Now()
762 }
763 for bi = 0; ; bi++ {
764     count,err := file.Read(buff)
765     if count <= 0 || err != nil {
766         break
767     }
768     if (sum.SumType & SUM_SUM64) != 0 {
769         s := sum.Sum64
770         for _,c := range buff[0:count] {
771             s += uint64(c)
772         }
773         sum.Sum64 = s
774     }
775     if (sum.SumType & SUM_UNIXFILE) != 0 {
776         sum.Crc32Val = byteCRC32add(sum.Crc32Val,buff,uint64(count))
777     }
778     if (sum.SumType & SUM_CRCIEEE) != 0 {
779         sum.Crc32Val = crc32.Update(sum.Crc32Val,&sum.Crc32Table,buff[0:count])
780     }
781     // <a href="https://en.wikipedia.org/wiki/BSD_checksum">BSD checksum</a>
782     if (sum.SumType & SUM_SUM16_BSD) != 0 {
783         s := sum.Sum16
784         for _,c := range buff[0:count] {
785             s = (s >> 1) + ((s & 1) << 15)
786             s += int(c)
787             s &= 0xFFFF
788             //fmt.Printf("BSDsum: %d[%d] %d\n",sum.Size+int64(i),i,s)
789         }
790         sum.Sum16 = s
791     }
792     if (sum.SumType & SUM_SUM16_SYSV) != 0 {
793         for bj := 0; bj < count; bj++ {
794             sum.Sum16 += int(buff[bj])
795         }
796     }
797     total += int64(count)
798 }
799 sum.Done = time.Now()
800 sum.Files += 1
801 sum.Size += total
802 if !isin("-s",argv) {
803     fmt.Printf("%v ",total)
804 }
805 return 0
806 }
807
808 // <a name="grep">grep</a>
809 // "lines", "lin" or "lnp" for "(text) line processor" or "scanner"
810 // a*,lab,c,... sequential combination of patterns
811 // what "LINE" is should be definable
812 // generic line-by-line processing
813 // grep [-v]
814 // cat -n -v
815 // uniq [-c]
816 // tail -f
817 // sed s/x/y/ or awk
818 // grep with line count like wc
819 // rewrite contents if specified
820 func (gsh*GshContext)xGrep(path string,rexpv[]string)(int){
821     file, err := os.OpenFile(path,os.O_RDONLY,0)
822     if err != nil {
823         fmt.Printf("--E-- grep %v (%v)\n",path,err)
824         return -1
825     }
826     defer file.Close()
827     if gsh.CmdTrace { fmt.Printf("--I-- grep %v %v\n",path,rexpv) }
828     //reader := bufio.NewReaderSize(file,LINESIZE)
829     reader := bufio.NewReaderSize(file,80)
830     li := 0
831     found := 0
832     for li = 0; ; li++ {
833         line, err := reader.ReadString('\n')
834         if len(line) <= 0 {
835             break
836         }
837         if 150 < len(line) {
838             // maybe binary
839             break;
840         }
841         if err != nil {
842             break
843         }
844         if 0 <= strings.Index(string(line),rexpv[0]) {
845             found += 1
846             fmt.Printf("%s:%d: %s",path,li,line)
847         }
848     }
849     //fmt.Printf("total %d lines %s\n",li,path)
850     //if( 0 < found ){ fmt.Printf("((found %d lines %s))\n",found,path); }
851     return found
852 }
853
854 // <a name="finder">Finder</a>
855 // finding files with it name and contents
856 // file names are ORed
857 // show the content with %x fmt list
858 // ls -R
859 // tar command by adding output
860 type fileSum struct {
861     Err int64 // access error or so
862     Size int64 // content size
863     DupSize int64 // content size from hard links
864     Blocks int64 // number of blocks (of 512 bytes)
865     DupBlocks int64 // Blocks pointed from hard links
866     HLinks int64 // hard links
867     Words int64
868     Lines int64
869     Files int64
870     Dirs int64 // the num. of directories
871     SymLink int64
872     Flats int64 // the num. of flat files
873     MaxDepth int64
874     MaxNamlen int64 // max. name length

```

```

875     nextRepo    time.Time
876 }
877 func showFusage(dir string, fusage *fileSum) {
878     bsume := float64(((fusage.Blocks - fusage.DupBlocks) / 2) * 1024) / 1000000.0
879     // bsumdup := float64((fusage.Blocks / 2) * 1024) / 1000000.0
880
881     fmt.Printf("%v: %v files (%vd %vs %vh) %.6f MB (%.2f MBK)\n",
882         dir,
883         fusage.Files,
884         fusage.Dirs,
885         fusage.Symlink,
886         fusage.HLinks,
887         float64(fusage.Size) / 1000000.0, bsume);
888 }
889 const (
890     S_IFMT    = 0170000
891     S_IFCHR   = 0020000
892     S_IFDIR   = 0040000
893     S_IFREG   = 0100000
894     S_IFLNK   = 0120000
895     S_IFSOCK  = 0140000
896 )
897 func cumFinfo(fsum *fileSum, path string, stater error, fstat syscall.Stat_t, argv []string, verb bool) (*fileSum) {
898     now := time.Now()
899     if time.Second < now.Sub(fsum.nextRepo) {
900         if !fsum.nextRepo.IsZero() {
901             tstamp := now.Format(time.Stamp)
902             showFusage(tstamp, fsum)
903         }
904         fsum.nextRepo = now.Add(time.Second)
905     }
906     if stater != nil {
907         fsum.Err += 1
908         return fsum
909     }
910     fsum.Files += 1
911     if l < fstat.Nlink {
912         // must count only once...
913         // at least ignore ones in the same directory
914         //if finfo.Mode().IsRegular() {
915         if (fstat.Mode & S_IFMT) == S_IFREG {
916             fsum.HLinks += 1
917             fsum.DupBlocks += int64(fstat.Blocks)
918             //fmt.Printf("---Dup HardLink %v %s\n", fstat.Nlink, path)
919         }
920     }
921     //fsum.Size += finfo.Size()
922     fsum.Size += fstat.Size
923     fsum.Blocks += int64(fstat.Blocks)
924     //if verb { fmt.Printf("(%dBlk) %s", fstat.Blocks/2, path) }
925     if isin("-ls", argv) {
926         //if verb { fmt.Printf("%4d %8d ", fstat.Blksize, fstat.Blocks) }
927     //    fmt.Printf("%d\t", fstat.Blocks/2)
928     }
929     //if finfo.IsDir()
930     if (fstat.Mode & S_IFMT) == S_IFDIR {
931         fsum.Dirs += 1
932     }
933     //if (finfo.Mode() & os.ModeSymlink) != 0
934     if (fstat.Mode & S_IFMT) == S_IFLNK {
935         //if verb { fmt.Printf("symlink(%v,%s)\n", fstat.Mode, finfo.Name()) }
936         //if verb { fmt.Printf("symlink(%o,%s)\n", fstat.Mode, finfo.Name()) }
937         fsum.Symlink += 1
938     }
939     return fsum
940 }
941 func (gsh *GshContext) xxFindEntv(depth int, total *fileSum, dir string, dstat syscall.Stat_t, ei int, entv []string, npatv []string, argv []string) (*fileSum) {
942     nols := isin("-grep", argv)
943     // sort entv
944     /*
945     if isin("-t", argv) {
946         sort.Slice(filev, func(i, j int) bool {
947             return 0 < filev[i].ModTime().Sub(filev[j].ModTime())
948         })
949     }
950     */
951     /*
952     if isin("-u", argv) {
953         sort.Slice(filev, func(i, j int) bool {
954             return 0 < filev[i].AccTime().Sub(filev[j].AccTime())
955         })
956     }
957     if isin("-U", argv) {
958         sort.Slice(filev, func(i, j int) bool {
959             return 0 < filev[i].CreateTime().Sub(filev[j].CreateTime())
960         })
961     }
962     */
963     /*
964     if isin("-S", argv) {
965         sort.Slice(filev, func(i, j int) bool {
966             return filev[j].Size() < filev[i].Size()
967         })
968     }
969     */
970     for _, filename := range entv {
971         for _, npat := range npatv {
972             match := true
973             if npat == "*" {
974                 match = true
975             } else {
976                 match, _ = filepath.Match(npat, filename)
977             }
978             path := dir + DIRSEP + filename
979             if !match {
980                 continue
981             }
982             var fstat syscall.Stat_t
983             stater := syscall.Lstat(path, &fstat)
984             if stater != nil {
985                 if isin("-w", argv) { fmt.Printf("ufind: %v\n", stater) }
986                 continue;
987             }
988             if isin("-du", argv) && (fstat.Mode & S_IFMT) == S_IFDIR {
989                 // should not show size of directory in "-du" mode ...
990             } else {
991                 if nols && !isin("-s", argv) && (!isin("-du", argv) || isin("-a", argv)) {
992                     if isin("-du", argv) {
993                         fmt.Printf("%d\t", fstat.Blocks/2)
994                     }
995                     showFileInfo(path, argv)
996                 }
997                 if true { // && isin("-du", argv)
998                     total = cumFinfo(total, path, stater, fstat, argv, false)
999                 }
1000             }

```



```

1000     /*
1001     if isin("-wc",argv) {
1002     }
1003     */
1004     if gsh.lastCheckSum.SumType != 0 {
1005         gsh.xCksum(path,argv,gsh.lastCheckSum);
1006     }
1007     x := isinX("-grep",argv); // -grep will be convenient like -ls
1008     if 0 < x && x+1 <= len(argv) { // -grep will be convenient like -ls
1009         if isRegFile(path){
1010             found := gsh.xGrep(path,argv[x+1:])
1011             if 0 < found {
1012                 foundv := gsh.CmdCurrent.FoundFile
1013                 if len(foundv) < 10 {
1014                     gsh.CmdCurrent.FoundFile =
1015                         append(gsh.CmdCurrent.FoundFile,path)
1016                 }
1017             }
1018         }
1019     }
1020     if !isin("-r0",argv) { // -d 0 in du, -depth n in find
1021         //total.Depth += 1
1022         if (fststat.Mode & S_IFMT) == S_IFLNK {
1023             continue
1024         }
1025         if dstat.Rdev != fststat.Rdev {
1026             fmt.Printf("--I-- don't follow differnet device %v(%v) %v(%v)\n",
1027                 dir,dstat.Rdev,path,fststat.Rdev)
1028         }
1029         if (fststat.Mode & S_IFMT) == S_IFDIR {
1030             total = gsh.xxFind(depth+1,total,path,npatv,argv)
1031         }
1032     }
1033 }
1034 }
1035 return total
1036 }
1037 func (gsh*GshContext)xxFind(depth int,total *fileSum,dir string,npatv[]string,argv[]string)(*fileSum){
1038     nols := isin("-grep",argv)
1039     dirfile,oerr := os.OpenFile(dir,os.O_RDONLY,0)
1040     if oerr == nil {
1041         //fmt.Printf("--I-- %v(%v)[%d]\n",dir,dirfile,dirfile.Fd())
1042         defer dirfile.Close()
1043     }else{
1044     }
1045
1046     prev := *total
1047     var dstat syscall.Stat_t
1048     staterr := syscall.Lstat(dir,&dstat) // should be lstat
1049
1050     if staterr != nil {
1051         if !isin("-w",argv){ fmt.Printf("ufind: %v\n",staterr) }
1052         return total
1053     }
1054     //filev,err := ioutil.ReadDir(dir)
1055     //_,err := ioutil.ReadDir(dir) // ReadDir() heavy and bad for huge directory
1056     /*
1057     if err != nil {
1058         if !isin("-w",argv){ fmt.Printf("ufind: %v\n",err) }
1059         return total
1060     }
1061     */
1062     if depth == 0 {
1063         total = cumFinfo(total,dir,staterr,dstat,argv,true)
1064         if !nols && !isin("-s",argv) && (!isin("-du",argv) || isin("-a",argv)) {
1065             showFileInfo(dir,argv)
1066         }
1067     }
1068     // it is not a directory, just scan it and finish
1069
1070     for ei := 0; ; ei++ {
1071         entv,rderr := dirfile.Readdirnames(8*1024)
1072         if len(entv) == 0 || rderr != nil {
1073             //if rderr != nil { fmt.Printf("[%d] len=%d (%v)\n",ei,len(entv),rderr) }
1074             break
1075         }
1076         if 0 < ei {
1077             fmt.Printf("--I-- xxFind[%d] %d large-dir: %s\n",ei,len(entv),dir)
1078         }
1079         total = gsh.xxFindEntv(depth,total,dir,dstat,ei,entv,npatv,argv)
1080     }
1081     if isin("-du",argv) {
1082         // if in "du" mode
1083         fmt.Printf("%d\t%s\n",total.Blocks-prev.Blocks)/2,dir)
1084     }
1085     return total
1086 }
1087
1088 // {ufind|fu|ls} [Files] [// Names] [-- Expressions]
1089 // Files is "." by default
1090 // Names is "*" by default
1091 // Expressions is "-print" by default for "ufind", or -du for "fu" command
1092 func (gsh*GshContext)xFind(argv[]string){
1093     if 0 < len(argv) && strBegins(argv[0],"?"){
1094         showFound(gsh,argv)
1095         return
1096     }
1097     if isin("-cksum",argv) || isin("-sum",argv) {
1098         gsh.lastCheckSum = CheckSum{}
1099         if isin("-sum",argv) && isin("-add",argv) {
1100             gsh.lastCheckSum.SumType |= SUM_SUM64
1101         }else{
1102             if isin("-sum",argv) && isin("-size",argv) {
1103                 gsh.lastCheckSum.SumType |= SUM_SIZE
1104             }else{
1105                 if isin("-sum",argv) && isin("-bsd",argv) {
1106                     gsh.lastCheckSum.SumType |= SUM_SUM16_BSD
1107                 }else{
1108                     if isin("-sum",argv) && isin("-sysv",argv) {
1109                         gsh.lastCheckSum.SumType |= SUM_SUM16_SYSV
1110                     }else{
1111                         if isin("-sum",argv) {
1112                             gsh.lastCheckSum.SumType |= SUM_SUM64
1113                         }
1114                     }
1115                     if isin("-unix",argv) {
1116                         gsh.lastCheckSum.SumType |= SUM_UNIXFILE
1117                         gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32UNIX)
1118                     }
1119                     if isin("-ieee",argv){
1120                         gsh.lastCheckSum.SumType |= SUM_CRCIEEE
1121                         gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32IEEE)
1122                     }
1123                     gsh.lastCheckSum.RusgAtStart = Getrusagev()
1124                 }
1125             }
1126         }
1127     }
1128     var total = fileSum{}

```

```

1125 npats := []string{}
1126 for _,v := range argv {
1127     if 0 < len(v) && v[0] != '-' {
1128         npats = append(npats,v)
1129     }
1130     if v == "/" { break }
1131     if v == "--" { break }
1132     if v == "-grep" { break }
1133     if v == "-ls" { break }
1134 }
1135 if len(npats) == 0 {
1136     npats = []string{"*"}
1137 }
1138 cwd := "."
1139 // if to be fullpath ::: cwd, _ := os.Getwd()
1140 if len(npats) == 0 { npats = []string{"*"} }
1141 fusage := gsh.xxFind(0,&total,cwd,npats,argv)
1142 if gsh.lastCheckSum.SumType != 0 {
1143     var sumi uint64 = 0
1144     sum := &gsh.lastCheckSum
1145     if (sum.SumType & SUM_SIZE) != 0 {
1146         sumi = uint64(sum.Size)
1147     }
1148     if (sum.SumType & SUM_SUM64) != 0 {
1149         sumi = sum.Sum64
1150     }
1151     if (sum.SumType & SUM_SUM16_SYSV) != 0 {
1152         s := uint32(sum.Sum16)
1153         r := (s & 0xFFFF) + ((s & 0xFFFFFFFF) >> 16)
1154         s = (r & 0xFFFF) + (r >> 16)
1155         sum.Crc32Val = uint32(s)
1156         sumi = uint64(s)
1157     }
1158     if (sum.SumType & SUM_SUM16_BSD) != 0 {
1159         sum.Crc32Val = uint32(sum.Sum16)
1160         sumi = uint64(sum.Sum16)
1161     }
1162     if (sum.SumType & SUM_UNIXFILE) != 0 {
1163         sum.Crc32Val = byteCRC32end(sum.Crc32Val,uint64(sum.Size))
1164         sumi = uint64(byteCRC32end(sum.Crc32Val,uint64(sum.Size)))
1165     }
1166     if 1 < sum.Files {
1167         fmt.Printf("%v %v // %v / %v files, %v/file\r\n",
1168             sumi,sum.Size,
1169             abssize(sum.Size),sum.Files,
1170             abssize(sum.Size/sum.Files))
1171     }else{
1172         fmt.Printf("%v %v %v\n",
1173             sumi,sum.Size,npats[0])
1174     }
1175 }
1176 if !isin("-grep",argv) {
1177     showFusage("total",fusage)
1178 }
1179 if !isin("-s",argv){
1180     hits := len(gsh.CmdCurrent.FoundFile)
1181     if 0 < hits {
1182         fmt.Printf("--I-- %d files hits // can be refered with !&df\n",
1183             hits,len(gsh.CommandHistory))
1184     }
1185 }
1186 if gsh.lastCheckSum.SumType != 0 {
1187     if isin("-ru",argv) {
1188         sum := &gsh.lastCheckSum
1189         sum.Done = time.Now()
1190         gsh.lastCheckSum.RusgAtEnd = Getrusagev()
1191         elps := sum.Done.Sub(sum.Start)
1192         fmt.Printf("--cksum-size: %v (%v) / %v files, %v/file\r\n",
1193             sum.Size,abssize(sum.Size),sum.Files,abssize(sum.Size/sum.Files))
1194         nanos := int64(elps)
1195         fmt.Printf("--cksum-time: %v/total, %v/file, %.1f files/s, %v\r\n",
1196             abstime(nanos),
1197             abstime(nanos/sum.Files),
1198             (float64(sum.Files)*1000000000.0)/float64(nanos),
1199             abbspeed(sum.Size,nanos))
1200         diff := RusageSubv(sum.RusgAtEnd,sum.RusgAtStart)
1201         fmt.Printf("--cksum-rusg: %v\n",sRusagef("",argv,diff))
1202     }
1203 }
1204 return
1205 }
1206
1207 func showFiles(files[]string){
1208     sp := ""
1209     for i,file := range files {
1210         if 0 < i { sp = " " } else { sp = "" }
1211         fmt.Printf(sp+"%s",escapeWhiteSP(file))
1212     }
1213 }
1214 func showFound(gshCtx *GshContext, argv[]string){
1215     for i,v := range gshCtx.CommandHistory {
1216         if 0 < len(v.FoundFile) {
1217             fmt.Printf("%d (%d) ",i,len(v.FoundFile))
1218             if isin("-ls",argv){
1219                 fmt.Printf("\n")
1220                 for _,file := range v.FoundFile {
1221                     fmt.Printf(" ") //sub number?
1222                     showFileInfo(file,argv)
1223                 }
1224             }else{
1225                 showFiles(v.FoundFile)
1226                 fmt.Printf("\n")
1227             }
1228         }
1229     }
1230 }
1231
1232 func showMatchFile(filev []os.FileInfo, npat,dir string, argv[]string)(string,bool){
1233     fname := ""
1234     found := false
1235     for _,v := range filev {
1236         match, _ := filepath.Match(npat,(v.Name()))
1237         if match {
1238             fname = v.Name()
1239             found = true
1240             //fmt.Printf("%d] %s\n",i,v.Name())
1241             showIfExecutable(fname,dir,argv)
1242         }
1243     }
1244     return fname,found
1245 }
1246 func showIfExecutable(name,dir string,argv[]string)(ffullpath string,ffound bool){
1247     var fullpath string
1248     if strBegins(name,DIRSEP){
1249         fullpath = name

```

```

1250 }else{
1251     fullpath = dir + DIRSEP + name
1252 }
1253 fi, err := os.Stat(fullpath)
1254 if err != nil {
1255     fullpath = dir + DIRSEP + name + ".go"
1256     fi, err = os.Stat(fullpath)
1257 }
1258 if err == nil {
1259     fm := fi.Mode()
1260     if fm.IsRegular() {
1261         // R_OK=4, W_OK=2, X_OK=1, F_OK=0
1262         if syscall.Access(fullpath,5) == nil {
1263             ffullpath = fullpath
1264             ffound = true
1265             if ! isin("-s", argv) {
1266                 showFileInfo(fullpath,argv)
1267             }
1268         }
1269     }
1270 }
1271 return ffullpath, ffound
1272 }
1273 func which(list string, argv []string) (fullpathv []string, itis bool){
1274     if len(argv) <= 1 {
1275         fmt.Printf("Usage: which comand [-s] [-a] [-ls]\n")
1276         return []string{"", false}
1277     }
1278     path := argv[1]
1279     if strBegins(path, "/") {
1280         // should check if executable?
1281         exOK := showIfExecutable(path, "/", argv)
1282         fmt.Printf("--D-- %v exOK=%v\n", path, exOK)
1283         return []string{path}, exOK
1284     }
1285     pathenv, efound := os.LookupEnv(list)
1286     if ! efound {
1287         fmt.Printf("--E-- which: no \"%s\" environment\n", list)
1288         return []string{"", false}
1289     }
1290     showall := isin("-a", argv) || 0 <= strings.Index(path, "*")
1291     dirv := strings.Split(pathenv, PATHSEP)
1292     ffound := false
1293     ffullpath := path
1294     for _, dir := range dirv {
1295         if 0 <= strings.Index(path, "*") { // by wild-card
1296             list, _ := ioutil.ReadDir(dir)
1297             ffullpath, ffound = showMatchFile(list, path, dir, argv)
1298         }else{
1299             ffullpath, ffound = showIfExecutable(path, dir, argv)
1300         }
1301         //if ffound && !isin("-a", argv) {
1302         if ffound && !showall {
1303             break;
1304         }
1305     }
1306     return []string{ffullpath}, ffound
1307 }
1308
1309 func stripLeadingWSParg(argv []string) ([]string){
1310     for i, 0 < len(argv); {
1311         if len(argv[0]) == 0 {
1312             argv = argv[1:]
1313         }else{
1314             break
1315         }
1316     }
1317     return argv
1318 }
1319 func xEval(argv []string, nlend bool){
1320     argv = stripLeadingWSParg(argv)
1321     if len(argv) == 0 {
1322         fmt.Printf("eval [%%format] [Go-expression]\n")
1323         return
1324     }
1325     pfmt := "%v"
1326     if argv[0][0] == '$' {
1327         pfmt = argv[0]
1328         argv = argv[1:]
1329     }
1330     if len(argv) == 0 {
1331         return
1332     }
1333     gocode := strings.Join(argv, " ");
1334     //fmt.Printf("eval [%v] [%v]\n", pfmt, gocode)
1335     fset := token.NewFileSet()
1336     rval, _ := types.Eval(fset, nil, token.NoPos, gocode)
1337     fmt.Printf(pfmt, rval.Value)
1338     if nlend { fmt.Printf("\n") }
1339 }
1340
1341 func getval(name string) (found bool, val int) {
1342     /* should expand the name here */
1343     if name == "gsh.pid" {
1344         return true, os.Getpid()
1345     }else{
1346         if name == "gsh.ppid" {
1347             return true, os.Getppid()
1348         }
1349     }
1350     return false, 0
1351 }
1352
1353 func echo(argv []string, nlend bool){
1354     for ai := 1; ai < len(argv); ai++ {
1355         if 1 < ai {
1356             fmt.Printf(" ");
1357         }
1358         arg := argv[ai]
1359         found, val := getval(arg)
1360         if found {
1361             fmt.Printf("%d", val)
1362         }else{
1363             fmt.Printf("%s", arg)
1364         }
1365     }
1366     if nlend {
1367         fmt.Printf("\n");
1368     }
1369 }
1370
1371 func resfile() string {
1372     return "gsh.tmp"
1373 }
1374 //var resF *File
1375 func resmap() {

```

```

1375 //_, err := os.OpenFile(resfile(), os.O_RDWR|os.O_CREATE, os.ModeAppend)
1376 // https://deveppaper.com/solution-to-golang-bad-file-descriptor-problem/
1377 //_, err := os.OpenFile(resfile(), os.O_RDWR|os.O_CREATE, 0600)
1378 if err != nil {
1379     fmt.Printf("refF could not open: %s\n",err)
1380 }else{
1381     fmt.Printf("refF opened\n")
1382 }
1383 }
1384
1385 // @@2020-0821
1386 func gshScanArg(str string,strip int)(argv []string){
1387     var si = 0
1388     var sb = 0
1389     var inBracket = 0
1390     var argl = make([]byte,LINESIZE)
1391     var ax = 0
1392     debug := false
1393
1394     for ; si < len(str); si++ {
1395         if str[si] != ' ' {
1396             break
1397         }
1398     }
1399     sb = si
1400     for ; si < len(str); si++ {
1401         if sb <= si {
1402             if debug {
1403                 fmt.Printf("--Da- +%d %2d-%2d %s ... %s\n",
1404                     inBracket,sb,si,argl[0:ax],str[si:])
1405             }
1406             ch := str[si]
1407             if ch == '{' {
1408                 inBracket += 1
1409                 if 0 < strip && inBracket <= strip {
1410                     //fmt.Printf("stripLEV %d <= %d?\n",inBracket,strip)
1411                     continue
1412                 }
1413             }
1414             if 0 < inBracket {
1415                 if ch == '}' {
1416                     inBracket -= 1
1417                     if 0 < strip && inBracket < strip {
1418                         //fmt.Printf("stripLEV %d < %d?\n",inBracket,strip)
1419                         continue
1420                     }
1421                 }
1422                 argl[ax] = ch
1423                 ax += 1
1424                 continue
1425             }
1426             if str[si] == ' ' {
1427                 argv = append(argv,string(argl[0:ax]))
1428                 if debug {
1429                     fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
1430                         -1+len(argv),sb,si,string(str[sb:si]),string(str[si:]))
1431                 }
1432                 sb = si+1
1433                 ax = 0
1434                 continue
1435             }
1436             argl[ax] = ch
1437             ax += 1
1438         }
1439     }
1440     if sb < si {
1441         argv = append(argv,string(argl[0:ax]))
1442         if debug {
1443             fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
1444                 -1+len(argv),sb,si,string(argl[0:ax]),string(str[si:]))
1445         }
1446     }
1447     if debug {
1448         fmt.Printf("--Da- %d [%s] => [%d]%v\n",strip,strip,len(argv),argv)
1449     }
1450     return argv
1451 }
1452
1453 // should get stderr (into tmpfile ?) and return
1454 func (gsh*GshContext)Popen(name,mode string)(pin*os.File,pout*os.File,err bool){
1455     var pv = []int{-1,-1}
1456     syscall.Pipe(pv)
1457
1458     xarg := gshScanArg(name,1)
1459     name = strings.Join(xarg," ")
1460
1461     pin = os.NewFile(uintptr(pv[0]),"StdoutOf-"+name+"")
1462     pout = os.NewFile(uintptr(pv[1]),"StdinOf-"+name+"")
1463     fdix := 0
1464     dir := "?"
1465     if mode == "r" {
1466         dir = "<"
1467         fdix = 1 // read from the stdout of the process
1468     }else{
1469         dir = ">"
1470         fdix = 0 // write to the stdin of the process
1471     }
1472     gshPA := gsh.gshPA
1473     savfd := gshPA.Files[fdix]
1474
1475     var fd uintptr = 0
1476     if mode == "r" {
1477         fd = pout.Fd()
1478         gshPA.Files[fdix] = pout.Fd()
1479     }else{
1480         fd = pin.Fd()
1481         gshPA.Files[fdix] = pin.Fd()
1482     }
1483     // should do this by Goroutine?
1484     if false {
1485         fmt.Printf("--Ip- Opened fd[%v] %s %v\n",fd,dir,name)
1486         fmt.Printf("--RED1 [%d,%d,%d]->[%d,%d,%d]\n",
1487             os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd(),
1488             pin.Fd(),pout.Fd(),pout.Fd())
1489     }
1490     savi := os.Stdin
1491     savo := os.Stdout
1492     save := os.Stderr
1493     os.Stdin = pin
1494     os.Stdout = pout
1495     os.Stderr = pout
1496     gsh.BackGround = true
1497     gsh.gshelllh(name)
1498     gsh.BackGround = false
1499     os.Stdin = savi

```

```

1500         os.Stdout = savo
1501         os.Stderr = save
1502
1503         gshPA.Files[fdix] = savfd
1504         return pin,pout,false
1505     }
1506
1507     // <a name="ex-commands">External commands</a>
1508     func (gsh *GshContext)execcommand(exec bool, argv []string) (notf bool,exit bool) {
1509         if gsh.CmdTrace { fmt.Printf("--I-- excommand[%v](%v)\n",exec,argv) }
1510
1511         gshPA := gsh.gshPA
1512         fullpath, itis := which("PATH",[]string{"which",argv[0],"-s"})
1513         if itis == false {
1514             return true,false
1515         }
1516         fullpath := fullpathv[0]
1517         argv = unescapeWhiteSP(argv)
1518         if 0 < strings.Index(fullpath,".go") {
1519             nargv := argv // []string{}
1520             gofullpath, itis := which("PATH",[]string{"which","go","-s"})
1521             if itis == false {
1522                 fmt.Printf("--F-- Go not found\n")
1523                 return false,true
1524             }
1525             gofullpath := gofullpathv[0]
1526             nargv = []string{ gofullpath, "run", fullpath }
1527             fmt.Printf("--I-- %s {s %s %s}\n",gofullpath,
1528                 nargv[0],nargv[1],nargv[2])
1529             if exec {
1530                 syscall.Exec(gofullpath,nargv,os.Environ())
1531             }else{
1532                 pid, _ := syscall.ForkExec(gofullpath,nargv,&gshPA)
1533                 if gsh.BackGround {
1534                     fmt.Fprintf(stderr,"--Ip- in Background pid[%d]d(%v)\n",pid,len(argv),nargv)
1535                     gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
1536                 }else{
1537                     rusage := syscall.Rusage {}
1538                     syscall.Wait4(pid,nil,0,&rusage)
1539                     gsh.LastRusage = rusage
1540                     gsh.CmdCurrent.Rusagev[1] = rusage
1541                 }
1542             }
1543         }else{
1544             if exec {
1545                 syscall.Exec(fullpath,argv,os.Environ())
1546             }else{
1547                 pid, _ := syscall.ForkExec(fullpath,argv,&gshPA)
1548                 //fmt.Printf("[%d]\n",pid); // ' &' to be background
1549                 if gsh.BackGround {
1550                     fmt.Fprintf(stderr,"--Ip- in Background pid[%d]d(%v)\n",pid,len(argv),argv)
1551                     gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
1552                 }else{
1553                     rusage := syscall.Rusage {}
1554                     syscall.Wait4(pid,nil,0,&rusage);
1555                     gsh.LastRusage = rusage
1556                     gsh.CmdCurrent.Rusagev[1] = rusage
1557                 }
1558             }
1559         }
1560         return false,false
1561     }
1562
1563     // <a name="builtin">Builtin Commands</a>
1564     func (gshCtx *GshContext) sleep(argv []string) {
1565         if len(argv) < 2 {
1566             fmt.Printf("Sleep 100ms, 100us, 100ns, ...)\n")
1567             return
1568         }
1569         duration := argv[1];
1570         d, err := time.ParseDuration(duration)
1571         if err != nil {
1572             d, err = time.ParseDuration(duration+"s")
1573             if err != nil {
1574                 fmt.Printf("duration ? %s (%s)\n",duration,err)
1575                 return
1576             }
1577         }
1578         //fmt.Printf("Sleep %v\n",duration)
1579         time.Sleep(d)
1580         if 0 < len(argv[2:]) {
1581             gshCtx.gshellv(argv[2:])
1582         }
1583     }
1584     func (gshCtx *GshContext)repeat(argv []string) {
1585         if len(argv) < 2 {
1586             return
1587         }
1588         start0 := time.Now()
1589         for ri, _ := strconv.Atoi(argv[1]); 0 < ri; ri-- {
1590             if 0 < len(argv[2:]) {
1591                 //start := time.Now()
1592                 gshCtx.gshellv(argv[2:])
1593                 end := time.Now()
1594                 elps := end.Sub(start0);
1595                 if( 1000000000 < elps ){
1596                     fmt.Printf("repeat#%d %v)\n",ri,elps);
1597                 }
1598             }
1599         }
1600     }
1601
1602     func (gshCtx *GshContext)gen(argv []string) {
1603         gshPA := gshCtx.gshPA
1604         if len(argv) < 2 {
1605             fmt.Printf("Usage: %s N\n",argv[0])
1606             return
1607         }
1608         // should br repeated by "repeat" command
1609         count, _ := strconv.Atoi(argv[1])
1610         fd := gshPA.Files[1] // Stdout
1611         file := os.NewFile(fd,"internalStdOut")
1612         fmt.Printf("--I-- Gen. Count=%d to [%d]\n",count,file.Fd())
1613         //buf := []byte{}
1614         outdata := "0123 5678 0123 5678 0123 5678 0123 5678\r"
1615         for gi := 0; gi < count; gi++ {
1616             file.WriteString(outdata)
1617         }
1618         //file.WriteString("\n")
1619         fmt.Printf("\n(%d B)\n",count*len(outdata));
1620         //file.Close()
1621     }
1622
1623     // <a name="rexec">Remote Execution</a> // 2020-0820
1624     func Elapsed(from time.Time)(string){

```

```

1625     elps := time.Now().Sub(from)
1626     if 1000000000 < elps {
1627         return fmt.Sprintf("[%5d.%02ds]", elps/1000000000, (elps%1000000000)/10000000)
1628     }else
1629     if 1000000 < elps {
1630         return fmt.Sprintf("[%3d.%03dms]", elps/1000000, (elps%1000000)/1000)
1631     }else{
1632         return fmt.Sprintf("[%3d.%03dus]", elps/1000, (elps%1000))
1633     }
1634 }
1635 func abftime(nanos int64)(string){
1636     if 1000000000 < nanos {
1637         return fmt.Sprintf("%d.%02ds", nanos/1000000000, (nanos%1000000000)/10000000)
1638     }else
1639     if 1000000 < nanos {
1640         return fmt.Sprintf("%d.%03dms", nanos/1000000, (nanos%1000000)/1000)
1641     }else{
1642         return fmt.Sprintf("%d.%03dus", nanos/1000, (nanos%1000))
1643     }
1644 }
1645 func abszsize(size int64)(string){
1646     fsize := float64(size)
1647     if 1024*1024*1024 < size {
1648         return fmt.Sprintf("%.2fGiB", fsize/(1024*1024*1024))
1649     }else
1650     if 1024*1024 < size {
1651         return fmt.Sprintf("%.3fMiB", fsize/(1024*1024))
1652     }else{
1653         return fmt.Sprintf("%.3fKiB", fsize/1024)
1654     }
1655 }
1656 func abszsize(size int64)(string){
1657     fsize := float64(size)
1658     if 1024*1024*1024 < size {
1659         return fmt.Sprintf("%.2fGiB", fsize/(1024*1024*1024))
1660     }else
1661     if 1024*1024 < size {
1662         return fmt.Sprintf("%.3fMiB", fsize/(1024*1024))
1663     }else{
1664         return fmt.Sprintf("%.3fKiB", fsize/1024)
1665     }
1666 }
1667 func abbspd(totalB int64, ns int64)(string){
1668     MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
1669     if 1000 <= MBs {
1670         return fmt.Sprintf("%6.3fGB/s", MBs/1000)
1671     }
1672     if 1 <= MBs {
1673         return fmt.Sprintf("%6.3fMB/s", MBs)
1674     }else{
1675         return fmt.Sprintf("%6.3fKB/s", MBs*1000)
1676     }
1677 }
1678 func abspsd(totalB int64, ns time.Duration)(string){
1679     MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
1680     if 1000 <= MBs {
1681         return fmt.Sprintf("%6.3fGBps", MBs/1000)
1682     }
1683     if 1 <= MBs {
1684         return fmt.Sprintf("%6.3fMBps", MBs)
1685     }else{
1686         return fmt.Sprintf("%6.3fKBps", MBs*1000)
1687     }
1688 }
1689 func fileRelay(what string, in*os.File, out*os.File, size int64, bsiz int)(wcount int64){
1690     Start := time.Now()
1691     buff := make([]byte, bsiz)
1692     var total int64 = 0
1693     var rem int64 = size
1694     nio := 0
1695     Prev := time.Now()
1696     var PrevSize int64 = 0
1697
1698     fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) START\n",
1699         what, abszsize(total), size, nio)
1700
1701     for i:= 0; ; i++ {
1702         var len = bsiz
1703         if int(rem) < len {
1704             len = int(rem)
1705         }
1706         Now := time.Now()
1707         Elps := Now.Sub(Prev);
1708         if 1000000000 < Now.Sub(Prev) {
1709             fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) %s\n",
1710                 what, abszsize(total), size, nio,
1711                 abspsd((total-PrevSize), Elps))
1712             Prev = Now;
1713             PrevSize = total
1714         }
1715         rlen := len
1716         if in != nil {
1717             // should watch the disconnection of out
1718             rcc, err := in.Read(buff[0:rlen])
1719             if err != nil {
1720                 fmt.Printf(Elapsed(Start)+"--En- X: %s read(%v,%v)<%v\n",
1721                     what, rcc, err, in.Name())
1722                 break
1723             }
1724             rlen = rcc
1725             if string(buff[0:10]) == "(SoftEOF " {
1726                 var ecc int64 = 0
1727                 fmt.Sscanf(string(buff), "(SoftEOF %v", &ecc)
1728                 fmt.Printf(Elapsed(Start)+"--En- X: %s Recv ((SoftEOF %v))&v\n",
1729                     what, ecc, total)
1730                 if ecc == total {
1731                     break
1732                 }
1733             }
1734         }
1735
1736         wlen := rlen
1737         if out != nil {
1738             wcc, err := out.Write(buff[0:rlen])
1739             if err != nil {
1740                 fmt.Printf(Elapsed(Start)+"--En-- X: %s write(%v,%v)>%v\n",
1741                     what, wcc, err, out.Name())
1742                 break
1743             }
1744             wlen = wcc
1745         }
1746         if wlen < rlen {
1747             fmt.Printf(Elapsed(Start)+"--En- X: %s incomplete write (%v/%v)\n",
1748                 what, wlen, rlen)
1749             break;

```

```

1750     }
1751
1752     nio += 1
1753     total += int64(rlen)
1754     rem -= int64(rlen)
1755     if rem <= 0 {
1756         break
1757     }
1758 }
1759 Done := time.Now()
1760 Elps := float64(Done.Sub(Start))/1000000000 //Seconds
1761 TotalMB := float64(total)/1000000 //MB
1762 MBps := TotalMB / Elps
1763 fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) %v %3fMB/s\n",
1764     what,total,size,nio,absize(total),MBps)
1765 return total
1766 }
1767 func tcpPush(clnt *os.File){
1768     // shrink socket buffer and recover
1769     usleep(100);
1770 }
1771 func (gsh*GshContext)RexecServer(argv[]string){
1772     debug := true
1773     Start0 := time.Now()
1774     Start := Start0
1775     // if local == "" { local = "0.0.0.0:9999" }
1776     local := "0.0.0.0:9999"
1777
1778     if 0 < len(argv) {
1779         if argv[0] == "-s" {
1780             debug = false
1781             argv = argv[1:]
1782         }
1783     }
1784     if 0 < len(argv) {
1785         argv = argv[1:]
1786     }
1787     port, err := net.ResolveTCPAddr("tcp",local);
1788     if err != nil {
1789         fmt.Printf("--En- S: Address error: %s (%s)\n",local,err)
1790         return
1791     }
1792     fmt.Printf(Elapsed(Start)+"--In- S: Listening at %s...\n",local);
1793     sconn, err := net.ListenTCP("tcp", port)
1794     if err != nil {
1795         fmt.Printf(Elapsed(Start)+"--En- S: Listen error: %s (%s)\n",local,err)
1796         return
1797     }
1798
1799     reqbuf := make([]byte,LINESIZE)
1800     res := ""
1801     for {
1802         fmt.Printf(Elapsed(Start0)+"--In- S: Listening at %s...\n",local);
1803         aconn, err := sconn.AcceptTCP()
1804         Start = time.Now()
1805         if err != nil {
1806             fmt.Printf(Elapsed(Start)+"--En- S: Accept error: %s (%s)\n",local,err)
1807             return
1808         }
1809         clnt, _ := aconn.File()
1810         fd := Clnt.Fd()
1811         ar := aconn.RemoteAddr()
1812         if debug { fmt.Printf(Elapsed(Start0)+"--In- S: Accepted TCP at %s [%d] <- %v\n",
1813             local,fd,ar) }
1814         res = fmt.Sprintf("220 GShell/%s Server\r\n",VERSION)
1815         fmt.Fprintf(clnt,"%s",res)
1816         if debug { fmt.Printf(Elapsed(Start)+"--In- S: %s",res) }
1817         count, err := clnt.Read(reqbuf)
1818         if err != nil {
1819             fmt.Printf(Elapsed(Start)+"--En- C: (%v %v) %v",
1820                 count,err,string(reqbuf))
1821         }
1822         req := string(reqbuf[:count])
1823         if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",string(req)) }
1824         reqv := strings.Split(string(req),"r")
1825         cmdv := gshScanArg(reqv[0],0)
1826         //cmdv := strings.Split(reqv[0]," ")
1827         switch cmdv[0] {
1828             case "HELO":
1829                 res = fmt.Sprintf("250 %v",req)
1830             case "GET":
1831                 // download {remotefile|-zN} [localfile]
1832                 var dsize int64 = 32*1024*1024
1833                 var bsize int = 64*1024
1834                 var fname string = ""
1835                 var in *os.File = nil
1836                 var pseudoEOF = false
1837                 if 1 < len(cmdv) {
1838                     fname = cmdv[1]
1839                     if strBegins(fname,"-z") {
1840                         fmt.Sscanf(fname[2:],"%d",&dsize)
1841                     }else
1842                     if strBegins(fname,"{") {
1843                         xin,xout,err := gsh.Popen(fname,"r")
1844                         if err {
1845                             }else{
1846                                 xout.Close()
1847                                 defer xin.Close()
1848                                 in = xin
1849                                 dsize = MaxStreamSize
1850                                 pseudoEOF = true
1851                             }
1852                     }else{
1853                         xin,err := os.Open(fname)
1854                         if err != nil {
1855                             fmt.Printf("--En- GET (%v)\n",err)
1856                         }else{
1857                             defer xin.Close()
1858                             in = xin
1859                             fi,_ := xin.Stat()
1860                             dsize = fi.Size()
1861                         }
1862                     }
1863                 }
1864                 //fmt.Printf(Elapsed(Start)+"--In- GET %v:%v\n",dsize,bsize)
1865                 res = fmt.Sprintf("200 %v\r\n",dsize)
1866                 fmt.Fprintf(clnt,"%v",res)
1867                 tcpPush(clnt); // should be separated as line in receiver
1868                 fmt.Printf(Elapsed(Start)+"--In- S: %v",res)
1869                 wcount := fileRelay("SendGET",in,clnt,dsize,bsize)
1870                 if pseudoEOF {
1871                     in.Close() // pipe from the command
1872                     // show end of stream data (its size) by OOB?
1873                     SoftEOF := fmt.Sprintf("({SoftEOF %v})",wcount)
1874                     fmt.Printf(Elapsed(Start)+"--In- S: Send %v\n",SoftEOF)

```

```

1875         tcpPush(clnt); // to let SoftEOF data appear at the top of received data
1876         fmt.Fprintf(clnt, "%v\r\n", SoftEOF)
1877         tcpPush(clnt); // to let SoftEOF alone in a packet (separate with 200 OK)
1878         // with client generated random?
1879         //fmt.Printf("--In- L: close %v (%v)\n", in.Fd(), in.Name())
1880     }
1881     res = fmt.Sprintf("200 GET done\r\n")
1882     case "PUT":
1883         // upload {srcfile|-zN} [dstfile]
1884         var dsz int64 = 32*1024*1024
1885         var bsize int = 64*1024
1886         var fname string = ""
1887         var out *os.File = nil
1888         if 1 < len(cmdv) { // localfile
1889             fmt.Sscanf(cmdv[1], "%d", &dsz)
1890         }
1891         if 2 < len(cmdv) {
1892             fname = cmdv[2]
1893             if fname == "" {
1894                 // nul dev
1895             }else{
1896                 if strBegins(fname, "(") {
1897                     xin, xout, err := gsh.Popen(fname, "w")
1898                     if err {
1899                         }else{
1900                             xin.Close()
1901                             defer xout.Close()
1902                             out = xout
1903                         }
1904                     }else{
1905                         // should write to temporary file
1906                         // should suppress ^C on tty
1907                         xout, err := os.OpenFile(fname, os.O_CREATE|os.O_RDWR|os.O_TRUNC, 0600)
1908                         //fmt.Printf("--In- S: open(%v) out(%v) err(%v)\n", fname, xout, err)
1909                         if err != nil {
1910                             fmt.Printf("--En- PUT (%v)\n", err)
1911                         }else{
1912                             out = xout
1913                         }
1914                     }
1915                 }
1916                 fmt.Printf(Elapsed(Start)+"--In- L: open(%v,w) %v (%v)\n",
1917                     fname, local, err)
1918             }
1919             fmt.Printf(Elapsed(Start)+"--In- PUT %v (%v)\n", dsz, bsize)
1920             fmt.Printf(Elapsed(Start)+"--In- S: 200 %v OK\r\n", dsz)
1921             fmt.Fprintf(clnt, "200 %v OK\r\n", dsz)
1922             fileRelay("RecvPUT", clnt, out, dsz, bsize)
1923             res = fmt.Sprintf("200 PUT done\r\n")
1924         default:
1925             res = fmt.Sprintf("400 What? %v", req)
1926     }
1927     swcc, serr := clnt.Write([]byte(res))
1928     if serr != nil {
1929         fmt.Printf(Elapsed(Start)+"--In- S: (wc=%v er=%v) %v", swcc, serr, res)
1930     }else{
1931         fmt.Printf(Elapsed(Start)+"--In- S: %v", res)
1932     }
1933     aconn.Close();
1934     clnt.Close();
1935 }
1936 sconn.Close();
1937 }
1938 func (gsh*GshContext)RexecClient(argv []string)(int, string){
1939     debug := true
1940     Start := time.Now()
1941     if len(argv) == 1 {
1942         return -1, "EmptyARG"
1943     }
1944     argv = argv[1:]
1945     if argv[0] == "-serv" {
1946         gsh.RexecServer(argv[1:])
1947         return 0, "Server"
1948     }
1949     remote := "0.0.0.0:9999"
1950     if argv[0][0] == '@' {
1951         remote = argv[0][1:]
1952         argv = argv[1:]
1953     }
1954     if argv[0] == "-s" {
1955         debug = false
1956         argv = argv[1:]
1957     }
1958     dport, err := net.ResolveTCPAddr("tcp", remote);
1959     if err != nil {
1960         fmt.Printf(Elapsed(Start)+"Address error: %s (%s)\n", remote, err)
1961         return -1, "AddressError"
1962     }
1963     fmt.Printf(Elapsed(Start)+"--In- C: Connecting to %s\n", remote)
1964     serv, err := net.DialTCP("tcp", nil, dport)
1965     if err != nil {
1966         fmt.Printf(Elapsed(Start)+"Connection error: %s (%s)\n", remote, err)
1967         return -1, "CannotConnect"
1968     }
1969     if debug {
1970         al := serv.LocalAddr()
1971         fmt.Printf(Elapsed(Start)+"--In- C: Connected to %v <- %v\n", remote, al)
1972     }
1973 }
1974 req := ""
1975 res := make([]byte, LINESIZE)
1976 count, err := serv.Read(res)
1977 if err != nil {
1978     fmt.Printf("--En- S: (%3d,%v) %v", count, err, string(res))
1979 }
1980 if debug { fmt.Printf(Elapsed(Start)+"--In- S: %v", string(res)) }
1981 }
1982 if argv[0] == "GET" {
1983     savPA := gsh.gshPA
1984     var bsize int = 64*1024
1985     req = fmt.Sprintf("%v\r\n", strings.Join(argv, " "))
1986     fmt.Printf(Elapsed(Start)+"--In- C: %v", req)
1987     fmt.Fprintf(serv, req)
1988     count, err = serv.Read(res)
1989     if err != nil {
1990     }else{
1991         var dsz int64 = 0
1992         var out *os.File = nil
1993         var out_tobeclosed *os.File = nil
1994         var fname string = ""
1995         var rcode int = 0
1996         var pid int = -1
1997         fmt.Sscanf(string(res), "%d %d", &rcode, &dsz)
1998         fmt.Printf(Elapsed(Start)+"--In- S: %v", string(res[0:count]))
1999         if 3 <= len(argv) {

```



```

2000         fname = argv[2]
2001         if strBegins(fname, "{") {
2002             xin,xout,err := gsh.Popen(fname,"w")
2003             if err {
2004                 }else{
2005                     xin.Close()
2006                     defer xout.Close()
2007                     out = xout
2008                     out_tobeclosed = xout
2009                     pid = 0 // should be its pid
2010                 }
2011             }else{
2012                 // should write to temporary file
2013                 // should suppress ^C on tty
2014                 xout,err := os.OpenFile(fname,os.O_CREATE|os.O_RDWR|os.O_TRUNC,0600)
2015                 if err != nil {
2016                     fmt.Printf("--En- %v\n",err)
2017                 }
2018                 out = xout
2019                 //fmt.Printf("--In-- %d > %s\n",out.Fd(),fname)
2020             }
2021         }
2022         in, _ := serv.File()
2023         fileRelay("RecvGET",in,out,dsize,bsize)
2024         if 0 <= pid {
2025             gsh.gshPA = savPA // recovery of Fd(), and more?
2026             fmt.Printf(Elapsed(Start)+"--In- L: close Pipe > %v\n",fname)
2027             out_tobeclosed.Close()
2028             //syscall.Wait4(pid,nil,0,nil) //@@
2029         }
2030     }
2031 }else
2032 if argv[0] == "PUT" {
2033     remote, _ := serv.File()
2034     var local *os.File = nil
2035     var dsize int64 = 32*1024*1024
2036     var bsize int = 64*1024
2037     var ofile string = "-"
2038     //fmt.Printf("--I-- Rex %v\n",argv)
2039     if 1 < len(argv) {
2040         fname := argv[1]
2041         if strBegins(fname, "-z") {
2042             fmt.Sscanf(fname[2:], "%d", &dsize)
2043         }else
2044         if strBegins(fname, "{") {
2045             xin,xout,err := gsh.Popen(fname,"r")
2046             if err {
2047                 }else{
2048                     xout.Close()
2049                     defer xin.Close()
2050                     //in = xin
2051                     local = xin
2052                     fmt.Printf("--In- [%d] < Upload output of %v\n",
2053                         local.Fd(),fname)
2054                     ofile = "-from."+fname
2055                     dsize = MaxStreamSize
2056                 }
2057             }else{
2058                 xlocal,err := os.Open(fname)
2059                 if err != nil {
2060                     fmt.Printf("--En- (%s)\n",err)
2061                     local = nil
2062                 }else{
2063                     local = xlocal
2064                     fi, _ := local.Stat()
2065                     dsize = fi.Size()
2066                     defer local.Close()
2067                     //fmt.Printf("--I-- Rex in(%v / %v)\n",ofile,dsize)
2068                 }
2069                 ofile = fname
2070                 fmt.Printf(Elapsed(Start)+"--In- L: open(%v,r)=%v %v (%v)\n",
2071                     fname,dsize,local,err)
2072             }
2073         }
2074         if 2 < len(argv) && argv[2] != "" {
2075             ofile = argv[2]
2076             //fmt.Printf("(%d)%v B.ofile=%v\n",len(argv),argv,ofile)
2077         }
2078         //fmt.Printf(Elapsed(Start)+"--I-- Rex out(%v)\n",ofile)
2079         fmt.Printf(Elapsed(Start)+"--In- PUT %v (%v)\n",dsize,bsize)
2080         req = fmt.Sprintf("PUT %v %v %v\n",dsize,ofile)
2081         if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
2082         fmt.Fprintf(serv,"%v",req)
2083         count,err = serv.Read(res)
2084         if debug { fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res[0:count])) }
2085         fileRelay("SendPUT",local,remote,dsize,bsize)
2086     }else{
2087         req = fmt.Sprintf("%v\n",strings.Join(argv, " "))
2088         if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
2089         fmt.Fprintf(serv,"%v",req)
2090         //fmt.Printf("--In- sending RexRequest(%v)\n",len(req))
2091     }
2092     //fmt.Printf(Elapsed(Start)+"--In- waiting RexResponse...\n")
2093     count,err = serv.Read(res)
2094     ress := ""
2095     if count == 0 {
2096         ress = "(nil)\n"
2097     }else{
2098         ress = string(res[:count])
2099     }
2100     if err != nil {
2101         fmt.Printf(Elapsed(Start)+"--En- S: (%d,%v) %v",count,err,ress)
2102     }else{
2103         fmt.Printf(Elapsed(Start)+"--In- S: %v",ress)
2104     }
2105     serv.Close()
2106     //conn.Close()
2107 }
2108 var stat string
2109 var rcode int
2110 fmt.Sscanf(ress, "%d %s", &rcode, &stat)
2111 //fmt.Printf("--D-- Client: %v (%v)",rcode,stat)
2112 return rcode,ress
2113 }
2114
2115 // <a name="remote-sh">Remote Shell</a>
2116 // gcp file [...] { [host]:[port]:[dir] | dir } // -p | -no-p
2117 func (gsh*GshContext)FileCopy(argv []string){
2118     var host = ""
2119     var port = ""
2120     var upload = false
2121     var download = false
2122     var xargv = []string{"rex-gcp"}
2123     var srcv = []string{}
2124     var dstv = []string{}

```

```

2125     argv = argv[1:]
2126     for _,v := range argv {
2127         /*
2128         if v[0] == '-' { // might be a pseudo file (generated date)
2129             continue
2130         }
2131         */
2132         obj := strings.Split(v,":")
2133         //fmt.Printf("%d %v %v\n",len(obj),v,obj)
2134         if 1 < len(obj) {
2135             host = obj[0]
2136             file := ""
2137             if 0 < len(host) {
2138                 gsh.LastServer.host = host
2139             }else{
2140                 host = gsh.LastServer.host
2141                 port = gsh.LastServer.port
2142             }
2143             if 2 < len(obj) {
2144                 port = obj[1]
2145                 if 0 < len(port) {
2146                     gsh.LastServer.port = port
2147                 }else{
2148                     port = gsh.LastServer.port
2149                 }
2150                 file = obj[2]
2151             }else{
2152                 file = obj[1]
2153             }
2154             if len(srcv) == 0 {
2155                 download = true
2156                 srcv = append(srcv,file)
2157                 continue
2158             }
2159             upload = true
2160             dstv = append(dstv,file)
2161             continue
2162         }
2163         /*
2164         idx := strings.Index(v,":")
2165         if 0 <= idx {
2166             remote = v[0:idx]
2167             if len(srcv) == 0 {
2168                 download = true
2169                 srcv = append(srcv,v[idx+1:])
2170                 continue
2171             }
2172             upload = true
2173             dstv = append(dstv,v[idx+1:])
2174             continue
2175         }
2176         */
2177         if download {
2178             dstv = append(dstv,v)
2179         }else{
2180             srcv = append(srcv,v)
2181         }
2182     }
2183     hostport := "@" + host + ":" + port
2184     if upload {
2185         if host != "" { xargv = append(xargv,hostport) }
2186         xargv = append(xargv,"PUT")
2187         xargv = append(xargv,srcv[0]...)
2188         xargv = append(xargv,dstv[0]...)
2189         //fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v // %v\n",hostport,dstv,srcv)
2190         gsh.RexecClient(xargv)
2191     }else{
2192         if host != "" { xargv = append(xargv,hostport) }
2193         xargv = append(xargv,"GET")
2194         xargv = append(xargv,srcv[0]...)
2195         xargv = append(xargv,dstv[0]...)
2196         //fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v // %v\n",hostport,srcv,dstv,xargv)
2197         gsh.RexecClient(xargv)
2198     }
2199     }
2200 }
2201 // target
2202 func (gsh*GshContext)Trelpath(rloc string)(string){
2203     cwd, _ := os.Getwd()
2204     os.Chdir(gsh.RWD)
2205     os.Chdir(rloc)
2206     twd, _ := os.Getwd()
2207     os.Chdir(cwd)
2208     tpath := twd + "/" + rloc
2209     return tpath
2210 }
2211 // join to rremote GShell - [user@]host[:port] or cd host[:port]:path
2212 func (gsh*GshContext)Rjoin(argv[]string){
2213     if len(argv) <= 1 {
2214         fmt.Printf("--I-- current server = %v\n",gsh.RSERV)
2215         return
2216     }
2217     serv := argv[1]
2218     servv := strings.Split(serv,":")
2219     if 1 <= len(servv) {
2220         if servv[0] == "lo" {
2221             servv[0] = "localhost"
2222         }
2223     }
2224     switch len(servv) {
2225     case 1:
2226         //if strings.Index(serv,":") < 0 {
2227             serv = servv[0] + ":" + fmt.Sprintf("%d",GSH_PORT)
2228         //}
2229     case 2: // host:port
2230         serv = strings.Join(servv,":")
2231     }
2232     xargv := []string{"rex-join","@"+serv,"HELO"}
2233     rcode,stat := gsh.RexecClient(xargv)
2234     if (rcode / 100) == 2 {
2235         fmt.Printf("--I-- OK Joined (%v) %v\n",rcode,stat)
2236         gsh.RSERV = serv
2237     }else{
2238         fmt.Printf("--I-- NG, could not joined (%v) %v\n",rcode,stat)
2239     }
2240 }
2241 func (gsh*GshContext)Rexec(argv[]string){
2242     if len(argv) <= 1 {
2243         fmt.Printf("--I-- rexec command [ | {file || {command} ]\n",gsh.RSERV)

```

```

2250     return
2251 }
2252
2253 /*
2254 nargv := gshScanArg(strings.Join(argv, " "), 0)
2255 fmt.Printf("--D-- nargc=%d [%v]\n", len(nargv), nargv)
2256 if nargv[1][0] != '{' {
2257     nargv[1] = "{" + nargv[1] + "}"
2258     fmt.Printf("--D-- nargc=%d [%v]\n", len(nargv), nargv)
2259 }
2260 argv = nargv
2261 */
2262 nargv := []string{}
2263 nargv = append(nargv, {"+strings.Join(argv[1:], " ")+"})
2264 fmt.Printf("--D-- nargc=%d [%v]\n", len(nargv), nargv)
2265 argv = nargv
2266
2267 xargv := []string{"rex-exec", "@"+gsh.RSERV, "GET"}
2268 xargv = append(xargv, argv...)
2269 xargv = append(xargv, "/dev/tty")
2270 rcode, stat := gsh.RexecClient(xargv)
2271 if (rcode / 100) == 2 {
2272     fmt.Printf("--I-- OK Rexec (%v) %v\n", rcode, stat)
2273 } else {
2274     fmt.Printf("--I-- NG Rexec (%v) %v\n", rcode, stat)
2275 }
2276 }
2277 func (gsh*GshContext)Rchdir(argv []string){
2278     if len(argv) <= 1 {
2279         return
2280     }
2281     cwd, _ := os.Getwd()
2282     os.Chdir(gsh.RWD)
2283     os.Chdir(argv[1])
2284     twd, _ := os.Getwd()
2285     gsh.RWD = twd
2286     fmt.Printf("--I-- JWD=%v\n", twd)
2287     os.Chdir(cwd)
2288 }
2289 func (gsh*GshContext)Rpwd(argv []string){
2290     fmt.Printf("%v\n", gsh.RWD)
2291 }
2292 func (gsh*GshContext)Rls(argv []string){
2293     cwd, _ := os.Getwd()
2294     os.Chdir(gsh.RWD)
2295     argv[0] = "-ls"
2296     gsh.xFind(argv)
2297     os.Chdir(cwd)
2298 }
2299 func (gsh*GshContext)Rput(argv []string){
2300     var local string = ""
2301     var remote string = ""
2302     if 1 < len(argv) {
2303         local = argv[1]
2304         remote = local // base name
2305     }
2306     if 2 < len(argv) {
2307         remote = argv[2]
2308     }
2309     fmt.Printf("--I-- jput from=%v to=%v\n", local, gsh.Trelpath(remote))
2310 }
2311 func (gsh*GshContext)Rget(argv []string){
2312     var remote string = ""
2313     var local string = ""
2314     if 1 < len(argv) {
2315         remote = argv[1]
2316         local = remote // base name
2317     }
2318     if 2 < len(argv) {
2319         local = argv[2]
2320     }
2321     fmt.Printf("--I-- jget from=%v to=%v\n", gsh.Trelpath(remote), local)
2322 }
2323
2324 // <a name="network">network</a>
2325 // -s, -si, -so // bi-directional, source, sync (maybe socket)
2326 func (gshCtx*GshContext)sconnect(inTCP bool, argv []string) {
2327     gshPA := gshCtx.gshPA
2328     if len(argv) < 2 {
2329         fmt.Printf("Usage: -s [host]:[port[.udp]]\n")
2330         return
2331     }
2332     remote := argv[1]
2333     if remote == ":" { remote = "0.0.0.0:9999" }
2334
2335     if inTCP { // TCP
2336         dport, err := net.ResolveTCPAddr("tcp", remote);
2337         if err != nil {
2338             fmt.Printf("Address error: %s (%s)\n", remote, err)
2339             return
2340         }
2341         conn, err := net.DialTCP("tcp", nil, dport)
2342         if err != nil {
2343             fmt.Printf("Connection error: %s (%s)\n", remote, err)
2344             return
2345         }
2346         file, _ := conn.File();
2347         fd := file.Fd()
2348         fmt.Printf("Socket: connected to %s, socket[%d]\n", remote, fd)
2349
2350         savfd := gshPA.Files[1]
2351         gshPA.Files[1] = fd;
2352         gshCtx.gshelly(argv[2:])
2353         gshPA.Files[1] = savfd
2354         file.Close()
2355         conn.Close()
2356     } else {
2357         //dport, err := net.ResolveUDPAddr("udp4", remote);
2358         dport, err := net.ResolveUDPAddr("udp", remote);
2359         if err != nil {
2360             fmt.Printf("Address error: %s (%s)\n", remote, err)
2361             return
2362         }
2363         //conn, err := net.DialUDP("udp4", nil, dport)
2364         conn, err := net.DialUDP("udp", nil, dport)
2365         if err != nil {
2366             fmt.Printf("Connection error: %s (%s)\n", remote, err)
2367             return
2368         }
2369         file, _ := conn.File();
2370         fd := file.Fd()
2371
2372         ar := conn.RemoteAddr()
2373         //al := conn.LocalAddr()
2374         fmt.Printf("Socket: connected to %s [%s], socket[%d]\n",

```

```

2375         remote,ar.String(),fd)
2376
2377         savfd := gshPA.Files[1]
2378         gshPA.Files[1] = fd;
2379         gshCtx.gshellyv(argv[2:])
2380         gshPA.Files[1] = savfd
2381         file.Close()
2382         conn.Close()
2383     }
2384 }
2385 func (gshCtx*GshContext)saccept(inTCP bool, argv []string) {
2386     gshPA := gshCtx.gshPA
2387     if len(argv) < 2 {
2388         fmt.Printf("Usage: -ac [host]:[port[:udp]]\n")
2389         return
2390     }
2391     local := argv[1]
2392     if local == "" { local = "0.0.0.0:9999" }
2393     if inTCP { // TCP
2394         port, err := net.ResolveTCPAddr("tcp",local);
2395         if err != nil {
2396             fmt.Printf("Address error: %s (%s)\n",local,err)
2397             return
2398         }
2399         //fmt.Printf("Listen at %s...\n",local);
2400         sconn, err := net.ListenTCP("tcp", port)
2401         if err != nil {
2402             fmt.Printf("Listen error: %s (%s)\n",local,err)
2403             return
2404         }
2405         //fmt.Printf("Accepting at %s...\n",local);
2406         aconn, err := sconn.AcceptTCP()
2407         if err != nil {
2408             fmt.Printf("Accept error: %s (%s)\n",local,err)
2409             return
2410         }
2411         file, _ := aconn.File()
2412         fd := file.Fd()
2413         fmt.Printf("Accepted TCP at %s [%d]\n",local,fd)
2414
2415         savfd := gshPA.Files[0]
2416         gshPA.Files[0] = fd;
2417         gshCtx.gshellyv(argv[2:])
2418         gshPA.Files[0] = savfd
2419
2420         sconn.Close();
2421         aconn.Close();
2422         file.Close();
2423     }else{
2424         //port, err := net.ResolveUDPAddr("udp4",local);
2425         port, err := net.ResolveUDPAddr("udp",local);
2426         if err != nil {
2427             fmt.Printf("Address error: %s (%s)\n",local,err)
2428             return
2429         }
2430         fmt.Printf("Listen UDP at %s...\n",local);
2431         //uconn, err := net.ListenUDP("udp4", port)
2432         uconn, err := net.ListenUDP("udp", port)
2433         if err != nil {
2434             fmt.Printf("Listen error: %s (%s)\n",local,err)
2435             return
2436         }
2437         file, _ := uconn.File()
2438         fd := file.Fd()
2439         ar := uconn.RemoteAddr()
2440         remote := ""
2441         if ar != nil { remote = ar.String() }
2442         if remote == "" { remote = "?" }
2443
2444         // not yet received
2445         //fmt.Printf("Accepted at %s [%d] <- %s\n",local,fd,"")
2446
2447         savfd := gshPA.Files[0]
2448         gshPA.Files[0] = fd;
2449         savenv := gshPA.Env
2450         gshPA.Env = append(savenv, "REMOTE_HOST="+remote)
2451         gshCtx.gshellyv(argv[2:])
2452         gshPA.Env = savenv
2453         gshPA.Files[0] = savfd
2454
2455         uconn.Close();
2456         file.Close();
2457     }
2458 }
2459
2460 // empty line command
2461 func (gshCtx*GshContext)xPwd(argv[]string){
2462     // execute context command, pwd + date
2463     // context notation, representation scheme, to be resumed at re-login
2464     cwd, _ := os.Getwd()
2465     switch {
2466     case isin("-a",argv):
2467         gshCtx.ShowChdirHistory(argv)
2468     case isin("-ls",argv):
2469         showFileInfo(cwd,argv)
2470     default:
2471         fmt.Printf("%s\n",cwd)
2472     case isin("-v",argv): // obsolete empty command
2473         t := time.Now()
2474         date := t.Format(time.UnixDate)
2475         exe, _ := os.Executable()
2476         host, _ := os.Hostname()
2477         fmt.Printf("PWD=\"%s\" ",cwd)
2478         fmt.Printf("HOST=\"%s\" ",host)
2479         fmt.Printf("DATE=\"%s\" ",date)
2480         fmt.Printf("TIME=\"%s\" ",t.String())
2481         fmt.Printf("PID=\"%d\" ",os.Getpid())
2482         fmt.Printf("EXE=\"%s\" ",exe)
2483         fmt.Printf("\n")
2484     }
2485 }
2486
2487 // <a name="history">History</a>
2488 // these should be browsed and edited by HTTP browser
2489 // show the time of command with -t and direcotry with -ls
2490 // openfile-history, sort by -a -m -c
2491 // sort by elapsed time by -t -s
2492 // search by "more" like interface
2493 // edit history
2494 // sort history, and wc or unig
2495 // CPU and other resource consumptions
2496 // limit showing range (by time or so)
2497 // export / import history
2498 func (gshCtx *GshContext)xHistory(argv []string){
2499     atWorkDir := -1

```

```

2500 if l < len(argv) && strBegins(argv[l], "e") {
2501     atWorkDirX, _ = strconv.Atoi(argv[l][1:])
2502 }
2503 //fmt.Printf("--D-- showHistory(%v)\n", argv)
2504 for i, v := range gshCtx.CommandHistory {
2505     // exclude commands not to be listed by default
2506     // internal commands may be suppressed by default
2507     if v.CmdLine == "" && !isin("-a", argv) {
2508         continue;
2509     }
2510     if 0 <= atWorkDirX {
2511         if v.WorkDirX != atWorkDirX {
2512             continue
2513         }
2514     }
2515     if !isin("-n", argv) { // like "fc"
2516         fmt.Printf("!%-2d ", i)
2517     }
2518     if isin("-v", argv) {
2519         fmt.Println(v) // should be with it date
2520     } else {
2521         if isin("-l", argv) || isin("-l0", argv) {
2522             elps := v.EndAt.Sub(v.StartAt);
2523             start := v.StartAt.Format(time.Stamp)
2524             fmt.Printf("@%d ", v.WorkDirX)
2525             fmt.Printf("[%v] %11v/t ", start, elps)
2526         }
2527         if isin("-l", argv) && !isin("-l0", argv) {
2528             fmt.Printf("%v", Rusagef("%t %u\t// %s", argv, v.Rusagev))
2529         }
2530         if isin("-at", argv) { // isin("-ls", argv) {
2531             dhi := v.WorkDirX // workdir history index
2532             fmt.Printf("@%d %s\t", dhi, v.WorkDir)
2533             // show the FileInfo of the output command??
2534         }
2535         fmt.Printf("%s", v.CmdLine)
2536         fmt.Printf("\n")
2537     }
2538 }
2539 }
2540 // ln - history index
2541 func searchHistory(gshCtx GshContext, gline string) (string, bool, bool) {
2542     if gline[0] == 'l' {
2543         hix, err := strconv.Atoi(gline[1:])
2544         if err != nil {
2545             fmt.Printf("--E-- (%s : range)\n", hix)
2546             return "", false, true
2547         }
2548         if hix < 0 || len(gshCtx.CommandHistory) <= hix {
2549             fmt.Printf("--E-- (%d : out of range)\n", hix)
2550             return "", false, true
2551         }
2552         return gshCtx.CommandHistory[hix].CmdLine, false, false
2553     }
2554     // search
2555     //for i, v := range gshCtx.CommandHistory {
2556     //}
2557     return gline, false, false
2558 }
2559 func (gsh*GshContext)cmdStringInHistory(hix int)(cmd string, ok bool){
2560     if 0 <= hix && hix < len(gsh.CommandHistory) {
2561         return gsh.CommandHistory[hix].CmdLine, true
2562     }
2563     return "", false
2564 }
2565 }
2566 // temporary adding to PATH environment
2567 // cd name -lib for LD_LIBRARY_PATH
2568 // chdir with directory history (date + full-path)
2569 // -s for sort option (by visit date or so)
2570 func (gsh*GshContext)ShowChdirHistory(i int, v CChdirHistory, argv []string){
2571     fmt.Printf("!%-2d ", v.CmdIndex) // the first command at this WorkDir
2572     fmt.Printf("@%d ", i)
2573     fmt.Printf("[%v] ", v.MovedAt.Format(time.Stamp))
2574     showFileInfo(v.Dir, argv)
2575 }
2576 func (gsh*GshContext)ShowChdirHistory(argv []string){
2577     for i, v := range gsh.ChdirHistory {
2578         gsh.ShowChdirHistory1(i, v, argv)
2579     }
2580 }
2581 func skipOpts(argv []string)(int){
2582     for i, v := range argv {
2583         if strBegins(v, "-") {
2584             }else{
2585                 return i
2586             }
2587     }
2588     return -1
2589 }
2590 func (gshCtx*GshContext)xChdir(argv []string){
2591     cdhist := gshCtx.ChdirHistory
2592     if isin("? ", argv) || isin("-t", argv) || isin("-a", argv) {
2593         gshCtx.ShowChdirHistory(argv)
2594         return
2595     }
2596     pwd, _ := os.Getwd()
2597     dir := ""
2598     if len(argv) <= 1 {
2599         dir = toFullpath("-")
2600     }else{
2601         i := skipOpts(argv[1:])
2602         if i < 0 {
2603             dir = toFullpath("-")
2604         }else{
2605             dir = argv[1+i]
2606         }
2607     }
2608     if strBegins(dir, "@") {
2609         if dir == "@0" { // obsolete
2610             dir = gshCtx.StartDir
2611         }else{
2612             if dir == "@1" {
2613                 index := len(cdhist) - 1
2614                 if 0 < index { index -= 1 }
2615                 dir = cdhist[index].Dir
2616             }else{
2617                 index, err := strconv.Atoi(dir[1:])
2618                 if err != nil {
2619                     fmt.Printf("--E-- xChdir(%v)\n", err)
2620                     dir = "?"
2621                 }else{
2622                     if len(gshCtx.ChdirHistory) <= index {
2623                         fmt.Printf("--E-- xChdir(history range error)\n")
2624                         dir = "?"

```

```

2625         }else{
2626             dir = cdhist[index].Dir
2627         }
2628     }
2629 }
2630 if dir != "?" {
2631     err := os.Chdir(dir)
2632     if err != nil {
2633         fmt.Printf("--E-- xChdir(%s)(%v)\n",argv[1],err)
2634     }else{
2635         cwd, _ := os.Getwd()
2636         if cwd != pwd {
2637             hist1 := GChdirHistory { }
2638             hist1.Dir = cwd
2639             hist1.MovedAt = time.Now()
2640             hist1.CmdIndex = len(gshCtx.CommandHistory)+1
2641             gshCtx.ChdirHistory = append(cdhist,hist1)
2642             if !isin("-s",argv){
2643                 //cwd, _ := os.Getwd()
2644                 //fmt.Printf("%s\n",cwd)
2645                 ix := len(gshCtx.ChdirHistory)-1
2646                 gshCtx.ShowChdirHistory1(ix,hist1,argv)
2647             }
2648         }
2649     }
2650 }
2651 if isin("-ls",argv){
2652     cwd, _ := os.Getwd()
2653     showFileInfo(cwd,argv);
2654 }
2655 }
2656 func TimeValSub(tv1 *syscall.Timeval, tv2 *syscall.Timeval){
2657     *tv1 = syscall.NsecToTimeval(tv1.Nano() - tv2.Nano())
2658 }
2659 func RusageSubv(ru1, ru2 [2]syscall.Rusage) ([2]syscall.Rusage){
2660     TimeValSub(&ru1[0].Utime,&ru2[0].Utime)
2661     TimeValSub(&ru1[0].Stime,&ru2[0].Stime)
2662     TimeValSub(&ru1[1].Utime,&ru2[1].Utime)
2663     TimeValSub(&ru1[1].Stime,&ru2[1].Stime)
2664     return ru1
2665 }
2666 func TimeValAdd(tv1 syscall.Timeval, tv2 syscall.Timeval)(syscall.Timeval){
2667     tvs := syscall.NsecToTimeval(tv1.Nano() + tv2.Nano())
2668     return tvs
2669 }
2670 /*
2671 func RusageAddv(ru1, ru2 [2]syscall.Rusage) ([2]syscall.Rusage){
2672     TimeValAdd(&ru1[0].Utime,&ru2[0].Utime)
2673     TimeValAdd(&ru1[0].Stime,&ru2[0].Stime)
2674     TimeValAdd(&ru1[1].Utime,&ru2[1].Utime)
2675     TimeValAdd(&ru1[1].Stime,&ru2[1].Stime)
2676     return ru1
2677 }
2678 */
2679
2680 // <a name="rusage">Resource Usage</a>
2681 func sRusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2682     // ru[0] self , ru[1] children
2683     ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
2684     st := TimeValAdd(ru[0].Stime,ru[1].Stime)
2685     uu := (ut.Sec*1000000 + int64(ut.Usec)) * 1000
2686     su := (st.Sec*1000000 + int64(st.Usec)) * 1000
2687     tu := uu + su
2688     ret := fmt.Sprintf("%v/sum",abftime(tu))
2689     ret += fmt.Sprintf(" %v/usr",abftime(uu))
2690     ret += fmt.Sprintf(" %v/sys",abftime(su))
2691     return ret
2692 }
2693 func Rusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2694     ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
2695     st := TimeValAdd(ru[0].Stime,ru[1].Stime)
2696     fmt.Printf("%d.%06ds/u ",ut.Sec,ut.Usec) //ru[1].Utime.Sec,ru[1].Utime.Usec)
2697     fmt.Printf("%d.%06ds/s ",st.Sec,st.Usec) //ru[1].Stime.Sec,ru[1].Stime.Usec)
2698     return ""
2699 }
2700 func Getrusagev()([2]syscall.Rusage){
2701     var ruv = [2]syscall.Rusage{}
2702     syscall.Getrusage(syscall.RUSAGE_SELF,&ruv[0])
2703     syscall.Getrusage(syscall.RUSAGE_CHILDREN,&ruv[1])
2704     return ruv
2705 }
2706 func showRusage(what string,argv []string, ru *syscall.Rusage){
2707     fmt.Printf("%s: ",what);
2708     fmt.Printf("Utr=%d.%06ds",ru.Utime.Sec,ru.Utime.Usec)
2709     fmt.Printf(" Sys=%d.%06ds",ru.Stime.Sec,ru.Stime.Usec)
2710     fmt.Printf(" Rss=%vB",ru.Maxrss)
2711     if isin("-l",argv) {
2712         fmt.Printf(" MinFlt=%v",ru.Minflt)
2713         fmt.Printf(" MajFlt=%v",ru.Majflt)
2714         fmt.Printf(" Ixrss=%vB",ru.Ixrss)
2715         fmt.Printf(" IdRSS=%vB",ru.Idrss)
2716         fmt.Printf(" Nswap=%vB",ru.Nswap)
2717         fmt.Printf(" Read=%v",ru.Inblock)
2718         fmt.Printf(" Write=%v",ru.Oublock)
2719     }
2720     fmt.Printf(" Snd=%v",ru.Msgsnd)
2721     fmt.Printf(" Rcv=%v",ru.Msgrcv)
2722     //if isin("-l",argv) {
2723         fmt.Printf(" Sig=%v",ru.Nsignals)
2724     //}
2725     fmt.Printf("\n");
2726 }
2727 func (gshCtx *GshContext)xTime(argv []string)(bool){
2728     if 2 <= len(argv){
2729         gshCtx.LastRusage = syscall.Rusage{}
2730         rusagev1 := Getrusagev()
2731         fin := gshCtx.gshellv(argv[1:])
2732         rusagev2 := Getrusagev()
2733         showRusage(argv[1],argv,&gshCtx.LastRusage)
2734         rusagev := RusageSubv(rusagev2,rusagev1)
2735         showRusage("self",argv,&rusagev[0])
2736         showRusage("chld",argv,&rusagev[1])
2737         return fin
2738     }else{
2739         rusage:= syscall.Rusage {}
2740         syscall.Getrusage(syscall.RUSAGE_SELF,&rusage)
2741         showRusage("self",argv, &rusage)
2742         syscall.Getrusage(syscall.RUSAGE_CHILDREN,&rusage)
2743         showRusage("chld",argv, &rusage)
2744         return false
2745     }
2746 }
2747 func (gshCtx *GshContext)xJobs(argv []string){
2748     fmt.Printf("%d Jobs\n",len(gshCtx.BackgroundJobs))
2749     for ji, pid := range gshCtx.BackgroundJobs {

```

```

2750 //wstat := syscall.WaitStatus {0}
2751 rusage := syscall.Rusage {}
2752 //wpid, err := syscall.Wait4(pid,&wstat,syscall.WNOHANG,&rusage);
2753 wpid, err := syscall.Wait4(pid,nil,syscall.WNOHANG,&rusage);
2754 if err != nil {
2755     fmt.Printf("--E-- %%%d [%d] (%v)\n",ji,pid,err)
2756 }else{
2757     fmt.Printf("%%d[%d](%d)\n",ji,pid,wpid)
2758     showRusage("chld",argv,&rusage)
2759 }
2760 }
2761 }
2762 func (gsh*GshContext)inBackground(argv[]string)(bool){
2763     if gsh.CmdTrace { fmt.Printf("--I-- inBackground(%v)\n",argv) }
2764     gsh.BackGround = true // set background option
2765     xfin := false
2766     xfin = gsh.gshelly(argv)
2767     gsh.BackGround = false
2768     return xfin
2769 }
2770 // -o file without command means just opening it and refer by #N
2771 // should be listed by "files" command
2772 func (gshCtx*GshContext)xOpen(argv[]string){
2773     var pv = []int{-1,-1}
2774     err := syscall.Pipe(pv)
2775     fmt.Printf("--I-- pipe()=[%d,%d](%v)\n",pv[0],pv[1],err)
2776 }
2777 func (gshCtx*GshContext)fromPipe(argv[]string){
2778 }
2779 func (gshCtx*GshContext)xClose(argv[]string){
2780 }
2781 }
2782 // <a name="redirect">redirect</a>
2783 func (gshCtx*GshContext)redirect(argv[]string)(bool){
2784     if len(argv) < 2 {
2785         return false
2786     }
2787 }
2788 cmd := argv[0]
2789 fname := argv[1]
2790 var file *os.File = nil
2791
2792 fdix := 0
2793 mode := os.O_RDONLY
2794
2795 switch {
2796 case cmd == "-i" || cmd == "<":
2797     fdix = 0
2798     mode = os.O_RDONLY
2799 case cmd == "-o" || cmd == ">":
2800     fdix = 1
2801     mode = os.O_RDWR | os.O_CREATE
2802 case cmd == "-a" || cmd == ">>":
2803     fdix = 1
2804     mode = os.O_RDWR | os.O_CREATE | os.O_APPEND
2805 }
2806 if fname[0] == '#' {
2807     fd, err := strconv.Atoi(fname[1:])
2808     if err != nil {
2809         fmt.Printf("--E-- (%v)\n",err)
2810         return false
2811     }
2812     file = os.NewFile(uintptr(fd),"MaybePipe")
2813 }else{
2814     xfile, err := os.OpenFile(argv[1], mode, 0600)
2815     if err != nil {
2816         fmt.Printf("--E-- (%s)\n",err)
2817         return false
2818     }
2819     file = xfile
2820 }
2821 gshPA := gshCtx.gshPA
2822 savfd := gshPA.Files[fdix]
2823 gshPA.Files[fdix] = file.Fd()
2824 fmt.Printf("--I-- Opened [%d] %s\n",file.Fd(),argv[1])
2825 gshCtx.gshelly(argv[2:])
2826 gshPA.Files[fdix] = savfd
2827 }
2828 return false
2829 }
2830 }
2831 //fmt.Fprintf(res, "GShell Status: %q", html.EscapeString(req.URL.Path))
2832 func httpHandler(res http.ResponseWriter, req *http.Request){
2833     path := req.URL.Path
2834     fmt.Printf("--I-- Got HTTP Request(%s)\n",path)
2835     {
2836         gshCtxBuf, _ := setupGshContext()
2837         gshCtx := &gshCtxBuf
2838         fmt.Printf("--I-- %s\n",path[1:])
2839         gshCtx.tgshelll(path[1:])
2840     }
2841     fmt.Fprintf(res, "Hello(^-^)\n%s\n",path)
2842 }
2843 func (gshCtx *GshContext) httpServer(argv []string){
2844     http.HandleFunc("/", httpHandler)
2845     accport := "localhost:9999"
2846     fmt.Printf("--I-- HTTP Server Start at [%s]\n",accport)
2847     http.ListenAndServe(accport,nil)
2848 }
2849 func (gshCtx *GshContext)xGo(argv[]string){
2850     go gshCtx.gshelly(argv[1:]);
2851 }
2852 func (gshCtx *GshContext) xPs(argv[]string)(){
2853 }
2854 }
2855 // <a name="plugin">Plugin</a>
2856 // plugin [-ls [names]] to list plugins
2857 // Reference: <a href="https://golang.org/src/plugin/">plugin</a> source code
2858 func (gshCtx *GshContext) whichPlugin(name string,argv[]string)(pi *PluginInfo){
2859     pi = nil
2860     for _,p := range gshCtx.PluginFuncs {
2861         if p.Name == name && pi == nil {
2862             pi = *p
2863         }
2864         if !isin("-s",argv){
2865             //fmt.Printf("%v %v ",i,p)
2866             if isin("-ls",argv){
2867                 showFileInfo(p.Path,argv)
2868             }else{
2869                 fmt.Printf("%s\n",p.Name)
2870             }
2871         }
2872     }
2873     return pi
2874 }

```

```

2875 func (gshCtx *GshContext) xPlugin(argv[]string) (error) {
2876     if len(argv) == 0 || argv[0] == "-ls" {
2877         gshCtx.whichPlugin("",argv)
2878         return nil
2879     }
2880     name := argv[0]
2881     Pin := gshCtx.whichPlugin(name,[]string{"-s"})
2882     if Pin != nil {
2883         os.Args = argv // should be recovered?
2884         Pin.Addr.(func())()
2885         return nil
2886     }
2887     sofile := toFullpath(argv[0] + ".so") // or find it by which($PATH)
2888
2889     p, err := plugin.Open(sofile)
2890     if err != nil {
2891         fmt.Printf("--E-- plugin.Open(%s)(%v)\n",sofile,err)
2892         return err
2893     }
2894     fname := "Main"
2895     f, err := p.Lookup(fname)
2896     if( err != nil ){
2897         fmt.Printf("--E-- plugin.Lookup(%s)(%v)\n",fname,err)
2898         return err
2899     }
2900     pin := PluginInfo {p,f,name,sofile}
2901     gshCtx.PluginFuncs = append(gshCtx.PluginFuncs,pin)
2902     fmt.Printf("--I-- added (%d)\n",len(gshCtx.PluginFuncs))
2903
2904     //fmt.Printf("--I-- first call(%s:%s)%v\n",sofile,fname,argv)
2905     os.Args = argv
2906     f.(func())()
2907     return err
2908 }
2909 func (gshCtx*GshContext)Args(argv[]string){
2910     for i,v := range os.Args {
2911         fmt.Printf("[%v] %v\n",i,v)
2912     }
2913 }
2914 func (gshCtx *GshContext) showVersion(argv[]string){
2915     if isin("-l",argv) {
2916         fmt.Printf("%v/%v (%v)",NAME,VERSION,DATE);
2917     }else{
2918         fmt.Printf("%v",VERSION);
2919     }
2920     if isin("-a",argv) {
2921         fmt.Printf(" %s",AUTHOR)
2922     }
2923     if !isin("-n",argv) {
2924         fmt.Printf("\n")
2925     }
2926 }
2927
2928 // <a name="scanf">Scanf</a> // string decomposer
2929 // scanf [format] [input]
2930 func scanv(sstr string)(strv[]string){
2931     strv = strings.Split(sstr," ")
2932     return strv
2933 }
2934 func scanUntil(src,end string)(rstr string,leng int){
2935     idx := strings.Index(src,end)
2936     if 0 <= idx {
2937         rstr = src[0:idx]
2938         return rstr,idx+leng(end)
2939     }
2940     return src,0
2941 }
2942
2943 // -bn -- display base-name part only // can be in some %fmt, for sed rewriting
2944 func (gsh*GshContext)printVal(fmts string, vstr string, optv[]string){
2945     //vint,err := strconv.Atoi(vstr)
2946     var ival int64 = 0
2947     n := 0
2948     err := error(nil)
2949     if strBegins(vstr,"_") {
2950         vx, _ := strconv.Atoi(vstr[1:])
2951         if vx < len(gsh.iValues) {
2952             vstr = gsh.iValues[vx]
2953         }else{
2954         }
2955     }
2956     // should use Eval()
2957     if strBegins(vstr,"0x") {
2958         n,err = fmt.Sscanf(vstr[2:],"%x",&ival)
2959     }else{
2960         n,err = fmt.Sscanf(vstr,"%d",&ival)
2961     }//fmt.Printf("--D-- n=%d err=(%v) (%s)=%v\n",n,err,vstr, ival)
2962 }
2963 if n == 1 && err == nil {
2964     //fmt.Printf("--D-- formatn(%v) ival(%v)\n",fmts,ival)
2965     fmt.Printf("%"+fmts,ival)
2966 }else{
2967     if isin("-bn",optv){
2968         fmt.Printf("%"+fmts,filepath.Base(vstr))
2969     }else{
2970         fmt.Printf("%"+fmts,vstr)
2971     }
2972 }
2973 }
2974 func (gsh*GshContext)printfv(fmts,div string,argv[]string,optv[]string,list[]string){
2975     //fmt.Printf("%d",len(list))
2976     //curfmt := "v"
2977     outlen := 0
2978     curfmt := gsh.iFormat
2979
2980     if 0 < len(fmts) {
2981         for xi := 0; xi < len(fmts); xi++ {
2982             fch := fmts[xi]
2983             if fch == '%' {
2984                 if xi+1 < len(fmts) {
2985                     curfmt = string(fmts[xi+1])
2986                 }
2987                 gsh.iFormat = curfmt
2988                 xi += 1
2989                 if xi+1 < len(fmts) && fmts[xi+1] == '(' {
2990                     vals,leng := scanUntil(fmts[xi+2:],")")
2991                     //fmt.Printf("--D-- show fmt(%v) val(%v) next(%v)\n",curfmt,vals,leng)
2992                     gsh.printVal(curfmt,vals,optv)
2993                     xi += 2+leng-1
2994                 }
2995                 outlen += 1
2996                 continue
2997             }
2998             if fch == '_' {
2999                 hi,leng := scanInt(fmts[xi+1:])

```



```

3000         if 0 < leng {
3001             if hi < len(gsh.iValues) {
3002                 gsh.printVal(curfmt, gsh.iValues[hi], optv)
3003                 outlen += 1 // should be the real length
3004             }else{
3005                 fmt.Printf("(out-range)")
3006             }
3007             xi += leng
3008             continue;
3009         }
3010     }
3011     fmt.Printf("%c", fch)
3012     outlen += 1
3013 }
3014 }else{
3015     //fmt.Printf("--D-- print (%s)\n")
3016     for i,v := range list {
3017         if 0 < i {
3018             fmt.Printf(div)
3019         }
3020         gsh.printVal(curfmt, v, optv)
3021         outlen += 1
3022     }
3023 }
3024 if 0 < outlen {
3025     fmt.Printf("\n")
3026 }
3027 }
3028 func (gsh*GshContext)Scanv(argv[]string){
3029     //fmt.Printf("--D-- Scnav(%v)\n", argv)
3030     if len(argv) == 1 {
3031         return
3032     }
3033     argv = argv[1:]
3034     fmts := ""
3035     if strBegins(argv[0], "-F") {
3036         fmts = argv[0]
3037         gsh.iDelimiter = fmts
3038         argv = argv[1:]
3039     }
3040     input := strings.Join(argv, " ")
3041     if fmts == "" { // simple decomposition
3042         v := scanv(input)
3043         gsh.iValues = v
3044         //fmt.Printf("%v\n", strings.Join(v, ","))
3045     }else{
3046         v := make([]string, 8)
3047         n, err := fmt.Sscanf(input, fmts, &v[0], &v[1], &v[2], &v[3])
3048         fmt.Printf("--D-- Sscanf ->(%v) n=%d err=(%v)\n", v, n, err)
3049         gsh.iValues = v
3050     }
3051 }
3052 func (gsh*GshContext)Printv(argv[]string){
3053     if false { //@@U
3054         fmt.Printf("%v\n", strings.Join(argv[1:], " "))
3055         return
3056     }
3057     //fmt.Printf("--D-- Printv(%v)\n", argv)
3058     //fmt.Printf("%v\n", strings.Join(gsh.iValues, ","))
3059     div := gsh.iDelimiter
3060     fmts := ""
3061     argv = argv[1:]
3062     if 0 < len(argv) {
3063         if strBegins(argv[0], "-F") {
3064             div = argv[0][2:]
3065             argv = argv[1:]
3066         }
3067     }
3068 }
3069 optv := []string{}
3070 for _,v := range argv {
3071     if strBegins(v, "-"){
3072         optv = append(optv, v)
3073         argv = argv[1:]
3074     }else{
3075         break;
3076     }
3077 }
3078 if 0 < len(argv) {
3079     fmts = strings.Join(argv, " ")
3080 }
3081 gsh.printfv(fmts, div, argv, optv, gsh.iValues)
3082 }
3083 func (gsh*GshContext)Basename(argv[]string){
3084     for i,v := range gsh.iValues {
3085         gsh.iValues[i] = filepath.Base(v)
3086     }
3087 }
3088 func (gsh*GshContext)Sortv(argv[]string){
3089     sv := gsh.iValues
3090     sort.Slice(sv, func(i,j int) bool {
3091         return sv[i] < sv[j]
3092     })
3093 }
3094 func (gsh*GshContext)Shiftv(argv[]string){
3095     vi := len(gsh.iValues)
3096     if 0 < vi {
3097         if isin("-r", argv) {
3098             top := gsh.iValues[0]
3099             gsh.iValues = append(gsh.iValues[1:], top)
3100         }else{
3101             gsh.iValues = gsh.iValues[1:]
3102         }
3103     }
3104 }
3105 }
3106 func (gsh*GshContext)Enq(argv[]string){
3107 }
3108 func (gsh*GshContext)Deq(argv[]string){
3109 }
3110 func (gsh*GshContext)Push(argv[]string){
3111     gsh.iValStack = append(gsh.iValStack, argv[1:])
3112     fmt.Printf("depth=%d\n", len(gsh.iValStack))
3113 }
3114 func (gsh*GshContext)Dump(argv[]string){
3115     for i,v := range gsh.iValStack {
3116         fmt.Printf("%d %v\n", i, v)
3117     }
3118 }
3119 func (gsh*GshContext)Pop(argv[]string){
3120     depth := len(gsh.iValStack)
3121     if 0 < depth {
3122         v := gsh.iValStack[depth-1]
3123         if isin("-cat", argv){
3124             gsh.iValues = append(gsh.iValues, v...)

```

```

3125     }else{
3126         gsh.iValues = v
3127     }
3128     gsh.iValStack = gsh.iValStack[0:depth-1]
3129     fmt.Printf("depth=%d %s\n", len(gsh.iValStack), gsh.iValues)
3130 }else{
3131     fmt.Printf("depth=%d\n", depth)
3132 }
3133 }
3134
3135 // <a name="interpreter">Command Interpreter</a>
3136 func (gshCtx*GshContext)gshellv(argv []string) (fin bool) {
3137     fin = false
3138
3139     if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr, "--I-- gshellv(%d)\n", len(argv)) }
3140     if len(argv) <= 0 {
3141         return false
3142     }
3143     xargv := []string{}
3144     for ai := 0; ai < len(argv); ai++ {
3145         xargv = append(xargv, strsubst(gshCtx, argv[ai], false))
3146     }
3147     argv = xargv
3148     if false {
3149         for ai := 0; ai < len(argv); ai++ {
3150             fmt.Printf("[%d] %s [%d]\n",
3151                 ai, argv[ai], len(argv[ai]), argv[ai])
3152         }
3153     }
3154     cmd := argv[0]
3155     if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr, "--I-- gshellv(%d)\n", len(argv), argv) }
3156     switch { // https://tour.golang.org/flowcontrol/11
3157     case cmd == "":
3158         gshCtx.xPwd([]string{}); // empty command
3159     case cmd == "-x":
3160         gshCtx.CmdTrace = ! gshCtx.CmdTrace
3161     case cmd == "-xt":
3162         gshCtx.CmdTime = ! gshCtx.CmdTime
3163     case cmd == "-ot":
3164         gshCtx.sconnect(true, argv)
3165     case cmd == "-ou":
3166         gshCtx.sconnect(false, argv)
3167     case cmd == "-it":
3168         gshCtx.saccept(true, argv)
3169     case cmd == "-iu":
3170         gshCtx.saccept(false, argv)
3171     case cmd == "-i" || cmd == "<" || cmd == "-o" || cmd == ">" || cmd == "-a" || cmd == ">>" || cmd == "-s" || cmd == "><":
3172         gshCtx.redirect(argv)
3173     case cmd == "|":
3174         gshCtx.fromPipe(argv)
3175     case cmd == "args":
3176         gshCtx.Args(argv)
3177     case cmd == "bg" || cmd == "-bg":
3178         rfin := gshCtx.inBackground(argv[1:])
3179         return rfin
3180     case cmd == "-bn":
3181         gshCtx.BaseName(argv)
3182     case cmd == "call":
3183         _ = gshCtx.excommand(false, argv[1:])
3184     case cmd == "cd" || cmd == "chdir":
3185         gshCtx.xChdir(argv)
3186     case cmd == "-cksum":
3187         gshCtx.xFind(argv)
3188     case cmd == "-sum":
3189         gshCtx.xFind(argv)
3190     case cmd == "-sumtest":
3191         str := ""
3192         if 1 < len(argv) { str = argv[1] }
3193         crc := strCRC32(str, uint64(len(str)))
3194         fprintf(stderr, "%v %v\n", crc, len(str))
3195     case cmd == "close":
3196         gshCtx.xClose(argv)
3197     case cmd == "gcp":
3198         gshCtx.FileCopy(argv)
3199     case cmd == "dec" || cmd == "decode":
3200         gshCtx.Dec(argv)
3201     case cmd == "#define":
3202     case cmd == "dic" || cmd == "d":
3203         xDic(argv)
3204     case cmd == "dump":
3205         gshCtx.Dump(argv)
3206     case cmd == "echo" || cmd == "e":
3207         echo(argv, true)
3208     case cmd == "enc" || cmd == "encode":
3209         gshCtx.Enc(argv)
3210     case cmd == "env":
3211         env(argv)
3212     case cmd == "eval":
3213         xEval(argv[1:], true)
3214     case cmd == "ev" || cmd == "events":
3215         dumpEvents(argv)
3216     case cmd == "exec":
3217         _ = gshCtx.excommand(true, argv[1:])
3218         // should not return here
3219     case cmd == "exit" || cmd == "quit":
3220         // write Result code EXIT to 3>
3221         return true
3222     case cmd == "fds":
3223         // dump the attributes of fds (of other process)
3224     case cmd == "-find" || cmd == "fin" || cmd == "ufind" || cmd == "uf":
3225         gshCtx.xFind(argv[1:])
3226     case cmd == "fu":
3227         gshCtx.xFind(argv[1:])
3228     case cmd == "fork":
3229         // mainly for a server
3230     case cmd == "-gen":
3231         gshCtx.gen(argv)
3232     case cmd == "-go":
3233         gshCtx.xGo(argv)
3234     case cmd == "-grep":
3235         gshCtx.xFind(argv)
3236     case cmd == "gdeg":
3237         gshCtx.Deq(argv)
3238     case cmd == "genq":
3239         gshCtx.Enq(argv)
3240     case cmd == "gpop":
3241         gshCtx.Pop(argv)
3242     case cmd == "gpush":
3243         gshCtx.Push(argv)
3244     case cmd == "history" || cmd == "hi": // hi should be alias
3245         gshCtx.xHistory(argv)
3246     case cmd == "jobs":
3247         gshCtx.xJobs(argv)
3248     case cmd == "lnsp" || cmd == "nlspl":
3249         gshCtx.SplitLine(argv)

```

```

3250 case cmd == "-ls":
3251     gshCtx.xFind(argv)
3252 case cmd == "hop":
3253     // do nothing
3254 case cmd == "pipe":
3255     gshCtx.xOpen(argv)
3256 case cmd == "plug" || cmd == "plugin" || cmd == "pin":
3257     gshCtx.xPlugin(argv[1:])
3258 case cmd == "print" || cmd == "-pr":
3259     // output internal slice // also sprintf should be
3260     gshCtx.Printv(argv)
3261 case cmd == "ps":
3262     gshCtx.xPs(argv)
3263 case cmd == "ps:title":
3264     // to be gsh.title
3265 case cmd == "rexeod" || cmd == "rexd":
3266     gshCtx.RexeServer(argv)
3267 case cmd == "rexoc" || cmd == "rex":
3268     gshCtx.RexeClient(argv)
3269 case cmd == "repeat" || cmd == "rep": // repeat cond command
3270     gshCtx.repeat(argv)
3271 case cmd == "replay":
3272     gshCtx.xReplay(argv)
3273 case cmd == "scan":
3274     // scan input (or so in fscanf) to internal slice (like Files or map)
3275     gshCtx.Scanv(argv)
3276 case cmd == "set":
3277     // set name ...
3278 case cmd == "serv":
3279     gshCtx.httpServer(argv)
3280 case cmd == "shift":
3281     gshCtx.Shiftv(argv)
3282 case cmd == "sleep":
3283     gshCtx.sleep(argv)
3284 case cmd == "-sort":
3285     gshCtx.Sortv(argv)
3286
3287 case cmd == "j" || cmd == "join":
3288     gshCtx.RJoin(argv)
3289 case cmd == "a" || cmd == "alpa":
3290     gshCtx.Rxec(argv)
3291 case cmd == "jcd" || cmd == "jchdir":
3292     gshCtx.Rchdir(argv)
3293 case cmd == "jget":
3294     gshCtx.Rget(argv)
3295 case cmd == "jls":
3296     gshCtx.Rls(argv)
3297 case cmd == "jput":
3298     gshCtx.Rput(argv)
3299 case cmd == "jpwd":
3300     gshCtx.Rpwd(argv)
3301
3302 case cmd == "time":
3303     fin = gshCtx.xTime(argv)
3304 case cmd == "ungets":
3305     if l < len(argv) {
3306         ungets(argv[l]+"\\n")
3307     }else{
3308     }
3309 case cmd == "pwd":
3310     gshCtx.xPwd(argv);
3311 case cmd == "ver" || cmd == "-ver" || cmd == "version":
3312     gshCtx.showVersion(argv)
3313 case cmd == "where":
3314     // data file or so?
3315 case cmd == "which":
3316     which("PATH",argv);
3317 default:
3318     if gshCtx.whichPlugin(cmd,[jstring{"-s"}]) != nil {
3319         gshCtx.xPlugin(argv)
3320     }else{
3321         notfound,_ := gshCtx.excommand(false,argv)
3322         if notfound {
3323             fmt.Printf("--E-- command not found (%v)\\n",cmd)
3324         }
3325     }
3326 }
3327 return fin
3328 }
3329 }
3330 func (gsh*GshContext)gshell(gline string) (rfin bool) {
3331     argv := strings.Split(string(gline)," ")
3332     fin := gsh.gshellv(argv)
3333     return fin
3334 }
3335 func (gsh*GshContext)tgshell(gline string)(xfin bool){
3336     start := time.Now()
3337     fin := gsh.gshell(gline)
3338     end := time.Now()
3339     elps := end.Sub(start);
3340     if gsh.CmdTime {
3341         fmt.Printf("--T-- " + time.Now().Format(time.Stamp) + " (%d.%09ds)\\n",
3342             elps/1000000000,elps%1000000000)
3343     }
3344     return fin
3345 }
3346 func Ttyid() (int) {
3347     fi, err := os.Stdin.Stat()
3348     if err != nil {
3349         return 0;
3350     }
3351     //fmt.Printf("Stdin: %v Dev=%d\\n",
3352     // fi.Mode(),fi.Mode()&os.ModeDevice)
3353     if (fi.Mode() & os.ModeDevice) != 0 {
3354         stat := syscall.Stat_t{};
3355         err := syscall.Fstat(0,&stat)
3356         if err != nil {
3357             //fmt.Printf("--I-- Stdin: (%v)\\n",err)
3358         }else{
3359             //fmt.Printf("--I-- Stdin: rdev=%d %d\\n",
3360             // stat.Rdev&0xFF,stat.Rdev);
3361             //fmt.Printf("--I-- Stdin: tty%d\\n",stat.Rdev&0xFF);
3362             return int(stat.Rdev & 0xFF)
3363         }
3364     }
3365     return 0
3366 }
3367 func (gshCtx *GshContext) ttyfile() string {
3368     //fmt.Printf("--I-- GSH_HOME=%s\\n",gshCtx.GshHomeDir)
3369     ttyfile := gshCtx.GshHomeDir + "/" + "gsh-tty" +
3370         fmt.Sprintf("%02d",gshCtx.TerminalId)
3371     //strconv.Itoa(gshCtx.TerminalId)
3372     //fmt.Printf("--I-- ttyfile=%s\\n",ttyfile)
3373     return ttyfile
3374 }

```

```

3375 func (gshCtx *GshContext) ttyline>(*os.File){
3376     file, err := os.OpenFile(gshCtx.ttyfile(),os.O_RDWR|os.O_CREATE|os.O_TRUNC,0600)
3377     if err != nil {
3378         fmt.Printf("--F-- cannot open %s (%s)\n",gshCtx.ttyfile(),err)
3379         return file;
3380     }
3381     return file
3382 }
3383 func (gshCtx *GshContext)getline(hix int, skipping bool, prevline string) (string) {
3384     if( skipping ){
3385         reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
3386         line, _, _ := reader.ReadLine()
3387         return string(line)
3388     }else
3389     if true {
3390         return xgetline(hix,prevline,gshCtx)
3391     }
3392     /*
3393     else
3394     if( with_exgetline && gshCtx.GetLine != "" ){
3395         //var xhix int64 = int64(hix); // cast
3396         newenv := os.Environ()
3397         newenv = append(newenv, "GSH_LINENO="+strconv.FormatInt(int64(hix),10) )
3398
3399         tty := gshCtx.ttyline()
3400         tty.WriteString(prevline)
3401         Pa := os.ProcAttr {
3402             "", // start dir
3403             newenv, //os.Environ(),
3404             []*os.File{os.Stdin,os.Stdout,os.Stderr,tty},
3405             nil,
3406         }
3407         //fmt.Printf("--I-- getline=%s // %s\n",gsh_getlinev[0],gshCtx.GetLine)
3408         proc, err := os.StartProcess(gsh_getlinev[0],[]string{"getline","getline"},&Pa)
3409         if err != nil {
3410             fmt.Printf("--F-- getline process error (%v)\n",err)
3411             // for ; { }
3412             return "exit (getline program failed)"
3413         }
3414         //stat, err := proc.Wait()
3415         proc.Wait()
3416         buff := make([]byte,LINESIZE)
3417         count, err := tty.Read(buff)
3418         //_, err = tty.Read(buff)
3419         //fmt.Printf("--D-- getline (%d)\n",count)
3420         if err != nil {
3421             if ! (count == 0) { // && err.String() == "EOF" } {
3422                 fmt.Printf("--E-- getline error (%s)\n",err)
3423             }
3424         }else{
3425             //fmt.Printf("--I-- getline OK \"%s\"\n",buff)
3426         }
3427         tty.Close()
3428         gline := string(buff[0:count])
3429         return gline
3430     }else
3431     */
3432     {
3433         // if isatty {
3434         //     fmt.Printf("%d",hix)
3435         //     fmt.Print(PROMPT)
3436         // }
3437         reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
3438         line, _, _ := reader.ReadLine()
3439         return string(line)
3440     }
3441 }
3442
3443 //== begin ===== getline
3444 /*
3445 * getline.c
3446 * 2020-0819 extracted from dog.c
3447 * getline.go
3448 * 2020-0822 ported to Go
3449 */
3450 /*
3451 package main // getline main
3452 import (
3453     "fmt" // <a href="https://golang.org/pkg/fmt/">fmt</a>
3454     "strings" // <a href="https://golang.org/pkg/strings/">strings</a>
3455     "os" // <a href="https://golang.org/pkg/os/">os</a>
3456     "syscall" // <a href="https://golang.org/pkg/syscall/">syscall</a>
3457     //"bytes" // <a href="https://golang.org/pkg/os/">os</a>
3458     //"os/exec" // <a href="https://golang.org/pkg/os/">os</a>
3459 )
3460 */
3461
3462 // C language compatibility functions
3463 var errno = 0
3464 var stdin *os.File = os.Stdin
3465 var stdout *os.File = os.Stdout
3466 var stderr *os.File = os.Stderr
3467 var EOF = -1
3468 var NULL = 0
3469 type FILE os.File
3470 type StrBuff []byte
3471 var NULL_FP *os.File = nil
3472 var NULLSP = 0
3473 //var LINESIZE = 1024
3474
3475 func system(cmdstr string)(int){
3476     PA := syscall.ProcAttr {
3477         "", // the starting directory
3478         os.Environ(),
3479         []uintptr{os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd()},
3480         nil,
3481     }
3482     argv := strings.Split(cmdstr, " ")
3483     pid,err := syscall.ForkExec(argv[0],argv,&PA)
3484     if( err != nil ){
3485         fmt.Printf("--E-- syscall(%v) err(%v)\n",cmdstr,err)
3486     }
3487     syscall.Wait4(pid,nil,0,nil)
3488
3489     /*
3490     argv := strings.Split(cmdstr, " ")
3491     fmt.Fprintf(os.Stderr,"--I-- system(%v)\n",argv)
3492     //cmd := exec.Command(argv[0:]...)
3493     cmd := exec.Command(argv[0],argv[1],argv[2])
3494     cmd.Stdin = strings.NewReader("output of system")
3495     var out bytes.Buffer
3496     cmd.Stdout = &out
3497     var serr bytes.Buffer
3498     cmd.Stderr = &serr
3499     err := cmd.Run()

```

```

3500     if err != nil {
3501         fmt.Fprintf(os.Stderr, "--E-- system(%v)err(%v)\n", argv, err)
3502         fmt.Printf("ERR:%s\n", serr.String())
3503     }else{
3504         fmt.Printf("%s", out.String())
3505     }
3506     /*
3507     return 0
3508 }
3509 func atoi(str string)(ret int){
3510     ret, err := fmt.Sscanf(str, "%d", &ret)
3511     if err == nil {
3512         return ret
3513     }else{
3514         // should set errno
3515         return 0
3516     }
3517 }
3518 func getenv(name string)(string){
3519     val, got := os.LookupEnv(name)
3520     if got {
3521         return val
3522     }else{
3523         return "?"
3524     }
3525 }
3526 func strcpy(dst StrBuff, src string){
3527     var i int
3528     srcb := []byte(src)
3529     for i = 0; i < len(src) && srcb[i] != 0; i++ {
3530         dst[i] = srcb[i]
3531     }
3532     dst[i] = 0
3533 }
3534 func xstrcpy(dst StrBuff, src StrBuff){
3535     dst = src
3536 }
3537 func strcat(dst StrBuff, src StrBuff){
3538     dst = append(dst, src...)
3539 }
3540 func strdup(str StrBuff)(string){
3541     return string(str[0:strlen(str)])
3542 }
3543 func sstrlen(str string)(int){
3544     return len(str)
3545 }
3546 func strlen(str StrBuff)(int){
3547     var i int
3548     for i = 0; i < len(str) && str[i] != 0; i++ {
3549     }
3550     return i
3551 }
3552 func sizeof(data StrBuff)(int){
3553     return len(data)
3554 }
3555 func isatty(fd int)(ret int){
3556     return 1
3557 }
3558 }
3559 func fopen(file string, mode string)(fp*os.File){
3560     if mode == "r" {
3561         fp, err := os.Open(file)
3562         if( err != nil ){
3563             fmt.Printf("--E-- fopen(%s,%s)=(%v)\n", file, mode, err)
3564             return NULL_FP;
3565         }
3566         return fp;
3567     }else{
3568         fp, err := os.OpenFile(file, os.O_RDWR|os.O_CREATE|os.O_TRUNC, 0600)
3569         if( err != nil ){
3570             return NULL_FP;
3571         }
3572         return fp;
3573     }
3574 }
3575 func fclose(fp*os.File){
3576     fp.Close()
3577 }
3578 func fflush(fp *os.File)(int){
3579     return 0
3580 }
3581 func fgetc(fp*os.File)(int){
3582     var buf [1]byte
3583     _, err := fp.Read(buf[0:1])
3584     if( err != nil ){
3585         return EOF;
3586     }else{
3587         return int(buf[0])
3588     }
3589 }
3590 func sfgets(str*string, size int, fp*os.File)(int){
3591     buf := make(StrBuff, size)
3592     var ch int
3593     var i int
3594     for i = 0; i < len(buf)-1; i++ {
3595         ch = fgetc(fp)
3596         //fprintf(stderr, "--fgets %d/%d %X\n", i, len(buf), ch)
3597         if( ch == EOF ){
3598             break;
3599         }
3600         buf[i] = byte(ch);
3601         if( ch == '\n' ){
3602             break;
3603         }
3604     }
3605     buf[i] = 0
3606     //fprintf(stderr, "--fgets %d/%d (%s)\n", i, len(buf), buf[0:i])
3607     return i
3608 }
3609 func fgets(buf StrBuff, size int, fp*os.File)(int){
3610     var ch int
3611     var i int
3612     for i = 0; i < len(buf)-1; i++ {
3613         ch = fgetc(fp)
3614         //fprintf(stderr, "--fgets %d/%d %X\n", i, len(buf), ch)
3615         if( ch == EOF ){
3616             break;
3617         }
3618         buf[i] = byte(ch);
3619         if( ch == '\n' ){
3620             break;
3621         }
3622     }
3623     buf[i] = 0
3624     //fprintf(stderr, "--fgets %d/%d (%s)\n", i, len(buf), buf[0:i])

```

```

3625     return i
3626 }
3627 func fputc(ch int , fp*os.File)(int){
3628     var buf [1]byte
3629     buf[0] = byte(ch)
3630     fp.Write(buf[0:1])
3631     return 0
3632 }
3633 func fputs(buf StrBuff, fp*os.File)(int){
3634     fp.Write(buf)
3635     return 0
3636 }
3637 func xfputss(str string, fp*os.File)(int){
3638     return fputs([]byte(str),fp)
3639 }
3640 func sscanf(str StrBuff,fmts string, params ...interface{})(int){
3641     fmt.Sscanf(string(str[0:strlen(str)]),fmts,params...)
3642     return 0
3643 }
3644 func fprintf(fp*os.File,fmts string, params ...interface{})(int){
3645     fmt.Fprintf(fp,fmts,params...)
3646     return 0
3647 }
3648
3649 // <a name="IME">Command Line IME</a>
3650 //----- MyIME
3651 var MyIMEVER = "MyIME/0.0.2";
3652 type RomKana struct {
3653     dic string // dictionaly ID
3654     pat string // input pattern
3655     out string // output pattern
3656     hit int64 // count of hit and used
3657 }
3658 var dicents = 0
3659 var romkana [1024]RomKana
3660 var Romkan []RomKana
3661
3662 func isinDic(str string)(int){
3663     for i,v := range Romkan {
3664         if v.pat == str {
3665             return i
3666         }
3667     }
3668     return -1
3669 }
3670 const (
3671     DIC_COM_LOAD = "im"
3672     DIC_COM_DUMP = "g"
3673     DIC_COM_LIST = "ls"
3674     DIC_COM_ENA = "en"
3675     DIC_COM_DIS = "di"
3676 )
3677 func helpDic(argv []string){
3678     out := stderr
3679     cmd := ""
3680     if 0 < len(argv) { cmd = argv[0] }
3681     fprintf(out,"--- %v Usage\n",cmd)
3682     fprintf(out,"... Commands\n")
3683     fprintf(out,"... %v %%-3v [dicName] [dicURL] -- Import dictionary\n",cmd,DIC_COM_LOAD)
3684     fprintf(out,"... %v %%-3v [pattern] -- Search in dictionary\n",cmd,DIC_COM_DUMP)
3685     fprintf(out,"... %v %%-3v [dicName] -- List dictionaries\n",cmd,DIC_COM_LIST)
3686     fprintf(out,"... %v %%-3v [dicName] -- Disable dictionaries\n",cmd,DIC_COM_DIS)
3687     fprintf(out,"... %v %%-3v [dicName] -- Enable dictionaries\n",cmd,DIC_COM_ENA)
3688     fprintf(out,"... Keys ... %v\n","ESC can be used for '\\')")
3689     fprintf(out,"... \\c -- Reverse the case of the last character\n",)
3690     fprintf(out,"... \\i -- Replace input with translated text\n",)
3691     fprintf(out,"... \\j -- On/Off translation mode\n",)
3692     fprintf(out,"... \\l -- Force Lower Case\n",)
3693     fprintf(out,"... \\u -- Force Upper Case (software CapsLock)\n",)
3694     fprintf(out,"... \\v -- Show translation actions\n",)
3695     fprintf(out,"... \\x -- Replace the last input character with it Hexa-Decimal\n",)
3696 }
3697 func xDic(argv[]string){
3698     if len(argv) <= 1 {
3699         helpDic(argv)
3700         return
3701     }
3702     argv = argv[1:]
3703     var debug = false
3704     var info = false
3705     var silent = false
3706     var dump = false
3707     var builtin = false
3708     cmd := argv[0]
3709     argv = argv[1:]
3710     opt := ""
3711     arg := ""
3712
3713     if 0 < len(argv) {
3714         arg1 := argv[0]
3715         if arg1[0] == '-' {
3716             switch arg1 {
3717                 default: fmt.Printf("--Ed-- Unknown option(%v)\n",arg1)
3718                     return
3719                 case "-b": builtin = true
3720                 case "-d": debug = true
3721                 case "-s": silent = true
3722                 case "-v": info = true
3723             }
3724             opt = arg1
3725             argv = argv[1:]
3726         }
3727     }
3728
3729     dicName := ""
3730     dicURL := ""
3731     if 0 < len(argv) {
3732         arg = argv[0]
3733         dicName = arg
3734         argv = argv[1:]
3735     }
3736     if 0 < len(argv) {
3737         dicURL = argv[0]
3738         argv = argv[1:]
3739     }
3740     if false {
3741         fprintf(stderr,"--Dd-- com(%v) opt(%v) arg(%v)\n",cmd,opt,arg)
3742     }
3743     if cmd == DIC_COM_LOAD {
3744         //dicType := ""
3745         dicBody := ""
3746         if !builtin && dicName != "" && dicURL == "" {
3747             f,err := os.Open(dicName)
3748             if err == nil {
3749                 dicURL = dicName

```

```

3750     }else{
3751         f,err = os.Open(dicName+".html")
3752         if err == nil {
3753             dicURL = dicName+".html"
3754         }else{
3755             f,err = os.Open("gshdic-"+dicName+".html")
3756             if err == nil {
3757                 dicURL = "gshdic-"+dicName+".html"
3758             }
3759         }
3760     }
3761     if err == nil {
3762         var buf = make([]byte,128*1024)
3763         count,err := f.Read(buf)
3764         f.Close()
3765         if info {
3766             fprintf(stderr,"--Id-- ReadDic(%v,%v)\n",count,err)
3767         }
3768         dicBody = string(buf[0:count])
3769     }
3770 }
3771 if dicBody == "" {
3772     switch arg {
3773     default:
3774         dicName = "WorldDic"
3775         dicURL = WorldDic
3776         if info {
3777             fprintf(stderr,"--Id-- default dictionary \"%v\"\n",
3778                 dicName);
3779         }
3780     case "wnn":
3781         dicName = "WnnDic"
3782         dicURL = WnnDic
3783     case "sumomo":
3784         dicName = "SumomoDic"
3785         dicURL = SumomoDic
3786     case "sijimi":
3787         dicName = "SijimiDic"
3788         dicURL = SijimiDic
3789     case "jkl":
3790         dicName = "JKLJaDic"
3791         dicURL = JA_JKLDic
3792     }
3793     if debug {
3794         fprintf(stderr,"--Id-- %v URL=%v\n\n",dicName,dicURL);
3795     }
3796     dicv := strings.Split(dicURL,",")
3797     if debug {
3798         fprintf(stderr,"--Id-- %v encoded data...\n",dicName)
3799         fprintf(stderr,"Type: %v\n",dicv[0])
3800         fprintf(stderr,"Body: %v\n",dicv[1])
3801         fprintf(stderr,"\n")
3802     }
3803     body,_ := base64.StdEncoding.DecodeString(dicv[1])
3804     dicBody = string(body)
3805 }
3806 if info {
3807     fmt.Printf("--Id-- %v %v\n",dicName,dicURL)
3808     fmt.Printf("%s\n",dicBody)
3809 }
3810 if debug {
3811     fprintf(stderr,"--Id-- dicName %v text...\n",dicName)
3812     fprintf(stderr,"%v\n",string(dicBody))
3813 }
3814 envv := strings.Split(dicBody,"\n");
3815 if info {
3816     fprintf(stderr,"--Id-- %v scan...\n",dicName);
3817 }
3818 var added int = 0
3819 var dup int = 0
3820 for i,v := range envv {
3821     var pat string
3822     var out string
3823     fmt.Sscanf(v,"%s %s",&pat,&out)
3824     if len(pat) <= 0 {
3825     }else{
3826         if 0 <= isinDic(pat) {
3827             dup += 1
3828             continue
3829         }
3830         romkana[dicents] = RomKana{dicName,pat,out,0}
3831         dicents += 1
3832         added += 1
3833         Romkan = append(Romkan,RomKana{dicName,pat,out,0})
3834         if debug {
3835             fmt.Printf("[%3v]:[%2v]%-8v [%2v]%-8v\n",
3836                 i,len(pat),pat,len(out),out)
3837         }
3838     }
3839 }
3840 if !silent {
3841     url := dicURL
3842     if strBegins(url,"data:") {
3843         url = "builtin"
3844     }
3845     fprintf(stderr,"--Id-- %v scan... %v added, %v dup. / %v total (%v)\n",
3846         dicName,added,dup,len(Romkan),url);
3847 }
3848 // should sort by pattern length for complete match, for performance
3849 if debug {
3850     arg = "" // search pattern
3851     dump = true
3852 }
3853 }
3854 if cmd == DIC_COM_DUMP || dump {
3855     fprintf(stderr,"--Id-- %v dump... %v entries:\n",dicName,len(Romkan));
3856     var match = 0
3857     for i := 0; i < len(Romkan); i++ {
3858         dic := Romkan[i].dic
3859         pat := Romkan[i].pat
3860         out := Romkan[i].out
3861         if arg == "" || 0 <= strings.Index(pat,arg) || 0 <= strings.Index(out,arg) {
3862             fmt.Printf("\t\t%v\t%v [%2v]%-8v [%2v]%-8v\n",
3863                 i,dic,len(pat),pat,len(out),out)
3864             match += 1
3865         }
3866     }
3867     fprintf(stderr,"--Id-- %v matched %v / %v entries:\n",arg,match,len(Romkan));
3868 }
3869 }
3870 func loadDefaultDic(dic int){
3871     if( 0 < len(Romkan) ){
3872         return
3873     }
3874     //fprintf(stderr,"\r\n")

```

```

3875     xDic([]string{"dic",DIC_COM_LOAD});
3876
3877     var info = false
3878     if info {
3879         fprintf(stderr,"--Id-- Conguraturations!! WorldDic is now activated.\r\n")
3880         fprintf(stderr,"--Id-- enter \"dic\" command for help.\r\n")
3881     }
3882 }
3883 func readDic()(int){
3884     /*
3885     var rk *os.File;
3886     var dic = "MyIME-dic.txt";
3887     //rk = fopen("romkana.txt","r");
3888     //rk = fopen("JK-JA-morse-dic.txt","r");
3889     rk = fopen(dic,"r");
3890     if( rk == NULL_FP ){
3891         if( true ){
3892             fprintf(stderr,"--s-- Could not load %s\n",MyIMEVER,dic);
3893         }
3894         return -1;
3895     }
3896     if( true ){
3897         var di int;
3898         var line = make(StrBuff,1024);
3899         var pat string
3900         var out string
3901         for di = 0; di < 1024; di++ {
3902             if( fgets(line,sizeof(line),rk) == NULLSP ){
3903                 break;
3904             }
3905             fmt.Sscanf(string(line[0:strlen(line)]),"%s %s",&pat,&out);
3906             //sscanf(line,"%s %[\r\n]",&pat,&out);
3907             romkana[di].pat = pat;
3908             romkana[di].out = out;
3909             //fprintf(stderr,"--Dd- %-10s %s\n",pat,out)
3910         }
3911         dicents += di
3912         if( false ){
3913             fprintf(stderr,"--s-- loaded romkana.txt [%d]\n",MyIMEVER,di);
3914             for di = 0; di < dicents; di++ {
3915                 fprintf(stderr,
3916                     "%s %s\n",romkana[di].pat,romkana[di].out);
3917             }
3918         }
3919     }
3920     fclose(rk);
3921
3922     //romkana[dicents].pat = "//ddump"
3923     //romkana[dicents].pat = "//ddump" // dump the dic. and clean the command input
3924     /*
3925     return 0;
3926 }
3927 func matchlen(stri string, pati string)(int){
3928     if strBegins(stri,pati) {
3929         return len(pati)
3930     }else{
3931         return 0
3932     }
3933 }
3934 func convs(src string)(string){
3935     var si int;
3936     var sx = len(src);
3937     var di int;
3938     var mi int;
3939     var dstb []byte
3940
3941     for si = 0; si < sx; { // search max. match from the position
3942         if strBegins(src[si:], "%x/") {
3943             // %x/integer/ // s/a/b/
3944             ix := strings.Index(src[si+3:], "/")
3945             if 0 < ix {
3946                 var iv int = 0
3947                 //fmt.Sscanf(src[si+3:si+3+ix], "%d",&iv)
3948                 fmt.Sscanf(src[si+3:si+3+ix], "%v",&iv)
3949                 sval := fmt.Sprintf("%x",iv)
3950                 bval := []byte(sval)
3951                 dstb = append(dstb,bval...)
3952                 si = si+3+ix+1
3953                 continue
3954             }
3955         }
3956         if strBegins(src[si:], "%d/") {
3957             // %d/integer/ // s/a/b/
3958             ix := strings.Index(src[si+3:], "/")
3959             if 0 < ix {
3960                 var iv int = 0
3961                 fmt.Sscanf(src[si+3:si+3+ix], "%v",&iv)
3962                 sval := fmt.Sprintf("%d",iv)
3963                 bval := []byte(sval)
3964                 dstb = append(dstb,bval...)
3965                 si = si+3+ix+1
3966                 continue
3967             }
3968         }
3969         if strBegins(src[si:], "%t") {
3970             now := time.Now()
3971             if true {
3972                 date := now.Format(time.Stamp)
3973                 dstb = append(dstb,[]byte(date)...)
3974                 si = si+3
3975             }
3976             continue
3977         }
3978         var maxlen int = 0;
3979         var len int;
3980         mi = -1;
3981         for di = 0; di < dicents; di++ {
3982             len = matchlen(src[si:],romkana[di].pat);
3983             if( maxlen < len ){
3984                 maxlen = len;
3985                 mi = di;
3986             }
3987         }
3988         if( 0 < maxlen ){
3989             out := romkana[mi].out;
3990             dstb = append(dstb,[]byte(out)...);
3991             si += maxlen;
3992         }else{
3993             dstb = append(dstb,src[si])
3994             si += 1;
3995         }
3996     }
3997     return string(dstb)
3998 }
3999 func trans(src string)(int){

```



```

4000     dst := convs(src);
4001     xfprintf(dst,stderr);
4002     return 0;
4003 }
4004
4005 //----- LINEEDIT
4006 // "?" at the top of the line means searching history
4007
4008 // should be compatible with Telnet
4009 const (
4010     EV_MODE     = 255
4011     EV_IDLE    = 254
4012     EV_TIMEOUT = 253
4013
4014     GO_UP      = 252 // k
4015     GO_DOWN   = 251 // j
4016     GO_RIGHT  = 250 // l
4017     GO_LEFT   = 249 // h
4018     DEL_RIGHT = 248 // x
4019     GO_TOPL   = 'A'-0x40 // 0
4020     GO_ENDL   = 'E'-0x40 // $
4021
4022     GO_TOPW   = 239 // b
4023     GO_ENDW  = 238 // e
4024     GO_NEXTW = 237 // w
4025
4026     GO_FORWCH = 229 // f
4027     GO_PAIRCH = 228 // %
4028
4029     GO_DEL    = 219 // d
4030
4031     HI_SRCH_FW = 209 // /
4032     HI_SRCH_BK = 208 // ?
4033     HI_SRCH_RFW = 207 // n
4034     HI_SRCH_RBK = 206 // N
4035 )
4036
4037 // should return number of octets ready to be read immediately
4038 //fprintf(stderr,"\n--Select(%v %v)\n",err,r.Bits[0])
4039
4040
4041 var EventRecvFd = -1 // file descriptor
4042 var EventSendFd = -1
4043 const EventFdOffset = 1000000
4044 const NormalFdOffset = 100
4045
4046 func putEvent(event int, evarg int){
4047     if true {
4048         if EventRecvFd < 0 {
4049             var pv = []int{-1,-1}
4050             syscall.Pipe(pv)
4051             EventRecvFd = pv[0]
4052             EventSendFd = pv[1]
4053             //fmt.Printf("--De-- EventPipe created[%v,%v]\n",EventRecvFd,EventSendFd)
4054         }
4055     }else{
4056         if EventRecvFd < 0 {
4057             // the document differs from this spec
4058             // https://golang.org/src/syscall/syscall_unix.go?s=8096:8158#L1340
4059             sv,err := syscall.Socketpair(syscall.AF_UNIX,syscall.SOCK_STREAM,0)
4060             EventRecvFd = sv[0]
4061             EventSendFd = sv[1]
4062             if err != nil {
4063                 fmt.Printf("--De-- EventSock created[%v,%v]({%v})\n",
4064                     EventRecvFd,EventSendFd,err)
4065             }
4066         }
4067     }
4068     var buf = []byte{ byte(event)}
4069     n,err := syscall.Write(EventSendFd,buf)
4070     if err != nil {
4071         fmt.Printf("--De-- putEvent[%v]({%3v})({%v %v})\n",EventSendFd,event,n,err)
4072     }
4073 }
4074 func ungets(str string){
4075     for _,ch := range str {
4076         putEvent(int(ch),0)
4077     }
4078 }
4079 func (gsh*GshContext)xReplay(argv[]string){
4080     hix := 0
4081     tempo := 1.0
4082     xtempo := 1.0
4083     repeat := 1
4084
4085     for _,a := range argv { // tempo
4086         if strBegins(a,"x") {
4087             fmt.Sscanf(a[1:], "%f",&xtempo)
4088             tempo = 1 / xtempo
4089             //fprintf(stderr,"--Dr-- tempo=[%v]%v\n",a[2:],tempo);
4090         }else
4091         if strBegins(a,"r") { // repeat
4092             fmt.Sscanf(a[1:], "%v",&repeat)
4093         }else
4094         if strBegins(a,"!") {
4095             fmt.Sscanf(a[1:], "%d",&hix)
4096         }else{
4097             fmt.Sscanf(a,"%d",&hix)
4098         }
4099     }
4100     if hix == 0 || len(argv) <= 1 {
4101         hix = len(gsh.CommandHistory)-1
4102     }
4103     fmt.Printf("--Ir-- Replay(!%v x%v r%v)\n",hix,xtempo,repeat)
4104     //dumpEvents(hix)
4105     //gsh.xScanReplay(hix,false,repeat,tempo,argv)
4106     go gsh.xScanReplay(hix,true,repeat,tempo,argv)
4107 }
4108
4109 // <a href="https://golang.org/pkg/syscall/#FdSet">syscall.Select</a>
4110 // 2020-0827 gShell-0.2.3
4111 /*
4112 func FpollIn1(fp *os.File,usec int)(uintptr){
4113     nfd := 1
4114
4115     rdv := syscall.FdSet {}
4116     fd1 := fp.Fd()
4117     bank1 := fd1/32
4118     mask1 := int32(1 << fd1)
4119     rdv.Bits[bank1] = mask1
4120
4121     fd2 := -1
4122     bank2 := -1
4123     var mask2 int32 = 0
4124

```

```

4125 if 0 <= EventRecvFd {
4126     fd2 = EventRecvFd
4127     nfd = fd2 + 1
4128     bank2 = fd2/32
4129     mask2 = int32(1 << fd2)
4130     rdv.Bits[bank2] |= mask2
4131     //fmt.Printf("--De-- EventPoll mask added [%d][%v][%v]\n",fd2,bank2,mask2)
4132 }
4133
4134 tout := syscall.NsecToTimeval(int64(usec*1000))
4135 //n,err := syscall.Select(nfd,&rdv,nil,nil,&tout) // spec. mismatch
4136 err := syscall.Select(nfd,&rdv,nil,nil,&tout)
4137 if err != nil {
4138     //fmt.Printf("--De-- select() err(%v)\n",err)
4139 }
4140 if err == nil {
4141     if 0 <= fd2 && (rdv.Bits[bank2] & mask2) != 0 {
4142         if false {
4143             fmt.Printf("--De-- got Event\n")
4144         }
4145         return uintptr(EventFdOffset + fd2)
4146     }else
4147     if (rdv.Bits[bank1] & mask1) != 0 {
4148         return uintptr(NormalFdOffset + fd1)
4149     }else{
4150         return 1
4151     }
4152 }else{
4153     return 0
4154 }
4155 }
4156 */
4157 func fgetcTimeout1(fp *os.File,usec int)(int){
4158     READ1:
4159     //readyFd := FpollIn1(fp,usec)
4160     readyFd := CFpollIn1(fp,usec)
4161     if readyFd < 100 {
4162         return EV_TIMEOUT
4163     }
4164
4165     var buf [1]byte
4166
4167     if EventFdOffset <= readyFd {
4168         fd := int(readyFd-EventFdOffset)
4169         _,err := syscall.Read(fd,buf[0:1])
4170         if( err != nil ){
4171             return EOF;
4172         }else{
4173             if buf[0] == EV_MODE {
4174                 recvEvent(fd)
4175                 goto READ1
4176             }
4177             return int(buf[0])
4178         }
4179     }
4180
4181     _,err := fp.Read(buf[0:1])
4182     if( err != nil ){
4183         return EOF;
4184     }else{
4185         return int(buf[0])
4186     }
4187 }
4188
4189 func visibleChar(ch int)(string){
4190     switch {
4191     case '!' <= ch && ch <= '-':
4192         return string(ch)
4193     }
4194     switch ch {
4195     case ' ': return "\\s"
4196     case '\n': return "\\n"
4197     case '\r': return "\\r"
4198     case '\t': return "\\t"
4199     }
4200     switch ch {
4201     case 0x00: return "NUL"
4202     case 0x07: return "BEL"
4203     case 0x08: return "BS"
4204     case 0x0E: return "SO"
4205     case 0x0F: return "SI"
4206     case 0x1B: return "ESC"
4207     case 0x7F: return "DEL"
4208     }
4209     switch ch {
4210     case EV_IDLE: return fmt.Sprintf("IDLE")
4211     case EV_MODE: return fmt.Sprintf("MODE")
4212     }
4213     return fmt.Sprintf("%X",ch)
4214 }
4215 func recvEvent(fd int){
4216     var buf = make([]byte,1)
4217     _,_ = syscall.Read(fd,buf[0:1])
4218     if( buf[0] != 0 ){
4219         romkanmode = true
4220     }else{
4221         romkanmode = false
4222     }
4223 }
4224 func (gsh*GshContext)xScanReplay(hix int,replay bool,repeat int,tempo float64,argv[string]){
4225     var Start time.Time
4226     var events = []Event{}
4227     for _,e := range Events {
4228         if hix == 0 || e.CmdIndex == hix {
4229             events = append(events,e)
4230         }
4231     }
4232     elen := len(events)
4233     if 0 < elen {
4234         if events[elen-1].event == EV_IDLE {
4235             events = events[0:elen-1]
4236         }
4237     }
4238     for r := 0; r < repeat; r++ {
4239         for i,e := range events {
4240             nano := e.when.Nanosecond()
4241             micro := nano / 1000
4242             if Start.Second() == 0 {
4243                 Start = time.Now()
4244             }
4245             diff := time.Now().Sub(Start)
4246             if replay {
4247                 if e.event != EV_IDLE {
4248                     putEvent(e.event,0)
4249                 if e.event == EV_MODE { // event with arg

```

```

4250         putEvent(int(e.evarg),0)
4251     }
4252 }
4253 }else{
4254     fmt.Printf("%7.3fms %#-3v !%-3v [%v.%06d] %3v %02X %-4v %10.3fms\n",
4255         float64(diff)/1000000.0,
4256         i,
4257         e.CmdIndex,
4258         e.when.Format(time.Stamp),micro,
4259         e.event,e.event,visibleChar(e.event),
4260         float64(e.evarg)/1000000.0)
4261 }
4262 if e.event == EV_IDLE {
4263     d := time.Duration(float64(time.Duration(e.evarg)) * tempo)
4264     //nsleep(time.Duration(e.evarg))
4265     nsleep(d)
4266 }
4267 }
4268 }
4269 }
4270 func dumpEvents(arg[]string){
4271     hix := 0
4272     if 1 < len(arg) {
4273         fmt.Sscanf(arg[1],"%d",&hix)
4274     }
4275     for i,e := range Events {
4276         nano := e.when.Nanosecond()
4277         micro := nano / 1000
4278         //if e.event != EV_TIMEOUT {
4279         if hix == 0 || e.CmdIndex == hix {
4280             fmt.Printf("%#-3v !%-3v [%v.%06d] %3v %02X %-4v %10.3fms\n",i,
4281                 e.CmdIndex,
4282                 e.when.Format(time.Stamp),micro,
4283                 e.event,e.event,visibleChar(e.event),float64(e.evarg)/1000000.0)
4284         }
4285         //}
4286     }
4287 }
4288 func fgetcTimeout(fp *os.File,usec int)(int){
4289     ch := fgetcTimeout1(fp,usec)
4290     if ch != EV_TIMEOUT {
4291         now := Time.Now()
4292         if 0 < len(Events) {
4293             last := Events[len(Events)-1]
4294             dura := int64(now.Sub(last.when))
4295             Events = append(Events,Event{last.when,EV_IDLE,dura,last.CmdIndex})
4296         }
4297         Events = append(Events,Event{time.Now(),ch,0,CmdIndex})
4298     }
4299     return ch
4300 }
4301 }
4302 var TtyMaxCol = 72 // to be obtained by ioctl?
4303 var EscTimeout = (100*1000)
4304 var (
4305     MODE_VicMode    bool    // vi compatible command mode
4306     MODE_ShowMode   bool    //
4307     romkanmode      bool    // shown translation mode, the mode to be retained
4308     MODE_Recursive  bool    // recursive translation
4309     MODE_CapsLock   bool    // software CapsLock
4310     MODE_LowerLock  bool    // force lower-case character lock
4311     MODE_Vinsert    int     // visible insert mode, should be like "I" icon in X Window
4312     MODE_ViTrace    bool    // output newline before translation
4313 )
4314 type IInput struct {
4315     lno      int
4316     lastlno int
4317     pch      []int // input queue
4318     prompt   string
4319     line     string
4320     right    string
4321     inJmode  bool
4322     pinJmode bool
4323     waitingMeta string // waiting meta character
4324     LastCmd   string
4325 }
4326 func (iin*IInput)Getc(timeoutUs int)(int){
4327     ch1 := EOF
4328     ch2 := EOF
4329     ch3 := EOF
4330     if( 0 < len(iin.pch) ){ // deQ
4331         ch1 = iin.pch[0]
4332         iin.pch = iin.pch[1:]
4333     }else{
4334         ch1 = fgetcTimeout(stdin,timeoutUs);
4335     }
4336     if( ch1 == 033 ){ // escape sequence
4337         ch2 = fgetcTimeout(stdin,EscTimeout);
4338         if( ch2 == EV_TIMEOUT ){
4339             }else{
4340                 ch3 = fgetcTimeout(stdin,EscTimeout);
4341                 if( ch3 == EV_TIMEOUT ){
4342                     iin.pch = append(iin.pch,ch2) // enQ
4343                 }else{
4344                     switch( ch2 ){
4345                         default:
4346                             iin.pch = append(iin.pch,ch2) // enQ
4347                             iin.pch = append(iin.pch,ch3) // enQ
4348                         case '[':
4349                             switch( ch3 ){
4350                                 case 'A': ch1 = GO_UP; // ^
4351                                 case 'B': ch1 = GO_DOWN; // v
4352                                 case 'C': ch1 = GO_RIGHT; // >
4353                                 case 'D': ch1 = GO_LEFT; // <
4354                                 case '3':
4355                                     ch4 := fgetcTimeout(stdin,EscTimeout);
4356                                     if( ch4 == '-' ){
4357                                         //fprintf(stderr,"x[%02X %02X %02X %02X]\n",ch1,ch2,ch3,ch4);
4358                                         ch1 = DEL_RIGHT
4359                                     }
4360                                 }
4361                         case '\\':
4362                             //ch4 := fgetcTimeout(stdin,EscTimeout);
4363                             //fprintf(stderr,"y[%02X %02X %02X %02X]\n",ch1,ch2,ch3,ch4);
4364                             switch( ch3 ){
4365                                 case '-': ch1 = DEL_RIGHT
4366                             }
4367                     }
4368                 }
4369             }
4370         }
4371     }
4372     return ch1
4373 }
4374 func (inn*IInput)clearline(){
4375     var i int

```

```

4375     fprintf(stderr, "\r");
4376     // should be ANSI ESC sequence
4377     for i = 0; i < TtyMaxCol; i++ { // to the max. position in this input action
4378         fputc(' ', os.Stderr);
4379     }
4380     fprintf(stderr, "\r");
4381 }
4382 func (iin*IInput)Redraw(){
4383     redraw(iin, iin.lno, iin.line, iin.right)
4384 }
4385 func redraw(iin *IInput, lno int, line string, right string){
4386     inMeta := false
4387     showMode := ""
4388     showMeta := "" // visible Meta mode on the cursor position
4389     showLino := fmt.Sprintf("%d", lno)
4390     insertMark := "" // in visible insert mode
4391
4392     if MODE_VicMode {
4393     }else
4394     if 0 < len(iin.right) {
4395         insertMark = " "
4396     }
4397
4398     if( 0 < len(iin.waitingMeta) ){
4399         inMeta = true
4400         if iin.waitingMeta[0] != 033 {
4401             showMeta = iin.waitingMeta
4402         }
4403     }
4404     if( romkanmode ){
4405         //romkanmark = " *";
4406     }else{
4407         //romkanmark = "";
4408     }
4409     if MODE_ShowMode {
4410         romkan := ""
4411         inmeta := ""
4412         inveri := ""
4413         if MODE_CapsLock {
4414             inmeta = "A"
4415         }
4416         if MODE_LowerLock {
4417             inmeta = "a"
4418         }
4419         if MODE_ViTrace {
4420             inveri = "v"
4421         }
4422         if MODE_VicMode {
4423             inveri = ":"
4424         }
4425         if romkanmode {
4426             romkan = "\343\201\202"
4427             if MODE_CapsLock {
4428                 inmeta = "R"
4429             }else{
4430                 inmeta = "r"
4431             }
4432         }
4433         if inMeta {
4434             inmeta = "\\ "
4435         }
4436         showMode = "["+romkan+inmeta+inveri+"]";
4437     }
4438     Pre := "\r" + showMode + showLino
4439     Output := ""
4440     Left := ""
4441     Right := ""
4442     if romkanmode {
4443         Left = convs(line)
4444         Right = InsertMark+convs(right)
4445     }else{
4446         Left = line
4447         Right = InsertMark+right
4448     }
4449     Output = Pre+Left
4450     if MODE_ViTrace {
4451         Output += iin.LastCmd
4452     }
4453     Output += showMeta+Right
4454     for len(Output) < TtyMaxCol { // to the max. position that may be dirty
4455         Output += " "
4456         // should be ANSI ESC sequence
4457         // not necessary just after newline
4458     }
4459     Output += Pre+Left+showMeta // to set the cursor to the current input position
4460     fprintf(stderr, "%s", Output)
4461
4462     if MODE_ViTrace {
4463         if 0 < len(iin.LastCmd) {
4464             iin.LastCmd = ""
4465             fprintf(stderr, "\r\n")
4466         }
4467     }
4468 }
4469 // <a href="https://golang.org/pkg/unicode/utf8/">utf8</a>
4470 func delHeadChar(str string)(rline string, head string){
4471     clen := utf8.DecodeRune([]byte(str))
4472     head = string(str[0:clen])
4473     return str[clen:], head
4474 }
4475 func delTailChar(str string)(rline string, last string){
4476     var i = 0
4477     var clen = 0
4478     for {
4479         _, siz := utf8.DecodeRune([]byte(str)[i:])
4480         if siz <= 0 { break }
4481         clen = siz
4482         i += siz
4483     }
4484     last = str[len(str)-clen:]
4485     return str[0:len(str)-clen], last
4486 }
4487
4488 // 3> for output and history
4489 // 4> for keylog?
4490 // <a name="getline">Command Line Editor</a>
4491 func xgetline(lno int, prevline string, gsh*GshContext)(string){
4492     var iin IInput
4493     iin.lastlno = lno
4494     iin.lno = lno
4495
4496     CmdIndex = len(gsh.CommandHistory)
4497     if( isatty(0) == 0 ){
4498         if( sfgets(&iin.line, LINESIZE, stdin) == NULL ){
4499             iin.line = "exit\n";

```

```

4500     }else{
4501     }
4502     return iin.line
4503 }
4504 if( true ){
4505     //var pts string;
4506     //pts = ptsname(0);
4507     //pts = ttyname(0);
4508     //fprintf(stderr,"--pts[0] = %s\n",pts?pts:"?");
4509 }
4510 if( false ){
4511     fprintf(stderr,"! ");
4512     fflush(stderr);
4513     sfgets(&iin.line,LINESIZE,stdin);
4514     return iin.line
4515 }
4516 system("/bin/stty -echo -icanon");
4517 xline := iin.xgetline(prevline,gsh)
4518 system("/bin/stty echo sane");
4519 return xline
4520 }
4521 func (iin*IInput)Translate(cmdch int){
4522     romkanmode = !romkanmode;
4523     if MODE_ViTrace {
4524         fprintf(stderr,"%v\r\n",string(cmdch));
4525     }else
4526     if( cmdch == 'J' ){
4527         fprintf(stderr,"J\r\n");
4528         iin.inJmode = true
4529     }
4530     iin.Redraw();
4531     loadDefaultDic(cmdch);
4532     iin.Redraw();
4533 }
4534 func (iin*IInput)Replace(cmdch int){
4535     iin.LastCmd = fmt.Sprintf("\\%v",string(cmdch))
4536     iin.Redraw();
4537     loadDefaultDic(cmdch);
4538     dst := convs(iin.line+iin.right);
4539     iin.line = dst
4540     iin.right = ""
4541     if( cmdch == 'I' ){
4542         fprintf(stderr,"I\r\n");
4543         iin.inJmode = true
4544     }
4545     iin.Redraw();
4546 }
4547 // aa 12 alal
4548 func isAlpha(ch rune)(bool){
4549     if 'a' <= ch && ch <= 'z' || 'A' <= ch && ch <= 'Z' {
4550         return true
4551     }
4552     return false
4553 }
4554 func isAlnum(ch rune)(bool){
4555     if 'a' <= ch && ch <= 'z' || 'A' <= ch && ch <= 'Z' {
4556         return true
4557     }
4558     if '0' <= ch && ch <= '9' {
4559         return true
4560     }
4561     return false
4562 }
4563
4564 // 0.2.8 2020-0901 created
4565 // <a href="https://golang.org/pkg/unicode/utf8/#DecodeRuneInString">DecodeRuneInString</a>
4566 func (iin*IInput)GotoTOPW(){
4567     str := iin.line
4568     i := len(str)
4569     if i <= 0 {
4570         return
4571     }
4572     //i0 := i
4573     i -= 1
4574     lastSize := 0
4575     var lastRune rune
4576     var found = -1
4577     for 0 < i { // skip preamble spaces
4578         lastRune, lastSize = utf8.DecodeRuneInString(str[i:])
4579         if !isAlnum(lastRune) { // character, type, or string to be searched
4580             i -= lastSize
4581             continue
4582         }
4583         break
4584     }
4585     for 0 < i {
4586         lastRune, lastSize = utf8.DecodeRuneInString(str[i:])
4587         if lastSize <= 0 { continue } // not the character top
4588         if !isAlnum(lastRune) { // character, type, or string to be searched
4589             found = i
4590             break
4591         }
4592         i -= lastSize
4593     }
4594     if found < 0 && i == 0 {
4595         found = 0
4596     }
4597     if 0 <= found {
4598         if isAlnum(lastRune) { // or non-kana character
4599             }else{ // when positioning to the top o the word
4600                 i += lastSize
4601             }
4602         iin.right = str[i:] + iin.right
4603         if 0 < i {
4604             iin.line = str[0:i]
4605         }else{
4606             iin.line = ""
4607         }
4608     }
4609     //fmt.Printf("\n(%d,%d,%d)[%s][%s]\n",i0,i,found,iin.line,iin.right)
4610     //fmt.Printf("") // set debug messae at the end of line
4611 }
4612 // 0.2.8 2020-0901 created
4613 func (iin*IInput)GotoENDW(){
4614     str := iin.right
4615     if len(str) <= 0 {
4616         return
4617     }
4618     lastSize := 0
4619     var lastRune rune
4620     var lastW = 0
4621     i := 0
4622     inWord := false
4623
4624     lastRune, lastSize = utf8.DecodeRuneInString(str[0:])

```

```

4625 if isAlnum(lastRune) {
4626     r,z := utf8.DecodeRuneInString(str[lastSize:])
4627     if 0 < z && isAlnum(r) {
4628         inWord = true
4629     }
4630 }
4631 for i < len(str) {
4632     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4633     if lastSize <= 0 { break } // broken data?
4634     if !isAlnum(lastRune) { // character, type, or string to be searched
4635         break
4636     }
4637     lastW = i // the last alnum if in alnum word
4638     i += lastSize
4639 }
4640 if inWord {
4641     goto DISP
4642 }
4643 for i < len(str) {
4644     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4645     if lastSize <= 0 { break } // broken data?
4646     if isAlnum(lastRune) { // character, type, or string to be searched
4647         break
4648     }
4649     i += lastSize
4650 }
4651 for i < len(str) {
4652     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4653     if lastSize <= 0 { break } // broken data?
4654     if !isAlnum(lastRune) { // character, type, or string to be searched
4655         break
4656     }
4657     lastW = i
4658     i += lastSize
4659 }
4660 DISP:
4661 if 0 < lastW {
4662     iin.line = iin.line + str[0:lastW]
4663     iin.right = str[lastW:]
4664 }
4665 //fmt.Printf("\n(%d)[%s][%s]\n",i,iin.line,iin.right)
4666 //fmt.Printf("") // set debug messae at the end of line
4667 }
4668 // 0.2.8 2020-0901 created
4669 func (iin*Input)GotoNEXTW(){
4670     str := iin.right
4671     if len(str) <= 0 {
4672         return
4673     }
4674     lastSize := 0
4675     var lastRune rune
4676     var found = -1
4677     i := 1
4678     for i < len(str) {
4679         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4680         if lastSize <= 0 { break } // broken data?
4681         if !isAlnum(lastRune) { // character, type, or string to be searched
4682             found = i
4683             break
4684         }
4685         i += lastSize
4686     }
4687     if 0 < found {
4688         if isAlnum(lastRune) { // or non-kana character
4689             }else{ // when positioning to the top o the word
4690                 found += lastSize
4691             }
4692         iin.line = iin.line + str[0:found]
4693         if 0 < found {
4694             iin.right = str[found:]
4695         }else{
4696             iin.right = ""
4697         }
4698     }
4699     //fmt.Printf("\n(%d)[%s][%s]\n",i,iin.line,iin.right)
4700     //fmt.Printf("") // set debug messae at the end of line
4701 }
4702 // 0.2.8 2020-0902 created
4703 func (iin*Input)GotoPAIRCH(){
4704     str := iin.right
4705     if len(str) <= 0 {
4706         return
4707     }
4708     lastRune,lastSize := utf8.DecodeRuneInString(str[0:])
4709     if lastSize <= 0 {
4710         return
4711     }
4712     forw := false
4713     back := false
4714     pair := ""
4715     switch string(lastRune){
4716     case "{": pair = "}"; forw = true
4717     case "}": pair = "{"; back = true
4718     case "(": pair = ")"; forw = true
4719     case ")": pair = "("; back = true
4720     case "[": pair = "]"; forw = true
4721     case "]": pair = "["; back = true
4722     case "<": pair = ">"; forw = true
4723     case ">": pair = "<"; back = true
4724     case "\'": pair = "\""; // context depednet, can be f" or back-double quote
4725     case "'": pair = "'"; // context depednet, can be f' or back-quote
4726     // case Japanese Kakkos
4727     }
4728     if forw {
4729         iin.SearchForward(pair)
4730     }
4731     if back {
4732         iin.SearchBackward(pair)
4733     }
4734 }
4735 // 0.2.8 2020-0902 created
4736 func (iin*Input)SearchForward(pat string)(bool){
4737     right := iin.right
4738     found := -1
4739     i := 0
4740     if strBegins(right,pat) {
4741         r,z := utf8.DecodeRuneInString(right[i:])
4742         if 0 < z {
4743             i += z
4744         }
4745     }
4746     for i < len(right) {
4747         if strBegins(right[i:],pat) {
4748             found = i
4749             break

```

```

4750     }
4751     _,z := utf8.DecodeRuneInString(right[i:])
4752     if z <= 0 { break }
4753     i += z
4754 }
4755 if 0 <= found {
4756     iin.line = iin.line + right[0:found]
4757     iin.right = iin.right[found:]
4758     return true
4759 }else{
4760     return false
4761 }
4762 }
4763 // 0.2.8 2020-0902 created
4764 func (iin*IInput)SearchBackward(pat string)(bool){
4765     line := iin.line
4766     found := -1
4767     i := len(line)-1
4768     for i = i; 0 <= i; i-- {
4769         _,z := utf8.DecodeRuneInString(line[i:])
4770         if z <= 0 {
4771             continue
4772         }
4773         //fprintf(stderr,"-- %v %v\n",pat,line[i:])
4774         if strBegins(line[i:],pat) {
4775             found = i
4776             break
4777         }
4778     }
4779     //fprintf(stderr,"--%d\n",found)
4780     if 0 <= found {
4781         iin.right = line[found:] + iin.right
4782         iin.line = line[0:found]
4783         return true
4784     }else{
4785         return false
4786     }
4787 }
4788 // 0.2.8 2020-0902 created
4789 // search from top, end, or current position
4790 func (gsh*GshContext)SearchHistory(pat string, forw bool)(bool,string){
4791     if forw {
4792         for _,v := range gsh.CommandHistory {
4793             if 0 <= strings.Index(v.CmdLine,pat) {
4794                 //fprintf(stderr,"\n--De-- found !%v [%v]%v\n",i,pat,v.CmdLine)
4795                 return true,v.CmdLine
4796             }
4797         }
4798     }else{
4799         hlen := len(gsh.CommandHistory)
4800         for i := hlen-1; 0 < i; i-- {
4801             v := gsh.CommandHistory[i]
4802             if 0 <= strings.Index(v.CmdLine,pat) {
4803                 //fprintf(stderr,"\n--De-- found !%v [%v]%v\n",i,pat,v.CmdLine)
4804                 return true,v.CmdLine
4805             }
4806         }
4807     }
4808     //fprintf(stderr,"\n--De-- not-found(%v)\n",pat)
4809     return false,"(Not Found in History)"
4810 }
4811 // 0.2.8 2020-0902 created
4812 func (iin*IInput)GotoFORWSTR(pat string,gsh*GshContext){
4813     found := false
4814     if 0 < len(iin.right) {
4815         found = iin.SearchForward(pat)
4816     }
4817     if !found {
4818         found,line := gsh.SearchHistory(pat,true)
4819         if found {
4820             iin.line = line
4821             iin.right = ""
4822         }
4823     }
4824 }
4825 func (iin*IInput)GotoBACKSTR(pat string, gsh*GshContext){
4826     found := false
4827     if 0 < len(iin.line) {
4828         found = iin.SearchBackward(pat)
4829     }
4830     if !found {
4831         found,line := gsh.SearchHistory(pat,false)
4832         if found {
4833             iin.line = line
4834             iin.right = ""
4835         }
4836     }
4837 }
4838 func (iin*IInput)getString1(prompt string)(string){ // should be editable
4839     iin.clearline();
4840     fprintf(stderr,"\r%v",prompt)
4841     str := ""
4842     for {
4843         ch := iin.Getc(10*1000*1000)
4844         if ch == '\n' || ch == '\r' {
4845             break
4846         }
4847         sch := string(ch)
4848         str += sch
4849         fprintf(stderr,"%s",sch)
4850     }
4851     return str
4852 }
4853 }
4854 // search pattern must be an array and selectable with ^N/^P
4855 var SearchPat = ""
4856 var SearchForw = true
4857 }
4858 func (iin*IInput)xgetline1(prevline string, gsh*GshContext)(string){
4859     var ch int;
4860 }
4861     MODE_ShowMode = false
4862     MODE_VicMode = false
4863     iin.Redraw();
4864     first := true
4865 }
4866     for cix := 0; ; cix++ {
4867         iin.pinJmode = iin.inJmode
4868         iin.inJmode = false
4869     }
4870     ch = iin.Getc(1000*1000)
4871 }
4872     if ch != EV_TIMEOUT && first {
4873         first = false
4874         mode := 0

```

```

4875     if romkanmode {
4876         mode = 1
4877     }
4878     now := time.Now()
4879     Events = append(Events, Event{now, EV_MODE, int64(mode), CmdIndex})
4880 }
4881 if ch == 033 {
4882     MODE_ShowMode = true
4883     MODE_VicMode = !MODE_VicMode
4884     iin.Redraw();
4885     continue
4886 }
4887 if MODE_VicMode {
4888     switch ch {
4889     case '0': ch = GO_TOPL
4890     case '$': ch = GO_ENDL
4891     case 'b': ch = GO_TOPW
4892     case 'e': ch = GO_ENDW
4893     case 'w': ch = GO_NEXTW
4894     case '$': ch = GO_PAIRCH
4895
4896     case 'j': ch = GO_DOWN
4897     case 'k': ch = GO_UP
4898     case 'h': ch = GO_LEFT
4899     case 'l': ch = GO_RIGHT
4900     case 'x': ch = DEL_RIGHT
4901     case 'a': MODE_VicMode = !MODE_VicMode
4902     case 'i': MODE_VicMode = !MODE_VicMode
4903     case 'i': MODE_VicMode = !MODE_VicMode
4904         iin.Redraw();
4905         continue
4906     case '-':
4907         right, head := delHeadChar(iin.right)
4908         if len([]byte(head)) == 1 {
4909             ch = int(head[0])
4910             if ('a' <= ch && ch <= 'z' ){
4911                 ch = ch + 'A'-'a'
4912             }else
4913             if ('A' <= ch && ch <= 'Z' ){
4914                 ch = ch + 'a'-'A'
4915             }
4916             iin.right = string(ch) + right
4917         }
4918         iin.Redraw();
4919         continue
4920     case 'f': // GO_FORWCH
4921         iin.Redraw();
4922         ch = iin.Getc(3*1000*1000)
4923         if ch == EV_TIMEOUT {
4924             iin.Redraw();
4925             continue
4926         }
4927         SearchPat = string(ch)
4928         SearchForw = true
4929         iin.GotoFORWSTR(SearchPat, gsh)
4930         iin.Redraw();
4931         continue
4932     case '/':
4933         SearchPat = iin.getstring1("/") // should be editable
4934         SearchForw = true
4935         iin.GotoFORWSTR(SearchPat, gsh)
4936         iin.Redraw();
4937         continue
4938     case '?':
4939         SearchPat = iin.getstring1("?") // should be editable
4940         SearchForw = false
4941         iin.GotoBACKSTR(SearchPat, gsh)
4942         iin.Redraw();
4943         continue
4944     case 'n':
4945         if SearchForw {
4946             iin.GotoFORWSTR(SearchPat, gsh)
4947         }else{
4948             iin.GotoBACKSTR(SearchPat, gsh)
4949         }
4950         iin.Redraw();
4951         continue
4952     case 'N':
4953         if !SearchForw {
4954             iin.GotoFORWSTR(SearchPat, gsh)
4955         }else{
4956             iin.GotoBACKSTR(SearchPat, gsh)
4957         }
4958         iin.Redraw();
4959         continue
4960     }
4961 }
4962 switch ch {
4963 case GO_TOPW:
4964     iin.GotoTOPW()
4965     iin.Redraw();
4966     continue
4967 case GO_ENDW:
4968     iin.GotoENDW()
4969     iin.Redraw();
4970     continue
4971 case GO_NEXTW:
4972     // To next space then
4973     iin.GotoNEXTW()
4974     iin.Redraw();
4975     continue
4976 case GO_PAIRCH:
4977     iin.GotoPAIRCH()
4978     iin.Redraw();
4979     continue
4980 }
4981
4982 //fprintf(stderr, "A[%02X]\n", ch);
4983 if( ch == '\\ ' || ch == 033 ){
4984     MODE_ShowMode = true
4985     metach := ch
4986     iin.waitingMeta = string(ch)
4987     iin.Redraw();
4988     // set cursor //fprintf(stderr, "???\b\b\b")
4989     ch = fgetcTimeout(stdin, 2000*1000)
4990     // reset cursor
4991     iin.waitingMeta = ""
4992
4993     cmdch := ch
4994     if( ch == EV_TIMEOUT ){
4995         if metach == 033 {
4996             continue
4997         }
4998         ch = metach
4999     }else

```



```

5000      /*
5001      if( ch == 'm' || ch == 'M' ){
5002          mch := fgetcTimeout(stdin,1000*1000)
5003          if mch == 'r' {
5004              romkanmode = true
5005          }else{
5006              romkanmode = false
5007          }
5008          continue
5009      }else
5010      */
5011      if( ch == 'k' || ch == 'K' ){
5012          MODE_Recursive = IMODE_Recursive
5013          iin.Translate(cmdch);
5014          continue
5015      }else
5016      if( ch == 'j' || ch == 'J' ){
5017          iin.Translate(cmdch);
5018          continue
5019      }else
5020      if( ch == 'i' || ch == 'I' ){
5021          iin.Replace(cmdch);
5022          continue
5023      }else
5024      if( ch == 'l' || ch == 'L' ){
5025          MODE_LowerLock = IMODE_LowerLock
5026          MODE_CapsLock = false
5027          if MODE_ViTrace {
5028              fprintf(stderr,"%v\r\n",string(cmdch));
5029          }
5030          iin.Redraw();
5031          continue
5032      }else
5033      if( ch == 'u' || ch == 'U' ){
5034          MODE_CapsLock = IMODE_CapsLock
5035          MODE_LowerLock = false
5036          if MODE_ViTrace {
5037              fprintf(stderr,"%v\r\n",string(cmdch));
5038          }
5039          iin.Redraw();
5040          continue
5041      }else
5042      if( ch == 'v' || ch == 'V' ){
5043          MODE_ViTrace = IMODE_ViTrace
5044          if MODE_ViTrace {
5045              fprintf(stderr,"%v\r\n",string(cmdch));
5046          }
5047          iin.Redraw();
5048          continue
5049      }else
5050      if( ch == 'c' || ch == 'C' ){
5051          if 0 < len(iin.line) {
5052              xline,tail := delTailChar(iin.line)
5053              if len([]byte(tail)) == 1 {
5054                  ch = int(tail[0])
5055                  if( 'a' <= ch && ch <= 'z' ){
5056                      ch = ch + 'A'-'a'
5057                  }else
5058                  if( 'A' <= ch && ch <= 'Z' ){
5059                      ch = ch + 'a'-'A'
5060                  }
5061                  iin.line = xline + string(ch)
5062              }
5063          }
5064          if MODE_ViTrace {
5065              fprintf(stderr,"%v\r\n",string(cmdch));
5066          }
5067          iin.Redraw();
5068          continue
5069      }else{
5070          iin.pch = append(iin.pch,ch) // push
5071          ch = '\\'
5072      }
5073  }
5074  switch( ch ){
5075      case 'P'-0x40: ch = GO_UP
5076      case 'N'-0x40: ch = GO_DOWN
5077      case 'B'-0x40: ch = GO_LEFT
5078      case 'F'-0x40: ch = GO_RIGHT
5079  }
5080  //fprintf(stderr,"B[802X]\n",ch);
5081  switch( ch ){
5082      case 0:
5083          continue;
5084
5085      case '\t':
5086          iin.Replace('j');
5087          continue
5088      case 'X'-0x40:
5089          iin.Replace('j');
5090          continue
5091
5092      case EV_TIMEOUT:
5093          iin.Redraw();
5094          if iin.pinJmode {
5095              fprintf(stderr,"\\J\r\n")
5096              iin.inJmode = true
5097          }
5098          continue
5099      case GO_UP:
5100          if iin.lno == 1 {
5101              continue
5102          }
5103          cmd,ok := gsh.cmdStringInHistory(iin.lno-1)
5104          if ok {
5105              iin.line = cmd
5106              iin.right = ""
5107              iin.lno = iin.lno - 1
5108          }
5109          iin.Redraw();
5110          continue
5111      case GO_DOWN:
5112          cmd,ok := gsh.cmdStringInHistory(iin.lno+1)
5113          if ok {
5114              iin.line = cmd
5115              iin.right = ""
5116              iin.lno = iin.lno + 1
5117          }else{
5118              iin.line = ""
5119              iin.right = ""
5120              if iin.lno == iin.lastlno-1 {
5121                  iin.lno = iin.lno + 1
5122              }
5123          }
5124          iin.Redraw();

```

```

5125         continue
5126     case GO_LEFT:
5127         if 0 < len(iin.line) {
5128             xline,tail := delTailChar(iin.line)
5129             iin.line = xline
5130             iin.right = tail + iin.right
5131         }
5132         iin.Redraw();
5133         continue;
5134     case GO_RIGHT:
5135         if( 0 < len(iin.right) && iin.right[0] != 0 ){
5136             xright,head := delHeadChar(iin.right)
5137             iin.right = xright
5138             iin.line += head
5139         }
5140         iin.Redraw();
5141         continue;
5142     case EOF:
5143         goto EXIT;
5144     case 'R'-0x40: // replace
5145         dst := convs(iin.line+iin.right);
5146         iin.line = dst
5147         iin.right = ""
5148         iin.Redraw();
5149         continue;
5150     case 'T'-0x40: // just show the result
5151         readDic();
5152         romkanmode = !romkanmode;
5153         iin.Redraw();
5154         continue;
5155     case 'L'-0x40:
5156         iin.Redraw();
5157         continue
5158     case 'K'-0x40:
5159         iin.right = ""
5160         iin.Redraw();
5161         continue
5162     case 'E'-0x40:
5163         iin.line += iin.right
5164         iin.right = ""
5165         iin.Redraw();
5166         continue
5167     case 'A'-0x40:
5168         iin.right = iin.line + iin.right
5169         iin.line = ""
5170         iin.Redraw();
5171         continue
5172     case 'U'-0x40:
5173         iin.line = ""
5174         iin.right = ""
5175         iin.clearline();
5176         iin.Redraw();
5177         continue;
5178     case DEL_RIGHT:
5179         if( 0 < len(iin.right) ){
5180             iin.right,_ = delHeadChar(iin.right)
5181             iin.Redraw();
5182         }
5183         continue;
5184     case 0x7F: // BS? not DEL
5185         if( 0 < len(iin.line) ){
5186             iin.line,_ = delTailChar(iin.line)
5187             iin.Redraw();
5188         }
5189         /*
5190         else
5191             if( 0 < len(iin.right) ){
5192                 iin.right,_ = delHeadChar(iin.right)
5193                 iin.Redraw();
5194             }
5195         */
5196         continue;
5197     case 'H'-0x40:
5198         if( 0 < len(iin.line) ){
5199             iin.line,_ = delTailChar(iin.line)
5200             iin.Redraw();
5201         }
5202         continue;
5203     }
5204     if( ch == '\n' || ch == '\r' ){
5205         iin.line += iin.right;
5206         iin.right = ""
5207         iin.Redraw();
5208         fputc(ch,stderr);
5209         break;
5210     }
5211     if MODE_CapsLock {
5212         if 'a' <= ch && ch <= 'z' {
5213             ch = ch+'A'-'a'
5214         }
5215     }
5216     if MODE_LowerLock {
5217         if 'A' <= ch && ch <= 'Z' {
5218             ch = ch+'a'-'A'
5219         }
5220     }
5221     iin.line += string(ch);
5222     iin.Redraw();
5223 }
5224 EXIT:
5225     return iin.line + iin.right;
5226 }
5227
5228 func getline_main(){
5229     line := Xgetline(0,"",nil)
5230     fprintf(stderr,"%s\n",line);
5231     /*
5232     dp = strpbrk(line,"\r\n");
5233     if( dp != NULL ){
5234         *dp = 0;
5235     }
5236
5237     if( 0 ){
5238         fprintf(stderr,"\n(%d)\n",int(strlen(line)));
5239     }
5240     if( lseek(3,0,0) == 0 ){
5241         if( romkanmode ){
5242             var buf [8*1024]byte;
5243             convs(line,buf);
5244             strcpy(line,buf);
5245         }
5246         write(3,line,strlen(line));
5247         ftruncate(3,lseek(3,0,SEEK_CUR));
5248         //fprintf(stderr,"outsize=%d\n",int)lseek(3,0,SEEK_END));
5249         lseek(3,0,SEEK_SET);

```

```

5250     close(3);
5251 }else{
5252     fprintf(stderr, "\r\ngetline: ");
5253     trans(line);
5254     //printf("%s\n", line);
5255     printf("\n");
5256 }
5257 */
5258 }
5259 //== end ===== getline
5260
5261 //
5262 // $USERHOME/.gsh/
5263 // gsh-rc.txt, or gsh-configure.txt
5264 // gsh-history.txt
5265 // gsh-aliases.txt // should be conditional?
5266 //
5267 func (gshCtx *GshContext)gshSetupHomedir()(bool) {
5268     homedir, found := userHomeDir()
5269     if !found {
5270         fmt.Printf("--E-- You have no UserHomeDir\n")
5271         return true
5272     }
5273     gshhome := homedir + "/" + GSH_HOME
5274     _, err2 := os.Stat(gshhome)
5275     if err2 != nil {
5276         err3 := os.Mkdir(gshhome, 0700)
5277         if err3 != nil {
5278             fmt.Printf("--E-- Could not Create %s (%s)\n",
5279                 gshhome, err3)
5280             return true
5281         }
5282         fmt.Printf("--I-- Created %s\n", gshhome)
5283     }
5284     gshCtx.GshHomeDir = gshhome
5285     return false
5286 }
5287 func setupGshContext()(GshContext, bool){
5288     gshPA := syscall.ProcAttr {
5289         "", // the starting directory
5290         os.Environ(), // environ[]
5291         []uintptr{os.Stdin.Fd(), os.Stdout.Fd(), os.Stderr.Fd()},
5292         nil, // OS specific
5293     }
5294     cwd, _ := os.Getwd()
5295     gshCtx := GshContext {
5296         cwd, // StartDir
5297         "", // GetLine
5298         []GchdirHistory { {cwd, time.Now(), 0} }, // ChdirHistory
5299         gshPA,
5300         []GCommandHistory {}, // something for invokation?
5301         GCommandHistory {}, // CmdCurrent
5302         false,
5303         []int {},
5304         syscall.Rusage {},
5305         "", // GshHomeDir
5306         Ttyid(),
5307         false,
5308         false,
5309         []PluginInfo {},
5310         []string {},
5311         "",
5312         "v",
5313         ValueStack {},
5314         GServer{"", ""}, // LastServer
5315         "", // RSERV
5316         cwd, // RND
5317         CheckSum {},
5318     }
5319     err := gshCtx.gshSetupHomedir()
5320     return gshCtx, err
5321 }
5322 func (gsh *GshContext)gshellh(gline string)(bool){
5323     ghist := gsh.CmdCurrent
5324     ghist.WorkDir_ = os.Getwd()
5325     ghist.WorkDirX = len(gsh.ChdirHistory)-1
5326     //fmt.Printf("--D--ChdirHistory(%d)\n", len(gsh.ChdirHistory))
5327     ghist.StartAt = time.Now()
5328     rusagev1 := Getrusagev()
5329     gsh.CmdCurrent.FoundFile = []string{}
5330     fin := gsh.tgshellh(gline)
5331     rusagev2 := Getrusagev()
5332     ghist.Rusagev = RusageSubv(rusagev2, rusagev1)
5333     ghist.EndAt = time.Now()
5334     ghist.CmdLine = gline
5335     ghist.FoundFile = gsh.CmdCurrent.FoundFile
5336
5337     /* record it but not show in list by default
5338     if len(gline) == 0 {
5339         continue
5340     }
5341     if gline == "hi" || gline == "history" { // don't record it
5342         continue
5343     }
5344     */
5345     gsh.CommandHistory = append(gsh.CommandHistory, ghist)
5346     return fin
5347 }
5348 // <a name="main">Main loop</a>
5349 func script(gshCtxGiven *GshContext) (_ GshContext) {
5350     gshCtxBuf, err0 := setupGshContext()
5351     if err0 {
5352         return gshCtxBuf;
5353     }
5354     gshCtx := &gshCtxBuf
5355
5356     //fmt.Printf("--I-- GSH_HOME=%s\n", gshCtx.GshHomeDir)
5357     //resmap()
5358
5359     /*
5360     if false {
5361         gsh_getlinev, with_exgetline :=
5362             which("PATH", []string{"which", "gsh-getline", "-s"})
5363         if with_exgetline {
5364             gsh_getlinev[0] = toFullpath(gsh_getlinev[0])
5365             gshCtx.GetLine = toFullpath(gsh_getlinev[0])
5366         }else{
5367             fmt.Printf("--W-- No gsh-getline found. Using internal getline.\n");
5368         }
5369     }
5370     */
5371
5372     ghist0 := gshCtx.CmdCurrent // something special, or gshrc script, or permanent history
5373     gshCtx.CommandHistory = append(gshCtx.CommandHistory, ghist0)
5374

```

```

5375 prevline := ""
5376 skipping := false
5377 for hix := len(gshCtx.CommandHistory); ; {
5378     gline := gshCtx.getline(hix,skipping,prevline)
5379     if skipping {
5380         if strings.Index(gline,"fi") == 0 {
5381             fmt.Printf("fi\n");
5382             skipping = false;
5383         }else{
5384             //fmt.Printf("%s\n",gline);
5385         }
5386         continue
5387     }
5388     if strings.Index(gline,"if") == 0 {
5389         //fmt.Printf("--D-- if start: %s\n",gline);
5390         skipping = true;
5391         continue
5392     }
5393     if false {
5394         os.Stdout.Write([]byte("gotline:"))
5395         os.Stdout.Write([]byte(gline))
5396         os.Stdout.Write([]byte("\n"))
5397     }
5398     gline = strsubst(gshCtx,gline,true)
5399     if false {
5400         fmt.Printf("fmt.Printf %v - %v\n",gline)
5401         fmt.Printf("fmt.Printf %s - %s\n",gline)
5402         fmt.Printf("fmt.Printf %x - %s\n",gline)
5403         fmt.Printf("fmt.Printf %U - %s\n",gline)
5404         fmt.Printf("Stoutt.Write -")
5405         os.Stdout.Write([]byte(gline))
5406         fmt.Printf("\n")
5407     }
5408     /*
5409     // should be cared in substitution ?
5410     if 0 < len(gline) && gline[0] == '!' {
5411         xgline, set, err := searchHistory(gshCtx,gline)
5412         if err {
5413             continue
5414         }
5415         if set {
5416             // set the line in command line editor
5417         }
5418         gline = xgline
5419     }
5420     */
5421     fin := gshCtx.gshellh(gline)
5422     if fin {
5423         break;
5424     }
5425     prevline = gline;
5426     hix++;
5427 }
5428 return *gshCtx
5429 }
5430 func main() {
5431     gshCtxBuf := GshContext{}
5432     gsh := &gshCtxBuf
5433     argv := os.Args
5434     if 1 < len(argv) {
5435         if isin("version",argv){
5436             gsh.showVersion(argv)
5437             return
5438         }
5439         comx := isinX("-c",argv)
5440         if 0 < comx {
5441             gshCtxBuf,err := setupGshContext()
5442             gsh := &gshCtxBuf
5443             if !err {
5444                 gsh.gshellv(argv[comx+1:])
5445             }
5446             return
5447         }
5448     }
5449     if 1 < len(argv) && isin("-s",argv) {
5450     }else{
5451         gsh.showVersion(append(argv,[]string{"-l","-a"}...))
5452     }
5453     script(nil)
5454     //gshCtx := script(nil)
5455     //gshell(gshCtx,"time")
5456 }
5457
5458 </div></details>
5459 <div id="gsh-todo"><summary>Considerations</summary><div class="gsh-src">
5460 // - inter gsh communication, possibly running in remote hosts -- to be remote shell
5461 // - merged histories of multiple parallel gsh sessions
5462 // - alias as a function or macro
5463 // - instant alias end environ export to the permanent > ~/.gsh/gsh-alias and gsh-environ
5464 // - retrieval PATH of files by its type
5465 // - gsh as an IME with completion using history and file names as dictionaies
5466 // - gsh a scheduler in precise time of within a millisecond
5467 // - all commands have its subcomand after "---" symbol
5468 // - filename expansion by "-find" command
5469 // - history of ext code and output of each commoand
5470 // - "script" output for each command by pty-tee or telnet-tee
5471 // - $BULLETIN command in PATH to show the priority
5472 // - "?" symbol in the command (not as in arguments) shows help request
5473 // - searching command with wild card like: which ssh-*
5474 // - longformat prompt after long idle time (should dismiss by BS)
5475 // - customizing by building plugin and dynamically linking it
5476 // - generating syntactic element like "if" by macro expansion (like CPP) >> alias
5477 // - "!" symbol should be used for negation, don't wast it just for job control
5478 // - don't put too long output to tty, record it into GSH_HOME/session-id/comand-id.log
5479 // - making canonical form of command at the start adding quotation or white spaces
5480 // - name(a,b,c) ... use "(" and ")" to show both delimiter and realm
5481 // - name? or name! might be useful
5482 // - htar format - packing directory contents into a single html file using data scheme
5483 // - filepath substitution should be done by each command, expecially in case of builtins
5484 // - @N substitution for the history of working directory, and &spec for more generic ones
5485 // - @dir prefix to do the command at there, that means like (chdir @dir; command)
5486 // - GSH_PATH for plugins
5487 // - standard command output: list of data with name, size, resouce usage, modified time
5488 // - generic sort key option -nm name, -sz size, -ru rusage, -ts start-time, -tm mod-time
5489 // -wc word-count, grep match line count, ...
5490 // - standard command execution result: a list of string, -tm, -ts, -ru, -sz, ...
5491 // - -tailfilename like tail -f filename, repeat close and open before read
5492 // - max. size and max. duration and timeout of (generated) data transfer
5493 // - auto. numbering, aliasing, IME completion of file name (especially rm of quieer name)
5494 // - IME "?" at the top of the command line means searching history
5495 // - IME %d/0x10000/ %x/ffff/
5496 // - IME ESC to go the edit mode like in vi, and use :command as :s/x/y/g to edit history
5497 // - gsh in WebAssembly
5498 // - gsh as a HTTP server of online-manual
5499 </div></div> (^-)/ITS more</div></details>

```

```

5500
5501 //<span class="gsh-golang-data">
5502
5503 var WorldDic = //<span id="gsh-world-dic">
5504 "data:text/dic;base64,+
5505 "Ly8gTXlJTUUVMc4wLjEg6L6e5pu4ICygmDIWLTA4MThLkQpzZWthaSDkuJbnlYwKa2g944GT"+
5506 "Cm5uIOOCkwpuaSDjgacKY2hpIOOBQp0aSDjgaEkaGeg44GvCnnLIOOBmwrYSDjgYKaSDj"+
5507 "gYQK";
5508 //</span>
5509
5510 var WnnDic = //<span id="gsh-wnn-dic">
5511 "data:text/dic;base64,+
5512 "Pgl1dEGyY2hhcn1l4D0iVVRGLTgiPgo8dGV4dGFyZWEgY29scz04MCIyYzZzPTQwPgovL2Rp"+
5513 "Y3ZlcglHU2h1bGx0c001NRVxzZG1jdG1vbmFyeVxzZm9yXHNxbm50cy8yXHMvLTA4MzAK"+
5514 "RlNOZkxscUdTaG9yY2h1bGx0c001NRVxzZG1jdG1vbmFyeVxzZm9yXHNxbm50cy8yXHMvLTA4MzAK"+
5515 "44G+44GICeWqJehJjOpuYWhZQnlkI3liY0K44G44GL44GuCeS4remjJgpaYWhbm8J5LlI"+
5516 "6YeOCndhCeOCjwpp0Yonjg8Kc2k44GXCnNoaQnjgzcKbm8J44GuCm5hCeOBggptYonjg4K"+
5517 "ZonjgYKaGeg44GvCm5hCeOBggprYonjgYsKbm8J44GuCmR1CeOBpwpzQonjgZkZVxzCWFj"+
5518 "ac8ZG1jCWRpYwplY2hvcWVjag8KcmvWbGF5CXJlcGkheQpyZXBLYXQJcmVwZWF0CmR0CWhh"+
5519 "dGvccysnJvKl8hVksLVIOiVNOiVTJwp0aW9uCXRPb24KJXQJXQJLY8dG8gYmUgYw4gYWN0"+
5520 "aW9uCXJwvdGV4dGFyZWE+Cg=="
5521 //</span>
5522
5523 var SumomoDic = //<span id="gsh-sumomo-dic">
5524 "data:text/dic;base64,+
5525 "Pgl1dEGyY2hhcn1l4D0iVVRGLTgiPgo8dGV4dGFyZWEgY29scz04MCIyYzZzPTQwPgovL3Zl"+
5526 "cg1HU2h1bGx0c001NRVxzZG1jdG1vbmFyeVxzZm9yXHNtdWlwbW90cy8yXHMvLTA4MzAK"+
5527 "c3UJ44GZCm1vCeOCgpubwnjga4KQonjgYK2hpCeOBQp0aQnjgaEKdWNoaQnlh0uKdXRp"+
5528 "CeWGHOpzd1vW8J44GZ44K44KCCn1bW9tb21vCeOBmeOCguOCguOCggtb21vCeahgpwt"+
5529 "b21vW8J5qGD44KCCiwsCeOAgQouLgnjgiIKPC90ZKh0YXJlYT4K"
5530 //</span>
5531
5532 var SijimiDic = //<span id="gsh-sijimi-dic">
5533 "data:text/dic;base64,+
5534 "Pgl1dEGyY2hhcn1l4D0iVVRGLTgiPgo8dGV4dGFyZWEgY29scz04MCIyYzZzPTQwPgovL3Zl"+
5535 "cg1HU2h1bGx0c001NRVxzZG1jdG1vbmFyeVxzZm9yXHNtAGlqaW1pXHMvL1x2MjAyM0w0MDwM"+
5536 "CnNpCeOB1wPzAgJ44GXCMppCeOBmApra2QonjgBk8bmEJ44GqCmplCeOBmOCQp4eXUJ44KF"+
5537 "CnuJ44GCGm5pCeOBgprbwnjgZMKYnUJ44GZCm5uCeOCkwpubwnjga4KY2hpCeOBQp0aQnj"+
5538 "gaEka2EJ44GLCnJhCeOCiQosLAnjgIEKLi4J44CCnhuYw5hCeS4gwp4anVlCeWnQp4bmkJ"+
5539 "5LqCmteAnlIeKa29XceWaiwprb3gY5VLCm5hbmFgdXVuaXgJNzIKbmFuYwpldW5pEgHj"+
5540 "77yX7YScm5hbmFgdXVuaVg377yX77yScuS4g+WnqeS6JhGJNzIKa29dW5uCeWai+Wihgp0"+
5541 "aWthcmFXceOBEOBi+OCiQp0aWthcmEJ5YqbcCmNoaWthcmEJ5YqbcjwvdGV4dGFyZWE+Cg=="
5542 //</span>
5543
5544 var JA_JKLDic = //<span id="gsh-ja-jkl-dic">
5545 "data:text/dic;base64,+
5546 "Ly92ZjJscU15SU1FamRyY2ptb3JzZWpKQWpKS0w0MjAyM0w0MDwMjAyM0w0MDwMjAyM0w0MDwM"+
5547 "CmtqamprbGtqa2tsa2psIOS4lueVjAgqamtqamJ44GqCmtqamAnjgYQKa2tqbAnjgYKamtq"+
5548 "amwJ44GICmtqamprbAnjgYQKa2pra2wJ44GLCmpra2rbAnjgYQKa2tFamwJ44GqCmpra2ps"+
5549 "CeOBKQpqa2pAmjgZMKamtqa2psCeOB1Qpqa2pAmjgZMKamtqamqAmjgZMKamtqamqAmjgZMKamtq"+
5550 "CeOBmpqa2pAmjgZMKamtqAmjgZMKamtqAmjgZMKamtqAmjgZMKamtqAmjgZMKamtqAmjgZMKamtq"+
5551 "a2tqa2tsCeOBgApramtsCeOBgqqa2pAmjgZMKamtqAmjgZMKamtqAmjgZMKamtqAmjgZMKamtq"+
5552 "banjga4Kamtra2wJ44GvCmpqa2tqbanjgIKampra2wJ44GLCm5hCeOBuApra2tsCeOBuwpq"+
5553 "a2tqbAnjgKa2tqa2psCeOBvwpqbanjgAKamtra2psCeOCQpqa2tqa2wJ44KCCmqtqamwJ"+
5554 "44KCCmpra2pqbAnjgYKampsCeOCiApra2tsCeOCiQpqa2psqamwJ44KCCmqtqamwJ"+
5555 "amwJ44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ"+
5556 "44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ44KCCmqtqamwJ"+
5557 "gIEK";
5558 //</span>
5559
5560 //</span>
5561 /*
5562 <details id="references"><summary>References</summary><div class="gsh-src">
5563 <p>
5564 <a href="https://golang.org">The Go Programming Language</a>
5565 <iframe src="https://golang.org" width="100%" height="300"></iframe>
5566
5567 <a href="https://developer.mozilla.org/ja/docs/Web">MDN web docs</a>
5568 <a href="https://developer.mozilla.org/ja/docs/Web/HTML/Element">HTML</a>
5569 CSS:
5570 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors">Selectors</a>
5571 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/background-repeat">repeat</a>
5572 HTTP
5573 JavaScript:
5574 ...
5575 </p>
5576 </div></details>
5577 */
5578 /*
5579 <details id="html-src" onclick="frame_open();"><summary>Raw Source</summary><div>
5580
5581 <!-- h2>The full of this HTML including the Go code is here.</h2 -->
5582 <details id="gsh-whole-view"><summary>Whole file</summary>
5583 <a name="whole-src-view"></a>
5584 <span id="src-frame"><!-- a window to show source code -->
5585 </details>
5586
5587 <details id="gsh-style-frame" onclick="fill_CSSView()"><summary>CSS part</summary>
5588 <a name="style-src-view"></a>
5589 <span id="gsh-style-view"></span>
5590 </details>
5591
5592 <details id="gsh-script-frame" onclick="fill_JavaScriptView()"><summary>JavaScript part</summary>
5593 <a name="script-src-view"></a>
5594 <span id="gsh-script-view"></span>
5595 </details>
5596
5597 <details id="gsh-data-frame" onclick="fill_DataView()"><summary>Builtin data part</summary>
5598 <a name="gsh-data-frame"></a>
5599 <span id="gsh-data-view"></span>
5600 </details>
5601
5602 </div></details>
5603 */
5604 /*
5605 <div id="GshFooter" style=""></div><!-- ----- END-OF-VISIBLE-PART ----- -->
5606
5607 <!--
5608 // 2020-0906 added,
5609 https://developer.mozilla.org/en-US/docs/Web/CSS/z-index
5610 https://developer.mozilla.org/en-US/docs/Web/CSS/position
5611 -->
5612 <span id="GshGrid">(^_^)<small>{Hit j k l h}</small></span>
5613
5614 <span id="GStat">
5615 [ GShell Status Line ]<br>
5616 </span>
5617 <div id="GPos" align="right"></div>
5618 <div id="GLog"></div>
5619 <span id="GMenu" onclick="GShellMenu(this)"></span>
5620 <span id="GTop"></span>
5621 <div id="GShellPlane" onclick="showGShellPlane();"></div>
5622 <div id="RawTextViewer"></div>
5623 <div id="RawTextViewerClose" onclick="hideRawTextViewer()"> CLOSE </div>
5624

```

```

5625 <style id="GshStyleDef">
5626 #LineNumbered table,tr,td {
5627     margin:0;
5628     padding:4px;
5629     spacing:0;
5630     border:12px;
5631 }
5632 textarea.LineNumber {
5633     font-size:12px;
5634     font-family:monospace,Courier New;
5635     color:#282;
5636     padding:4px;
5637     text-align:right;
5638 }
5639 textarea.LineNumbered {
5640     font-size:12px;
5641     font-family:monospace,Courier New;
5642     padding:4px;
5643     wrap:off;
5644     spellcheck:false;
5645 }
5646 #RawTextViewer{
5647     z-index:0;
5648     position:fixed; top:0px; left:0px;
5649     width:100%; height:50px;
5650     overflow:auto;
5651     color:#fff; background-color:rgba(128,128,256,0.4);
5652     font-size:12px;
5653     spellcheck:false;
5654 }
5655 #RawTextViewerClose{
5656     z-index:0;
5657     position:fixed; top:-100px; left:-100px;
5658     color:#fff; background-color:rgba(128,128,256,0.4);
5659     font-size:20px; font-family:Georgia;
5660     white-space:pre;
5661 }
5662 #GShellPlane{
5663     z-index:0;
5664     position:fixed; top:0px; left:0px;
5665     width:100%; height:50px;
5666     overflow:auto;
5667     color:#fff; background-color:rgba(128,128,256,0.6);
5668     font-size:12px;
5669 }
5670 #CTop{
5671     z-index:9;
5672     opacity:1.0;
5673     position:fixed; top:0px; left:0px;
5674     width:320px; height:20px;
5675     color:#fff; background-color:rgba(0,0,0,0.4);
5676     color:#fff; font-size:12px;
5677 }
5678 #GPos{
5679     z-index:12;
5680     position:fixed; top:0px; left:0px;
5681     opacity:1.0;
5682     width:640px; height:30px;
5683     color:#fff; background-color:rgba(0,0,0,0.4);
5684     color:#fff; font-size:12px;
5685 }
5686 #GMenu{
5687     z-index:2000;
5688     position:fixed; top:250px; left:0px;
5689     opacity:1.0;
5690     width:100px; height:100px;
5691     color:#fff;
5692     color:#fff; background-color:rgba(0,0,0,0.0);
5693     color:#fff; font-size:16px; font-family:Georgia;
5694     background-repeat:no-repeat;
5695 }
5696 #GStat{
5697     z-index:8;
5698     xopacity:0.0;
5699     position:fixed; top:20px; left:0px;
5700     width:640px; height:90px;
5701     color:#fff; background-color:rgba(0,0,0,0.4);
5702     font-size:20px; font-family:Georgia;
5703 }
5704 #GLog{
5705     z-index:10;
5706     position:fixed; top:50px; left:0px;
5707     opacity:1.0;
5708     width:640px; height:60px;
5709     color:#fff; background-color:rgba(0,0,64,0.2);
5710     font-size:12px;
5711 }
5712 #GshGrid {
5713     z-index:11;
5714     xopacity:0.0;
5715     position:fixed; top:0px; left:0px;
5716     width:320px; height:30px;
5717     color:#9f9; font-size:16px;
5718 }
5719 xbody {display:none;}
5720 .gsh-link{color:green;}
5721 #gsh {border-width:1;margin:0;padding:0;}
5722 #gsh {font-family:monospace,Courier New;color:#ddf;font-size:8px;}
5723 #gsh header{height:100px;}
5724 #xgsh header{height:100px;background-image:url(GShell-Logo00.png);}
5725 #GshMenu{font-size:14pt;color:#f88;}
5726 #GshFooter{height:100px;background-size:80px;background-repeat:no-repeat;}
5727 #gsh note{color:#000;font-size:10pt;}
5728 #gsh h2{color:#24a;font-family:Georgia;font-size:18pt;}
5729 #gsh h3{color:#24a;font-family:Georgia;font-size:16pt;}
5730 #gsh details{color:#888;background-color:#fff;font-family:monospace;}
5731 #gsh summary{font-size:16pt;color:#fff;background-color:#8af;height:30px;}
5732 #gsh pre{font-size:11pt;color:#223;background-color:#faffff;}
5733 #gsh a{color:#24a;}
5734 #gsh a[name]{color:#24a;font-size:16pt;}
5735 #gsh .gsh-src{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
5736 #gsh .gsh-src{background-color:#faffff;color:#223;}
5737 #gsh-src-src{spellcheck:false}
5738 #src-frame-textarea{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
5739 #src-frame-textarea{background-color:#faffff;color:#223;}
5740 .gsh-code {white-space:pre;font-family:monospace !important;}
5741 .gsh-code {color:#088;font-size:11pt; background-color:#eef;}
5742 .gsh-golang-data {display:none;}
5743 #gsh-WinId {color:#000;font-size:14pt;}
5744
5745 .gsh-document {font-size:11pt;background-color:#fff;font-family:Georgia;}
5746 .gsh-document {color:#000;background-color:#fff !important;}
5747 .gsh-document > h2{color:#000;background-color:#fff !important;}
5748 .gsh-document details{color:#000;background-color:#fff;font-family:Georgia;}
5749 .gsh-document p{max-width:550pt;color:#000;background-color:#fff;font-family:Georgia;}

```



```

5750 .gsh-document address{width:500pt;color:#000;background-color:#fff;font-family:Georgia;}
5751
5752 @media print {
5753 #gsh_pre{font-size:11pt !important;}
5754 }
5755 </style>
5756
5757 <!--
5758 // Logo image should be drawn by JavaScript from a meta-font.
5759 // CSS seems not follow line-splitted URL
5760 -->
5761 <script id="gsh-data">
5762 //GSellLogo="QR-ITS-more.jp.png"
5763 GSellLogo="data:image/png;base64,\
5764 iVBORw0KGgoAAAANSUHQAAAQAAQAAAB/CAYAAADvs3f4AAAAAAXNSR0IArs4c6QAAAHhWELm\
5765 TUOAKgAAAQABEAAAUAAAAAABAAPgEBAUAAAABAAAAAQAoAAWAAAAAABAAAAIAAIpPAQAAAA\
5766 AAAATgAAAAAABIAAAAAQAAAEgAAAABAAQADAAAQAQAAACgAAEAQAAAAAQAAOCqGwAE\
5767 AAAAQAAAH8AAAAAYx1BhgAAAlwSF1zAAALEWAAACMBAAJcGAAAF3RJREFAeAhtnQuUFNWZ\
5768 xt+TukZ3iCggo/yj60sb8WgmZAvn7uG4+biSTR7YnXqDQPCkCj2aNwLd2MS1RkeUaPnoCdu\
5769 4iuX7jriJz50DGMf2VqIBeiSggCoLmMA+mu+vu//ZMD9Uldau6a2Ubyv91Gkrq3vvd6/q\
5770 fnXdx8rB88IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8\
5771 IA8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8\
5772 IA8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8\
5773 IA8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8\
5774 IA8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8IAE8\
5775 npDIkXEZClFIRMS3JUSaq98ocqUei+2kK3StuOnY5reBKKJ7Qw7MvKec2TqOoiZwoIjhFS\
5776 jBOVHcsthrb3UXEJ8hFu7DsdmFb2+u4vWmFVxbgMeZUIAE/hcKogab6e6KOLNykh56PC\
5777 HxH2VVBKORkqh3UeKiLyaOaFONJ56KoiI6w5BwomOQlyPziON9DLmXpFK/60p2P/Piyovf\
5778 8mfmM+uJWNGjn3wKqOTOlVGSFt2p2Rillgn3iJovk7YsoVMzEuVFP1RKYdfoAK2LRSB0Q\
5779 zrWocCOG6EhvgRaCj/dkt3j3g7dXKH4gKN6ArS0zPzYerqS6RAozDQqfk79SKTRXHu/+e9FM\
5780 L6aes88pu/PhlPn1TLQJKS73dPXR20ur7iWPC8QhbNnCyhu1LlryOTqVYF5JfvqBL7jX\
5781 +cNHj9j5gJrydLjHy39o84D40H2Qx8THaPeFIOUo+wlC+Knykh5FGEVOWGqAExB8xeMoLYH\
5782 rIkbq9HPE52Vgq14h89FUA6KJyFyBbQbnzLj4zFieanDHCwVooeiVQob/5CY9P10UeOH\
5783 +zghUhn9SgOqrm0WguriK9RpBD4Y6uqQdD5TUOM63zD3MHeey14V491sbdKyxkGH1CPFR\
5784 UJ6toACF7F9Vf58NBdHT0MBAE74Ent+eWr+Lz/QTw60AdB7QUjUjs/OA7COobNBCEmuZ\
5785 tCu/COG28f1pVKElTPFV8JuRasEahHvXaR1gucobPpYUdo4+ofeBdyb8L4tz9XeSXPAMOC\
5786 bgGvov0glzgGw47F392XnHhdC+Mwf3JTjnt2y2YlJYBJXNU25KIKyck1sxXRld66mcewV\
5787 aJovy/VbacMeqvEP46/z1nJj9jxi17VL53Z15MtVap1QGLNHw5PdqYkNTQ1Z2b8ncMGZ2V\
5788 aOoJsdYvV0AZzEYaidv6FJ35CS4jXZk9hir7e27zmp3T8hLpKjYicJpV1HtK/DJFU4Jw1\
5789 1ImhXMSIR9fzgzRkx4w/C+HQSPe+krbIYrN3qEPTNahsHaL2x5S5Q5CoppVdPggcgqm/8e\
5790 7/zdoAptag/mLKJ7U0VG0xybTdx/Ex/Ptfa/iTr7Ku+cSoiCXUwrohtUx16WvE9H+ccVJ\
5791 pd/CE42AK2IUP1YTK1L/sjY5E5PVhqr728NzvftzuvDODGy9GoopuuhNLNfctX48YHLZP\
5792 f/8hpXvu/43Og9xtq6Ytcv1KDC3EmNDQn9nbE2le7wKE1bOK65icBu0Egh3JaW8zdwKFuH\
5793 hrauc62wckdjZK8EUXMae71ZuqCu2nb16eVn1Ji9/P7eW+ioAoAgF+nI3JL5f8dn9ipA\
5794 WNW4rPy9j3xupeEd/HXzNzTsvesL2vWHWIR9mu5rvVZ9foS4v/LfmgdIpfHGdLfm2cu\
5795 gJly2wOENPa23Eci+vd+ZYNCNjYrNyhYGAo8JroJ7AMrRiQcCjnrW5FpTn+frTwdh4s1Uv\
5796 bV1WbhfLcRF04qagRD7176/zBjKyLD5pBi25Wi4wQu7tikPecOpuW+Kj0sgP8GHNAzuwL\
5797 iO2yWdh9zEr2xODRqVQF15QxH6owvJrKAAM46pyT+RxAJVLjw7vY9/+CeUBmk168/rPQn\
5798 mCufKzaldfN/y18Ga5iWc3dkIKhsyvZuCYSGV/KHewhFWRKAMMEd8EKex+rHF12A9bt2d172\
5799 2qNzOvzCDYmEtNy7QogDXWIKAIQI7COQzchyADWnerqN5VXtkc3sdGp2OrtmWJU7A+EH7\
5800 yhbUgm1IX7f7K1DwaRyUFN42F1uxNDVvEtamL6sYCR267bZaW2px8Nfmehz3EM+mgso1k\
5801 d3/zNBGELXPWuzXg1Cq5eW5/zBgy54aWogWkfnWbaptcewT4FUBov32gew8DLzDTMAj\
5802 augp7t/bMXh+yw/egJGKoTksy2d+gFbb9vOdvX5B12TOR+Wfjy0p6UOXOYNgR/3a7a3B\
5803 Fgeua6qv247v88Fdv3r1dBw34SGP9i0DC9s5Xmknh9KaAmMyJ6dk1Pzm2D3cu7vtw5C\
5804 h/YrG1p7wXp/VvuRudC+wsq54ym+8zXKOgYRSPRA4IKoGz1i8b6ytagcEPmb9v/m09eUATz\
5805 Jow6VnPCmKHzj+sNnpHsCJyja6cRrRMyrGkiw4F15UouliL1RW7fmmLEx3z2+/GfW1LU2\
5806 Y72b6EazkfYoPctJi1501nJyLdrFRUzP1/3pmkuG/yN9gAoGYWTF7neVixv/6CHUgh1lUH/\
5807 f9Uvo+qG70q3zzFL8xq+zW+/8FP6Wf7x7XshInlayWdz2X1ULm/4uLmPwNoA5ucgdOL9\
5808 ZFAecgoxzhT6G4Q1NR5D0j9xuv1cy+rFbcuJVanlkV0CefphUbICLRmV1+9KP4vngHq6F2\
5809 NCqMSiCsnCkFxd+mtfLBwuxdmFBozQ7/194225Y3TCzPQWhthG2zHraJo/yb0kdhpanZq\
5810 KxWF66/8Cb5AHCzbDpnhUjce6YFowlgzeMmtqNcDekTxiXVuc3Lk4YvTJepuq5tgSFWKXdA\
5811 ufu9MfiWiG3sgntk736+3EXQ0WvzVeqSpvrZmc2afYsVv461+O4KvYvGicCugG2r2poyPv\
5812 U01m2JwZEO+eKf0DfTKXfW2U9x70/bgzct5zoPoio1+vdpyDjcdxrd34U9XceHrloSktt3ug\
5813 AcwtX0O9PFn+gWdS60DCFoDrAxneCfRXWUSO93PzBXN7VAe+qvr506/204LXgngLbrC\
5814 7HGhdRvHz2LIMYVVMvgm5zTPP5+7volRR/zJ1OYlx+8oHzEb+CV/OTU5ic3NGfjks30MZ\
5815 tFute1l+Y4iyfAcwkjzqpZy6h1gJebWpLXyo09/j8k//WW3x32qQPHv5AmTPI1DFN2Op6\
5816 fz5yW4HfmdX+Buy4NvU73yEFBOK65icot+zjP+8qf4JKyitNgKTB/gzT0MKKAC18jJPL\
5817 A4PCxYmPKOtkrY84HpyOsws/BsqyT2RGZ6r210gA9sBhEp46hsE2ratmoJegrugBWB2Pv\
5818 NYD1B4OSTYmBmcds2E/GG2zvrF7Uejsqw/7A7gUEH6Ky19q3fpoQvGxt4dz+Ueg+Lmy5V\
5819 bjjYtO+b5Lsgp5Nz6nwbfFHudaYgemZy4ap1z5dlbBYA3NQTc4F3RKYfOTKAuF9XryOLwU8\
5820 dMCC/H29vo0TGMV1C+izhTua27rgAebk4+8H3P553oOyu/WHj21Zwbd7z2XLuv4f1gmQSV\
5821 2ML+6LkmhorvaWqne11yZ/gLlX+IBncn2PQ7F9Y5XQEn/qUa+Hr3URAgLMTLRG3bfPyEtP\
5822 6m5doYcZcZmX9nQ2JAggbYmXSL9vzQSGbfxUBjHpbXbzM+vKueRBRIotE/Bw8ogf/LIHz\
5823 /97cnsh681T7DgnQRE18VtZ29eWTS5JF71FSzovLYfTlvqUT0b62etccBRO1lHes68S8Y\
5824 20X2degmRW7S7ng7dkRv19rLztoMPBK73naAYrdZfm+5DZsymDymaHnClkvPOVHGSFrQs\
5825 wCY6RwU9DkxSMU9vQXMAx+ePguLw8/dvfg6ULLPvsPbpXspOniQwagE1sm9ggNxc0EOg1vj5\
5826 7tBBBjAdhKmpdV0/q/irWbf44t5CNKQKwAq7DsuJzh1Cz8bk+1u2u78FXyWfklQ4/qY2x\
5827 TyvJX8boyWmz6w9/Ojwz7pUtV1pLpNQ2UXLo8PKODMluvoOTDjLyxcrNWHHEjQWsyKrkPs\
5828 2JH14LpJicQXOyp6nMs5fysKeille0G95+WXeJ3m5mcmj5e5b+1yBZVELXgJrMdnY/HMK0\
5829 aP7Md34PueY28DWDovSjzXVF/xsFe+Lpz/jjQ9eih94ZwqV562+CUHv31MtnjShfKorChf\
5830 wKgz9PwIRCTCRjWh5+/ocSLQ1zG52BvtG+wOpqXRYEwcarfRdbSgC5BD/PysXBhakPWO\
5831 qzX9Y4L100AAB4x5we8qDSH06++bDnwzjzFXyAUvY6Ece0017SAZkx0uqxmZB9RcaVyx\
5832 2CMBjAdTRWwYkrIwy4myTH9zt3R93/8X1j0ESWety7qPIj1odwkAmhFEA2K6D1dNe6h\
5833 H52HwYwIALQHQOUZvYrznTlS7rgu40YBJq4JBWJCayRhTyeY4X8/xCw+rUS9L5ye50A+W\
5834 8v0wON2ZxAW7VADPzCEDpXpdsLXoKefrEM+yj47eAa7YxZjXm+61FZUL46ch70Od6Q0\
5835 Wncf9BTVXbs6z3nXpVimkjhJubTFKkrBaqLQCMIwubiPtYKlHhWZag8YKoeMxji9Y9LY\
5836 PwK79U/55Bk75FSXmchwhj79Y35xY7qu8YspvtbqSG+55hdjn6Y5E6rEfyQV02XoeLrhmj\
5837 YwqG5S2p1IOK5jzgs+2LB1B4Z6/g+uoa6yUWOY1jzcCuog411qxVQOYep1wul1X4pPR\
5838 zD3GL6w1VE4jA35xePkINLSuBb/34RcwB6JXGzGzrflBBjBhJH7t1WBGDV4dbieXgPpb\
5839 NQT31qMhZTETHUrxnV45r8FPQWRNdQfV2qB1XF16+rQDLV82CTnVYB1dBS2JfBPwMJ\
5840 AW3rXYbmq9KMLmChJcnvUN5fKMRc2LbzJbK8mU55en4x/2rLdJQz2n1KkyuU01pdqcmfz\
5841 qKpC/ahfXooVi+JtoFimZujN8F7QHmAMXdAaUeTK6C7F075UUKygq5Oz33vV/Z0C7b+scH\
5842 Ltnp1tH3YeW841pgt4JWAnu7Pn5xwqjxb4IMabC3Q8rflZPCJfTc0Sf08nAdzSFWqfYHBU\
5843 nmlDJITHGH3eSRt+42Mk5KwCtsrFme35RjTvorP3rm49VMogfP80iD191X61dvbKmkqjvb\
5844 NfydX9m8Wim1zMLKzESL/VzQSkDPzcdYcye71q/B4XKfKQANeK3mL47r29fQL/gat+/vrEO\
5845 dTDX0U9UwBKVUMfH9MYLzJpVxxu0fPOO/pTehdod/1XXGzawFuXp6G1Lz+eme2X9lbo\
5846 OXuU119F0hLAKGqHafa5NVPXhK7X0gLUOmRm+JAFesfnaKzLrHZXLYBf5edIUwKc1/wd7\
5847 fdJL72vEtDEPIqWkZj6zFP/d5duzt+ZHXihfKlnhs7umT01AJjkyVScenpJ1WA1AACzAE\
5848 dq2Sx/s+NLD0pELXVtD/SkUr+JL5/9vsbl75z+bvNS8Q2EuQN/Oa3x1/FJZS/VZ30EGCBg\
5849 ePdCYCRORCKR3q6VlOpof7EXvDAaVzcgJOECZ56CYcmxZ/7CyuWar2IIN2x4NOC075/\
5850 4yMTRk3XuyfGJgmxT/xdBpt8uSR1F711luoFJtQm3U17cKXfygMwF5DvwpV9RPAeh07FRV\
5851 hUL4693pwu1Yn+FXOC+Cy0vRWXzyh/w3n7fiiBreUtSvURMitjpkWRWYmPKZEmd2FCiM\
5852 dmf1f6+eW10/651MCCDD2YFE12dFycqj38ARAbQSPG1XSCUGCaCrDOUysazvucZ6zAVTF\
5853 LLGqFPXjFyIthCkphr+cN+r76LoLJ1d3d451+sndV9Yr4cWvG9+SrTx6G/areLXB4WX\
5854 tgzv7Wk4n+Z8f/FfzUKIa3ky5ULmo9CE8N3HGLinI5ISrny32hsXoRnTBMbWmiP9T7o3\
5855 j0g8vnn35zeCGFY1gCmlw2/fvicjoJxytIeoL10XvRGHMynZ1/IJL6W3j5y8j+711dy057\
5856 xLdJmM+XoFOgtrucgEUTDVIpFcnvWAE2KAeVrAeG5T3tjBGQT+5rCIU+U1BzP1PjRmPVP\
5857 4YEuz9wP9xlfv/oppuyDp9NpYih91/XXNovNSd5dGG8Bwms31CzfrkCQUTCSZHj+wm8Q\
5858 JV7X3Xm6wqjL5r6LVB68ToEXTHJ/4Cdw24+uzFvsJrT11RkFOOALtznPzfd2SD12QRQ\
5859 8Y588pdsboVhRLQD6exvrEOj9y49DQPkC5Zmjzj021LdV7y3zfl8qmsDmOPARTWFC3i\
5860 N1NQCwK1JevagRMz78D2ZVeFmHcdPFUC86nFBBS5KF1fPMRHE6FOOS0Aotvm/d8Vv8km7D\
5861 C58YrseFULvsvpLpx79z64erdZNYUNLkileJdaUak7Joor315x+YA9CbQBDf/ck7JKHDdB\
5862 E5sg69OKMH9pdRjd6v3vgEVybdOcuc51VM9nO/QaPP3KZ1ve82WcmJjK30Kx+3ORQ0E8K1wN\
5863 blXaEhe29JqBL8oF8GKaM6n5P9mdG5BmtIkpmc22R7BHSKjJp0kMctCt/KAM1soJtXeJK\
5864 v7q/Ozmzbn/Z5IOHT3+NPgZn2eyx7uiZODJM9xoyTcZBTOya+vndqW3URP1jYxmbDoe1\
5865 zq4BzYggsLmphdLIXcmLwXskzBGwa94ostvEbf7141iK3oiO5Mxod+r9/I2Vx80P9Ec3\
5866 xp7QYUu8JTMcatc0+NeY/99v3xb+21bh03cnot1JdfCZnzkeapsDN/VdJ4XP4Cn8+hWp\
5867 9zduKz2Q3fev05lytqto30pz9Ec5sHOY+FXFV150r6xr5HkDFMGAadKQ3yA09dydfdj\
5868 ppf5kjq6grN13dfyKI5h1400Kj1azehBJNtWTFBAGv1IawS2vTtAHf5B0dYerpEaP\
5869 mR1f1XOm8Xm4n/fyb6aVg2fTy5SkWno2mPFSF3sgCf3o4UGGSj/wI54BwLfbvWab720b\
5870 Xx/MwrgL9zXPQmBmX5CiAfjIhTyXhS7BkMfG8mT+D3CdFP2qod1vNN3v3d6owx7f\
5871 koSF0PepkZFeqJWQtld70c6dnp1H7zi0z933hOLHMYU1rEhZ7ptxeV69XW+3Jdsam6tO\
5872 E8NSY1G5j8EajzNR0adga7IevOR21BSCcV820u5Ue1JbspvqHEcusjRkYkLW0VSSU1nTmW\
5873 LaycEXhPswIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIK\
5874 QAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIKQAIK

```





```

6000 2yDwtdlsVHHx9FdrtlhlFgMKMF/29W2qY7kZuDUzlbutncUdLMKyR600L450y2RSIuy8E\
6001 213vcxgbegYZDF3bH6gAT7f3yc0VArNdIcmX/886DlmVSB1TZPt55DnaCmhXwpHmaF0mmbfm\
6002 vhdsgJNGjXhr80Q3u841F/WeBP4PAJLZGLgtD1Zu7VBWBRp9q/ubaB6EYVKIL/ulF8EkgSvC\
6003 V9egEnDQ/DSrWQYGRXRlQ34E0x/ssvPD73Wk9owbTpeEzP++jFtSoqyOazc6xR/Ofj5QrDY\
6004 BnasW4zhsvt+2cYr5qZQvdp5Wclt/GQSumzYw6HgmGfPjRO/y4aaU+7GfYl+LS0Kkwc+3\
6005 AjxxwWLD+BYJ07RAG6LHTuc8jclEpnNz5qZ4P88XK09B9p55d283TmJnu8zUPT+OL/PC\
6006 dc2P4UFahmsfn47H6RP12Vnujzrz5LuifLwSLBe0Y772KozqJjYlzn2FL00agK4U6b8+BXyQ\
6007 TkKwenvfahpuTqZ2fH6qhg26/jB8aPKnoB59jZLh9L+084E59USUQhki65Vwg6P3njyDw\
6008 85ziR500l+qabrR6Tso+rzbMqXwv2xrc0csSSmqL/fCFY7LPdZ2LJzrSK+C5dELh6ixITLW\
6009 V1/nm4/cmbCw+nXmNwee48EznelWaoF+EkyUro1IDOGpL3PawpZRGfPlnIhtCOXYQ5CLgPQW\
6010 RCj4fbl+LbuV3VncC8bZf4KvlizeZfVNVu6qj8Qv++Y0t29BuzqWrgjwZDI1oBJwXzE18Vix\
6011 KEPL9fdBxp/2X6sgXKM3dfClatfdBadBN1U0UNh1AEwg6Bw93sevgj1HMULPw/b14a6npj\
6012 pkSWLw06fYmN+v3MLU6MwfbD3KvyWtXWj2zUcu04jSj+6WUYjBTLLjPSV1okE327S/NdWx\
6013 Mg5+21W6tW1T14TY/bUnDxms71u9baga20YX5DbUX1z9BRpGfVdrDHJ51k3m3z394VgdsYp\
6014 qZnk1kQbbVhBteHI610vu/zgszaeFLR+tNOBCXy90Ka7q7BbtQ6tbaU/vYiPhu8xhzA4R\
6015 o1Ma0u3Q4pY2WHwCt1aI7Ndi3bXo2P7v2p70cmEfyecw8L4Q6770Ev+htzPnED+mNFY/W2\
6016 9LRArH5EJ/vQ5gFllw7stTREMd4gAu5+Q3T6aRSqdmx7z+/GB47ui290Uwv9jX4AnJ71LIS\
6017 16Y+xi8srm6YcrOxul4KlyS86Be91l0YHs791/4cxvgnh2jWB1j1XxexYQuZU0g5Wduq\
6018 Y6xMFg2XReb6wTrTjP5NO7zuP3v9rmdNn4F5eSkoH0t.ab6aStQot7/bEugSubPmhc27B\
6019 0YQh510JvclY04FkKoz+kgw+oaJdVEsgVER9PehP+SrXWnkMNL6VpOnUikLzm+PveQgF\
6020 h4F8J1j9WwOrt+64stf500WEzd2G5tCd/FZS/VXH3nagrQUL+4B2j8m8Ss/Fm19D3Mebjwo\
6021 kRn3KXzuzgZpsKZU1cbOcrjMPhQlabXm1gdsu1315d3JlR9y0Wm9Np1kTie/CQYIdtW2VZ\
6022 8/kpxqKkvclbdveIDDt0Usc+Rxsmd1Pnxm1w78tYm6HZGdCt2fgZn+8xzSRBvQ4zrhE9\
6023 H926s9VJSOHRzRdII7AAPQwzKI7LepLurBj0Qpxybyb/8dmn2//ll/qgnago2AwgF/38+NEl\
6024 I4af5Q5EXMARkA0I2CCP2xjNV+LMZ78Lk3V27LWv2n9w4/+6jgdkJPLq7b1TADkw1p\
6025 nIhS+QSt+HiEwSRPvengV20d6Nf7K0tloP1dj/jkUatCEBEiABEiABEiABEiABEiABEiAB\
6026 EiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiAB\
6027 EiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiAB\
6028 EiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiABEiAB\
6029 EiABEiABEiABEiABEiABEiABEiABEhD/B9wOq7SGUV++AAAAAE1PTkSuQmCC";
6030
6031 ITSmoreQR="data:image/png;base64,\
6032 iVBORwOKGgoAAANSUHEUGAAGSAAABVACMAAADYCVwJAAAAB1BMVEX///9BaeFHqDaJAAAB\
6033 HkLEQVQ4jdXtsa2EMaUYGCMX7sICKVqjXVaCBe7CarASXdaTlAWg54HwM5zEVS+mvSgS+ZBQ\
6034 8gcb4BdHYzvw8szMSaUBHNM+KA4QC8LDpDn8ogT4UpPGci2jI8IGFX3eLWpWahKnyZBQ\
6035 UEBDXaB0X2anJueYDOZnklQassPCKj4nW3E1SfwqYk6ju/vAKPhg0AlSFhve8T0dkwDMw\
6036 yMGS5uPyWHAr19k0tKv2sb3sdW2rUCqW88q4RplA9s1JPv9cTpnRD4FXkn8XaQC1wT6LzQ\
6037 Z08dHw/4+U2Gzq1S8gbqVmkfr1N6YXR8QdL00mLGTWvzPERA8AL9vboiFpSol33fsVytrL\
6038 S9wiqDznhUI38v5n783/gBU5s2ELgic8GAAAABJR0U5ErkJggg=";
6039
6040 </script>
6041
6042 <script id="gsh-script">
6043 //document.getElementById('GshFaviconURL').href = GShellFavicon
6044 document.getElementById('GshFaviconURL').href = GShellInsideIcon
6045 //document.getElementById('GshFaviconURL').href = ITSmoreQR
6046 //document.getElementById('GshFaviconURL').href = GShellLogo
6047
6048 // id of GShell HTML elements
6049 var E_BANNER = "GshBanner" // banner element in HTML
6050 var E_FOOTER = "GshFooter" // footer element in HTML
6051 var E_GINDEX = "gsh-gindex" // index of Golang code of GShell
6052 var E_GOCODE = "gsh-gocode" // Golang code of GShell
6053 var E_TODO = "gsh-todo" // TODO of GShell
6054 var E_DICT = "gsh-dict" // Dictionary of GShell
6055
6056 function bannerElem(){ return document.getElementById(E_BANNER); }
6057 function bannerStyleFunc(){ return bannerElem().style; }
6058 var bannerStyle = bannerStyleFunc()
6059 bannerStyle.backgroundImage = "url("+GShellLogo+")";
6060 //bannerStyle.backgroundImage = "url("+GShellInsideIcon+")";
6061 //bannerStyle.backgroundImage = "url("+GShellFavicon+")";
6062 GMenu.style.backgroundImage = "url("+GShellInsideIcon+")";
6063
6064 function footerElem(){ return document.getElementById(E_FOOTER); }
6065 function footerStyle(){ return footerElem().style; }
6066 footerElem().style.backgroundImage="url("+ITSmoreQR+")";
6067 //footerStyle().backgroundImage = "url("+ITSmoreQR+")";
6068
6069 function html_fold(e){
6070   if( e.innerHTML == "Fold" ){
6071     e.innerHTML = "Unfold"
6072     document.getElementById('gsh-menu-exit').innerHTML=""
6073     document.getElementById('GshStatement').open=false
6074     GshFeatures.open = false
6075     document.getElementById('html-src').open=false
6076     document.getElementById(E_GINDEX).open=false
6077     document.getElementById(E_GOCODE).open=false
6078     document.getElementById(E_TODO).open=false
6079     document.getElementById('references').open=false
6080   }else{
6081     e.innerHTML = "Fold"
6082     document.getElementById('GshStatement').open=true
6083     GshFeatures.open = true
6084     document.getElementById(E_GINDEX).open=true
6085     document.getElementById(E_GOCODE).open=true
6086     document.getElementById(E_TODO).open=true
6087     document.getElementById('references').open=true
6088   }
6089 }
6090 function html_pure(e){
6091   if( e.innerHTML == "Pure" ){
6092     document.getElementById('gsh').style.display=true
6093     //document.style.display = false
6094     e.innerHTML = "Unpure"
6095   }else{
6096     document.getElementById('gsh').style.display=false
6097     //document.style.display = true
6098     e.innerHTML = "Pure"
6099   }
6100 }
6101
6102 var bannerIsStopping = false
6103 //NOTE: .com/JSREF/prop_style_backgroundposition.asp
6104 function shiftBG(){
6105   bannerIsStopping = !bannerIsStopping
6106   bannerStyle.backgroundPosition = "0 0";
6107 }
6108 // status should be inherited on Window Fork(), so use the status in DOM
6109 function html_stop(e,toggle){
6110   if( toggle ){
6111     if( e.innerHTML == "Stop" ){
6112       bannerIsStopping = true
6113       e.innerHTML = "Start"
6114     }else{
6115       bannerIsStopping = false
6116       e.innerHTML = "Stop"
6117     }
6118   }else{
6119     // update JavaScript variable from DOM status
6120     if( e.innerHTML == "Stop" ){ // shown if it's running
6121       bannerIsStopping = false
6122     }else{
6123       bannerIsStopping = true
6124     }
6125   }
6126 }

```

```

6125 }
6126 }
6127 html_stop(document.getElementById('GshMenuStop'),false) // onInit.
6128 //html_stop(bannerElem(),false) // onInit.
6129
6130 //https://www.w3schools.com/jsref/met_win_setinterval.asp
6131 function shiftBanner(){
6132     var now = new Date().getTime();
6133     //console.log("now"+(now%10))
6134     if( !bannerIsStopping ){
6135         bannerStyle.backgroundPosition = ((now/10)%100000)+" 0";
6136     }
6137 }
6138 setInterval(shiftBanner,10); // onInit.
6139
6140 // <a href="https://developer.mozilla.org/ja/docs/Web/API/Window/open">window.open()</a>
6141 // from embedded html to standalone page
6142 var MyChildren = 0
6143 function html_fork(){
6144     MyChildren += 1
6145     WinId = document.getElementById('gsh-WinId').innerHTML + "." + MyChildren;
6146     newwin = window.open("",WinId,"");
6147     src = document.getElementById("gsh");
6148     newwin.document.write("<*<+>html>\n");
6149     newwin.document.write("<+>span id=\"gsh\">");
6150     newwin.document.write(src.innerHTML);
6151     newwin.document.write("<+>/span><+>/html>\n"); // gsh span
6152     newwin.document.getElementById('gsh-menu-exit').innerHTML = "Close";
6153     newwin.document.getElementById('gsh-WinId').innerHTML = WinId;
6154     newwin.document.close();
6155     newwin.focus();
6156 }
6157 function html_close(){
6158     window.close()
6159 }
6160 function win_jump(win){
6161     //win = window.top;
6162     win = window.opener; // https://developer.mozilla.org/ja/docs/Web/API/window.opener
6163     if( win == null ){
6164         console.log("jump to window.opener("+win+") (Error)\n")
6165     }else{
6166         console.log("jump to window.opener("+win+")\n")
6167         win.focus();
6168     }
6169 }
6170
6171 // 0.2.9 2020-0902 created checksum of HTML
6172 CRC32UNIX = 0x04C11DB7 // Unix cksum
6173 function byteCRC32add(bigcrc,octstr,octlen){
6174     var crc = new Int32Array(1)
6175     crc[0] = bigcrc
6176
6177     let oi = 0
6178     for( ; oi < octlen; oi++){
6179         var oct = new Int8Array(1)
6180         oct[0] = octstr[oi]
6181         for( bi = 0; bi < 8; bi++){
6182             //console.log("--CRC32 "+crc[0]+" "+oct[0].toString(16)+" ["+oi+"."+bi+"]\n")
6183             ovf1 = crc[0] < 0 ? 1 : 0
6184             ovf2 = oct[0] < 0 ? 1 : 0
6185             ovf = ovf1 ^ ovf2
6186             oct[0] <<= 1
6187             crc[0] <<= 1
6188             if( ovf ){ crc[0] ^= CRC32UNIX }
6189         }
6190     }
6191     //console.log("--CRC32 byteAdd return crc="+crc[0]+","+oi+"/"+octlen+"\n")
6192     return crc[0];
6193 }
6194 function strCRC32add(bigcrc,stri,strlen){
6195     var crc = new Uint32Array(1)
6196     crc[0] = bigcrc
6197     var code = new Uint8Array(strlen);
6198     for( i = 0; i < strlen; i++){
6199         code[i] = stri.charCodeAt(i) // not charAt() !!!!
6200         //console.log("=== "+code[i].toString(16)+" <<== "+stri[i]+\n")
6201     }
6202     crc[0] = byteCRC32add(crc,code,strlen)
6203     //console.log("--CRC32 strAdd return crc="+crc[0]+\n")
6204     return crc[0]
6205 }
6206 function byteCRC32end(bigcrc,len){
6207     var crc = new Uint32Array(1)
6208     crc[0] = bigcrc
6209     var slen = new Uint8Array(4)
6210     let li = 0
6211     for( ; li < 4; ){
6212         slen[li] = len
6213         li += 1
6214         len >>= 8
6215         if( len == 0 ){
6216             break
6217         }
6218     }
6219     crc[0] = byteCRC32add(crc[0],slen,li)
6220     crc[0] ^= 0xFFFFFFFF
6221     return crc[0]
6222 }
6223 function strCRC32(stri,len){
6224     var crc = new Uint32Array(1)
6225     crc[0] = 0
6226     crc[0] = strCRC32add(0,stri,len)
6227     crc[0] = byteCRC32end(crc[0],len)
6228     //console.log("--CRC32 "+crc[0]+" "+len+"\n")
6229     return crc[0]
6230 }
6231 function getSourceText(){
6232     //alert("cksum="+strCRC32("",0))
6233     //alert("cksum="+strCRC32("0",1))
6234     //return
6235
6236     version = document.getElementById('GshVersion').innerHTML
6237     sfavico = document.getElementById('GshFaviconURL').href;
6238     sbanner = document.getElementById('GshBanner').style.backgroundImage;
6239     spositi = document.getElementById('GshBanner').style.backgroundPosition;
6240     sfooter = document.getElementById('GshFooter').style.backgroundImage;
6241
6242     // these should be removed by CSS selector or class, after sevaed to non-printed attribute
6243     GshBanner.removeAttribute('style');
6244     GshFooter.removeAttribute('style');
6245     document.getElementById('GshMenuSign').removeAttribute("style");
6246     GMenu = GMenu.getAttribute("style")
6247     GMenu.removeAttribute("style");
6248     GStat = GStat.getAttribute("style")
6249     GStat.removeAttribute("style");

```

```

6250 styleGTop = GTop.getAttribute("style")
6251 GTop.removeAttribute("style");
6252 styleGshGrid = GshGrid.getAttribute("style")
6253 GshGrid.removeAttribute("style");
6254 styleGPos = GPos.getAttribute("style");
6255 GPos.removeAttribute("style");
6256 GPos.innerHTML = "";
6257 styleGLog = GLog.getAttribute("style");
6258 GLog.removeAttribute("style");
6259 GLog.innerHTML = "";
6260 styleGShellPlane = GShellPlane.getAttribute("style")
6261 GShellPlane.removeAttribute("style")
6262 styleRawTextViewer = RawTextViewer.getAttribute("style")
6263 RawTextViewer.removeAttribute("style")
6264 styleRawTextViewerClose = RawTextViewerClose.getAttribute("style")
6265 RawTextViewerClose.removeAttribute("style")
6266
6267 GshFaviconURL.href = "";
6268 //alert("outerHTML:"+document.getElementById("gsh").outerHTML);
6269
6270 GshBanner.removeAttribute('style');
6271 GshFooter.removeAttribute('style');
6272
6273 textarea = document.createElement("textarea")
6274 srchtml = document.getElementById("gsh").innerHTML;
6275 // textarea = document.createElement("textarea")
6276 // 2020-0910 ?? ... this causes inserting style="" to Banner and Footer,
6277 // with Chromium?/ after reloading from file:///
6278 textarea.innerHTML = srchtml
6279 // <a href="https://stackoverflow.com/questions/5796718/html-entity-decode">Thanks</a>
6280 var rawtext = textarea.value
6281 //textarea.destroy()
6282 var orgtext = ""
6283 + "</>" + "html>\n" // lost preamble text
6284 + "<+" + "span id=\"gsh\">" // lost preamble text
6285 + rawtext
6286 + "<+" + "span><+" + "html>\n" // lost trail text
6287 ;
6288
6289 tlen = orgtext.length
6290 //console.log("length="+tlen+"\n"+orgtext)
6291 console.log("length="+tlen+"\n")
6292 //alert("cksum: " + strCRC32(orgtext,tlen) + " " + tlen + " " + version)
6293
6294 document.getElementById('GshFaviconURL').href = sfavico;
6295 document.getElementById('GshBanner').style.backgroundImage = sbanner;
6296 document.getElementById('GshBanner').style.backgroundPosition = spositi;
6297 document.getElementById('GshFooter').style.backgroundImage = sfooter;
6298
6299 CStat.setAttribute("style",styleGStat)
6300 GMenu.setAttribute("style",styleGMenu)
6301 CTop.setAttribute("style",styleGTop)
6302 GLog.setAttribute("style",styleGLog)
6303 GPos.setAttribute("style",styleGPos)
6304 GshGrid.setAttribute("style",styleGshGrid)
6305 GShellPlane.setAttribute("style",styleGShellPlane)
6306 RawTextViewer.setAttribute("style",styleRawTextViewer)
6307 RawTextViewerClose.setAttribute("style",styleRawTextViewerClose)
6308 return orgtext
6309 }
6310 function getDigest(){
6311 var text = ""
6312 text = getSourceText()
6313 var digest = ""
6314 tlen = text.length
6315 digest = strCRC32(text,tlen) + " " + tlen
6316 return { text, digest }
6317 }
6318 function html_digest(){
6319 version = document.getElementById('GshVersion').innerHTML
6320 let {text, digest} = getDigest()
6321 alert("cksum: " + digest + " " + version)
6322 }
6323 function charsIn(stri,char){
6324 ln = 0;
6325 for( i = 0; i < stri.length; i++){
6326 if( stri.charCodeAt(i) == char.charCodeAt(0) )
6327 ln++;
6328 }
6329 return ln;
6330 }
6331
6332 //class digestElement extends HTMLElement { }
6333 //< script>customElements.define('digest',digestElement)< /script>
6334 function showDigest(e){
6335 result = 'version=' + GshVersion.innerHTML + '\n'
6336 result += 'lines=' + e.dataset.lines + '\n'
6337 + 'length=' + e.dataset.length + '\n'
6338 + 'crc32u=' + e.dataset.crc32u + '\n'
6339 + 'time=' + e.dataset.time + '\n';
6340
6341 alert(result)
6342 }
6343
6344 function html_sign(e){
6345 if( RawTextViewer.style.zIndex == 1000 ){
6346 hideRawTextViewer()
6347 return
6348 }
6349 //gsh_digest_.innerHTML = "";
6350 text = getSourceText() // the original text
6351 tlen = text.length
6352 digest = strCRC32(text,tlen)
6353 //gsh_digest_.innerHTML = digest + " " + tlen
6354 //text = getSourceText() // the text with its digest
6355 Lines = charsIn(text,'\n')
6356
6357 name = "gsh"
6358 sid = name + "-digest"
6359 d = new Date()
6360 signedAt = d.getTime()
6361 sign = '<+' + "span id=" + sid + " data-target-id=" + name + "'
6362 + ' data-crc32u=" + digest + "'
6363 + ' data-length=" + tlen + "'
6364 + ' data-lines=" + Lines + "'
6365 + ' data-time=" + signedAt + "'
6366 + '></span>\n'
6367 text = sign + text
6368
6369 txhtml = '<' + 'table id="LineNumbered"><' + 'tr><' + 'td>'
6370 + '<' + 'textarea cols=5 rows=' + Lines + ' class="LineNumber">'
6371 for( i = 1; i <= Lines; i++){
6372 txhtml += i.toString() + '\n'
6373 }
6374 txhtml += ""

```

```

6375     + '<' + '/textarea>'
6376     + '<' + '/td<' + 'td>'
6377     + '<' + 'textarea cols=150 rows=' + Lines + ' class="LineNumbered">'
6378     + text + '<' + '/textarea>'
6379     + '<' + '/td<' + '/tr><' + '/table>'
6380
6381 for( i = 1; i <= 30; i++ ){
6382     txthtml += '<br>\n'
6383 }
6384 RawTextViewer.innerHTML = txthtml
6385
6386 btn = e
6387 e.style.color = "rgba(128,128,255,0.9)";
6388 y = e.getBoundingClientRect().top.toFixed(0)
6389 //h = e.getBoundingClientRect().height.toFixed(0)
6390 RawTextViewer.style.top = Number(y) + 30
6391 RawTextViewer.style.left = 100;
6392 RawTextViewer.style.height = window.innerHeight - 20;
6393 //RawTextViewer.style.Opacity = 1.0;
6394 //RawTextViewer.style.backgroundColor = "rgba(0,0,0,0.0)";
6395 RawTextViewer.style.backgroundColor = "rgba(255,255,255,0.8)";
6396 RawTextViewer.style.zIndex = 1000;
6397 RawTextViewer.style.display = true;
6398
6399 if( RawTextViewerClose.style == null ){
6400     RawTextViewerClose.style = "";
6401 }
6402 RawTextViewerClose.style.top = Number(y) + 10
6403 RawTextViewerClose.style.left = 100;
6404 RawTextViewerClose.style.zIndex = 1001;
6405
6406 ScrollToElement(CurElement,RawTextViewerClose)
6407 }
6408 function hideRawTextViewer(){
6409     RawTextViewer.style.left = 10000;
6410     RawTextViewer.style.zIndex = -100;
6411     RawTextViewer.style.Opacity = 0.0;
6412     RawTextViewer.style = null
6413     RawTextViewer.innerHTML = "";
6414
6415     GshMenuSign.style.color = "rgba(255,128,128,1.0)";
6416     RawTextViewerClose.style.top = 0;
6417     RawTextViewerClose.style = null
6418 }
6419
6420 // source code viewr
6421 function frame_close(){
6422     srcframe = document.getElementById("src-frame");
6423     srcframe.innterHTML = "";
6424     //srcframe.style.cols = 1;
6425     srcframe.style.rows = 1;
6426     srcframe.style.height = 0;
6427     srcframe.style.display = false;
6428     src = document.getElementById("src-frame-textarea");
6429     src.innterHTML = ""
6430     //src.cols = 0
6431     src.rows = 0
6432     src.display = false
6433     //alert("--closed--")
6434 }
6435 //<!-- | <span onclick="html_view();">Source</span> -->
6436 //<!-- | <span onclick="frame_close();">SourceClose</span> -->
6437 //<!--| <span>Download</span> -->
6438 function frame_open(){
6439     document.getElementById('GshFaviconURL').href = "";
6440     oldsrc = document.getElementById("GENSRC");
6441     if( oldsrc != null ){
6442         //alert("--I--(erasing old text)")
6443         oldsrc.innterHTML = "";
6444         return
6445     }else{
6446         //alert("--I--(no old text)")
6447     }
6448     styleBanner = GshBanner.getAttribute("style")
6449     GshBanner.removeAttribute("style")
6450     styleFooter = GshFooter.getAttribute("style")
6451     GshFooter.removeAttribute("style")
6452
6453     GshFaviconURL.href = "";
6454     GStat.removeAttribute('style')
6455     GshGrid.removeAttribute('style')
6456     GshMenuSign.removeAttribute('style')
6457     GPos.removeAttribute('style')
6458     GPos.innerHTML = "";
6459     GLog.removeAttribute('style')
6460     GLog.innerHTML = "";
6461     GMenu.removeAttribute('style')
6462     GTop.removeAttribute('style')
6463     GShellPlane.removeAttribute('style')
6464     RawTextViewer.removeAttribute('style')
6465     RawTextViewerClose.removeAttribute('style')
6466
6467     src = document.getElementById("gsh");
6468     srcframe = document.getElementById("src-frame");
6469     srcframe.innerHTML = ""
6470     + "<"+cite id="GENSRC">\n"
6471     + "<"+style>\n"
6472     + "#GENSRC textarea{tab-size:4;}\n"
6473     + "#GENSRC textarea(-o-tab-size:4;)\n"
6474     + "#GENSRC textarea(-moz-tab-size:4;)\n"
6475     + "#GENSRC textarea(spellcheck:false;)\n"
6476     + "<"+style>\n"
6477     + "<"+textarea id="src-frame-textarea" cols=100 rows=20 class="gsh-code">'
6478     + /*<+html>\n" // lost preamble text
6479     + "<"+span id="gsh">" // lost preamble text
6480     + src.innerHTML
6481     + "<"+/span>+<+html>\n" // lost trail text
6482     + "<"+textarea>\n"
6483     + "<"+cite>!-- GENSRC -->\n";
6484
6485     //srcframe.style.cols = 80;
6486     //srcframe.style.rows = 80;
6487
6488     GshBanner.setAttribute('style',styleBanner)
6489     GshFooter.setAttribute('style',styleFooter)
6490 }
6491 function fill_CSSView(){
6492     part = document.getElementById('GshStyleDef')
6493     view = document.getElementById('gsh-style-view')
6494     view.innerHTML = ""
6495     + "<"+textarea cols=100 rows=20 class="gsh-code">'
6496     + part.innerHTML
6497     + "<"+/textarea>"
6498 }
6499 function fill_JavaScriptView(){

```

```

6500  jspart = document.getElementById('gsh-script')
6501  view = document.getElementById('gsh-script-view')
6502  view.innerHTML =
6503  + "<"+textarea cols=100 rows=20 class="gsh-code">"
6504  + jspart.innerHTML
6505  + "<"+/textarea>"
6506  }
6507  function fill_DataView(){
6508  part = document.getElementById('gsh-data')
6509  view = document.getElementById('gsh-data-view')
6510  view.innerHTML = ""
6511  + "<"+textarea cols=100 rows=20 class="gsh-code">"
6512  + part.innerHTML
6513  + "<"+/textarea>"
6514  }
6515  function jumpto_StyleView(){
6516  jsview = document.getElementById('html-src')
6517  jsview.open = true
6518  jsview = document.getElementById('gsh-style-frame')
6519  jsview.open = true
6520  fill_CSSView()
6521  }
6522  function jumpto_JavaScriptView(){
6523  jsview = document.getElementById('html-src')
6524  jsview.open = true
6525  jsview = document.getElementById('gsh-script-frame')
6526  jsview.open = true
6527  fill_JavaScriptView()
6528  }
6529  function jumpto_DataView(){
6530  jsview = document.getElementById('html-src')
6531  jsview.open = true
6532  jsview = document.getElementById('gsh-data-frame')
6533  jsview.open = true
6534  fill_DataView()
6535  }
6536  function jumpto_WholeView(){
6537  jsview = document.getElementById('html-src')
6538  jsview.open = true
6539  jsview = document.getElementById('gsh-whole-view')
6540  jsview.open = true
6541  frame_open()
6542  }
6543  function html_view(){
6544  html_stop();
6545  }
6546  banner = document.getElementById('GshBanner').style.backgroundImage;
6547  footer = document.getElementById('GshFooter').style.backgroundImage;
6548  document.getElementById('GshBanner').style.backgroundImage = "";
6549  document.getElementById('GshBanner').style.backgroundPosition = "";
6550  document.getElementById('GshFooter').style.backgroundImage = "";
6551  }
6552  //srcwin = window.open("", "CodeView2", "");
6553  srcwin = window.open("", "", "");
6554  srcwin.document.write("<span id='gsh'\>\n");
6555  }
6556  src = document.getElementById("gsh");
6557  srcwin.document.write("<"+style>\n");
6558  srcwin.document.write("textarea{tab-size:4;}\n");
6559  srcwin.document.write("textarea{-o-tab-size:4;}\n");
6560  srcwin.document.write("textarea{-moz-tab-size:4;}\n");
6561  srcwin.document.write("</style>\n");
6562  srcwin.document.write("<h2>\n");
6563  srcwin.document.write("<"+span onclick='\window.close();'\>Close</span> | \n");
6564  //srcwin.document.write("<"+span onclick='\html_stop();'\>Run</span>\n");
6565  srcwin.document.write("</h2>\n");
6566  srcwin.document.write("<textarea id='gsh-src-src'\ cols=100 rows=60>");
6567  srcwin.document.write("/<"+html>\n");
6568  srcwin.document.write("<"+span id='gsh'\>");
6569  srcwin.document.write(src.innerHTML);
6570  srcwin.document.write("<"+/span><"+/html>\n");
6571  srcwin.document.write("</"+textarea>\n");
6572  }
6573  document.getElementById('GshBanner').style.backgroundImage = banner;
6574  document.getElementById('GshFooter').style.backgroundImage = footer
6575  }
6576  sty = document.getElementById("GshStyleDef");
6577  srcwin.document.write("<"+style>\n");
6578  srcwin.document.write(sty.innerHTML);
6579  srcwin.document.write("<"+/style>\n");
6580  }
6581  run = document.getElementById("gsh-script");
6582  srcwin.document.write("<"+script>\n");
6583  srcwin.document.write(run.innerHTML);
6584  srcwin.document.write("<"+/script>\n");
6585  }
6586  srcwin.document.write("<"+/span><"+/html>\n"); // gsh span
6587  srcwin.document.close();
6588  srcwin.focus();
6589  }
6590  GSH = document.getElementById("gsh")
6591  }
6592  //GSH.onclick = "alert('Ouch!')"
6593  //GSH.css = "{background-color:#eef;}"
6594  //GSH.style = "background-color:#eef;"
6595  //GSH.style.display = false;
6596  //alert('Ouch0!')
6597  //GSH.style.display = true;
6598  }
6599  // 2020-0904 created, tentative
6600  document.addEventListener('keydown', jgshCommand);
6601  //CurElement = GshStatement
6602  CurElement = GshMenu
6603  MemElement = GshMenu
6604  }
6605  function nextSib(e){
6606  n = e.nextSibling;
6607  for( i = 0; i < 100; i++ ){
6608  if( n == null ){
6609  break;
6610  }
6611  if( n.nodeName == "DETAILS" ){
6612  return n;
6613  }
6614  n = n.nextSibling;
6615  }
6616  return null;
6617  }
6618  function prevSib(e){
6619  n = e.previousSibling;
6620  for( i = 0; i < 100; i++ ){
6621  if( n == null ){
6622  break;
6623  }
6624  if( n.nodeName == "DETAILS" ){

```

```

6625         return n;
6626     }
6627     n = n.previousSibling;
6628 }
6629 return null;
6630 }
6631 function setColor(e,eName,eColor){
6632     if( e.hasChildNodes() ){
6633         s = e.childNodes;
6634         if( s != null ){
6635             for( ci = 0; ci < s.length; ci++ ){
6636                 if( s[ci].nodeName == eName ){
6637                     s[ci].style.color = eColor;
6638                     //s[ci].style.backgroundColor = eColor;
6639                     break;
6640                 }
6641             }
6642         }
6643     }
6644 }
6645
6646 // https://docs.microsoft.com/en-us/previous-versions//hh781509(v=vs.85)
6647 function showCurElementPosition(ev){
6648     if( document.getElementById("GPos") == null ){
6649         return;
6650     }
6651     if( GPos == null ){
6652         return;
6653     }
6654     e = CurElement
6655     y = e.getBoundingClientRect().top.toFixed(0)
6656     x = e.getBoundingClientRect().left.toFixed(0)
6657
6658     h = ev + " "
6659     h += 'y='+y+", "+ 'x='+x" -- "
6660     h += "w=" + window.innerWidth + ", h=" + window.innerHeight + " -- "
6661     //GPos.test = h
6662     //GPos.innerHTML = h
6663     GPos.innerHTML = h
6664 }
6665
6666 function DateLong(e){
6667     d = new Date()
6668     return d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
6669         + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
6670         + "." + d.getMilliseconds()
6671         + " " + d.getTimezoneOffset()/60
6672         + " " + d.getTime() + "." + d.getMilliseconds()
6673 }
6674
6675 }
6676 function GShellMenu(e){
6677     GLog.innerHTML = "Hello, World! (" + DateLong() + ")"
6678     showGShellPlane()
6679 }
6680 // placements of planes
6681 function GShellResizeX(ev){
6682     //if( document.getElementById("GMenu") != null ){
6683         GMenu.style.left = window.innerWidth - 100
6684         GMenu.style.top = window.innerHeight - 90
6685         console.log("place GMENU "+GMenu.style.left+" "+GMenu.style.top)
6686     //}
6687
6688     GStat.style.width = window.innerWidth
6689     //if( document.getElementById("GPos") != null ){
6690         GPos.style.width = window.innerWidth
6691         GPos.style.top = window.innerHeight - 30; //GPos.style.height
6692     //}
6693     if( document.getElementById("GLog") != null ){
6694         GLog.style.width = window.innerWidth
6695         GLog.innerHTML = ""
6696     }
6697     if( document.getElementById("GLog") != null ){
6698         //GLog.innerHTML = "Resize: w=" + window.innerWidth +
6699         //", h=" + window.innerHeight
6700     }
6701     showCurElementPosition(ev)
6702 }
6703 function GShellResize(){
6704     GShellResizeX("RESIZE")
6705 }
6706 window.onresize = GShellResize
6707
6708 function ScrollToElement(oe,ne){
6709     ne.scrollIntoView()
6710     ny = ne.getBoundingClientRect().top.toFixed(0)
6711     nx = ne.getBoundingClientRect().left.toFixed(0)
6712     //GLog.innerHTML = "["+ny+", "+nx+"]"
6713     //window.scrollTo(0,0)
6714
6715     GTop.style.backgroundColor = "rgba(0,0,0,0.0)"
6716     GshGrid.style.left = '250px';
6717     GshGrid.style.zIndex = 0
6718     return
6719     oy = oe.getBoundingClientRect().top.toFixed(0)
6720     ox = oe.getBoundingClientRect().left.toFixed(0)
6721     y = e.getBoundingClientRect().top.toFixed(0)
6722     x = e.getBoundingClientRect().left.toFixed(0)
6723     window.scrollTo(x,y)
6724     ny = e.getBoundingClientRect().top.toFixed(0)
6725     nx = e.getBoundingClientRect().left.toFixed(0)
6726     GLog.innerHTML = "["+oy+", "+ox+"]->["+y+", "+x+"]->["+ny+", "+nx+"]"
6727 }
6728 var MyHistory = ""
6729 function showGShellPlane(){
6730     if( GShellPlane.style.zIndex == 0 ){
6731         GShellPlane.style.zIndex = 1000;
6732         GShellPlane.style.left = 30;
6733         GShellPlane.style.height = 320;
6734         GShellPlane.innerHTML = DateLong(null) + "<br>" +
6735         "-- History --<br>" + MyHistory;
6736     }else{
6737         GShellPlane.style.zIndex = 0;
6738         GShellPlane.style.left = 0;
6739         GShellPlane.style.height = 50;
6740         GShellPlane.innerHTML = "";
6741     }
6742 }
6743 function jgshCommand(event){
6744     key = event
6745     keycode = key.code
6746     //GStat.style.width = window.innerWidth
6747     GStat.style.backgroundColor = "rgba(0,0,0,0.4)"
6748
6749     console.log("JSGsh-Key:"+keycode+"(^~/)")

```

```

6750 if( keycode == "Digit0" ){ // fold side-bar
6751 // Zero page
6752 showGShellPlane();
6753 }else
6754 if( keycode == "Digit1" ){ // fold side-bar
6755 primary.style.width = "94%"
6756 secondary.style.width = "0%"
6757 secondary.style.opacity = 0
6758 GStat.innerHTML = "[Single Column View]"
6759 }else
6760 if( keycode == "Digit2" ){ // unfold side-bar
6761 primary.style.width = "58%"
6762 secondary.style.width = "36%"
6763 secondary.style.opacity = 1
6764 GStat.innerHTML = "[Double Column View]"
6765 }else
6766 if( keycode == "KeyU" ){ // fold/unfold all
6767 html_fold(GshMenuFold);
6768 location.href = "#"+CurElement.id;
6769 }else
6770 if( keycode == "KeyO" || keycode == "ArrowRight" ){ // fold the element
6771 CurElement.open = !CurElement.open;
6772 }else
6773 if( keycode == "ArrowRight" ){ // unfold the element
6774 CurElement.open = true
6775 }else
6776 if( keycode == "ArrowLeft" ){ // unfold the element
6777 CurElement.open = false
6778 }else
6779 if( keycode == "KeyI" ){ // inspect the element
6780 e = CurElement
6781 GLog.innerHTML = "Current Element: " + e + "<br>"
6782 + "name="+e.nodeName + ", "
6783 + "id="+e.id + ", "
6784 + "children="+e.childNodes.length + ", "
6785 + "parent="+e.parentNode.id + "<br>"
6786 + "text="+e.textContent
6787 GStat.style.backgroundColor = "rgba(0,0,0,0.8)"
6788 return
6789 }else
6790 if( keycode == "KeyM" ){ // memory the position
6791 MemElement = CurElement
6792 }else
6793 if( keycode == "KeyN" || keycode == "ArrowDown" ){ // next element
6794 e = nextSib(CurElement)
6795 if( e != null ){
6796 setColor(CurElement,"SUMMARY","#fff")
6797 setColor(e,"SUMMARY","#8f8") // should be complement ?
6798 oe = CurElement
6799 CurElement = e
6800 //location.href = "#"+e.id;
6801 ScrollToElement(oe,e)
6802 }
6803 }else
6804 if( keycode == "KeyP" || keycode == "ArrowUp" ){ // previous element
6805 oe = CurElement
6806 e = prevSib(CurElement)
6807 if( e != null ){
6808 setColor(CurElement,"SUMMARY","#fff")
6809 setColor(e,"SUMMARY","#8f8") // should be complement ?
6810 CurElement = e
6811 //location.href = "#"+e.id;
6812 ScrollToElement(oe,e)
6813 }else{
6814 e = document.getElementById("GshBanner")
6815 if( e != null ){
6816 setColor(CurElement,"SUMMARY","#fff")
6817 CurElement = e
6818 ScrollToElement(oe,e)
6819 }else{
6820 e = document.getElementById("primary")
6821 if( e != null ){
6822 setColor(CurElement,"SUMMARY","#fff")
6823 CurElement = e
6824 ScrollToElement(oe,e)
6825 }
6826 }
6827 }
6828 }else
6829 if( keycode == "KeyR" ){
6830 location.reload()
6831 }else
6832 if( keycode == "KeyJ" ){
6833 GshGrid.style.top = '120px';
6834 GshGrid.innerHTML = '(>_<){Down}';
6835 }else
6836 if( keycode == "KeyK" ){
6837 GshGrid.style.top = '0px';
6838 GshGrid.innerHTML = '^_^){Up}';
6839 }else
6840 if( keycode == "KeyH" ){
6841 GshGrid.style.left = '0px';
6842 GshGrid.innerHTML = '(_){Left}';
6843 }else
6844 if( keycode == "KeyL" ){
6845 GLog.innerHTML +=
6846 'screen='+screen.width+'px'+<br>'+
6847 'window='+window.innerWidth+'px'+<br>'+
6848 GshGrid.style.left = (document.documentElement.clientWidth-160).toString(10)+'px';
6849 GshGrid.innerHTML = '@_@){Right}';
6850 }else
6851 if( keycode == "KeyS" ){
6852 html_stop(GshMenuStop,true)
6853 }else
6854 if( keycode == "KeyF" ){
6855 html_fork()
6856 }else
6857 if( keycode == "KeyC" ){
6858 window.close()
6859 }else
6860 if( keycode == "KeyD" ){
6861 html_digest()
6862 }else
6863 if( keycode == "KeyV" ){
6864 e = document.getElementById('gsh-digest')
6865 if( e != null ){
6866 showDigest(e)
6867 }
6868 }
6869 }
6870 showCurElementPosition(["+key.code+" --"]);
6871 if( document.getElementById("GPos") != null ){
6872 //GPos.innerHTML += ["+key.code+" --"]
6873 }
6874 //GShellResizeX(["+key.code+" --"]);

```

```
6875 }
6876 GShellResizeX(["INIT"]);
6877
6878 DisplaySize = "-- Display: "
6879 + 'screen'+screen.width+'px'
6880 + ', '+window+'window.innerWidth+'px';
6881
6882 // 2020-0909 added, permanet local storage
6883 // https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
6884 Permanent = localStorage;
6885 MyHistory = Permanent.getItem('MyHistory')
6886 if( MyHistory == null ){ MyHistory = "" }
6887 d = new Date()
6888 MyHistory = d.getTime()/1000+ " +document.URL+"<br>" + MyHistory
6889 Permanent.setItem('MyHistory',MyHistory)
6890
6891 //Permanent.setItem('MyWindow',window)
6892 let {text, digest} = getDigest()
6893 GLog.innerHTML +=
6894   "-- GShell: " + GshVersion.innerHTML
6895   + "<br>" + "-- Digest: " + digest
6896   //+ "<br>" + DisplaySize
6897   //+ "<br>" + "-- LastVisit:<br>" + MyHistory
6898 GShellResizeX(null);
6899
6900 // <a href="https://www.w3.org/TR/WebCryptoAPI/">Web Cryptography API</a>
6901 //Convert a string into an ArrayBuffer
6902 //from https://developers.google.com/web/updates/2012/06/How-to-convert-ArrayBuffer-to-and-from-String
6903 function str2ab(str) {
6904   const buf = new ArrayBuffer(str.length);
6905   const bufView = new Uint8Array(buf);
6906   for (let i = 0, strLen = str.length; i < strLen; i++) {
6907     bufView[i] = str.charCodeAt(i);
6908   }
6909   return buf;
6910 }
6911 function importPrivateKey(pem) {
6912   const binaryDerString = window.atob(pemContents);
6913   const binaryDer = str2ab(binaryDerString);
6914   return window.crypto.subtle.importKey(
6915     "pkcs8",
6916     binaryDer,
6917     {
6918       name: "RSA-PSS",
6919       modulusLength: 2048,
6920       publicExponent: new Uint8Array([1, 0, 1]),
6921       hash: "SHA-256",
6922     },
6923     true,
6924     ["sign"]
6925   );
6926 }
6927 //importPrivateKey(ppem)
6928
6929 //key = {}
6930 //buf = "abc"
6931 //enc = "xyzxxxxxx"; //crypto.publicEncrypt(key,buf)
6932 //b64 = btoa(enc)
6933 //dec = atob(b64)
6934 //GLog.innerHTML = "enc:" + b64 + ", dec:" + dec
6935
6936 </script>
6937
6938 *//<br></span></html>
6939
```