

```

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" gsh--0.4.7--2020-09-20--SatoxITS</span>
7 <header id="Gshell" Gshell-0.4.7 by SatoxITS<title>
8 <div align="right"><note><a href="http://archive.gshell.org">GShell</a> version 0.4.7 // 2020-09-20 // 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 id="gsh-menu-pure" onclick="html_pure(this);">Pure</span> -->
26 </span>
27 /*
28 */
29 <details id="GshStatement" class="gsh-document"><summary>Statement</summary>
30 <h3>Fun to create a shell</h3>
31 <p>For a programmer, it must be far easy and fun to create his own simple shell
32 rightly fitting to his favor and necessities, than learning existing shells with
33 complex full features that he never use.
34 I, as one of programmers, am writing this tiny shell for my own real needs,
35 totally from scratch, with fun.
36 </p><p>
37 For a programmer, it is fun to learn new computer languages. For long years before
38 writing this software, I had been specialized to C and early HTML2 :).
39 Now writing this software, I'm learning Go language, HTML5, JavaScript and CSS
40 on demand as a novice of these, with fun.
41 </p><p>
42 This single file "gsh.go", that is executable by Go, contains all of the code written
43 in Go. Also it can be displayed as "gsh.go.html" by browsers. It is a standalone
44 HTML file that works as the viewer of the code of itself, and as the "home page" of
45 this software.
46 </p><p>
47 Because this HTML file is a Go program, you may run it as a real shell program
48 on your computer.
49 But you must be aware that this program is written under situation like above.
50 Needless to say, there is no warranty for this program in any means.
51 </p>
52 <address>Aug 2020, SatoxITS (sato@its-more.jp)</address>
53 </details>
54 /*
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="#finder">finder</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 redireciton
88   <a href="#history">history</a> // command history
89   <a href="#usage">usage</a> // resouce usage
90   <a href="#encode">encode</a> // encode / decode
91   <a href="#IME">IME</a> // command line IME
92   <a href="#getline">getline</a> // line editor
93   <a href="#scanf">scanf</a> // string decomposer
94   <a href="#interpreter">interpreter</a> // command interpreter
95   <a href="#main">main</a>
96 </span>
97 JavaScript part
98   <a href="#script-src-view" class="gsh-link" onclick="jumpto_JavaScriptView();">Source</a>
99   <a href="#gsh-data-frame" class="gsh-link" onclick="jumpto_DataView();">Builtin data</a>
100 CSS part
101   <a href="#style-src-view" class="gsh-link" onclick="jumpto_StyleView();">Source</a>
102 References
103   <a href="#" class="gsh-link" onclick="jumpto_WholeView();">Internal</a>
104   <a href="#gsh-reference" class="gsh-link" onclick="jumpto_ReferenceView();">External</a>
105 Whole parts
106   <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Source</a>
107   <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Download</a>
108   <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Dump</a>
109 </div>
110 </details>
111 /*
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 package main // gsh main
119
120 // <a name="import">Imported packages</a> // <a href="https://golang.org/pkg/">Packages</a>
121 import (
122   "fmt" // <a href="https://golang.org/pkg/fmt/">fmt</a>
123   "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 "golang.org/x/net/websocket"
144 }
145
146 // // 2020-0906 added,
147 // // <a href="https://golang.org/cmd/cgo/">CGo</a>
148 // #include "poll.h" // <poll.h> to be closed as HTML tag :-p
149 // typedef struct { struct pollfd fdv[8]; } pollFdv;
150 // int pollx(pollFd *fdv, int nfds, int timeout);
151 // return poll(fdv->fdv,nfds,timeout);
152 //}
153 import "C"
154
155 // // 2020-0906 added,
156 func CFpollIn1(fp*os.File, timeoutUs int)(ready uintptr{
157     var fdv = C.pollFd{ }
158     var nfds = 1
159     var timeout = timeoutUs/1000
160
161     fdv.fdv[0].fd = C.int(fp.Fd())
162     fdv.fdv[0].events = C.POLLIN
163     if( 0 < EventRecvFd ){
164         fdv.fdv[1].fd = C.int(EventRecvFd)
165         fdv.fdv[1].events = C.POLLIN
166         nfds += 1
167     }
168     r := C.pollx(&fdv,C.int(nfds),C.int(timeout))
169     if( r <= 0 ){
170         return 0
171     }
172     if (int(fdv.fdv[1].revents) & int(C.POLLIN)) != 0 {
173         //fprintf(stderr,"--De-- got Event\n");
174         return uintptr(EventFdOffset + fdv.fdv[1].fd)
175     }
176     if (int(fdv.fdv[0].revents) & int(C.POLLIN)) != 0 {
177         return uintptr(NormalFdOffset + fdv.fdv[0].fd)
178     }
179     return 0
180 }
181
182 const (
183     NAME = "gsh"
184     VERSION = "0.4.7"
185     DATE = "2020-09-20"
186     AUTHOR = "SatoxITS(^_^)//"
187 )
188 var (
189     GSH_HOME = ".gsh" // under home directory
190     GSH_PORT = 9999
191     MaxStreamSize = int64(128*1024*1024*1024) // 128GiB is too large?
192     PROMPT = "> "
193     LINESIZE = (8*1024)
194     PATHSEP = ":" // should be ";" in Windows
195     DIRSEP = "/" // canbe \ in Windows
196 )
197
198 // --xx logging control
199 // --A-- all
200 // --I-- info.
201 // --D-- debug
202 // --T-- time and resource usage
203 // --W-- warning
204 // --E-- error
205 // --F-- fatal error
206 // --Xn-- network
207
208 // <a name="struct">Structures</a>
209 type GCommandHistory struct {
210     StartAt    time.Time // command line execution started at
211     EndAt     time.Time // command line execution ended at
212     ResCode    int       // exit code of (external command)
213     CmdError   error    // error string
214     OutData   *os.File  // output of the command
215     Foundfile []string  // output - result of ufind
216     Rusageev [2]syscall.Rusage // Resource consumption, CPU time or so
217     CmdId     int       // maybe with identified with arguments or impact
218     // redirecton commands should not be the CmdId
219     Workdir   string   // working directory at start
220     WorkdirX  int      // index in ChdirHistory
221     CmdLine   string   // command line
222 }
223 type GChdirHistory struct {
224     Dir      string
225     MovedAt  time.Time
226     CmdIndex int
227 }
228 type CmdMode struct {
229     BackGround bool
230 }
231 type Event struct {
232     when    time.Time
233     event   int
234     evarg   int64
235     CmdIndex int
236 }
237 var CmdIndex int
238 var Events []Event
239 type PluginInfo struct {
240     Spec      *plugin.Plugin
241     Addr      plugin.Symbol
242     Name      string // maybe relative
243     Path      string // this is in Plugin but hidden
244 }
245 type GServer struct {
246     host      string
247     port      string
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 = 0x020000 // 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 [1]GchdirHistory // the 1st entry is wd at the start
295     gshPA syscall.ProcAttr
296     CommandHistory [1]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
317 func nsleep(ns time.Duration){
318     time.Sleep(ns)
319 }
320 func usleep(ns time.Duration){
321     nsleep(ns*1000)
322 }
323 func msleep(ns time.Duration){
324     nsleep(ns*1000000)
325 }
326 func sleep(ns time.Duration){
327     nsleep(ns*1000000000)
328 }
329
330 func strBegins(str, pat string)(bool){
331     if len(pat) < len(str){
332         yes := str[0:len(pat)] == pat
333         //fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat,yes)
334         return yes
335     }
336     //fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat,false)
337     return false
338 }
339 func isin(what string, list []string) bool {
340     for _, v := range list {
341         if v == what {
342             return true
343         }
344     }
345     return false
346 }
347 func isinX(what string, list[]string)(int){
348     for i,v := range list {
349         if v == what {
350             return i
351         }
352     }
353     return -1
354 }
355
356 func env(opts []string) {
357     env := os.Environ()
358     if isin("-a", opts){
359         sort.Slice(env, func(i,j int) bool {
360             return env[i] < env[j]
361         })
362     }
363     for _, v := range env {
364         fmt.Printf("%v\n",v)
365     }
366 }
367
368 // - rewriting should be context dependent
369 // - should postpone until the real point of evaluation
370 // - should rewrite only known notation of symbol
371 func scanInt(str string)(val int,leng int){
372     leng = -1

```

```

373     for i,ch := range str {
374         if '0' <= ch && ch <= '9' {
375             leng = i+1
376         }else{
377             break
378         }
379     }
380     if 0 < leng {
381         ival,_ := strconv.Atoi(str[0:leng])
382         return ival,leng
383     }else{
384         return 0,0
385     }
386 }
387 func substHistory(gshCtx *GshContext,str string,i int,rstr string)(leng int,rst string){
388     if len(str[i+1:]) == 0 {
389         return 0,rstr
390     }
391     hi := 0
392     histlen := len(gshCtx.CommandHistory)
393     if str[i+1] == '!' {
394         hi = histlen - 1
395         leng = 1
396     }else{
397         hi,leng = scanInt(str[i+1:])
398         if leng == 0 {
399             return 0,rstr
400         }
401         if hi < 0 {
402             hi = histlen + hi
403         }
404     }
405     if 0 <= hi && hi < histlen {
406         var ext byte
407         if i < len(str[i+leng:]) {
408             ext = str[i+leng:][1]
409         }
410         //fmt.Printf("-D-- %v(%c)\n",str[i+leng:],str[i+leng])
411         if ext == 'f' {
412             leng += 1
413             xlist := []string{}
414             list := gshCtx.CommandHistory[hi].FoundFile
415             for _,v := range list {
416                 //list[i] = escapeWhiteSP(v)
417                 xlist = append(xlist,escapeWhiteSP(v))
418             }
419             //rstr += strings.Join(list," ")
420             rstr += strings.Join(xlist," ")
421         }else
422         if ext == 'e' || ext == 'd' {
423             // !N@ .. workdir at the start of the command
424             leng += 1
425             rstr += gshCtx.CommandHistory[hi].WorkDir
426         }else{
427             rstr += gshCtx.CommandHistory[hi].CmdLine
428         }
429     }else{
430         leng = 0
431     }
432     return leng,rstr
433 }
434 func escapeWhiteSP(str string)(string){
435     if len(str) == 0 {
436         return "\z" // empty, to be ignored
437     }
438     rstr := ""
439     for _,ch := range str {
440         switch ch {
441             case '\\': rstr += "\\\\"
442             case '\n': rstr += "\\s"
443             case '\t': rstr += "\\t"
444             case '\r': rstr += "\\r"
445             case '\n': rstr += "\\n"
446             default: rstr += string(ch)
447         }
448     }
449     return rstr
450 }
451 func unescapeWhiteSP(str string)(string){ // strip original escapes
452     rstr := ""
453     for i := 0; i < len(str); i++ {
454         ch := str[i]
455         if ch == '\\' {
456             if i+1 < len(str) {
457                 switch str[i+1] {
458                     case 'z':
459                         continue;
460                 }
461             }
462             rstr += string(ch)
463         }
464     }
465     return rstr
466 }
467 func unescapeSPV(strv []string)([]string){ // strip original escapes
468     ustrv := []string{}
469     for _,v := range strv {
470         ustrv = append(ustrv,unescapeWhiteSP(v))
471     }
472     return ustrv
473 }
474
475 // <a name="comexpansion">str-expansion</a>
476 // - this should be a macro processor
477 func strsubst(gshCtx *GshContext,str string,histonly bool) string {
478     rbuff := []byte{}
479     if false {
480         //@U Unicode should be cared as a character
481         return str
482     }
483     //rstr := ""
484     inEsc := 0 // escape character mode
485     for i := 0; i < len(str); i++ {
486         //fmt.Printf("--Subst %v:%v\n",i,str[i:])
487         ch := str[i]
488         if inEsc == 0 {
489             if ch == '!' {
490                 //len,xrstr := substHistory(gshCtx,str,i,rstr)
491                 leng,rs := substHistory(gshCtx,str,i,"")
492                 if 0 < leng {
493                     _,rs := substHistory(gshCtx,str,i,"")
494                     rbuff = append(rbuff,[]byte(rs)...)
495                     i += leng
496                     //rstr = xrstr

```

```

497         continue
498     }
499 }
500 switch ch {
501     case '\\': inEsc = '\\'; continue
502     //case '%': inEsc = '%'; continue
503     case '$':
504 }
505 }
506 switch inEsc {
507 case '\\':
508     switch ch {
509         case '\\': ch = '\\'
510         case 's': ch = ' '
511         case 't': ch = '\t'
512         case 'r': ch = '\r'
513         case 'n': ch = '\n'
514         case 'z': inEsc = 0; continue // empty, to be ignored
515     }
516     inEsc = 0
517 case '%':
518     switch {
519         case ch == '%': ch = '%'
520         case ch == 't':
521             //rstr = rstr + time.Now().Format(time.Stamp)
522             rs := time.Now().Format(time.Stamp)
523             rbuff = append(rbuff,[byte(rs)...)
524             inEsc = 0
525             continue;
526         default:
527             // postpone the interpretation
528             //rstr = rstr + "%" + string(ch)
529             rbuff = append(rbuff,ch)
530             inEsc = 0
531             continue;
532     }
533     inEsc = 0
534 }
535 //rstr = rstr + string(ch)
536 rbuff = append(rbuff,ch)
537 }
538 //fmt.Printf("--D--subst(%s)(%s)\n",str,string(rbuff))
539 return string(rbuff)
540 //return rstr
541 }
542 func showFileInfo(path string, opts []string) {
543     if isin("-l",opts) || isin("-ls",opts) {
544         fi, err := os.Stat(path)
545         if err != nil {
546             fmt.Printf("----- ((%v))",err)
547         }else{
548             mod := fi.ModTime()
549             date := mod.Format(time.Stamp)
550             fmt.Printf("%v %v %s ",fi.Mode(),fi.Size(),date)
551         }
552     }
553     fmt.Printf("%s",path)
554     if isin("-sp",opts) {
555         fmt.Println(" ")
556     }else
557     if ! isin("-n",opts) {
558         fmt.Println("\n")
559     }
560 }
561 func userHomeDir()(string,bool){
562     /*
563     homedir,_ = os.UserHomeDir() // not implemented in older Golang
564     */
565     homedir,found := os.LookupEnv("HOME")
566     //fmt.Printf("--I-- HOME=%v(%v)\n",homedir,found)
567     if !found {
568         return "/tmp",found
569     }
570     return homedir,found
571 }
572
573 func toFullPath(path string) (fullpath string) {
574     if path[0] == '/' {
575         return path
576     }
577     pathv := strings.Split(path,DIRSEP)
578     switch {
579     case pathv[0] == ".": 
580         pathv[0],_ = os.Getwd()
581     case pathv[0] == "..": // all ones should be interpreted
582         cwd, _ := os.Getwd()
583         ppathv := strings.Split(cwd,DIRSEP)
584         pathv[0] = strings.Join(ppathv,DIRSEP)
585     case pathv[0] == "~":
586         pathv[0],_ = userHomeDir()
587     default:
588         cwd, _ := os.Getwd()
589         pathv[0] = cwd + DIRSEP + pathv[0]
590     }
591     return strings.Join(pathv,DIRSEP)
592 }
593
594 func IsRegFile(path string)(bool){
595     fi, err := os.Stat(path)
596     if err == nil {
597         fm := fi.Mode()
598         return fm.IsRegular();
599     }
600     return false
601 }
602
603 // <a name="encode">Encode / Decode</a>
604 // <a href="https://golang.org/pkg/encoding/base64/#example_NewEncoder">Encoder</a>
605 func (gshCtx *GshContext)Enc(argv[]string){
606     file := os.Stdin
607     buff := make([]byte,LINESIZE)
608     li := 0
609     encoder := base64.NewEncoder(base64.StdEncoding,os.Stdout)
610     for li = 0; ; li++ {
611         count, err := file.Read(buff)
612         if count <= 0 {
613             break
614         }
615         if err != nil {
616             break
617         }
618         encoder.Write(buff[0:count])
619     }
620     encoder.Close()

```

```

621 }
622 func (gshCtx *GshContext)Dec(argv[]string){
623     decoder := base64.NewDecoder(base64.StdEncoding,os.Stdin)
624     li := 0
625     buff := make([]byte,INESIZE)
626     for li = 0; ; li++ {
627         count, err := decoder.Read(buff)
628         if count <= 0 {
629             break
630         }
631         if err != nil {
632             break
633         }
634         os.Stdout.Write(buff[0:count])
635     }
636 }
637 // lnsp [N] [-crlf]{-C \\}
638 func (gshCtx *GshContext)SplitLine(argv[]string){
639     strRep := isin("-str",argv) // "...+
640     reader := bufio.NewReaderSize(os.Stdin,64*1024)
641     ni := 0
642     toi := 0
643     for ni = 0; ; ni++ {
644         line, err := reader.ReadString('\n')
645         if len(line) <= 0 {
646             if err != nil {
647                 fmt.Fprintf(os.Stderr,"--I-- lnsp %d to %d (%v)\n",ni,toi,err)
648                 break
649             }
650         }
651         off := 0
652         ilen := len(line)
653         remlen := len(line)
654         if strRep { os.Stdout.Write([]byte("\n")) }
655         for oi := 0; 0 < remlen; oi++ {
656             olen := remlen
657             addnl := false
658             if 72 < olen {
659                 olen = 72
660                 addnl = true
661             }
662             fmt.Fprintf(os.Stderr,"--D-- write %d [%d.%d] %d %d/%d/%d\n",
663                         toi,ni,oi,off,olen,remlen,ilen)
664             toi += 1
665             os.Stdout.Write([]byte(line[0:olen]))
666             if addnl {
667                 if strRep {
668                     os.Stdout.Write([]byte("\r\n"))
669                 }else{
670                     //os.Stdout.Write([]byte("\r\n"))
671                     os.Stdout.Write([]byte("\\"))
672                     os.Stdout.Write([]byte("\n"))
673                 }
674             line = line[olen:]
675             off += olen
676             remlen -= olen
677         }
678         if strRep { os.Stdout.Write([]byte("\r\n")) }
679     }
680     fmt.Fprintf(os.Stderr,"--I-- lnsp %d to %d\n",ni,toi)
681 }
682 }
683
684 // CRC32 <a href="http://golang.jp/pkg/hash-crc32">crc32</a>
685 // 1 0000 0100 1100 0001 1101 1011 0111
686 var CRC32UNIX uint32 = uint32(0x04C11DB7) // Unix cksum
687 var CRC32IEEE uint32 = uint32(0xEDB88320)
688 func byteCRC32add(crc uint32,str[]byte,len uint64)(uint32){
689     var oi uint64
690     for oi = 0; oi < len; oi++ {
691         var oct = str[oi]
692         for bi := 0; bi < 8; bi++ {
693             //fprintf(stderr,"--CRC32 %d %X (%d.%d)\n",crc,oct,oi,bi)
694             ovf1 := (crc & 0x80000000) != 0
695             ovf2 := (oct & 0x80) != 0
696             ovf := (ovf1 && !ovf2) || (!ovf1 && ovf2)
697             oct <<= 1
698             crc <<= 1
699             if ovf { crc ^= CRC32UNIX }
700         }
701     }
702     //fprintf(stderr,"--CRC32 return %d %d\n",crc,len)
703     return crc;
704 }
705 func byteCRC32end(crc uint32, len uint64)(uint32){
706     var slen = make([]byte,4)
707     var li = 0
708     for li = 0; li < 4; {
709         slen[li] = byte(len)
710         li += 1
711         len >= 8
712         if( len == 0 ){
713             break
714         }
715     }
716     crc = byteCRC32add(crc,slen,uint64(li))
717     crc ^= 0xFFFFFFFF
718     return crc
719 }
720 func strCRC32(str string,len uint64)(crc uint32){
721     crc = byteCRC32add(0,[],byte(str),len)
722     crc = byteCRC32end(crc,len)
723     //fprintf(stderr,"--CRC32 %d %d\n",crc,len)
724     return crc
725 }
726 func CRC32Finish(crc uint32, table *crc32.Table, len uint64)(uint32){
727     var slen = make([]byte,4)
728     var li = 0
729     for li = 0; li < 4; {
730         slen[li] = byte(len & 0xFF)
731         li += 1
732         len >>= 8
733         if( len == 0 ){
734             break
735         }
736     }
737     crc = crc32.Update(crc,table,slen)
738     crc ^= 0xFFFFFFFF
739     return crc
740 }
741
742 func (gsh*GshContext)xCksum(path string,argv[]string, sum*CheckSum)(int64){
743     if isin("-type/f",argv) && !IsRegFile(path){
744         return 0

```

```

745     }
746     if isin("-type/d", argv) && IsRegFile(path){
747         return 0
748     }
749     file, err := os.OpenFile(path,os.O_RDONLY,0)
750     if err != nil {
751         fmt.Printf("--E-- cksum %v (%v)\n",path,err)
752         return -1
753     }
754     defer file.Close()
755     if gsh.CmdTrace { fmt.Printf("--I-- cksum %v %v\n",path,argv) }
756
757     bi := 0
758     var buff = make([]byte,32*1024)
759     var total int64 = 0
760     var initTime = time.Time{}
761     if sum.Start == initTime {
762         sum.Start = time.Now()
763     }
764     for bi = 0; ; bi++ {
765         count,err := file.Read(buff)
766         if count <= 0 || err != nil {
767             break
768         }
769         if (sum.SumType & SUM_SUM64) != 0 {
770             s := sum.Sum64
771             for _,c := range buff[0:count] {
772                 s += uint64(c)
773             }
774             sum.Sum64 = s
775         }
776         if (sum.SumType & SUM_UNIXFILE) != 0 {
777             sum.Crc32Val = byteCrc32add(sum.Crc32Val,buff,uint64(count))
778         }
779         if (sum.SumType & SUM_CRCIEEE) != 0 {
780             sum.Crc32Val = crc32.Update(sum.Crc32Val,&sum.Crc32Table,buff[0:count])
781         }
782         // <a href="https://en.wikipedia.org/wiki/BSD_checksum">BSD checksum</a>
783         if (sum.SumType & SUM_SUM16_BSD) != 0 {
784             s := sum.Sum16
785             for _,c := range buff[0:count] {
786                 s = (s >> 1) + ((s & 1) << 15)
787                 s += int(c)
788                 s &= 0xFFFF
789                 //fmt.Printf("BSDsum: %d%d %d\n",sum.Size+int64(i),i,s)
790             }
791             sum.Sum16 = s
792         }
793         if (sum.SumType & SUM_SUM16_SYSV) != 0 {
794             for bj := 0; bj < count; bj++ {
795                 sum.Sum16 += int(buff[bj])
796             }
797         }
798         total += int64(count)
799     }
800     sum.Done = time.Now()
801     sum.Files += 1
802     sum.Size += total
803     if !isin("-s",argv) {
804         fmt.Printf("%v ",total)
805     }
806     return 0
807 }
808
809 // <a name="grep">grep</a>
810 // "lines", "lin" or "lnp" for "(text) line processor" or "scanner"
811 // a*,!ab,c, ... sequential combination of patterns
812 // what "LINE" is should be definable
813 // generic line-by-line processing
814 // grep [-v]
815 // cat -n -v
816 // uniq [-c]
817 // tail -f
818 // sed s/x/y/ or awk
819 // grep with line count like wc
820 // rewrite contents if specified
821 func (gsh*GshContext)xGrep(path string,rexpv[]string)(int){
822     file, err := os.OpenFile(path,os.O_RDONLY,0)
823     if err != nil {
824         fmt.Printf("--E-- grep %v (%v)\n",path,err)
825         return -1
826     }
827     defer file.Close()
828     if gsh.CmdTrace { fmt.Printf("--I-- grep %v %v\n",path,rexpv) }
829     //reader := bufio.NewReaderSize(file,LINESIZE)
830     reader := bufio.NewReaderSize(file,80)
831     li := 0
832     found := 0
833     for li = 0; ; li++ {
834         line, err := reader.ReadString('\n')
835         if len(line) <= 0 {
836             break
837         }
838         if 150 < len(line) {
839             // maybe binary
840             break;
841         }
842         if err != nil {
843             break
844         }
845         if 0 <= strings.Index(string(line),rexpv[0]) {
846             found += 1
847             fmt.Printf("%s:%d: %s",path,li,line)
848         }
849     }
850     //fmt.Printf("total %d lines %s\n",li,path)
851     //if( 0 < found){ fmt.Printf("(found %d lines %s)\n",found,path); }
852     return found
853 }
854
855 // <a name="finder">Finder</a>
856 // finding files with it name and contents
857 // file names are OKed
858 // show the content with %x fmt list
859 // ls -R
860 // tar command by adding output
861 type fileSum struct {
862     Err int64 // access error or so
863     Size int64 // content size
864     DupSize int64 // content size from hard links
865     Blocks int64 // number of blocks (of 512 bytes)
866     DupBlocks int64 // Blocks pointed from hard links
867     HLinks int64 // hard links
868     Words int64

```

```

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

```

```

993     if isin("-du",argv) {
994         fmt.Printf("%d\t",fstat.Blocks/2)
995     }
996     showFileInfo(path,argv)
997 }
998 if true { // && isin("-du",argv)
999     total = cumFileInfo(total,path,staterr,fstat,argv,false)
1000 }
1001 /*
1002 if isin("-wc",argv) {
1003 }
1004 */
1005 if gsh.lastCheckSum.SumType != 0 {
1006     gsh.xCksum(path,argv,&gsh.lastCheckSum);
1007 }
1008 x := isin("-grep",argv); // -grep will be convenient like -ls
1009 if 0 <= x && x+1 <= len(argv) { // -grep will be convenient like -ls
1010     if IsRegfile(path){
1011         found := gsh.xGrep(path,argv[x+1:])
1012         if 0 < found {
1013             foundv := gsh.CmdCurrent.FoundFile
1014             if len(foundv) < 10 {
1015                 gsh.CmdCurrent.FoundFile =
1016                     append(gsh.CmdCurrent.FoundFile,path)
1017             }
1018         }
1019     }
1020 }
1021 if !isin("-r0",argv) { // -d 0 in du, -depth n in find
1022     //total.Depth += 1
1023     if (fstat.Mode & S_IFMT) == S_IFLNK {
1024         continue
1025     }
1026     if dstat.Rdev != fstat.Rdev {
1027         fmt.Printf("--I-- don't follow differnet device %v %v\n",
1028               dir,dstat.Rdev,path,fstat.Rdev)
1029     }
1030     if (fstat.Mode & S_IFMT) == S_IFDIR {
1031         total = gsh.xxFind(depth+1,total,path,npatv,argv)
1032     }
1033 }
1034 }
1035 }
1036 return total
1037 }
1038 func (gsh*GshContext)xxFind(depth int,total *fileSum,dir string,npatv[]string,argv[]string)(*fileSum{
1039     nols := isin("-grep",argv)
1040     dirfile,err := os.OpenFile(dir,os.O_RDONLY,0)
1041     if err == nil {
1042         //fmt.Printf("--I-- %v(%d)\n",dir,dirfile,dirfile.Fd())
1043         defer dirfile.Close()
1044     }else{
1045     }
1046     prev := *total
1047     var dstat syscall.Stat_t
1048     staterr := syscall.Lstat(dir,&dstat) // should be flistat
1049     if staterr != nil {
1050         if !isin("-w",argv){ fmt.Printf("ufind: %v\n",staterr) }
1051         return total
1052     }
1053     //if 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         total = 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 }
1068 // it is not a directory, just scan it and finish
1069
1070 for ei := 0; ei++ {
1071     entv,rderr := dirfile.Readdirnames(8*1024)
1072     if len(entv) == 0 || rderr != nil {
1073         //if rderr != nil { fmt.Printf("[%d] len=%d (%v)\n",ei,len(entv),rderr) }
1074         break
1075     }
1076     if 0 < ei {
1077         fmt.Printf("--I-- xxFind[%d] %d large-dir: %s\n",ei,len(entv),dir)
1078     }
1079     total = gsh.xxFindEntv(depth,total,dir,dstat,ei,entv,npatv,argv)
1080 }
1081 if isin("-du",argv) {
1082     // if in "du" mode
1083     fmt.Printf("%d\t%s\n", (total.Blocks-prev.Blocks)/2,dir)
1084 }
1085 }
1086 return total
1087 }
1088
1089 // {ufind|fu|ls} [Files] [-- Expressions]
1090 // Files is " " by default
1091 // Names is "*" by default
1092 // Expressions is "-print" by default for "ufind", or -du for "fu" command
1093 func (gsh*GshContext)xFind(argv[]string){
1094     if 0 < len(argv) && strBegins(argv[0],"?"){
1095         showFound(gsh,argv)
1096         return
1097     }
1098     if isin("-cksum",argv) || isin("-sum",argv) {
1099         gsh.lastCheckSum = CheckSum{}
1100         if isin("-sum",argv) && isin("-add",argv) {
1101             gsh.lastCheckSum.SumType |= SUM_SUM64
1102         }else
1103             if isin("-sum",argv) && isin("-size",argv) {
1104                 gsh.lastCheckSum.SumType |= SUM_SIZE
1105             }else
1106                 if isin("-sum",argv) && isin("-bsd",argv) {
1107                     gsh.lastCheckSum.SumType |= SUM_SUM16_BSD
1108                 }else
1109                     if isin("-sum",argv) && isin("-sysv",argv) {
1110                         gsh.lastCheckSum.SumType |= SUM_SUM16_SYSV
1111                     }else
1112                         if isin("-sum",argv) {
1113                             gsh.lastCheckSum.SumType |= SUM_SUM64
1114                         }
1115                         if isin("-unix",argv) {
1116                             gsh.lastCheckSum.SumType |= SUM_UNIXFILE

```

```

1117     gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32UNIX)
1118 }
1119 if !isin("-ieee",argv){
1120     gsh.lastCheckSum.SumType |= SUM_CRCIEEE
1121     gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32IEEE)
1122 }
1123 gsh.lastCheckSum.RusgAtStart = Getrusagev()
1124 }
1125 var total = fileSum{}
1126 npats := []string{}
1127 for _v := range argv {
1128     if 0 < len(v) && v[0] != '-' {
1129         npats = append(npats,v)
1130     }
1131     if v == "//" { break }
1132     if v == "--" { break }
1133     if v == "-grep" { break }
1134     if v == "-ls" { break }
1135 }
1136 if len(npats) == 0 {
1137     npats = []string{"*"}
1138 }
1139 cwd := "."
1140 // if to be fullpath :: cwd, _ := os.Getwd()
1141 if len(npats) == 0 { npats = []string{"*"} }
1142 fusage := gsh.xxFind(0,&total,cwd,npats,argv)
1143 if gsh.lastCheckSum.SumType != 0 {
1144     var sumi uint64 = 0
1145     sum := &gsh.lastCheckSum
1146     if (sum.SumType & SUM_SUM) != 0 {
1147         sumi = uint64(sum.Size)
1148     }
1149     if (sum.SumType & SUM_SUM64) != 0 {
1150         sumi = sum.Sum64
1151     }
1152     if (sum.SumType & SUM_SUM16_SYSV) != 0 {
1153         s := uint32(sum.Sum16)
1154         r := (s & 0xFFFF) + ((s & 0xFFFFFFFF) >> 16)
1155         s = (r & 0xFFFF) + (r >> 16)
1156         sum.Crc32Val = uint32(s)
1157         sumi = uint64(s)
1158     }
1159     if (sum.SumType & SUM_SUM16_BSD) != 0 {
1160         sum.Crc32Val = byteCRC32end(sum.Crc32Val,uint64(sum.Size))
1161         sumi = uint64(sum.Sum16)
1162     }
1163     if (sum.SumType & SUM_UNIXFILE) != 0 {
1164         sum.Crc32Val = byteCRC32end(sum.Crc32Val,uint64(sum.Size))
1165         sumi = uint64(byteCRC32end(sum.Crc32Val,uint64(sum.Size)))
1166     }
1167     if 1 < sum.Files {
1168         fmt.Printf("%v %v // %v / %v files, %v/file\r\n",
1169             sumi,sum.Size,
1170             abssize(sum.Size),sum.Files,
1171             abssize(sum.Size/sum.Files))
1172     }else{
1173         fmt.Printf("%v %v %v\r\n",
1174             sumi,sum.Size,npats[0])
1175     }
1176 }
1177 if !isin("-grep",argv) {
1178     showUsage("total",fusage)
1179 }
1180 if !isin("-s",argv){
1181     hits := len(gsh.CmdCurrent.FoundFile)
1182     if 0 < hits {
1183         fmt.Printf("--I-- %d files hits // can be referred with !%df\n",
1184             hits,len(gsh.CommandHistory))
1185     }
1186 }
1187 if gsh.lastCheckSum.SumType != 0 {
1188     if !isin("-ru",argv) {
1189         sum := &gsh.lastCheckSum
1190         sum.Done = time.Now()
1191         gsh.lastCheckSum.RusgAtEnd = Getrusagev()
1192         elps := sum.Done.Sub(sum.Start)
1193         fmt.Printf("--cksum-size: %v (%v) / %v files, %v/file\r\n",
1194             sum.Size,abssize(sum.Size),sum.Files,abssize(sum.Size/sum.Files))
1195         nanos := int64(elps)
1196         fmt.Printf("--cksum-time: %v/total, %v/file, %if files/s, %v\r\n",
1197             abbtme(nanos),
1198             abbtme(nanos/sum.Files),
1199             (float64(sum.Files)*1000000000.0)/float64(nanos),
1200             abbspeed(sum.Size,nanos))
1201         diff := RusageSubv(sum.RusgAtEnd,sum.RusgAtStart)
1202         fmt.Printf("--cksum-rusg: %v\r\n",sRusgef("",argv,diff))
1203     }
1204 }
1205 return
1206 }
1207
1208 func showFiles(files[]string){
1209     sp := ""
1210     for i,file := range files {
1211         if 0 < i { sp = " " } else { sp = "" }
1212         fmt.Printf(sp+"%s",escapeWhiteSP(file))
1213     }
1214 }
1215 func showFound(gshCtxx *GshContext, argv[]string){
1216     for i,v := range gshCtxx.CommandHistory {
1217         if 0 < len(v.FoundFile) {
1218             fmt.Printf("!%d (%d) ",i,len(v.FoundFile))
1219             if !isin("-ls",argv){
1220                 fmt.Printf("\n")
1221                 for _file := range v.FoundFile {
1222                     fmt.Printf("%") //sub number?
1223                     showFileInfo(file,argv)
1224                 }
1225             }else{
1226                 showFiles(v.FoundFile)
1227                 fmt.Printf("\n")
1228             }
1229         }
1230     }
1231 }
1232
1233 func showMatchfile(filev []os.FileInfo, npat,dir string, argv[]string)(string,bool){
1234     fname := ""
1235     found := false
1236     for _,v := range filev {
1237         match, _ := filepath.Match(npata,(v.Name()))
1238         if match {
1239             fname = v.Name()
1240             found = true

```

```

1241         //fmt.Printf("[%d] %s\n",i,v.Name())
1242         showIfExecutable(fname,dir,argv)
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(ffullpath,argv)
1267                 }
1268             }
1269         }
1270     }
1271 }
1272 return ffullpath, ffound
1273 }
1274 func which(list string, argv []string) (fullpathv []string, itis bool){
1275     if len(argv) <= 1 {
1276         fmt.Println("Usage: which command [-s] [-a] [-ls]\n")
1277         return []string{}, false
1278     }
1279     path := argv[1]
1280     if strBegins(path,"/") {
1281         // should check if executable?
1282         ,exOK := showIfExecutable(path,"/",argv)
1283         fmt.Printf("--D-- %v exOK=%v\n",path,exOK)
1284         return []string{path},exOK
1285     }
1286     pathenv, efound := os.LookupEnv(list)
1287     if ! efound {
1288         fmt.Printf("--E-- which: no \"%s\" environment\n",list)
1289         return []string{}, false
1290     }
1291     showall := isin("-a",argv) || 0 <= strings.Index(path,"*")
1292     dirv := strings.Split(pathenv,PATHSEP)
1293     ffound := false
1294     ffullpath := path
1295     for _, dir := range dirv {
1296         if 0 <= strings.Index(path,"*") { // by wild-card
1297             list, := ioutil.ReadDir(dir)
1298             ffullpath, ffound = showMatchFile(list,path,dir,argv)
1299         }else{
1300             ffullpath, ffound = showIfExecutable(path,dir,argv)
1301         }
1302         //if ffound && !isin("-a", argv) {
1303         if ffound && !showall {
1304             break;
1305         }
1306     }
1307     return []string{ffullpath}, ffound
1308 }
1309
1310 func stripLeadingWSArgv(argv[]string)([]string){
1311     for ; 0 < len(argv); {
1312         if len(argv[0]) == 0 {
1313             argv = argv[1:]
1314         }else{
1315             break
1316         }
1317     }
1318     return argv
1319 }
1320 func xEval(argv []string, nlenlnd bool){
1321     argv = stripLeadingWSArgv(argv)
1322     if len(argv) == 0 {
1323         fmt.Printf("eval [%s] [Go-expression]\n")
1324         return
1325     }
1326     pfmt := "%v"
1327     if argv[0][0] == '%' {
1328         pfmt = argv[0]
1329         argv = argv[1:]
1330     }
1331     if len(argv) == 0 {
1332         return
1333     }
1334     gocode := strings.Join(argv, " ")
1335     //fmt.Printf("eval [%v] [%v]\n",pfmt,gocode)
1336     fset := token.NewFileSet()
1337     rval, _ := types.Eval(fset,nil,token.NoPos,gocode)
1338     fmt.Printf(pfmt,rval.Value)
1339     if nlenlnd { fmt.Printf("\n") }
1340 }
1341
1342 func getval(name string) (found bool, val int) {
1343     /* should expand the name here */
1344     if name == "gsh.pid" {
1345         return true, os.Getpid()
1346     }else{
1347     if name == "gsh.ppid" {
1348         return true, os.Getppid()
1349     }
1350     return false, 0
1351 }
1352
1353 func echo(argv []string, nlenlnd bool){
1354     for ai := 1; ai < len(argv); ai++ {
1355         if 1 < ai {
1356             fmt.Printf(" ");
1357         }
1358         arg := argv[ai]
1359         found, val := getval(arg)
1360         if found {
1361             fmt.Printf("%d",val)
1362         }else{
1363             fmt.Printf("%s",arg)
1364         }
1365     }
1366 }

```

```

1365     }
1366     if nlen {
1367         fmt.Printf("\n")
1368     }
1369 }
1370
1371 func resfile() string {
1372     return "gsh.tmp"
1373 }
1374 //var resF *File
1375 func resmap() {
1376     //_, err := os.OpenFile(resfile(), os.O_RDWR|os.O_CREATE, os.ModeAppend)
1377     // https://developpaper.com/solution-to-golang-bad-file-descriptor-problem/
1378     // , err := os.OpenFile(resfile(), os.O_RDWR|os.O_CREATE, 0600)
1379     if err != nil {
1380         fmt.Printf("refF could not open: %s\n",err)
1381     }else{
1382         fmt.Printf("refF opened\n")
1383     }
1384 }
1385
1386 // @@2020-0821
1387 func gshScanArg(str string,strip int)(argv []string){
1388     var si = 0
1389     var sb = 0
1390     var inBracket = 0
1391     var argl = make([]byte,LINESIZE)
1392     var ax = 0
1393     debug := false
1394
1395     for ; si < len(str); si++ {
1396         if str[si] != ' ' {
1397             break
1398         }
1399     }
1400     sb = si
1401     for ; si < len(str); si++ {
1402         if sb <= si {
1403             if debug {
1404                 fmt.Printf("--Da- +%d %2d-%2d %s ... %s\n",
1405                         inBracket,sb,si,argl[0:ax],str[si:])
1406             }
1407         ch := str[si]
1408         if ch == '(' {
1409             inBracket += 1
1410             if 0 < strip && inBracket <= strip {
1411                 //fmt.Printf("stripLEV %d <= %d?\n",inBracket,strip)
1412                 continue
1413             }
1414         }
1415         if 0 < inBracket {
1416             if ch == ')' {
1417                 inBracket -= 1
1418                 if 0 < strip && inBracket < strip {
1419                     //fmt.Printf("stripLEV %d < %d?\n",inBracket,strip)
1420                     continue
1421                 }
1422             }
1423             argl[ax] = ch
1424             ax += 1
1425             continue
1426         }
1427         if str[si] == ' ' {
1428             argv = append(argv,string(argl[0:ax]))
1429             if debug {
1430                 fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
1431                         -1+len(argv),sb,si,str[sb:si],string(str[si:]))
1432             }
1433             sb = si+1
1434             ax = 0
1435             continue
1436         }
1437         argl[ax] = ch
1438         ax += 1
1439     }
1440     if sb < si {
1441         argv = append(argv,string(argl[0:ax]))
1442         if debug {
1443             fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
1444                 -1+len(argv),sb,si,string(argl[0:ax]),string(str[si:]))
1445         }
1446     }
1447     if debug {
1448         fmt.Printf("--Da- %d [%s] => [%d]%v\n",strip,str,len(argv),argv)
1449     }
1450 }
1451
1452 }
1453
1454 // should get stderr (into tmpfile ?) and return
1455 func (gsh*GshContext)Popen(name,mode string)(pin*os.File,pout*os.File,err bool){
1456     var pv = []int{-1,-1}
1457     syscall.Pipe(pv)
1458
1459     xarg := gshScanArg(name,1)
1460     name = strings.Join(xarg," ")
1461
1462     pin = os.NewFile(uintptr(pv[0]),"StdoutOf-"+name)
1463     pout = os.NewFile(uintptr(pv[1]),"StdinOf-"+name)
1464     fdir := 0
1465     dir := "?"
1466     if mode == "r" {
1467         dir = "<"
1468         fdir = 1 // read from the stdout of the process
1469     }else{
1470         dir = ">"
1471         fdir = 0 // write to the stdin of the process
1472     }
1473     gshPA := gsh.gshPA
1474     savfd := gshPA.Files[fdir]
1475
1476     var fd uintptr = 0
1477     if mode == "r" {
1478         fd = pout.Fd()
1479         gshPA.Files[fdir] = pout.Fd()
1480     }else{
1481         fd = pin.Fd()
1482         gshPA.Files[fdir] = pin.Fd()
1483     }
1484     // should do this by Goroutine?
1485     if false {
1486         fmt.Printf("--Ip- Opened fd[%v] %s %v\n",fd,dir,name)
1487         fmt.Printf("--RE01 [%d,%d,%d]->[%d,%d,%d]\n",
1488             os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd(),
1489

```

```

1489     pin.Fd(),pout.Fd(),pout.Fd())
1490 }
1491     savi := os.Stdin
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
1508 // <a name="ex-commands">External commands</a>
1509 func (gsh*GshContext)execCommand(exec bool, argv []string) (notf bool,exit bool) {
1510     if gsh.CmdTrace { fmt.Printf("--I-- excommand(%v)(%v)\n",exec,argv) }
1511
1512     gshPA := gsh.gshPA
1513     fullpathv, itis := which("PATH",[]string{"which",argv[0],"-s"})
1514     if itis == false {
1515         return true,false
1516     }
1517     fullpath := fullpathv[0]
1518     argv = unescapeWhiteSP(argv)
1519     if 0 < strings.Index(fullpath,".go") {
1520         argv := argv // ([]string{
1521         gofullpath, itis := which("PATH",[]string{"which","go","-s"})
1522         if itis == false {
1523             fmt.Println("-F-- Go not found\n")
1524             return false,true
1525         }
1526         gofullpath := gofullpathv[0]
1527         nargv = []string{gofullpath, "run", fullpath }
1528         fmt.Println("--I-- %s (%s %s)\n",gofullpath,
1529                     nargv[0],nargv[1],nargv[2])
1530         if exec {
1531             syscall.Exec(gofullpath,nargv,os.Environ())
1532         }else{
1533             pid, _ := syscall.ForkExec(gofullpath,nargv,&gshPA)
1534             if gsh.BackGround {
1535                 fmt.Fprintf(stderr,"--Ip- in Background pid[%d](%v)\n",pid,len(argv),nargv)
1536                 gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
1537             }else{
1538                 rusage := syscall.Rusage {}
1539                 syscall.Wait4(pid,nil,0,&rusage)
1540                 gsh.LastRusage = rusage
1541                 gsh.CmdCurrent.Rusagev[1] = rusage
1542             }
1543         }
1544     }else{
1545         if exec {
1546             syscall.Exec(fullpath,argv,os.Environ())
1547         }else{
1548             pid, _ := syscall.ForkExec(fullpath,argv,&gshPA)
1549             //fmt.Println("%d\n",pid); // '&' to be background
1550             if gsh.BackGround {
1551                 fmt.Fprintf(stderr,"--Ip- in Background pid[%d](%v)\n",pid,len(argv),argv)
1552                 gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
1553             }else{
1554                 rusage := syscall.Rusage {}
1555                 syscall.Wait4(pid,nil,0,&rusage);
1556                 gsh.LastRusage = rusage
1557                 gsh.CmdCurrent.Rusagev[1] = rusage
1558             }
1559         }
1560     }
1561     return false,false
1562 }
1563
1564 // <a name="builtin">Built-in Commands</a>
1565 func (gshCtx *GshContext) sleep(argv []string) {
1566     if len(argv) < 2 {
1567         fmt.Println("Sleep 100ms, 100us, 100ns, ...\n")
1568         return
1569     }
1570     duration := argv[1];
1571     d, err := time.ParseDuration(duration)
1572     if err != nil {
1573         d, err = time.ParseDuration(duration+"s")
1574     if err != nil {
1575         fmt.Println("duration ? %s (%s)\n",duration,err)
1576         return
1577     }
1578     }
1579     //fmt.Printf("Sleep %v\n",duration)
1580     time.Sleep(d)
1581     if 0 < len(argv[2:]) {
1582         gshCtx.gshellv(argv[2:])
1583     }
1584 }
1585 func (gshCtx *GshContext) repeat(argv []string) {
1586     if len(argv) < 2 {
1587         return
1588     }
1589     start0 := time.Now()
1590     for ri,_ := strconv.Atoi(argv[1]); 0 < ri; ri-- {
1591         if 0 < len(argv[2:]) {
1592             //start := time.Now()
1593             gshCtx.gshellv(argv[2:])
1594             end := time.Now()
1595             elps := end.Sub(start0);
1596             if( 1000000000 < elps ){
1597                 fmt.Printf("(repeat#%d %v)\n",ri,elps);
1598             }
1599         }
1600     }
1601 }
1602
1603 func (gshCtx *GshContext) gen(argv []string) {
1604     gshPA := gshCtx.gshPA
1605     if len(argv) < 2 {
1606         fmt.Println("Usage: %s N\n",argv[0])
1607         return
1608     }
1609     // should be repeated by "repeat" command
1610     count, _ := strconv.Atoi(argv[1])
1611     fd := gshPA.Files[1] // Stdout
1612     file := os.NewFile(fd,"internalStdOut")

```

```

1613     fmt.Printf("---I-- Gen. Count=%d to [%d]\n",count,file.Fd())
1614     //buf := []byte{}
1615     outdata := "0123 5678 0123 5678 0123 5678 0123 5678\r"
1616     for gi := 0; gi < count; gi++ {
1617         file.WriteString(outdata)
1618     }
1619     //file.WriteString("\n")
1620     fmt.Printf("\n(%d B)\n",count*len(outdata));
1621     //file.Close()
1622 }
1623
1624 // <a name="rexec">Remote Execution</a> // 2020-08-02
1625 func Elapsed(from time.Time)(string{
1626     elps := time.Now().Sub(from)
1627     if 1000000000 < elps {
1628         return fmt.Sprintf("[%5d.%02ds]",elps/1000000000,(elps%1000000000)/1000000)
1629     }else{
1630         if 1000000 < elps {
1631             return fmt.Sprintf("[%3d.%03dms]",elps/1000000,(elps%1000000)/1000)
1632         }else{
1633             return fmt.Sprintf("[%3d.%03dus]",elps/1000,(elps%1000))
1634         }
1635     }
1636     func abftime(nanos int64)(string){
1637         if 1000000000 < nanos {
1638             return fmt.Sprintf("%d.%02ds",nanos/1000000000,(nanos%1000000000)/1000000)
1639         }else{
1640             if 1000000 < nanos {
1641                 return fmt.Sprintf("%d.%03dms",nanos/1000000,(nanos%1000000)/1000)
1642             }else{
1643                 return fmt.Sprintf("%d.%03dus",nanos/1000,(nanos%1000))
1644             }
1645         }
1646     func abssize(size int64)(string){
1647         fsize := float64(size)
1648         if 1024*1024*1024 < size {
1649             return fmt.Sprintf("%.2fGiB",fsize/(1024*1024*1024))
1650         }else{
1651             if 1024*1024 < size {
1652                 return fmt.Sprintf("%.3fMiB",fsize/(1024*1024))
1653             }else{
1654                 return fmt.Sprintf("%.3fKiB",fsize/1024)
1655             }
1656         }
1657     func absize(size int64)(string){
1658         fsize := float64(size)
1659         if 1024*1024*1024 < size {
1660             return fmt.Sprintf("%.2fGiB",fsize/(1024*1024*1024))
1661         }else{
1662             if 1024*1024 < size {
1663                 return fmt.Sprintf("%.3fMiB",fsize/(1024*1024))
1664             }else{
1665                 return fmt.Sprintf("%.3fKiB",fsize/1024)
1666             }
1667         }
1668     func abbspeed(totalB int64,ns int64)(string{
1669         MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
1670         if 1000 <= MBs {
1671             return fmt.Sprintf("%6.3fGB/s",MBs/1000)
1672         }else{
1673             if 1 <= MBs {
1674                 return fmt.Sprintf("%6.3fMB/s",MBs)
1675             }else{
1676                 return fmt.Sprintf("%6.3fKB/s",MBs*1000)
1677             }
1678         }
1679     func absspeed(totalB int64,ns time.Duration)(string{
1680         MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
1681         if 1000 <= MBs {
1682             return fmt.Sprintf("%6.3fGBps",MBs/1000)
1683         }else{
1684             if 1 <= MBs {
1685                 return fmt.Sprintf("%6.3fMBps",MBs)
1686             }else{
1687                 return fmt.Sprintf("%6.3fKbps",MBs*1000)
1688             }
1689         }
1690     func fileRelay(what string,in*os.File,out*os.File,size int64,bsiz int)(wcount int64){
1691         Start := time.Now()
1692         buff := make([]byte,bsiz)
1693         var total int64 = 0
1694         var rem int64 = size
1695         nio := 0
1696         Prev := time.Now()
1697         var PrevSize int64 = 0
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             }
1713             Prev = Now;
1714             PrevSize = total
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             }
1725             rlen = rcc
1726             if string(buff[0:10]) == "((SoftEOF " {
1727                 var ecc int64 = 0
1728                 fmt.Sscanf(string(buff),"((SoftEOF %v",&ecc)
1729                 fmt.Printf(Elapsed(Start)+"--En- X: %s Recv ((SoftEOF %v))/%v\n",
1730                     what,ecc,total)
1731                 if ecc == total {
1732                     break
1733                 }
1734             }
1735         }
1736     }

```

```

1737     wlen := rlen
1738     if out != nil {
1739         wcc,err := out.Write(buff[0:rlen])
1740         if err != nil {
1741             fmt.Printf(Elapsed(Start)+"--En-- X: %s write(%v,%v)>%v\n",
1742                         what,wcc,err,out.Name())
1743             break
1744         }
1745         wlen = wcc
1746     }
1747     if wlen < rlen {
1748         fmt.Printf(Elapsed(Start)+"--En-- X: %s incomplete write (%v/%v)\n",
1749                         what,wlen,rlen)
1750         break;
1751     }
1752     nio += 1
1753     total += int64(rlen)
1754     rem -= int64(rlen)
1755     if rem <= 0 {
1756         break
1757     }
1758 }
1759 }
1760 Done := time.Now()
1761 Elps := float64(Done.Sub(Start))/1000000000 //Seconds
1762 TotalMB := float64(total)/1000000 //MB
1763 MBps := TotalMB / Elps
1764 fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v) %v %.3fMB/s\n",
1765                         what,total,size,nio,absize(total),MBps)
1766 return total
1767 }
1768 func tcpPush(clnt *os.File){
1769     // shrink socket buffer and recover
1770     usleep(100);
1771 }
1772 func (gsh*GshContext)RexecServer(argv[]string{
1773     debug := true
1774     Start0 := time.Now()
1775     Start := Start0
1776     // if local == ":" { local = "0.0.0.0:9999" }
1777     local := "0.0.0.0:9999"
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
1800 reqbuf := make([]byte,LINESIZE)
1801 res := ""
1802 for {
1803     fmt.Printf(Elapsed(Start0)+"--In- S: Listening at %s...\n",local);
1804     aconn, err := sconn.AcceptTCP()
1805     Start = time.Now()
1806     if err != nil {
1807         fmt.Printf(Elapsed(Start)+"--En- S: Accept error: %s (%s)\n",local,err)
1808         return
1809     }
1810     clnt, _ := aconn.File()
1811     fd := clnt.Fd()
1812     ar := aconn.RemoteAddr()
1813     if debug { fmt.Printf(Elapsed(Start0)+"--In- S: Accepted TCP at %s [%d] <- %v\n",
1814                 local,fd,ar) }
1815     res = fmt.Sprintf("220 GShell/%s Server\r\n",VERSION)
1816     fmt.Fprintf(clnt,"%s",res)
1817     if debug { fmt.Printf(Elapsed(Start)+"--In- S: %s",res) }
1818     count, err := clnt.Read(reqbuf)
1819     if err != nil {
1820         fmt.Printf(Elapsed(Start)+"--En- C: (%v %v) %v",
1821                     count,err,string(reqbuf))
1822     }
1823     req := string(reqbuf[:count])
1824     if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",string(req)) }
1825     reqv := strings.Split(string(req),"\\r")
1826     cmdv := gshScanArg(reqv[0],0)
1827     //cmdv := strings.Split(reqv[0]," ")
1828     switch cmdv[0] {
1829     case "HELP":
1830         res = fmt.Sprintf("250 %v",req)
1831     case "GET":
1832         // download (remotefile|-zN) [localfile]
1833         var dsize int64 = 32*1024*1024
1834         var bsize int = 64*1024
1835         var fname string = ""
1836         var in *os.File = nil
1837         var pseudoEOF = false
1838         if 1 < len(cmdv) {
1839             fname = cmdv[1]
1840             if strBegins(fname,"-z") {
1841                 fmt.Sscanf(fname[2:], "%d", &dsize)
1842             }else{
1843                 if strBegins(fname,"(") {
1844                     xin,xout,err := gsh.Popen(fname,"r")
1845                     if err {
1846                         if else{
1847                             xout.Close()
1848                             defer xin.Close()
1849                             in = xin
1850                             dsize = MaxStreamSize
1851                             pseudoEOF = true
1852                         }
1853                     }else{
1854                         xin,err := os.Open(fname)
1855                         if err != nil {
1856                             fmt.Printf("--En- GET (%v)\n",err)
1857                         }else{
1858                             defer xin.Close()
1859                             in = xin
1860                             fi,_ := xin.Stat()
1861                         }
1862                     }
1863                 }
1864             }
1865         }
1866     }
1867 }
```

```

1861             dszie = fi.Size()
1862         }
1863     }
1864 }
1865 //fmt.Printf(Elapsed(Start)+"--In- GET %v:%v\n",dszie,bsize)
1866 res = fmt.Sprintf("200 %v\r\n",dszie)
1867 fmt.Fprintf(clnt,"%v",res)
1868 tcpPush(clnt); // should be separated as line in receiver
1869 fmt.Printf(Elapsed(Start)+"--In- S: %v",res)
1870 wcount := fileRelay("SendGET",in,clnt,dszie,bsize)
1871 if pseudoEOF {
1872     in.Close() // pipe from the command
1873     // show end of stream data (its size) by OOB?
1874     SoftEOF := fmt.Sprintf("(SoftEOF %v)",wcount)
1875     fmt.Printf(Elapsed(Start)+"--In- S: Send %v\n",SoftEOF)
1876
1877     tcpPush(clnt); // to let SoftEOF data appear at the top of received data
1878     fmt.Fprintf(clnt,"%v\r\n",SoftEOF)
1879     tcpPush(clnt); // to let SoftEOF alone in a packet (separate with 200 OK)
1880     // with client generated random?
1881     //fmt.Printf("--In- L: close %v (%v)\n",in.Fd(),in.Name())
1882 }
1883 res = fmt.Sprintf("200 GET done\r\n")
1884 case "PUT":
1885     // upload {srcfile|-zN} {dstfile}
1886     var dszie int64 = 32*1024*1024
1887     var bsize int = 64*1024
1888     var fname string = ""
1889     var out *os.File = nil
1890     if 1 < len(cmdv) { // localfile
1891         fmt.Sscanf(cmdv[1],"%d",&dszie)
1892     }
1893     if 2 < len(cmdv) {
1894         fname = cmdv[2]
1895         if fname == "-" {
1896             // nul dev
1897         }else{
1898             if strBegins(fname,"(") {
1899                 xin,xout,err := gsh.Popen(fname,"w")
1900                 if err {
1901                     }else{
1902                         xin.Close()
1903                         defer xout.Close()
1904                         out = xout
1905                     }
1906                 }else{
1907                     // should write to temporary file
1908                     // should suppress 'C on tty'
1909                     xout,err := os.OpenFile(fname,os.O_CREATE|os.O_RDWR|os.O_TRUNC,0600)
1910                     //fmt.Printf("--In- S: open(%v) out(%v) err(%v)\n",fname,xout,err)
1911                     if err != nil {
1912                         fmt.Printf("--En- PUT (%v)\n",err)
1913                     }else{
1914                         out = xout
1915                     }
1916                 }
1917                 fmt.Printf(Elapsed(Start)+"--In- L: open(%v,w) %v (%v)\n",
1918                         fname,local,err)
1919             }
1920             fmt.Printf(Elapsed(Start)+"--In- PUT %v (%v)\n",dszie,bsize)
1921             fmt.Printf(Elapsed(Start)+"--In- S: 200 %v OK\r\n",dszie)
1922             fmt.Fprintf(clnt,"200 %v OK\r\n",dszie)
1923             fileRelay("RecvPUT",clnt,out,dszie,bsize)
1924             res = fmt.Sprintf("200 PUT done\r\n")
1925         default:
1926             res = fmt.Sprintf("400 What? %v",req)
1927         }
1928         swcc,serr := clnt.Write([]byte(res))
1929         if serr != nil {
1930             fmt.Printf(Elapsed(Start)+"--In- S: (wc=%v er=%v) %v",swcc,serr,res)
1931         }else{
1932             fmt.Printf(Elapsed(Start)+"--In- S: %v",res)
1933         }
1934         aconn.Close();
1935         clnt.Close();
1936     }
1937     sconn.Close();
1938 }
1939 func (gsh*GshContext)RexecClient(argv[]string)(int,string){
1940     debug := true
1941     Start := time.Now()
1942     if len(argv) == 1 {
1943         return -1,"EmptyARG"
1944     }
1945     argv = argv[1:]
1946     if argv[0] == "-serv" {
1947         gsh.RexecServer(argv[1:])
1948         return 0,"Server"
1949     }
1950     remote := "0.0.0.0:9999"
1951     if argv[0][0] == '@' {
1952         remote = argv[0][1:]
1953         argv = argv[1:]
1954     }
1955     if argv[0] == "-s" {
1956         debug = false
1957         argv = argv[1:]
1958     }
1959     dport, err := net.ResolveTCPAddr("tcp",remote);
1960     if err != nil {
1961         fmt.Printf(Elapsed(Start)+"Address error: %s (%s)\n",remote,err)
1962         return -1,"AddressError"
1963     }
1964     fmt.Printf(Elapsed(Start)+"--In- C: Connecting to %s\n",remote)
1965     serv, err := net.DialTCP("tcp",nil,dport)
1966     if err != nil {
1967         fmt.Printf(Elapsed(Start)+"Connection error: %s (%s)\n",remote,err)
1968         return -1,"CannotConnect"
1969     }
1970     if debug {
1971         al := serv.LocalAddr()
1972         fmt.Printf(Elapsed(Start)+"--In- C: Connected to %v <- %v\n",remote,al)
1973     }
1974
1975     req := ""
1976     res := make([]byte,LINESEIZE)
1977     count,err := serv.Read(res)
1978     if err != nil {
1979         fmt.Printf("--En- S: (%d,%v) %v",count,err,string(res))
1980     }
1981     if debug { fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res)) }
1982
1983     if argv[0] == "GET" {
1984         savPA := gsh.gshPA

```

```

1985 var bsize int = 64*1024
1986 req = fmt.Sprintf("%v\r\n",strings.Join(argv, " "))
1987 fmt.Printf(Elapsed(Start)+"--In- C: %v",req)
1988 fmt.Fprintf(serv,req)
1989 count,err = serv.Read(res)
1990 if err != nil {
1991 }else{
1992     var dszie int64 = 0
1993     var out *os.File = nil
1994     var out_tobeclosed *os.File = nil
1995     var fname string = ""
1996     var rcode int = 0
1997     var pid int = -1
1998     fmt.Sscanf(string(res),"%d %d",&rcode,&dszie)
1999     fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res[0:count]))
2000     if 3 <= len(argv) {
2001         fname = argv[2]
2002         if strBegins(fname,"{") {
2003             xin,xout,err := gsh.Popen(fname,"w")
2004             if err {
2005                 }else{
2006                     xin.Close()
2007                     defer xout.Close()
2008                     out = xout
2009                     out_tobeclosed = xout
2010                     pid = 0 // should be its pid
2011                 }
2012             }else{
2013                 // should write to temporary file
2014                 // should suppress ^C on tty
2015                 xout,err := os.OpenFile(fname,os.O_CREATE|os.O_RDWR|os.O_TRUNC,0600)
2016                 if err != nil {
2017                     fmt.Println("--En- %v\n",err)
2018                 }
2019                 out = xout
2020                 //fmt.Printf("--In-- %d > %s\n",out.Fd(),fname)
2021             }
2022         }
2023         in,_ := serv.File()
2024         fileRelay("RecvGET",in,out,dszie,bsize)
2025         if 0 <= pid {
2026             gsh.gshPA = savPA // recovery of Fd(), and more?
2027             fmt.Printf(Elapsed(Start)+"--In- L: close Pipe > %v\n",fname)
2028             out_tobeclosed.Close()
2029             //syscall.Wait4(pid,nil,0,nil) //@@
2030         }
2031     }
2032 }else
2033 if argv[0] == "PUT" {
2034     remote,_ := serv.File()
2035     var local *os.File = nil
2036     var dszie int64 = 32*1024*1024
2037     var bsize int = 64*1024
2038     var ofile string = ""
2039     //fmt.Printf("--I-- Rex %v\n",argv)
2040     if 1 < len(argv) {
2041         fname := argv[1]
2042         if strBegins(fname,"-z") {
2043             fmt.Sscanf(fname[2:], "%d", &dszie)
2044         }else
2045             if strBegins(fname,"{") {
2046                 xin,xout,err := gsh.Popen(fname,"r")
2047                 if err {
2048                     }else{
2049                         xout.Close()
2050                         defer xin.Close()
2051                         /in = xin
2052                         local = xin
2053                         fmt.Printf("--In- [%d] < Upload output of %v\n",
2054                             local.Fd(),fname)
2055                         ofile = "-from."+fname
2056                         dszie = MaxStreamSize
2057                     }
2058                 }else{
2059                     xlocal,err := os.Open(fname)
2060                     if err != nil {
2061                         fmt.Println("--En- (%s)\n",err)
2062                         local = nil
2063                     }else{
2064                         local = xlocal
2065                         fi,_ := local.Stat()
2066                         dszie = fi.Size()
2067                         defer local.Close()
2068                         //fmt.Printf("--I-- Rex in(%v / %v)\n",ofile,dszie)
2069                     }
2070                     ofile = fname
2071                     fmt.Printf(Elapsed(Start)+"--In- L: open(%v,r)=%v %v (%v)\n",
2072                         fname,dszie,local,err)
2073                 }
2074     }else
2075     if 2 < len(argv) && argv[2] != "" {
2076         ofile = argv[2]
2077         //fmt.Printf("(%)v B.ofile=%v\n",len(argv),argv,ofile)
2078     }
2079     //fmt.Printf(Elapsed(Start)+"--I-- Rex out(%v)\n",ofile)
2080     fmt.Printf(Elapsed(Start)+"--In- PUT %v (%v)\n",dszie,bsize)
2081     req = fmt.Sprintf("PUT %v %v \r\n",dszie,ofile)
2082     if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
2083     fmt.Fprintf(serv,"%v",req)
2084     count,err = serv.Read(res)
2085     if debug { fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res[0:count])) }
2086     fileRelay("SendPUT",local,remote,dszie,bsize)
2087 }else{
2088     req = fmt.Sprintf("%v\r\n",strings.Join(argv, " "))
2089     if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
2090     fmt.Fprintf(serv,"%v",req)
2091     //fmt.Printf("--In- sending RexRequest(%v)\n",len(req))
2092 }
2093 //fmt.Printf(Elapsed(Start)+"--In- waiting RexResponse...\n")
2094 count,err = serv.Read(res)
2095 ress := ""
2096 if count == 0 {
2097     ress = "(nil)\r\n"
2098 }else{
2099     ress = string(res[:count])
2100 }
2101 if err != nil {
2102     fmt.Printf(Elapsed(Start)+"--En- S: (%d,%v) %v",count,err,ress)
2103 }else{
2104     fmt.Printf(Elapsed(Start)+"--In- S: %v",ress)
2105 }
2106 serv.Close()
2107 //conn.Close()
2108

```

```

2109     var stat string
2110     var rcode int
2111     fmt.Sscanf(res,"%d %s",&rcode,&stat)
2112     //fmt.Printf("--D-- Client: %v (%v)",rcode,stat)
2113     return rcode,res
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             }
2159             continue
2160         }
2161         upload = true
2162         dstv = append(dstv,file)
2163         continue
2164     }
2165     /*
2166     idx := strings.Index(v,":")
2167     if 0 <= idx {
2168         remote = v[0:idx]
2169         if len(srcv) == 0 {
2170             download = true
2171             srcv = append(srcv,v[idx+1:])
2172             continue
2173         }
2174         upload = true
2175         dstv = append(dstv,v[idx+1:])
2176         continue
2177     }
2178     */
2179     if download {
2180         dstv = append(dstv,v)
2181     }else{
2182         srcv = append(srcv,v)
2183     }
2184 }
2185 hostport := "@" + host + ":" + port
2186 if upload {
2187     if host != "" { xargv = append(xargv,hostport) }
2188     xargv = append(xargv,"PUT")
2189     xargv = append(xargv,srcv[0]...)
2190     xargv = append(xargv,dstv[0]...)
2191     //fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v // %v\n",hostport,dstv,srcv,xargv)
2192     fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v\n",hostport,dstv,srcv)
2193     gsh.RexecClient(xargv)
2194 }else
2195     if download {
2196         if host != "" { xargv = append(xargv,hostport) }
2197         xargv = append(xargv,"GET")
2198         xargv = append(xargv,srcv[0]...)
2199         xargv = append(xargv,dstv[0]...)
2200     //fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v // %v\n",hostport,srcv,dstv,xargv)
2201     fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v\n",hostport,srcv,dstv)
2202     gsh.RexecClient(xargv)
2203 }else{
2204 }
2205 }
2206
2207 // target
2208 func (gsh*GshContext)Trelpath(rloc string)(string){
2209     cwd, _ := os.Getwd()
2210     os.Chdir(gsh.RWD)
2211     os.Chdir(rloc)
2212     twd, _ := os.Getwd()
2213     os.Chdir(cwd)
2214
2215     tpath := twd + "/" + rloc
2216     return tpath
2217 }
2218 // join to remote GShell - [user@]host[:port] or cd host:[port]:path
2219 func (gsh*GshContext)Rjoin(argv[]string){
2220     if len(argv) <= 1 {
2221         fmt.Printf("--I-- current server = %v\n",gsh.RSERV)
2222         return
2223     }
2224     serv := argv[1]
2225     servv := strings.Split(servv,":")
2226     if 1 <= len(servv) {
2227         if servv[0] == "lo" {
2228             servv[0] = "localhost"
2229         }
2230     }
2231     switch len(servv) {
2232         case 1:

```

```

2233     //if strings.Index(serv,:) < 0 {
2234     serv = serv[0] + ":" + fmt.Sprintf("%d",GSH_PORT)
2235     //}
2236     case 2: // host:port
2237     serv = strings.Join(servv,:")
2238   }
2239   xargv := []string{"rex-join","@"+serv,"HELO"}
2240   rcode,stat := gsh.RexecClient(xargv)
2241   if (rcode / 100) == 2 {
2242     fmt.Printf("--I-- OK Joined (%v) %v\n",rcode,stat)
2243     gsh.RSERV = serv
2244   }else{
2245     fmt.Printf("--I-- NG, could not joined (%v) %v\n",rcode,stat)
2246   }
2247 }
2248 func (gsh*GshContext)Rexec(argv[]string){
2249   if len(argv) <= 1 {
2250     fmt.Println("--I-- rexec command [ | {file || {command} ]\n",gsh.RSERV)
2251     return
2252   }
2253   /*
2254   nargv := gshScanArg(strings.Join(argv," "),0)
2255   fmt.Println("--D-- nargc=%d [%v]\n",len(nargv),nargv)
2256   if nargv[1][0] != '{' {
2257     nargv[1] = "(" + nargv[1] + ")"
2258     fmt.Println("--D-- nargc=%d [%v]\n",len(nargv),nargv)
2259   }
2260   argv = nargv
2261   */
2262   nargv := []string{}
2263   nargv = append(nargv,""+strings.Join(argv[1:]," ")+"")
2264   fmt.Println("--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.Println("--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   }
2310   fmt.Println("--I-- jput from=%v to=%v\n",local,gsh.Trepath(remote))
2311 }
2312 func (gsh*GshContext)Rget(argv[]string){
2313   var remote string = ""
2314   var local string = ""
2315   if 1 < len(argv) {
2316     remote = argv[1]
2317     local = remote // base name
2318   }
2319   if 2 < len(argv) {
2320     local = argv[2]
2321   }
2322   fmt.Println("--I-- jget from=%v to=%v\n",gsh.Trepath(remote),local)
2323 }
2324
2325 // <a name="network">network</a>
2326 // -s, -si, -so // bi-directional, source, sync (maybe socket)
2327 func (gshCtx*GshContext)sconnect(inTCP bool, argv []string) {
2328   gshPA := gshCtx.gshPA
2329   if len(argv) < 2 {
2330     fmt.Printf("Usage: -s [host]:[port[.udp]]\n")
2331     return
2332   }
2333   remote := argv[1]
2334   if remote == ":" { remote = "0.0.0.0:9999" }
2335
2336   if inTCP { // TCP
2337     dport, err := net.ResolveTCPAddr("tcp",remote);
2338     if err != nil {
2339       fmt.Printf("Address error: %s (%s)\n",remote,err)
2340       return
2341     }
2342     conn, err := net.DialTCP("tcp",nil,dport)
2343     if err != nil {
2344       fmt.Printf("Connection error: %s (%s)\n",remote,err)
2345       return
2346     }
2347     file, _ := conn.File();
2348     fd := file.Fd()
2349     fmt.Printf("Socket: connected to %s, socket(%d)\n",remote,fd)
2350
2351     savfd := gshPA.Files[1]
2352     gshPA.Files[1] = fd;
2353     gshCtx.gshelly(argv[2:])
2354     gshPA.Files[1] = savfd
2355     file.Close()
2356     conn.Close()

```

```

2357     }else{
2358         //dport, err := net.ResolveUDPAddr("udp4",remote);
2359         dport, err := net.ResolveUDPAddr("udp",remote);
2360         if err != nil {
2361             fmt.Printf("Address error: %s (%s)\n",remote,err)
2362             return
2363         }
2364         //conn, err := net.DialUDP("udp4",nil,dport)
2365         conn, err := net.DialUDP("udp",nil,dport)
2366         if err != nil {
2367             fmt.Printf("Connection error: %s (%s)\n",remote,err)
2368             return
2369         }
2370         file, _:= conn.File();
2371         fd := file.Fd()
2372
2373         ar := conn.RemoteAddr()
2374         //al := conn.LocalAddr()
2375         fmt.Printf("Socket: connected to %s [%s], socket[%d]\n",
2376             remote,ar.String(),fd)
2377
2378         savfd := gshPA.Files[1]
2379         gshPA.Files[1] = fd;
2380         gshCtx.gshellv(argv[2:])
2381         gshPA.Files[1] = savfd
2382         file.Close()
2383         conn.Close()
2384     }
2385 }
2386 func (gshCtx*GshContext)xaccept(inTCP bool, argv []string) {
2387     gshPA := gshCtx.gshPA
2388     if len(argv) < 2 {
2389         fmt.Printf("Usage: -ac [host]:[port[.udp]]\n")
2390         return
2391     }
2392     local := argv[1]
2393     if local == ":" { local = "0.0.0.0:9999" }
2394     if inTCP { // TCP
2395         port, err := net.ResolveTCPAddr("tcp",local);
2396         if err != nil {
2397             fmt.Printf("Address error: %s (%s)\n",local,err)
2398             return
2399         }
2400         //fmt.Printf("Listen at %s...\n",local);
2401         sconn, err := net.ListenTCP("tcp", port)
2402         if err != nil {
2403             fmt.Printf("Listen error: %s (%s)\n",local,err)
2404             return
2405         }
2406         //fmt.Printf("Accepting at %s...\n",local);
2407         aconn, err := sconn.AcceptTCP()
2408         if err != nil {
2409             fmt.Printf("Accept error: %s (%s)\n",local,err)
2410             return
2411         }
2412         file, _:= aconn.File()
2413         fd := file.Fd()
2414         fmt.Printf("Accepted TCP at %s [%d]\n",local,fd)
2415
2416         savfd := gshPA.Files[0]
2417         gshPA.Files[0] = fd;
2418         gshCtx.gshellv(argv[2:])
2419         gshPA.Files[0] = savfd
2420
2421         sconn.Close();
2422         aconn.Close();
2423         file.Close();
2424     }else{
2425         //port, err := net.ResolveUDPAddr("udp4",local);
2426         port, err := net.ResolveUDPAddr("udp",local);
2427         if err != nil {
2428             fmt.Printf("Address error: %s (%s)\n",local,err)
2429             return
2430         }
2431         fmt.Printf("Listen UDP at %s...\n",local);
2432         //uconn, err := net.ListenUDP("udp4", port)
2433         uconn, err := net.ListenUDP("udp", port)
2434         if err != nil {
2435             fmt.Printf("Listen error: %s (%s)\n",local,err)
2436             return
2437         }
2438         file, _:= uconn.File()
2439         fd := file.Fd()
2440         ar := uconn.RemoteAddr()
2441         remote := ""
2442         if ar != nil { remote = ar.String() }
2443         if remote == "" { remote = "?" }
2444
2445         // not yet received
2446         //fmt.Printf("Accepted at %s [%d] <- %s\n",local,fd,"")
2447
2448         savfd := gshPA.Files[0]
2449         gshPA.Files[0] = fd;
2450         savenv := gshPA.Env
2451         gshPA.Env = append(savenv, "REMOTE_HOST="+remote)
2452         gshCtx.gshellv(argv[2:])
2453         gshPA.Env = savenv
2454         gshPA.Files[0] = savfd
2455
2456         uconn.Close();
2457         file.Close();
2458     }
2459 }
2460
2461 // empty line command
2462 func (gshCtx*GshContext)xPwd(argv[]string){
2463     // execute context command, pwd + date
2464     // context notation, representation scheme, to be resumed at re-login
2465     cwd, _ := os.Getwd()
2466     switch {
2467     case isin("-",argv):
2468         gshCtx.ShowChdirHistory(argv)
2469     case isin("ls",argv):
2470         showFileInfo(cwd,argv)
2471     default:
2472         fmt.Println("%s\n",cwd)
2473     case isin("-v",argv): // obsolete emtpy command
2474         t := time.Now()
2475         date := t.Format(time.UnixDate)
2476         exe, _ := os.Executable()
2477         host, _ := os.Hostname()
2478         fmt.Printf("{PWD=%s\n", cwd)
2479         fmt.Printf("HOST=%s\n", host)
2480         fmt.Printf(" DATE=%s\n",date)
2481     }
2482 }
```

```

2481     fmt.Printf(" TIME=%s\"",t.String())
2482     fmt.Printf(" PID=%d\"",os.Getpid())
2483     fmt.Printf(" EXE=\"%s\"",exe)
2484     fmt.Printf("\n")
2485 }
2486 }
2487
2488 // <a name="history">History</a>
2489 // these should be browsed and edited by HTTP browser
2490 // show the time of command with -t and direcotry with -ls
2491 // openfile-history, sort by -a -m -c
2492 // sort by elapsed time by -t -s
2493 // search by "more" like interface
2494 // edit history
2495 // sort history, and wc or uniq
2496 // CPU and other resource consumptions
2497 // limit showing range (by time or so)
2498 // export / import history
2499 func (gshCtx *GshContext)xHistory(argv []string){
2500     atWorkDirX := -1
2501     if i < len(argv) && strBegins(argv[1],"-") {
2502         atWorkDirX,_ = strconv.Atoi(argv[1][1:])
2503     }
2504     //fmt.Printf("--D-- showHistory(%v)\n",argv)
2505     for i, v := range gshCtx.CommandHistory {
2506         // exclude commands not to be listed by default
2507         // internal commands may be suppressed by default
2508         if v.CmdLine == "" && !isin("-a",argv) {
2509             continue;
2510         }
2511         if 0 <= atWorkDirX {
2512             if v.WorkDirX != atWorkDirX {
2513                 continue
2514             }
2515         }
2516         if !isin("-n",argv){ // like "fc"
2517             fmt.Printf("%-2d ",i)
2518         }
2519         if isin("-v",argv){
2520             fmt.Println(v) // should be with it date
2521         }else{
2522             if isin("-l",argv) || isin("-lo",argv) {
2523                 elps := v.EndAt.Sub(v.StartAt);
2524                 start := v.StartAt.Format(time.Stamp)
2525                 fmt.Printf("%d %v",v.WorkDirX)
2526                 fmt.Printf("%v %1v/t ",start,elps)
2527             }
2528             if isin("-l",argv) && !isin("-lo",argv){
2529                 fmt.Printf("%v",Rusagef("%t/%s",argv,v.Rusage))
2530             }
2531             if isin("-at",argv) { // isin("-ls",argv){
2532                 hdi := v.WorkDirX // workdir history index
2533                 fmt.Printf("%d %v",hdi,v.WorkDir)
2534                 // show the FileInfo of the output command??
2535             }
2536             fmt.Println(v.CmdLine)
2537             fmt.Println("\n")
2538         }
2539     }
2540 }
2541 // in - history index
2542 func searchHistory(gshCtx GshContext, gline string) (string, bool, bool){
2543     if gline[0] == '!' {
2544         hix, err := strconv.Atoi(gline[1:])
2545         if err != nil {
2546             fmt.Printf("--E-- (%s : range)\n",hix)
2547             return "", false, true
2548         }
2549         if hix < 0 || len(gshCtx.CommandHistory) <= hix {
2550             fmt.Printf("--E-- (%d : out of range)\n",hix)
2551             return "", false, true
2552         }
2553         return gshCtx.CommandHistory[hix].CmdLine, false, false
2554     }
2555     // search
2556     //for i, v := range gshCtx.CommandHistory {
2557     //}
2558     return gline, false, false
2559 }
2560 func (gsh*GshContext)cmdStringInHistory(hix int)(cmd string, ok bool){
2561     if 0 <= hix && hix < len(gsh.CommandHistory) {
2562         return gsh.CommandHistory[hix].CmdLine,true
2563     }
2564     return "",false
2565 }
2566
2567 // temporary adding to PATH environment
2568 // cd name -lib for LD_LIBRARY_PATH
2569 // chdir with directory history (date + full-path)
2570 // -s for sort option (by visit date or so)
2571 func (gsh*GshContext>ShowChdirHistory(i int,v GChdirHistory, argv []string){
2572     fmt.Printf("%-2d ",v.CmdIndex) // the first command at this WorkDir
2573     fmt.Printf("%d ",i)
2574     fmt.Printf("%v ",v.MovedAt.Format(time.Stamp))
2575     showFileInfo(v.Dir,argv)
2576 }
2577 func (gsh*GshContext>ShowChdirHistory(argv []string){
2578     for i, v := range gsh.ChdirHistory {
2579         gsh.ShowChdirHistory(i,v,argv)
2580     }
2581 }
2582 func skipOpts(argv[]string)(int){
2583     for i,v := range argv {
2584         if strBegins(v,"-") {
2585             }else{
2586                 return i
2587             }
2588         }
2589     return -1
2590 }
2591 func (gshCtx*GshContext)xChdir(argv []string){
2592     cdhist := gshCtx.ChdirHistory
2593     if isin("?",argv) || isin("-t",argv) || isin("-a",argv) {
2594         gshCtx.ShowChdirHistory(argv)
2595         return
2596     }
2597     pwd, _ := os.Getwd()
2598     dir := ""
2599     if len(argv) <= 1 {
2600         dir = toFullPath("-")
2601     }else{
2602         i := skipOpts(argv[1:])
2603         if i < 0 {
2604             dir = toFullPath("-")
2605         }
2606     }
2607 }

```

```

2605     }else{
2606         dir = argv[1+i]
2607     }
2608 }
2609 if strBegins(dir,"@") {
2610     if dir == "@0" { // obsolete
2611         dir = gshCtx.StartDir
2612     }else
2613     if dir == "@!" {
2614         index := len(cdhist) - 1
2615         if 0 < index { index -= 1 }
2616         dir = cdhist[index].Dir
2617     }else{
2618         index, err := strconv.Atoi(argv[1:])
2619         if err != nil {
2620             fmt.Printf("--E-- xChdir(%v)\n",err)
2621             dir = "?"
2622         }else
2623         if len(gshCtx.CkdirHistory) <= index {
2624             fmt.Printf("--E-- xChdir(history range error)\n")
2625             dir = "?"
2626         }else{
2627             dir = cdhist[index].Dir
2628         }
2629     }
2630 }
2631 if dir != "?" {
2632     err := os.Ckdir(dir)
2633     if err != nil {
2634         fmt.Printf("--E-- xChdir(%s)(%v)\n",argv[1],err)
2635     }else{
2636         cwd, _ := os.Getwd()
2637         if cwd != pwd {
2638             hist1 := GChdirHistory { }
2639             hist1.Dir = cwd
2640             hist1.MovedAt = time.Now()
2641             hist1.CmdIndex = len(gshCtx.CommandHistory)+1
2642             gshCtx.CkdirHistory = append(cdhist,hist1)
2643             if !isin("-s",argv){
2644                 // cwd, _ := os.Getwd()
2645                 //fmt.Printf("%s\n", cwd)
2646                 ix := len(gshCtx.CkdirHistory)-1
2647                 gshCtx.ShowCkdirHistory1(ix,hist1,argv)
2648             }
2649         }
2650     }
2651 }
2652 if isin("-ls",argv){
2653     cwd, _ := os.Getwd()
2654     showFileInfo(cwd,argv);
2655 }
2656 }
2657 func TimeValSub(tv1 *syscall.Timeval, tv2 *syscall.Timeval){
2658     *tv1 = syscall.NsecToTimeval(tv1.Nano() - tv2.Nano())
2659 }
2660 func RusageSubv(rul, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
2661     TimeValSub(&rul[0].Utime,&ru2[0].Utime)
2662     TimeValSub(&rul[0].Stime,&ru2[0].Stime)
2663     TimeValSub(&rul[1].Utime,&ru2[1].Utime)
2664     TimeValSub(&rul[1].Stime,&ru2[1].Stime)
2665     return rul
2666 }
2667 func TimeValAdd(tv1 syscall.Timeval, tv2 syscall.Timeval)(syscall.Timeval){
2668     tvs := syscall.NsecToTimeval(tv1.Nano() + tv2.Nano())
2669     return tvs
2670 }
2671 /*
2672 func RusageAddv(rul, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
2673     TimeValAdd(rul[0].Utime,ru2[0].Utime)
2674     TimeValAdd(rul[0].Stime,ru2[0].Stime)
2675     TimeValAdd(rul[1].Utime,ru2[1].Utime)
2676     TimeValAdd(rul[1].Stime,ru2[1].Stime)
2677     return rul
2678 }
2679 */
2680
2681 // <a name="rusage">Resource Usage</a>
2682 func Rusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2683     // ru[0] self , ru[1] children
2684     ut := TimeValadd(ru[0].Utime,ru[1].Utime)
2685     st := TimeValadd(ru[0].Stime,ru[1].Stime)
2686     uu := (ut.Sec*1000000 + int64(ut.Usec)) * 1000
2687     su := (st.Sec*1000000 + int64(st.Usec)) * 1000
2688     tu := uu + su
2689     ret := fmt.Sprintf("@v/sum",abbttime(tu))
2690     ret += fmt.Sprintf(", @v/usr",abbttime(uu))
2691     ret += fmt.Sprintf(", @v/sys",abbttime(su))
2692     return ret
2693 }
2694 func Rusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2695     ut := TimeValadd(ru[0].Utime,ru[1].Utime)
2696     st := TimeValadd(ru[0].Stime,ru[1].Stime)
2697     fmt.Printf("@d.@06ds/u @.Sec,@.Usec") //ru[1].Utime.Sec,ru[1].Utime.Usec)
2698     fmt.Printf("@d.@06ds/s @.Sec,st.Usec") //ru[1].Stime.Sec,ru[1].Stime.Usec)
2699     return ""
2700 }
2701 func Getrusagev()([2]syscall.Rusage){
2702     var rvv = [2]syscall.Rusage{}
2703     syscall.Getrusage(syscall.RUSAGE_SELF,&rvv[0])
2704     syscall.Getrusage(syscall.RUSAGE_CHILDREN,&rvv[1])
2705     return rvv
2706 }
2707 func ShowRusage(what string,argv []string, ru *syscall.Rusage){
2708     fmt.Printf("@s: %s",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     fmt.Printf(" Rss=@vB",ru.Maxrss)
2712     if isin("-l",argv) {
2713         fmt.Printf(" MinFlt=@v",ru.Minflt)
2714         fmt.Printf(" MajFlt=@v",ru.Majflt)
2715         fmt.Printf(" IxRSS=@vB",ru.Ixrss)
2716         fmt.Printf(" IdRSS=@vB",ru.Idrss)
2717         fmt.Printf(" Nswap=@vB",ru.Nswap)
2718         fmt.Printf(" Read=@v",ru.Inblock)
2719         fmt.Printf(" Write=@v",ru.Outblock)
2720     }
2721     fmt.Printf(" Snd=@v",ru.Msgsnd)
2722     fmt.Printf(" Rcv=@v",ru.Msgrcv)
2723     //if isin("-l",argv) {
2724     //    fmt.Printf(" Sig=@v",ru.Nsignals)
2725     //}
2726     fmt.Printf("\n");
2727 }
2728 func (gshCtx *GshContext)xTime(argv []string)(bool){

```

```

2729     if 2 <= len(argv){
2730         gshCtx.LastRusage = syscall.Rusage{}
2731         rusagev1 := Getrusagev()
2732         fin := gshCtx.gshellv(argv[1:])
2733         rusagev2 := Getrusagev()
2734         showRusage(argv[1], argv, &gshCtx.LastRusage)
2735         rusagev := RusageSubv(rusagev2, rusagev1)
2736         showRusage("self", argv, &rusagev[0])
2737         showRusage("child", argv, &rusagev[1])
2738         return fin
2739     }else{
2740         rusage:= syscall.Rusage {}
2741         syscall.Getrusage(syscall.RUSAGE_SELF,&rusage)
2742         showRusage("self",argv, &rusage)
2743         syscall.Getrusage(syscall.RUSAGE_CHILDREN,&rusage)
2744         showRusage("child",argv, &rusage)
2745         return false
2746     }
2747 }
2748 func (gshCtx *GshContext)xJobs(argv[]string){
2749     fmt.Printf("%d Jobs\n",len(gshctx.BackGroundJobs))
2750     for ji, pid := range gshCtx.BackGroundJobs {
2751         //wstat := syscall.WaitStatus {0}
2752         rusage := syscall.Rusage {}
2753         //wpid, err := syscall.Wait4(pid,&wstat,syscall.WNOHANG,&rusage);
2754         wpid, err := syscall.Wait4(pid,nil,syscall.WNOHANG,&rusage);
2755         if err != nil {
2756             fmt.Printf("--E-- %d (%v)\n",ji,pid,err)
2757         }else{
2758             fmt.Printf("%d(%d)\n",ji,pid,wpid)
2759             showRusage("child",argv, &rusage)
2760         }
2761     }
2762 }
2763 func (gsh*GshContext)inBackground(argv[]string)(bool){
2764     if gsh.CmdTrace { fmt.Println("--I-- inBackground(%v)\n",argv) }
2765     gsh.BackGround = true // set background option
2766     xfin := false
2767     xfin = gsh.gshellv(argv)
2768     gsh.BackGround = false
2769     return xfin
2770 }
2771 // -o file without command means just opening it and refer by #N
2772 // should be listed by "files" command
2773 func (gshCtx*GshContext)xOpen(argv[]string){
2774     var pv = []int{-1,-1}
2775     err := syscall.Pipe(pv)
2776     fmt.Printf("--I-- pipe()=%#d,#%d(%v)\n",pv[0],pv[1],err)
2777 }
2778 func (gshCtx*GshContext)fromPipe(argv[]string){
2779 }
2780 func (gshCtx*GshContext)xClose(argv[]string){
2781 }
2782
2783 // <a name="redirect">redirect</a>
2784 func (gshCtx*GshContext)redirect(argv[]string)(bool){
2785     if len(argv) < 2 {
2786         return false
2787     }
2788     cmd := argv[0]
2789     fname := argv[1]
2790     var file *os.File = nil
2792     ffix := 0
2794     mode := os.O_RDONLY
2795
2796     switch {
2797     case cmd == "-i" || cmd == "<":
2798         ffix = 0
2799         mode = os.O_RDONLY
2800     case cmd == "-o" || cmd == ">":
2801         ffix = 1
2802         mode = os.O_RDWR | os.O_CREATE
2803     case cmd == "-a" || cmd == ">>":
2804         ffix = 1
2805         mode = os.O_RDWR | os.O_CREATE | os.O_APPEND
2806     }
2807     if fname[0] == '#' {
2808         fd, err := strconv.Atoi(fname[1:])
2809         if err != nil {
2810             fmt.Printf("--E-- (%v)\n",err)
2811             return false
2812         }
2813         file = os.NewFile(uintptr(fd),"MaybePipe")
2814     }else{
2815         xfile, err := os.OpenFile(argv[1], mode, 0600)
2816         if err != nil {
2817             fmt.Printf("--E-- (%s)\n",err)
2818             return false
2819         }
2820         file = xfile
2821     }
2822     gshPA := gshCtx.gshPA
2823     savfd := gshPA.Files[ffix]
2824     gshPA.Files[ffix] = file.Fd()
2825     fmt.Printf("--I-- Opened %d %s\n",file.Fd(),argv[1])
2826     gshCtx.gshellv(argv[2:])
2827     gshPA.Files[ffix] = savfd
2828
2829     return false
2830 }
2831
2832 //fmt.Fprintf(res, "GShell Status: %q", html.EscapeString(req.URL.Path))
2833 func httpHandler(res http.ResponseWriter, req *http.Request){
2834     path := req.URL.Path
2835     fmt.Printf("--I-- Got HTTP Request(%s)\n",path)
2836     {
2837         gshCtxBuf, _ := setupGshContext()
2838         gshCtx := &gshCtxBuf
2839         fmt.Printf("--I-- %s\n",path[1:])
2840         gshCtx.tgshelll(path[1:])
2841     }
2842     fmt.Fprintf(res, "Hello(^_~)//\n%s\n",path)
2843 }
2844 func (gshCtx *GshContext) httpServer(argv []string){
2845     http.HandleFunc("/", httpHandler)
2846     accport := "localhost:9999"
2847     fmt.Printf("--I-- HTTP Server Start at [%s]\n",accport)
2848     http.ListenAndServe(accport,nil)
2849 }
2850 func (gshCtx *GshContext)xGo(argv[]string){
2851     go gshCtx.gshellv(argv[1:]);
2852 }

```

```

2853 func (gshCtx *GshContext) xPs(argv[]string)(){
2854 }
2855
2856 // <a name="plugin">Plugin</a>
2857 // plugin [-ls [names]] to list plugins
2858 // Reference: <a href="https://golang.org/src/plugin/">plugin</a> source code
2859 func (gshCtx *GshContext) whichPlugin(name string,argv[]string)(pi *PluginInfo){
2860     pi = nil
2861     for _p := range gshCtx.PluginFuncs {
2862         if p.Name == name && pi == nil {
2863             pi = &p
2864         }
2865         if !isin("-s",argv){
2866             //fmt.Printf("%v %v ",i,p)
2867             if !isin("-ls",argv){
2868                 showFileInfo(p.Path,argv)
2869             }else{
2870                 fmt.Printf("%s\n",p.Name)
2871             }
2872         }
2873     }
2874     return pi
2875 }
2876 func (gshCtx *GshContext) xPlugin(argv[]string) (error) {
2877     if len(argv) == 0 || argv[0] == "-ls" {
2878         gshCtx.whichPlugin("",argv)
2879         return nil
2880     }
2881     name := argv[0]
2882     Pin := gshCtx.whichPlugin(name,[]string{"-s"})
2883     if Pin != nil {
2884         os.Args = argv // should be recovered?
2885         Pin.Addr.(func())()
2886         return nil
2887     }
2888     sofile := toFullPath(argv[0] + ".so") // or find it by which($PATH)
2889
2890     p, err := plugin.Open(sofile)
2891     if err != nil {
2892         fmt.Printf("--E-- plugin.Open(%s)(%v)\n",sofile,err)
2893         return err
2894     }
2895     fname := "Main"
2896     f, err := p.Lookup(fname)
2897     if( err != nil ){
2898         fmt.Printf("--E-- plugin.Lookup(%s)(%v)\n",fname,err)
2899         return err
2900     }
2901     pin := PluginInfo {p,f,name,sofile}
2902     gshCtx.PluginFuncs = append(gshCtx.PluginFuncs,pin)
2903     fmt.Printf("--I-- added (%d)\n",len(gshCtx.PluginFuncs))
2904
2905     //fmt.Printf("--I-- first call(%s:%s)%v\n",sofile,fname,argv)
2906     os.Args = argv
2907     f.(func())()
2908     return err
2909 }
2910 func (gshCtx*GshContext)Args(argv[]string){
2911     for i,v := range os.Args {
2912         fmt.Printf("[%v] %v\n",i,v)
2913     }
2914 }
2915 func (gshCtx *GshContext) showVersion(argv[]string){
2916     if !isin("-l",argv) {
2917         fmt.Printf("%v/%v (%v)",NAME,VERSION,DATE);
2918     }else{
2919         fmt.Println(VERSION);
2920     }
2921     if !isin("-a",argv) {
2922         fmt.Println(AUTHOR)
2923     }
2924     if !isin("-n",argv) {
2925         fmt.Println("\n")
2926     }
2927 }
2928
2929 // <a name="scant">Scant</a> // string decomposer
2930 // scant [format] [input]
2931 func scant(sstr string)(strv[]string){
2932     strv = strings.Split(sstr," ")
2933     return strv
2934 }
2935 func scanUtil(src,end string)(rstr string,leng int){
2936     idx := strings.Index(src,end)
2937     if 0 <= idx {
2938         rstr = src[0:idx]
2939         return rstr,idx+lend(end)
2940     }
2941     return src,0
2942 }
2943
2944 // -bn -- display base-name part only // can be in some %fmt, for sed rewriting
2945 func (gsh*GshContext)printVal(fmts string, vstr string, optv[]string){
2946     //vint,err := strconv.Atoi(vstr)
2947     var ival int64 = 0
2948     n := 0
2949     err := error(nil)
2950     if strBegins(vstr,"_") {
2951         vx,_ := strconv.Atoi(vstr[1:])
2952         if vx < len(gsh.iValues) {
2953             vstr = gsh.iValues[vx]
2954         }else{
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("%s"+fmts,ival)
2966     }else{
2967         if !isin("-bn",optv){
2968             fmt.Printf("%s"+fmts,filepath.Base(vstr))
2969         }else{
2970             fmt.Printf("%s"+fmts,vstr)
2971         }
2972     }
2973 }
2974 }
2975 func (gsh*GshContext)printfv(fmts,div string,argv[]string,optv[]string,list[]string){
2976     //fmt.Printf("%d",len(list))

```

```

2977     //curfmt := "v"
2978     outlen := 0
2979     curfmt := gsh.iFormat
2980
2981     if 0 < len(fmts) {
2982         for xi := 0; xi < len(fmts); xi++ {
2983             fch := fmts[xi]
2984             if fch == '%' {
2985                 if xi+1 < len(fmts) {
2986                     curfmt = string(fmts[xi+1])
2987                 }
2988                 gsh.iFormat = curfmt
2989                 xi += 1
2990                 if xi+1 < len(fmts) && fmts[xi+1] == '(' {
2991                     vals,leng := scanUntil(fmts[xi+2:],")")
2992                     //fmt.Printf("--D-- show fmt(%v) val(%v) next(%v)\n",curfmt,vals,leng)
2993                     gsh.printVal(curfmt,vals,optv)
2994                     xi += 2+leng-1
2995                     outlen += 1
2996                 }
2997                 continue
2998             }
2999             if fch == '_' {
3000                 hi,leng := scanInt(fmts[xi+1:])
3001                 if 0 < leng {
3002                     if hi < len(gsh.iValues) {
3003                         gsh.printVal(curfmt,gsh.iValues[hi],optv)
3004                         outlen += 1 // should be the real length
3005                     }else{
3006                         fmt.Printf("(out-range)")
3007                     }
3008                     xi += leng
3009                     continue;
3010                 }
3011                 fmt.Printf("%c",fch)
3012                 outlen += 1
3013             }
3014         }
3015     }else{
3016         //fmt.Printf("--D-- print %s\n")
3017         for i,v := range list {
3018             if 0 < i {
3019                 fmt.Printf(div)
3020             }
3021             gsh.printVal(curfmt,v,optv)
3022             outlen += 1
3023         }
3024     }
3025     if 0 < outlen {
3026         fmt.Printf("\n")
3027     }
3028 }
3029 func (gsh*GshContext)Scany(argv[]string){
3030     //fmt.Printf("--D-- Scanv(%v)\n",argv)
3031     if len(argv) == 1 {
3032         return
3033     }
3034     argv = argv[1:]
3035     fmts := ""
3036     if strBegins(argv[0],"-F") {
3037         fmts = argv[0]
3038         gsh.iDelimiter = fmts
3039         argv = argv[1:]
3040     }
3041     input := strings.Join(argv," ")
3042     if fmts == "" { // simple decomposition
3043         v := scanv(input)
3044         gsh.iValues = v
3045         //fmt.Printf("%v\n",strings.Join(v,","))
3046     }else{
3047         v := make([]string,8)
3048         n,err := fmt.Sscanf(input,fmts,&v[0],&v[1],&v[2],&v[3])
3049         fmt.Printf("--D-- Scanf ->(%v) n=%d err=(%v)\n",v,n,err)
3050         gsh.iValues = v
3051     }
3052 }
3053 func (gsh*GshContext)Printv(argv[]string{
3054     if false { //@0U
3055         fmt.Printf("%v\n",strings.Join(argv[1:]," "))
3056     }
3057     //fmt.Printf("--D-- Printv(%v)\n",argv)
3058     //fmt.Printf("%v\n",strings.Join(gsh.iValues,","))
3059     div := gsh.iDelimiter
3060     fmts := ""
3061     argv = argv[1:]
3062     if 0 < len(argv) {
3063         if strBegins(argv[0],"-F") {
3064             div = argv[0][2:]
3065             argv = argv[1:]
3066         }
3067     }
3068 }
3069 optv := []string{}
3070 for _,v := range argv {
3071     if strBegins(v,"_"){
3072         optv = append(optv,v)
3073         argv = argv[1:]
3074     }else{
3075         break;
3076     }
3077 }
3078 }
3079 if 0 < len(argv) {
3080     fmts = strings.Join(argv," ")
3081 }
3082 gsh.printfv(fmts,div,argv,optv,gsh.iValues)
3083 }
3084 func (gsh*GshContext)Basename(argv[]string){
3085     for i,v := range gsh.iValues {
3086         gsh.iValues[i] = filepath.Base(v)
3087     }
3088 }
3089 func (gsh*GshContext)Sortv(argv[]string{
3090     sv := gsh.iValues
3091     sort.Slice(sv , func(i,j int) bool {
3092         return sv[i] < sv[j]
3093     })
3094 }
3095 func (gsh*GshContext)Shiftv(argv[]string){
3096     vi := len(gsh.iValues)
3097     if 0 < vi {
3098         if isin("r",argv) {
3099             top := gsh.iValues[0]
3100             gsh.iValues = append(gsh.iValues[1:],top)
3101         }
3102     }
3103 }
```

```

3101     }else{
3102         gsh.iValues = gsh.iValues[1:]
3103     }
3104 }
3105 }
3106
3107 func (gsh*GshContext)Enq(argv[]string){
3108 }
3109 func (gsh*GshContext)Deq(argv[]string){
3110 }
3111 func (gsh*GshContext)Push(argv[]string){
3112     gsh.iValStack = append(gsh.iValStack,argv[1:])
3113     fmt.Printf("depth=%d\n",len(gsh.iValStack))
3114 }
3115 func (gsh*GshContext)Dump(argv[]string){
3116     for i,v := range gsh.iValStack {
3117         fmt.Printf("%d %v\n",i,v)
3118     }
3119 }
3120 func (gsh*GshContext)Pop(argv[]string){
3121     depth := len(gsh.iValStack)
3122     if 0 < depth {
3123         v := gsh.iValStack[depth-1]
3124         if isn("cat",argv){
3125             gsh.iValues = append(gsh.iValues,v...)
3126         }else{
3127             gsh.iValues = v
3128         }
3129         gsh.iValStack = gsh.iValStack[0:depth-1]
3130         fmt.Printf("depth=%d %s\n",len(gsh.iValStack),gsh.iValues)
3131     }else{
3132         fmt.Printf("depth=%d\n",depth)
3133     }
3134 }
3135
3136 // <a name="interpreter">Command Interpreter</a>
3137 func (gshCtx*GshContext)gshellv(argv []string) (fin bool) {
3138     fin = false
3139
3140     if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr,"--I-- gshellv(%d)\n",len(argv)) }
3141     if len(argv) <= 0 {
3142         return false
3143     }
3144     argv := []string{}
3145     for ai := 0; ai < len(argv); ai++ {
3146         argv = append(argv,strsubst(gshCtx,argv[ai],false))
3147     }
3148     argv = xargv
3149     if false {
3150         for ai := 0; ai < len(argv); ai++ {
3151             fmt.Printf("%d %s [%d]\n",
3152                     ai,argv[ai],len(argv[ai]),argv[ai])
3153         }
3154     }
3155     cmd := argv[0]
3156     if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr,"--I-- gshellv(%d)%v\n",len(argv),argv) }
3157     switch { // https://tour.golang.org/flowcontrol/11
3158     case cmd == "":
3159         gshCtx.xPwd([]string{}); // empty command
3160     case cmd == "x":
3161         gshCtx.CmdTrace = ! gshCtx.CmdTrace
3162     case cmd == "xt":
3163         gshCtx.CmdTime = ! gshCtx.CmdTime
3164     case cmd == "ot":
3165         gshCtx.sconnect(true, argv)
3166     case cmd == "-ou":
3167         gshCtx.sconnect(false, argv)
3168     case cmd == "-it":
3169         gshCtx.saccept(true , argv)
3170     case cmd == "-in":
3171         gshCtx.saccept(false, argv)
3172     case cmd == "i" || cmd == "<" || cmd == "-o" || cmd == ">" || cmd == "-a" || cmd == ">>" || cmd == "-s" || cmd == "><":
3173         gshCtx.redirect(argv)
3174     case cmd == "|":
3175         gshCtx.frompipe(argv)
3176     case cmd == "args":
3177         gshCtx.Args(argv)
3178     case cmd == "bg" || cmd == "-bg":
3179         rfin := gshctx.inBackground(argv[1:])
3180         return rfin
3181     case cmd == "-bn":
3182         gshCtx.Basename(argv)
3183     case cmd == "call":
3184         _r_ = gshctx.excommand(false,argv[1:])
3185     case cmd == "cd" || cmd == "chdir":
3186         gshCtx.xChdir(argv);
3187     case cmd == "-cksum":
3188         gshCtx.xFind(argv)
3189     case cmd == "sum":
3190         gshCtx.xFind(argv)
3191     case cmd == "-sumtest":
3192         str := ""
3193         if 1 < len(argv) { str = argv[1] }
3194         crc := strCRC32(str,uint64(len(str)))
3195         fprintf(stderr,"%v %v\n",crc,len(str))
3196     case cmd == "close":
3197         gshCtx.xClose(argv)
3198     case cmd == "gcp":
3199         gshCtx.FileCopy(argv)
3200     case cmd == "dec" || cmd == "decode":
3201         gshCtx.Dec(argv)
3202     case cmd == "#define":
3203     case cmd == "dic" || cmd == "d":
3204         xbic(argv)
3205     case cmd == "dump":
3206         gshCtx.Dump(argv)
3207     case cmd == "echo" || cmd == "e":
3208         echo(argv,true)
3209     case cmd == "enc" || cmd == "encode":
3210         gshCtx.Enc(argv)
3211     case cmd == "env":
3212         env(argv)
3213     case cmd == "eval":
3214         xEval(argv[1:],true)
3215     case cmd == "ev" || cmd == "events":
3216         dumpEvents(argv)
3217     case cmd == "exec":
3218         _ = gshctx.excommand(true,argv[1:])
3219         /* should not return here
3220     case cmd == "exit" || cmd == "quit":
3221         // write Result code EXIT to 3>
3222         return true
3223     case cmd == "fds":
3224         // dump the attributes of fds (of other process)

```

```

3225 case cmd == "-find" || cmd == "fin" || cmd == "ufind" || cmd == "uf":
3226     gshCtx.xFind(argv[1:])
3227 case cmd == "fu":
3228     gshCtx.xFind(argv[1:])
3229 case cmd == "fork":
3230     // mainly for a server
3231 case cmd == "gen":
3232     gshCtx.gen(argv)
3233 case cmd == "go":
3234     gshCtx.xGo(argv)
3235 case cmd == "grep":
3236     gshCtx.xFind(argv)
3237 case cmd == "ddeg":
3238     gshCtx.Deg(argv)
3239 case cmd == "geng":
3240     gshCtx.Eng(argv)
3241 case cmd == "gpop":
3242     gshCtx.Pop(argv)
3243 case cmd == "gpush":
3244     gshCtx.Push(argv)
3245 case cmd == "history" || cmd == "hi": // hi should be alias
3246     gshCtx.xHistory(argv)
3247 case cmd == "jobs":
3248     gshCtx.xJobs(argv)
3249 case cmd == "lisp" || cmd == "nlsp":
3250     gshCtx.SplitLine(argv)
3251 case cmd == "ls":
3252     gshCtx.xFind(argv)
3253 case cmd == "nop":
3254     // do nothing
3255 case cmd == "pipe":
3256     gshCtx.xOpen(argv)
3257 case cmd == "plug" || cmd == "plugin" || cmd == "pin":
3258     gshCtx.xPlugin(argv[1:])
3259 case cmd == "print" || cmd == "pr":
3260     // output internal slice // also sprintf should be
3261     gshCtx.Println(argv)
3262 case cmd == "ps":
3263     gshCtx.xPs(argv)
3264 case cmd == "pstitle":
3265     // to be gsh.title
3266 case cmd == "rexecd" || cmd == "rexd":
3267     gshCtx.RexecServer(argv)
3268 case cmd == "rexec" || cmd == "rex":
3269     gshCtx.RexecClient(argv)
3270 case cmd == "repeat" || cmd == "rep": // repeat cond command
3271     gshCtx.repeat(argv)
3272 case cmd == "replay":
3273     gshCtx.xReplay(argv)
3274 case cmd == "scan":
3275     // scan input (or so in fscanf) to internal slice (like Files or map)
3276     gshCtx.Scany(argv)
3277 case cmd == "set":
3278     // set name ...
3279 case cmd == "serv":
3280     gshCtx.httpServer(argv)
3281 case cmd == "shift":
3282     gshCtx.Shift(argv)
3283 case cmd == "sleep":
3284     gshCtx.sleep(argv)
3285 case cmd == "sort":
3286     gshCtx.Sort(argv)
3287
3288 case cmd == "j" || cmd == "join":
3289     gshCtx.Rjoin(argv)
3290 case cmd == "a" || cmd == "alpa":
3291     gshCtx.Rexec(argv)
3292 case cmd == "jcd" || cmd == "jchdir":
3293     gshCtx.Rchdir(argv)
3294 case cmd == "jget":
3295     gshCtx.Rget(argv)
3296 case cmd == "jls":
3297     gshCtx.Rls(argv)
3298 case cmd == "jput":
3299     gshCtx.Rput(argv)
3300 case cmd == "jpwd":
3301     gshCtx.Rpwd(argv)
3302
3303 case cmd == "time":
3304     fin = gshCtx.XTime(argv)
3305 case cmd == "ungets":
3306     if l < len(argv) {
3307         ungets(argv[1]+\n)
3308     }else{
3309     }
3310 case cmd == "pwd":
3311     gshCtx.xPwd(argv);
3312 case cmd == "ver" || cmd == "-ver" || cmd == "version":
3313     gshCtx.showVersion(argv)
3314 case cmd == "where":
3315     // data file or so
3316 case cmd == "which":
3317     which("PATH",argv);
3318 default:
3319     if gshCtx.whichPlugin(cmd,[]string{"-s"}) != nil {
3320         gshCtx.xPlugin(argv)
3321     }else{
3322         notfound,_ := gshCtx.excommand(false,argv)
3323         if notfound {
3324             fmt.Printf("--E-- command not found (%v)\n",cmd)
3325         }
3326     }
3327 }
3328 }
3329 return fin
3330
3331 func (gsh*GshContext)gshelll(gline string) (rfin bool) {
3332     argv := strings.Split(string(gline), " ")
3333     fin := gsh.gshellv(argv)
3334     return fin
3335 }
3336 func (gsh*GshContext)tgshelll(gline string)(xfin bool){
3337     start := time.Now()
3338     fin := gsh.gshelll(gline)
3339     end := time.Now()
3340     elps := end.Sub(start);
3341     if gsh.CmdTime {
3342         fmt.Printf("--T-- " + time.Now().Format(time.Stamp) + "(%d.%09ds)\n",
3343             elps/1000000000,elps%1000000000)
3344     }
3345 }
3346 }
3347 func Ttyid() (int) {
3348     fi, err := os.Stdin.Stat()

```

```

3349     if err != nil {
3350         return 0;
3351     }
3352     //fmt.Printf("Stdin: %v Dev=%d\n",
3353     // fi.Mode(), fi.Mode()&os.ModeDevice)
3354     if (fi.Mode() & os.ModeDevice) != 0 {
3355         stat := syscall.Stat_t{};
3356         err := syscall.Fstat(0,&stat);
3357         if err != nil {
3358             //fmt.Printf("--I-- Stdin: (%v)\n",err)
3359         }else{
3360             //fmt.Printf("--I-- Stdin: rdev=%d %d\n",
3361             // stat.Rdev&0xFF,stat.Rdev);
3362             //fmt.Printf("--I-- Stdin: tty%d\n",stat.Rdev&0xFF);
3363             return int(stat.Rdev & 0xFF)
3364         }
3365     }
3366     return 0
3367 }
3368 func (gshCtx *GshContext) ttyfile() string {
3369     //fmt.Printf("--I-- GSH_HOME=%s\n",gshCtx.GshHomeDir)
3370     ttyfile := gshCtx.GshHomeDir + "/" + "gsh-tty" +
3371     fmt.Sprintf("%02d",gshCtx.TerminalId)
3372     //strconv.Itoa(gshCtx.TerminalId)
3373     //fmt.Printf("--I-- ttyfile=%s\n",ttyfile)
3374     return ttyfile
3375 }
3376 func (gshCtx *GshContext) ttystream(*os.File){
3377     file, err := os.OpenFile(gshCtx.ttyfile(),os.O_RDWR|os.O_CREATE|os.O_TRUNC,0600)
3378     if err != nil {
3379         fmt.Printf("--F-- cannot open %s (%s)\n",gshCtx.ttyfile(),err)
3380         return file;
3381     }
3382     return file
3383 }
3384 func (gshCtx *GshContext)getline(hix int, skipping bool, prevline string) (string) {
3385     if( skipping ){
3386         reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
3387         line,_ := reader.ReadLine()
3388         return string(line)
3389     }else
3390     if true {
3391         return xgetline(hix,prevline,gshCtx)
3392     }/*
3393     else
3394     if( with_exgetline && gshCtx.GetLine != "" ){
3395         //var xhix int64 = int64(hix); // cast
3396         newenv := os.Environ()
3397         newenv = append(newenv, "GSH_LINENO="+strconv.FormatInt(int64(hix),10) )
3398
3399         tty := gshCtx.ttyline()
3400         tty.WriteString(prevline)
3401         Pa := os.ProcAttr {
3402             "", // start dir
3403             newenv, //os.Environ(),
3404             []os.File{os.Stdin,os.Stdout,os.Stderr,tty},
3405             nil,
3406         }
3407     }
3408     //fmt.Printf("--I-- getline=%s // %s\n",gsh_getline[0],gshCtx.GetLine)
3409     proc, err := os.StartProcess(gsh_getline[0],[]string{"getline","getline"},&Pa)
3410     if err != nil {
3411         fmt.Printf("--F-- getline process error (%v)\n",err)
3412         // for ; ; {
3413         return "exit (getline program failed)"
3414     }
3415     //stat, err := proc.Wait()
3416     proc.Wait()
3417     buff := make([]byte,LINESIZE)
3418     count, err := tty.Read(buff)
3419     //, err := tty.Read(buff)
3420     //fmt.Printf("--D-- getline (%d)\n",count)
3421     if err != nil {
3422         if ! (count == 0) { // && err.String() == "EOF" } {
3423             fmt.Printf("--E-- getline error (%s)\n",err)
3424         }
3425     }else{
3426         //fmt.Printf("--I-- getline OK \"%s\"\n",buff)
3427     }
3428     tty.Close()
3429     gline := string(buff[0:count])
3430     return gline
3431 }
3432 */
3433 {
3434     // if isatty {
3435     fmt.Printf("I%d",hix)
3436     fmt.Println(PROMPT)
3437     //}
3438     reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
3439     line,_ := reader.ReadLine()
3440     return string(line)
3441 }
3442 }
3443 /**
3444 //== begin ===== getline ===== getline
3445 */
3446 /* getline.c
3447 * 2020-0819 extracted from dog.c
3448 * getline.go
3449 * 2020-0822 ported to Go
3450 */
3451 /*
3452 package main // getline main
3453 import (
3454     "fmt"      // <a href="https://golang.org/pkg/fmt/">fmt</a>
3455     "strings"   // <a href="https://golang.org/pkg/strings/">strings</a>
3456     "os"        // <a href="https://golang.org/pkg/os/">os</a>
3457     "syscall"   // <a href="https://golang.org/pkg/syscall/">syscall</a>
3458     //"bytes"    // <a href="https://golang.org/pkg/os/">os</a>
3459     //"os/exec"  // <a href="https://golang.org/pkg/os/">os</a>
3460 )
3461 */
3462
3463 // C language compatibility functions
3464 var errno = 0
3465 var stdin *os.File = os.Stdin
3466 var stdout *os.File = os.Stdout
3467 var stderr *os.File = os.Stderr
3468 var EOF = -1
3469 var NULL = 0
3470 type FILE os.File
3471 type StrBuff []byte
3472 var NULL_FP *os.File = nil

```

```

3473 var NULLSP = 0
3474 //var LINESIZE = 1024
3475
3476 func system(cmdstr string)(int){
3477     PA := syscall.ProcAttr {
3478         "", // the starting directory
3479         os.Environ(),
3480         []uintptr{os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd()},
3481         nil,
3482     }
3483     argv := strings.Split(cmdstr," ")
3484     pid,err := syscall.ForkExec(argv[0],argv,&PA)
3485     if( err != nil ){
3486         fmt.Printf("--E-- syscall(%v) err(%v)\n",cmdstr,err)
3487     }
3488     syscall.Wait4(pid,nil,0,nil)
3489
3490 /*
3491 argv := strings.Split(cmdstr," ")
3492 fmt.Fprintf(os.Stderr, "--I-- system(%v)\n",argv)
3493 //cmd := exec.Command(argv[0]...)
3494 cmd := exec.Command(argv[0],argv[1],argv[2])
3495 cmd.Stdin = strings.NewReader("output of system")
3496 var out bytes.Buffer
3497 cmd.Stdout = &out
3498 var serr bytes.Buffer
3499 cmd.Stderr = &serr
3500 err := cmd.Run()
3501 if err != nil {
3502     fmt.Fprintf(os.Stderr,"--E-- system(%v)err(%v)\n",argv,err)
3503     fmt.Println("ERR:%s\n",serr.String())
3504 }else{
3505     fmt.Printf("%s",out.String())
3506 }
3507 */
3508 return 0
3509 }
3510 func atoi(str string)(ret int){
3511     ret,err := fmt.Sscanf(str,"%d",ret)
3512     if err == nil {
3513         return ret
3514     }else{
3515         // should set errno
3516         return 0
3517     }
3518 }
3519 func getenv(name string)(string){
3520     val,got := os.LookupEnv(name)
3521     if got {
3522         return val
3523     }else{
3524         return "?"
3525     }
3526 }
3527 func strcpy(dst StrBuff, src string){
3528     var i int
3529     srcb := []byte(src)
3530     for i = 0; i < len(src) && srcb[i] != 0; i++ {
3531         dst[i] = srcb[i]
3532     }
3533     dst[i] = 0
3534 }
3535 func xstrcpy(dst StrBuff, src StrBuff){
3536     dst = src
3537 }
3538 func strcat(dst StrBuff, src StrBuff){
3539     dst = append(dst,src...)
3540 }
3541 func strdup(str StrBuff)(string){
3542     return string(str[0:strlen(str)])
3543 }
3544 func sstrlen(str string)(int){
3545     return len(str)
3546 }
3547 func strlen(str StrBuff)(int){
3548     var i int
3549     for i = 0; i < len(str) && str[i] != 0; i++ {
3550     }
3551     return i
3552 }
3553 func sizeof(data StrBuff)(int){
3554     return len(data)
3555 }
3556 func isatty(fd int)(ret int){
3557     return 1
3558 }
3559
3560 func fopen(file string,mode string)(fp*os.File){
3561     if mode == "r" {
3562         fp,err := os.Open(file)
3563         if( err != nil ){
3564             fmt.Printf("--E-- fopen(%s,%s)=(%v)\n",file,mode,err)
3565             return NULL_FPP;
3566         }
3567         return fp;
3568     }else{
3569         fp,err := os.OpenFile(file,os.O_RDWR|os.O_CREATE|os.O_TRUNC,0600)
3570         if( err != nil ){
3571             return NULL_FPP;
3572         }
3573         return fp;
3574     }
3575 }
3576 func fclose(fp*os.File){
3577     fp.Close()
3578 }
3579 func fflush(fp *os.File)(int){
3580     return 0
3581 }
3582 func fgetc(fp*os.File)(int){
3583     var buf [1]byte
3584     _,err := fp.Read(buf[0:1])
3585     if( err != nil ){
3586         return EOF;
3587     }else{
3588         return int(buf[0])
3589     }
3590 }
3591 func sfgets(str*string, size int, fp*os.File)(int){
3592     buf := make(StrBuff,size)
3593     var ch int
3594     var i int
3595     for i = 0; i < len(buf)-1; i++ {
3596         ch = fgetc(fp)
3597     }
3598     buf[i] = 0
3599     str = string(buf)
3600 }

```

```

3597     //fprintf(stderr,"--fgets %d/%d %X\n",i,len(buf),ch)
3598     if( ch == EOF ){
3599         break;
3600     }
3601     buf[i] = byte(ch);
3602     if( ch == '\n' ){
3603         break;
3604     }
3605 }
3606 buf[i] = 0
3607 //fprintf(stderr,"--fgets %d/%d (%s)\n",i,len(buf),buf[0:i])
3608 return i
3609 }
3610 func fgets(buf StrBuff, size int, fp*os.File)(int){
3611     var ch int
3612     var i int
3613     for i = 0; i < len(buf)-1; i++ {
3614         ch = fgetc(fp)
3615         //fprintf(stderr,"--fgets %d/%d %X\n",i,len(buf),ch)
3616         if( ch == EOF ){
3617             break;
3618         }
3619         buf[i] = byte(ch);
3620         if( ch == '\n' ){
3621             break;
3622         }
3623     }
3624     buf[i] = 0
3625 //fprintf(stderr,"--fgets %d/%d (%s)\n",i,len(buf),buf[0:i])
3626 return i
3627 }
3628 func fputc(ch int , fp*os.File)(int){
3629     var buf [1]byte
3630     buf[0] = byte(ch)
3631     fp.Write(buf[0:1])
3632     return 0
3633 }
3634 func fputs(buf StrBuff, fp*os.File)(int){
3635     fp.Write(buf)
3636     return 0
3637 }
3638 func xputss(str string, fp*os.File)(int){
3639     return fputs([]byte(str),fp)
3640 }
3641 func sscanf(str StrBuff,fmts string, params ...interface{})(int){
3642     fmt.Sscanf(string(str[0:strlen(str)]),fmts,params...)
3643     return 0
3644 }
3645 func fprintf(fp*os.File,fmts string, params ...interface{})(int){
3646     fmt.Fprintf(fp,fmts,params...)
3647     return 0
3648 }
3649
3650 //-- <a name="IME">Command Line IME</a>
3651 //----- MyIME
3652 var MyIMEVER = "MyIME/0.0.2";
3653 type RomKana struct {
3654     dic string // dictionary ID
3655     pat string // input pattern
3656     out string // output pattern
3657     hit int64 // count of hit and used
3658 }
3659 var dicens = 0
3660 var romkana [1024]RomKana
3661 var Romkan []RomKana
3662
3663 func isinDic(str string)(int){
3664     for i,v := range Romkan {
3665         if v.pat == str {
3666             return i
3667         }
3668     }
3669     return -1
3670 }
3671 const (
3672     DIC_COM_LOAD = "im"
3673     DIC_COM_DUMP = "s"
3674     DIC_COM_LIST = "ls"
3675     DIC_COM_ENA = "en"
3676     DIC_COM_DIS = "di"
3677 )
3678 func helpDic(argv []string{
3679     out := stderr
3680     cmd := ""
3681     if 0 < len(argv) { cmd = argv[0] }
3682     fprintf(out,"-- %v Usage\n",cmd)
3683     fprintf(out,"... Commands\n")
3684     fprintf(out,"... %v %-3v [dicName] [dicURL] -- Import dictionary\n",cmd,DIC_COM_LOAD)
3685     fprintf(out,"... %v %-3v [pattern] -- Search in dictionary\n",cmd,DIC_COM_DUMP)
3686     fprintf(out,"... %v %-3v [dicName] -- List dictionaries\n",cmd,DIC_COM_LIST)
3687     fprintf(out,"... %v %-3v [dicName] -- Disable dictionaries\n",cmd,DIC_COM_DIS)
3688     fprintf(out,"... %v %-3v [dicName] -- Enable dictionaries\n",cmd,DIC_COM_ENA)
3689     fprintf(out,"... Keys ... %v\n", "ESC can be used for '\\\'')
3690     fprintf(out,"... \\c -- Reverse the case of the last character\n")
3691     fprintf(out,"... \\i -- Replace input with translated text\n")
3692     fprintf(out,"... \\j -- On/Off translation mode\n")
3693     fprintf(out,"... \\l -- Force Lower Case\n")
3694     fprintf(out,"... \\u -- Force Upper Case (software CapsLock)\n")
3695     fprintf(out,"... \\v -- Show translation actions\n")
3696     fprintf(out,"... \\x -- Replace the last input character with it Hexa-Decimal\n")
3697 }
3698 func xDic(argv[]string{
3699     if len(argv) <= 1 {
3700         helpdic(argv)
3701         return
3702     }
3703     argv = argv[1:]
3704     var debug = false
3705     var info = false
3706     var silent = false
3707     var dump = false
3708     var builtin = false
3709     cmd := argv[0]
3710     argv = argv[1:]
3711     opt := ""
3712     arg := ""
3713
3714     if 0 < len(argv) {
3715         arg1 := argv[0]
3716         if arg1[0] == '-' {
3717             switch arg1 {
3718                 default: fmt.Printf("--Ed-- Unknown option(%v)\n",arg1)
3719                 return
3720                 case "-b": builtin = true

```

```

3721         case "-d": debug = true
3722         case "-s": silent = true
3723         case "-v": info = true
3724     }
3725     opt = arg1
3726     argv = argv[1:]
3727 }
3728 }
3729 dicName := ""
3730 dicURL := ""
3731 if 0 < len(argv) {
3732     arg = argv[0]
3733     dicName = arg
3734     argv = argv[1:]
3735 }
3736 if 0 < len(argv) {
3737     dicURL = argv[0]
3738     argv = argv[1:]
3739 }
3740 if false {
3741     fprintf(stderr,"--Dd-- com(%v) opt(%v) arg(%v)\n",cmd,opt,arg)
3742 }
3743 if cmd == DIC_COM_LOAD {
3744     //dicType := ""
3745     dicBody := ""
3746     if !builtin && dicName != "" && dicURL == "" {
3747         f,err := os.Open(dicName)
3748         if err == nil {
3749             dicURL = dicName
3750         }else{
3751             f,err = os.Open(dicName+".html")
3752             if err == nil {
3753                 dicURL = dicName+".html"
3754             }else{
3755                 f,err = os.Open("gshdic-"+dicName+".html")
3756                 if err == nil {
3757                     dicURL = "gshdic-"+dicName+".html"
3758                 }
3759             }
3760         }
3761     if err == nil {
3762         var buf = make([]byte,128*1024)
3763         count,err := f.Read(buf)
3764         f.Close()
3765         if info {
3766             fprintf(stderr,"--Id-- ReadDic(%v,%v)\n",count,err)
3767         }
3768         dicBody = string(buf[0:count])
3769     }
3770 }
3771 if dicBody == "" {
3772     switch arg {
3773         default:
3774             dicName = "WorldDic"
3775             dicURL = WorldDic
3776             if info {
3777                 fprintf(stderr,"--Id-- default dictionary \'%v\'\n",
3778                         dicName);
3779         }
3780         case "wnn":
3781             dicName = "WnnDic"
3782             dicURL = WnnDic
3783             case "sumomo":
3784                 dicName = "SumomoDic"
3785                 dicURL = SumomoDic
3786             case "sijimi":
3787                 dicName = "SijimiDic"
3788                 dicURL = SijimiDic
3789             case "jkl":
3790                 dicName = "JKLJadic"
3791                 dicURL = JA_JKLDic
3792             }
3793         if debug {
3794             fprintf(stderr,"--Id-- %v URL=%v\n",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     }
3804     body,_ := base64.StdEncoding.DecodeString(dicv[1])
3805     dicBody = string(body)
3806 }
3807 if info {
3808     fmt.Printf("--Id-- %v %v\n",dicName,dicURL)
3809     fmt.Println("%s",dicBody)
3810 }
3811 if debug {
3812     fprintf(stderr,"--Id-- dicName %v text...\n",dicName)
3813     fprintf(stderr,"%v\n",string(dicBody))
3814 }
3815 env := strings.Split(dicBody,"\n");
3816 if info {
3817     fprintf(stderr,"--Id-- %v scan...\n",dicName);
3818 }
3819 var added int = 0
3820 var dup int = 0
3821 for i,v := range env {
3822     var pat string
3823     var out string
3824     fmt.Sscanf(v,"%s %s",&pat,&out)
3825     if len(pat) <= 0 {
3826     }else{
3827         if 0 <= isinDic(pat) {
3828             dup += 1
3829             continue
3830         }
3831         romkana[dicents] = RomKana{dicName,pat,out,0}
3832         dicents += 1
3833         added += 1
3834         RomKan = append(Romkan,RomKana{dicName,pat,out,0})
3835         if debug {
3836             fmt.Printf("[%3v]:[%2v]%-8v [%2v]%v\n",
3837                         i,len(pat),pat,len(out),out)
3838         }
3839     }
3840 }
3841 if !silent {
3842     url := dicURL
3843     if strBegins(url,"data:") {
3844         url = "builtin"

```

```

3845         }
3846         fprintf(stderr,"--Id-- %v scan... %v added, %v dup. / %v total (%v)\n",
3847                 dicName,added,dup,len(Romkan),url);
3848     }
3849     // should sort by pattern length for conclete match, for performance
3850     if debug {
3851         arg = "" // search pattern
3852         dump = true
3853     }
3854 }
3855 if cmd == DIC_COM_DUMP || dump {
3856     fprintf(stderr,"--Id-- %v dump... %v entries:\n",dicName,len(Romkan));
3857     var match = 0
3858     for i := 0; i < len(Romkan); i++ {
3859         dic := Romkan[i].dic
3860         pat := Romkan[i].pat
3861         out := Romkan[i].out
3862         if arg == "" || 0 < strings.Index(pat,arg) || 0 < strings.Index(out,arg) {
3863             fmt.Printf("\\\\%v\\t%v [%2v]-%8v [%2v]@v\n",
3864                     i,dic,len(pat),pat,len(out),out)
3865             match += 1
3866         }
3867     }
3868     fprintf(stderr,"--Id-- %v matched %v / %v entries:\n",arg,match,len(Romkan));
3869 }
3870 }
3871 func loadDefaultDic(dic int){
3872     if( 0 < len(Romkan) ){
3873         return
3874     }
3875     //fprintf(stderr,"r\n")
3876     xDic([]string{"dic",DIC_COM_LOAD});
3877
3878     var info = false
3879     if info {
3880         fprintf(stderr,"--Id-- Conguratuations!! WorldDic is now activated.\r\n")
3881         fprintf(stderr,"--Id-- enter \"dic\" command for help.\r\n")
3882     }
3883 }
3884 func readDic()(int){
3885     /*
3886     var rk *os.File;
3887     var dic = "MyIMB-dic.txt";
3888     //rk = fopen("romkana.txt","r");
3889     //rk = fopen("JK-JA-morse-dic.txt","r");
3890     rk = fopen(dic,"r");
3891     if( rk == NULL_fp ){
3892         if( true ){
3893             fprintf(stderr,"--%s-- Could not load %s\n",MyIMEVER,dic);
3894         }
3895         return -1;
3896     }
3897     if( true ){
3898         var di int;
3899         var line = make(StrBuff,1024);
3900         var pat string
3901         var out string
3902         for di = 0; di < 1024; di++ {
3903             if( fgets(line,sizeof(line),rk) == NULLSP ){
3904                 break;
3905             }
3906             fmt.Sscanf(string(line[0:strlen(line)]),"s s",&pat,&out);
3907             //sscanf(line,"%s %^r\n",&pat,&out);
3908             romkana[di].pat = pat;
3909             romkana[di].out = out;
3910             //fprintf(stderr,"--Dd- %-10s %s\n",pat,out)
3911         }
3912         dicents += di
3913         if( false ){
3914             fprintf(stderr,"--%s-- loaded romkana.txt [%d]\n",MyIMEVER,di);
3915             for di = 0; di < dicents; di++ {
3916                 fprintf(stderr,
3917                         "%s %s\n",romkana[di].pat,romkana[di].out);
3918             }
3919         }
3920     }
3921     fclose(rk);
3922
3923     //romkana[dicents].pat = "//ddump"
3924     //romkana[dicents].pat = "//ddump" // dump the dic. and clean the command input
3925 */
3926     return 0;
3927 }
3928 func matchlen(stri string, pati string)(int){
3929     if strBegins(stri,pati) {
3930         return len(pati)
3931     }else{
3932         return 0
3933     }
3934 }
3935 func convs(src string)(string){
3936     var si int;
3937     var sx = len(src);
3938     var di int;
3939     var mi int;
3940     var dstb []byte
3941
3942     for si = 0; si < sx; { // search max. match from the position
3943         if strBegins(src[si:],"/tx/") {
3944             // %x/integer//s/a/b/
3945             ix := strings.Index(src[si+3:],"/")
3946             if 0 < ix {
3947                 var iv int = 0
3948                 //fmt.Sscanf(src[si+3:si+3+ix],"d",&iv)
3949                 fmt.Sscanf(src[si+3:si+3+ix],"v",&iv)
3950                 sval := fmt.Sprintf("%x",iv)
3951                 bval := []byte(sval)
3952                 dstb = append(dstb,bval...)
3953                 si = si+3+ix+1
3954                 continue
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                 fmt.Sprintf("%d",iv)
3963                 bval := []byte(sval)
3964                 dstb = append(dstb,bval...)
3965                 si = si+3+ix+1
3966                 continue
3967             }
3968     }

```

```

3969     }
3970     if strBegins(src[si:], "%t") {
3971         now := time.Now()
3972         if true {
3973             date := now.Format(time.Stamp)
3974             dstb = append(dstb, []byte(date)...)
3975             si = si+3
3976         }
3977         continue
3978     }
3979     var maxlen int = 0;
3980     var len int;
3981     mi = -1;
3982     for di = 0; di < dicents; di++ {
3983         len = matchlen(src[si:], romkana[di].pat);
3984         if( maxlen < len ) {
3985             maxlen = len;
3986             mi = di;
3987         }
3988     }
3989     if( 0 < maxlen ){
3990         out := romkana[mi].out;
3991         dstb = append(dstb, []byte(out)...);
3992         si += maxlen;
3993     }else{
3994         dstb = append(dstb, src[si])
3995         si += 1;
3996     }
3997 }
3998 return string(dstb)
3999 }
4000 func trans(src string)(int){
4001     dst := convs(src);
4002     xputss(dst,stderr);
4003     return 0;
4004 }
4005
4006 //----- LINEEDIT
4007 // "?" at the top of the line means searching history
4008
4009 // should be compatilbe with Telnet
4010 const (
4011     EV_MODE      = 255
4012     EV_IDLE      = 254
4013     EV_TIMEOUT   = 253
4014
4015     GO_UP        = 252    // k
4016     GO_DOWN      = 251    // j
4017     GO_RIGHT     = 250    // l
4018     GO_LEFT      = 249    // h
4019     DEL_RIGHT    = 248    // x
4020     GO_TOPL      = 'A'-'0x40 // 0
4021     GO_ENDL      = 'E'-'0x40 // $
4022
4023     GO_TOPW      = 239    // b
4024     GO_ENDW      = 238    // e
4025     GO_NEXTW     = 237    // w
4026
4027     GO_FORWCH    = 229    // f
4028     GO_PAIRCH    = 228    // %
4029
4030     GO_DEL       = 219    // d
4031
4032     HI_SRCH_FW   = 209    // /
4033     HI_SRCH_BK   = 208    // ?
4034     HI_SRCH_RFW  = 207    // n
4035     HI_SRCH_RBK  = 206    // N
4036 )
4037
4038 // should return number of octets ready to be read immediately
4039 //fprintf(stderr,"%n--Select(%v %v)\n",err,r.Bits[0])
4040
4041
4042 var EventRecvFd = -1 // file descriptor
4043 var EventSendFd = -1
4044 const EventFdOffset = 1000000
4045 const NormalFdOffset = 100
4046
4047 func putEvent(event int, evarg int){
4048     if true {
4049         if EventRecvFd < 0 {
4050             var pv = []int{-1,-1}
4051             syscall.Pipe(pv)
4052             EventRecvFd = pv[0]
4053             EventSendFd = pv[1]
4054             //fmt.Printf("--De-- EventPipe created[%v,%v]\n",EventRecvFd,EventSendFd)
4055         }
4056     }else{
4057         if EventRecvFd < 0 {
4058             // the document differs from this spec
4059             // https://golang.org/src/syscall/syscall_unix.go?s=8096:8158#L340
4060             sv,err := syscall.Socketpair(syscall.AF_UNIX,syscall.SOCK_STREAM,0)
4061             EventRecvFd = sv[0]
4062             EventSendFd = sv[1]
4063             if err != nil {
4064                 fmt.Printf("--De-- EventSock created[%v,%v](%v)\n",
4065                         EventRecvFd,EventSendFd,err)
4066             }
4067         }
4068     }
4069     var buf = []byte{ byte(event) }
4070     n,err := syscall.Write(EventSendFd,buf)
4071     if err != nil {
4072         fmt.Printf("--De-- putEvent[%v](%v)(%v)\n",EventSendFd,event,n,err)
4073     }
4074 }
4075 func ungets(str string){
4076     for _,ch := range str {
4077         putEvent(int(ch),0)
4078     }
4079 }
4080 func (gsh*GshContext)xReplay(argv[]string){
4081     hix := 0
4082     tempo := 1.0
4083     xtempo := 1.0
4084     repeat := 1
4085
4086     for _,a := range argv { // tempo
4087         if strBegins(a,"x") {
4088             fmt.Sscanf(a[1],"%f",&xtempo)
4089             tempo = 1 / xtempo
4090             //fprintf(stderr,"--Dr-- tempo=[%v]\n",a[2:],tempo);
4091         }else
4092             if strBegins(a,"r") { // repeat

```

```

4093     fmt.Sscanf(a[1],"%v",&repeat)
4094   }else
4095     if strBegins(a,"!") {
4096       fmt.Sscanf(a[1],"%d",&hix)
4097     }else{
4098       fmt.Sscanf(a,"%d",&hix)
4099     }
4100   }
4101   if hix == 0 || len(argv) <= 1 {
4102     hix = len(gsh.CommandHistory)-1
4103   }
4104   fmt.Printf("--Ir-- Replay(!v x%v r%v)\n",hix,xtempo,repeat)
4105   //dumpEvents(hix)
4106   //gsh.xScanReplay(hix,false,repeat,tempo,argv)
4107   go gsh.xScanReplay(hix,true,repeat,tempo,argv)
4108 }
4109
4110 // <a href="https://golang.org/pkg/syscall/#FdSet">syscall.Select</a>
4111 // 2020-0827 GShell-0.2.3
4112 /*
4113 func FpollInl(fp *os.File,usec int)(uintptr){
4114   nfd := 1
4115
4116   rdv := syscall.FdSet {}
4117   fd1 := fp.Fd()
4118   bank1 := fd1/32
4119   mask1 := int32(1 << fd1)
4120   rdv.Bits[bank1] = mask1
4121
4122   fd2 := -1
4123   bank2 := -1
4124   var mask2 int32 = 0
4125
4126   if 0 <= EventRecvFd {
4127     fd2 = EventRecvFd
4128     nfd = fd2 + 1
4129     bank2 = fd2/32
4130     mask2 = int32(1 << fd2)
4131     rdv.Bits[bank2] |= mask2
4132     //fmt.Printf("--De-- EventPoll mask added [%d][%v][%v]\n",fd2,bank2,mask2)
4133   }
4134
4135   tout := syscall.NsecToTimeval(int64(usec*1000))
4136   //n,err := syscall.Select(nfd,&rdv,nil,nil,&tout) // spec. mismatch
4137   err := syscall.Select(nfd,&rdv,nil,nil,&tout)
4138   if err != nil {
4139     //fmt.Printf("--De-- select() err(%v)\n",err)
4140   }
4141   if err == nil {
4142     if 0 <= fd2 && (rdv.Bits[bank2] & mask2) != 0 {
4143       if false {
4144         fmt.Printf("--De-- got Event\n")
4145       }
4146       return uintptr(EventFdOffset + fd2)
4147     }else
4148       if (rdv.Bits[bank1] & mask1) != 0 {
4149         return uintptr(NormalFdOffset + fd1)
4150       }else{
4151         return 1
4152       }
4153     }else{
4154       return 0
4155     }
4156   }
4157 */
4158 func fgetcTimeout1(fp *os.File,usec int)(int){
4159   READ1:
4160   //readyFd := FpollInl(fp,usec)
4161   readyFd := CfpollInl(fp,usec)
4162   if readyFd < 100 {
4163     return EV_TIMEOUT
4164   }
4165
4166   var buf [1]byte
4167
4168   if EventFdOffset <= readyFd {
4169     fd := int(readyFd-EventFdOffset)
4170     _,err := syscall.Read(fd,buf[0:1])
4171     if( err != nil ){
4172       return EOF;
4173     }else{
4174       if buf[0] == EV_MODE {
4175         recvEvent(fd)
4176         goto READ1
4177       }
4178       return int(buf[0])
4179     }
4180   }
4181
4182   _,err := fp.Read(buf[0:1])
4183   if( err != nil ){
4184     return EOF;
4185   }else{
4186     return int(buf[0])
4187   }
4188 }
4189
4190 func visibleChar(ch int)(string){
4191   switch {
4192     case '!' <= ch && ch <= '~':
4193       return string(ch)
4194   }
4195   switch ch {
4196     case ':': return "\\s"
4197     case '\n': return "\\n"
4198     case '\r': return "\\r"
4199     case '\t': return "\\t"
4200   }
4201   switch ch {
4202     case 0x00: return "NUL"
4203     case 0x07: return "BEL"
4204     case 0x08: return "BS"
4205     case 0x0E: return "SO"
4206     case 0x0F: return "SI"
4207     case 0x1B: return "ESC"
4208     case 0x7F: return "DEL"
4209   }
4210   switch ch {
4211     case EV_IDLE: return fmt.Sprintf("IDLE")
4212     case EV_MODE: return fmt.Sprintf("MODE")
4213   }
4214   return fmt.Sprintf("%X",ch)
4215 }
4216 func recvEvent(fd int){

```

```

4217     var buf = make([]byte,1)
4218     _,_ = syscall.Read(fd,buf[0:1])
4219     if( buf[0] != 0 ){
4220         romkanmode = true
4221     }else{
4222         romkanmode = false
4223     }
4224 }
4225 func (gsh*GshContext)xScanReplay(hix int,replay bool,repeat int,tempo float64,argv[]string){
4226     var Start time.Time
4227     var events = []Event{
4228         for i,e := range Events {
4229             if hix == 0 || e.CmdIndex == hix {
4230                 events = append(events,e)
4231             }
4232         }
4233     }
4234     elen := len(events)
4235     if 0 < elen {
4236         if events[elen-1].event == EV_IDLE {
4237             events = events[0:elen-1]
4238         }
4239         for r := 0; r < repeat; r++ {
4240             for i,e := range events {
4241                 nano := e.when.Nanosecond()
4242                 micro := nano / 1000
4243                 if Start.Second() == 0 {
4244                     Start = time.Now()
4245                 }
4246                 diff := time.Now().Sub(Start)
4247                 if replay {
4248                     if e.event != EV_IDLE {
4249                         putEvent(e.event,0)
4250                         if e.event == EV_MODE { // event with arg
4251                             putEvent(int(e.evarg),0)
4252                         }
4253                     }
4254                 }else{
4255                     fmt.Printf("%7.3fms #%-3v !%-3v [%v.%06d] %3v %02X %-4v %10.3fms\n",
4256                         float64(diff)/1000000.0,
4257                         i,
4258                         e.CmdIndex,
4259                         e.when.Format(time.Stamp),micro,
4260                         e.event,e.event,visibleChar(e.event),
4261                         float64(e.evarg)/1000000.0)
4262                 }
4263                 if e.event == EV_IDLE {
4264                     d := time.Duration(float64(time.Duration(e.evarg)) * tempo)
4265                     //nsleep(time.Duration(e.evarg))
4266                     nsleep(d)
4267                 }
4268             }
4269         }
4270     }
4271     func dumpEvents(argv[]string){
4272         hix := 0
4273         if 1 < len(argv) {
4274             fmt.Sscanf(argv[1],"%d",&hix)
4275         }
4276         for i,e := range Events {
4277             nano := e.when.Nanosecond()
4278             micro := nano / 1000
4279             //if e.event != EV_TIMEOUT {
4280             if hix == 0 || e.CmdIndex == hix {
4281                 fmt.Printf("#%-3v !%-3v [%v.%06d] %3v %02X %-4v %10.3fms\n",i,
4282                     e.CmdIndex,
4283                     e.when.Format(time.Stamp),micro,
4284                     e.event,e.event,visibleChar(e.event),float64(e.evarg)/1000000.0)
4285             }
4286         //}
4287     }
4288 }
4289 func fgetcTimeout(fp *os.File,usec int)(int){
4290     ch := fgetcTimeout1(fp,usec)
4291     if ch != EV_TIMEOUT {
4292         now := time.Now()
4293         if 0 < len(Events) {
4294             last := Events[len(Events)-1]
4295             dura := int64(now.Sub(last.when))
4296             Events = append(Events,Event{last.when,EV_IDLE,dura,last.CmdIndex})
4297         }
4298         Events = append(Events,Event{time.Now(),ch,0,CmdIndex})
4299     }
4300     return ch
4301 }
4302
4303 var TtyMaxCol = 72 // to be obtained by ioctl?
4304 var EscTimeout = (100*1000)
4305 var (
4306     MODE_VicMode    bool    // vi compatible command mode
4307     MODE_ShowMode   bool    // shown translation mode, the mode to be retained
4308     romkanmode      bool    // shown translation mode, the mode to be retained
4309     MODE_Recursive   bool    // recursive translation
4310     MODE_CapsLock  bool    // software CapsLock
4311     MODE_LowerLock bool    // force lower-case character lock
4312     MODE_ViInsert   int     // visible insert mode, should be like "I" icon in X Window
4313     MODE_ViTrace   bool    // output newline before translation
4314 )
4315 type IInput struct {
4316     lno int
4317     lastlno int
4318     pch []int // input queue
4319     prompt string
4320     line string
4321     right string
4322     inJMode bool
4323     pinJMode bool
4324     waitingMeta string // waiting meta character
4325     LastCmd string
4326 }
4327 func (iin*IInput)Getc(timeoutUs int)(int){
4328     ch1 := EOF
4329     ch2 := EOF
4330     ch3 := EOF
4331     if( 0 < len(iin.pch) ){ // deq
4332         ch1 = iin.pch[0]
4333         iin.pch = iin.pch[1:]
4334     }else{
4335         ch1 = fgetcTimeout(stdin,timeoutUs);
4336     }
4337     if( ch1 == 033 ){ // escape sequence
4338         ch2 = fgetcTimeout(stdin,EscTimeout);
4339         if( ch2 == EV_TIMEOUT ){
4340             }else{

```

```

4341 ch3 = fgetcTimeout(stdin,EscTimeout);
4342 if( ch3 == EV_TIMEOUT ){
4343     iin.pch = append(iin.pch,ch2) // enQ
4344 }else{
4345     switch( ch2 ){
4346         default:
4347             iin.pch = append(iin.pch,ch2) // enQ
4348             iin.pch = append(iin.pch,ch3) // enQ
4349     case '[':
4350         switch( ch3 ){
4351             case 'A': ch1 = GO_UP; // ^
4352             case 'B': ch1 = GO_DOWN; // v
4353             case 'C': ch1 = GO_RIGHT; // >
4354             case 'D': ch1 = GO_LEFT; // <
4355             case '3':
4356         ch4 := fgetcTimeout(stdin,EscTimeout);
4357             if( ch4 == '~' ){
4358 //fprintf(stderr,"%02X %02X %02X %02X]\n",ch1,ch2,ch3,ch4);
4359             ch1 = DEL_RIGHT
4360         }
4361     }
4362     case '\\\\':
4363 //ch4 := fgetcTimeout(stdin,EscTimeout);
4364 //fprintf(stderr,"y%02X %02X %02X %02X]\n",ch1,ch2,ch3,ch4);
4365         switch( ch3 ){
4366             case '_': ch1 = DEL_RIGHT
4367         }
4368     }
4369 }
4370 }
4371 return ch1
4373 }
4374 func (iin*IInput)clearline(){
4375 var i int
4376 fprintf(stderr,"%r");
4377 // should be ANSI ESC sequence
4378 for i = 0; i < TtyMaxCol; i++ { // to the max. position in this input action
4379     fputc(' ',os.Stderr);
4380 }
4381 fprintf(stderr,"%r");
4382 }
4383 func (iin*IInput)Redraw(){
4384     redraw(iin,iin.lno,iin.line,iin.right)
4385 }
4386 func redraw(iin *IInput,lno int,line string,right string){
4387     inMeta := false
4388     showMode := ""
4389     showMeta := "" // visible Meta mode on the cursor position
4390     showLino := fmt.Sprintf("%d ",lno)
4391     InsertMark := "" // in visible insert mode
4392
4393     if MODE_VicMode {
4394     }else{
4395         if 0 < len(iin.right) {
4396             InsertMark = " "
4397         }
4398
4399         if( 0 < len(iin.waitingMeta) ){
4400             inMeta = true
4401             if iin.waitingMeta[0] != 033 {
4402                 showMeta = iin.waitingMeta
4403             }
4404         }
4405         if( romkanmode ){
4406             //romkanmark = "*";
4407         }else{
4408             //romkanmark = "";
4409         }
4410
4411         if MODE_ShowMode {
4412             romkan := "-"
4413             inmeta := "-"
4414             inveri := ""
4415             if MODE_CapsLock {
4416                 inmeta = "A"
4417             }
4418             if MODE_LowerLock {
4419                 inmeta = "a"
4420             }
4421             if MODE_ViTrace {
4422                 inveri = "v"
4423             }
4424             if MODE_VicMode {
4425                 inveri = ":"}
4426
4427             if romkanmode {
4428                 romkan = "\u343/201\202"
4429                 if MODE_CapsLock {
4430                     inmeta = "R"
4431                 }else{
4432                     inmeta = "r"
4433                 }
4434             if inMeta {
4435                 inmeta = "\\"
4436             }
4437             showMode = "[" +romkan+inmeta+inveri+"]";
4438         }
4439         Pre := "\r" + showMode + showLino
4440         Output := ""
4441         Left := ""
4442         Right := ""
4443         if romkanmode {
4444             Left = convs(line)
4445             Right = InsertMark+convs(right)
4446         }else{
4447             Left = line
4448             Right = InsertMark+right
4449         }
4450         Output = Pre+Left
4451         if MODE_ViTrace {
4452             Output += iin.LastCmd
4453         }
4454         Output += showMeta+Right
4455         for len(Output) < TtyMaxCol { // to the max. position that may be dirty
4456             Output += " "
4457             // should be ANSI ESC sequence
4458             // not necessary just after newline
4459         }
4460         Output += Pre+Left+showMeta // to set the cursor to the current input position
4461         fprintf(stderr,"%s",Output)
4462
4463         if MODE_ViTrace {
4464             if 0 < len(iin.LastCmd) {
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6099
6100
6101
6102
6103
6104
6105
6106
6107
6108
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6180
6181
6182
6183
6184
6185
6186
6187
6188
6189
6190
6191
6192
6193
6194
6195
6196
6197
6198
6199
6200
6201
6202
6203
6204
6205
6206
6207
6208
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6220
6221
6222
6223
6224
6225
6226
6227
6228
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6240
6241
6242
6243
6244
6245
6246
6247
6248
6249
6250
6251
6252
6253
6254
6255
6256
6257
6258
6259
6260
6261
6262
6263
6264
6265
6266
6267
6268
6269
6270
6271
6272
6273
6274
6275
6276
6277
6278
6279
6280
6281
6282
6283
6284
6285
6286
6287
6288
6289
6290
6291
6292
6293
6294
6295
6296
6297
6298
6299
6300
6301
6302
6303
6304
6305
6306
6307
6308
6309
6310
6311
6312
6313
6314
6315
6316
6317
6318
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6370
6371
6372
6373
6374
6375
6376
6377
6378
6379
6380
6381
6382
6383
6384
6385
6386
6387
6388
6389
6390
6391
6392
6393
6394
6395
6396
6397
6398
6399
6400
6401
6402
6403
6404
6405
6406
6407
6408
6409
6410
6411
6412
6413
6414
6415
6416
6417
6418
6419
6420
6421
6422
6423
6424
6425
6426
6427
6428
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6460
6461
6462
6463
6464
6465
6466
6467
6468
6469
6470
6471
6472
6473
6474
6475
6476
6477
6478
6479
6480
6481
6482
6483
6484
6485
6486
6487
6488
6489
6490
6491
6492
6493
6494
6495
6496
6497
6498
6499
6500
6501
6502
6503
6504
6505
6506
6507
6508
6509
6510
6511
6512
6513
6514
6515
6516
6517
6518
6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558
6559
6560
6561
6562
6563
6564
6565
6566
6567
6568
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579
6580
6581
6582
6583
6584
6585
6586
6587
6588
6589
6590
6591
6592
6593
6594
6595
6596
6597
6598
6599
6600
6601
6602
6603
6604
6605
6606
6607
6608
6609
6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6640
6641
6642
6643
6644
6645
6646
6647
664
```

```

4465         iin.LastCmd = ""
4466         fprintf(stderr, "\r\n")
4467     }
4468 }
4469 }
4470 // <a href="https://golang.org/pkg/unicode/utf8/">utf8</a>
4471 func delHeadChar(str string)(rline string, head string){
4472     _,clen := utf8.DecodeRune([]byte(str))
4473     head = str[0:clen]
4474     return str[clen:],head
4475 }
4476 func delTailChar(str string)(rline string, last string){
4477     var i = 0
4478     var clen = 0
4479     for {
4480         _,siz := utf8.DecodeRune([]byte(str)[i:])
4481         if siz <= 0 { break }
4482         clen = siz
4483         i += siz
4484     }
4485     last = str[len(str)-clen:]
4486     return str[0:len(str)-clen],last
4487 }
4488
4489 // 3> for output and history
4490 // 4> for keylog?
4491 // <a name="getline">Command Line Editor</a>
4492 func xgetline(lno int, prevline string, gsh*GshContext)(string){
4493     var iin IInput
4494     iin.lastlno = lno
4495     iin.lno = lno
4496
4497     CmdIndex = len(gsh.CommandHistory)
4498     if( isatty(0) == 0 ){
4499         if( sfgets(&iin.line,LINESIZE,stdin) == NULL ){
4500             iin.line = "exit\n";
4501         }else{
4502         }
4503         return iin.line
4504     }
4505     if( true ){
4506         //var pts string;
4507         //pts = ptsname(0);
4508         //pts = ttynname(0);
4509         //fprintf(stderr,"--pts[0] = %s\n",pts?pts:"?");
4510     }
4511     if( false ){
4512         fprintf(stderr,! " ");
4513         fflush(stderr);
4514         sfgets(&iin.line,LINESIZE,stdin);
4515         return iin.line
4516     }
4517     system("/bin/stty -echo -icanon");
4518     xline := iin.xgetline1(prevline,gsh)
4519     system("/bin/stty echo sane");
4520     return xline
4521 }
4522 func (iin*IInput)Translate(cmdch int){
4523     romkanmode = !romkanmode;
4524     if MODE_ViTrace {
4525         fprintf(stderr,"%v\r\n",string(cmdch));
4526     }else{
4527         if( cmdch == 'J' ){
4528             fprintf(stderr,"J\r\n");
4529             iin.inMode = true
4530         }
4531         iin.Redraw();
4532         loadDefaultDic(cmdch);
4533         iin.Redraw();
4534     }
4535     func (iin*IInput)Replace(cmdch int){
4536         iin.LastCmd = fmt.Sprintf("\v\v",string(cmdch))
4537         iin.Redraw();
4538         loadDefaultDic(cmdch);
4539         dst := convs(iin.line+iin.right);
4540         iin.line = dst
4541         iin.right = ""
4542         if( cmdch == 'I' ){
4543             fprintf(stderr,"I\r\n");
4544             iin.inMode = true
4545         }
4546         iin.Redraw();
4547     }
4548 // aa 12 ala1
4549 func isAlpha(ch rune)(bool){
4550     if 'a' <= ch && ch <= 'z' || 'A' <= ch && ch <= 'Z' {
4551         return true
4552     }
4553     return false
4554 }
4555 func isAlnum(ch rune)(bool){
4556     if 'a' <= ch && ch <= 'z' || 'A' <= ch && ch <= 'Z' {
4557         return true
4558     }
4559     if '0' <= ch && ch <= '9' {
4560         return true
4561     }
4562     return false
4563 }
4564
4565 // 0.2.8 2020-0901 created
4566 // <a href="https://golang.org/pkg/unicode/utf8/#DecodeRuneInString">DecodeRuneInString</a>
4567 func (iin*IInput)GotoTOPW(){
4568     str := iin.line
4569     i := len(str)
4570     if i <= 0 {
4571         return
4572     }
4573     //i0 := i
4574     i -= 1
4575     lastSize := 0
4576     var lastRune rune
4577     var found = -1
4578     for 0 < i { // skip preamble spaces
4579         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4580         if !isalnum(lastRune) { // character, type, or string to be searched
4581             i -= lastSize
4582             continue
4583         }
4584         break
4585     }
4586     for 0 < i {
4587         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4588         if lastSize <= 0 { continue } // not the character top

```

```

4589     if !isAlnum(lastRune) { // character, type, or string to be searched
4590         found = i
4591         break
4592     }
4593     i -= lastSize
4594   }
4595   if found < 0 && i == 0 {
4596     found = 0
4597   }
4598   if 0 <= found {
4599     if isAlnum(lastRune) { // or non-kana character
4600       }else{ // when positioning to the top o the word
4601         i += lastSize
4602       }
4603       iin.right = str[i:] + iin.right
4604       if 0 < i {
4605         iin.line = str[0:i]
4606       }else{
4607         iin.line = ""
4608       }
4609     }
4610   //fmt.Printf("\n(%d,%d,%d)[%s]\n",i0,i,found,iin.line,iin.right)
4611   //fmt.Println("") // set debug messae at the end of line
4612 }
4613 // 0.2.8 2020-0901 created
4614 func (iin*IInput)GotoENDW(){
4615   str := iin.right
4616   if len(str) <= 0 {
4617     return
4618   }
4619   lastSize := 0
4620   var lastRune rune
4621   var lastW = 0
4622   i := 0
4623   inWord := false
4624
4625   lastRune,lastSize = utf8.DecodeRuneInString(str[0:])
4626   if isAlnum(lastRune) {
4627     r,z := utf8.DecodeRuneInString(str[lastSize:])
4628     if 0 < z && isAlnum(r) {
4629       inWord = true
4630     }
4631   }
4632   for i < len(str) {
4633     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4634     if lastSize <= 0 { break } // broken data?
4635     if !isAlnum(lastRune) { // character, type, or string to be searched
4636       break
4637     }
4638     lastW = i // the last alnum if in alnum word
4639     i += lastSize
4640   }
4641   if inWord {
4642     goto DISP
4643   }
4644   for i < len(str) {
4645     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4646     if lastSize <= 0 { break } // broken data?
4647     if isAlnum(lastRune) { // character, type, or string to be searched
4648       break
4649     }
4650     i += lastSize
4651   }
4652   for i < len(str) {
4653     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4654     if lastSize <= 0 { break } // broken data?
4655     if !isAlnum(lastRune) { // character, type, or string to be searched
4656       break
4657     }
4658     lastW = i
4659     i += lastSize
4660   }
4661 DISP:
4662   if 0 < lastW {
4663     iin.line = iin.line + str[0:lastW]
4664     iin.right = str[lastW:]
4665   }
4666   //fmt.Printf("\n(%d)[%s]\n",i,iin.line,iin.right)
4667   //fmt.Println("") // set debug messae at the end of line
4668 }
4669 // 0.2.8 2020-0901 created
4670 func (iin*IInput)GotoNEXTW(){
4671   str := iin.right
4672   if len(str) <= 0 {
4673     return
4674   }
4675   lastSize := 0
4676   var lastRune rune
4677   var found = -1
4678   i := 1
4679   for i < len(str) {
4680     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4681     if lastSize <= 0 { break } // broken data?
4682     if !isAlnum(lastRune) { // character, type, or string to be searched
4683       found = i
4684       break
4685     }
4686     i += lastSize
4687   }
4688   if 0 < found {
4689     if isAlnum(lastRune) { // or non-kana character
4690       }else{ // when positioning to the top o the word
4691         found += lastSize
4692       }
4693       iin.line = iin.line + str[0:found]
4694       if 0 < found {
4695         iin.right = str[found:]
4696       }else{
4697         iin.right = ""
4698       }
4699     }
4700   //fmt.Printf("\n(%d)[%s]\n",i,iin.line,iin.right)
4701   //fmt.Println("") // set debug messae at the end of line
4702 }
4703 // 0.2.8 2020-0902 created
4704 func (iin*IInput)GotoPAIRCH(){
4705   str := iin.right
4706   if len(str) <= 0 {
4707     return
4708   }
4709   lastRune,lastSize := utf8.DecodeRuneInString(str[0:])
4710   if lastSize <= 0 {
4711     return
4712   }

```

```

4713     forw := false
4714     back := false
4715     pair := ""
4716     switch string(lastRune){
4717         case "(":
4718             pair = "("; forw = true
4719         case ")":
4720             pair = ")"; back = true
4721         case "[":
4722             pair = "["; back = true
4723         case "]":
4724             pair = "]"; back = true
4725         case "<":
4726             pair = "<"; forw = true
4727         case ">":
4728             pair = ">"; back = true
4729         case "\"":
4730             pair = "\""; // context depednet, can be f" or back-double quote
4731         case "'":
4732             pair = "'"; // context depednet, can be f' or back-quote
4733         case `:
4734             pair = ``;
4735     }
4736 // 0.2.8 2020-0902 created
4737 func (iin*IInput)SearchForward(pat string)(bool){
4738     right := iin.right
4739     found := -1
4740     i := 0
4741     if strBegins(right,pat) {
4742         _,z := utf8.DecodeRuneInString(right[i:])
4743         if 0 < z {
4744             i += z
4745         }
4746     }
4747     for i < len(right) {
4748         if strBegins(right[i:],pat) {
4749             found = i
4750             break
4751         }
4752         _,z := utf8.DecodeRuneInString(right[i:])
4753         if z <= 0 { break }
4754         i += z
4755     }
4756     if 0 <= found {
4757         iin.line = iin.line + right[0:found]
4758         iin.right = iin.right[found:]
4759         return true
4760     }else{
4761         return false
4762     }
4763 }
4764 // 0.2.8 2020-0902 created
4765 func (iin*IInput)SearchBackward(pat string)(bool){
4766     line := iin.line
4767     found := -1
4768     i := len(line)-1
4769     for i = i; 0 <= i; i-- {
4770         _,z := utf8.DecodeRuneInString(line[i:])
4771         if z <= 0 {
4772             continue
4773         }
4774         //fprintf(stderr,"-- %v %v\n",pat,line[i:])
4775         if strBegins(line[i:],pat) {
4776             found = i
4777             break
4778         }
4779     }
4780     //fprintf(stderr,"--%d\n",found)
4781     if 0 <= found {
4782         iin.right = line[found:] + iin.right
4783         iin.line = line[0:found]
4784         return true
4785     }else{
4786         return false
4787     }
4788 }
4789 // 0.2.8 2020-0902 created
4790 // search from top, end, or current position
4791 func (gsh*GshContext)SearchHistory(pat string, forw bool)(bool,string){
4792     if forw {
4793         for ,v := range gsh.CommandHistory {
4794             if 0 <= strings.Index(v.CmdLine,pat) {
4795                 //fprintf(stderr,"n--De-- found !%v [%v]\n",i,pat,v.CmdLine)
4796                 return true,v.CmdLine
4797             }
4798         }
4799     }else{
4800         hlen := len(gsh.CommandHistory)
4801         for i := hlen-1; 0 < i ; i-- {
4802             v := gsh.CommandHistory[i]
4803             if 0 <= strings.Index(v.CmdLine,pat) {
4804                 //fprintf(stderr,"n--De-- found !%v [%v]\n",i,pat,v.CmdLine)
4805                 return true,v.CmdLine
4806             }
4807         }
4808     }
4809     //fprintf(stderr,"n-- not-found(%v)\n",pat)
4810     return false,"(Not Found in History)"
4811 }
4812 // 0.2.8 2020-0902 created
4813 func (iin*IInput)gotoFORWSTR(pat string,gsh*GshContext){
4814     found := false
4815     if 0 < len(iin.right) {
4816         found = iin.SearchForward(pat)
4817     }
4818     if !found {
4819         found,line := gsh.SearchHistory(pat,true)
4820         if found {
4821             iin.line = line
4822             iin.right = ""
4823         }
4824     }
4825 }
4826 func (iin*IInput)GotoBACKSTR(pat string, gsh*GshContext){
4827     found := false
4828     if 0 < len(iin.line) {
4829         found = iin.SearchBackward(pat)
4830     }
4831     if !found {
4832         found,line := gsh.SearchHistory(pat,false)
4833         if found {
4834             iin.line = line
4835             iin.right = ""
4836         }
4837 }

```

```

4837     }
4838 }
4839 func (iin*IInput)getStringl(prompt string)(string){ // should be editable
4840     iin.clearline();
4841     fprintf(stderr, "\r\n", prompt);
4842     str := ""
4843     for {
4844         ch := iin.Getc(10*1000*1000)
4845         if ch == '\n' || ch == '\r' {
4846             break
4847         }
4848         sch := string(ch)
4849         str += sch
4850         fprintf(stderr, "%s", sch)
4851     }
4852     return str
4853 }
4854
4855 // search pattern must be an array and selectable with ^N/^P
4856 var SearchPat = ""
4857 var SearchForw = true
4858
4859 func (iin*IInput)xgetline1(prevline string, gsh*GshContext)(string){
4860     var ch int;
4861
4862     MODE_ShowMode = false
4863     MODE_VicMode = false
4864     iin.Redraw();
4865     first := true
4866
4867     for cix := 0; ; cix++ {
4868         iin.pinJmode = iin.inJmode
4869         iin.inJmode = false
4870
4871         ch = iin.Getc(1000*1000)
4872
4873         if ch != EV_TIMEOUT && first {
4874             first = false
4875             mode := 0
4876             if romkanemode {
4877                 mode = 1
4878             }
4879             now := time.Now()
4880             Events = append(Events, Event{now, EV_MODE, int64(mode), CmdIndex})
4881         }
4882         if ch == 033 {
4883             MODE_ShowMode = true
4884             MODE_VicMode = !MODE_VicMode
4885             iin.Redraw();
4886             continue
4887         }
4888         if MODE_VicMode {
4889             switch ch {
4890                 case '0': ch = GO_TOPL
4891                 case '$': ch = GO_ENDL
4892                 case 'b': ch = GO_TOPW
4893                 case 'e': ch = GO_ENDW
4894                 case 'w': ch = GO_NEXTW
4895                 case '%': ch = GO_PAIRCH
4896
4897                 case 'j': ch = GO_DOWN
4898                 case 'k': ch = GO_UP
4899                 case 'h': ch = GO_LEFT
4900                 case 'l': ch = GO_RIGHT
4901                 case 'x': ch = DEL_RIGHT
4902                 case 'a': MODE_VicMode = !MODE_VicMode
4903                     ch = GO_RIGHT
4904                 case 'i': MODE_VicMode = !MODE_VicMode
4905                     iin.Redraw();
4906                     continue
4907                 case '-':
4908                     right,head := delHeadChar(iin.right)
4909                     if len([]byte(head)) == 1 {
4910                         ch = int(head[0])
4911                         if( 'a' <= ch && ch <= 'z' ){
4912                             ch = ch + 'A'-'a'
4913                         }else{
4914                             if( 'A' <= ch && ch <= 'Z' ){
4915                                 ch = ch + 'a'-'A'
4916                             }
4917                         iin.right = string(ch) + right
4918                     }
4919                     iin.Redraw();
4920                     continue
4921                 case 'f': // GO_FORWCH
4922                     iin.Redraw();
4923                     ch = iin.Getc(3*1000*1000)
4924                     if ch == EV_TIMEOUT {
4925                         iin.Redraw();
4926                         continue
4927                     }
4928                     SearchPat = string(ch)
4929                     SearchForw = true
4930                     iin.GotoFORWSTR(SearchPat,gsh)
4931                     iin.Redraw();
4932                     continue
4933                 case '/':
4934                     SearchPat = iin.getStringl("//") // should be editable
4935                     SearchForw = true
4936                     iin.GotoFORWSTR(SearchPat,gsh)
4937                     iin.Redraw();
4938                     continue
4939                 case '?':
4940                     SearchPat = iin.getStringl("?) // should be editable
4941                     SearchForw = false
4942                     iin.GotoBACKSTR(SearchPat,gsh)
4943                     iin.Redraw();
4944                     continue
4945                 case 'n':
4946                     if SearchForw {
4947                         iin.GotoFORWSTR(SearchPat,gsh)
4948                     }else{
4949                         iin.GotoBACKSTR(SearchPat,gsh)
4950                     }
4951                     iin.Redraw();
4952                     continue
4953                 case 'N':
4954                     if !SearchForw {
4955                         iin.GotoFORWSTR(SearchPat,gsh)
4956                     }else{
4957                         iin.GotoBACKSTR(SearchPat,gsh)
4958                     }
4959                     iin.Redraw();
4960                     continue

```

```

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
4983 //fprintf(stderr,"A[%02X]\n",ch);
4984 if( ch == '\\\' || ch == 033 ){
4985     MODE_ShowMode = true;
4986     metach := ch;
4987     iin.waitingMeta = string(ch);
4988     iin.Redraw();
4989     // set cursor //fprintf(stderr,"???\b\b\b")
4990     ch = fgettimeout(stdin,2000*1000)
4991     // reset cursor
4992     iin.waitingMeta = "";
4993
4994     cmdch := ch;
4995     if( ch == EV_TIMEOUT ){
4996         if metach == 033 {
4997             continue;
4998         }
4999         ch = metach;
5000     }else
5001     /*
5002     if( ch == 'm' || ch == 'M' ){
5003         mch := fgettimeout(stdin,1000*1000)
5004         if mch == 'r' {
5005             romkanmode = true;
5006         }else{
5007             romkanmode = false;
5008         }
5009         continue;
5010     }else
5011     /*
5012     if( ch == 'k' || ch == 'K' ){
5013         MODE_Recursive = !MODE_Recursive;
5014         iin.Translate(cmdch);
5015         continue;
5016     }else
5017     if( ch == 'j' || ch == 'J' ){
5018         iin.Translate(cmdch);
5019         continue;
5020     }else
5021     if( ch == 'i' || ch == 'I' ){
5022         iin.Replace(cmdch);
5023         continue;
5024     }else
5025     if( ch == 'l' || ch == 'L' ){
5026         MODE_LowerLock = !MODE_LowerLock;
5027         MODE_CapsLock = false;
5028         if MODE_ViTrace {
5029             fprintf(stderr,"%v\r\n",string(cmdch));
5030         }
5031         iin.Redraw();
5032         continue;
5033     }else
5034     if( ch == 'u' || ch == 'U' ){
5035         MODE_CapsLock = !MODE_CapsLock;
5036         MODE_LowerLock = false;
5037         if MODE_ViTrace {
5038             fprintf(stderr,"%v\r\n",string(cmdch));
5039         }
5040         iin.Redraw();
5041         continue;
5042     }else
5043     if( ch == 'v' || ch == 'V' ){
5044         MODE_ViTrace = !MODE_ViTrace;
5045         if MODE_ViTrace {
5046             fprintf(stderr,"%v\r\n",string(cmdch));
5047         }
5048         iin.Redraw();
5049         continue;
5050     }else
5051     if( ch == 'c' || ch == 'C' ){
5052         if 0 < len(iin.line) {
5053             xline,tail := delTailChar(iin.line);
5054             if len(jbyte(tail)) == 1 {
5055                 ch = int(tail[0]);
5056                 if( 'a' <= ch && ch <= 'z' ){
5057                     ch = ch + 'A'-'a';
5058                 }else{
5059                     if( 'A' <= ch && ch <= 'Z' ){
5060                         ch = ch + 'a'-'A';
5061                     }
5062                     iin.line = xline + string(ch);
5063                 }
5064             }
5065             if MODE_ViTrace {
5066                 fprintf(stderr,"%v\r\n",string(cmdch));
5067             }
5068             iin.Redraw();
5069             continue;
5070         }else{
5071             iin.pch = append(iin.pch,ch) // push
5072             ch = '\\\' ;
5073         }
5074     }switch( ch ){
5075         case 'P'-0x40: ch = GO_UP;
5076         case 'N'-0x40: ch = GO_DOWN;
5077         case 'B'-0x40: ch = GO_LEFT;
5078         case 'F'-0x40: ch = GO_RIGHT;
5079     }
5080
5081 //fprintf(stderr,"B[%02X]\n",ch);
5082 switch( ch ){
5083     case 0:
5084         continue;

```

```

5085     case '\t':
5086         iin.Replace('j');
5087         continue
5088     case 'X'-0x40:
5089         iin.Replace('j');
5090         continue
5091     case EV_TIMEOUT:
5092         iin.Redraw();
5093         if iin.pinMode {
5094             fprintf(stderr, "\\J\\r\\n")
5095             iin.inJMode = true
5096         }
5097         continue
5098     case GO_UP:
5099         if iin.lno == 1 {
5100             continue
5101         }
5102         cmd,ok := gsh.cmdStringInHistory(iin.lno-1)
5103         if ok {
5104             iin.line = cmd
5105             iin.right = ""
5106             iin.lno = iin.lno - 1
5107         }
5108         iin.Redraw();
5109         continue
5110     case GO_DOWN:
5111         cmd,ok := gsh.cmdStringInHistory(iin.lno+1)
5112         if ok {
5113             iin.line = cmd
5114             iin.right = ""
5115             iin.lno = iin.lno + 1
5116         }else{
5117             iin.line = ""
5118             iin.right = ""
5119             if iin.lno == iin.lastlno-1 {
5120                 iin.lno = iin.lno + 1
5121             }
5122         }
5123         iin.Redraw();
5124         continue
5125     case GO_LEFT:
5126         if 0 < len(iin.line) {
5127             xline,tail := delTailChar(iin.line)
5128             iin.line = xline
5129             iin.right = tail + iin.right
5130         }
5131         iin.Redraw();
5132         continue;
5133     case GO_RIGHT:
5134         if( 0 < len(iin.right) && iin.right[0] != 0 ){
5135             xright,head := delHeadChar(iin.right)
5136             iin.right = xright
5137             iin.line += head
5138         }
5139         iin.Redraw();
5140         continue;
5141     case EOF:
5142         goto EXIT;
5143     case 'R'-0x40: // replace
5144         dst := convs(iin.line+iin.right);
5145         iin.line = dst
5146         iin.right = ""
5147         iin.Redraw();
5148         continue;
5149     case 'T'-0x40: // just show the result
5150         readDic();
5151         romkanmode = !romkanmode;
5152         iin.Redraw();
5153         continue;
5154     case 'L'-0x40:
5155         iin.Redraw();
5156         continue
5157     case 'K'-0x40:
5158         iin.right = "";
5159         iin.Redraw();
5160         continue
5161     case 'E'-0x40:
5162         iin.line += iin.right
5163         iin.right = ""
5164         iin.Redraw();
5165         continue
5166     case 'A'-0x40:
5167         iin.right = iin.line + iin.right
5168         iin.line = ""
5169         iin.Redraw();
5170         continue
5171     case 'U'-0x40:
5172         iin.line = ""
5173         iin.right = ""
5174         iin.clearline();
5175         iin.Redraw();
5176         continue;
5177     case DEL_RIGHT:
5178         if( 0 < len(iin.right) ){
5179             iin.right,_ = delHeadChar(iin.right)
5180             iin.Redraw();
5181         }
5182         continue;
5183     case 0xF: // BS? not DEL
5184         if( 0 < len(iin.line) ){
5185             iin.line,_ = delTailChar(iin.line)
5186             iin.Redraw();
5187         }
5188         /*
5189         else
5190             if( 0 < len(iin.right) ){
5191                 iin.right,_ = delHeadChar(iin.right)
5192                 iin.Redraw();
5193             }
5194         */
5195         continue;
5196     case 'H'-0x40:
5197         if( 0 < len(iin.line) ){
5198             iin.line,_ = delTailChar(iin.line)
5199             iin.Redraw();
5200         }
5201         continue;
5202     }
5203     if( ch == '\n' || ch == '\r' ){
5204         iin.line += iin.right;
5205         iin.right = ""
5206         iin.Redraw();
5207     }
5208 
```

```

5209         fputc(ch,stderr);
5210         break;
5211     }
5212     if MODE_CapsLock {
5213         if 'a' <= ch && ch <= 'z' {
5214             ch = ch+'A'-'a';
5215         }
5216     } else if MODE_LowerLock {
5217         if 'A' <= ch && ch <= 'Z' {
5218             ch = ch+'a'-'A';
5219         }
5220     }
5221     iin.line += string(ch);
5222     iin.Redraw();
5223 }
5224 }
5225 EXIT:
5226     return iin.line + iin.right;
5227 }
5228
5229 func getline_main(){
5230     line := xgetline(0,"",nil)
5231     fprintf(stderr,"%s\n",line);
5232 /*
5233     dp = strpbrk(line,"\r\n");
5234     if( dp != NULL ){
5235         *dp = 0;
5236     }
5237
5238     if( 0 ){
5239         fprintf(stderr,"\n%d\n",int(strlen(line)));
5240     }
5241     if( lseek(3,0,0) == 0 ){
5242         if( romkanmode ){
5243             var buf [8*1024]byte;
5244             convs(line,buf);
5245             strcpy(line,buf);
5246         }
5247         write(3,line,strlen(line));
5248         ftruncate(3,lseek(3,0,SEEK_CUR));
5249         //fprintf(stderr,"outsize=%d\n",(int)lseek(3,0,SEEK_END));
5250         lseek(3,0,SEEK_SET);
5251         close(3);
5252     }else{
5253         fprintf(stderr,"\r\ngetline: ");
5254         trans(line);
5255         //printf("%s\n",line);
5256         printf("\n");
5257     }
5258 */
5259 }
5260 //== end ===== getline
5261
5262 //
5263 // $USERHOME/.gsh/
5264 //     gsh-rc.txt, or gsh-configure.txt
5265 //     gsh-history.txt
5266 //     gsh-aliases.txt // should be conditional?
5267 //
5268 func (gshCtx *GshContext)gshSetupHomedir()(bool) {
5269     homedir,found := userHomeDir()
5270     if !found {
5271         fmt.Println("--E-- You have no UserHomeDir\n")
5272         return true
5273     }
5274     gshhome := homedir + "/" + GSH_HOME
5275     _,err2 := os.Stat(gshhome)
5276     if err2 != nil {
5277         err3 := os.Mkdir(gshhome,0700)
5278         if err3 != nil {
5279             fmt.Println("--E-- Could not Create %s (%s)\n",
5280                     gshhome,err3)
5281             return true
5282         }
5283         fmt.Println("--I-- Created %s\n",gshhome)
5284     }
5285     gshCtx.GshHomeDir = gshhome
5286     return false
5287 }
5288 func setupGshContext()(GshContext,bool){
5289     gshPA := syscall.PrcAttr {
5290         "", // the staring directory
5291         os.Environ(), // environ[]
5292         []uintptr{os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd()},
5293         nil, // OS specific
5294     }
5295     cwd, _ := os.Getwd()
5296     gshCtx := GshContext {
5297         cwd, // Startdir
5298         "", // GetLine
5299         []GChdirHistory { {cwd,time.Now(),0} }, // ChdirHistory
5300         gshPA,
5301         []GCommandHistory{}, //something for invocation?
5302         GCommandHistory{}, // CmdCurrent
5303         false,
5304         []int{},
5305         syscall.Rusage{},
5306         "", // GshHomeDir
5307         Ttyid(),
5308         false,
5309         false,
5310         []PluginInfo{},
5311         []string{},
5312         " ",
5313         "v",
5314         ValueStack{},
5315         GServer{"",""}, // LastServer
5316         "", // RSERV
5317         cwd, // RWD
5318         CheckSum{},
5319     }
5320     err := gshCtx.gshSetupHomedir()
5321     return gshCtx, err
5322 }
5323 func (gsh*GshContext)gshellh(gline string)(bool){
5324     ghist := gsh.CmdCurrent
5325     ghist.Workdir,_ = os.Getwd()
5326     ghist.WorkdirX = len(gsh.ChdirHistory)-1
5327     //fmt.Printf("--D--ChdirHistory(%d)\n",len(gsh.ChdirHistory))
5328     ghist.StartAt = time.Now()
5329     rusagev1 := Getrusagev()
5330     gsh.CmdCurrent.Foundfile = []string{}
5331     fin := gsh.tgshell1(gline)
5332     rusagev2 := Getrusagev()

```

```

5333     ghist.Rusagev = RusageSubv(rusagev2,rusagev1)
5334     ghist.EndAt = time.Now()
5335     ghist.CmdLine = gline
5336     ghist.FoundFile = gsh.CmdCurrent.FoundFile
5337
5338     /* record it but not show in list by default
5339     if len(gline) == 0 {
5340         continue
5341     }
5342     if gline == "hi" || gline == "history" { // don't record it
5343         continue
5344     }
5345 */
5346     gsh.CommandHistory = append(gsh.CommandHistory, ghist)
5347     return fin
5348 }
5349 // <a name="main">Main loop</a>
5350 func script(gshCtxGiven *GshContext) (_ GshContext) {
5351     gshCtxBuf,err0 := setupGshContext()
5352     if err0 {
5353         return gshCtxBuf;
5354     }
5355     gshCtx := &gshCtxBuf
5356
5357     //fmt.Printf("--I-- GSH_HOME=%s\n",gshCtx.GshHomeDir)
5358     //resmap()
5359
5360 /*
5361 if false {
5362     gsh_getlinev, with_exgetline :=
5363         which("PATH",[]string{"which","gsh-getline","-s"})
5364     if with_exgetline {
5365         gsh_getlinev[0] = toFullPath(gsh_getlinev[0])
5366         gshCtx.GetLine = toFullPath(gsh_getlinev[0])
5367     }else{
5368         fmt.Println("--W-- No gsh-getline found. Using internal getline.\n");
5369     }
5370 }
5371 */
5372
5373     ghist0 := gshCtx.CmdCurrent // something special, or gshrc script, or permanent history
5374     gshCtx.CommandHistory = append(gshCtx.CommandHistory,ghist0)
5375
5376     prevline := ""
5377     skipping := false
5378     for hix := len(gshCtx.CommandHistory); ; {
5379         gline := gshCtx.getline(hix,skipping,prevline)
5380         if skipping {
5381             if strings.Index(gline,"fi") == 0 {
5382                 fmt.Printf("fi\n");
5383                 skipping = false;
5384             }else{
5385                 //fmt.Println("%s\n",gline);
5386             }
5387             continue
5388         }
5389         if strings.Index(gline,"if") == 0 {
5390             //fmt.Println("--D-- if start: %s\n",gline);
5391             skipping = true;
5392             continue
5393         }
5394         if false {
5395             os.Stdout.Write([]byte("gotline:"))
5396             os.Stdout.Write([]byte(gline))
5397             os.Stdout.Write([]byte("\n"))
5398         }
5399         gline = strsubst(gshCtx,gline,true)
5400         if false {
5401             fmt.Printf("fmt.Println %v - %v\n",gline)
5402             fmt.Printf("fmt.Println %s - %s\n",gline)
5403             fmt.Printf("fmt.Println %x - %s\n",gline)
5404             fmt.Printf("fmt.Println %U - %s\n",gline)
5405             fmt.Printf("Stout.Write -")
5406             os.Stdout.Write([]byte(gline))
5407             fmt.Printf("\n")
5408         }
5409     /*
5410 // should be cared in substitution ?
5411 if 0 < len(gline) && gline[0] == '!' {
5412     xgline, set, err := searchHistory(gshCtx,gline)
5413     if err {
5414         continue
5415     }
5416     if set {
5417         // set the line in command line editor
5418     }
5419     gline = xgline
5420 }
5421 */
5422     fin := gshCtx.gshellh(gline)
5423     if fin {
5424         break;
5425     }
5426     prevline = gline;
5427     hix++;
5428 }
5429     return *gshCtx
5430 }
5431 func main() {
5432     gshCtxBuf := GshContext{}
5433     gsh := &gshCtxBuf
5434     argv := os.Args
5435
5436     if( isin("wss",argv) ){
5437         ws_server(argv[1:]);
5438         return;
5439     }
5440     if( isin("wsc",argv) ){
5441         ws_client(argv[1:]);
5442         return;
5443     }
5444     if 1 < len(argv) {
5445         if isin("version",argv){
5446             gsh.showVersion(argv)
5447             return
5448         }
5449         comx := isinX("-c",argv)
5450         if 0 < comx {
5451             gshCtxBuf,err := setupGshContext()
5452             gsh := &gshCtxBuf
5453             if !err {
5454                 gsh.gshellv(argv[comx+1:])
5455             }
5456         }
5457     }
5458 }
```

```

5457     }
5458   }  

5459   if l < len(argv) && isin("-s", argv) {  

5460     gsh.showVersion(append(argv, []string{"-l", "-a"}...))  

5461   }  

5462   script(nil)  

5463   //gshCtx := script(nil)  

5465   //gshell(gshCtx, "time")  

5466 }  

5467  

5468 //</div></details>  

5469 //<details id="gsh-todo"><summary>Considerations</summary><div class="gsh-src">  

5470 // - inter gsh communication, possibly running in remote hosts -- to be remote shell  

5471 // - merged histories of multiple parallel gsh sessions  

5472 // - alias as a function or macro  

5473 // - instant alias end environ export to the permanent > ~/.gsh/gsh-alias and gsh-environ  

5474 // - retrieval PATH of files by its type  

5475 // - gsh as an IME with completion using history and file names as dictionaires  

5476 // - gsh a scheduler in precise time of within a millisecond  

5477 // - all commands have its subucomand after "___" symbol  

5478 // - filename expansion by "find" command  

5479 // - history of ext code and output of each command  

5480 // - "script" output for each command by pty-tee or telnet-tee  

5481 // - $BUILTIN command in PATH to show the priority  

5482 // - "?" symbol in the command (not as in arguments) shows help request  

5483 // - searching command with wild card like: which ssh-*  

5484 // - longformat prompt after long idle time (should dismiss by BS)  

5485 // - customizing by building plugin and dynamically linking it  

5486 // - generating syntactic element like "if" by macro expansion (like CPP) >> alias  

5487 // - "!" symbol should be used for negation, don't wast it just for job control  

5488 // - don't put too long output to tty, record it into GSH_HOME/session-id/command-id.log  

5489 // - making canonical form of command at the start adding quotation or white spaces  

5490 // - name(a,b,c) ... use "(" and ")" to show both delimiter and realm  

5491 // - name? or name! might be useful  

5492 // - htar format - packing directory contents into a single html file using data scheme  

5493 // - filepath substitution shold be done by each command, especially in case of builtins  

5494 // - @N substition for the history of working directory, and @spec for more generic ones  

5495 // - @dir prefix to do the command at there, that means like (chdir @dir; command)  

5496 // - GSH_PATH for plugins  

5497 // - standard command output: list of data with name, size, resouce usage, modified time  

5498 // - generic sort key option -nm name, -sz size, -ru rusage, -ts start-time, -tm mod-time  

5499 // - wc word-count, grep match line count, ...  

5500 // - standard command execution result: a list of string, -tm, -ts, -ru, -sz, ...  

5501 // - tailf-filename like tail -f filename, repeat close and open before read  

5502 // - max. size and max. duration and timeout of (generated) data transfer  

5503 // - auto. numbering, aliasing, IME completion of file name (especially rm of queier name)  

5504 // - IME "?" at the top of the command line means searching history  

5505 // - IME #d/0x10000 %x/ffff/  

5506 // - IME ESC to go the edit mode like in vi, and use :command as :s/x/y/g to edit history  

5507 // - gsh in WebAssembly  

5508 // - gsh as a HTTP server of online-manual  

5509 //---END--- (^~^)/ITS more</div></details>  

5510  

5511 //<span class="gsh-golang-data">  

5512  

5513 var WorldDic = //<span id="gsh-world-dic">  

5514 "data:text/dic;base64," +  

5515 "Ly8gTJYUWVzZG4L665pu4ICgyMDiWLT4MTlhKOpzzWthaSDkuJbnlywKa28g44GT"+  

5516 "Cm5ui0OcKwpuaSDjgasKy2hpI0OBoOp0aSDjgaKeG4g4GvCnNlIOOBmwprySDkgysKaSDj"+  

5517 "gYQK";  

5518 //</span>  

5519  

5520 var WnnDic = //<span id="gsh-wnn-dic">  

5521 "data:text/dic;base64," +  

5522 "PG1ldGEgY2hhcnNldD0iVVRLTgiPg08dGV4dGFyZWEgY29scz04MCByb3dzPTQwPg0vL2Rp"+  

5523 "Y3ZlcglHU2hlbGcc01NRVxzZGljdGlvbmfYeVxZm9yXHNkbm5ccy8vXHMyMDiWLT4MzAK"+  

5524 "R1No2WxsCUdAgvSArjgo/jg2/jg2Cz56ebCndhdGFzaGk156eBCndhdGFzaQnnp4E4K4Gq"+  

5525 "44G14GICeWojeWjQpuYwlhZqnkI3l1iv0K44G44GL44GuCeS4remHjgpuYWhthbm8J5Llit"+  

5526 "6yeOCndhCeOCjwpx0Qnjg28Kc2k44GxCnNa0QnjgzcKbmJ44GuCm5hCeObBggpYQnjgbK"+  

5527 "Z0njqgYKacE44GvcMsheObqqgrYOnjqgYsKbm8J44CuCmrlCeObpvpzdQnjgzkZVxzCwVj"+  

5528 "agK2G1jCWRpIwpl2hvCWVjaG8KcmvwbGP5CXJlcgXheQpyZXBlYQJcmVwZWFOCmR0CwRh"+  

5529 "dGVccysnJVKlsvklSVI01VNOiVtJwp0aW9uCRpb24KJXQJXQJY8gdG8gYmUgYw4gYwn0"+  

5530 "aW9uCjwvdV4dGfyZWE+Cg=="  

5531 //</span>  

5532  

5533 var SumomoDic = //<span id="gsh-sumomo-dic">  

5534 "data:text/dic;base64," +  

5535 "PG1ldGEgY2hhcnNldD0iVVRLTgiPg08dGV4dGFyZWEgY29scz04MCByb3dzPTQwPg0vL3z1"+  

5536 "cg1HU2lbGcc01NRVxzZGljdGlvbmfYeVxZm9yXHNkbm5ccy8vXHMyMDiWLT4MzAK"+  

5537 "c3UJ44GZG2mlCeOcgppubwnjg4Kd0njqgYKKY2hpcOboQoap0aQnjgakDWNaoaQnlhOUkdXrp"+  

5538 "CeGhOpzdwlvbwN8J44GK44KC44KCCn1lbW9t2b1vce0meOCguOCguOCgppbt21vCeahgwp"+  

5539 "b21vbW8J5qGD44KCCIwsCeOAqouQnlngjgIKPc902Xh0YXJ1YT4K"  

5540 //</span>  

5541  

5542 var SijimiDic = //<span id="gsh-sijimi-dic">  

5543 "data:text/dic;base64," +  

5544 "PG1ldGEgY2hhcnNldD0iVVRLTgiPg08dGV4dGFyZWEgY29scz04MCByb3dzPTQwPg0vL3z1"+  

5545 "cg1HU2lbGcc01NRVxzZGljdGlvbmfYeVxZm9yXHNkbm5ccy8vXHMyMDiWLT4MzAK"+  

5546 "CnTpCeOB1lwpa2Gk44GXCmpCeOBmAptaQnjg1KbmJ44GcmppiceOBmOochp4eXUJ4K4Fp"+  

5547 "CnUJ44GCM5pCeOBwprbwjgMKYnuJ44G2Cm5uceOCkwpubwnjg4KX2hpcEoboQoap0Qnjg"+  

5548 "gaEka2EJ44GLCnJhCeOciQjosLanjgIEKLli444CCnhuW5Hcs4gpw4anV1CeWngQp4bmkJ"+  

5549 "LqMCMtveCa29xCeWaiprb3gj5YCLCm5hbfnqdXuaxgJzKbmFuWYp1dw5peHgj"+  

5550 "77yX77yScm5hbmfpqdXuavg77yX77yScuS4gWhge6JhgNzika29idW5uCeWai+iWhgpo"+  

5551 "atWhcmfxCeOboeBi+OciQp0aWthcmEJ5YqbCmNa0WthcmEJ5YqbCjwvdGv4dGfyZWE+Cg=="  

5552 //</span>  

5553  

5554 var JA_JKLDic = //<span id="gsh-ja-jkl-dic">  

5555 "data:text/dic;base64," +  

5556 "Ly22XjsCU15SU1FamRpI2ptb2JzZWPkQWpKS0wMjAyMGowODE5KSheLV4pL1NhG94SVRT"+  

5557 "Cmtqamprbtq2a2tspsIOS4lueVjApqamtgamwJ44GCCmtqanjgYQK2tqdnjgYYKamtq"+  

5558 "anwJ44GICmtqa2trAnjgYQK2praw2J44GLCmrramtRAnjgYQK2trawmJ44GCPmpamps"+  

5559 "CeOBpkqgampqbAnjg2KmAtpsOBnwpr2prbAnjgKa2pqa2w44GCKmtq2pgbAnjgYaYK"+  

5560 "a2tq2tsCeOBqApmrtsCeOBqgppq2psCeOBqgq2tqAnjg2KmAtpsOBnwpr2prbAnjgKa2traw2J44GChmpq"+  

5561 "a2tq2tsCeOBqApmrtsCeOBqgppq2psCeOBqgq2tqAnjg2KmAtpsOBnwpr2prbAnjgKa2traw2J44GChmpq"+  

5562 "a2tq2tsCeOBqApmrtsCeOBqgppq2psCeOBqgq2tqAnjg2KmAtpsOBnwpr2prbAnjgKa2traw2J44GChmpq"+  

5563 "a2tq2tsCeOBqApmrtsCeOBqgppq2psCeOBqgq2tqAnjg2KmAtpsOBnwpr2prbAnjgKa2traw2J44GChmpq"+  

5564 "a44KEmpr2pgbAnjgjkoKampsCeOciApra2tsCeOciQpamtbsCeOciQpamtbsCeOciQpamtbsCeOciQpamtbs"+  

5565 "a44KCMtq2prbAnjgjkoKampsCeOciApra2tsCeOciQpamtbsCeOciQpamtbsCeOciQpamtbsCeOciQpamtbs"+  

5566 "44KSCmtq2prbAnjgjkoKampsCeOciApra2tsCeOciQpamtbsCeOciQpamtbsCeOciQpamtbsCeOciQpamtbs"+  

5567 "gIEK";  

5568 //</span>  

5569  

5570 //</span>  

5571 /*  

5572 <details id="references"><summary>References</summary><div class="gsh-src">  

5573 <p>  

5574 <a href="https://golang.org">The Go Programming Language</a>  

5575 <!--  

5576 <iframe src="https://golang.org" width="100%" height="300"></iframe>  

5577 -->  

5578  

5579 <a href="https://developer.mozilla.org/ja/docs/Web">MDN web docs</a>  

5580 <a href="https://developer.mozilla.org/ja/docs/Web/HTML/Element">HTML</a>

```

```
5581 CSS:
5582   <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors">Selectors</a>
5583   <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/background-repeat">repeat</a>
5584 HTTP
5585 JavaScript:
5586 ...
5587 </p>
5588 </div></details>
5589 */
5590 /*
5591 <details id="html-src" onclick="frame_open()"><summary>Raw Source</summary><div>
5592 <!-- h2:The full of this HTML including the Go code is here.</h2 -->
5593 <details id="gsh-whole-view"><summary>Whole file</summary>
5594 <a name="whole-src-view"></a>
5595 <span id="src-frame"></span><!-- a window to show source code -->
5596 </details>
5597 </details>
5598 <details id="gsh-style-frame" onclick="fill_CSSView()"><summary>CSS part</summary>
5599 <a name="style-src-view"></a>
5600 <span id="gsh-style-view"></span>
5601 </details>
5602 <details id="gsh-script-frame" onclick="fill_JavaScriptView()"><summary>JavaScript part</summary>
5603 <a name="script-src-view"></a>
5604 <span id="gsh-script-view"></span>
5605 </details>
5606 <details id="gsh-data-frame" onclick="fill_DataView()"><summary>Builtin data part</summary>
5607 <a name="gsh-data-frame"></a>
5608 <span id="gsh-data-view"></span>
5609 </details>
5610 <div id="GshFooter"></div>
5611 </div></details>
5612 */
5613
5614 /**
5615 <!-- 2020-09-17 SatoxITS, visible script -->
5616 <details><summary>GJScript</summary>
5617 <style>.gjscript { font-family:Georgia; }</style>
5618 <pre id="gjscript_1" class="gjscript">
5619   gtest1() { alert('Hello GJScript!'); }
5620   gtest1()
5621 </pre>
5622 <script>
5623   gjs = document.getElementById('gjscript_1');
5624   //eval(gjs.innerHTML);
5625   //gjs.outerHTML = "";
5626 </script>
5627 </details><!-- ----- END-OF-VISIBLE-PART ----- -->
5628 */
5629
5630 /**
5631 <!--
5632 // 2020-0906 added,
5633 https://developer.mozilla.org/en-US/docs/Web/CSS/z-index
5634 https://developer.mozilla.org/en-US/docs/Web/CSS/position
5635 -->
5636 <span id="GshGrid">^_`</span>
5637 <span id="GStat"><br>
5638 </span>
5639 <span id="GMenu" onclick="GShellMenu(this)"></span>
5640 <span id="GTop"></span>
5641 <div id="GshellPlane" onclick="showGShellPlane();"></div>
5642 <div id="RawTextViewer"></div>
5643 <div id="RawTextViewerClose" onclick="hideRawTextViewer()"> CLOSE </div>
5644
5645 <style id="GshStyleDef">
5646   #lineNumbered table,tr,td {
5647     margin:0;
5648     padding:4px;
5649     spacing:0;
5650     border:12px;
5651   }
5652   textarea.LineNumber {
5653     font-size:12px;
5654     font-family:monospace,Courier New;
5655     color:#282;
5656     padding:4px;
5657     text-align:right;
5658   }
5659   textarea.LineNumbered {
5660     font-size:12px;
5661     font-family:monospace,Courier New;
5662     padding:4px;
5663     wrap:off;
5664   }
5665   #RawTextViewer{
5666     z-index:0;
5667     position:fixed; top:0px; left:0px;
5668     width:100%; height:50px;
5669     overflow:auto;
5670     color:#fff; background-color:rgba(128,128,256,0.2);
5671     font-size:12px;
5672     spellcheck:false;
5673   }
5674   #RawTextViewerClose{
5675     z-index:0;
5676     position:fixed; top:-100px; left:-100px;
5677     color:#fff; background-color:rgba(128,128,256,0.2);
5678     font-size:20px; font-family:Georgia;
5679     white-space:pre;
5680   }
5681   #GshellPlane{
5682     z-index:0;
5683     position:fixed; top:0px; left:0px;
5684     width:100%; height:50px;
5685     overflow:auto;
5686     color:#fff; background-color:rgba(128,128,256,0.6);
5687     font-size:12px;
5688   }
5689   #GTop{
5690     z-index:9;
5691     opacity:1.0;
5692     position:fixed; top:0px; left:0px;
5693     width:320px; height:20px;
5694     color:#fff; background-color:rgba(32,32,160,0.2);
5695     color:#fff; font-size:12px;
5696   }
5697   #GPos{
5698     z-index:12;
5699     position:fixed; top:0px; left:0px;
```

```

5705 opacity:1;
5706 width:640px; height:30px;
5707 color:#fff; background-color:rgba(0,0,0,0.4);
5708 cursor:#fff; font-size:12px;
5709 }
5710 #GMenu {
5711 z-index:2000;
5712 position:fixed; top:250px; left:0px;
5713 opacity:1;
5714 width:100px; height:100px;
5715 color:#fff;
5716 color:#fff; background-color:rgba(0,0,0,0.0);
5717 color:#fff; font-size:16px; font-family:Georgia;
5718 background-repeat:no-repeat;
5719 }
5720 #GStat {
5721 z-index:8;
5722 xopacity:0.0;
5723 position:fixed; top:20px; left:0px;
5724 width:640px;
5725 width:100%; height:90px;
5726 color:#fff; background-color:rgba(0,0,128,0.10);
5727 font-size:20px; font-family:Georgia;
5728 }
5729 #GLog {
5730 z-index:10;
5731 position:fixed; top:50px; left:0px;
5732 opacity:1.0;
5733 width:640px; height:60px;
5734 color:#fff; background-color:rgba(0,0,128,0.10);
5735 font-size:12px;
5736 }
5737 #GGrid {
5738 z-index:11;
5739 xopacity:0.0;
5740 position:fixed; top:0px; left:0px;
5741 width:320px; height:30px;
5742 color:#f9; font-size:16px;
5743 }
5744 xbody {display:none;}
5745 gsh-link{color:green;}
5746 gsh {border-width:1; margin:0; padding:0; }
5747 gsh {font-family:monospace,Courier New;color:#ddf;font-size:8px; }
5748 gsh header{height:100px;}
5749 xgsh header{height:100px;background-image:url(GShell-Logo00.png);}
5750 gshMenu{font-size:14pt;color:#44; }
5751 gshMenu{font-size:14pt;color:#2a2;padding:4px; }
5752 gshMenu:hover{font-size:14pt;color:#fff;font-weight:bold;background-color:#2a2; }
5753 gshFooter{height:100px;background-size:80px;background-repeat:no-repeat; }
5754 gsh note{color:#000;font-size:10pt; }
5755 gsh h2{color:#24a;font-family:Georgia;font-size:18pt; }
5756 gsh h3{color:#24a;font-family:Georgia;font-size:16pt; }
5757 gsh details{color:#888;background-color:#fff;font-family:monospace; }
5758 gsh summary{font-size:16pt;color:#fff;background-color:#8af;height:30px; }
5759 gsh pre{font-size:11pt;color:#223;background-color:#faffff; }
5760 gsh a{color:#24a; }
5761 gsh a[name]{color:#24a;font-size:16pt; }
5762 gsh .gsh-src{white-space:pre;font-family:monospace,Courier New;font-size:11pt; }
5763 gsh .gsh-src{background-color:#faffff;color:#223; }
5764 gsh .gsh-src{srcfilecheck:false; }
5765 #src-frame-textarea{white-space:pre;font-family:monospace,Courier New;font-size:11pt; }
5766 #src-frame-textarea{background-color:#faffff;color:#223; }
5767 gsh .code {white-space:pre;font-family:monospace !important; }
5768 gsh .code {color:#088;font-size:11pt; background-color:#eef; }
5769 gsh .golang-data {display:none; }
5770 gsh .WinId {color:#000;font-size:14pt; }

5771 .gsh-document {font-size:11pt;background-color:#fff;font-family:Georgia; }
5772 .gsh-document {color:#000;background-color:#fff !important; }
5773 .gsh-document > h2{color:#000;background-color:#fff !important; }
5774 .gsh-document details{color:#000;background-color:#fff;font-family:Georgia; }
5775 .gsh-document p{max-width:550pt;color:#000;background-color:#fff;font-family:Georgia; }
5776 .gsh-document address{width:500pt;color:#000;background-color:#fff;font-family:Georgia; }

5777 @media print {
5778 #gsh pre{font-size:11pt !important; }
5779 }
5780 </style>
5781
5782 <!--
5783 // Logo image should be drawn by JavaScript from a meta-font.
5784 // CSS seems not follow line-splitted URL
5785 -->
5786 <script id="gsh-data">
5787 //GShellLogo="QR-ITS-more.jp.png"
5788 GShellLogo="data:image/png;base64,
5789 iVBORw0KGgoAAAANSQAABAAQAB/CAYAAADvs3f4AAAAAXNSR0IaRs4c6QAAAhh1WElm
5790 TU0AKAgAAAABgAeAAUAAABAAAABgBaaAAAABAAgEoAAAMAAAABAAIAAIdpAAQAAAAB/
5791 AAAATGAAAAMAAAABIAAAAQAAAAGAAAABAAQoAgQAAAAMAAAQABAAGagAEAAAAAQAAQgAwAE/
5792 AAAQAAAhh8AAAAAyx1BhgAAAAlwsFlzAAleWaACxBMqJcgAAA3RJRFUEaHtInQuOpUNWZ/
5793 ++t+u7k213Cgg0+j6OsBwWmZvun7uB+biSTRy7nQdPCKo+j2Nw1D2MS1rkeuOpAnodCu/
5794 4iuX7jx7iV250DGmF2Vq1BeIsgqCoIMMa+uW/+2M9U1da62aUbv91Gkr3vvxd6/q/
5795 fknXvd7B8A8SIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES/
5796 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES/
5797 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES/
5798 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES/
5799 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES/
5800 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES/
5801 2Ex9S9h+ftSKsdx1ic2qd67Yus+1glaKfnY5ysokhHWEptDk4M0F25u0Ez1LysaU15/
5802 npibLXZEB2C1F53JSuAg9cSqcU6i+2k3StuNy5reEGKJ70w7OvKec2Toq0zWoljhFS/
5803 jbdOVCHStLmrB3U79EJ8h7uBdfmB2+u4xWVVFxbpmeZULame/hcKogAb66EG0Lnykh56PC/
5804 Hx2VBBKoRKhg3Uqe1KyladofJN560kd16w5BwmnnQlyPziOn9DlmXpFx/60p2/Piyov/
5805 N8fm#/+JNWNgjw9KgOTolVGSgt2p2Rilgn3j1v0k7YsowWMzEuwpFPLRkYxfQoak2LrsBQg/
5806 rzWccOG6EhvgRacj/dkt7dXXHgK6NaRS0zpYerqS6RaodZdqfq7K9SKTRXHx/e+9FN/
5807 L66gas8puJwPm1lQgKcs3dPxSr20ur7iwPcc8hbnhNyU1rlyT0QvVf5JfvglB7jx/
5808 +cmHjsqyRy1d7Hy3843d404Tq3xtBThpaFrl0u+w1+cKhnYK5FGEv0WgGAEK0zWmOLy/
5809 rikbd9gqE5V2g0l489FUgA6KjYyFbbQoZJg4zfiesndHCwvUe1Q0b/5CF9Y9d1UueoH/
5810 +zGhUh9SqQomuKw919RpjBd4Y6u0cQd5U0w63zD3MHeS1y49i5bpxghGHLcprFR/
5811 UtJe0aCFT79F5PNB#dHTM0ba74En+eWrW+lZ/Tw60Ad7QJUjp8oA7CooBNBCeMuZ/
5812 tt/cu/co28fLpvKE1LTPFVjusasEhbHvx1guoeBPYfuD04+0feBdy8L4t2zXeSXFMACo/
5813 bgGvg0vlgzGgW4jF392xHnDc+Mwf7J3jfnt2z1yCJBXNU5KIKYcklsxrXldR6dbmcveN/
5814 aJovy/BeMeqvPg4/Elnjz9+jx17V1L5215MytPqlG1NHw5pDQxyNt02Lw82dGmcG22V/
5815 qOfqfjdyvVA02zdafyideFJ35Cs4jZkZhri7e2zm6p3T8hJpkYicJpV1Htk/DJfU4Jw1/
5816 l1mxH51Rf9zKqK4w+K+HQSPE+kBrIyN93EPNTnahsLaLds2xh5Q5NCoPPDvEpgqcbm/8e/
5817 7/zdoHaPtgap/m1K77U0VG0xbvfdx/Er/PifA/17r7ku+coSiCxJwroh1xP16wV9H+ccVg1/
5818 pd/Cfu4ZK2IuPlVtK1L/s5jyE5PVHqr728NzvfUuvP0DGY9GoopuhmLNfCtX48YHL2gH/
5819 f/8hpXu/43qR9qxc6Ytvc1XDC3fMwDQ9nf2le7wK1b0K651cBu0Rqhd18w82dWkPUw/
5820 hrauc6ZCwkcjU2K6UEXMaJ7uQwCu2hnieVi19j/P7ew+iMoAgF+N13iLSfhd9ipai/
5821 NWNP4rPyjxuPdL/Hx2LdwRyUu5wzvSwsVH1m5q9rVvzX9fso4v/LfmqdeIPDGF1m2Uw/
5822 gjYj2wOEPAz3fcfIcvd+zYNcNcJyNtYh9Ac0jRoTUAmRiqOCJRm5WPfTN+frTwdh4siUv/
5823 Bv1WvbhLlRCF04qgK215w14Vq7iPv+RxaJvJw7V7y9+v/CoEMBk168/rPon/
5824 iOzwyDhQ92L2a2oRgQfP5xkh60WvJRKAW46VqU7+RxaJvJw7V7y9+v/CoEMBk168/rPon/
5825 mCufkZ2Lw17yA5i1g5vC3dXIKhsyvzuC9V8GCV/KhchWFDRKAMMcD8EKK+rHF12A9bt2d172/
5826 2qN0zVcdYmfEtN79oxQDnXKA1Q7c0Q2chyADWhnqr5XvtcJsdGp20twgmWJU7A+Eb7/
5827 yhYbgUm17XW7f7KLdwRyUu42fDnVdNbtemL6sX9R26Vtb2awPpx8nfmeh3SD1+mgsolk/
5828 d3/bnBGF1lXGU7f7KLdwRyUu42fDnVdNbtemL6sX9R26Vtb2awPpx8nfmeh3SD1+mgsolk/

```

aup7d/BWtMxYw+jevJKoTksy2d+gFBvBbDw5Lb1R70F+wJybp6U0XGNgR/guta3Bv
Fgeua6qv2d7vnwdf3rd1w3r1dw34aGSP91o9dH95WkXnhkAjaMny16ldk1p2m7d3t3n7wtv5C
h/YrcIpl7Nxp/VvuRduct+wsg54ymm+8zzKoGyRSPr4kOgZl18b6ytacPEmp9b/m9c0UatZ
Jowt6VpnCMcBxz+jsnNhpSjCya6csRsMyWkf41501u0111R7wLnEx3z+2+Gf1Ulu2'
Y57262EakFpfofCjt15jn1p9lydrFzU2p1/3pmkug/Y9ngAc0yGm7hevNx1p6/6ChQph1lh1u
f9u+voGt703g/rfl8xQw2+n7w/F8W6f7wvXsXln1aywv2D21U4/Um1PwMn05uGed9L9'
Zf46cgoxzh76q4nR5Djoxy1vcy1r/FrcJvusLnnkkvCopefPhUCLRvH1v9K4PgngHf6C2'
NCgMS1cNscfKxDeM+tf1BwuxdmD0z7/194253Y3TrzCwpRbH0JyQbhdpanZq
GxWfF66/285bA5hdbpnhUjeGfWY1p0e2hEgmtNDCekTeTxv3Lk4YVqfpuq5WpkXkDa
ufu9mfWiq53nTcx76+33EXQWwVegSpvzr2Ca2fYsy461+04KyvciCug2rpoPyTveJ
u20m12JWZed+b6k0dFLnxw2uX707/b2c5z2p0i+Po1yPdyjCdx34U9X9CHeLs0kt3u9
Acw1k099F2Fn+wGwLds6DdFodXraNeocFrTxuNs0K93pBzXN7w+eVg501/24XngLbCr
5769dGveteB2LWMLVWq5qm2T5p7+5vRLr1zJ0y9h0zBbz2p0
tFu1ilY4+ycFkzqj2p6h1gJbHewLpgJv09/j8//WW3X332gOpV5t1p0b1IDF206p
fs5wyFAhMnd+/Buy+Anu4V7u+eB6E05ic+jot+p8+8f4jy1kTnphq/gSt0Mka18+jrgCl
A4PCxYNpmk0tjRev84HpyOsWs/bScyT2Rg2zrl0g9aBsHbP46hsP2ratm0JeGrubHwdB2Pw'
NTD140SNTBmcnd32/G2/E2B27v7rfUesjyw/7ATguH61y91q3frfqvv9Xq4x+d2+Ueq+yIm5v
bjyj0+ebsL5psq5n62whFwPbHdaUgemy4z1ap2l5b1yA30nTC4F3RKYFtkaU9x9rylw8
sm/H29cGv0TNTV1+1zH2t7rgebb4k+83P35q0oYu/Wy112zdWb2zLxUw+eVg501/24XngLbCr
2GML+6hmravaQwgne1lyz/gLLXBN2C2FQ79Y5QXwq/Qu+H+tr3uAGG1MltG3bFpyEtp
m6d5c9y2CjxMz9Qn2jQa2gbpByMxL9vQ5SbqfBjBpBkbz2m+uevRBRi0t/Ew/B6t0
/9Tcnbs61817dtQn29e5W5f7f1s2oVlyfTlvUtbQd62etccrb011Hes68SYt
202UdegWm7T8757n7gdK19rLzt0pMbk73N4Ayr2df5d+5z9y5dyma0nf1ckwvP0Hvg5FqrOs
wcyW69rU9jk5Mu9QXAm+ePuglw8/fgv61ULPwPsPbSpohiQawEls9gnQxc0eloy1v5
7BBBjdAkhMpDy0/q/ir/wl44t5CnKQWk9q7BsuJh16Lz1C2b+lu2+78FXWYKf104/gY2x
tYv9j8boyN6wzcw/Ojwz7pwt1lpnQK0PdMu1wvooTojJyLxcrNwhHb0jwShyKrpks
2JH14LpikQoxyP6m5f5ksele0G95+wKcJ3m5cmjN5b+1yH2YELGjHmDn/HtmK0S
aP7D14PueAvsDw05jxvXfe/Spef099i9eH42W6q26+CuhV1mtNjHsXfchXo
wKg9z9pHfIcrTrJw/jwh5+/osLzQ1z5B2v1tB7+wpQgXryEcrafrsbzCs5bd/PySxhakPw0
q2x9y410uAB4X5kWe8pHs0+60n+2pxFaUyV6e00175A2xQugzUx29cVyyxx
2CbMhBjAdcrRwkyW4y1kTn9y4m7H9t3R9/8x1j0esWetyy7f1jlo2hkmAfA2ED261nW6h
H52HuW1alQhQOuUzWry6zntL7s/r4y0Bj4jBWCJyRahYtewXyXb/Cw+rsu9y5c03A+w
8v0w0N2ZxAvAPD2CpDxpdsLkD0keFerwEmy+j74a7a7y7xm2/61F2u146c0d60/M
WncfBvtWbS63h3NrxPv1mJh1Ub7KtKbRgaQlCwibuiLpTylKhw9y8zKoqyj19iy9ly
Pwk79u/55B75xSMXchb7w9j35Yx75yv0sYspvBgh5+S5hdjhjnsEsfyvglOzLeRbwj
Ywgk521p10K5dJzgs+jB1L8B46g+uwas6yu1lJzucCoG411gqxQyvOpelyu1xL4pP
z3G6L16V4yaJ35xP1kLmBu/34rcwB6JXGz6f1RbbJh17tLbGDvRd4b1xepPbnH
NQ73iqJ9H7etUvhRx4v54RfpWfR0nDvIge72vBxL2tVYbId3BjBwpMjP
aw3RtYBqgnq9XlmNnChjCnVn5fKm2c1BzLkbJkM54n5c4x/2Lzd0QzNjKtyu01pdcqfcz
gKp/AlfKoov+J7ofimw+jun87F0HmhxMaahuTxE67c707uSuuyqgk03z3v/207c+bch
Ltnpl3t3y8w41pt4jW4Anh75Pnwcjy84Ba1M8bc30qFzLpcJ7cf0sOfBnawDzFqyfbhu
nnldj1iTGHn3esRt+42MK5KwtxF35Me3R7wOrF3m49Vm0gFp0i191X6IdvBxmKqjv
Nfyd9xw8m1l2M1Lz6sEzL+QzSKdpdzcYcete7lq/B4FkRqKAhne3M47/r29fQl/ga/+v/rEo
g/tBtt09uW9bUvrmfUv9H9yLz2vFpxxx0f0/P0/fpredoh/1Xgg2axwP6xGelz+me2x9Lb0
0xu119P0laKqgohaFaNvHxj7X9u0lRmuRjwAfEsfnnakz2LxH2Lb1y5ed1uWkC1/w7d
fd+J17vt2EdPElqK2j6Zp/d5duz+2HxfkLnhb7w0m101A1KkyVScenJw1AIAAC2E
dg2xS8/sN+ln0dPle1xVtd/skrJ+5Ls/9v5LsB7+z+bsN8sQ2oUy/03a1/JFZ3/Z30EgBg
ePdtCYCR0Rkr3g6Lp0lofPx7vfaDvaB7p0t8sr8t7l1uoJ0tqM317xK7qyfVsdWpVqRpAeh07FrV
nnldj1iTGHn3esRt+42MK5KwtxF35Me3R7wOrF3m49Vm0gFp0i191X6IdvBxmKqjv
Nfyd9xw8m1l2M1Lz6sEzL+QzSKdpdzcYcete7lq/B4FkRqKAhne3M47/r29fQl/ga/+v/rEo
g/tBtt09uW9bUvrmfUv9H9yLz2vFpxxx0f0/P0/fpredoh/1Xgg2axwP6xGelz+me2x9Lb0
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5uLm9c8E98N3HgLin151sRn32xsHx0nRtbmVwip97z703
j0q8wn835zeGfYg1yCmlw+vPiCjoxJyv8n7F0HmxhN1/1L7tL6w3j5y8+7j1d7u5y
xLjdjm+x0PQgrucqgUTDvPfcoNwaf2KAEvArG5T3bjBqGt+5rCiU+1BzXp1BjumPvR/
4tYeB9w29y1xwv/OpuyxPw9nPi9y1n/1X9vnsd5dgc88Cwms13C1rKfCQTC2Sh+jw+8q
JXVE3xM6gj5LsR6LbW668/0teExJtj/4Cdw4+2uFvsJrs11RkP0oQALtzNzdf2dS12Q0r
88YSV8pBdsohVQD6xQewRey9j+g9DQPC5mjjy021Lvd7y3zBf1L8g5MsDmOpFtrVWFC31
NINQ9Qw1jEavqM7z2D2vEfmedPcUf8Cn6B5K5f1pRfMHE6F050Atavm/d8vWkm7D
C58YrseFvLspvLph7624erdNy11kL1eJaluk7J00r17+YAC9QcbDf/c7JkHdDb
E5s6909KHMh9pJd6D13v3EvYgbdQucwS1Vm9n0/PAo3K2Ple1w3Cmjkj3K0x3+0RKEkw1N
b1xafeh92JQb8LsF8Gkam65P9MdP5BhmL+3Dc72+R7HskHj1p0kmktcfk1/AM0jS0xtjeK
v7qj/Ozmbz2510ch3+Tnq2pNeuy7x1j0zJD9XoyTCBZD7p7HskHj1p0kmktcfk1/AM0jS0xtjeK
zq4BrggysLmphDlKxcmwLkszBw940as0tB+f7s1410z5md09pdrd+1/27Xb09P9Ec3
xq7X0y8J4TgCmatc0nNe/y99vxbh+17ly1h3b0n1ljf2zNk2zNp6d3n0v3Ry3d0w7hy
h14L693puv1yly+FxC0+Vix1wzY1x7yfzv1l7t1birettsVutMr1jKwPjKmZhdFciM
dhf1f6e16/01/65LmC2D2f12dFyj3ca8AbQSpQ5lcsUcC0kRarD0uzsavgzX6z2Avt
LttGQf1xpj1y7i+hkPbhr+c+h761.0Ld345i+sndv9+y4wCv9g+Sx+6z/arezLrX4W8
tgtzWk74v+28f/PfzZu1k3y5u


```
6077 border:1.2px dashed #282; xborder-radius:2px;
6078 margin:0px; padding:10px !important;
6079 width:340px; height:340px;
6080 flex-wrap: wrap;
6081 color:#ffff; background-color:rgba(0,0,0,0.0);
6082 line-height:0.0;
6083 xxcolor:#22a !important;
6084 text-shadow:2px 2px #ddf;
6085 }
6086 .GJFactory h1,h2,h3,h4 {
6087   xxcolor:#22a !important;
6088 }
6089 input {
6090   border:1px dashed #0f0; border-radius:0px;
6091 }
6092 .GJWin:hover{
6093   color:#df8 !important;
6094   background-color:rgba(32,32,160,0.8) !important;
6095   line-height:0.0;
6096 }
6097 .GJWin:active{
6098   color:#df8 !important;
6099   background-color:rgba(224,32,32,0.8) !important;
6100   line-height:0.0;
6101 }
6102 .GJWin:focus{
6103   color:#df8 !important;
6104   background-color:rgba(32,32,32,1.0) !important;
6105   line-height:0.0;
6106 }
6107 .GJWin{
6108   z-index:10000;
6109   display:inline;
6110   position:relative;
6111   flex-wrap: wrap;
6112   top:0; left:0px;
6113   width:285px !important; height:205px !important;
6114   border:1px solid #eaa; border-radius:2px;
6115   margin:0px; padding:0px;
6116   font-size:8pt;
6117   line-height:0.0;
6118   color:#ffff; background-color:rgba(0,0,64,0.1) !important;
6119 }
6120 .GJTab{
6121   display:inline;
6122   position:relative;
6123   top:0px; left:0px;
6124   margin:0px; padding:2px;
6125   border:0px solid #000; border-radius:2px;
6126   width:90px; height:20px;
6127   font-family:Georgia;
6128   font-size:9pt;
6129   line-height:1.0;
6130   white-space:nowrap;
6131   color:#ffff; background-color:rgba(0,0,64,0.7);
6132   text-align:center;
6133   vertical-align:middle;
6134 }
6135 .GJStat:focus{
6136   color:#df8 !important;
6137   background-color:rgba(32,32,32,1.0) !important;
6138   line-height:1.0;
6139 }
6140 .GJStat{
6141   display:inline;
6142   position:relative;
6143   top:0px; left:0px;
6144   margin:0px; padding:2px;
6145   border:0px solid #00f; border-radius:2px;
6146   width:166px; height:20px;
6147   font-family:monospace;
6148   font-size:9pt;
6149   line-height:1.0;
6150   color:#ffff; background-color:rgba(0,0,64,0.2);
6151   text-align:center;
6152   vertical-align:middle;
6153 }
6154 .GJIcon{
6155   display:inline;
6156   position:relative;
6157   top:0px; left:1px;
6158   border:2px solid #44a;
6159   margin:0px; padding:1px;
6160   width:13.2; height:13.2px;
6161   border-radius:2px;
6162   font-family:Georgia;
6163   font-size:13.2px;
6164   line-height:1.0;
6165   white-space:nowrap;
6166   color:#ffff; background-color:rgba(32,32,160,0.8);
6167   text-align:center;
6168   vertical-align:middle;
6169   text-shadow:0px 0px;
6170 }
6171 .GJText:focus{
6172   color:#fff !important;
6173   background-color:rgba(32,32,160,0.8) !important;
6174   line-height:1.0;
6175 }
6176 .GJText{
6177   display:inline;
6178   position:relative;
6179   top:0px; left:0px;
6180   border:0px solid #000; margin:0px; padding:0px;
6181   width:280px; height:160px;
6182   border:0px;
6183   font-family:Courier New,monospace !important;
6184   font-size:8pt;
6185   line-height:1.0;
6186   white-space:pre;
6187   color:#fff; xbackground-color:rgba(0,0,64,0.5);
6188   background-color:rgba(32,32,128,0.8) !important;
6189 }
6190 .GJMode{
6191   display:inline;
6192   position:relative;
6193   top:0px; left:0px;
6194   border:0px solid #000; border-radius:0px;
6195   margin:0px; padding:0px;
6196   width:280px; height:20px;
6197   font-size:9pt;
6198   line-height:1.0;
6199   white-space:nowrap;
6200   color:#fff; background-color:rgba(0,0,64,0.7);
```

```
6201     text-align:left;
6202     vertical-align:middle;
6203   }
6204 </style>
6205 <script id="gsh-script">
6206 // 2020-0909 added, permanet local storage
6207 // https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
6208 var MyHistory = ""
6209 Permanent = localStorage;
6210 MyHistory = Permanent.getItem('MyHistory')
6211 if( MyHistory == null ){ MyHistory = "" }
6212 d = new Date()
6213 MyHistory = d.getTime()/1000+" "+document.URL+"\n" + MyHistory
6214 Permanent.setItem('MyHistory',MyHistory)
6215 //Permanent.setItem('MyWindow',window)
6216
6217 var GJLog_Win = null
6218 var GJLog_Tab = null
6219 var GJLog_Stat = null
6220 var GJLog_Text = null
6221 var GJWin_Mode = null
6222
6223 var FProductInterval = 0
6224
6225 var GJ_FactoryID = -1
6226 var GJFactory = null
6227 if( e = document.getElementById('GJFactory_0') ){
6228   GJFactory_1.height = 0
6229   GJFactory = e
6230   e.setAttribute('class','GJFactory')
6231   var GJ_FactoryID = 0
6232 }else{
6233   GJFactory = GJFactory_1
6234   var GJ_FactoryID = 1
6235 }
6236
6237 function GJFactory_Destroy(){
6238   gif = GJFactory
6239   //gif = document.getElementById('GJFactory')
6240   //alert('gif='+gif)
6241   if( gif != null ){
6242     if( gif.childNodes != null ){
6243       for( i = 0; i < gif.childNodes.length; i++ ){
6244         gif.removeChild(gif.childNodes[i])
6245       }
6246     }
6247     gif.innerHTML = ''
6248     gif.style.width = 0
6249     gif.style.height = 0
6250     gif.removeAttribute('style')
6251     GJLog_Win = GJLog_Tab = GJLog_Stat = GJLog_Text = GJWin_Mode = null
6252     window.clearInterval(FProductInterval)
6253     return '-- Destroy: work product destroyed'
6254   }else{
6255     return '-- Destroy: work product not exist'
6256   }
6257 }
6258
6259 var TransMode = false
6260 var OnKeyControl = false
6261 var OnKeyShift = false
6262 var OnKeyAlt = false
6263 var OnKeyJ = false
6264 var OnKeyK = false
6265 var OnKeyL = false
6266
6267 function GJWin_OnKeyUp(ev){
6268   keycode = ev.code;
6269   if( keycode == 'ShiftLeft' ){
6270     OnKeyShift = false
6271   }else
6272     if( keycode == 'ControlLeft' ){
6273       OnKeyControl = false
6274     }else
6275       if( keycode == 'AltLeft' ){
6276         OnKeyAlt = false
6277       }else
6278         if( keycode == 'KeyJ' ){ OnKeyJ = false }else
6279           if( keycode == 'KeyK' ){ OnKeyK = false }else
6280             if( keycode == 'KeyL' ){ OnKeyL = false }else
6281             {
6282             }
6283   ev.preventDefault()
6284 }
6285 function and(a,b){ if(a){ if(b){ return true; } return false; } }
6286 function GJWin_OnKeyDown(ev){
6287   keycode = ev.code;
6288   mode = ''
6289   key = ''
6290   if( keycode == 'ControlLeft' ){
6291     onKeyControl = true
6292     ev.preventDefault()
6293     return;
6294   }else
6295     if( keycode == 'ShiftLeft' ){
6296       OnKeyShift = true
6297       ev.preventDefault()
6298       return;
6299     }else
6300       if( keycode == 'AltLeft' ){
6301         ev.preventDefault()
6302         OnKeyAlt = true
6303         return;
6304       }else
6305         if( keycode == 'Backquote' ){
6306           TransMode = !TransMode
6307           ev.preventDefault()
6308         }else
6309           if( and(keycode == 'Space', OnKeyShift ) ){
6310             TransMode = !TransMode
6311             ev.preventDefault()
6312           }else
6313             if( keycode == 'ShiftRight' ){
6314               TransMode = !TransMode
6315             }else
6316               if( keycode == 'Escape' ){
6317                 TransMode = true
6318                 ev.preventDefault()
6319               }else
6320                 if( keycode == 'Enter' ){
6321                   TransMode = false
6322                   //ev.preventDefault()
6323                 }
6324   if( keycode == 'KeyJ' ){ OnKeyJ = true }else
```

```

6325     if( keycode == 'KeyK' ){ OnKeyK = true }else
6326     if( keycode == 'KeyL' ){ OnKeyL = true }else
6327     {
6328     }
6329
6330     if( ev.altKey ){ key += 'Alt+' }
6331     if( onKeyControl ){ key += 'Ctrl+' }
6332     if( onKeyShift ){ key += 'Shift+' }
6333     if( and(keycode != 'KeyJ', OnKeyJ) ){ key += 'J+' }
6334     if( and(keycode != 'KeyK', OnKeyK) ){ key += 'K+' }
6335     if( and(keycode != 'KeyL', OnKeyL) ){ key += 'L+' }
6336     key += keycode
6337
6338     if( TransMode ){
6339         //mode = "[343\201\202r]"
6340         mode = "[ðr]"
6341     }else{
6342         mode = '[---]'
6343     }
6344     ////  /gjmode.innerHTML = "[---]"
6345     GJWin_Mode.innerHTML = mode + ' ' + key
6346     //alert('Key:' +keycode)
6347     ev.stopPropagation()
6348     //ev.preventDefault()
6349 }
6350 function GJWin_Onscroll(ev){
6351     x = DragStartX = gsh.getBoundingClientRect().left.toFixed(0)
6352     y = DragStatty = gsh.getBoundingClientRect().top.toFixed(0)
6353     GJLog_append('OnScroll: x=' +x+ ',y=' +y)
6354 }
6355 document.addEventListener('scroll',GJWin_Onscroll)
6356 function GJWin_Onresize(ev){
6357     w = window.innerWidth
6358     h = window.innerHeight
6359     GJLog_append('OnResize: w=' +w+',h=' +h)
6360 }
6361 window.addEventListener('resize',GJWin_OnResize)
6362
6363 var DragStartX = 0
6364 var DragStartY = 0
6365 function GJWin_DragStart(ev){
6366     // maybe this is the grabbing position
6367     this.style.position = 'fixed'
6368     x = DragStartX = this.getBoundingClientRect().left.toFixed(0)
6369     y = DragStartY = this.getBoundingClientRect().top.toFixed(0)
6370     GJLog_Stat.value = 'DragStart: x=' +x+ ',y=' +y
6371 }
6372 function GJWin_Drag(ev){
6373     x = ev.clientX; y = ev.clientY // x = ev.pageX; y = ev.pageY
6374     this.style.left = x - DragStartX
6375     this.style.top = y - DragStartY
6376     this.style.zIndex = '30000'
6377     this.style.position = 'fixed'
6378     x = this.getBoundingClientRect().left.toFixed(0)
6379     y = this.getBoundingClientRect().top.toFixed(0)
6380     GJLog_Stat.value = 'x=' +x+ ',y=' +y
6381     ev.preventDefault()
6382     ev.stopPropagation()
6383 }
6384 function GJWin_DragEnd(ev){
6385     x = ev.clientX; y = ev.clientY
6386     //x = ev.pageX; y = ev.pageY
6387     this.style.left = x - DragStartX
6388     this.style.top = y - DragStartY
6389     this.style.zIndex = '30000'
6390     this.style.position = 'fixed'
6391     if( true ){
6392         console.log("Dropped: "+this.nodeName+'#'+this.id+' x=' +x+ ' y=' +y
6393         +' parent=' +this.parentNode.id)
6394     }
6395     x = this.getBoundingClientRect().left.toFixed(0)
6396     y = this.getBoundingClientRect().top.toFixed(0)
6397     GJLog_Stat.value = 'x=' +x+ ',y=' +y
6398     ev.preventDefault()
6399     ev.stopPropagation()
6400 }
6401 function GJWin_DragIgnore(ev){
6402     ev.preventDefault()
6403     ev.stopPropagation()
6404 }
// 2020-09-15 let every object have console view!
6405 var GJ_ConsoleID = 0
6406 var PrevReport = new Date()
6407 function GJLog_StatUpdate(){
6408     tx1 = GJLog_Stat;
6409     if( tx1 == null ){
6410         return;
6411     }
6412     tmLap0 = new Date();
6413     p = tx1.parentNode;
6414     pw = tx1.getBoundingClientRect().width;
6415     ph = tx1.getBoundingClientRect().height;
6416     //txa.value += '#'+p.id+ ' pw=' +pw+, ph=' +ph+'\n';
6417     tx1 = '#'+p.id+ ' pw=' +pw+, ph=' +ph+'\n';
6418
6419     w = tx1.getBoundingClientRect().width;
6420     h = tx1.getBoundingClientRect().height;
6421     //txa.value += 'w=' +w+, h=' +h+'\n';
6422     tx1 += 'w=' +w+, h=' +h+'\n';
6423
6424     //txa.value += '\n';
6425     //txa.value += DateShort() + '\n';
6426     tx1 += '\n';
6427     tx1 += DateShort() + '\n';
6428     tmLap1 = new Date();
6429
6430     tx1.value += tx1;
6431     tmLap2 = new Date();
6432
6433     // vertical centering of the last line
6434     sHeight = tx1.scrollHeight - 30; // depends on the font-size
6435     tmLap3 = new Date();
6436
6437     tx1.scrollTop = sHeight; // depends on the font-size
6438     tmLap4 = new Date();
6439
6440     now = tmLap0.getTime();
6441     if( PrevReport == 0 || 10000 <= now-PrevReport ){
6442         PrevReport = now;
6443         console.log('StatusBarUpdate:'
6444             + 'length=' + tx1.value.length + ' byte, '
6445             + 'times=' + (tmLap4-tmLap0) + 'ms {'
6446             + 'tadds=' + (tmLap2-tmLap1) + ', '
6447             + 'hcal=' + (tmLap3-tmLap2) + ', '
6448

```

```

6449         + 'scrl=' + (tmLap4 - tmLap3) + '}';
6450     );
6451   }
6452 }
6453 GJWin_StatUpdate = GJLog_StatUpdate;
6454 function GJ_showTime1(wid){
6455   //e = document.getElementById(wid);
6456   //console.log(wid.id+' .value.length=' +wid.value.length)
6457   if( e != null ){
6458     //e.value = DateShort();
6459   }else{
6460     // should remove the Listener
6461   }
6462 }
6463 function GJWin_OnResizeTextarea(ev){
6464   this.value += 'resized:' + '\n'
6465 }
6466 function GJ_NewConsole(wname){
6467   wid = wname + '_' + GJ_ConsoleID
6468   GJ_ConsoleID += 1
6469 }
6470 GJFactory.style.setProperty('width',360+'px'); //GJFsize
6471 GJFactory.style.setProperty('height',320+'px')
6472 e = GJFactory;
6473 console.log('GJFa #' + e.id + ' from w=' + e.style.width + ', h=' + e.style.height)
6474
6475 if( GJFactory.innerHTML == "" ){
6476   GJFactory.innerHTML = '<'+H3>GJ Factory_'+ GJ_FactoryID +'<'+H3><'+hr>\n'
6477 }else{
6478   GJFactory.innerHTML += '<'+hr>\n'
6479 }
6480
6481 gjwin = GJLog_Win = document.createElement('span')
6482 gjwin.id = wid
6483 gjwin.setAttribute('class','GJWin')
6484 gjwin.setAttribute('draggable','true')
6485 gjwin.addEventListener('dragstart',GJWin_DragStart)
6486 gjwin.addEventListener('drag',GJWin_Drag)
6487 gjwin.addEventListener('dragend',GJWin_Drag)
6488 gjwin.addEventListener('dragover',GJWin_DragIgnore)
6489 gjwin.addEventListener('dragenter',GJWin_DragIgnore)
6490 gjwin.addEventListener('dragleave',GJWin_DragIgnore)
6491 gjwin.addEventListener('dragexit',GJWin_DragIgnore)
6492 gjwin.addEventListener('drop',GJWin_DragIgnore)
6493 gjwin.addEventListener('keydown',GJWin_OnKeyDown)
6494
6495 gjtab = GJLog_Tab = document.createElement('textare')
6496 gjtab.addEventListener('keydown',GJWin_OnKeyDown)
6497 gjtab.style.readonly = true
6498 gjtab.contentEditable = false
6499 gjtab.value = wid
6500 gjtab.id = wid + ' Tab'
6501 gjtab.setAttribute('class','GJTab')
6502 gjtab.setAttribute('spellcheck','false')
6503 gjwin.appendChild(gjtab)
6504
6505 gjstat = GJLog_Stat = document.createElement('textare')
6506 gjstat.addEventListener('keydown',GJWin_OnKeyDown)
6507 gjstat.id = wid + '_Stat'
6508 gjstat.value = DateShort()
6509 gjstat.setAttribute('class','GJStat')
6510 gjstat.setAttribute('spellcheck','false')
6511 gjwin.appendChild(gjstat)
6512
6513 gjicon = document.createElement('span')
6514 gjicon.addEventListener('keydown',GJWin_OnKeyDown)
6515 gjicon.id = wid + '_Icon'
6516 gjicon.innerHTML = '<font color="#ff44>J</font>'
6517 gjicon.setAttribute('class','GJIcon')
6518 gjicon.setAttribute('spellcheck','false')
6519 gjwin.appendChild(gjicon)
6520
6521 gjtext = GJLog_Text = document.createElement('textare')
6522 gjtext.addEventListener('keydown',GJWin_OnKeyDown)
6523 gjtext.addEventListener('keyup',GJWin_OnKeyUp)
6524 gjtext.addEventListener('resize',GJWin_OnResizeTextarea)
6525 gjtext.id = wid + '_Text'
6526 gjtext.setAttribute('class','GJText')
6527 gjtext.setAttribute('spellcheck','false')
6528 gjwin.appendChild(gjtext)
6529
6530
6531 // user's mode as of IME
6532 gjmode = GJWin_Mode = document.createElement('textare')
6533 gjmode.addEventListener('keydown',GJWin_OnKeyDown)
6534 gjmode.addEventListener('keydown',GJWin_OnKeyDown)
6535 gjmode.id = wid + '_Mode'
6536 gjmode.setAttribute('class','GJMode')
6537 gjmode.setAttribute('spellcheck','false')
6538 gjmode.innerHTML = '[---]'
6539 gjwin.appendChild(gjmode)
6540
6541 gjwin.zIndex = 30000
6542 GJFactory.appendChild(gjwin)
6543
6544 gjtab.scrollTop = 0
6545 gjstat.scrollTop = 0
6546
6547 //x = gjwin.getBoundingClientRect().left.toFixed(0)
6548 //y = gjwin.getBoundingClientRect().top.toFixed(0)
6549 //gjwin.style.position = 'static'
6550 //gjwin.style.left = 0
6551 //gjwin.style.top = 0
6552
6553 //update = ('+' +wid+'.value=DateShort()');
6554 update = '(GJ_showTime1('+wid+'))';
6555 // 2020-09-19 this causes memory leaks
6556 //FPProductInterval = window.setInterval(update,200)
6557 //FPProductInterval = window.setInterval(GJWin_StatUpdate, 200)
6558 //FPProductInterval = window.setInterval(GJ_showTime1,200,wid);
6559 FPProductInterval = window.setInterval(GJ_showTime1,200,gjstat);
6560 return update
6561 }
6562 function xxxGJF_StripClass(){
6563   GJLog_Win.style.removeProperty('width')
6564   GJLog_Tab.style.removeProperty('width')
6565   GJLog_Stat.style.removeProperty('width')
6566   GJLog_Text.style.removeProperty('width')
6567   return "Stripped classes"
6568 }
6569 function isElem(id){
6570   return document.getElementById(id) != null
6571 }
6572 function GJLog_append(...args){

```

```

6573     txt = GJLog_Text;
6574     if( txt == null ){
6575         return; // maybe GJLog element is removed
6576     }
6577     logs = args.join(' ')
6578     txt.value += logs + '\n'
6579     txt.scrollTop = txt.scrollHeight
6580     //GJLog_Stat.value = DateShort()
6581 }
6582 //window.addEventListener('time',GJLog_StatUpdate)
6583 window.setInterval(GJLog_StatUpdate,1000);
6584 GJ_NewConsole('GJ_Console')
6585
6586 e = GJFactory;
6587 console.log('GJF0 #' + e.id + ' from w=' + e.style.width + ', h=' + e.style.height)
6588 e.style.width = 360; //GJFsize
6589 e.style.height = 320;
6590 console.log('GJF0 #' + e.id + ' to w=' + e.style.width + ', h=' + e.style.height)
6591
6592 var StopConsoleLog = true
6593 // 2020-09-15 added,
6594 // log should be saved to permanent memory
6595 // const px = new Proxy(console.log,{ alert() })
6596 __console_log = console.log
6597 __console_info = console.info
6598 __console_warn = console.warn
6599 __console_error = console.error
6600 __console_exception = console.exception
6601 // should pop callstack info.
6602 console.exception = function(...args){
6603     __console_exception(...args)
6604     alert('-- got console.exception("' + args + '")')
6605 }
6606 console.error = function(...args){
6607     __console_error(...args)
6608     alert('-- got console.error("' + args + '")')
6609 }
6610 console.warn = function(...args){
6611     __console_warn(...args)
6612     alert('-- got console.warn("' + args + '")')
6613 }
6614 console.info = function(...args){
6615     alert('-- got console.info("' + args + '")')
6616     __console_info(...args)
6617 }
6618 console.log = function(...args){
6619     __console_log(...args)
6620     if( StopConsoleLog ){
6621         return;
6622     }
6623     if( 0 <= args[0].indexOf('!') ){
6624         //alert('-- got console.log("' + args + '")')
6625     }
6626     GJLog_append(...args)
6627 }
6628 console.log('Hello, GJShell!')
6629
6630 //document.getElementById('GshFaviconURL').href = GShellFavicon
6631 document.getElementById('GshFaviconURL').href = GShellInsideIcon
6632 //document.getElementById('GshFaviconURL').href = ITSmoreQR
6633 //document.getElementById('GshFaviconURL').href = GSellLogo
6634
6635 // id of GShell HTML elements
6636 var E_BANNER = "GshBanner" // banner element in HTML
6637 var E_FOOTER = "GshFooter" // footer element in HTML
6638 var E_GINDEX = "gsh-gindex" // index of Golang code of GShell
6639 var E_GOCODE = "gsh-gocode" // Golang code of GShell
6640 var E_TODO = "gsh-todo" // TODO of GShell
6641 var E_DICT = "gsh-dict" // Dictionary of GShell
6642
6643 function bannerElem(){ return document.getElementById(E_BANNER); }
6644 function bannerStyleFunc(){ return bannerElem().style; }
6645 var bannerStyle = bannerStyleFunc()
6646 bannerStyle.backgroundImage = "url(\"+GSellLogo+)";
6647 //bannerStyle.backgroundImage = "url(\"+GShellInsideIcon+)";
6648 //bannerStyle.backgroundImage = "url(\"+GShellFavicon+)";
6649 GMenu.style.backgroundImage = "url(\"+GShellInsideIcon+)";
6650
6651 function footerElem(){ return document.getElementById(E_FOOTER); }
6652 function footerStyle(){ return footerElem().style; }
6653 footerElem().style.backgroundImage="url(\"+ITSmoreQR+)";
6654 //footerStyle().backgroundImage = "url(\"+ITSmoreQR+)";
6655
6656 function html_fold(e){
6657     if( e.innerHTML == "Fold" ){
6658         e.innerHTML = "Unfold"
6659         document.getElementById('gsh-menu-exit').innerHTML=""
6660         document.getElementById('GshStatement').open=false
6661         GshFeatures.open = false
6662         document.getElementById('html-src').open=false
6663         document.getElementById(E_GINDEX).open=false
6664         document.getElementById(E_GOCODE).open=false
6665         document.getElementById(E_TODO).open=false
6666         document.getElementById('references').open=false
6667     }else{
6668         e.innerHTML = "Fold"
6669         document.getElementById('GshStatement').open=true
6670         GshFeatures.open = true
6671         document.getElementById(E_GINDEX).open=true
6672         document.getElementById(E_GOCODE).open=true
6673         document.getElementById(E_TODO).open=true
6674         document.getElementById('references').open=true
6675     }
6676 }
6677 function html_pure(e){
6678     if( e.innerHTML == "Pure" ){
6679         document.getElementById('gsh').style.display=true
6680         //document.style.display = false
6681         e.innerHTML = "Unpure"
6682     }else{
6683         document.getElementById('gsh').style.display=false
6684         //document.style.display = true
6685         e.innerHTML = "Pure"
6686     }
6687 }
6688
6689 var bannerIsStopping = false
6690 //NOTE: .com/JSRREF/prop_style_backgroundposition.asp
6691 function shiftBG(){
6692     bannerIsStopping = !bannerIsStopping
6693     bannerStyle.backgroundPosition = "0 0";
6694 }
6695 // status should be inherited on Window Fork(), so use the status in DOM
6696 function html_stop(e,toggle){

```

```

6697     if( toggle ){
6698         if( e.innerHTML == "Stop" ){
6699             bannerIsStopping = true
6700             e.innerHTML = "Start"
6701         }else{
6702             bannerIsStopping = false
6703             e.innerHTML = "Stop"
6704         }
6705     }else{
6706         // update JavaScript variable from DOM status
6707         if( e.innerHTML == "Stop" ){ // shown if it's running
6708             bannerIsStopping = false
6709         }else{
6710             bannerIsStopping = true
6711         }
6712     }
6713 }
6714 html_stop(document.getElementById('GshMenuStop'),false) // onInit.
6715 //html_stop(bannerElem(),false) // onInit.
6716
6717 //https://www.w3schools.com/jsref/met_win_setinterval.asp
6718 function shiftBanner(){
6719     var now = new Date().getTime();
6720     //"console.log("now"+(now%10))
6721     if( !bannerIsStopping ){
6722         bannerStyle.backgroundPosition = ((now/10)%100000)+" 0";
6723     }
6724 }
6725 window.setInterval(shiftBanner,10); // onInit.
6726
6727 // <a href="https://developer.mozilla.org/ja/docs/Web/API/Window/open">window.open()</a>
6728 // from embedded html to standalone page
6729 var MyChildren = 0
6730 function html_fork(){
6731     GJFactory_Destroy()
6732     MyChildren += 1
6733     WinId = document.getElementById('gsh-WinId').innerHTML + "." + MyChildren;
6734     newwin = window.open("",WinId,"");
6735     src = document.getElementById('gsh');
6736     srchtml = src.outerHTML
6737     newwin.document.write('/<'+srchtml+'\n');
6738     newwin.document.write(srchtml);
6739     newwin.document.write("<"+"/html>\n");
6740     newwin.document.getElementById('gsh-menu-exit').innerHTML = "Close";
6741     newwin.document.getElementById('gsh-WinId').innerHTML = WinId;
6742     newwin.document.close();
6743     newwin.focus();
6744 }
6745 function html_close(){
6746     window.close()
6747 }
6748 function win_jump(win){
6749     //win = window.top;
6750     win = window.opener; // https://developer.mozilla.org/ja/docs/Web/API/window.opener
6751     if( win == null ){
6752         console.log("jump to window.opener(\"+win+)\")\n")
6753     }else{
6754         console.log("jump to window.opener(\"+win+)\")\n")
6755         win.focus();
6756     }
6757 }
6758
6759 // 0.2.9 2020-0902 created checksum of HTML
6760 CRC32UNIX = 0x04C1DB7 // Unix cksum
6761 function byteCRC32add(bigcrc,octstr,octlen){
6762     var crc = new Int32Array(1)
6763     crc[0] = bigcrc
6764
6765     let oi = 0
6766     for( ; oi < octlen; oi++ ){
6767         var oct = new Int8Array(1)
6768         oct[0] = octstr[oi]
6769         for( bi = 0; bi < 8; bi++ ){
6770             //console.log("--CRC32 "+crc[0]+""+oct[0].toString(16)+" ["+oi+"."+bi+"]\n")
6771             ovf1 = crc[0] < 0 ? 1 : 0
6772             ovf2 = oct[0] < 0 ? 1 : 0
6773             ovf = ovf1 ^ ovf2
6774             oct[0] <= 1
6775             crc[0] <= 1
6776             if( ovf ){ crc[0] ^= CRC32UNIX }
6777         }
6778     }
6779     //console.log("--CRC32 byteAdd return crc="+crc[0]+","+"oi+"/"+octlen+"\n")
6780     return crc[0];
6781 }
6782 function strCRC32add(bigcrc,str,strlen){
6783     var crc = new Uint32Array(1)
6784     crc[0] = bigcrc
6785     var code = new Uint8Array(strlen);
6786     for( i = 0; i < strlen; i++){
6787         code[i] = str.charCodeAt(i) // not charAt() !!!!
6788         //console.log("== "+code[i].toString(16)+" <== "+str[i]+\n")
6789     }
6790     crc[0] = byteCRC32add(crc,code,strlen)
6791     //console.log("--CRC32 strAdd return crc="+crc[0]+\n")
6792     return crc[0];
6793 }
6794 function byteCRC32end(bigcrc,len){
6795     var crc = new Uint32Array(1)
6796     crc[0] = bigcrc
6797     var slen = new Uint8Array(4)
6798     let li = 0
6799     for( ; li < 4; ){
6800         selen[li] = len
6801         li += 1
6802         len >= 8
6803         if( len == 0 ){
6804             break
6805         }
6806     }
6807     crc[0] = byteCRC32add(crc[0],slen,li)
6808     crc[0] ^= 0xFFFFFFFF
6809     return crc[0];
6810 }
6811 function strCRC32(stri,len){
6812     var crc = new Uint32Array(1)
6813     crc[0] = 0
6814     crc[0] = strCRC32add(0,stri,len)
6815     crc[0] = byteCRC32end(crc[0],len)
6816     //console.log("--CRC32 "+crc[0]+""+len+"\n")
6817     return crc[0];
6818 }
6819 function getSourceText(){
6820     version = document.getElementById('GshVersion').innerHTML

```

```

6821 sfavico = document.getElementById('GshFaviconURL').href;
6822 sbanner = document.getElementById('GshBanner').style.backgroundImage;
6823 spositi = document.getElementById('GshBanner').style.backgroundPosition;
6824 sfooter = document.getElementById('GshFooter').style.backgroundImage;
6825
6826 if( document.getElementById('GJC_1') != null ){ GJC_1.remove() }
6827
6828 // these should be removed by CSS selector or class, after seavaed to non-printed attribute
6829 GshBanner.removeAttribute('style');
6830 GshFooter.removeAttribute('style');
6831 document.getElementById('GshMenuSign').removeAttribute("style");
6832 styleGMenu = GMenu.getAttribute("style")
6833 GMENU.removeAttribute("style");
6834 styleGStat = GStat.getAttribute("style")
6835 GStat.removeAttribute("style");
6836 GTop = GTop.getAttribute("style")
6837 GTop.removeAttribute("style");
6838 styleGshGrid = GshGrid.getAttribute("style")
6839 GshGrid.removeAttribute("style");
6840 //styleGPos = GPos.getAttribute("style");
6841 //GPos.removeAttribute("style");
6842 //GPos.innerHTML = "";
6843 //styleGLog = GLog.getAttribute("style");
6844 //GLog.removeAttribute("style");
6845 //GLog.innerHTML = "";
6846 styleGshellPlane = GshellPlane.getAttribute("style")
6847 GshellPlane.removeAttribute("style")
6848 styleRawTextViewer = RawTextViewer.getAttribute("style")
6849 RawTextViewer.removeAttribute("style")
6850 styleRawTextViewerClose = RawTextViewerClose.getAttribute("style")
6851 RawTextViewerClose.removeAttribute("style")
6852
6853 GshFaviconURL.href = "";
6854
6855 //it seems that interHTML and outerHTML generate style="" for these (??)
6856 //GshBanner.removeAttribute('style');
6857 //GshFooter.removeAttribute('style');
6858 //GshMenuSign.removeAttribute('style');
6859 GshBanner.style="";
6860 GshFooter.style="";
6861 GshMenuSign.style="";
6862
6863 textarea = document.createElement("textarea")
6864 srchtml = document.getElementById("gsh").outerHTML;
6865 //textarea = document.createElement("textarea")
6866 // 2020-0910 ?? ... this causes inserting style="" to Banner and Footer,
6867 // with Chromium?/ after reloading from file:///
6868 textarea.innerHTML = srchtml
6869 // <a href="https://stackoverflow.com/questions/5796718/html-entity-decode">Thanks</a>
6870 var rawtext = textarea.value
6871 //textarea.destroy()
6872 //rawtext = gsh.textContent // this removes #include <FILENAME> too
6873 var orgtext = ""
6874 + /*<html>\n" // lost preamble text
6875 + rawtext
6876 + "<"</html>\n" // lost trail text
6877 ;
6878
6879 tlen = orgtext.length
6880 //console.log("getSourceText: length='"+tlen+"\n")
6881 document.getElementById('GshFaviconURL').href = sfavico;
6882
6883 document.getElementById('GshBanner').style.backgroundImage = sbanner;
6884 document.getElementById('GshBanner').style.backgroundPosition = spositi;
6885 document.getElementById('GshFooter').style.backgroundImage = sfooter;
6886
6887 GStat.setAttribute("style",styleGStat)
6888 GMENU.setAttribute("style",styleGMENU)
6889 GTop.setAttribute("style",styleGTop)
6890 //GLog.setAttribute("style",styleGLog)
6891 //GPos.setAttribute("style",styleGPos)
6892 GshGrid.setAttribute("style",styleGshGrid)
6893 GshellPlane.setAttribute("style",styleGshellPlane)
6894 RawTextViewer.setAttribute("style",styleRawTextViewer)
6895 RawTextViewerClose.setAttribute("style",styleRawTextViewerClose)
6896 canontext = orgtext.replace(' style=""','');
6897 // open="" too
6898 return canontext
6899 }
6900 function getDigest(){
6901 var text = ""
6902 text = getSourceText()
6903 var digest = ""
6904 tlen = text.length
6905 digest = strCRC32(text,tlen) + " " + tlen
6906 return { text, digest }
6907 }
6908 function html_digest(){
6909 version = document.getElementById('GshVersion').innerHTML
6910 let {text, digest} = getDigest()
6911 alert("cksum: " + digest + " " + version)
6912 }
6913 function charsin(stri,char){
6914 ln = 0;
6915 for( i = 0; i < stri.length; i++ ){
6916 if( stri.charCodeAt(i) == char.charCodeAt(0) )
6917 ln++;
6918 }
6919 return ln;
6920 }
6921
6922 //class digestElement extends HTMLElement { }
6923 //<script>customElements.define('digest',digestElement)</script>
6924 function showDigest(e){
6925 result = 'version=' + GshVersion.innerHTML + '\n'
6926 result += 'lines=' + e.dataset.lines + '\n'
6927 + 'length=' + e.dataset.length + '\n'
6928 + 'crc32u=' + e.dataset.crc32u + '\n'
6929 + 'time=' + e.dataset.time + '\n';
6930
6931 alert(result)
6932 }
6933
6934 function html_sign(e){
6935 if( RawTextViewer.style.zIndex == 1000 ){
6936 hideRawTextViewer()
6937 return
6938 }
6939 GJFactory_Destroy()
6940 //gsh_digest_.innerHTML = "";
6941 text = getSourceText() // the original text
6942 tlen = text.length
6943 digest = strCRC32(text,tlen)
6944 //gsh_digest_.innerHTML = digest + " " + tlen

```

```

6945 //text = getSourceText() // the text with its digest
6946 Lines = charsin(text,'`n')
6947
6948 name = "gsh"
6949 sid = name + "-digest"
6950 d = new Date()
6951 signedAt = d.getTime()
6952
6953 sign = '/'+*`+'+`span`n'
6954     + ` id=` + sid + `n`
6955     + ` class="digest"``n'
6956     + ` data-target-id=`+name+``n'
6957     + ` data-crc32u=`` + digest + `n'
6958     + ` data-length=`` + tlen + `n'
6959     + ` data-lines=`` + Lines + `n'
6960     + ` data-times=`` + signedAt + `n'
6961     + ` ><` + `/span>`n*`+`n'
6962
6963 text = sign + text
6964
6965 txthtml = '<' + `table id="LineNumbered">><' + `tr><' + `td>'
6966     + `<' + `textarea cols=5 rows=' + Lines + ` class="LineNumber">>'
6967 for( i = 1; i <= Lines; i++){
6968     txthtml += i.toString() + `n'
6969 }
6970 txthtml += `n
6971     + `<' + `/textarea`+
6972     + `<' + `/td><' + `/td`+
6973     + `<' + `textarea cols=150 rows=' + Lines + ` spellcheck="false"``+
6974     + ` class="LineNumber"``+
6975     + text + `<' + `/textarea`+
6976     + `<' + `/td><' + `/tr><' + `/table>`+
6977
6978 for( i = 1; i <= 30; i++ ){
6979     txthml += `n`+
6980 }
6981 RawTextViewer.innerHTML = txthml
6982
6983 btn = e
6984 e.style.color = "rgba(128,128,255,0.9)";
6985 y = e.getBoundingClientRect().top.toFixed(0)
6986 //h = e.getBoundingClientRect().height.toFixed(0)
6987 RawTextViewer.style.top = Number(y) + 30
6988 RawTextViewer.style.height = window.innerHeight - 20;
6989 //RawTextViewer.style.opacity = 1.0;
6990 //RawTextViewer.style.backgroundColor = "rgba(0,0,0,0.0)";
6991 RawTextViewer.style.backgroundColor = "rgba(255,255,255,0.8)";
6992 RawTextViewer.style.zIndex = 1000;
6993 RawTextViewer.style.display = true;
6994
6995 if( RawTextViewerClose.style == null ){
6996     RawTextViewerClose.style = ``;
6997 }
6998 RawTextViewerClose.style.top = Number(y) + 10
7000 RawTextViewerClose.style.left = 100;
7001 RawTextViewerClose.style.zIndex = 1001;
7002 ScrollToElement(CurElement,RawTextViewerClose)
7003
7004 }
7005 function hideRawTextViewer(){
7006     RawTextViewer.style.left = 10000;
7007     RawTextViewer.style.zIndex = -100;
7008     RawTextViewer.style.opacity = 0.0;
7009     RawTextViewer.style = null
7010     RawTextViewer.innerHTML = `n`;
7011
7012 GshMenuSign.style.color = "rgba(255,128,128,1.0)";
7013 RawTextViewerClose.style.top = 0;
7014 RawTextViewerClose.style = null
7015 }
7016
7017 // source code viewr
7018 function frame_close(){
7019     srcframe = document.getElementById("src-frame");
7020     srcframe.innerHTML = `n`;
7021     //srcframe.style.cols = 1;
7022     srcframe.style.rows = 1;
7023     srcframe.style.height = 0;
7024     srcframe.style.display = false;
7025     src = document.getElementById("src-frame-textarea");
7026     src.innerHTML = `n`;
7027     //src.cols = 0
7028     src.rows = 0
7029     src.display = false
7030     //alert("--closed--")
7031 }
7032 //<!-- | <span onclick="html_view();">Source</span> -->
7033 //<!-- | <span onclick="frame_close();">SourceClose</span> -->
7034 //<!-- | <span>Download</span> -->
7035 function frame_open(){
7036     document.getElementById('GshFaviconURL').href = `n`;
7037     oldsrc = document.getElementById("GENSRC");
7038     if( oldsrc != null ){
7039         //alert("-I--(erasing old text)")
7040         oldsrc.innerHTML = `n`;
7041         return
7042     }else{
7043         //alert("--I--(no old text)")
7044     }
7045     styleBanner = GshBanner.getAttribute("style")
7046     GshBanner.removeAttribute("style")
7047     styleFooter = GshFooter.getAttribute("style")
7048     GshFooter.removeAttribute("style")
7049     if( document.getElementById('GJC_1') ) { GJC_1.remove() }
7050
7051 GshFaviconURL.href = `n`;
7052 GStat.removeAttribute('style')
7053 GshGrid.removeAttribute('style')
7054 GshMenuSign.removeAttribute('style')
7055 //GPos.removeAttribute('style')
7056 //GPos.innerHTML = `n`;
7057 //GLog.removeAttribute('style')
7058 //GLog.innerHTML = `n`;
7059 GMenu.removeAttribute('style')
7060 GTop.removeAttribute('style')
7061 GShellPlane.removeAttribute('style')
7062 RawTextViewer.removeAttribute('style')
7063 RawTextViewerClose.removeAttribute('style')
7064
7065 GJFactory_Destroy()
7066
7067 src = document.getElementById("gsh");
7068 src.innerHTML = src.outerHTML;

```

```

7069 srcframe = document.getElementById("src-frame");
7070 srcframe.innerHTML = ""
7071 + "<+" + "cite id=\"GENSRC\">\n"
7072 + "<+" + "style>\n"
7073 + "#GENSRC textarea{tab-size:4;}\n"
7074 + "#GENSRC textarea{-o-tab-size:4;}\n"
7075 + "#GENSRC textarea{-moz-tab-size:4;}\n"
7076 + "#GENSRC textarea{spellcheck:false;}\n"
7077 + "</" + "style>\n"
7078 + "<+" + "textarea id=\"src-frame-textarea\" cols=100 rows=20 class=\"gsh-code\">"
7079 + "/<+" + "html>\n" // lost preamble text
7080 + srchtml
7081 + "<+" + "/html>\n" // lost trail text
7082 + "<+" + "textarea>\n"
7083 + "</" + "cite><!-- GENSRC -->\n";
7084
7085 //srcframe.style.cols = 80;
7086 //srcframe.style.rows = 80;
7087
7088 GshBanner.setAttribute('style',styleBanner)
7089 GshFooter.setAttribute('style',styleFooter)
7090 }
7091 function fill_CSSView(){
7092 part = document.getElementById('GshStyleDef')
7093 view = document.getElementById('gsh-style-view')
7094 view.innerHTML = ""
7095 + "<+" + "textarea cols=100 rows=20 class=\"gsh-code\">"
7096 + part.innerHTML
7097 + "<+" + "/textarea>"}
7098 }
7099 function fill_JavaScriptView(){
7100 jspart = document.getElementById('gsh-script')
7101 view = document.getElementById('gsh-script-view')
7102 view.innerHTML = ""
7103 + "<+" + "textarea cols=100 rows=20 class=\"gsh-code\">"
7104 + jspart.innerHTML
7105 + "<+" + "/textarea>"}
7106 }
7107 function fill_DataView(){
7108 part = document.getElementById('gsh-data')
7109 view = document.getElementById('gsh-data-view')
7110 view.innerHTML = ""
7111 + "<+" + "textarea cols=100 rows=20 class=\"gsh-code\">"
7112 + part.innerHTML
7113 + "<+" + "/textarea>"}
7114 }
7115 function jumpTo_StyleView(){
7116 jsvview = document.getElementById('html-src')
7117 jsvview.open = true
7118 jsvview = document.getElementById('gsh-style-frame')
7119 jsvview.open = true
7120 fill_CSSView()
7121 }
7122 function jumpTo_JavaScriptView(){
7123 jsvview = document.getElementById('html-src')
7124 jsvview.open = true
7125 jsvview = document.getElementById('gsh-script-frame')
7126 jsvview.open = true
7127 fill_JavaScriptView()
7128 }
7129 function jumpTo_DataView(){
7130 jsvview = document.getElementById('html-src')
7131 jsvview.open = true
7132 jsvview = document.getElementById('gsh-data-frame')
7133 jsvview.open = true
7134 fill_DataView()
7135 }
7136 function jumpTo_WholeView(){
7137 jsvview = document.getElementById('html-src')
7138 jsvview.open = true
7139 jsvview = document.getElementById('gsh-whole-view')
7140 jsvview.open = true
7141 frame_open()
7142 }
7143 function html_view(){
7144 html_stop();
7145
7146 banner = document.getElementById('GshBanner').style.backgroundImage;
7147 footer = document.getElementById('GshFooter').style.backgroundImage;
7148 document.getElementById('GshBanner').style.backgroundImage = "";
7149 document.getElementById('GshBanner').style.backgroundPosition = "";
7150 document.getElementById('GshFooter').style.backgroundImage = "";
7151
7152 //srcwin = window.open("", "CodeView2", "");
7153 srcwin = window.open("", "", "");
7154 srcwin.document.write("<span id=\"gsh\">\n");
7155
7156 src = document.getElementById("gsh");
7157 srcwin.document.write("<+" + "style>\n");
7158 srcwin.document.write("textarea{tab-size:4;}\n");
7159 srcwin.document.write("textarea{-o-tab-size:4;}\n");
7160 srcwin.document.write("textarea{-moz-tab-size:4;}\n");
7161 srcwin.document.write("</style>\n");
7162 srcwin.document.write("<h2>\n");
7163 srcwin.document.write("<+" + "span onclick=\"window.close();\">Close</span> | \n");
7164 //srcwin.document.write("<+" + "span onclick=\"html_stop();\">Run</span>\n");
7165 srcwin.document.write("</h2>\n");
7166 srcwin.document.write("<textarea id=\"gsh-src-src\" cols=100 rows=60>");
7167 srcwin.document.write("/<+" + "html>\n");
7168 srcwin.document.write("<+" + "span id=\"gsh\">");
7169 srcwin.document.write(src.innerHTML);
7170 srcwin.document.write("<+" + "span><+" + "html>\n");
7171 srcwin.document.write("</" + "textarea>\n");
7172
7173 document.getElementById('GshBanner').style.backgroundImage = banner;
7174 document.getElementById('GshFooter').style.backgroundImage = footer
7175
7176 sty = document.getElementById("GshStyleDef");
7177 srcwin.document.write("<+" + "style>\n");
7178 srcwin.document.write(sty.innerHTML);
7179 srcwin.document.write("<+" + "style>\n");
7180
7181 run = document.getElementById("gsh-script");
7182 srcwin.document.write("<+" + "script>\n");
7183 srcwin.document.write(run.innerHTML);
7184 srcwin.document.write("<+" + "/script>\n");
7185
7186 srcwin.document.write("<+" + "span><+" + "html>\n"); // gsh span
7187 srcwin.document.close();
7188 srcwin.focus();
7189 }
7190 GSH = document.getElementById("gsh")
7191 //GSH.onclick = "alert('Ouch!')"
7192

```

```

7193 //GSH.css = {"background-color:#eef;"}
7194 //GSH.style = "background-color:#eef;"
7195 //GSH.style.display = false;
7196 //alert('Ouch0!');
7197 //GSH.style.display = true;
7198
7199 // 2020-0904 created, tentative
7200 document.addEventListener('keydown',jgshCommand);
7201 //CurElement = GshStatement
7202 CurElement = GshMenu
7203 MemElement = GshMenu
7204
7205 function nextSib(e){
7206     n = e.nextSibling;
7207     for( i = 0; i < 100; i++ ){
7208         if( n == null ){
7209             break;
7210         }
7211         if( n.nodeName == "DETAILS" ){
7212             return n;
7213         }
7214         n = n.nextSibling;
7215     }
7216     return null;
7217 }
7218 function prevSib(e){
7219     n = e.previousSibling;
7220     for( i = 0; i < 100; i++ ){
7221         if( n == null ){
7222             break;
7223         }
7224         if( n.nodeName == "DETAILS" ){
7225             return n;
7226         }
7227         n = n.previousSibling;
7228     }
7229     return null;
7230 }
7231 function setColor(e,eName,eColor){
7232     if( e.hasChildNodes() ){
7233         s = e.childNodes;
7234         if( s != null ){
7235             for( ci = 0; ci < s.length; ci++ ){
7236                 if( s[ci].nodeName == eName ){
7237                     s[ci].style.color = eColor;
7238                     //s[ci].style.backgroundColor = eColor;
7239                     break;
7240                 }
7241             }
7242         }
7243     }
7244 }
7245
7246 // https://docs.microsoft.com/en-us/previous-versions//hh781509(v=vs.85)
7247 function showCurElementPosition(ev){
7248 //  if( document.getElementById("GPos") == null ){
7249 //      return;
7250 //  }
7251 //  if( GPos == null ){
7252 //      return;
7253 //  }
7254     e = CurElement
7255     y = e.getBoundingClientRect().top.toFixed(0)
7256     x = e.getBoundingClientRect().left.toFixed(0)
7257
7258     h = ev + " "
7259     h += "y='"+y+"', "+ 'x='+'x'+ " -- "
7260     h += "w=" + window.innerWidth + ", h=" + window.innerHeight + " -- "
7261     //GPos.test = h
7262     //GPos.innerHTML = h
7263 //    GPos.innerHTML = h
7264 }
7265
7266 function DateShort(){
7267     d = new Date()
7268     return d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7269     + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7270 }
7271 function DateLong(){
7272     d = new Date()
7273     return d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7274     + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7275     + " " + d.getMilliseconds()
7276     + " " + d.getTimezoneOffset()/60
7277     + " " + d.getTime() + "." + d.getMilliseconds()
7278 }
7279
7280 }
7281 function GShellMenu(e){
7282     //GLog.innerHTML = "Hello, World! (" + DateLong() + ")"
7283     showGShellplane()
7284 }
7285 // placements of planes
7286 function GShellResizeX(ev){
7287     //if( document.getElementById("GMenu") != null ){
7288     //    GMenu.style.left = window.innerWidth - 100
7289     //    GMenu.style.top = window.innerHeight - 90 - 200
7290     //    //console.log("place GMENU "+GMenu.style.left+" "+GMenu.style.top)
7291
7292     //}
7293     GStat.style.width = window.innerWidth
7294     //if( document.getElementById("GPos") != null ){
7295     //    GPos.style.width = window.innerWidth
7296     //    GPos.style.top = window.innerHeight - 30; //GPos.style.height
7297     //}
7298     //if( document.getElementById("GLog") != null ){
7299     //    GLog.style.width = window.innerWidth
7300     //    GLog.innerHTML = ""
7301     //}
7302     //if( document.getElementById("GLog") != null ){
7303     //    GLog.innerHTML = "Resize: w=" + window.innerWidth +
7304     //    ", h=" + window.innerHeight
7305     //}
7306     showCurElementPosition(ev)
7307 }
7308 function GShellResize(){
7309     GShellResizeX("[RESIZE]")
7310 }
7311 window.onresize = GshellResize
7312 var prevNode = null
7313 function GSH_OnMouseMove(ev){
7314     x = ev.clientX
7315     y = ev.clientY
7316     d = new Date()

```

```

7317     t = d.getTime() / 1000
7318     if( document.elementFromPoint ){
7319         e = document.elementFromPoint(x,y)
7320         if( e != null ){
7321             if( e == prevNode ){
7322                 console.log(t+'('+x+','+y+')'
7323                     +e.nodeType+' '+e.tagName+'#'+e.id)
7324                 prevNode = e
7325             }
7326         } else{
7327             console.log(t+'('+x+','+y+') no element')
7328         }
7329     } else{
7330         console.log(t+'('+x+','+y+') no elementFromPoint')
7331     }
7332 }
7333 }
7334 window.addEventListener('mousemove',GJSH_OnMouseMove);
7335
7336 function GJSH_OnMouseMoveScreen(ev){
7337     x = ev.screenX
7338     y = ev.screenY
7339     d = new Date()
7340     t = d.getTime() / 1000
7341     console.log(t+'('+x+','+y+') no elementFromPoint')
7342 }
7343 //screen.addEventListener('mousemove',GJSH_OnMouseMoveScreen);
7344
7345 function ScrollToElement(oe,ne){
7346     ne.scrollIntoView()
7347     ny = ne.getBoundingClientRect().top.toFixed(0)
7348     nx = ne.getBoundingClientRect().left.toFixed(0)
7349     //GLog.innerHTML = "[ "+ny+","+nx+" ]"
7350     //window.scrollTo(0,0)
7351
7352     GTop.style.backgroundColor = "rgba(0,0,0,0.0)"
7353     GshGrid.style.left = '250px';
7354     GshGrid.style.zIndex = 0
7355     if( false ){
7356         oy = oe.getBoundingClientRect().top.toFixed(0)
7357         ox = oe.getBoundingClientRect().left.toFixed(0)
7358         y = e.getBoundingClientRect().top.toFixed(0)
7359         x = e.getBoundingClientRect().left.toFixed(0)
7360         window.scrollTo(x,y)
7361         ny = e.getBoundingClientRect().top.toFixed(0)
7362         nx = e.getBoundingClientRect().left.toFixed(0)
7363         //GLog.innerHTML = "[ "+oy+","+ox+" ]->[ "+y+","+x+" ]->[ "+ny+","+nx+" ]"
7364     }
7365 }
7366 function showGShellPlane(){
7367     if( GShellPlane.style.zIndex == 0 ){
7368         GShellPlane.style.zIndex = 1000;
7369         GShellPlane.style.left = 30;
7370         GShellPlane.style.height = 320;
7371         GShellPlane.innerHTML = DateLong() + "<br>" +
7372             "-- History --<br>" + MyHistory;
7373     } else{
7374         GShellPlane.style.zIndex = 0;
7375         GShellPlane.style.left = 0;
7376         GShellPlane.style.height = 50;
7377         GShellPlane.innerHTML = "";
7378     }
7379 }
7380 var SuppressGJShell = false
7381 function jgshCommand(kevent){
7382     if( SuppressGJShell ){
7383         return
7384     }
7385     key = kevent
7386     keycode = key.code
7387     //GStat.style.width = window.innerWidth
7388     GStat.style.backgroundColor = "rgba(0,0,0,0.4)"
7389
7390     console.log("JSGsh-Key:"+keycode+"(^~)/")
7391     if( keycode == "Slash" ){
7392         console.log('('+x+','+y+') ')
7393         e = document.elementFromPoint(x,y)
7394         console.log('('+x+','+y+') '+e.nodeType+' '+e.tagName+'#'+e.id)
7395     } else{
7396         if( keycode == "Digit0" ){ // fold side-bar
7397             // "Zero page"
7398             showGShellPlane();
7399         } else{
7400             if( keycode == "Digit1" ){ // fold side-bar
7401                 primary.style.width = "94%"
7402                 secondary.style.width = "0%"
7403                 secondary.style.opacity = 0
7404                 GStat.innerHTML = "[Single Column View]"
7405             } else{
7406                 if( keycode == "Digit2" ){ // unfold side-bar
7407                     primary.style.width = "58%"
7408                     secondary.style.width = "36%"
7409                     secondary.style.opacity = 1
7410                     GStat.innerHTML = "[Double Column View]"
7411             } else{
7412                 if( keycode == "KeyU" ){ // fold/unfold all
7413                     html_fold(GshMenuFold);
7414                     location.href = "#" + CurElement.id;
7415                 } else{
7416                     if( keycode == "KeyO" || keycode == "ArrowRight" ){ // fold the element
7417                         CurElement.open = !CurElement.open;
7418                     } else{
7419                         if( keycode == "ArrowRight" ){ // unfold the element
7420                             CurElement.open = true
7421                         } else{
7422                             if( keycode == "ArrowLeft" ){ // unfold the element
7423                                 CurElement.open = false
7424                             } else{
7425                                 if( keycode == "KeyI" ){ // inspect the element
7426                                     e = CurElement
7427                                     //GLog.innerHTML =
7428                                     GJLog_append("Current Element: " + e + "<br>" +
7429                                         + "name='"+e.nodeName+"', "
7430                                         + "id='"+e.id+"', "
7431                                         + "children='"+e.childNodes.length+"', "
7432                                         + "parent='"+e.parentNode.id+"<br>" +
7433                                         + "text='"+e.textContent+"')
7434                                     GStat.style.backgroundColor = "rgba(0,0,0,0.8)"
7435                                     return
7436                                 } else{
7437                                     if( keycode == "KeyM" ){ // memory the position
7438                                         MemElement = CurElement
7439                                     } else{
7440                                         if( keycode == "KeyN" || keycode == "ArrowDown" ){ // next element
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7599
7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632
7633
7634
7635
7636
7637
7638
7639
7640
7641
7642
7643
7644
7645
7646
7647
7648
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658
7659
7659
7660
7661
7662
7663
7664
7665
7666
7667
7668
7669
7669
7670
7671
7672
7673
7674
7675
7676
7677
7678
7679
7679
7680
7681
7682
7683
7684
7685
7686
7687
7688
7689
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7809
7810
7811
7812
7813
7814
7815
7816
7817
7818
7819
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
7999
8000
8001
8002
8003
8004
8005
8006
8007
8008
8009
8009
8010
8011
8012
8013
8014
8015
8016
8017
8018
8019
8019
8020
8021
8022
8023
8024
8025
8026
8027
8028
8029
8029
8030
8031
8032
8033
8034
8035
8036
8037
8038
8039
8039
8040
8041
8042
8043
8044
8045
8046
8047
8048
8049
8049
8050
8051
8052
8053
8054
8055
8056
8057
8058
8059
8059
8060
8061
8062
8063
8064
8065
8066
8067
8068
8069
8069
8070
8071
8072
8073
8074
8075
8076
8077
8078
8079
8079
8080
8081
8082
8083
8084
8085
8086
8087
8088
8089
8089
8090
8091
8092
8093
8094
8095
8096
8097
8098
8099
8099
8100
8101
8102
8103
8104
8105
8106
8107
8108
8109
8109
8110
8111
8112
8113
8114
8115
8116
8117
8118
8119
8119
8120
8121
8122
8123
8124
8125
8126
8127
8128
8129
8129
8130
8131
8132
8133
8134
8135
8136
8137
8138
8139
8139
8140
8141
8142
8143
8144
8145
8146
8147
8148
8149
8149
8150
8151
8152
8153
8154
8155
8156
8157
8158
8159
8159
8160
8161
8162
8163
8164
8165
8166
8167
8168
8169
8169
8170
8171
8172
8173
8174
8175
8176
8177
8178
8179
8179
8180
8181
8182
8183
8184
8185
8186
8187
8188
8189
8189
8190
8191
8192
8193
8194
8195
8196
8197
8198
8199
8199
8200
8201
8202
8203
8204
8205
8206
8207
8208
8209
8209
8210
8211
8212
8213
8214
8215
8216
8217
8218
8219
8219
8220
8221
8222
8223
8224
8225
8226
8227
8228
8229
8229
8230
8231
8232
8233
8234
8235
8236
8237
8238
8239
8239
8240
8241
8242
8243
8244
8245
8246
8247
8248
8249
8249
8250
8251
8252
8253
8254
8255
8256
8257
8258
8259
8259
8260
8261
8262
8263
8264
8265
8266
8267
8268
8269
8269
8270
8271
8272
8273
8274
8275
8276
8277
8278
8279
8279
8280
8281
8282
8283
8284
8285
8286
8287
8288
8289
8289
8290
8291
8292
8293
8294
8295
8296
8297
8298
8299
8299
8300
8301
8302
8303
8304
8305
8306
8307
8308
8309
8309
8310
8311
8312
8313
8314
8315
8316
8317
8318
8319
8320
8321
8322
8323
8324
8325
8326
8327
8328
8329
8329
8330
8331
8332
8333
8334
8335
8336
8337
8338
8339
8339
8340
8341
8342
8343
8344
8345
8346
8347
8348
8349
8349
8350
8351
8352
8353
8354
8355
8356
8357
8358
8359
8359
8360
8361
8362
8363
8364
8365
8366
8367
8368
8369
8369
8370
8371
8372
8373
8374
8375
8376
8377
8378
8379
8379
8380
8381
8382
8383
8384
8385
8386
8387
8388
8389
8389
8390
8391
8392
8393
8394
8395
8396
8397
8398
8399
8399
8400
8401
8402
8403
8404
8405
8406
8407
8408
8409
8409
8410
8411
8412
8413
8414
8415
8416
8417
8418
8419
8419
8420
8421
8422
8423
8424
8425
8426
8427
8428
8429
8429
8430
8431
8432
8433
8434
8435
8436
8437
8438
8439
8439
8440
8441
8442
8443
8444
8445
8446
8447
8448
8449
8449
8450
8451
8452
8453
8454
8455
8456
8457
8458
8459
8459
8460
8461
8462
8463
8464
8465
8466
8467
8468
8469
8469
8470
8471
8472
8473
8474
8475
8476
8477
8478
8479
8479
8480
8481
8482
8483
8484
8485
8486
8487
8488
8489
8489
8490
8491
8492
8493
8494
8495
8496
8497
8498
8499
8499
8500
8501
8502
8503
8504
8505
8506
8507
8508
8509
8509
8510
8511
8512
8513
8514
8515
8516
8517
8518
8519
8519
8520
8521
8522
8523
8524
8525
8526
8527
8528
8529
8529
8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8539
8540
8541
8542
8543
8544
8545
8546
8547
8548
8549
8549
8550
8551
8552
8553
8554
8555
8556
8557
8558
8559
8559
8560
8561
8562
8563
8564
8565
8566
8567
8568
8569
8569
8570
8571
8572
8573
8574
8575
8576
8577
8578
8579
8579
8580
8581
8582
8583
8584
8585
8586
8587
8588
8589
8589
8590
8591
8592
8593
8594
8595
8596
8597
8598
8599
8599
8600
8601
8602
8603
8604
8605
8606
8607
8608
8609
8609
8610
8611
8612
8613
8614
8615
8616
8617
8618
8619
8619
8620
8621
8622
8623
8624
8625
8626
8627
8628
8629
8629
8630
8631
8632
8633
8634
8635
8636
8637
8638
8639
8639
8640
8641
8642
8643
8644
8645
8646
8647
8648
8649
8649
8650
8651
8652
8653
8654
8655
8656
8657
8658
8659
8659
8660
8661
8662
8663
8664
8665
8666
8667
8668
8669
8669
8670
8671
8672
8673
8674
8675
8676
8677
8678
8679
8679
8680
8681
8682
8683
8684
8685
8686
8687
8688
8689
8689
8690
8691
8692
8693
8694
8695
8696
8697
8698
8699
8699
8700
8701
8702
8703
8704
8705
8706
8707
8708
8709
8709
8710
8711
8712
8713
8714
8715
8716
8717
8718
8719
8719
8720
8721
8722
8723
8724
8725
8726
8727
8728
8729
8729
8730
8731
8732
8733
8734
8735
8736
8737
8738
8739
8739
8740
8741
8742
8743
8744
8745
8746
8747
8748
8749
8749
8750
8751
8752
8753
8754
8755
8756
8757
8758
8759
8759
8760
8761
8762
8763
8764
8765
8766
8767
8768
8769
8769
8770
8771
8772
8773
8774
8775
8776
8777
8778
8779
8779
8780
8781
8782
8783
8784
8785
8786
8787
8788
8789
8789
8790
8791
8792
8793
8794
8795
8796
8797
8798
8799
8799
8800
8801
8802
8803
8804
8805
8806
8807
8808
8809
8809
8810
8811
8812
8813
8814
8815
8816
8817
8818
8819
8819
8820
8821
8822
8823
8824
8825
8826
8827
8828
8829
8829
8830
8831
8832
8833
8834
8835
8836
8837
8838
8839
8839
8840
8841
8842
8843
8844
8845
8846
8847
8848
8849
8849
8850
8851
8852
8853
8854
8855
8856
8857
8858
8859
8859
8860
8861
8862
8863
8864
8865
8866
8867
8868
8869
8869
8870
8871
8872
8873
8874
8875
8876
8877
8878
8879
8879
8880
8881
8882
8883
8884
8885
8886
8887
8888
8889
8889
8890
8891
8892
8893
8894
8895
8896
8897
8898
8899
8899
8900
8901
8902
8903
8904
8905
8906
8907
8908
8909
8909
8910
8911
8912
8913
8914
8915
8916
8917
8918
8919
8919
8920
8921
8922
8923
8924
8925
8926
8927
8928
8929
8929
8930
8931
8932
8933
8934
8935
8936
8937
8938
8939
8939
8940
8941
8942
8943
8944
8945
8946
8947
8948
8949
8949
8950
8951
8952
8953
8954
8955
8956
8957
8958
8959
8959
8960
8961
8962
8963
8964
8965
8966
8967
8968
8969
8969
8970
8971
8972
8973
8974
8975
8976
8977
8978
8979
8979
8980
8981
8982
8983
8984
8985
8986
8987
8988
8989
8989
8990
8991
8992
8993
8994
8995
8996
8997
8998
8999
8999
9000
9001
9002
9003
9004
9005
9006
9007
9008
9009
9009
9010
9011
9012
9013
9014
9015
9016
9017
9018
9019
9019
9020
9021
9022
9023
9024
9025
9026
9027
9028
9029
9029
9030
9031
9032
9033
9034
9035
9036
9037
9038
9039
9039
9040
9041
9042
9043
9044
9045
9046
9047
9048
9049
9049
9050
9051
9052
9053
9054
9055
9056
9057
9058
9059
9059
9060
9061
9062
9063
9064
9065
9066
9067
9068
9069
9069
9070
9071
9072
9073
9074
9075
9076
9077
9078
9079
9079
9080
9081
9082
9083
9084
9085
9086
9087
9088
9089
9089
9090
9091
9092
9093
9094
9095
9096
9097
9098
9099
9099
9100
9101
9102
9103
9104
9105
9106
9107
9108
9109
9109
9110
9111
9112
9113
9114
9115
9116
9117
9118
9119
9119
9120
9121
9122
9123
9124
9125
9126
9127
9128
9129
9129
9130
9131
9132
9133
9134
9135
9136
9137
9138
9139
9139
9140
9141
9142
9143
9144
9145
9146
9147
9148
9149
9149
9150
9151
9152
9153
9154
9155
9156
9157
9158
9159
9159
9160
9161
9162
9163
9164
9165
9166
9167
9168
9169
9169
9170
9171
9172
9
```

```

7441     e = nextSib(CurElement)
7442     if( e != null ){
7443         setColor(CurElement,"SUMMARY","#fff")
7444         setColor(e,"SUMMARY","#8f8") // should be complement ?
7445         oe = CurElement
7446         CurElement = e
7447         //location.href = "#" + e.id;
7448         ScrollToElement(oe,e)
7449     }
7450 }else{
7451     if( keycode == "KeyP" || keycode == "ArrowUp" ){ // previous element
7452         oe = CurElement
7453         e = prevSib(CurElement)
7454         if( e != null ){
7455             setColor(CurElement,"SUMMARY","#fff")
7456             setColor(e,"SUMMARY","#8f8") // should be complement ?
7457             CurElement = e
7458             //location.href = "#" + e.id;
7459             ScrollToElement(oe,e)
7460         }else{
7461             e = document.getElementById("GshBanner")
7462             if( e != null ){
7463                 setColor(CurElement,"SUMMARY","#fff")
7464                 CurElement = e
7465                 ScrollToElement(oe,e)
7466             }else{
7467                 e = document.getElementById("primary")
7468                 if( e != null ){
7469                     setColor(CurElement,"SUMMARY","#fff")
7470                     CurElement = e
7471                     ScrollToElement(oe,e)
7472                 }
7473             }
7474         }
7475     }else
7476     if( keycode == "KeyR" ){
7477         location.reload()
7478     }else
7479     if( keycode == "KeyJ" ){
7480         GshGrid.style.top = '120px';
7481         GshGrid.innerHTML = '(>_<){Down}';
7482     }else
7483     if( keycode == "KeyK" ){
7484         GshGrid.style.top = '0px';
7485         GshGrid.innerHTML = '(^_-){Up}';
7486     }else
7487     if( keycode == "KeyH" ){
7488         GshGrid.style.left = '0px';
7489         GshGrid.innerHTML = "(__-){Left}";
7490     }else
7491     if( keycode == "KeyL" ){
7492         //GLog.innerHTML += 
7493         GJLog_append(
7494             'screen=' + screen.width + 'px' + '<br>' +
7495             'window=' + window.innerWidth + 'px' + '<br>' +
7496         )
7497         GshGrid.style.left = (document.documentElement.clientWidth - 160).toString(10) + 'px';
7498         GshGrid.innerHTML = '({_-_}{Right)';
7499     }else
7500     if( keycode == "KeyS" ){
7501         html_stop(GshMenuStop, true)
7502     }else
7503     if( keycode == "KeyF" ){
7504         html_fork()
7505     }else
7506     if( keycode == "KeyC" ){
7507         window.close()
7508     }else
7509     if( keycode == "KeyD" ){
7510         html_digest()
7511     }else
7512     if( keycode == "KeyV" ){
7513         e = document.getElementById('gsh-digest')
7514         if( e != null ){
7515             showDigest(e)
7516         }
7517     }
7518     showCurElementPosition("[+key.code+]");
7519     //if( document.getElementById("GPos") != null ){
7520     //    GPos.innerHTML += "[+key.code+] --"
7521     //}
7522     //GShellResizeX("[+key.code+] --");
7523 }
7524 GShellResizeX("[INIT]");
7525
7526 DisplaySize = '-- Display: ' + 'screen=' + screen.width + 'px, ' + 'window=' + window.innerWidth + 'px';
7527
7528 let {text, digest} = getDigest()
7529 //GLog.innerHTML += 
7530 GJLog_append(
7531     '-- GShell: ' + GshVersion.innerHTML + '\n' +
7532     '-- Digest: ' + digest + '\n' +
7533     DisplaySize
7534     //+ "<br>" + "-- LastVisit:<br>" + MyHistory
7535 )
7536 GShellResizeX(null);
7537
7538 // <a href="https://www.w3.org/TR/WebCryptoAPI/">Web Cryptography API</a>
7539 //Convert a string into an ArrayBuffer
7540 //from https://developers.google.com/web/updates/2012/06/How-to-convert-ArrayBuffer-to-and-from-String
7541 function str2ab(str) {
7542     const buf = new ArrayBuffer(str.length);
7543     const bufView = new Uint8Array(buf);
7544     for (let i = 0, strLen = str.length; i < strLen; i++) {
7545         bufView[i] = str.charCodeAt(i);
7546     }
7547     return buf;
7548 }
7549
7550 function importPrivateKey(pem) {
7551     const binaryDerString = window.atob(pemContents);
7552     const binaryDer = str2ab(binaryDerString);
7553     return window.crypto.subtle.importKey(
7554         "pkcs8",
7555         binaryDer,
7556         {
7557             name: "RSA-PSS",
7558             modulusLength: 2048,
7559             publicExponent: new Uint8Array([1, 0, 1]),
7560             hash: "SHA-256",
7561         },
7562         true,
7563         ["sign"]
7564     );
}

```

```

7565 }
7566 //importPrivateKey(ppem)
7567
7568 //key = {}
7569 //buf = "abc"
7570 //enc = "xyzxxxxx"; //crypto.publicEncrypt(key,buf)
7571 //b64 = btoa(enc)
7572 //dec = atob(b64)
7573 //GLog.innerHTML = "enc:" + b64 + ", dec:" + dec
7574
7575 </script>
7576
7577 <span id="gjc" data-title="GJConsole" data-author="sato@its-more.jp">
7578 <!-- ----- GJConsole BEGIN { ----- -->
7579 <p>
7580 <span id="GJE_RootNode0"></span>
7581 </p>
7582 <style id="GJConsoleStyle">
7583   .GJConsole {
7584     z-index:1000;
7585     width:400; height:200px;
7586     margin:2px;
7587     color:#fff; background-color:#66a;
7588     font-size:12px; font-family:monospace,Courier New;
7589   }
7590 </style>
7591
7592 <script id="GJConsoleScript" class="GJConsole">
7593   var PS1 = "% "
7594   function GJC_KeyDown(keyevent){
7595     key = keyevent.code
7596     if( key == "Enter" ){
7597       GJC_Command(this)
7598       this.value += "\n" + PS1 // prompt
7599     }else
7600     if( key == "Escape" ){
7601       SuppressGJShell = false
7602       GshMenu.focus() // should be previous focus
7603     }
7604   }
7605   var GJC_SessionId
7606   function GJC_SetSessionId(){
7607     var xd = new Date()
7608     GJC_SessionId = xd.getTime() / 1000
7609   }
7610   GJC_SetSessionId()
7611   function GJC_Memory(mem,args,text){
7612     argv = args.split(' ')
7613     cmd = argv[0]
7614     argv.shift()
7615     args = argv.join(' ')
7616     ret = ""
7617
7618     if( cmd == 'clear' ){
7619       Permanent.setItem(mem,'')
7620     }else
7621     if( cmd == 'read' ){
7622       ret = Permanent.getItem(mem)
7623     }else
7624     if( cmd == 'save' ){
7625       val = Permanent.getItem(mem)
7626       if( val == null ){ val = "" }
7627       d = new Date()
7628       val += d.getTime()/1000+" "+GJC_SessionId+" "+document.URL+" "+args+"\n"
7629       val += text.value
7630       Permanent.setItem(mem,val)
7631     }else
7632     if( cmd == 'write' ){
7633       val = Permanent.getItem(mem)
7634       if( val == null ){ val = "" }
7635       d = new Date()
7636       val += d.getTime()/1000+" "+GJC_SessionId+" "+document.URL+" "+args+"\n"
7637       Permanent.setItem(mem,val)
7638     }else{
7639       ret = "Commands: write | read | save | clear"
7640     }
7641   return ret
7642 }
7643 // -- 2020-09-14 added TableEditor
7644 var GJE_CurElement = null; //GJE_RootNode
7645 GJE_NodeSaved = null
7646 GJE_TableNo = 1
7647 function GJE_StyleKeyCommand(kev){
7648   keycode = kev.code
7649   console.log('GJE-Key: '+keycode)
7650   if( keycode == 'Escape' ){
7651     GJE_SetStyle(this);
7652   }
7653   kev.stopPropagation()
7654   // https://developer.mozilla.org/en-US/docs/Web/API/Event/stopPropagation
7655 }
7656 var GJE_CommandMode = false
7657 function GJE_TableKeyCommand(kev,tab){
7658   wasCmdMode = GJE_CommandMode
7659   key = kev.code
7660   if( key == 'Escape' ){
7661     console.log("To command mode: "+tab.nodeName+"#"+tab.id)
7662     //tab.setAttribute('contenteditable','false')
7663     tab.style.caretColor = "blue"
7664     GJE_CommandMode = true
7665   }else
7666   if( key == "KeyA" ){
7667     tab.style.caretColor = "red"
7668     GJE_CommandMode = false
7669   }else
7670   if( key == "KeyI" ){
7671     tab.style.caretColor = "red"
7672     GJE_CommandMode = false
7673   }else
7674   if( key == "KeyO" ){
7675     tab.style.caretColor = "red"
7676     GJE_CommandMode = false
7677   }else
7678   if( key == "KeyJ" ){
7679     console.log("ROW-DOWN")
7680   }else
7681   if( key == "KeyK" ){
7682     console.log("ROW-UP")
7683   }else
7684   if( key == "KeyW" ){
7685     console.log("COL-FORW")
7686   }else
7687   if( key == "KeyB" ){
7688     console.log("COL-BACK")
7689   }

```

```

7689 }
7690 kev.stopPropagation()
7691 if( wasCmdMode ){
7692   kev.preventDefault()
7693 }
7694 }
7695 function GJE_DragEvent(ev,elem){
7696   x = ev.clientX
7697   y = ev.clientY
7698   console.log("Dragged: "+this.nodeName+'#'+this.id+ ' x=' +x+ ' y=' +y)
7700 }
7701 // https://developer.mozilla.org/en-US/docs/Web/API/DragEvent
7702 // https://www.w3.org/TR/uievents/#events-mouseevents
7703 function GJE_DropEvent(ev,elem){
7704   x = ev.clientX
7705   y = ev.clientY
7706   this.style.x = x
7707   this.style.y = y
7708   this.style.position = 'absolute' // 'fixed'
7709   this.parentNode = gsh // just for test
7710   console.log("Dropped: "+this.nodeName+'#'+this.id+ ' x=' +x+ ' y=' +y
7711   +' parent=' +this.parentNode.id)
7712 }
7713 function GJE_SetTableStyle(ev){
7714   this.innerHTML = this.value; // sync. for external representation?
7715   if(false){
7716     stid = this.parentNode.id+this.id
7717     // and remove "_span" at the end
7718     e = document.getElementById(stid)
7719     //alert('SetTableStyle #' +e.id+'\n'+this.value)
7720     if( e != null ){
7721       e.innerHTML = this.value
7722     }else{
7723       console.log('Style Not found: '+stid)
7724     }
7725     //alert('event StopPropagation: '+ev)
7726   }
7727 }
7728 function setCSSofClass(cclass,cstyle){
7729   const ss = document.styleSheets[3]; // 0, 1, 2, 3, ... ?
7730   rlen = ss.cssRules.length;
7731   let tabrule = null;
7732   rulex = -1
7733   // should skip white space at the top of cstyle
7735   sel = cstyle.charAt(0);
7736   selector = sel+cclass;
7737   console.log('-- search style rule for '+selector)
7738   for(let i = 0; i < rlen; i++){
7739     cr = ss.cssRules[i];
7740     console.log('CSS rule ['+i+'/'+rlen+'] '+cr.selectorText);
7741     if( cr.selectorText === selector ){ // css class selector
7742       tabrule = ss.cssRules[i];
7743       console.log('CSS rule found for:['+i+'/'+rlen+'] '+selector);
7744       ss.deleteRule(i);
7745       //rlen = ss.cssRules.length;
7746       rulex = i
7747       // should search and replace the property here
7748     }
7749   }
7750   // https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleSheet/insertRule
7751   if( tabrule == null ){
7752     console.log('CSS rule NOT found for:[ '+rlen+'] '+selector);
7753     ss.insertRule(cstyle,rlen);
7754     ss.insertRule(cstyle,0); // override by 0?
7755     console.log('CSS rule inserted:[ '+rlen+']\n'+cstyle);
7756   }else{
7757     ss.insertRule(cstyle,rlen);
7758     ss.insertRule(cstyle,0);
7759     console.log('CSS rule replaced:[ '+rlen+']\n'+cstyle);
7760   }
7761 }
7762 }
7763 function GJE_SetStyle(te){
7764   console.log('Apply the style to:' +te.id+'\n');
7765   console.log('Apply the style to:' +te.parentNode.id+'\n');
7766   console.log('Apply the style to:' +te.parentNode.class+'\n');
7767   cclass = te.parentNode.class;
7768   setCSSofClass(cclass,te.value); // should get selector part from
7769   // selector { rules }
7770
7771   if(false){
7772     //console.log('Apply the style:')
7773     //stid = this.parentNode.id+this.id+
7774     //stid = this.id+'.style"
7775     css = te.value
7776     stid = te.parentNode.id+'.style'
7777     e = document.getElementById(stid)
7778     if( e != null ){
7779       //console.log('Apply the style:' +e.id+'\n'+te.value);
7780       console.log('Apply the style:' +e.id+'\n'+css);
7781     // e.innerHTML = css; //te.value;
7782     // ncss = e.sheet;
7783     // ncss.insertRule(te.value,ncss.cssRules.length);
7784     }else{
7785       console.log('No element to Apply the style: '+stid)
7786     }
7787     tblid = te.parentNode.id+"table";
7788     e = document.getElementById(tblid);
7789     if( e != null ){
7790       //e.setAttribute('style',css);
7791       e.setProperty('style',css,'!important');
7792     }
7793   }
7794 }
7795 function makeTable(argv){
7796   //tid = ''
7797   cwe = GJE_CurElement
7798   tid = 'table_' + GJE_TableNo
7799
7800   nt = new Text('\n')
7801   cwe.appendChild(nt)
7802
7803   ne = document.createElement('span'); // the container
7804   cwe.appendChild(ne)
7805   ne.id = tid + '-span'
7806   ne.setAttribute('contenteditable',true)
7807
7808   hspan = document.createElement('span'); // html part
7809   //hspan.id = tid + '-html'
7810   //ne.innerHTML = '\n'
7811   nt = new Text('\n')
7812   ne.appendChild(nt)

```

```

7813     ne.appendChild(htspan)
7814
7815     htspan.id = tid
7816     htspan.setAttribute('class',tid)
7817
7818     ne.setAttribute('draggable','true')
7819     ne.addEventListener('drag',GJE_DragEvent);
7820     ne.addEventListener('dragend',GJE_DropEvent);
7821
7822     var col = 3
7823     var row = 2
7824     if( argv[0] != null ){
7825         col = argv[0]
7826         argv.shift()
7827     }
7828     if( argv[0] != null ){
7829         row = argv[0]
7830         argv.shift()
7831     }
7832
7833     //ne.setAttribute('class',tid)
7834     ht = "\n"
7835     //ht += '<'+table' + 'id="'+tid+'" + ' class="'+tid+'"
7836     ht += '<'+table'
7837     + ' onkeydown="GJE_TableKeyCommand(event,this)"'
7838     //+ ' ondrag="GJE_DragEvent(event,this)\"\n'
7839     //+ ' ondragend="GJE_DropEvent(event,this)\"\n'
7840     //+ ' draggable="true"\n'
7841     //+ ' contenteditable="true"'
7842     + '>\n'
7843     ht += '<'+tbody>\n';
7844     for( r = 0; r < row; r++ ){
7845         ht += "<"+tr>\n"
7846         for( c = 0; c < col; c++ ){
7847             ht += "<"+td>"
7848             ht += " ABCDEFGHIJKLMNOPQRSTUVWXYZ.charAt(c) + r
7849             ht += "<"/td>\n"
7850         }
7851         ht += "<"/tr>\n"
7852     }
7853     ht += '<'+tbody>\n';
7854     ht += '<'+table>\n';
7855     htspan.innerHTML = ht;
7856     nt = new Text('\n')
7857     ne.appendChild(nt)
7858
7859     st = '#'+tid+' *{\n' // # for instance specific
7860     +' '+border:1px solid #aaa;\n'
7861     +' '+background-color:#efe;\n'
7862     +' '+color:#222;\n'
7863     +' '+font-size:#14pt !important;\n'
7864     +' '+font-family:monospace,Courier New !important;\n'
7865     +' } /* hit ESC to apply */\n'
7866
7867     // wish script to be included
7868     //nj = document.createElement('script')
7869     //ne.appendChild(nj)
7870     //ne.innerHTML = 'function SetStyle(e){'
7871
7872     // selector seems lost in dynamic style appending
7873     if(false){
7874         ns = document.createElement('style')
7875         ne.appendChild(ns)
7876         ns.id = tid + '.style'
7877         ns.innerHTML = '\n'+st
7878         nt = new Text('\n')
7879         ne.appendChild(nt)
7880     }
7881     setCSSofClass(tid,st); // should be in JavaScript script?
7882
7883     nx = document.createElement('textare')
7884     ne.appendChild(nx)
7885     nx.id = tid + '-style_def'
7886     nx.setAttribute('class','GJ_StyleEditor')
7887     nx.spellcheck = false
7888     nx.cols = 60
7889     nx.rows = 10
7890     nx.innerHTML = '\n'+st
7891     nx.addEventListener('change',GJE_SetTableStyle);
7892     nx.addEventListener('keydown',GJE_StyleKeyCommand);
7893     //nx.addEventListener('click',GJE_SetTableStyle);
7894
7895     nt = new Text('\n')
7896     cwe.appendChild(nt)
7897
7898     GJE_TableNo += 1
7899     return 'created TABLE id="'+tid+'"
7900 }
function GJE_NodeEdit(argv){
7901     cwe = GJE_CurElement
7902     cmd = argv[0]
7903     argv.shift()
7904     args = argv.join(' ')
7905     ret = ""
7906
7907     if( cmd == '.u' || cmd == '.un' || cmd == 'undo' ){
7908         if( GJE_NodeSaved != null ){
7909             xn = GJE_RootNode
7910             GJE_RootNode = GJE_NodeSaved
7911             GJE_NodeSaved = xn
7912             ret = '-- did undo'
7913         }else{
7914             ret = '-- could not undo'
7915         }
7916     }
7917     return ret
7918 }
GJE_NodeSaved = GJE_RootNode.cloneNode()
7919 if( cmd == '.c' || cmd == '.cd' || cmd == 'cd' ){
7920     if( argv[0] == null ){
7921         ne = GJE_RootNode
7922     }else{
7923         if( argv[0] == '..' ){
7924             ne = cwe.parentNode
7925         }else{
7926             ne = document.getElementById(argv[0])
7927         }
7928         if( ne != null ){
7929             GJE_CurElement = ne
7930             ret = "-- current node: " + ne.id
7931         }else{
7932             ret = "-- not found: " + argv[0]
7933         }
7934     }
7935 }
if( cmd == '.mkt' || cmd == '.mktable' ){

```

```

7937     makeTable(argv)
7938   }else
7939     if( cmd == '.m' || cmd == '.mk' || cmd == 'mk' ){
7940       ne = document.createElement(argv[0])
7941       //ne.id = argv[0]
7942       ret = "-- created " + ne + " under " + cwe.tagName + "#" + cwe.id
7943       cwe.appendChild(ne)
7944       if( cmd == '.m' || cmd == '.mk' ){
7945         GJE_CurElement = ne
7946       }
7947     }else
7948       if( cmd == '.n' || cmd == '.nm' || cmd == 'nm' ){
7949         cwe.id = argv[0]
7950       }else
7951         if( cmd == '.r' || cmd == '.rm' || cmd == 'rm' ){
7952       }else
7953         if( cmd == '.h' || cmd == '.sh' || cmd == 'sh' ){
7954           s = argv.join(' ')
7955           cwe.innerHTML = s
7956       }else
7957         if( cmd == '.a' || cmd == '.sa' || cmd == 'sa' ){
7958           cwe.setAttribute(argv[0],argv[1])
7959         }else
7960           if( cmd == '.l' ){
7961         }else
7962           if( cmd == '.i' || cmd == '.ih' || cmd == 'ih' ){
7963             ret = cwe.innerHTML
7964           }else
7965             if( cmd == '.p' || cmd == '.pw' || cmd == 'pw' ){
7966               ret = cwe.nodeType + " " + cwe.tagName + " " + cwe.id
7967               for( we = cwe.parentNode; we != null; ){
7968                 ret += "\n" + " " + we.nodeType + " " + we.tagName + " " + we.id
7969                 we = we.parentNode
7970               }
7971             }else
7972               {
7973                 ret = "Command: mk | rm \n"
7974                 ret += "    pw -- print current node\n"
7975                 ret += "    mk type -- make node with name and type\n"
7976                 ret += "    nm name -- set the id #name of current node\n"
7977                 ret += "    rm name -- remove named node\n"
7978                 ret += "    cd name -- change current node\n"
7979               }
7980             //alert(ret)
7981           return ret
7982         }
7983       function GJC_Command(text){
7984         lines = text.value.split('\n')
7985         line = lines[lines.length-1]
7986         argv = line.split(' ')
7987         text.value += '\n'
7988         if( argv[0] == '%' ){ argv.shift() }
7989         args0 = argv.join(' ')
7990         cmd = argv[0]
7991         argv.shift()
7992         args = argv.join(' ')
7993
7994         if( cmd == 'nolog' ){
7995           StopConsoleLog = true
7996         }else
7997           if( cmd == 'new' ){
7998             if( argv[0] == 'table' ){
7999               argv.shift()
8000               console.log('argv=' + argv)
8001               text.value += makeTable(argv)
8002             }else
8003               if( argv[0] == 'console' ){
8004                 text.value += GJ_NewConsole('GJ_Console')
8005               }else{
8006                 text.value += '-- new { console | table }'
8007               }
8008           }else
8009             if( cmd == 'strip' ){
8010               //text.value += GJF_StripClass()
8011             }else
8012               if( cmd == 'css' ){
8013                 sel = '#table_1'
8014                 if(argv[0]==''0')
8015                   rule1 = sel+'{color:#000 !important; background-color:#fff !important;}';
8016                 else
8017                   rule1 = sel+'{color:#f00 !important; background-color:#eef !important;}';
8018                 document.styleSheets[3].deleteRule(0);
8019                 document.styleSheets[3].insertRule(rule1,0);
8020                 text.value += 'CSS rule added: '+rule1
8021             }else
8022               if( cmd == 'print' ){
8023                 e = null;
8024                 if( e == null ){
8025                   e = document.getElementById('GJFactory_0')
8026                 }else
8027                   if( e == null ){
8028                     e = document.getElementById('GJFactory_1')
8029                   }else
8030                     if( argv[0] != null ){
8031                       id = argv[0]
8032                       if( id == 'f' ){
8033                         //e = document.getElementById('GJE_RootNode');
8034                         e = document.getElementById(id)
8035                       }else{
8036                         e = document.getElementById(id)
8037                       }else
8038                         if( e != null ){
8039                           text.value += e.outerHTML
8040                         }else{
8041                           text.value += "Not found: " + id
8042                         }
8043                       text.value += GJE_RootNode.outerHTML
8044                       //text.value += e.innerHTML
8045                     }else
8046                       if( cmd == 'destroy' ){
8047                         text.value += GJFactory_Destroy()
8048                       }else
8049                         if( cmd == 'save' ){
8050                           e = document.getElementById('GJFactory')
8051                           Permanent.setItem('GJFactory-1',e.innerHTML)
8052                           text.value += "-- Saved GJFactory"
8053                         }else
8054                           if( cmd == 'load' ){
8055                             gif = Permanent.getItem('GJFactory-1')
8056                             e = document.getElementById('GJFactory')
8057                             e.innerHTML = gif
8058                             // must restore EventListener
8059                             text.value += "-- EventListener was not restored"
8060                         }

```

```

8061     }else
8062     if( cmd.charAt(0) == '.' ){
8063         argv0 = args0.split(' ')
8064         text.value += GJE_NodeEdit(argv0)
8065     }else
8066     if( cmd == 'cont' ){
8067         bannerIsStopping = false
8068         GshMenuStop.innerHTML = "Stop"
8069     }else
8070     if( cmd == 'date' ){
8071         text.value += DateLong()
8072     }else
8073     if( cmd == 'echo' ){
8074         text.value += args
8075     }else
8076     if( cmd == 'fork' ){
8077         html_fork()
8078     }else
8079     if( cmd == 'last' ){
8080         text.value += MyHistory
8081         //h = document.createElement("span")
8082         //h.innerHTML = MyHistory
8083         //text.value += h.innerHTML
8084         //tx = MyHistory.replace("\n","");
8085         //text.value += tx.replace("<"+>br>","\n") + "xxxx<"+>br>yyyy"
8086     }else
8087     if( cmd == 'ne' ){
8088         text.value += GJE_NodeEdit(argv)
8089     }else
8090     if( cmd == 'reload' ){
8091         location.reload()
8092     }else
8093     if( cmd == 'mem' ){
8094         text.value += GJC_Memory('GJC_Storage',args,text)
8095     }else
8096     if( cmd == 'stop' ){
8097         bannerIsStopping = true
8098         GshMenuStop.innerHTML = "Start"
8099     }else
8100     if( cmd == 'who' ){
8101         text.value += "SessionId="+GJC_SessionId+" "+document.URL
8102     }else
8103     if( cmd == 'wall' ){
8104         text.value += GJC_Memory('GJC_Wall','write',text)
8105     }else
8106     {
8107         text.value += "Commands: help | echo | date | last \n"
8108         + '          new | save | load | mem \n'
8109         + '          who | wall | fork | nife'
8110     }
8111 }
8112
8113 function GJC_Input(){
8114     if( this.value.endsWith("\n") ){ // remove NL added by textarea
8115         this.value = this.value.slice(0,this.value.length-1)
8116     }
8117 }
8118
8119 var GCJ_Id = null
8120 function GJC_Resize(){
8121     GCJ_Id.style.zIndex = 20000
8122     GCJ_Id.style.width = window.innerWidth - 16
8123     GCJ_Id.style.height = 300
8124     GCJ_Id.style.backgroundColor = "rgba(0,64,16,1.0)" // blackboard color
8125     GCJ_Id.style.color = "rgba(255,255,255,1.0)"
8126 }
8127 function GJC_FocusIn(){
8128     this.spellCheck = false
8129     SuppressGJShell = true
8130     this.onkeydown = GJC_KeyDown
8131     GJC_Resize()
8132 }
8133 function GJC_FocusOut(){
8134     SuppressGJShell = false
8135     this.removeEventListener('keydown',GJC_KeyDown);
8136 }
8137 window.addEventListener('resize',GJC_Resize);
8138
8139 function GJC_OnStorage(e){
8140     //alert('Got Message')
8141     //GJC.value += "\n((ReceivedMessage))\n"
8142 }
8143 window.addEventListener('storage',GJC_OnStorage);
8144 //window.addEventListener('storage',()=>(alert('GotMessage')))
8145
8146 function GJC_Setup(gjcId){
8147     gjcid.style.width = gsh.getBoundingClientRect().width
8148     gjcid.value = "GJShell Console // " + GshVersion.innerHTML + "\n"
8149     //gjcid.value += "Date: " + DateLong() + "\n"
8150     gjcid.value += PS1
8151     gjcid.onfocus = GJC_FocusIn
8152     gjcid.addEventListener('input',GJC_Input);
8153     gjcid.addEventListener('focusout',GJC_FocusOut);
8154     GCJ_Id = gjcid
8155 }
8156 function GJC_Clear(id){
8157 }
8158 if( document.getElementById("GJC_0") != null ){
8159     GJC_Setup(GJC_0)
8160 }else{
8161     document.write('<'+textarea id="GJC_1" class="GJConsole"><'+/textarea>')
8162     GJC_Setup(GJC_1)
8163     factory = document.createElement('span');
8164     gsh.appendChild(factory)
8165     GJE_RootNode = factory;
8166     GJE_CurElement = GJE_RootNode;
8167 }
8168
8169 // TODO: focus handling
8170 </script>
8171 <style>
8172 .GJ_StyleEditor {
8173     font-size:9pt !important;
8174     font-family:Courier New, monospace !important;
8175 }
8176 </style>
8177
8178 <!-- ----- GJConsole END } ----- -->
8179 </span>
8180 */
8181
8182 // <details id="GshWebSocket" class="gsh-src"><summary>WebSocket</summary>
8183 // 2020-09-20 created
8184 // <a href="https://pkg.go.dev/golang.org/x/net/websocket">WS</a>

```

```
8185 // <a href="https://godoc.org/golang.org/x/net/websocket">WS</a>
8186 // INSTALL: go get golang.org/x/net/websocket
8187 // import "golang.org/x/net/websocket"
8188 const gshws_origin = "http://localhost:9999"
8189 const gshws_port = "localhost:9999"
8190 const gshws_path = "gshws"
8191 const gshws_url = "ws://" + gshws_port + "/" + gshws_path
8192 const GSHWS_MSGSIZE = (8*1024)
8193 func fmtstring(fmts string, params ...interface{})(string){
8194     return fmt.Sprintf(fmts,params...)
8195 }
8196 func GSHWS_MARK(what string)(string){
8197     now := time.Now()
8198     us := fmt.Sprintf("%06d",now.Nanosecond() / 1000)
8199     return "[" + now.Format(time.Stamp) + "." + us + "] --WS-- " + what + " : "
8200 }
8201 func gchk(what string,err error){
8202     if( err != nil ){
8203         panic(GSHWS_MARK(what)+err.Error())
8204     }
8205 }
8206 func glog(what string, fmts string, params ...interface{}{
8207     fmt.Println(GSHWS_MARK(what))
8208     fmt.Printf(fmts+"\n",params...)
8209 }
8210
8211 func servl(ws *websocket.Conn) {
8212     var reqb = make([]byte,GSHWS_MSGSIZE)
8213     for {
8214         rn, rerr := ws.Read(reqb)
8215         if( rerr != nil || rn < 0 ){
8216             glog("SQ",fmt.Sprintf("(%v,%v)",rn,rerr))
8217             break
8218         }
8219         req := string(reqb[0:rn])
8220         glog("SQ",fmt.Sprintf("(OK: %v",rn,req))
8221         res := fmtstring("OK: %v",req)
8222         wn, werr := ws.Write([]byte(res))
8223         gchk("SE",werr)
8224         glog("SR",fmtstring("(%v) %v",wn,string(res)))
8225     }
8226     glog("SF","WS response finish")
8227 }
8228 func ws_server(argv []string) {
8229     port := gshws_port
8230     glog("LS",fmtstring("listening at %v",gshws_url))
8231     http.Handle("/"+gshws_path,websocket.Handler(servl))
8232     err := http.ListenAndServe(port,nil)
8233     gchk("LE",err)
8234 }
8235
8236 func ws_client(argv []string) {
8237     glog("CS",fmtstring("connecting to %v",gshws_url))
8238     ws, err := websocket.Dial(gshws_url,"",gshws_origin)
8239     gchk("C",err)
8240
8241     var resb = make([]byte, GSHWS_MSGSIZE)
8242     for qi := 0; qi < 3; qi++{
8243         req := fmtstring("Hello, GShell! (%v)",qi)
8244         wn, werr := ws.Write([]byte(req))
8245         glog("QM",fmtstring("(%v) %v",wn,req))
8246         gchk("QE",werr)
8247         rn, rerr := ws.Read(resb)
8248         gchk("RE",rerr)
8249         glog("RM",fmtstring("(%v) %v",rn,string(resb)))
8250     }
8251     glog("CF","WS request finish")
8252 }
8253 // </details>
8254 /*
8255 <script>
8256 var ws = new WebSocket("ws://localhost:9999/gshws");
8257 console.log('ws:' + ws);
8258 ws.addEventListener('open', function(event){
8259     now = new Date().getTime() / 1000;
8260     msg = 'Hello, GSHELL in Golang! [' + now + ']';
8261     console.log([' + now + '] SendingToServer: ' + msg);
8262     ws.send(msg);
8263 });
8264 });
8265 ws.addEventListener('message', function(event){
8266     now = new Date().getTime() / 1000;
8267     console.log([' + now + '] ServerSays: ' + event.data);
8268 });
8269 </script>
8270 */
8271 /*//<br></span></html>
8272
```