

```

1 /*<html>
2 <span id="gsh" data-title="GShell" data-author="sato@its-more.jp">
3 <meta charset="UTF-8">
4 <meta name="viewport" content="width=device-width, initial-scale=1.0">
5 <link rel="icon" id="GshFaviconURL" href=""><!-- place holder -->
6 <span hidden="" id="GshVersion" data-title="GShell-0.4.5--2020-09-18--SatoxITS</span>
7 <header id="GshHeader" height="100px" onclick="shiftBG();">
8 <div align="right"><note><a href="http://archive.gshell.org">GShell</a> version 0.4.5 // 2020-09-18 // SatoxITS</note></div>
9 </header>
10 <h2>GShell // a General purpose Shell built on the top of Golang</h2>
11 <p>
12 <note>
13 It is a shell for myself, by myself, of myself. --SatoxITS(^-^)
14 </note>
15 </p>
16 <div id="GJFactory_x"></div>
17 <span id="gsh-Wnid" onclick="win_jump('0.1');">0</span>
18 <span id="GshMenu">
19 <span class="GshMenu1" id="gsh-menu-exit" onclick="html_close();"></span>
20 <span class="GshMenu1" id="gsh-menu-fork" onclick="html_fork();">Fork</span>
21 <span class="GshMenu1" id="gshMenuStop" onclick="html_stop(this,true);">Stop</span>
22 <span class="GshMenu1" id="gshMenuFold" onclick="html_fold(this);">Unfold</span>
23 <span class="GshMenu1" id="gsh-menu-csum" onclick="html_digest();">Digest</span>
24 <span class="GshMenu1" id="gshMenuSign" onclick="html_sign(this);">Source</span>
25 <span class="GshMenu1" id="gsh-menu-pure" onclick="html_pure(this);">Pure</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 0 $ %</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="#findr">findr</a> // builtin find + du
81   <a href="#grep">grep</a> // builtin grep + wc + csum + ...
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 redirection
88   <a href="#history">history</a> // command history
89   <a href="#usage">usage</a> // resource 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]; } pollFd;
149 // int pollx(pollFd *fdv, int nfds, int timeout);
150 // return poll(fdv->fdv,nfds,timeout);
151 //
152 import "C"
153
154 // // 2020-0906 added,
155 func CFpollIn1(fp*os.File, timeoutUs int)(ready uintptr){
156     var fdv = C.pollFd{ }
157     var nfds = 1
158     var timeout = timeoutUs/1000
159
160     fdv.fd[0].fd = C.int(fp.Fd())
161     fdv.fd[0].events = C.POLLIN
162     if( 0 < EventRecvFd ){
163         fdv.fd[1].fd = C.int(EventRecvFd)
164         fdv.fd[1].events = C.POLLIN
165         nfds += 1
166     }
167     r := C.pollx(&fdv,C.int(nfds),C.int(timeout))
168     if( r <= 0 ){
169         return 0
170     }
171     if (int(fdv.fd[1].revents) & int(C.POLLIN)) != 0 {
172         //fprintf(stderr,"--De- got Event\n");
173         return uintptr(EventFdOffset + fdv.fd[1].fd)
174     }
175     if (int(fdv.fd[0].revents) & int(C.POLLIN)) != 0 {
176         return uintptr(NormalFdOffset + fdv.fd[0].fd)
177     }
178     return 0
179 }
180
181 const (
182     NAME = "gsh"
183     VERSION = "0.4.5"
184     DATE = "2020-09-18"
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 // -X 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 ufind
215     Rusageev [2]syscall.Rusage // Resource consumption, CPU time or so
216     Cmdid     int       // maybe with identified with arguments or impact
217     // redirecton 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     RSERV 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 symbols
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             // !N@ .. 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         //@@U Unicode should be cared as a character
480         return str
481     }
482     //rstr := ""
483     inEsc := 0 // escape character 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 = '\n'
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.Println(" ")
555     }else
556     if ! isin("-n",opts) {
557         fmt.Println("\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         ppnthv := strings.Split(cwd,DIRSEP)
583         pathv[0] = strings.Join(ppnthv,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 // lnsp [N] [-crlf][-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-- lnsp %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("\r\n"))
668                 }else{
669                     //os.Stdout.Write([]byte("\r\n"))
670                     os.Stdout.Write([]byte("\\""))
671                     os.Stdout.Write([]byte("\n"))
672                 }
673                 line = line[olen:]
674                 off += olen
675                 remlen -= olen
676             }
677             if strRep { os.Stdout.Write([]byte("\n")) }
678         }
679     }
680     fmt.Fprintf(os.Stderr,"--I-- lnsp %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)xCksum(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-- csum %v (%v)\n", path, err)
751         return -1
752     }
753     defer file.Close()
754     if gsh.CmdTrace { fmt.Printf("--I-- csum %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/BSB_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
806 }
807
808 // <a name="grep">grep</a>
809 // "lines", "lin" or "lnp" for "(text) line processor" or "scanner"
810 // a*,!ab,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         //fmt.Printf("total %d lines %s\n",li,path)
849         //if( 0 < found){ fmt.Printf("(found %d lines %s))\n",found,path); }
850     }
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     MaxNameLen int64 // max. name length
875     nextRepo time.Time
876 }
877 func showFusage(dir string,fusage *fileSum){
878     bsum := 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,bsum);
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, staterr 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 staterr != 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         //fsum.Size += finfo.Size()
921         fsum.Size += fstat.Size()
922         fsum.Blocks += int64(fstat.Blocks)
923         //if verb { fmt.Printf("(%8dBlk) %s",fstat.Blocks/2,path) }
924         if isin("-ls",argv){
925             //if verb { fmt.Printf("%4d %8d ",fstat.Blksize,fstat.Blocks) }
926             //fmt.Printf("%d\t",fstat.Blocks/2)
927         }
928         //if finfo.IsDir()
929         if (fstat.Mode & S_IFMT) == S_IFDIR {
930             fsum.Dirs += 1
931         }
932         //if (finfo.Mode() & os.ModeSymlink) != 0
933         if (fstat.Mode & S_IFMT) == S_IFLINK {
934             //if verb { fmt.Printf("symlink(%v,%s)\n",fstat.Mode,finfo.Name()) }
935             //fmt.Printf("symlink(%o,%s)\n",fstat.Mode,finfo.Name())
936             fsum.Symlink += 1
937         }
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     if isin("-u",argv){
952         sort.Slice(filev, func(i,j int) bool {
953             return 0 < filev[i].AccTime().Sub(filev[j].AccTime())
954         })
955     }
956     if isin("-U",argv){
957         sort.Slice(filev, func(i,j int) bool {
958             return 0 < filev[i].CreatTime().Sub(filev[j].CreatTime())
959         })
960     }
961     */
962     /*
963     if isin("-S",argv){
964         sort.Slice(filev, func(i,j int) bool {
965             return filev[j].Size() < filev[i].Size()
966         })
967     }*/
968     /*
969     for _,filename := range entv {
970         for _,npat := range npatv {
971             match := true
972             if npat == "*" {
973                 match = true
974             }else{
975                 match, _ = filepath.Match(npat,filename)
976             }
977             path := dir + DIRSEP + filename
978             if !match {
979                 continue
980             }
981             var fstat syscall.Stat_t
982             staterr := syscall.Istat(path,&fstat)
983             if staterr != nil {
984                 if !isin("-w",argv){fmt.Printf("ufind: %v\n",staterr) }
985                 continue;
986             }
987             if isin("-du",argv) && (fstat.Mode & S_IFMT) == S_IFDIR {
988                 // should not show size of directory in "-du" mode ...
989             }else{
990                 if !nols && !isin("-s",argv) && (!isin("-du",argv) || isin("-a",argv)) {
991                     if isin("-du",argv) {
992

```

```

993         fmt.Printf("%d\t", fstat.Blocks/2)
994     }
995     showFileInfo(path,argv)
996 }
997 if true { // && isin("-du",argv)
998     total = cumFileInfo(total,path,staterr,fstat,argv,false)
999 }
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 (fstat.Mode & S_IFMT) == S_IFLNK {
1023         continue
1024     }
1025     if dstat.Rdev != fstat.Rdev {
1026         fmt.Printf("--I-- don't follow differnet device %v(%v) %v(%v)\n",
1027             dir,dstat.Rdev,path,fstat.Rdev)
1028     }
1029     if (fstat.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(%d)\n",dir,dirfile,dirfile.Fd())
1042         defer dirfile.Close()
1043     }else{
1044     }
1045     prev := *total
1046     var dstat syscall.Stat_t
1047     staterr := syscall.Lstat(dir,&dstat) // should be fstat
1048
1049     if staterr != nil {
1050         if !isin("-w",argv){ fmt.Printf("ufind: %v\n",staterr) }
1051         return total
1052     }
1053     //filev,err := ioutil.ReadDir(dir)
1054     //_,err := ioutil.ReadDir(dir) // ReadDir() heavy and bad for huge directory
1055     /*
1056     if err != nil {
1057         if !isin("-w",argv){ fmt.Printf("ufind: %v\n",err) }
1058         return total
1059     }
1060     */
1061     if depth == 0 {
1062         depth = cumFileInfo(total,dir,staterr,dstat,argv,true)
1063         if !nols && !isin("-s",argv) && (!isin("-du",argv) || isin("-a",argv)) {
1064             showFileInfo(dir,argv)
1065         }
1066     }
1067     // it is not a directory, just scan it and finish
1068
1069     for ei := 0 ; ei++ {
1070         entv,rderr := dirfile.Readdirnames(8*1024)
1071         if len(entv) == 0 || rderr != nil {
1072             //if rderr != nil { fmt.Printf("[%d] len=%d (%v)\n",ei,len(entv),rderr) }
1073             break
1074         }
1075         if 0 < ei {
1076             fmt.Printf("--I-- xxFind[%d] %d large-dir: %s\n",ei,len(entv),dir)
1077         }
1078         total = gsh.xxFindEnv(depth,total,dir,dstat,ei,entv,npatv,argv)
1079     }
1080     if isin("-du",argv) {
1081         // if in "du" mode
1082         fmt.Printf("%d\t%s\n", (total.Blocks-prev.Blocks)/2,dir)
1083     }
1084 }
1085 }
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                         if isin("-unix",argv) {
1115                             gsh.lastCheckSum.SumType |= SUM_UNIXFILE
1116                             gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32UNIX)

```

```

1117
1118     if isin("-ieee",argv){
1119         gsh.lastCheckSum.SumType |= SUM_CRCIEEE
1120         gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32IEEE)
1121     }
1122     gsh.lastCheckSum.RusgAtStart = Getrusagev()
1123 }
1124 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 & 0xFFFFFFF) >> 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     showUsage("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                 abbtme(nanos),
1197                 abbtme(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 (%s)",i,len(v.Foundfile))
1218             if isin("-ls",argv){
1219                 fmt.Printf("\n")
1220                 for _file := range v.Foundfile {
1221                     fmt.Printf("%s //sub number?\n", _file)
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         }
1242     }
1243     return fname,found
1244 }
```

```

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 command [-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 func stripLeadingWSParg(argv[]string)([]string){
1309     for ; 0 < len(argv); {
1310         if len(argv[0]) == 0 {
1311             argv = argv[1:]
1312         }else{
1313             break
1314         }
1315     }
1316     return argv
1317 }

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.Println("\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     return false, 0
1350 }

1351 }

1352 func echo(argv []string, nlend bool){
1353     for ai := 1; ai < len(argv); ai++ {
1354         if 1 < ai {
1355             fmt.Printf(" ");
1356         }
1357         arg := argv[ai]
1358         found, val := getval(arg)
1359         if found {
1360             fmt.Printf("%d",val)
1361         }else{
1362             fmt.Printf("%s",arg)
1363         }
1364     }

```

```

1365     if nlen {
1366         fmt.Printf("\n")
1367     }
1368 }
1369
1370 func resfile() string {
1371     return "gsh.tmp"
1372 }
1373 //var resF *File
1374 func remap() {
1375     _, err := os.OpenFile(resfile(), os.O_RDWR|os.O_CREATE, os.ModeAppend)
1376     // https://developpaper.com/solution-to-golang-bad-file-descriptor-problem/
1377     if err != nil {
1378         fmt.Printf("refF could not open: %s\n",err)
1379     }else{
1380         fmt.Printf("refF opened\n")
1381     }
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,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     if sb < si {
1440         argv = append(argv,string(argl[0:ax]))
1441         if debug {
1442             fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
1443                 -1+len(argv),sb,si,string(argl[0:ax]),string(str[si:]))
1444         }
1445     }
1446     if debug {
1447         fmt.Printf("--Da- %d [%s] => [%d]%v\n",strip,str,len(argv),argv)
1448     }
1449     return argv
1450 }
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 = [1]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     fdi := 0
1464     dir := "?"
1465     if mode == "r" {
1466         dir = "<"
1467         fdi = 1 // read from the stdout of the process
1468     }else{
1469         dir = ">"
1470         fdi = 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("--RE01 [%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     }

```

```

1489     }
1490     savi := os.Stdin
1491     save := os.Stdout
1492     save := os.Stderr
1493     os.Stdin = pin
1494     os.Stdout = pout
1495     os.Stderr = pout
1496     gsh.BackGround = true
1497     gsh.gshellh(name)
1498     gsh.BackGround = false
1499     os.Stdin = savi
1500     os.Stdout = save
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 := fullpath[0]
1517     argv = unescapeWhiteSP(argv)
1518     if 0 < strings.Index(fullpath,".go") {
1519         argv := 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 := gofullpath[0]
1526         argv = []string{gofullpath, "run", fullpath }
1527         fmt.Printf("--I-- %s (%s %s)%n",gofullpath,
1528             argv[0],argv[1],argv[2])
1529         if exec {
1530             syscall.Exec(gofullpath,argv,os.Environ())
1531         }else{
1532             pid, _ := syscall.ForkExec(gofullpath,argv,&gshPA)
1533             if gsh.BackGround {
1534                 fmt.Fprintf(stderr,"--Ip- in Background pid(%d)%d(%v)%n",pid,len(argv),argv)
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.Println("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.Println("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)/1000000)
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 abssize(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 absize(size int64)(string){
1657     fsize := float64(size)
1658     if 1024*1024*1024 < size {
1659         return fmt.Sprintf("%8.2fGiB",fsize/(1024*1024*1024))
1660     }else
1661     if 1024*1024 < size {
1662         return fmt.Sprintf("%8.3fMiB",fsize/(1024*1024))
1663     }else{
1664         return fmt.Sprintf("%8.3fKiB",fsize/1024)
1665     }
1666 }
1667 func abbspeed(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 absspeed(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,absize(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,absize(total),size,nio,
1711             absspeed((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         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     nio += 1
1752     total += int64(rlen)
1753     rem -= int64(rlen)
1754     if rem < 0 {
1755         break
1756     }
1757 }
1758 Done := time.Now()
1759 Elps := float64(Done.Sub(Start))/1000000000 //Seconds
1760 TotalMB := float64(total)/1000000
1761 MBps := TotalMB / Elps
1762 fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v) %v %.3fMB/s\n",
1763                         what,total,size,nio,absize(total),MBps)
1764 return total
1765
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%s",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 "HELP":
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                             dszie = 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                             dszie = fi.Size()
1861                         }
1862                     }
1863                 }
1864             }
1865         }
1866     }
1867 }
```

```

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
1876     tcpPush(clnt); // to let SoftEOF data appear at the top of received data
1877     fmt.Fprintf(clnt,"%v\r\n",SoftEOF)
1878     tcpPush(clnt); // to let SoftEOF alone in a packet (separate with 200 OK)
1879     // with client generated random?
1880     //fmt.Printf("--In- L: close %v (%v)\n",in.Fd(),in.Name())
1881 }
1882 res = fmt.Sprintf("200 GET done\r\n")
1883 case "PUT":
1884     // upload {srcfile|-zN} [dstfile]
1885     var dsize int64 = 32*1024*1024
1886     var bsize int = 64*1024
1887     var fname string = ""
1888     var out *os.File = nil
1889     if 1 < len(cmdv) { // localfile
1890         fmt.Sscanf(cmdv[1],"%d",&dsiz
1891     }
1892     if 2 < len(cmdv) {
1893         fname = cmdv[2]
1894         if fname == "-" {
1895             // nul dev
1896         }else
1897             if strBegins(fname,"(") {
1898                 xin,xout,err := gsh.Popen(fname,"w")
1899                 if err {
1900                     }else{
1901                         xin.Close()
1902                         defer xout.Close()
1903                         out = xout
1904                     }
1905                 }else{
1906                     // should write to temporary file
1907                     // should suppress ^C on tty
1908                     xout,err := os.OpenFile(fname,os.O_CREATE|os.O_RDWR|os.O_TRUNC,0600)
1909                     //fmt.Printf("--In- S: open(%v) out(%v) err(%v)\n",fname,xout,err)
1910                     if err != nil {
1911                         fmt.Printf("--En- PUT (%v)\n",err)
1912                     }else{
1913                         out = xout
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",dsize,bsize)
1920             fmt.Printf(Elapsed(Start)+"--In- S: 200 %v OK\r\n",dsize)
1921             fmt.Fprintf(clnt,"200 %v OK\r\n",dsize)
1922             fileRelay("RecvPUT",clnt,out,dsize,bsize)
1923             res = fmt.Sprintf("200 PUT done\r\n")
1924         default:
1925             res = fmt.Sprintf("400 What? %v",req)
1926     }
1927     swcc,err := clnt.Write([]byte(res))
1928     if err != nil {
1929         fmt.Printf(Elapsed(Start)+"--In- S: (wc=%v er=%v) %v",swcc,err,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,LINE_SIZE)
1976     count,err := serv.Read(res)
1977     if err != nil {
1978         fmt.Printf("--En- S: (%d,%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.Println(Blapsed(Start)+"--In- C: %v",req)
1987 fmt.Fprintf(serv,req)
1988 count,err = serv.Read(res)
1989 if err != nil {
1990 }else{
1991     var dsize 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,&dsize)
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.Println("--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 dsiz int64 = 32*1024*1024
2036         var bsiz 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",&dsiz)
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                             dsiz = MaxStreamSize
2056                         }
2057                     }else{
2058                         xlocal,err := os.Open(fname)
2059                         if err != nil {
2060                             fmt.Println("--En- (%s)\n",err)
2061                             local = nil
2062                         }else{
2063                             local = xlocal
2064                             fi,_ := local.Stat()
2065                             dsiz = fi.Size()
2066                             defer local.Close()
2067                             //fmt.Printf("--I-- Rex in(%v / %v)\n",ofile,dsiz)
2068                         }
2069                     ofile = fname
2070                     fmt.Printf(Elapsed(Start)+"--In- L: open(%v,r)=%v %v (%v)\n",
2071                         fname,dsiz,local,err)
2072                 }
2073         }
2074         if 2 < len(argv) && argv[2] != "" {
2075             ofile = argv[2]
2076             //fmt.Printf("(%)v B.ofile=%v\n",len(argv),argv,ofile)
2077         }
2078         //fmt.Printf(Elapsed(Start)+"--I-- Rex out(%v)\n",ofile)
2079         fmt.Println(Elapsed(Start)+"--In- PUT %v (%v)\n",dsiz,bsize)
2080         req = fmt.Sprintf("PUT %v %v (%v)",dsiz,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,dsiz,bsize)
2086     }else{
2087         req = fmt.Sprintf("%v\r\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)\r\n"
2097     }else{
2098         ress = string(res[:count])
2099     }
2100     if err != nil {
2101         fmt.Println(Elapsed(Start)+"--En- S: (%d,%v) %v",count,err,ress)
2102     }else{
2103         fmt.Println(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(res,"%d %s",rcode,stat)
2111     //fmt.Printf("--D-- Client: %v (%v)",rcode,stat)
2112     return rcode,res
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
2127     for _v := range argv {
2128         /*
2129             if v[0] == '-' { // might be a pseudo file (generated date)
2130                 continue
2131             }
2132             */
2133             obj := strings.Split(v,":")
2134             //fmt.Printf("%d %v %v\n",len(obj),v,obj)
2135             if 1 < len(obj) {
2136                 host = obj[0]
2137                 file := ""
2138                 if 0 < len(host) {
2139                     gsh.LastServer.host = host
2140                 }else{
2141                     host = gsh.LastServer.host
2142                     port = gsh.LastServer.port
2143                 }
2144                 if 2 < len(obj) {
2145                     port = obj[1]
2146                     if 0 < len(port) {
2147                         gsh.LastServer.port = port
2148                     }else{
2149                         port = gsh.LastServer.port
2150                     }
2151                     file = obj[2]
2152                 }else{
2153                     file = obj[1]
2154                 }
2155                 if len(srcv) == 0 {
2156                     download = true
2157                     srcv = append(srcv,file)
2158                     continue
2159                 }
2160                 upload = true
2161                 dstv = append(dstv,file)
2162                 continue
2163             }
2164             /*
2165             idx := strings.Index(v,":")
2166             if 0 << idx {
2167                 remote = v[0:idx]
2168                 if len(srcv) == 0 {
2169                     download = true
2170                     srcv = append(srcv,v[idx+1:])
2171                     continue
2172                 }
2173                 upload = true
2174                 dstv = append(dstv,v[idx+1:])
2175                 continue
2176             }
2177             */
2178             if download {
2179                 dstv = append(dstv,v)
2180             }else{
2181                 srcv = append(srcv,v)
2182             }
2183         }
2184         hostport := "@" + host + ":" + port
2185         if upload {
2186             if host != "" { xargv = append(xargv,hostport) }
2187             xargv = append(xargv,"PUT")
2188             xargv = append(xargv,srcv[0]...)
2189             xargv = append(xargv,dstv[0]...)
2190             //fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v // %v\n",hostport,dstv,srcv,xargv)
2191             fmt.Println("--I-- FileCopy PUT gsh://%s/%v < %v\n",hostport,dstv,srcv)
2192             gsh.RexecClient(xargv)
2193         }else{
2194             if download {
2195                 if host != "" { xargv = append(xargv,hostport) }
2196                 xargv = append(xargv,"GET")
2197                 xargv = append(xargv,srcv[0]...)
2198                 xargv = append(xargv,dstv[0]...)
2199             //fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v // %v\n",hostport,srcv,dstv,xargv)
2200             fmt.Println("--I-- FileCopy GET gsh://%v/%v > %v\n",hostport,srcv,dstv)
2201             gsh.RexecClient(xargv)
2202         }else{
2203     }
2204 }
2205
2206 // target
2207 func (gsh*GshContext)Treelpath(rloc string)(string){
2208     cwd, _ := os.Getwd()
2209     os.Chdir(gsh.RWD)
2210     os.Chdir(rloc)
2211     twd, _ := os.Getwd()
2212     os.Chdir(cwd)
2213
2214     tpath := twd + "/" + rloc
2215     return tpath
2216 }
2217 // join to remote GShell - [user@]host:[port] or cd host:[port]:path
2218 func (gsh*GshContext)Rjoin(argv[]string){
2219     if len(argv) << 1 {
2220         fmt.Printf("--I-- current server = %v\n",gsh.RSERV)
2221         return
2222     }
2223     serv := argv[1]
2224     servv := strings.Split(serv,":")
2225     if 1 << len(servv) {
2226         if servv[0] == "lo" {
2227             servv[0] = "localhost"
2228         }
2229     }
2230     switch len(servv) {
2231     case 1:
2232         //if strings.Index(serv,":") < 0 {

```

```

2233     serv = servv[0] + ":" + fmt.Sprintf("%d",GSH_PORT)
2234     //}
2235     case 2: // host:port
2236     serv = strings.Join(servv,":")
2237   }
2238 xargv := []string{"rex-join","@"+serv,"HELO"}
2239 rcode,stat := gsh.RexecClient(xargv)
2240 if (rcode / 100) == 2 {
2241   fmt.Printf("--I-- OK Joined (%v) %v\n",rcode,stat)
2242   gsh.RSERV = serv
2243 }else{
2244   fmt.Printf("--I-- NG, could not joined (%v) %v\n",rcode,stat)
2245 }
2246 }
2247 func (gsh*GshContext)Rexec(argv[]string){
2248 if len(argv) <= 1 {
2249   fmt.Printf("--I-- rexec command [ | {file || {command} ]\n",gsh.RSERV)
2250   return
2251 }
2252 /*
2253 nargv := gshScanArg(strings.Join(argv," "),0)
2254 fmt.Printf("--D-- nargc=%d [%v]\n",len(nargv),nargv)
2255 if nargv[1][0] != '{' {
2256   nargv[1] = "(" + nargv[1] + ")"
2257   fmt.Printf("--D-- nargc=%d [%v]\n",len(nargv),nargv)
2258 }
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.Println("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.Trepath(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.Trepath(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 if inTCP { // TCP
2335   dport, err := net.ResolveTCPAddr("tcp",remote);
2336   if err != nil {
2337     fmt.Printf("Address error: %s (%s)\n",remote,err)
2338   }
2339   Conn, err := net.DialTCP("tcp",nil,dport)
2340   if err != nil {
2341     fmt.Printf("Connection error: %s (%s)\n",remote,err)
2342   }
2343   file, _ := conn.File();
2344   fd := file.Fd()
2345   fmt.Printf("Socket: connected to %s, socket[%d]\n",remote,fd)
2346   savfd := gshPA.Files[1]
2347   gshPA.Files[1] = fd;
2348   gshCtx.gshelly(argv[2:])
2349   gshPA.Files[1] = savfd
2350   file.Close()
2351   conn.Close()
2352 }else{
2353 }
```

```

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.gshellv(argv[2:])
2380 gshPA.Files[1] = savfd
2381 file.Close()
2382 conn.Close()
2383 }
2384 }
2385 func (gshCtx*GshContext)accept(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.Println("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.Println("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.gshellv(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.Println("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.Println("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.gshellv(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 emtpy command
2473         t := time.Now()
2474         date := t.Format(time.UnixDate)
2475         exe, _ := os.Executable()
2476         host, _ := os.Hostname()
2477         fmt.Printf("{PWD=%s\n", cwd)
2478         fmt.Printf("HOST=%s\n", host)
2479         fmt.Printf("DATE=%s\n", date)
2480         fmt.Printf("TIME=%s\n", t.String())
2481     }
2482 }
```

```

2481     fmt.Printf(" PID=%d\\",os.Getpid())
2482     fmt.Printf(" EXE=\"%s\\",exe)
2483     fmt.Printf("}\\n")
2484 }
2485 }
2486 // <a name="history">History</a>
2487 // these should be browsed and edited by HTTP browser
2488 // show the time of command with -t and direcotry with -ls
2489 // openfile-history, sort by -a -m -c
2490 // sort by elapsed time by -t -s
2491 // search by "more" like interface
2492 // edit history
2493 // sort history, and wc or uniq
2494 // CPU and other resource consumptions
2495 // limit showing range (by time or so)
2496 // export / import history
2497 func (gshCtx *GshContext)xHistory(argv []string){
2498     atWorkDirX := -1
2499     if i < len(argv) && strBegins(argv[1],"@") {
2500         atWorkDirX,_ = strconv.Atoi(argv[1][1:])
2501     }
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("-lo",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 %1v/t ",start,elps)
2526         }
2527         if !isin("-l",argv) && !isin("-lo",argv){
2528             fmt.Printf("%v",Rusagef("%t %u/t// %s",argv,v.Rusage))
2529         }
2530         if !isin("-at",argv) { // isin("ls",argv){
2531             dhi := v.WorkDirX // workdir history index
2532             fmt.Printf("%d %v",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 // In - history index
2541 func searchHistory(gshCtx GshContext, gline string) (string, bool, bool){
2542     if gline[0] == '!' {
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 GChdirHistory, 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.ShowChdirHistory(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     chhist := 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 == "@!" {
2613     index := len(cdhist) - 1
2614     if 0 < index { index -= 1 }
2615     dir = cdhist[index].Dir
2616   }else{
2617     index, err := strconv.Atoi(argv[1:])
2618     if err != nil {
2619       fmt.Printf("--E-- xChdir(%v)\n",err)
2620       dir = "?"
2621     }else
2622     if len(gshCtx.CkdirHistory) <= 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.Ckdir(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.CkdirHistory = append(cdhist,hist1)
2642       if !isin("-s",argv){
2643         cwd, _ := os.Getwd()
2644         //fmt.Printf("%s\n", cwd)
2645         ix := len(gshCtx.CkdirHistory)-1
2646         gshCtx.ShowCkdirHistory1(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(rul, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
2660   TimeValSub(&rul[0].Utime,&ru2[0].Utime)
2661   TimeValSub(&rul[0].Stime,&ru2[0].Stime)
2662   TimeValSub(&rul[1].Utime,&ru2[1].Utime)
2663   TimeValSub(&rul[1].Stime,&ru2[1].Stime)
2664   return rul
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(rul, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
2672   TimeValAdd(rul[0].Utime,ru2[0].Utime)
2673   TimeValAdd(rul[0].Stime,ru2[0].Stime)
2674   TimeValAdd(rul[1].Utime,ru2[1].Utime)
2675   TimeValAdd(rul[1].Stime,ru2[1].Stime)
2676   return rul
2677 }
2678 */
2679 // <a name="rusage">Resource Usage</a>
2680 func sRusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2681   // ru[0] self , ru[1] children
2682   ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
2683   st := TimeValAdd(ru[0].Stime,ru[1].Stime)
2684   uu := (ut.Sec*1000000 + int64(ut.Usec)) * 1000
2685   su := (st.Sec*1000000 + int64(st.Usec)) * 1000
2686   tu := uu + su
2687   ret := fmt.Sprintf("sv/sum",abbttime(tu))
2688   ret += fmt.Sprintf(", sv/usr",abbttime(uu))
2689   ret += fmt.Sprintf(", sv/sys",abbttime(su))
2690   return ret
2691 }
2692 func Rusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2693   ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
2694   st := TimeValAdd(ru[0].Stime,ru[1].Stime)
2695   fmt.Printf("%.06ds/u %.06ds,%.06ds,%.06ds,%.06ds,%.06ds\n",ut.Sec,ut.Usec,ru[1].Utime.Sec,ru[1].Utime.Usec)
2696   fmt.Printf("%.06ds/s %.06ds,%.06ds,%.06ds,%.06ds,%.06ds\n",st.Sec,st.Usec,ru[1].Stime.Sec,ru[1].Stime.Usec)
2697   return ""
2698 }
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   }
2705   return ruv
2706 }
2707 func showRusage(what string,argv []string, ru *syscall.Rusage){
2708   fmt.Println("@" : ",what");
2709   fmt.Printf("User=%d.%06ds",ru.Utime.Sec,ru.Utime.Usec)
2710   fmt.Printf(" Sys=%d.%06ds,ru.Stime.Sec,ru.Stime.Usec)
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.Outblock)
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.Println("\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.CmdFrace { fmt.Println("--I-- inBackground(%v)\n",argv) }
2764   gsh.BackGround = true // set background option
2765   xfin := false
2766   xfin = gsh.gshellv(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   ffdix := 0
2793   mode := os.O_RDONLY
2794
2795   switch {
2796   case cmd == "-i" || cmd == "<":
2797     ffdix = 0
2798     mode = os.O_RDONLY
2799   case cmd == "-o" || cmd == ">":
2800     ffdix = 1
2801     mode = os.O_RDWR | os.O_CREATE
2802   case cmd == "-a" || cmd == ">>":
2803     ffdix = 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[ffdix]
2823   gshPA.Files[ffdix] = file.Fd()
2824   fmt.Printf("--I-- Opened (%d) %s\n",file.Fd(),argv[1])
2825   gshCtx.gshellv(argv[2:])
2826   gshPA.Files[ffdix] = 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.gshellv(argv[1:]);
2851 }
2852 func (gshCtx *GshContext) xPs(argv[]string)(){

```

```

2853 }
2854
2855 // <a name="plugin">Plugin</a>
2856 // plugin [-s [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 scanf(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+lend(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     }
2962     //fmt.Printf("--D-- n=%d err=%v) {s)=%v\n",n,err,vstr, ival)
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                     gsh.iFormat = curfmt
2987                     xi += 1
2988                 if xi+1 < len(fmts) && fmts[xi+1] == '(' {
2989                     vals,leng := scanUntil(fmts[xi+2:],")")
2990                     //fmt.Printf("--D-- show fmt(%v) val(%v) next(%v)\n",curfmt,vals,leng)
2991                     gsh.printVal(curfmt,vals,optv)
2992                     xi += 2+leng-1
2993                     outlen += 1
2994                 }
2995                 continue
2996             }
2997             if fch == '_' {
2998                 hi,leng := scanInt(fmts[xi+1:])
2999                 if 0 < leng {
3000                     if hi < len(gsh.iValues) {
3001                         gsh.printVal(curfmt,gsh.iValues[hi],optv)
3002                         outlen += 1 // should be the real length
3003                     }else{
3004                         fmt.Printf("((out-range))")
3005                     }
3006                     xi += leng
3007                     continue;
3008                 }
3009             }
3010             fmt.Printf("%c",fch)
3011             outlen += 1
3012         }
3013     }else{
3014         //fmt.Printf("--D-- print %s\n")
3015         for i,v := range list {
3016             if 0 < i {
3017                 fmt.Printf(div)
3018             }
3019             gsh.printVal(curfmt,v,optv)
3020             outlen += 1
3021         }
3022     }
3023     if 0 < outlen {
3024         fmt.Println("\n")
3025     }
3026 }
3027 }
3028 func (gsh*GshContext)Scav(argv[]string){
3029     //fmt.Printf("--D-- Scav(%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-- Scanf ->(%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     }
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 optv := []string{}
3069 for _,v := range argv {
3070     if strBegins(v,"-"){
3071         optv = append(optv,v)
3072         argv = argv[1:]
3073     }else{
3074         break;
3075     }
3076 }
3077 if 0 < len(argv) {
3078     fmts = strings.Join(argv," ")
3079 }
3080 gsh.printfv(fmts,div,argv,optv,gsh.iValues)
3081 }
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
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3797
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3989
3990
3991
3992
3993
3994
3995
3996
3997
3997
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4088
4089
4089
4090
4091
4092
4093
4094
4095
4096
4096
4097
4098
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4196
4197
4198
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4297
4298
4299
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4396
4397
4398
4398
4399
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4496
4497
4498
4498
4499
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4598
4599
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4698
4699
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4789
4790
4791
4792
4793
4794
4795
4796
4797
4797
4798
4799
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
48
```

```

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     argv := []string{}
3144     for ai := 0; ai < len(argv); ai++ {
3145         argv = append(argv,strsubst(gshCtx,argv[ai],false))
3146     }
3147     argv = xargv
3148     if false {
3149         for ai := 0; ai < len(argv); ai++ {
3150             fmt.Printf("%[1]d %[2]s %[3]d%T\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)%v\n",len(argv),argv) }
3156     switch { // https://tour.golang.org/flowcontrol/11
3157     case cmd == "":
3158         gshCtx.xPwd([]string{}); // emtpy 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 == "-in":
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 == "cp":
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 == "fdls":
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 == "dseq":
3237         gshCtx.Dseq(argv)
3238     case cmd == "geng":
3239         gshCtx.Eng(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 == "lisp" || cmd == "nlisp":
3249         gshCtx.SplitLine(argv)
3250     case cmd == "ls":
3251         gshCtx.xFind(argv)
3252     case cmd == "nop":
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 == "pstitle":
3264         // to be gsh.title
3265     case cmd == "rexecd" || cmd == "rexrd":
3266         gshCtx.RexecServer(argv)
3267     case cmd == "rexec" || cmd == "rex":
3268         gshCtx.RexecClient(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.Scancv(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.Rexec(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 1 < len(argv) {
3306             ungets(argv[1]+"\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,[]string{"-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.gshell(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%100000000)
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     else
3393     if( with_exgetline && gshCtx.GetLine != "" ){
3394         //var xhix int64 = int64(hix); // cast
3395         newenv := os.Environ()
3396         newenv = append(newenv, "GSH_LINENO="+strconv.FormatInt(int64(hix),10) )
3397
3398         tty := gshCtx.ttyline()
3399         tty.WriteString(prevline)
3400         Pa := os.ProcAttr {
3401             "" // start dir
3402             newenv, //os.Environ(),
3403             []*os.File{os.Stdin,os.Stdout,os.Stderr,tty},
3404             nil,
3405         }
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 }
3431 */
3432 {
3433     // if isatty {
3434         fmt.Printf("!%d",hix)
3435         fmt.Println(PROMPT)
3436     //}
3437     reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
3438     line, _, _ := reader.ReadLine()
3439     return string(line)
3440 }
3441 }
3442 /* begin ===== getline
3443 */
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/">bytes</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_FPL *os.File = nil
3472 var NULL_SPL = 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.Println("ERR:%s\n",serr.String())
3503     }else{
3504         fmt.Println("%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 strlen(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:i])
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 // dictionary 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 RomKana []RomKana
3661
3662 func isinDic(str string)(int){
3663     for i,v := range RomKana {
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 = "s"
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 %v [dicName] [dicURL] -- Import dictionary\n",cmd,DIC_COM_LOAD)
3684     fprintf(out,... %v %v [pattern] -- Search in dictionary\n",cmd,DIC_COM_DUMP)
3685     fprintf(out,... %v %v [dicName] -- List dictionaries\n",cmd,DIC_COM_LIST)
3686     fprintf(out,... %v %v [dicName] -- Disable dictionaries\n",cmd,DIC_COM_DIS)
3687     fprintf(out,... %v %v [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 dicName := ""
3729 dicURL := ""
3730 if 0 < len(argv) {
3731     arg = argv[0]
3732     dicName = arg
3733     argv = argv[1:]
3734 }
3735 if 0 < len(argv) {
3736     dicURL = argv[0]
3737     argv = argv[1:]
3738 }
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",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     envt := 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 envt {
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]%v\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 concrete 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("\\\\$v\\t%v [%2v]-%v [%2v]\\$v\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-- Conguratuations!! 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 = "MyIMB-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_FPP ){
3891         if( true ){
3892             fprintf(stderr,"--$-- Could not load %s\n",MyIMEVER,dic);
3893         }
3894         return -1;
3895     }
3896     if( true ){
3897         var di int;
3898         var line = make(StrBuff,1024);
3899         var pat string
3900         var out string
3901         for di = 0; di < 1024; di++ {
3902             if( fgets(line,sizeof(line),rk) == NULLSP ){
3903                 break;
3904             }
3905             fmt.Sscanf(string(line[0:strlen(line)]),"s %s",&pat,&out);
3906             //sscanf(line,"%[^\\r\\n]",&pat,&out);
3907             romkana[di].pat = pat;
3908             romkana[di].out = out;
3909             //fprintf(stderr,"--Dd- %-10s %s\n",pat,out)
3910         }
3911         dicents += di;
3912         if( false ){
3913             fprintf(stderr,"--$-- 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 si = 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 }
```

```

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 < dicnts; 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     xputss(dst,stderr);
4002     return 0;
4003 }
4004
4005 //----- LINEEDIT
4006 // "?" at the top of the line means searching history
4007
4008 // should be compatilbe 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#L340
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](%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\n",a[2:],tempo);
4090         }else{
4091             if strBegins(a,"r") { // repeat
4092                 fmt.Sscanf(a[1],"%f",&repeat)

```

```

4093     }else
4094     if strBegins(a,"!") {
4095         fmt.Sscanf(a[1:], "%d", &hix)
4096     }else{
4097         fmt.Sscanf(a, "%d", &hix)
4098     }
4099     if hix == 0 || len(argv) <= 1 {
4100         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 FpollInl(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 := FpollInl(fp,usec)
4160     readyFd := CFpollInl(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     _f_ = 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 i,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] %v %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(argv[]string){
4271     hix := 0
4272     if 1 < len(argv) {
4273         fmt.Sscanf(argv[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    // shown translation mode, the mode to be retained
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_ViInsert   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             }
4342         }
4343     }
4344 }
```

```

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]\n",ch1,ch2,ch3,ch4);
4358                             ch1 = DEL_RIGHT
4359                         }
4360                     case '\\':
4361                         //ch4 := fgetcTimeout(stdin,EscTimeout);
4362                         //fprintf(stderr,"y[%02X %02X %02X]\n",ch1,ch2,ch3,ch4);
4363                         switch( ch3 ){
4364                             case '~': ch1 = DEL_RIGHT
4365                         }
4366                     }
4367                 }
4368             }
4369         }
4370     }
4371     return ch1
4372 }
4373 func (iin*IInput)clearline(){
4374     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         }
4466     }

```

```

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(iin 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,INESIZE,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 = ttynname(0);
4508     //fprintf(stderr,"--pts[0] = %s\\n",pts?pts:"?");
4509   }
4510   if( false ){
4511     fprintf(stderr,"! ");
4512     fflush(stderr);
4513     sfgets(&iin.line,INESIZE,stdin);
4514     return iin.line
4515   }
4516   system("/bin/stty -echo -icanon");
4517   xline := iin.xgetlinel(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_Vtrace {
4524     fprintf(stderr,"%v\\r\\n",string(cmdch));
4525   }else{
4526     if( cmdch == 'J' ){
4527       fprintf(stderr,"J\\r\\n");
4528       iin.indMode = 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(in.line+iin.right);
4539     iin.line = dst
4540     iin.right = ""
4541     if( cmdch == 'I' ){
4542       fprintf(stderr,"I\\r\\n");
4543       iin.indMode = true
4544     }
4545     iin.Redraw();
4546   }
4547 // aa 12 ala1
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]\n",i0,i,found,iin.line,iin.right)
4610   //fmt.Println("") // 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]\n",i,iin.line,iin.right)
4666   //fmt.Println("") // set debug messae at the end of line
4667 }
4668 // 0.2.8 2020-0901 created
4669 func (iin*IInput)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]\n",i,iin.line,iin.right)
4700   //fmt.Println("") // set debug messae at the end of line
4701 }
4702 // 0.2.8 2020-0902 created
4703 func (iin*IInput)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*IInput)SearchForward(pat string)(bool){
4737     right := iin.right
4738     found := -1
4739     i := 0
4740     if strBegins(right,pat) {
4741         _,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,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]\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]\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)getStringl(prompt string)(string){ // should be editable
4839     iin.clearline();
4840     fprintf(stderr, "\r\n", 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.indMode = false
4869
4870         ch = iin.Getc(1000*1000)
4871
4872         if ch != EV_TIMEOUT && first {
4873             first = false
4874             mode := 0
4875             if romkanemode {
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                     ch = GO_RIGHT
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.getStringl("//") // should be editable
4934                     SearchForw = true
4935                     iin.GotoFORWSTR(SearchPat,gsh)
4936                     iin.Redraw();
4937                     continue
4938                 case '?':
4939                     SearchPat = iin.getStringl("?) // 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     }
4963     switch ch {
4964         case GO_TOPW:
4965             iin.GotoTOPW();
4966             iin.Redraw();
4967             continue;
4968         case GO_ENDW:
4969             iin.GotoENDW();
4970             iin.Redraw();
4971             continue;
4972         case GO_NEXTW:
4973             // to next space then
4974             iin.GotoNEXTW();
4975             iin.Redraw();
4976             continue;
4977         case GO_PAIRCH:
4978             iin.GotoPAIRCH();
4979             iin.Redraw();
4980             continue;
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 = fgettimeout(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         }/*
5000         if( ch == 'm' || ch == 'M' ){
5001             mch := fgettimeout(stdin,1000*1000)
5002             if mch == 'r' {
5003                 romkanmode = true
5004             }else{
5005                 romkanmode = false
5006             }
5007             continue
5008         }else
5009     */
5010     if( ch == 'k' || ch == 'K' ){
5011         MODE_Recursive = !MODE_Recursive
5012         iin.Translate(cmdch);
5013         continue
5014     }else
5015     if( ch == 'j' || ch == 'J' ){
5016         iin.Translate(cmdch);
5017         continue
5018     }else
5019     if( ch == 'i' || ch == 'I' ){
5020         iin.Replace(cmdch);
5021         continue
5022     }else
5023     if( ch == 'l' || ch == 'L' ){
5024         MODE_LowerLock = !MODE_LowerLock
5025         MODE_CapsLock = false
5026         if MODE_ViTrace {
5027             fprintf(stderr,"%v\r\n",string(cmdch));
5028         }
5029         iin.Redraw();
5030         continue
5031     }else
5032     if( ch == 'u' || ch == 'U' ){
5033         MODE_CapsLock = !MODE_CapsLock
5034         MODE_LowerLock = false
5035         if MODE_ViTrace {
5036             fprintf(stderr,"%v\r\n",string(cmdch));
5037         }
5038         iin.Redraw();
5039         continue
5040     }else
5041     if( ch == 'v' || ch == 'V' ){
5042         MODE_ViTrace = !MODE_ViTrace
5043         if MODE_ViTrace {
5044             fprintf(stderr,"%v\r\n",string(cmdch));
5045         }
5046         iin.Redraw();
5047         continue
5048     }else
5049     if( ch == 'c' || ch == 'C' ){
5050         if 0 < len(iin.line) {
5051             xline,tail := delTailChar(iin.line)
5052             if len([jbyte(tail)]) == 1 {
5053                 ch = int(tail[0])
5054                 if( 'a' <= ch && ch <= 'z' ){
5055                     ch = ch + 'A'-'a'
5056                 }else
5057                 if( 'A' <= ch && ch <= 'Z' ){
5058                     ch = ch + 'a'-'A'
5059                 }
5060                 iin.line = xline + string(ch)
5061             }
5062         }
5063         if MODE_ViTrace {
5064             fprintf(stderr,"%v\r\n",string(cmdch));
5065         }
5066         iin.Redraw();
5067         continue
5068     }else{
5069         iin.pch = append(iin.pch,ch) // push
5070         ch = '\\\' // push
5071     }
5072 }
5073 switch( ch ){
5074     case 'P'-0x40: ch = GO_UP
5075     case 'N'-0x40: ch = GO_DOWN
5076     case 'B'-0x40: ch = GO_LEFT
5077     case 'F'-0x40: ch = GO_RIGHT
5078 }
5079 //fprintf(stderr,"B[%02X]\n",ch);
5080 switch( ch ){
5081     case 0:
5082         continue;
5083
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.pindMode {
5095             fprintf(stderr, "\\J\r\n")
5096             iin.indMode = 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 0xF7: // 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     if MODE_LowerLock {
5216         if 'A' <= ch && ch <= 'z' {
5217             ch = ch+'a'-'A'
5218         }
5219     }
5220     iin.line += string(ch);
5221     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     if( 0 ){
5237         fprintf(stderr,"%d\n",int(strlen(line)));
5238     }
5239     if( lseek(3,0,0) == 0 ){
5240         if( romkanmode ){
5241             var buf [8*1024]byte;
5242             convs(line,buf);
5243             strcpy(line,buf);
5244         }
5245         write(3,line,strlen(line));
5246         ftruncate(3,lseek(3,0,SEEK_CUR));
5247         //fprintf(stderr,"outsize=%d\n", (int)lseek(3,0,SEEK_END));
5248         lseek(3,0,SEEK_SET);
5249         close(3);
5250     }else{
5251         fprintf(stderr,"\r\ngetline: ");
5252         trans(line);
5253         //printf("%s\n",line);
5254         printf("\n");
5255     }
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 invocation?
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, // RWD
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.tgshell1(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     gshCtxtBuf,err0 := setupGshContext()
5351     if err0 {
5352         return gshCtxtBuf,
5353     }
5354     gshCtx := &gshCtxtBuf
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.Println("--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.Println("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.Println("Stout.WriteString")
5405         os.Stdout.Write([]byte(gline))
5406         fmt.Println("\n")
5407     }
5408     /*
5409     // should be cared in substitution ?
5410     if 0 < len(gline) && gline[0] == '!' {
5411         xline, 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 = xline
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     gshCtxtBuf := GshContext{}
5432     gsh := &gshCtxtBuf
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             gshCtxtBuf,err := setupGshContext()
5442             gsh := &gshCtxtBuf
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 //--></div></details>
5458 //--><details id="gsh-todo"><summary>Considerations</summary><div class="gsh-src">
5459 // - inter gsh communication, possibly running in remote hosts -- to be remote shell
5460 // - merged histories of multiple parallel gsh sessions
5461 // - alias as a function or macro
5462 // - instant alias and environ export to the permanent > ~/.gsh/gsh-alias and gsh-environ
5463 // - retrieval PATH of files by its type
5464 // - gsh as an IME with completion using history and file names as dictionaires
5465 // - gsh a scheduler in precise time of within a millisecond
5466 // - all commands have its subcommand after "___" symbol
5467 // - filename expansion by "find" command
5468 // - history of ext code and output of each command
5469 // - "script" output for each command by pty-tee or telnet-tee
5470 // - $BUILTIN command in PATH to show the priority
5471 // - "?" symbol in the command (not as in arguments) shows help request
5472 // - searching command with wild card like: which ssh-*
5473 // - longformat prompt after long idle time (should dismiss by BS)
5474 // - customizing by building plugin and dynamically linking it
5475 // - generating syntactic element like "if" by macro expansion (like CPP) >> alias
5476 // - "!" symbol should be used for negation, don't wast it just for job control
5477 // - don't put too long output to tty, record it into GSH_HOME/session-id/comand-id.log
5478 // - making canonical form of command at the start adding quataion or white spaces
5479 // - name(a,b,c) ... use "(" and ")" to show both delimiter and realm
5480 // - name? or name! might be useful
5481 // - htar format - packing directory contents into a single html file using data scheme
5482 // - filepattern substitution shold be done by each command, especially in case of builtins
5483 // - @N substitution for the history of working directory, and @spec for more generic ones
5484 // - @dir prefix to do the command at there, that means like (chdir @dir; command)
5485 // - GSH_PATH for plugins
5486 // - standard command output: list of data with name, size, resource usage, modified time
5487 // - generic sort key option -nm name, -sz size, -ru usrage, -ts start-time, -tm mod-time
5488 // - wc word-count, grep match line count, ...
5489 // - standard command execution result: list of string, -tm, -ts, -ru, -sz, ...
5490 // - tailf-filename like tail -f filename, repeat close and open before read
5491 // - max. size and max. duration and timeout of (generated) data transfer
5492 // - auto. numbering, aliasing, IME completion of file name (especially rm of queier name)
5493 // - IME "?" at the top of the command line means searching history
5494 // - IME @d/0x10000 /ix/ffff/
5495 // - IME ESC to go the edit mode like in vi, and use :command as :s/x/y/g to edit history
5496 // - gsh in WebAssembly
5497 // - gsh as a HTTP server of online-manual
5498 //---END--- (^~)//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 "Ly8gTUIJTUVCM4LwjbGg6l6e5pu4ICgyMDiWLT4MTlhKOpzzWthaSDkuJbnlyWka28g44GJt"+
5506 "Cm5uOOCkwpuaSdjgasKy2hpIOOBop0aSDjgaEKAEGh4GvCnNlIOOBmwprYSDjgYsKaSDj+"
5507 "gYQK";
5508 //</span>
5509
5510 var WnnDic = //<span id="gsh-wnn-dic">
5511 "data:text/dic;base64,"+
5512 "PG1ldGEgY2hhcnNldD0iVVRGLTgiPg08dGV4dGFyZWEgY29scz04MCByb3dzPTQwPgovL2Rp"+ 
5513 "Y32lcg1HU2h1bGxco0LNvxzZGljdGlvbmFyeVxz2m9yXHNyM5cyc8vXHMyMDiWLT4MzAK"+ 
5514 "RlNo2WxsCudAgsVbsArjgo/jg2/jg2cJ56eBchndGFzaGkJ56eBchndGFzaQnnp4E44Gk+" 
5515 "4AG+4G1CeWQjeWjJpuWlhZqn1kI3l1Y0K44Gg44GL4GuCeS4remHjgpuyWtbm8J5Llt+" 
5516 "6yeOchndCeOcjwp0yDnjg28czk2J44GvCnNoaQnjgZkDm8j44GuCm5hCeObggptYQnjgb4K+" 
5517 "ZQnjgYgkAcEJ44Gvcm5hCeOBqgprYQnjgYSkbm8J44GuCm1CeOBpwzQnjg2kKZVxzCwVj+" 
5518 "agB8K2G1jCWRpYwp1Y2hvcWVjaG8KcmwQjwGf5CXkJcGxheQpyZBLYXQJcmVwZWFcR0CwRh+" 
5519 "GvccysnbdV4dGfYzWE+Cg==" 
5520 "aw9uJjwvdCV4dGfYzWE+Cg==" 
5521 //</span>
5522
5523 var SumomoDic = //<span id="gsh-sumomo-dic">
5524 "data:text/dic;base64,"+
5525 "PG1ldGEgY2hhcnNldD0iVVRGLTgiPg08dGV4dGFyZWEgY29scz04MCByb3dzPTQwPgovL3z1"+ 
5526 "cg1HU2h1bGxco0LNvxzZGljdGlvbmFyeVxz2m9yXHNyTaG1qlawlpXHMv1lxzMjAyMC0wODMw+" 
5527 "c3UJ44G2Cm1yCeCgppubnwjg4KdnjeYYKKYhpcceBQp0aQnjgakDmN0aQnlhOUkDxpR+" 
5528 "CeWghQpdw1vbW8J44G244KCcn1lb9w9bz1vceBmeOcguOcgppbt21vCeahgwpT+" 
5529 "b21vbW8J5gD44KCIiwCsOaQouIgngjI1KPC902Xh0YXJ1YT4R" 
5530 //</span>
5531
5532 var SijimiDic = //<span id="gsh-sijimi-dic">
5533 "data:text/dic;base64,"+
5534 "PG1ldGEgY2hhcnNldD0iVVRGLTgiPg08dGV4dGFyZWEgY29scz04MCByb3dzPTQwPgovL3z1"+ 
5535 "cg1HU2h1bGxco0LNvxzZGljdGlvbmFyeVxz2m9yXHNyTaG1qlawlpXHMv1lxzMjAyMC0wODMw+" 
5536 "c11ldGEb1p2zaGkJ44GXCmpCe0BmptAqnjgb2KbmJ44GgCmp1ceBm0OChp0eXuJ44KF+" 
5537 "CuNj44GCGCm5pCeBwpwrbwnjgMKYnuJ44G2Cm5uCeOkwpubnwjg4K2hpCeOb0Qp0Qnj+" 
5538 "gkAeKa2J44GLCnJhCeOciQosLAnjg1eEKLi4J44CCnnuW5hCeS4gpw4an1CeWngQp4bmJ+" 
5539 "5LqMCmtveaNg1sCeWaiprb3j5vCLCm5hbmfpqdxUvaqgJNz1KbmFuWyp1dw5pHeHqJ+" 
5540 "77x77y7ScM5hbmfpqdxUvaqgJ77y7x77y7ScuS4g+wNgeS6jHgJNz1Ka29idW5uCeWai+WiHgpo+" 
5541 "awthcmrxCeOboe0B1+OClQp0aWthcmEJ5YqbCmNoaWthcmEJ5YqbCjwvdGV4dGfYzWE+Cg==" 
5542 //</span>
5543
5544 var JA_JKLDic = //<span id="gsh-ja-jkl-dic">
5545 "data:text/dic;base64,"+
5546 "Ly922XjsCU15SU1FamkPjY2ptb3JzZwKpQwPsK0woMjAyMGowODE5KSheLV4pL1NhG94SVRT+" 
5547 "Cmtqamprbgta2tsa2psIOS4lueVjApqamtqamwJ44GCCmtqbanjgYQKa2tgbAnjgYKKamtq+" 
5548 "amwJ44G1Cmtnq2atrbAnjgYQka2praw2J44GLCmpramtrAnjgYQKa2trawmJ44GPcmpramps+" 
5549 "ce0BkQpgampqpbAnjgZKamtnqa2wJ44GXCmpqamtqbanjg2Kka2pgqnts+" 
5550 "ce0BmwvpgampqpbAnjgZOKamtsCe0Bnwptra2prbAnjgka2pqa2wJ44GkCmtq2pgpAnjgayaK+" 
5551 "a2tqg2tsCe0BqgqaprbAnjgask2a2r2wJ44GsCmpq2psCe0BrOprq2pq+" 
5552 "banjgak4Kamtra2wJ44GvCmpaq2tgbAnjgkIBKamtra2wJ44G1CmtnsCeBuApq2tsCe0Buwpg+" 
5553 "a2tqbAnjg4K2a2tqa2psCe0BvpgpAnjgkoAkmtra2psCeOcqgqpa2tqa2wJ44KCmtqamwJ+" 
5554 "4KxEcmprbg2pgbAnjgkoAkmtpa2psCeOciQpgq2pgq2wJ44KLCmpq+" 
5555 "amwJ44Kmtnq2apsCeOcj0pgq2psCeOjwpramtrJ44KQcmtnqtrAnjgpkEka2pqaamwJ+" 
5556 "44KSCmtq2prbAnjgpkEka2pqa2psCe0DvAprrazWJ44KbCmtramprbAnjgpwka2prantqAnj+" 
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 </p>
5566 <!--
5567 <iframe src="https://golang.org" width="100%" height="300"/>
5568 -->
5569 <a href="https://developer.mozilla.org/ja/docs/Web">MDN web docs</a>
5570 <a href="https://developer.mozilla.org/ja/docs/Web/HTML/Element">HTML</a>
5571 CSS:
5572 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors">Selectors</a>
5573 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/background-repeat">repeat</a>
5574 HTTP
5575 JavaScript:
5576 ...
5577 </p>
5578 </div></details>
5579 */
5580 /*
```

```
5581 <details id="html-src" onclick="frame_open();"><summary>Raw Source</summary><div>
5582 <!-- h2>The full of this HTML including the Go code is here.</h2 -->
5583 <details id="gsh-whole-view"><summary>Whole file</summary>
5584 <a name="whole-src-view"></a>
5585 <span id="src-frame"><span><!-- a window to show source code -->
5586 </span></span>
5587 </details>
5588
5589 <details id="gsh-style-frame" onclick="fill_CSSView()"><summary>CSS part</summary>
5590 <a name="style-src-view"></a>
5591 <span id="gsh-style-view"></span>
5592 </details>
5593
5594 <details id="gsh-script-frame" onclick="fill_JavaScriptView()"><summary>JavaScript part</summary>
5595 <a name="script-src-view"></a>
5596 <span id="gsh-script-view"></span>
5597 </details>
5598
5599 <details id="gsh-data-frame" onclick="fill_DataView()"><summary>Built-in data part</summary>
5600 <a name="gsh-data-frame"></a>
5601 <span id="gsh-data-view"></span>
5602 </details>
5603
5604 <div id="GshFooter"></div>
5605 </div></details>
5606 /*
5607 */
5608 /*
5609 <!-- 2020-09-17 SatoxITS, visible script -->
5610 <details><summary>GJS script</summary>
5611 <style>.gjscript { font-family: Georgia; }</style>
5612 <pre id="gjscript_1" class="gjscript">
5613   function gitest1() { alert('Hello GJS script!'); }
5614   gitest1()
5615 </pre>
5616 <script>
5617   gjs = document.getElementById('gjscript_1');
5618   //eval(gjs.innerHTML);
5619   //gjs.outerHTML = "";
5620 </script>
5621 </details><!-- ----- END-OF-VISIBLE-PART ----- -->
5622 */
5623 /*
5624 <!--
5625 // 2020-09-06 added,
5626 https://developer.mozilla.org/en-US/docs/Web/CSS/z-index
5627 https://developer.mozilla.org/en-US/docs/Web/CSS/position
5628 -->
5629 <span id="GshGrid">(&^_^)/<small>(Hit j k l h)</small></span>
5630
5631 <span id="GStat"><br>
5632 </span>
5633 <span id="GMenu" onclick="GShellMenu(this)"></span>
5634 <span id="GTop"></span>
5635 <div id="GshellPlane" onclick="showGShellPlane();"></div>
5636 <div id="RawTextViewer"></div>
5637 <div id="RawTextViewerClose" onclick="hideRawTextViewer()"> CLOSE </div>
5638
5639 <style id="GshStyleDef">
5640 #LineNumbered table, tr, td {
5641   margin:0;
5642   padding:4px;
5643   spacing:0;
5644   border:12px;
5645 }
5646 textarea.LineNumber {
5647   font-size:12px;
5648   font-family:monospace,Courier New;
5649   color:#282;
5650   padding:4px;
5651   text-align:right;
5652 }
5653 textarea.LineNumbered {
5654   font-size:12px;
5655   font-family:monospace,Courier New;
5656   padding:4px;
5657   wrap:off;
5658 }
5659 #RawTextViewer{
5660   z-index:0;
5661   position:fixed; top:0px; left:0px;
5662   width:100%; height:50px;
5663   overflow:auto;
5664   color:#fff; background-color:rgba(128,128,256,0.2);
5665   font-size:12px;
5666   spellcheck:false;
5667 }
5668 #RawTextViewerClose{
5669   z-index:0;
5670   position:fixed; top:-100px; left:-100px;
5671   color:#fff; background-color:rgba(128,128,256,0.2);
5672   font-size:20px; font-family:Georgia;
5673   white-space:pre;
5674 }
5675 #GshellPlane{
5676   z-index:0;
5677   position:fixed; top:0px; left:0px;
5678   width:100%; height:50px;
5679   overflow:auto;
5680   color:#fff; background-color:rgba(128,128,256,0.6);
5681   font-size:12px;
5682 }
5683 #GTop{
5684   z-index:9;
5685   opacity:1.0;
5686   position:fixed; top:0px; left:0px;
5687   width:320px; height:20px;
5688   color:#fff; background-color:rgba(32,32,160,0.2);
5689   color:#fff; font-size:12px;
5690 }
5691 #GPos{
5692   z-index:12;
5693   position:fixed; top:0px; left:0px;
5694   opacity:1.0;
5695   width:640px; height:30px;
5696   color:#fff; background-color:rgba(0,0,0,0.4);
5697   color:#fff; font-size:12px;
5698 }
5699 }
5700 #GMenu{
5701   z-index:2000;
5702   position:fixed; top:250px; left:0px;
5703   opacity:1.0;
5704   width:100px; height:100px;
```

```

5705     color:#fff;
5706     color:#fff; background-color:rgba(0,0,0,0.0);
5707     color:#fff; font-size:16px; font-family:Georgia;
5708     background-repeat:no-repeat;
5709   }
5710   #GStat{
5711     z-index:8;
5712     opacity:0.0;
5713     position:fixed; top:20px; left:0px;
5714     width:640px;
5715     width:100%; height:90px;
5716     color:#fff; background-color:rgba(0,0,128,0.10);
5717     font-size:20px; font-family:Georgia;
5718   }
5719   #GLog{
5720     z-index:10;
5721     position:fixed; top:50px; left:0px;
5722     opacity:1.0;
5723     width:640px; height:60px;
5724     color:#fff; background-color:rgba(0,0,128,0.10);
5725     font-size:12px;
5726   }
5727   #GshGrid {
5728     z-index:11;
5729     opacity:0.0;
5730     position:fixed; top:0px; left:0px;
5731     width:320px; height:30px;
5732     color:#f9f; font-size:16px;
5733   }
5734   xbody {display:none;}
5735   .gsh-link{color:green;}
5736   #gsh {border-width:1; margin:0; padding:0;}
5737   #gsh {font-family:monospace,Courier New;color:#ddf;font-size:8px;}
5738   #gsh {header;height:100px;}
5739   #xgsh {header; height:100px; background-image:url(GShell-Logo00.png);}
5740   #GshMenu{font-size:14pt;color:#c44;}
5741   .GshMenu{font-size:14pt;color:#2a2;padding:4px;}
5742   .GshMenu:hover{font-size:14pt;color:#fff;font-weight:bold;background-color:#2a2;}
5743   #GshFooter{height:100px;background-size:80px;background-repeat:no-repeat;}
5744   #gsh note{color:#000;font-size:10pt;}
5745   #gsh h2{color:#24a;font-family:Georgia;font-size:18pt;}
5746   #gsh h3{color:#24a;font-family:Georgia;font-size:16pt;}
5747   #gsh details{color:#888;background-color:#fff;font-family:monospace;}
5748   #gsh summary{font-size:16pt;color:#fff;background-color:#8af;height:30px;}
5749   #gsh pre{font-size:11pt;color:#223;background-color:#faaaff;}
5750   #gsh a{color:#24a;}
5751   #gsh a[name]{color:#24a;font-size:16pt;}
5752   #gsh .gsh-src{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
5753   #gsh .gsh-src{background-color:#faaaff;color:#223;}
5754   #gsh-src-src{spellcheck:false}
5755   #src-frame-textarea{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
5756   #src-frame-textarea{background-color:#faaaff;color:#223;}
5757   .gsh-code {white-space:pre;font-family:monospace !important;}
5758   .gsh-code {color:#088;font-size:11pt; background-color:#eef;}
5759   .gsh-golang-data {display:none;}
5760   #gsh-WinId {color:#000;font-size:14pt;}
5761
5762   .gsh-document {font-size:11pt;background-color:#fff;font-family:Georgia;}
5763   .gsh-document {color:#000;background-color:#fff !important;}
5764   .gsh-document > h2{color:#000;background-color:#fff !important;}
5765   .gsh-document details{color:#000;background-color:#fff;font-family:Georgia;}
5766   .gsh-document p{max-width:550pt;color:#000;background-color:#fff;font-family:Georgia;}
5767   .gsh-document address{width:500pt;color:#000;background-color:#fff;font-family:Georgia;}
5768
5769 @media print {
5770   #gsh pre{font-size:11pt !important;}
5771 }
5772 </style>
5773
5774 <!--
5775 // Logo image should be drawn by JavaScript from a meta-font.
5776 // CSS seems not follow line-splitted URL
5777 -->
5778 <script id="gsh-data">
5779 //GsellLogo="QR-ITS-more.jp.png"
5780 GsellLogo="data:image/png;base64,
5781 iVBORw0KGgoAAAANSUhEUgAAQAEAAACAYAAADvs3f4AAAAAXNSR0IArs4c6QAAAHhlWEIm\ 
5782 TU0AkAgAAAbAAUAAAABAAAPBebAAAUAABAAAArgEoAAMAAAABAAIAIdpAAQAAAAB\ 
5783 AAAATgAAAABAAIAAAAQAEEgAAAABAAQAgQDAAAAAQABAACgAgEAAAAAQAAAQGgAwA\ 
5784 AAAAQAAHgAAAAlYbhgAAAlwSflzAAALeWAcxMbaJqgCGAAAF3RJRFETaHtnQuUFNWZ\ 
5785 x+tt7uk23icgg0/jY6Osbb8WgqAzzAvn7uG4+bISTR7YnQxdpCkgj2aNw1D2MSlRkeuA\ 
5786 4iuNx7xjrii2qIBElSggCo1MMA+mu+v//2MD9U1dau6a2albv91GKrq3vwdx6/q\ 
5787 fNvdxz8tBA8tAIESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES\ 
5788 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES\ 
5789 IAESIAESIAESIAESTASIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES\ 
5790 IAESIAESIAESIAESTASIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESI\ 
5791 Zexs9H9+ftSK5dixnsic2zqddE7ysU+7igaa1Ktn5ysoMhwEptdK4mQfz3zUeEx1bLYsayU1\ 
5792 npDILkxE2C1F1RM53JSUag9ScqoUi+2kk3StuN5yreEGKJ7Qw7m0vkec2Tqoqizw01jhFS\ 
5793 pdV0HCt0MRb3USXEHfipu7Dmfmb2+x4v+wWFWVkbpMe7uKoGabe6ekG01nh56PC\ 
5794 Hx2VVVBKorRkqhg3qUyIyDafOnJ56Okd16WkOQdyPz1ON9LmXpF/60p2P/Piyov\ 
5795 N8mfm+IwN9Gnjw9KqO7oLvgSFt2pR11gn3i0Vkt7xsoWMzeuVpfplRKyfdoak2LRSB0d\ 
5796 zrWocCo66gPhvgRacj/atkj3g7xKH4gKN6aRSo2zpYzerqSGRa0zDQqfk79SKTRXH\ 
5797 L66as8pU/Pn1pN1LQJJK8c73dX58r20ur7iiwpc8ObnbNcyh1lrryy0tQyF5JfvglB7jx\ 
5798 +chNbhbj5gJryblJH3y9o84d40H20tx87HaPeFu101+w1C-knyh5FGEv0WGGExB83eMoLV\ 
5799 rikbd9gHEP52Vg01h489PUA6kJyVphbzbLJg4zfiesnHCwvUoeiVOOb/5c9F9Y9D1ueOH\ 
5800 +zGuh9NsqQcm0w7urk19RpjBD4Y6uQcQd5TUOW63zD3MHezy14V9iShdKyxbGH1CpFR\ 
5801 UJ6-toCP7F9VF58NBEDHTOMBaE74Ent+eWrWr+Lz/QWv60AdB7QJUjps/0A7c0oBNBCeMUz\ 
5802 ttCu/coG28Ltpv1LTPFV8juRasEahBhvxaRguoBPyfUDo4+OfeBdyb8L4t2z9xExSFAMo\ 
5803 bgGov01gZggW4jJ392xNmhd-Mwf3J3TjfntZzy1CYBjXNUT5K1Kyc1ksxRld16Bmcvev\ 
5804 aJoyv/GeAKmEqvEP46/lnjjt9xj17VL53215Mtvpap1QGLNHw5pqQgkxNyQ1z2b8wGCMGZ2V\ 
5805 qOfjdxyV0A2zayidv6FJ3CSj4Xk9h1r7e27zmp3T8hLJpkYicJpV1Ht/Kd4J9w1\ 
5806 L56mxFhR9fzzgRKJw/C+HQSpE+krb1lyN3gEP7NaHsHaLds2xh5Q5CnCPPDvePgcgbw/8e\ 
5807 7/zd0aHptag/mlKJ77UVOGxbyb7dx/Ex/Frf1i7z7Ku+SciCxUw0hUxF16wV9H+cCvg1\ 
5808 pd/cf4u2K2IUP1vTK1L/Jjy1ESPVHq728NzvUzvzODGy9GoupuuhmLNlfctx48YHL2q\ 
5809 f/8hpVx/43rg9xtg6tvcl3cfmWQn9nbfc1le7wKE1bOK65iBuEhdh3IaW82dwKUpw\ 
5810 hravuc62cWdkc1j528EUXMae71zuqCwzNb1neVn1j9/P7w+iMaQgF+NI3JLSf8dn91p\ 
5811 WNW4rPryjJxuPleL/HxNzgTsves1d2wsNWIH95rVvXZ9fo4v/IfmqdEipD1fM2uCW\ 
5812 gJy12w0ENPa2f3fEcivd-2YNNCNTJyra08jRoJTAUmrigQCljnRW5fpN+frTwdh4SiUv\ 
5813 bvLvbfflCRF04qaZD176/Bjy1kD5pB1z5W14wQn7tikPbeOpw+Kj30sqP8GHInOA2uw\ 
5814 iOzuywDhQ9Br2xoDRqjQFp15oxRH6w0VjRKAAW46pvT+RxAJV1jW7vY9/+CeUBMk168/rQn\ 
5815 mCuFkzaldFN/y1gA5iWc3dkIKhsyvZuCI5VG/RKewhFWRKAMMcdbEXX+rHf12A9bt2d172\ 
5816 2qNzOvzCDImfb7tNy7oogDXWIKAI07/cQZchyADWnergNsVxttcjsGdp2otwmQWUta+rh7\ 
5817 yhYbUgmlXt7f1K1DwaryUfn42FluxNDDVtAmLsYC9826Vtb2aw2px8Nimehz3EM+mgso1k\ 
5818 d3/znbGE1XPWbzXg1Ycg5eW5/BzGy54aNoGwKwfNwbqpccevW14FUbvov32gew8DzLdTmAj\ 
5819 aqpg7t/BMXX+yw/ejGKotksy2d+gFbb9VodvX5B1ZTOR-Wfjyb0pP6U0XGOYyR/quata3vB\ 
5820 Fgeua6gv2d7vn8dfDv3r1ldbw34GSPg91.0D9G95kWkhn9kaHMyj6dkLPzmd3cnu77w7w5C\ 
5821 h/YrGlP7wxp/VvuRDUc+wsq54ym+8zzKQgyRSPRa4IKoG118b6ytagcPmb9w/m09cUAtz\ 
5822 Jow6tbnPcmHxZ+j+snNpHsCJyja6csrRsMyrGkwiF415uoli1LLR7fmceX3z2+/GfW1lU2\ 
5823 Y572b6EazkfYpCetJ15QlnJyLdrFrU2p1/3pmku/y99aGoGyMT7neVivx/6CHuQh1luh/\ 
5824 f9Uvo+g703qzrxFLBx0+z+w/8F6Pw6Fv7xShxNlLayvWdz21Ulm/4uL1mpwNa5Gcd0L9\ 
5825 2Fa6cgioxzhT6GQ41NR5D0j9xuvlcy+rPbcujsnLkVwCepphBICLRwv+9Kp4vngHg6Fc2\ 
5826 NCqMSiCsnCkfxel+mflBwuxdmPbo2q7/194225Y37CrzpoWhthGzHraO/ybokkdhpanZq\ 
5827 GxWf66/8C5b8HcbzdpnhUje6YFow1g2eMmtqCDeKzTlXVuc2Lk4yvT2epug5tgSWFkxkd\ 
5828 ufu9MfW1G3sqNtCx76+3xExQWW2VeqSpvz2McafyISVY461+04KvyVgicCugG2rp0yPtvej\
```





```
6077     xxxcolor:#22a !important;
6078 }
6079 input {
6080   border:1px dashed #0f0; border-radius:0px;
6081 }
6082 .GJWin:hover{
6083   color:#df8 !important;
6084   background-color:rgba(32,32,160,0.8) !important;
6085   line-height:0.0;
6086 }
6087 .GJWin:active{
6088   color:#df8 !important;
6089   background-color:rgba(224,32,32,0.8) !important;
6090   line-height:0.0;
6091 }
6092 .GJWin:focus{
6093   color:#df8 !important;
6094   background-color:rgba(32,32,32,1.0) !important;
6095   line-height:0.0;
6096 }
6097 .GJWin{
6098   z-index:10000;
6099   display:inline;
6100   position:relative;
6101   flex-wrap: wrap;
6102   top:0; left:0px;
6103   width:285px !important; height:205px !important;
6104   border:1px solid #eaa; border-radius:2px;
6105   margin:0px; padding:0px;
6106   font-size:8pt;
6107   line-height:0.0;
6108   color:#fff; background-color:rgba(0,0,64,0.1) !important;
6109 }
6110 .GJTab{
6111   display:inline;
6112   position:relative;
6113   top:0px; left:0px;
6114   margin:0px; padding:2px;
6115   border:0px solid #000; border-radius:2px;
6116   width:90px; height:20px;
6117   font-family:Georgia;
6118   font-size:9pt;
6119   line-height:1.0;
6120   white-space:nowrap;
6121   color:#fff; background-color:rgba(0,0,64,0.7);
6122   text-align:center;
6123   vertical-align:middle;
6124 }
6125 .GJStat:focus{
6126   color:#df8 !important;
6127   background-color:rgba(32,32,32,1.0) !important;
6128   line-height:1.0;
6129 }
6130 .GJStat{
6131   display:inline;
6132   position:relative;
6133   top:0px; left:0px;
6134   margin:0px; padding:2px;
6135   border:0px solid #00f; border-radius:2px;
6136   width:16px; height:0px;
6137   font-family:monospace;
6138   font-size:9pt;
6139   line-height:1.0;
6140   color:#fff; background-color:rgba(0,0,64,0.2);
6141   text-align:center;
6142   vertical-align:middle;
6143 }
6144 .GJIcon{
6145   display:inline;
6146   position:relative;
6147   top:0px; left:1px;
6148   border:2px solid #44a;
6149   margin:0px; padding:1px;
6150   width:13.2; height:13.2px;
6151   border-radius:2px;
6152   font-family:Georgia;
6153   font-size:13.2px;
6154   line-height:1.0;
6155   white-space:nowrap;
6156   color:#fff; background-color:rgba(32,32,160,0.8);
6157   text-align:center;
6158   vertical-align:middle;
6159   text-shadow:0px 0px;
6160 }
6161 .GJText:focus{
6162   color:#fff !important;
6163   background-color:rgba(32,32,160,0.8) !important;
6164   line-height:1.0;
6165 }
6166 .GJText{
6167   display:inline;
6168   position:relative;
6169   top:0px; left:0px;
6170   border:0px solid #000; margin:0px; padding:0px;
6171   width:280px; height:160px;
6172   border:0px;
6173   font-family:Courier New,monospace !important;
6174   font-size:8pt;
6175   line-height:1.0;
6176   white-space:pre;
6177   color:#fff; xbackground-color:rgba(0,0,64,0.5);
6178   background-color:rgba(32,32,128,0.8) !important;
6179 }
6180 .GJMode{
6181   display:inline;
6182   position:relative;
6183   top:0px; left:0px;
6184   border:0px solid #000; border-radius:0px;
6185   margin:0px; padding:0px;
6186   width:280px; height:20px;
6187   font-size:9pt;
6188   line-height:1.0;
6189   white-space:nowrap;
6190   color:#fff; background-color:rgba(0,0,64,0.7);
6191   text-align:left;
6192   vertical-align:middle;
6193 }
6194 </style>
6195
6196 <script id="gsh-script">
6197   // 2020-0909 added, permanet local storage
6198   // https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
6199   var MyHistory = ""
6200   Permanent = localStorage;
```

```

6201 MyHistory = Permanent.getItem('MyHistory')
6202 if( MyHistory == null ){ MyHistory = "" }
6203 d = new Date()
6204 MyHistory = d.getTime()/1000+ " "+document.URL+"\n" + MyHistory
6205 Permanent.setItem('MyHistory',MyHistory)
6206 //Permanent.setItem('MyWindow',window)
6207
6208 var GJLog_Win = null
6209 var GJLog_Tab = null
6210 var GJLog_Stat = null
6211 var GJLog_Text = null
6212 var GJWin_Mode = null
6213 var FProductInterval = 0
6214
6215 var GJ_FactoryID = -1
6216 var GJFactory = null
6217 if( e = document.getElementById('GJFactory_0') ){
6218     GJFactory_1.height = 0
6219     GJFactory = e
6220     e.setAttribute('class','GJFactory')
6221     var GJ_FactoryID = 0
6222 }else{
6223     GJFactory = GJFactory_1
6224     var GJ_FactoryID = 1
6225 }
6226
6227 function GJFactory_Destroy(){
6228     gif = GJFactory
6229     //gif = document.getElementById('GJFactory')
6230     //alert('gif=' + gif)
6231     if( gif != null ){
6232         if( gif.childNodes != null ){
6233             for( i = 0; i < gif.childNodes.length; i++ ){
6234                 gif.removeChild(gif.childNodes[i])
6235             }
6236         }
6237         gif.innerHTML = ''
6238         gif.style.width = 0
6239         gif.style.height = 0
6240         gif.removeAttribute('style')
6241         GJLog_Win = GJLog_Tab = GJLog_Stat = GJLog_Text = GJWin_Mode = null
6242         window.clearInterval(FProductInterval)
6243         return '-- Destroy: work product destroyed'
6244     }else{
6245         return '-- Destroy: work product not exist'
6246     }
6247 }
6248
6249 var TransMode = false
6250 var onKeyControl = false
6251 var OnKeyShift = false
6252 var OnKeyAlt = false
6253 var OnKeyJ = false
6254 var OnKeyK = false
6255 var OnKeyL = false
6256
6257 function GJWin_OnKeyUp(ev){
6258     keycode = ev.code;
6259     if( keycode == 'ShiftLeft' ){
6260         OnKeyShift = false
6261     }else
6262     if( keycode == 'ControlLeft' ){
6263         onKeyControl = false
6264     }else
6265     if( keycode == 'AltLeft' ){
6266         OnKeyAlt = false
6267     }else
6268     if( keycode == 'KeyJ' ){ OnKeyJ = false }else
6269     if( keycode == 'KeyK' ){ OnKeyK = false }else
6270     if( keycode == 'KeyL' ){ OnKeyL = false }else
6271     {
6272     }
6273     ev.preventDefault()
6274 }
6275 function and(a,b){ if(a){ if(b){ return true; } return false; } }
6276 function GJWin_OnKeyDown(ev){
6277     keycode = ev.code;
6278     mode = '';
6279     key = ''
6280     if( keycode == 'ControlLeft' ){
6281         onKeyControl = true
6282         ev.preventDefault()
6283         return;
6284     }else
6285     if( keycode == 'ShiftLeft' ){
6286         OnKeyShift = true
6287         ev.preventDefault()
6288         return;
6289     }else
6290     if( keycode == 'AltLeft' ){
6291         ev.preventDefault()
6292         OnKeyAlt = true
6293         return;
6294     }else
6295     if( keycode == 'Backquote' ){
6296         TransMode = !TransMode
6297         ev.preventDefault()
6298     }else
6299     if( and(keycode == 'Space', OnKeyShift ) ){
6300         TransMode = !TransMode
6301         ev.preventDefault()
6302     }else
6303     if( keycode == 'ShiftRight' ){
6304         TransMode = !TransMode
6305     }else
6306     if( keycode == 'Escape' ){
6307         TransMode = true
6308         ev.preventDefault()
6309     }else
6310     if( keycode == 'Enter' ){
6311         TransMode = false
6312         //ev.preventDefault()
6313     }
6314     if( keycode == 'KeyJ' ){ OnKeyJ = true }else
6315     if( keycode == 'KeyK' ){ OnKeyK = true }else
6316     if( keycode == 'KeyL' ){ OnKeyL = true }else
6317     {
6318     }
6319
6320     if( ev.altKey ){ key += 'Alt+' }
6321     if( onKeyControl ){ key += 'Ctrl+' }
6322     if( OnKeyShift ){ key += 'Shift+' }
6323     if( and(keycode != 'KeyJ', OnKeyJ ) ){ key += 'J+' }
6324     if( and(keycode != 'KeyK', OnKeyK ) ){ key += 'K+' }

```

```

6325     if( and(keycode != 'KeyL', OnKeyL) ){ key += 'L+' }
6326     key += keycode
6327
6328     if( TransMode ){
6329         //mode = "[\343\201\202r]"
6330         mode = "[\u{202r}]"
6331     }else{
6332         mode = '[---]'
6333     }
6334     ////  /gjmode.innerHTML = "[---]"
6335     GJWin_Mode.innerHTML = mode + ' ' + key
6336     //alert('Key:' +keycode)
6337     ev.stopPropagation()
6338     //ev.preventDefault()
6339 }
6340 function GJWin_OnScroll(ev){
6341     x = DragStartX = gsh.getBoundingClientRect().left.toFixed(0)
6342     y = DragStartY = gsh.getBoundingClientRect().top.toFixed(0)
6343     GJLog_append('OnScroll: x=' +x+ ',y=' +y)
6344 }
6345 document.addEventListener('scroll',GJWin_OnScroll)
6346 function GJWin_OnResize(ev){
6347     w = window.innerWidth
6348     h = window.innerHeight
6349     GJLog_append('OnResize: w=' +w+ ',h=' +h)
6350 }
6351 window.addEventListener('resize',GJWin_OnResize)
6352
6353 var DragStartX = 0
6354 var DragStartY = 0
6355 function GJWin_DragStart(ev){
6356     // maybe this is the grabbing position
6357     this.style.position = 'fixed'
6358     x = DragStartX = this.getBoundingClientRect().left.toFixed(0)
6359     y = DragStartY = this.getBoundingClientRect().top.toFixed(0)
6360     GJLog_Stat.value = 'DragStart: x=' +x+ ',y=' +y
6361 }
6362 function GJWin_Drag(ev){
6363     x = ev.clientX; y = ev.clientY // x = ev.pageX; y = ev.pageY
6364     this.style.left = x - DragStartX
6365     this.style.top = y - DragStartY
6366     this.style.zIndex = '30000'
6367     this.style.position = 'fixed'
6368     x = this.getBoundingClientRect().left.toFixed(0)
6369     y = this.getBoundingClientRect().top.toFixed(0)
6370     GJLog_Stat.value = 'x=' +x+ ',y=' +y
6371     ev.preventDefault()
6372     ev.stopPropagation()
6373 }
6374 function GJWin_DragEnd(ev){
6375     x = ev.clientX; y = ev.clientY
6376     //x = ev.pageX; y = ev.pageY
6377     this.style.left = x - DragStartX
6378     this.style.top = y - DragStartY
6379     this.style.zIndex = '30000'
6380     this.style.position = 'fixed'
6381     if( true ){
6382         console.log("Dropped: " +this.nodeName + '#' +this.id + ' x=' +x+ ' y=' +y
6383             + ' parent=' +this.parentNode.id)
6384     }
6385     x = this.getBoundingClientRect().left.toFixed(0)
6386     y = this.getBoundingClientRect().top.toFixed(0)
6387     GJLog_Stat.value = 'x=' +x+ ',y=' +y
6388     ev.preventDefault()
6389     ev.stopPropagation()
6390 }
6391 function GJWin_DragIgnore(ev){
6392     ev.preventDefault()
6393     ev.stopPropagation()
6394 }
6395 // 2020-09-15 let every object have console view!
6396 var GJ_ConsoleID = 0
6397 function GJLog_StatUpdate(){
6398     txa = GJLog_Stat;
6399     if( txa == null ){
6400         return;
6401     }
6402     p = txa.parentNode;
6403     pw = txa.getBoundingClientRect().width;
6404     ph = txa.getBoundingClientRect().height;
6405     txa.value += '#'+p.id+' pw=' +pw+', ph=' +ph+'\n';
6406
6407     w = txa.getBoundingClientRect().width;
6408     h = txa.getBoundingClientRect().height;
6409     txa.value += 'w=' +w+ ', h=' +h+'\n';
6410
6411     txa.value += '\n';
6412     txa.value += DateShort() + '\n';
6413     // vertical centering of the last line
6414     txa.scrollTop = txa.scrollHeight - 30; // depends on the font-size
6415 }
6416 function GJ_showTime1(wid){
6417     e = document.getElementById(wid);
6418     if( e != null ){
6419         e.value = DateShort();
6420     }else{
6421         // should remove the Listener
6422     }
6423 }
6424 function GJWin_OnResizeTextarea(ev){
6425     this.value += 'resized:' + '\n'
6426 }
6427 function GJ_NewConsole(wname){
6428     wid = wname + ' ' + GJ_ConsoleID
6429     GJ_ConsoleID += 1
6430
6431     GJFactory.style.setProperty('width', 360 + 'px'); //GJFsize
6432     GJFactory.style.setProperty('height', 320 + 'px')
6433     e = GJFactory;
6434     console.log('GJFa #' +e.id+ ' from w=' +e.style.width+ ', h=' +e.style.height)
6435
6436     if( GJFactory.innerHTML == "" ){
6437         GJFactory.innerHTML = '<' +H3>GJ Factory_ ' + GJ_FactoryID + '<' +H3><' +hr>\n'
6438     }else{
6439         GJFactory.innerHTML += '<' +hr>\n'
6440     }
6441
6442     gjwin = GJLog_Win = document.createElement('span')
6443     gjwin.id = wid
6444     gjwin.setAttribute('class', 'GJWin')
6445     gjwin.setAttribute('draggable', 'true')
6446     gjwin.addEventListener('dragstart', GJWin_DragStart)
6447     gjwin.addEventListener('drag', GJWin_Drag)
6448     gjwin.addEventListener('dragend', GJWin_Drag)

```

```

6449 gjwin.addEventListener('dragover',GJWin_DragIgnore)
6450 gjwin.addEventListener('dragenter',GJWin_DragIgnore)
6451 gjwin.addEventListener('dragleave',GJWin_DragIgnore)
6452 gjwin.addEventListener('dragexit',GJWin_DragIgnore)
6453 gjwin.addEventListener('drop',GJWin_DragIgnore)
6454 gjwin.addEventListener('keydown',GJWin_OnKeyDown)
6455
6456 gjtab = GJLog_Tab = document.createElement('textare')
6457 gjtab.addEventListener('keydown',GJWin_OnKeyDown)
6458 gjtab.style.readonly = true
6459 gjtab.contenteditable = false
6460 gjtab.value = wid
6461 gjtab.id = wid + ' Tab'
6462 gjtab.setAttribute('class','GJTab')
6463 gjtab.setAttribute('spellcheck','false')
6464 gjwin.appendChild(gjtab)
6465
6466 gjstat = GJLog_Stat = document.createElement('textare')
6467 gjstat.addEventListener('keydown',GJWin_OnKeyDown)
6468 gjstat.id = wid + '_Stat'
6469 gjstat.value = DateShort()
6470 gjstat.setAttribute('class','GJStat')
6471 gjstat.setAttribute('spellcheck','false')
6472 gjwin.appendChild(gjstat)
6473
6474 gjicon = document.createElement('span')
6475 gjicon.addEventListener('keydown',GJWin_OnKeyDown)
6476 gjicon.id = wid + '_Icon'
6477 gjicon.innerHTML = '<font color="#f44">J</font>'
6478 gjicon.setAttribute('class','GJIcon')
6479 gjicon.setAttribute('spellcheck','false')
6480 gjwin.appendChild(gjicon)
6481
6482 gjtext = GJLog_Text = document.createElement('textare')
6483 gjtext.addEventListener('keydown',GJWin_OnKeyDown)
6484 gjtext.addEventListener('keyup',GJWin_OnKeyUp)
6485 gjtext.addEventListener('resize',GJWin_OnResizeTextarea)
6486 gjtext.id = wid + '_Text'
6487 gjtext.setAttribute('class','GJText')
6488 gjtext.setAttribute('spellcheck','false')
6489 gjwin.appendChild(gjtext)
6490
6491
6492 // user's mode as of IME
6493 gjmode = GJWin_Mode = document.createElement('textare')
6494 gjmode.addEventListener('keydown',GJWin_OnKeyDown)
6495 gjmode.addEventListener('keyup',GJWin_OnKeyDown)
6496 gjmode.id = wid + '_Mode'
6497 gjmode.setAttribute('class','GJMode')
6498 gjmode.setAttribute('spellcheck','false')
6499 gjmode.innerHTML = '[---]'
6500 gjwin.appendChild(gjmode)
6501
6502 gjwin.zIndex = 30000
6503 GJFactory.appendChild(gjwin)
6504
6505 gjtab.scrollTop = 0
6506 gjstat.scrollTop = 0
6507
6508 //x = gjwin.getBoundingClientRect().left.toFixed(0)
6509 //y = gjwin.getBoundingClientRect().top.toFixed(0)
6510 //gjwin.style.position = 'static'
6511 //gjwin.style.left = 0
6512 //gjwin.style.top = 0
6513
6514 //update = +'wid'.value=DateShort());
6515 update = '(GJ.showTime('+wid+'))';
6516 FProductInterval = window.setInterval(update,200)
6517 return update
6518 }
6519 function xxxGJF_StripClass(){
6520 GJLog_Win.style.removeProperty('width')
6521 GJLog_Tab.style.removeProperty('width')
6522 GJLog_Stat.style.removeProperty('width')
6523 GJLog_Text.style.removeProperty('width')
6524 return "Stripped classes"
6525 }
6526 function isElem(id){
6527   return document.getElementById(id) != null
6528 }
6529 function GJLog_append(...args){
6530   txt = GJLog_Text;
6531   if( txt == null ){
6532     return; // maybe GJLog element is removed
6533   }
6534   logs = args.join(' ')
6535   txt.value += logs + '\n'
6536   txt.scrollTop = txt.scrollHeight
6537   //GJLog_Stat.value = DateShort()
6538 }
6539 //window.addEventListener('time',GJLog_StatUpdate)
6540 window.setInterval(GJLog_StatUpdate,1000);
6541 GJ_NewConsole('GJ_Console')
6542
6543 e = GJFactory;
6544 console.log('GJF0 #'+e.id+' from w='+e.style.width+', h='+e.style.height)
6545 e.style.width = 360; //GJFsize
6546 e.style.height = 320;
6547 console.log('GJF0 #'+e.id+' to w='+e.style.width+', h='+e.style.height)
6548
6549 var StopConsoleLog = true
6550 // 2020-09-15 added,
6551 // log should be saved to permanent memory
6552 // const px = new Proxy(console.log,{ alert() })
6553 __console_log = console.log
6554 __console_info = console.info
6555 __console_warn = console.warn
6556 __console_error = console.error
6557 __console_exception = console.exception
6558 // should pop callstack info.
6559 console.exception = function(...args){
6560   __console_exception(...args)
6561   alert('-- got console.exception("'+args+'")')
6562 }
6563 console.error = function(...args){
6564   __console_error(...args)
6565   alert('-- got console.error("'+args+'")')
6566 }
6567 console.warn = function(...args){
6568   __console_warn(...args)
6569   alert('-- got console.warn("'+args+'")')
6570 }
6571 console.info = function(...args){
6572   alert('-- got console.info("'+args+'")')

```

```

6573     __console_info(...args)
6574   }
6575   console.log = function(...args){
6576     __console_log(...args)
6577     if( StopConsoleLog ){
6578       return;
6579     }
6580     if( 0 <= args[0].indexOf('!') ){
6581       //alert('-- got console.log("'+args+'")')
6582     }
6583     GJLog_append(...args)
6584   }
6585   console.log('Hello, GJShell!')
6586
6587 //document.getElementById('GshFaviconURL').href = GShellFavicon
6588 document.getElementById('GshFaviconURL').href = GShellInsideIcon
6589 //document.getElementById('GshFaviconURL').href = ITsmoreQR
6590 //document.getElementById('GshFaviconURL').href = GSellLogo
6591
6592 // id of GShell HTML elements
6593 var E_BANNER = "GshBanner" // banner element in HTML
6594 var E_FOOTER = "GshFooter" // footer element in HTML
6595 var E_GINDEX = "gsh-gindex" // index of Golang code of GShell
6596 var E_GOCODE = "gsh-gocode" // Golang code of GShell
6597 var E_TODO = "gsh-todo" // TODO of GShell
6598 var E_DICT = "gsh-dict" // Dictionary of GShell
6599
6600 function bannerElem(){ return document.getElementById(E_BANNER); }
6601 function bannerStyleFunc(){ return bannerElem().style; }
6602 var bannerStyle = bannerStyleFunc()
6603 bannerStyle.backgroundImage = "url("+GSellLogo+)";
6604 //bannerStyle.backgroundImage = "url("+GShellInsideIcon+)";
6605 //bannerStyle.backgroundImage = "url("+GShellFavicon+)";
6606 GMenu.style.backgroundImage = "url("+GShellInsideIcon+)";
6607
6608 function footerElem(){ return document.getElementById(E_FOOTER); }
6609 function footerStyle(){ return footerElem().style; }
6610 footerElem().style.backgroundImage="url("+ITsmoreQR+)";
6611 //footerStyle().backgroundImage = "url("+ITsmoreQR+)";
6612
6613 function html_fold(e){
6614   if( e.innerHTML == "Fold" ){
6615     e.innerHTML = "Unfold"
6616     document.getElementById('gsh-menu-exit').innerHTML=""
6617     document.getElementById('GshStatement').open=false
6618     GshFeatures.open = false
6619     document.getElementById('html-src').open=false
6620     document.getElementById(E_GINDEX).open=false
6621     document.getElementById(E_GOCODE).open=false
6622     document.getElementById(E_TODO).open=false
6623     document.getElementById('references').open=false
6624   }else{
6625     e.innerHTML = "Fold"
6626     document.getElementById('GshStatement').open=true
6627     GshFeatures.open = true
6628     document.getElementById(E_GINDEX).open=true
6629     document.getElementById(E_GOCODE).open=true
6630     document.getElementById(E_TODO).open=true
6631     document.getElementById('references').open=true
6632   }
6633 }
6634 function html_pure(e){
6635   if( e.innerHTML == "Pure" ){
6636     document.getElementById('gsh').style.display=true
6637     //document.style.display = false
6638     e.innerHTML = "Unpure"
6639   }else{
6640     document.getElementById('gsh').style.display=false
6641     //document.style.display = true
6642     e.innerHTML = "Pure"
6643   }
6644 }
6645
6646 var bannerIsStopping = false
6647 //NOTE: .com/JSEEF/prop_style_backgroundposition.asp
6648 function shiftBG(){
6649   bannerIsStopping = !bannerIsStopping
6650   bannerStyle.backgroundPosition = "0 0";
6651 }
6652 // status should be inherited on Window Fork(), so use the status in DOM
6653 function html_stop(e,toggle){
6654   if( toggle ){
6655     if( e.innerHTML == "Stop" ){
6656       bannerIsStopping = true
6657       e.innerHTML = "Start"
6658     }else{
6659       bannerIsStopping = false
6660       e.innerHTML = "Stop"
6661     }
6662   }else{
6663     // update JavaScript variable from DOM status
6664     if( e.innerHTML == "Stop" ){ // shown if it's running
6665       bannerIsStopping = false
6666     }else{
6667       bannerIsStopping = true
6668     }
6669   }
6670 }
6671 html_stop(document.getElementById('GshMenuStop'),false) // onInit.
6672 //html_stop(bannerElem(),false) // onInit.
6673
6674 //https://www.w3schools.com/jseef/met_win_setinterval.asp
6675 function shiftBanner(){
6676   var now = new Date().getTime();
6677   //"console.log("now"+(now%10))
6678   if( !bannerIsStopping ){
6679     bannerStyle.backgroundPosition = ((now/10)%100000)+" 0";
6680   }
6681 }
6682 window.setInterval(shiftBanner,10); // onInit.
6683
6684 // <a href="https://developer.mozilla.org/ja/docs/Web/API/Window/open">window.open()</a>
6685 // from embedded html to standalone page
6686 var MyChildren = 0
6687 function html_fork(){
6688   GJFactory_Destroy()
6689   MyChildren += 1
6690   WinId = document.getElementById('gsh-WinId').innerHTML + "." + MyChildren;
6691   newwin = window.open("",WinId,"");
6692   src = document.getElementById("gsh");
6693   srchtml = src.outerHTML
6694   newwin.document.write('/<+"html>\n');
6695   newwin.document.write(srchtml);
6696   newwin.document.write("<+"/html>\n");

```

```

6697 newwin.document.getElementById('gsh-menu-exit').innerHTML = "Close";
6698 newwin.document.getElementById('gsh-WinId').innerHTML = WinId;
6699 newwin.document.close();
6700 newwin.focus();
6701 }
6702 function html_close(){
6703   window.close()
6704 }
6705 function win_jump(win){
6706   //win = window.top;
6707   win = window.opener; // https://developer.mozilla.org/ja/docs/Web/API/window.opener
6708   if( win == null ){
6709     console.log("jump to window.opener(\"+win+)\")\n")
6710   }else{
6711     console.log("jump to window.opener(\"+win+)\")\n")
6712     win.focus();
6713   }
6714 }
6715
6716 // 0.2.9 2020-0902 created checksum of HTML
6717 CRC32UNIX = 0x04C1DB7 // Unix cksum
6718 function byteCRC32add(bigcrc,octstr,octlen){
6719   var crc = new Int32Array(1)
6720   crc[0] = bigcrc
6721
6722   let oi = 0
6723   for( ; oi < octlen; oi++ ){
6724     var oct = new Int8Array(1)
6725     oct[0] = octstr[oi]
6726     for( bi = 0; bi < 8; bi++ ){
6727       //console.log("--CRC32 "+crc[0]+" "+oct[0].toString(16)+" ["+oi+"."+bi+"]\n")
6728       ovf1 = crc[0] < 0 ? 1 : 0
6729       ovf2 = oct[0] < 0 ? 1 : 0
6730       ovf = ovf1 ^ ovf2
6731       oct[0] <= 1
6732       crc[0] <= 1
6733       if( ovf ){ crc[0] ^= CRC32UNIX }
6734     }
6735   }
6736   //console.log("--CRC32 byteAdd return crc="+crc[0]+", "+oi+"/"+octlen+"\n")
6737   return crc[0];
6738 }
6739 function strCRC32add(bigcrc,str,rlen){
6740   var crc = new Uint32Array(1)
6741   crc[0] = bigcrc
6742   var code = new Uint8Array(rlen);
6743   for( i = 0; i < rlen; i++){
6744     code[i] = str.charCodeAt(i) // not charAt() !!!!
6745     //console.log("== "+code[i].toString(16)+" <== "+str[i]+\n")
6746   }
6747   crc[0] = byteCRC32add(crc,code,rlen)
6748   //console.log("--CRC32 strAdd return crc="+crc[0]+\n")
6749   return crc[0]
6750 }
6751 function byteCRC32end(bigcrc,len){
6752   var crc = new Uint32Array(1)
6753   crc[0] = bigcrc
6754   var slen = new Uint8Array(4)
6755   let li = 0
6756   for( ; li < 4; ){
6757     selen[li] = len
6758     li += 1
6759     len >= 8
6760     if( len == 0 ){
6761       break
6762     }
6763   }
6764   crc[0] = byteCRC32add(crc[0],slen,li)
6765   crc[0] ^= 0xFFFFFFFF
6766   return crc[0]
6767 }
6768 function strCRC32(stri,len){
6769   var crc = new Uint32Array(1)
6770   crc[0] = 0
6771   crc[0] = strCRC32add(0,stri,len)
6772   crc[0] = byteCRC32end(crc[0],len)
6773   //console.log("--CRC32 "+crc[0]+" "+len+"\n")
6774   return crc[0]
6775 }
6776 function getSourceText(){
6777   version = document.getElementById('GshVersion').innerHTML
6778   sfavicon = document.getElementById('GshFaviconURL').href;
6779   sbanner = document.getElementById('GshBanner').style.backgroundImage;
6780   spositi = document.getElementById('GshBanner').style.backgroundPosition;
6781   sfooter = document.getElementById('GshFooter').style.backgroundImage;
6782
6783   if( document.getElementById('GJC_1') != null ) { GJC_1.remove() }
6784
6785   // these should be removed by CSS selector or class, after seavaed to non-printed attribute
6786   GshBanner.removeAttribute('style');
6787   GshFooter.removeAttribute('style');
6788   document.getElementById('GshMenuSign').removeAttribute("style");
6789   styleGMenu = GMenu.getAttribute("style")
6790   GMENU.removeAttribute("style");
6791   styleGStat = GStat.getAttribute("style")
6792   GStat.removeAttribute("style");
6793   styleGTop = GTop.getAttribute("style")
6794   GTop.removeAttribute("style");
6795   styleGshGrid = GshGrid.getAttribute("style")
6796   GshGrid.removeAttribute("style");
6797   //styleGPos = GPos.getAttribute("style");
6798   //GPos.removeAttribute("style");
6799   //GPos.innerHTML = "";
6800   //styleGLog = GLog.getAttribute("style");
6801   //GLog.removeAttribute("style");
6802   //GLog.innerHTML = "";
6803   styleGshellPlane = GshellPlane.getAttribute("style")
6804   GshellPlane.removeAttribute("style");
6805   styleRawTextViewer = RawTextViewer.getAttribute("style")
6806   RawTextViewer.removeAttribute("style")
6807   styleRawTextViewerClose = RawTextViewerClose.getAttribute("style")
6808   RawTextViewerClose.removeAttribute("style")
6809
6810   GshFaviconURL.href = ""
6811
6812   //it seems that interHTML and outerHTML generate style="" for these (??)
6813   //GshBanner.removeAttribute('style');
6814   //GshFooter.removeAttribute('style');
6815   //GshMenuSign.removeAttribute('style');
6816   GshBanner.style=""
6817   GshFooter.style=""
6818   GshMenuSign.style=""
6819
6820   textarea = document.createElement("textarea")

```

```

6821 srchtml = document.getElementById("gsh").outerHTML;
6822 //<textarea = document.createElement("textarea")
6823 // 2020-0910 ?? ... this causes inserting style="" to Banner and Footer,
6824 // with Chromium?? after reloading from file:///
6825 textarea.innerHTML = srchtml
6826 // <a href="https://stackoverflow.com/questions/5796718/html-entity-decode">Thanks</a>
6827 var rawtext = textarea.value
6828 //textarea.destroy()
6829 //rawtex = gsh.textContent // this removes #include <FILENAME> too
6830 var orgtext = ""
6831 + "*"+html>\n" // lost preamble text
6832 + rawtext
6833 + "<"+"/html>\n" // lost trail text
6834 ;
6835
6836 tlen = orgtext.length
6837 //console.log("getSourceText: length="+tlen+"\n")
6838 document.getElementById('GshFaviconURL').href = sfavico;
6839
6840 document.getElementById('GshBanner').style.backgroundImage = sbanner;
6841 document.getElementById('GshBanner').style.backgroundPosition = spositi;
6842 document.getElementById('GshFooter').style.backgroundImage = sfooter;
6843
6844 GStat.setAttribute("style",styleGStat)
6845 GMenu.setAttribute("style",styleMenu)
6846 GTop.setAttribute("style",styleGTop)
6847 //GLog.setAttribute("style",styleGLog)
6848 //GPos.setAttribute("style",styleGPos)
6849 GshGrid.setAttribute("style",styleGshGrid)
6850 GShellPlane.setAttribute("style",styleGshellPlane)
6851 RawTextViewer.setAttribute("style",styleRawTextViewer)
6852 RawTextViewerClose.setAttribute("style",styleRawTextViewerClose)
6853 canontext = orgtext.replace(' style=""','');
6854 // open="" too
6855
6856 }
6857 function getDigest(){
6858 var text = ""
6859 text = getSourceText()
6860 var digest = ""
6861 tlen = text.length
6862 digest = strCRC32(text,tlen) + " " + tlen
6863 return { text, digest }
6864 }
6865 function html_digest(){
6866 version = document.getElementById('GshVersion').innerHTML
6867 let {text, digest} = getDigest()
6868 alert("cksum: " + digest + " " + version)
6869 }
6870 function charsin(stri,char){
6871 ln = 0;
6872 for( i = 0; i < stri.length; i++ ){
6873 if( stri.charCodeAt(i) == char.charCodeAt(0) )
6874 ln++;
6875 }
6876 return ln;
6877 }
6878
6879 //class digestElement extends HTMLElement { }
6880 //< script>customElements.define('digest',digestElement)< /script>
6881 function showDigest(e){
6882 result = 'version=' + GshVersion.innerHTML + '\n'
6883 result += 'lines=' + e.dataset.lines + '\n'
6884 + 'length=' + e.dataset.length + '\n'
6885 + 'crc32u=' + e.dataset.crc32u + '\n'
6886 + 'time=' + e.dataset.time + '\n';
6887
6888 alert(result)
6889 }
6890
6891 function html_sign(e){
6892 if( RawTextViewer.style.zIndex == 1000 ){
6893 hideRawTextViewer()
6894 return
6895 }
6896 GJFactory_Destroy()
6897 //gsh_digest._innerHTML = "";
6898 text = getSourceText() // the original text
6899 tlen = text.length
6900 digest = strCRC32(text,tlen)
6901 //gsh_digest._innerHTML = digest + " " + tlen
6902 //text = getSourceText() // the text with its digest
6903 Lines = charsin(text,'\n')
6904
6905 name = "gsh"
6906 sid = name + "-digest"
6907 d = new Date()
6908 signedAt = d.getTime()
6909
6910 sign = '/'+*<'+'+span\n'
6911 + ' id="'+ sid + '"\n'
6912 + ' class="digest"\n'
6913 + ' data-target-id="'+name+'\n'
6914 + ' data-crc32u="'+ digest + '"\n'
6915 + ' data-length="'+ tlen + '"\n'
6916 + ' data-lines="'+ Lines + '"\n'
6917 + ' data-time="'+ signedAt + '"\n'
6918 + '>' + '/span>\n*'+/\n'
6919
6920 text = sign + text
6921
6922 txthtml = '<' + 'table id="LineNumbered"><' + 'tr><' + 'td>'
6923 + '<' + 'textarea cols=5 rows=' + Lines + ' class="LineNumbered">'
6924 for( i = 1; i <= Lines; i++ ){
6925 txthml += i.toString() + '\n'
6926 }
6927 txthml += "
6928 + '<' + '/textarea>'
6929 + '<' + '/td>' + 'td>
6930 + '<' + 'textarea cols=150 rows=' + Lines + ' spellcheck="false"'
6931 + ' class="LineNumbered">'
6932 + text + '</' + '/textarea>
6933 + '<' + '/td>' + '/tr><' + '/table>
6934
6935 for( i = 1; i <= 30; i++ ){
6936 txthml += '<br>\n'
6937 }
6938 RawTextViewer.innerHTML = txthml
6939
6940 btn = e
6941 e.style.color = "rgba(128,128,255,0.9)";
6942 y = e.getBoundingClientRect().top.toFixed(0)
6943 //h = e.getBoundingClientRect().height.toFixed(0)
6944 RawTextViewer.style.top = Number(y) + 30

```

```

6945 RawTextViewer.style.left = 100;
6946 RawTextViewer.style.height = window.innerHeight - 20;
6947 //RawTextViewer.style.opacity = 1.0;
6948 //RawTextViewer.style.backgroundColor = "rgba(0,0,0,0.0)";
6949 RawTextViewer.style.backgroundColor = "rgba(255,255,255,0.8)";
6950 RawTextViewer.style.zIndex = 1000;
6951 RawTextViewer.style.display = true;
6952
6953 if( RawTextViewerClose.style == null ){
6954   RawTextViewerClose.style = "";
6955 }
6956 RawTextViewerClose.style.top = Number(y) + 10;
6957 RawTextViewerClose.style.left = 100;
6958 RawTextViewerClose.style.zIndex = 1001;
6959
6960 ScrollToElement(CurElement,RawTextViewerClose)
6961 }
6962 function hideRawTextViewer(){
6963   RawTextViewer.style.left = 10000;
6964   RawTextViewer.style.zIndex = -100;
6965   RawTextViewer.style.opacity = 0.0;
6966   RawTextViewer.style = null;
6967   RawTextViewer.innerHTML = "";
6968
6969 GshMenuSign.style.color = "rgba(255,128,128,1.0)";
6970 RawTextViewerClose.style.top = 0;
6971 RawTextViewerClose.style = null;
6972 }
6973
6974 // source code viewer
6975 function frame_close(){
6976   srcframe = document.getElementById("src-frame");
6977   srcframe.innerHTML = "";
6978   //srcframe.style.cols = 1;
6979   srcframe.style.rows = 1;
6980   srcframe.style.height = 0;
6981   srcframe.style.display = false;
6982   src = document.getElementById("src-frame-textarea");
6983   src.innerHTML = "";
6984   //src.cols = 0
6985   src.rows = 0
6986   src.display = false
6987   //alert("--closed--")
6988 }
6989 //<!-- | <span onclick="html_view();">Source</span> -->
6990 //<!-- | <span onclick="frame_close();">SourceClose</span> -->
6991 //<!--| <span>Download</span>-->
6992 function frame_open(){
6993   document.getElementById('GshFaviconURL').href = "";
6994   oldsrc = document.getElementById("GENSRC");
6995   if( oldsrc != null ){
6996     //alert("--I--(erasing old text)")
6997     oldsrc.innerHTML = "";
6998     return;
6999   }else{
7000     //alert("--I--(no old text)")
7001   }
7002   styleBanner = GshBanner.getAttribute("style")
7003   GshBanner.removeAttribute("style")
7004   styleFooter = GshFooter.getAttribute("style")
7005   GshFooter.removeAttribute("style")
7006   if( document.getElementById('GJC_1') ) { GJC_1.remove() }
7007
7008 GshFaviconURL.href = "";
7009 GStat.removeAttribute('style')
7010 GshGrid.removeAttribute('style')
7011 GshMenuSign.removeAttribute('style')
7012 //GPos.removeAttribute('style')
7013 //GPos.innerHTML = "";
7014 //GLog.removeAttribute('style')
7015 //GLog.innerHTML = "";
7016 GMenu.removeAttribute('style')
7017 GTop.removeAttribute('style')
7018 GShellPlane.removeAttribute('style')
7019 RawTextViewer.removeAttribute('style')
7020 RawTextViewerClose.removeAttribute('style')
7021
7022 GJFactory_Destroy()
7023
7024 src = document.getElementById("gsh");
7025 srchtml = src.outerHTML
7026 srcframe = document.getElementById("src-frame");
7027 srcframe.innerHTML = ""
7028 + "<+"+cite id=\\"GENSRC\\\">\n"
7029 + "<+"+style>\n"
7030 + "#GENSRC textarea{tab-size:4;}\n"
7031 + "#GENSRC textarea{-o-tab-size:4;}\n"
7032 + "#GENSRC textarea{-moz-tab-size:4;}\n"
7033 + "#GENSRC textarea{spellcheck:false;}\n"
7034 + "</"+"style>\n"
7035 + "<+"+textarea id=\\"src-frame-textarea\\\" cols=100 rows=20 class=\\"gsh-code\\\">\n"
7036 + "/<+"+html>\n" // lost preamble text
7037 + srchtml
7038 + "<+"+html>\n" // lost trail text
7039 + "<+"+textarea>\n"
7040 + "<+"+cite><!-- GENSRC -->\n";
7041
7042 //srcframe.style.cols = 80;
7043 //srcframe.style.rows = 80;
7044
7045 GshBanner.setAttribute('style',styleBanner)
7046 GshFooter.setAttribute('style',styleFooter)
7047 }
7048 function fill_CSSView(){
7049   part = document.getElementById('GshStyleDef')
7050   view = document.getElementById('gsh-style-view')
7051   view.innerHTML = ""
7052   + "<+"+textarea cols=100 rows=20 class=\\"gsh-code\\\">\n"
7053   + part.innerHTML
7054   + "<+"+textarea>\n"
7055 }
7056 function fill_JavaScriptView(){
7057   jspart = document.getElementById('gsh-script')
7058   view = document.getElementById('gsh-script-view')
7059   view.innerHTML = ""
7060   + "<+"+textarea cols=100 rows=20 class=\\"gsh-code\\\">\n"
7061   + jspart.innerHTML
7062   + "<+"+textarea>\n"
7063 }
7064 function fill_DataView(){
7065   part = document.getElementById('gsh-data')
7066   view = document.getElementById('gsh-data-view')
7067   view.innerHTML = ""
7068   + "<+"+textarea cols=100 rows=20 class=\\"gsh-code\\\">\n"

```

```

7069     + part.innerHTML
7070     + "<+""/textarea>" 
7071 }
7072 function jumpTo_StyleView(){
7073     jsview = document.getElementById('html-src')
7074     jsview.open = true
7075     jsview = document.getElementById('gsh-style-frame')
7076     jsview.open = true
7077     fill_CSSView()
7078 }
7079 function jumpTo_JavaScriptView(){
7080     jsview = document.getElementById('html-src')
7081     jsview.open = true
7082     jsview = document.getElementById('gsh-script-frame')
7083     jsview.open = true
7084     fill_JavaScriptView()
7085 }
7086 function jumpTo_DataView(){
7087     jsview = document.getElementById('html-src')
7088     jsview.open = true
7089     jsview = document.getElementById('gsh-data-frame')
7090     jsview.open = true
7091     fill_DataView()
7092 }
7093 function jumpTo_WholeView(){
7094     jsview = document.getElementById('html-src')
7095     jsview.open = true
7096     jsview = document.getElementById('gsh-whole-view')
7097     jsview.open = true
7098     frame_open()
7099 }
7100 function html_view(){
7101     html_stop();
7102 }
7103 banner = document.getElementById('GshBanner').style.backgroundImage;
7104 footer = document.getElementById('GshFooter').style.backgroundImage;
7105 document.getElementById('GshBanner').style.backgroundImage = "";
7106 document.getElementById('GshBanner').style.backgroundPosition = "";
7107 document.getElementById('GshFooter').style.backgroundImage = "";
7108
7109 //srcwin = window.open("", "CodeView2", "");
7110 srcwin = window.open("", "", "");
7111 srcwin.document.write("<span id=\\"gsh\\">\n");
7112
7113 src = document.getElementById("gsh");
7114 srcwin.document.write("<"+style>\n");
7115 srcwin.document.write("textarea{tab-size:4;}\n");
7116 srcwin.document.write("textarea{o-tab-size:4;}\n");
7117 srcwin.document.write("textarea{moz-tab-size:4;}\n");
7118 srcwin.document.write("</style>\n");
7119 srcwin.document.write("<h2>\n");
7120 srcwin.document.write("<"+span onclick=\\"window.close();\\>Close</span> | \n");
7121 //srcwin.document.write("<"+span onclick=\\"html_stop();\\>Run</span>\n");
7122 srcwin.document.write("</h2>\n");
7123 srcwin.document.write("<textarea id=\\"gsh-src-src\\" cols=100 rows=60>");
7124 srcwin.document.write("<"+html>\n");
7125 srcwin.document.write(span id=\\"gsh\\>"); 
7126 srcwin.document.write(src.innerHTML);
7127 srcwin.document.write("<"+span<"+/html>\n");
7128 srcwin.document.write("</"+"textarea>\n");
7129
7130 document.getElementById('GshBanner').style.backgroundImage = banner;
7131 document.getElementById('GshFooter').style.backgroundImage = footer;
7132
7133 sty = document.getElementById("GshStyleDef");
7134 srcwin.document.write("<"+style>\n");
7135 srcwin.document.write(sty.innerHTML);
7136 srcwin.document.write("<"+style>\n");
7137
7138 run = document.getElementById("gsh-script");
7139 srcwin.document.write("<"+script>\n");
7140 srcwin.document.write(run.innerHTML);
7141 srcwin.document.write("<"+script>\n");
7142
7143 srcwin.document.write("<"+span><"+/html>\n"); // gsh span
7144 srcwin.document.close();
7145 srcwin.focus();
7146 }
7147 GSH = document.getElementById("gsh")
7148
7149 //GSH.onclick = "alert('Ouchi!')"
7150 //GSH.css = "{background-color:#eef;}"
7151 //GSH.style = "background-color:#eef;";
7152 //GSH.style.display = false;
7153 //alert('Ouchi!')
7154 //GSH.style.display = true;
7155
7156 // 2020-0904 created, tentative
7157 document.addEventListener('keydown',jgshCommand);
7158 //CurElement = GshStatement
7159 CurElement = GshMenu
7160 MemElement = GshMenu
7161
7162 function nextSib(e){
7163     n = e.nextSibling;
7164     for( i = 0; i < 100; i++ ){
7165         if( n == null ){
7166             break;
7167         }
7168         if( n.nodeName == "DETAILS" ){
7169             return n;
7170         }
7171         n = n.nextSibling;
7172     }
7173     return null;
7174 }
7175 function prevSib(e){
7176     n = e.previousSibling;
7177     for( i = 0; i < 100; i++ ){
7178         if( n == null ){
7179             break;
7180         }
7181         if( n.nodeName == "DETAILS" ){
7182             return n;
7183         }
7184         n = n.previousSibling;
7185     }
7186     return null;
7187 }
7188 function setColor(e,eName,eColor){
7189     if( e.hasChildNodes() ){
7190         s = e.childNodes;
7191         if( s != null ){
7192             for( ci = 0; ci < s.length; ci++ ){

```

```

7193         if( s[ci].nodeName == eName ){
7194             s[ci].style.color = eColor;
7195             //s[ci].style.backgroundColor = eColor;
7196             break;
7197         }
7198     }
7199   }
7200 }
7201 }
7202
7203 // https://docs.microsoft.com/en-us/previous-versions//hh781509(v=vs.85)
7204 function showCurElementPosition(ev){
7205 //  if( document.getElementById("GPos") == null ){
7206 //    return;
7207 //  }
7208 //  if( GPos == null ){
7209 //    return;
7210 //  }
7211   e = CurElement
7212   y = e.getBoundingClientRect().top.toFixed(0)
7213   x = e.getBoundingClientRect().left.toFixed(0)
7214
7215   h = ev + " "
7216   h += "y='"+y+"', "+ 'x=' +x+" -- "
7217   h += "w=" + window.innerWidth + ", h=" + window.innerHeight + " -- "
7218   //GPos.test = h
7219   //GPos.innerHTML = h
7220   // GPos.innerHTML = h
7221 }
7222
7223 function DateShort(){
7224   d = new Date()
7225   return d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7226   + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7227 }
7228 function DateLong(){
7229   d = new Date()
7230   return d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7231   + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7232   + " " + d.getMilliseconds()
7233   + " " + d.getTimezoneOffset()/60
7234   + " " + d.getTime() + "." + d.getMilliseconds()
7235
7236 }
7237
7238 function GShellMenu(e){
7239   //GLog.innerHTML = "Hello, World! (" + DateLong() + ")"
7240   showGShellPlane()
7241 }
7242 // placements of planes
7243 function GshellResizeX(ev){
7244   //if( document.getElementById("GMenu") != null ){
7245     GMenu.style.left = window.innerWidth - 100
7246     GMenu.style.top = window.innerHeight - 90 - 200
7247   //console.log("place GMENU "+GMenu.style.left+" "+GMenu.style.top)
7248
7249   //}
7250   GStat.style.width = window.innerWidth
7251   //if( document.getElementById("GPos") != null ){
7252     //GPos.style.width = window.innerWidth
7253     //GPos.style.top = window.innerHeight - 30; //GPos.style.height
7254   //}
7255   //if( document.getElementById("GLog") != null ){
7256     // GLog.style.width = window.innerWidth
7257     //GLog.innerHTML = ""
7258   //}
7259   //if( document.getElementById("GLog") != null ){
7260     //GLog.innerHTML = "Resize: w=" + window.innerWidth +
7261     //", h=" + window.innerHeight
7262   //}
7263   showCurElementPosition(ev)
7264 }
7265 function GShellResize(){
7266   GshellResizeX("RESIZE")
7267 }
7268 window.onresize = GShellResize
7269 var prevNode = null
7270 function GJSH_OnMouseMove(ev){
7271   x = ev.clientX
7272   y = ev.clientY
7273   d = new Date()
7274   t = d.getTime() / 1000
7275   if( document.elementFromPoint() ){
7276     e = document.elementFromPoint(x,y)
7277     if( e != null ){
7278       if( e == prevNode ){
7279         console.log(t+'('+x+', '+y+') ' +
7280           +e.nodeType+' '+e.tagName+'#'+e.id)
7281         prevNode = e
7282       }
7283     } else{
7284       console.log(t+'('+x+', '+y+') no element')
7285     }
7286   } else{
7287     console.log(t+'('+x+', '+y+') no elementFromPoint')
7288   }
7289 }
7290 window.addEventListener('mousemove',GJSH_OnMouseMove);
7291
7292 function GJSH_OnMouseMoveScreen(ev){
7293   x = ev.screenX
7294   y = ev.screenY
7295   d = new Date()
7296   t = d.getTime() / 1000
7297   console.log(t+'('+x+', '+y+') no elementFromPoint')
7298 }
7299 //screen.addEventListener('mousemove',GJSH_OnMouseMoveScreen);
7300
7301 function ScrollToElement(oe,ne){
7302   ne.scrollIntoView()
7303   ny = ne.getBoundingClientRect().top.toFixed(0)
7304   nx = ne.getBoundingClientRect().left.toFixed(0)
7305   //GLog.innerHTML = "["+ny+","+nx+"]"
7306   //window.scrollTo(0,0)
7307
7308   GTop.style.backgroundColor = "rgba(0,0,0,0.0)"
7309   GshGrid.style.left = '250px';
7310   GshGrid.style.zIndex = 0
7311
7312   if( false ){
7313     oy = oe.getBoundingClientRect().top.toFixed(0)
7314     ox = oe.getBoundingClientRect().left.toFixed(0)
7315     y = e.getBoundingClientRect().top.toFixed(0)
7316     x = e.getBoundingClientRect().left.toFixed(0)

```

```

7317     window.scrollTo(x,y)
7318     ny = e.getBoundingClientRect().top.toFixed(0)
7319     nx = e.getBoundingClientRect().left.toFixed(0)
7320     //GLog.innerHTML = "["+oy+","+ox+"]->["+y+","+x+"]->["+ny+","+nx+"]"
7321   }
7322 }
7323 function showGShellPlane(){
7324   if( GShellPlane.style.zIndex == 0 ){
7325     GShellPlane.style.zIndex = 1000;
7326     GShellPlane.style.left = 30;
7327     GShellPlane.style.height = 320;
7328     GShellPlane.innerHTML = DateLong() + "<br>" +
7329       "-- History --<br>" + MyHistory;
7330   }else{
7331     GShellPlane.style.zIndex = 0;
7332     GShellPlane.style.left = 0;
7333     GShellPlane.style.height = 50;
7334     GShellPlane.innerHTML = "";
7335   }
7336 }
7337 var SuppressGJShell = false
7338 function jgshCommand(keyevent){
7339   if( SuppressGJShell ){
7340     return
7341   }
7342   key = keyevent
7343   keycode = key.code
7344   //GStat.style.width = window.innerWidth
7345   GStat.style.backgroundColor = "rgba(0,0,0,0.4)"
7346
7347   console.log("JGSh-Key:"+keycode+"(^~)//")
7348   if( keycode == "Slash" ){
7349     console.log('('+'x+', '+'y+')')
7350     e = document.elementFromPoint(x,y)
7351     console.log('('+'x+', '+'y+')' +e.nodeType+ ' '+e.tagName+' #' +e.id)
7352   }else
7353   if( keycode == "Digit0" ){ // fold side-bar
7354     // "Zero page"
7355     showGShellPlane();
7356   }else
7357   if( keycode == "Digit1" ){ // fold side-bar
7358     primary.style.width = "94%"
7359     secondary.style.width = "0%"
7360     secondary.style.opacity = 0
7361     GStat.innerHTML = "[Single Column View]"
7362   }else
7363   if( keycode == "Digit2" ){ // unfold side-bar
7364     primary.style.width = "58%"
7365     secondary.style.width = "36%"
7366     secondary.style.opacity = 1
7367     GStat.innerHTML = "[Double Column View]"
7368   }else
7369   if( keycode == "KeyU" ){ // fold/unfold all
7370     html_fold(GshMenuFold);
7371     location.href = "#"+CurElement.id;
7372   }else
7373   if( keycode == "KeyO" || keycode == "ArrowRight" ){ // fold the element
7374     CurElement.open = !CurElement.open;
7375   }else
7376   if( keycode == "ArrowRight" ){ // unfold the element
7377     CurElement.open = true
7378   }else
7379   if( keycode == "ArrowLeft" ){ // unfold the element
7380     CurElement.open = false
7381   }else
7382   if( keycode == "KeyI" ){ // inspect the element
7383     e = CurElement
7384     //GLog.innerHTML =
7385     GJLog_append("Current Element: " + e + "<br>" +
7386       + "name='"+e.nodeName + ",'" +
7387       + "id='"+e.id + ",'" +
7388       + "children='"+e.childNodes.length + ",'" +
7389       + "parent='"+e.parentNode.id + "<br>" +
7390       + "text='"+e.textContent)
7391     GStat.style.backgroundColor = "rgba(0,0,0,0.8)"
7392     return
7393   }else
7394   if( keycode == "KeyM" ){ // memory the position
7395     MemElement = CurElement
7396   }else
7397   if( keycode == "KeyN" || keycode == "ArrowDown" ){ // next element
7398     e = nextSib(CurElement)
7399     if( e != null ){
7400       setColor(CurElement,"SUMMARY","#ffff")
7401       setColor(e,"SUMMARY","#8f8") // should be complement ?
7402       oe = CurElement
7403       CurElement = e
7404       //location.href = "#"+e.id;
7405       ScrollToElement(oe,e)
7406     }
7407   }else
7408   if( keycode == "KeyP" || keycode == "ArrowUp" ){ // previous element
7409     oe = CurElement
7410     e = prevSib(CurElement)
7411     if( e != null ){
7412       setColor(CurElement,"SUMMARY","#ffff")
7413       setColor(e,"SUMMARY","#8f8") // should be complement ?
7414       CurElement = e
7415       //location.href = "#"+e.id;
7416       ScrollToElement(oe,e)
7417     }
7418   }else
7419   if( e != null ){
7420     setColor(CurElement,"SUMMARY","#ffff")
7421     CurElement = e
7422     ScrollToElement(oe,e)
7423   }else{
7424     e = document.getElementById("GshBanner")
7425     if( e != null ){
7426       setColor(CurElement,"SUMMARY","#ffff")
7427       CurElement = e
7428       ScrollToElement(oe,e)
7429     }
7430   }
7431 }else
7432 if( keycode == "KeyR" ){
7433   location.reload()
7434 }else
7435 if( keycode == "KeyJ" ){
7436   GshGrid.style.top = '120px';
7437   GshGrid.innerHTML = '<>_{Down}';
7438 }else
7439 if( keycode == "KeyK" ){
7440

```

```

7441     GshGrid.style.top = '0px';
7442     GshGrid.innerHTML = '(^-^){Up}';
7443   }else
7444     if( keycode == "KeyH" ){
7445       GshGrid.style.left = '0px';
7446       GshGrid.innerHTML = "(_){Left}";
7447     }else
7448     if( keycode == "KeyL" ){
7449       //GLog.innerHTML +=
7450       GJLog_append(
7451         'screen='+screen.width+'px'+'<br>'+
7452         'window='+window.innerWidth+'px'+'<br>' +
7453       )
7454       GshGrid.style.left = (document.documentElement.clientWidth-160).toString(10)+'px';
7455       GshGrid.innerHTML = '(@_@){Right}';
7456     }else
7457     if( keycode == "Keys" ){
7458       html_stop(GshMenuStop,true);
7459     }else
7460     if( keycode == "KeyF" ){
7461       html_fork();
7462     }else
7463     if( keycode == "KeyC" ){
7464       window.close();
7465     }else
7466     if( keycode == "KeyD" ){
7467       html_digest();
7468     }else
7469     if( keycode == "KeyV" ){
7470       e = document.getElementById('gsh-digest')
7471       if( e != null ){
7472         showDigest(e)
7473       }
7474     }
7475   showCurElementPosition("[+key.code+" ] --");
7476   //if( document.getElementById("GPos") != null ){
7477   //  GPos.innerHTML += "[+key.code+" ] --"
7478   //}
7479   //GShellResizeX("[+key.code+" ] --");
7480 }
7481 GShellResizeX("[INIT]");
7482
7483 DisplaySize = '-- Display: '+ 'screen=' + screen.width +'px', '+window=' + window.innerWidth +'px';
7484
7485 let {text, digest} = getDigest()
7486 //GLog.innerHTML +=
7487 GJLog_append(
7488   '-- GShell: ' + GshVersion.innerHTML + '\n' +
7489   '-- Digest: ' + digest + '\n' +
7490   DisplaySize
7491   //+ "<br>" + "-- LastVisit:<br>" + MyHistory
7492 )
7493 GShellResizeX(null);
7494
7495 // <a href="https://www.w3.org/TR/WebCryptoAPI/">Web Cryptography API</a>
7496 //Convert a string into an ArrayBuffer
7497 //from https://developers.google.com/web/updates/2012/06/How-to-convert-ArrayBuffer-to-and-from-String
7498 function str2ab(str) {
7500   const buf = new ArrayBuffer(str.length);
7501   const bufView = new Uint8Array(buf);
7502   for (let i = 0, strLen = str.length; i < strLen; i++) {
7503     bufView[i] = str.charCodeAt(i);
7504   }
7505   return buf;
7506 }
7507 function importPrivateKey(pem) {
7508   const binaryDerString = window.atob(pemContents);
7509   const binaryDer = str2ab(binaryDerString);
7510   return window.crypto.subtle.importKey(
7511     "pkcs8",
7512     binaryDer,
7513     {
7514       name: "RSA-PSS",
7515       modulusLength: 2048,
7516       publicExponent: new Uint8Array([1, 0, 1]),
7517       hash: "SHA-256",
7518     },
7519     true,
7520     ["sign"]
7521   );
7522 }
7523 //importPrivateKey(ppem);
7524
7525 //key = {}
7526 //buf =
7527 //enc = "xyxxxxxxxx"; //crypto.publicEncrypt(key,buf)
7528 //b64 = btoa(enc)
7529 //dec = atob(b64)
7530 //GLog.innerHTML = "enc:" + b64 + ", dec:" + dec
7531
7532 </script>
7533
7534 <span id="gjc" data-title="GJConsole" data-author="sato@its-more.jp">
7535 <!-- ----- GJConsole BEGIN { ----- -->
7536 <p>
7537 <span id="GJE_RootNode0"></span>
7538 </p>
7539 <style id="GJConsoleStyle">
7540   .GJConsole {
7541     z-index:1000;
7542     width:400; height:200px;
7543     margin:2px;
7544     color:#fff; background-color:#66a;
7545     font-size:12px; font-family:monospace,Courier New;
7546   }
7547 </style>
7548
7549 <script id="GJConsoleScript" class="GJConsole">
7550   var PS1 = "% "
7551   function GJC_KeyDown(keyevent){
7552     key = keyevent.code
7553     if( key == "Enter" ){
7554       GJC_Command(this)
7555       this.value += "\n" + PS1 // prompt
7556     }else
7557     if( key == "Escape"){
7558       SuppressGJShell = false
7559       GshMenu.focus() // should be previous focus
7560     }
7561   }
7562   var GJC_SessionId
7563   function GJC_SetSessionId(){
7564     var xd = new Date()

```

```

7565     GJC_SessionId = xd.getTime() / 1000
7566   }
7567   GJC_SetSessionId()
7568   function GJC_Memory(mem,args,text){
7569     argv = args.split(' ')
7570     cmd = argv[0]
7571     argv.shift()
7572     args = argv.join(' ')
7573     ret = ""
7574
7575     if( cmd == 'clear' ){
7576       Permanent.setItem(mem,'')
7577     }else{
7578       if( cmd == 'read' ){
7579         ret = Permanent.getItem(mem)
7580       }else{
7581         if( cmd == 'save' ){
7582           val = Permanent.getItem(mem)
7583           if( val == null ){ val = "" }
7584           d = new Date()
7585           val += d.getTime()/1000+" "+GJC_SessionId+" "+document.URL+" "+args+"\n"
7586           val += text.value
7587           Permanent.setItem(mem,val)
7588         }else{
7589           if( cmd == 'write' ){
7590             val = Permanent.getItem(mem)
7591             if( val == null ){ val = "" }
7592             d = new Date()
7593             val += d.getTime()/1000+" "+GJC_SessionId+" "+document.URL+" "+args+"\n"
7594             Permanent.setItem(mem,val)
7595           }else{
7596             ret = "Commands: write | read | save | clear"
7597           }
7598         }
7599       }
7600     // -- 2020-09-14 added TableEditor
7601     var GJE_CurElement = null; //GJE_RootNode
7602     GJE_NodeSaved = null
7603     GJE_TableNo = 1
7604     function GJE_StyleKeyCommand(kev){
7605       keycode = kev.code
7606       console.log('GJE-Key: '+keycode)
7607       if( keycode == 'Escape' ){
7608         GJE_SetStyle(this);
7609       }
7610       kev.stopPropagation()
7611     // https://developer.mozilla.org/en-US/docs/Web/API/Event/stopPropagation
7612   }
7613   var GJE_CommandMode = false
7614   function GJE_TableKeyCommand(kev,tab){
7615     wasCmdMode = GJE_CommandMode
7616     key = kev.code
7617     if( key == 'Escape' ){
7618       console.log("To command mode: "+tab.nodeName+"#"+tab.id)
7619       //tab.setAttribute('contenteditable','false')
7620       tab.style.caretColor = "blue"
7621       GJE_CommandMode = true
7622     }else{
7623       if( key == "KeyA" ){
7624         tab.style.caretColor = "red"
7625         GJE_CommandMode = false
7626       }else{
7627         if( key == "KeyI" ){
7628           tab.style.caretColor = "red"
7629           GJE_CommandMode = false
7630         }else{
7631           if( key == "KeyO" ){
7632             tab.style.caretColor = "red"
7633             GJE_CommandMode = false
7634           }else{
7635             if( key == "KeyT" ){
7636               console.log("ROW-DOWN")
7637             }else{
7638               if( key == "KeyR" ){
7639                 console.log("ROW-UP")
7640               }else{
7641                 if( key == "KeyW" ){
7642                   console.log("COL-FORW")
7643                 }else{
7644                   if( key == "KeyB" ){
7645                     console.log("COL-BACK")
7646                   }
7647
7648                 kev.stopPropagation()
7649                 if( wasCmdMode ){
7650                   kev.preventDefault()
7651                 }
7652               }
7653             function GJE_DragEvent(ev,elem){
7654               x = ev.clientX
7655               y = ev.clientY
7656               console.log("dragged: "+this.nodeName+'#'+this.id+' x='+x+' y='+y)
7657             }
7658           // https://developer.mozilla.org/en-US/docs/Web/API/DragEvent
7659           // https://www.w3.org/TR/uievents/#events-mouseevents
7660           function GJE_DropEvent(ev,elem){
7661             x = ev.clientX
7662             y = ev.clientY
7663             this.style.x = x
7664             this.style.y = y
7665             this.style.position = 'absolute' // 'fixed'
7666             this.parentNode = gsh // just for test
7667             console.log("Dropped: "+this.nodeName+'#'+this.id+' x='+x+' y='+y
7668               +' parent='+this.parentNode.id)
7669           }
7670           function GJE_SetTableStyle(ev){
7671             this.innerHTML = this.value; // sync. for external representation?
7672             if(false){
7673               stdid = this.parentNode.id+this.id
7674               // and remove "_span" at the end
7675               e = document.getElementById(stdid)
7676               //alert('SetTableStyle #' + e.id + '\n' + this.value)
7677               if( e != null ){
7678                 e.innerHTML = this.value
7679               }else{
7680                 console.log('Style Not found: '+stdid)
7681               }
7682             } //alert('event StopPropagation: '+ev)
7683           }
7684         }
7685         function setCSSofClass(cclass,cstyle){
7686           const ss = document.styleSheets[3]; // 0, 1, 2, 3, ...
7687           rlen = ss.cssRules.length;
7688           let tabrule = null;

```

```

7689 rulex = -1
7690
7691 // should skip white space at the top of cstyle
7692 sel = cstyle.charAt(0);
7693 selector = sel+cclass;
7694 console.log('-- search style rule for '+selector)
7695
7696 for(let i = 0; i < rlen; i++){
7697   cr = ss.cssRules[i];
7698   console.log('CSS rule ['+i+'/'+rlen+'] '+cr.selectorText);
7699   if( cr.selectorText === selector ){ // css class selector
7700     tabrule = ss.cssRules[i];
7701     console.log('CSS rule found for:['+i+'/'+rlen+'] '+selector);
7702     ss.deleteRule(i);
7703     //rlen = ss.cssRules.length;
7704     rulex = i
7705     // should search and replace the property here
7706   }
7707 }
7708 // https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleSheet/insertRule
7709 if( tabrule == null ){
7710   console.log('CSS rule NOT found for:['+rlen+'] '+selector);
7711   ss.insertRule(cstyle,rlen);
7712   ss.insertRule(cstyle,0); // override by 0?
7713   console.log('CSS rule inserted:['+(rlen+1)+']\n'+cstyle);
7714 }else{
7715   ss.insertRule(cstyle,rlen);
7716   ss.insertRule(cstyle,0);
7717   console.log('CSS rule replaced:['+(rlen+1)+']\n'+cstyle);
7718 }
7719
7720 function GJE_SetStyle(te){
7721   console.log('Apply the style to:' + te.id+'\n');
7722   console.log('Apply the style to:' + te.parentNode.id+'\n');
7723   console.log('Apply the style to:' + te.parentNode.className+'\n');
7724   cclass = te.parentNode.className;
7725   setCSSofClass(cclass,te.value); // should get selector part from
7726   // selector { rules }
7727
7728 if(false){
7729   //console.log('Apply the style:');
7730   //stid = this.parentNode.id+this.id+
7731   //stid = this.id+".style"
7732   css = te.value
7733   stid = te.parentNode.id+".style"
7734   e = document.getElementById(stid)
7735   if( e != null ){
7736     //console.log('Apply the style:' + e.id+'\n' + te.value);
7737     console.log('Apply the style:' + e.id+'\n' + css);
7738   // e.innerHTML = css; //te.value;
7739   // ncss = e.sheet;
7740   // ncss.insertRule(te.value,ncss.cssRules.length);
7741   }else{
7742     console.log('No element to Apply the style: ' + stid)
7743   }
7744   tblid = te.parentNode.id+".table";
7745   e = document.getElementById(tblid);
7746   if( e != null ){
7747     //e.setAttribute('style',css);
7748     e.setProperty('style',css,'!important');
7749   }
7750 }
7751
7752 function makeTable(argv){
7753   //tid = ''
7754   cwe = GJE_CurElement
7755   tid = 'table_' + GJE_TableNo
7756
7757   nt = new Text('\n')
7758   cwe.appendChild(nt)
7759
7760   ne = document.createElement('span'); // the container
7761   cwe.appendChild(ne)
7762   ne.id = tid + '-span'
7763   ne.setAttribute('contenteditable',true)
7764
7765   hspan = document.createElement('span'); // html part
7766   //hspan.id = tid + '-html'
7767   //ne.innerHTML = '\n'
7768   nt = new Text('\n')
7769   ne.appendChild(nt)
7770   ne.appendChild(hspan)
7771
7772   hspan.id = tid
7773   hspan.setAttribute('class',tid)
7774
7775   ne.setAttribute('draggable','true')
7776   ne.addEventListener('drag',GJE_DragEvent);
7777   ne.addEventListener('dragend',GJE_DropEvent);
7778
7779   var col = 3
7780   var row = 2
7781   if( argv[0] != null ){
7782     col = argv[0]
7783     argv.shift()
7784   }
7785   if( argv[0] != null ){
7786     row = argv[0]
7787     argv.shift()
7788   }
7789
7790   //ne.setAttribute('class',tid)
7791   ht = "\n"
7792   //ht += '<+' + 'table ' + ' id="' + tid + '" ' + ' class="' + tid + '"'
7793   ht += '<+' + 'table '
7794   + ' onkeydown="GJE_KeypadCommand(event,this)"'
7795   //+ ' ondrag="GJE_DragEvent(event,this)"\n'
7796   //+ ' ondragend="GJE_DropEvent(event,this)"\n'
7797   //+ ' dragable="true"\n'
7798   //+ ' contenteditable="true"'
7799   + '>\n'
7800   ht += '<+' + 'tbody>\n';
7801   for( r = 0; r < row; r++ ){
7802     ht += "<+" + "tr>\n"
7803     for( c = 0; c < col; c++ ){
7804       ht += "<+" + "td>\n"
7805       ht += "ABCDEFGHIJKLMNOPQRSTUVWXYZ".charAt(c) + r
7806       ht += "<+" + "/td>\n"
7807     }
7808     ht += "<+" + "/tr>\n"
7809   }
7810   ht += '<+' + '/tbody>\n';
7811   ht += '<+' + '/table>\n';
7812   hspan.innerHTML = ht;

```

```

7813 nt = new Text('\n')
7814 ne.appendChild(nt)
7815
7816 st = '#'+tid+' *{\n' // # for instance specific
7817   +' '+border:1px solid #aaa;\n'
7818   +' '+background-color:#efe;\n'
7819   +' '+color:#222;\n'
7820   +' '+font-size:#14pt !important;\n'
7821   +' '+font-family:monospace,Courier New !important;\n'
7822   +' } /* hit ESC to apply */\n'
7823
7824 // wish script to be included
7825 //nj = document.createElement('script')
7826 //ne.appendChild(nj)
7827 //ne.innerHTML = 'function SetStyle(e){}\n'
7828
7829 // selector seems lost in dynamic style appending
7830 if(false){
7831   ns = document.createElement('style')
7832   ne.appendChild(ns)
7833   ns.id = tid + '.style'
7834   ns.innerHTML = '\n'+st
7835   nt = new Text('\n')
7836   ne.appendChild(nt)
7837 }
7838 setCSSofClass(tid,st); // should be in JavaScript script?
7839
7840 nx = document.createElement('textarea')
7841 ne.appendChild(nx)
7842 nx.id = tid + '-style_def'
7843 nx.setAttribute('class','GJ_StyleEditor')
7844 nx.spellcheck = false
7845 nx.cols = 60
7846 nx.rows = 10
7847 nx.innerHTML = '\n'*st
7848 nx.addEventListener('change',GJE_SetTableStyle);
7849 nx.addEventListener('keydown',GJE_StyleKeyCommand);
7850 //nx.addEventListener('click',GJE_SetTableStyle);
7851
7852 nt = new Text('\n')
7853 cwe.appendChild(nt)
7854
7855 GJE_TableNo += 1
7856 return 'created TABLE id="'+tid+'"'
7857 }
7858 function GJE_NodeEdit(argv){
7859   cwe = GJE_CurElement
7860   cmd = argv[0]
7861   argv.shift()
7862   args = argv.join(' ')
7863   ret = ""
7864
7865   if( cmd == '.u' || cmd == '.un' || cmd == 'undo' ){
7866     if( GJE_NodeSaved != null ){
7867       xn = GJE_RootNode
7868       GJE_RootNode = GJE_NodeSaved
7869       GJE_NodeSaved = xn
7870       ret = '-- did undo'
7871     }else{
7872       ret = '-- could not undo'
7873     }
7874   return ret
7875 }
7876 GJE_NodeSaved = GJE_RootNode.cloneNode()
7877 if( cmd == '.c' || cmd == '.cd' || cmd == 'cd' ){
7878   if( argv[0] == null ){
7879     ne = GJE_RootNode
7880   }else{
7881     if( argv[0] == '..' ){
7882       ne = cwe.parentNode
7883     }else{
7884       ne = document.getElementById(argv[0])
7885     }
7886     if( ne != null ){
7887       GJE_CurElement = ne
7888       ret = "-- current node: " + ne.id
7889     }else{
7890       ret = "-- not found: " + argv[0]
7891     }
7892   }
7893   if( cmd == '.mkt' || cmd == '.mktable' ){
7894     makeTable(argv)
7895   }else{
7896     if( cmd == '.m' || cmd == '.mk' || cmd == 'mk' ){
7897       ne = document.createElement(argv[0])
7898       //ne.id = argv[0]
7899       ret = "-- created " + ne + " under " + cwe.tagName + "#" + cwe.id
7900       cwe.appendChild(ne)
7901       if( cmd == '.m' || cmd == '.mk' ){
7902         GJE_CurElement = ne
7903       }
7904     }else{
7905       if( cmd == '.n' || cmd == '.nm' || cmd == 'nm' ){
7906         cwe.id = argv[0]
7907       }else{
7908         if( cmd == '.r' || cmd == '.rm' || cmd == 'rm' ){
7909       }else{
7910         if( cmd == '.h' || cmd == '.sh' || cmd == 'sh' ){
7911           s = argv.join(',')
7912           cwe.innerHTML = s
7913         }else{
7914           if( cmd == 'a' || cmd == '.sa' || cmd == 'sa' ){
7915             cwe.setAttribute(argv[0],argv[1])
7916           }else{
7917             if( cmd == '.l' ){
7918               if( cmd == '.i' || cmd == '.ih' || cmd == 'ih' ){
7919                 ret = cwe.innerHTML
7920               }else{
7921                 if( cmd == '.p' || cmd == '.pw' || cmd == 'pw' ){
7922                   ret = cwe.nodeType + " " + cwe.tagName + " " + cwe.id
7923                   for( we = cwe.parentNode; we != null; ){
7924                     ret += "\n" + " " + we.nodeType + " " + we.tagName + " " + we.id
7925                     we = we.parentNode
7926                   }
7927                 }
7928               }
7929             {
7930               ret = "Command: mk | rm \n"
7931               ret += " pw -- print current node\n"
7932               ret += " mk type -- make node with name and type\n"
7933               ret += " nm name -- set the id #name of current node\n"
7934               ret += " rm name -- remove named node\n"
7935               ret += " cd name -- change current node\n"
7936             }
7937           }
7938         }
7939       }
7940     }
7941   }
7942 }
```

```

7937     //alert(ret)
7938     return ret
7939   }
7940   function GJC_Command(text){
7941     lines = text.value.split('\n')
7942     line = lines.length-1
7943     argv = line.split(' ')
7944     text.value += '\n'
7945     if( argv[0] == '%' ){ argv.shift() }
7946     args0 = argv.join(' ')
7947     cmd = argv[0]
7948     argv.shift()
7949     args = argv.join(' ')
7950
7951     if( cmd == 'nolog' ){
7952       StopConsoleLog = true
7953     }else{
7954       if( cmd == 'new' ){
7955         if( argv[0] == 'table' ){
7956           argv.shift()
7957           console.log('argv=' + argv)
7958           text.value += makeTable(argv)
7959         }else{
7960           if( argv[0] == 'console' ){
7961             text.value += GJ_NewConsole('GJ_Console')
7962           }else{
7963             text.value += '-- new { console | table }'
7964           }
7965         }else{
7966           if( cmd == 'strip' ){
7967             //text.value += GJF_StripClass()
7968           }else{
7969             if( cmd == 'css' ){
7970               sel = '#table_1'
7971               if(argv[0]==0)
7972                 rule1 = sel+'{color:#000 !important; background-color:#fff !important;}';
7973               else
7974                 rule1 = sel+'{color:#f00 !important; background-color:#eef !important;}';
7975               document.styleSheets[3].deleteRule(0);
7976               document.styleSheets[3].insertRule(rule1,0);
7977               text.value += 'CSS rule added: '+rule1
7978             }else{
7979               if( cmd == 'print' ){
7980                 e = null;
7981                 if( e == null ){
7982                   e = document.getElementById('GJFactory_0')
7983                 }else{
7984                   e = document.getElementById('GJFactory_1')
7985                 }
7986                 if( argv[0] != null ){
7987                   id = argv[0]
7988                   if( id == 'f' ){
7989                     //e = document.getElementById('GJE_RootNode');
7990                   }else{
7991                     e = document.getElementById(id)
7992                   }
7993                   if( e != null ){
7994                     text.value += e.outerHTML
7995                   }else{
7996                     text.value += "Not found: " + id
7997                   }
7998                 }else{
7999                   text.value += GJE_RootNode.outerHTML
8000                   //text.value += e.innerHTML
8001                 }
8002               }else{
8003                 if( cmd == 'destroy' ){
8004                   text.value += GJFactory_Destroy()
8005                 }else{
8006                   if( cmd == 'save' ){
8007                     e = document.getElementById('GJFactory')
8008                     Permanent.setItem('GJFactory-1',e.innerHTML)
8009                     text.value += "-- Saved GJFactory"
8010                   }else{
8011                     if( cmd == 'load' ){
8012                       gif = Permanent.getItem('GJFactory-1')
8013                       e = document.getElementById('GJFactory')
8014                       e.innerHTML = gif
8015                       // must restore EventListener
8016                       text.value += "-- EventListener was not restored"
8017                     }else{
8018                       if( cmd.charAt(0) == '.' ){
8019                         argv0 = args0.split(' ')
8020                         text.value += GJE_NodeEdit(argv0)
8021                       }else{
8022                         if( cmd == 'cont' ){
8023                           bannerIsStopping = false
8024                           GshMenuStop.innerHTML = "Stop"
8025                         }else{
8026                           if( cmd == 'date' ){
8027                             text.value += DateLong()
8028                           }else{
8029                             if( cmd == 'echo' ){
8030                               text.value += args
8031                             }else{
8032                               if( cmd == 'fork' ){
8033                                 html_fork()
8034                               }else{
8035                                 if( cmd == 'last' ){
8036                                   text.value += MyHistory
8037                                   //h = document.createElement("span")
8038                                   //h.innerHTML = MyHistory
8039                                   //text.value += h.innerHTML
8040                                   //tx = MyHistory.replace("\n","");
8041                                   //text.value += tx.replace("<"+br>","\n") + "xxxx<"+br>yyyy"
8042                                 }else{
8043                                   if( cmd == 'ne' ){
8044                                     text.value += GJE_NodeEdit(argv)
8045                                   }else{
8046                                     if( cmd == 'reload' ){
8047                                       location.reload()
8048                                     }else{
8049                                       if( cmd == 'mem' ){
8050                                         text.value += GJC_Memory('GJC_Storage',args,text)
8051                                       }else{
8052                                         if( cmd == 'stop' ){
8053                                           bannerIsStopping = true
8054                                           GshMenuStop.innerHTML = "Start"
8055                                         }else{
8056                                           if( cmd == 'who' ){
8057                                             text.value += "SessionId=" + GJC_SessionId + " " + document.URL
8058                                           }else{
8059                                             if( cmd == 'wall' ){
8060

```

```

8061     text.value += GJC_Memory('GJC_Wall','write',text)
8062   }else
8063   {
8064     text.value += "Commands: help | echo | date | last \n"
8065     + '          new | save | load | mem \n'
8066     + '          who | wall | fork | nife'
8067   }
8068 }
8069
8070 function GJC_Input(){
8071   if( this.value.endsWith("\n") ){ // remove NL added by textarea
8072     this.value = this.value.slice(0,this.value.length-1)
8073   }
8074 }
8075
8076 var GJC_Id = null
8077 function GJC_Resize(){
8078   GJC_Id.style.zIndex = 20000
8079   GJC_Id.style.width = window.innerWidth - 16
8080   GJC_Id.style.height = 300
8081   GJC_Id.style.backgroundColor = "rgba(0,64,16,1.0)" // blackboard color
8082   GJC_Id.style.color = "rgba(255,255,255,1.0)"
8083 }
8084 function GJC_FocusIn(){
8085   this.spellcheck = false
8086   SuppressGJShell = true
8087   this.onkeydown = GJC_KeyDown
8088   GJC_Resize()
8089 }
8090 function GJC_FocusOut(){
8091   SuppressGJShell = false
8092   this.removeEventListener('keydown',GJC_KeyDown);
8093 }
8094 window.addEventListener('resize',GJC_Resize);
8095
8096 function GJC_OnStorage(e){
8097   //alert('Got Message')
8098   //GJC.value += "\n((ReceivedMessage))\n"
8099 }
8100 window.addEventListener('storage',GJC_OnStorage);
8101 //window.addEventListener('storage',()=>{alert('GotMessage')})
8102
8103 function GJC_Setup(gjcid){
8104   gjcid.style.width = gsh.getBoundingClientRect().width
8105   gjcid.value = "GJShell Console // " + GshVersion.innerHTML + "\n"
8106   //gjid.value += "Date: " + DateLong() + "\n"
8107   gjcid.value += PS1
8108   gjcid.onfocus = GJC_FocusIn
8109   gjcid.addEventListener('input',GJC_Input);
8110   gjcid.addEventListener('focusout',GJC_FocusOut);
8111   GJC_Id = gjcid
8112 }
8113 function GJC_Clear(id){
8114 }
8115 if( document.getElementById("GJC_0") != null ){
8116   GJC_Setup(GJC_0)
8117 }else{
8118   document.write('<'+textarea id="GJC_1" class="GJConsole"><+'/textarea>')
8119   GJC_Setup(GJC_1)
8120   factory = document.createElement('span');
8121   gsh.appendChild(factory)
8122   GJE_RootNode = factory;
8123   GJE_CurElement = GJE_RootNode;
8124 }
8125
8126 // TODO: focus handling
8127 </script>
8128 <style>
8129 .GJ_StyleEditor {
8130   font-size:9pt !important;
8131   font-family:Courier New, monospace !important;
8132 }
8133 </style>
8134
8135 <!-- ----- GJConsole END } ----- -->
8136 </span>
8137
8138 *///<br></span></html>
8139

```