

```

1 /*<html>
2 <span id="gsh" data-title="GShell" data-author="sato@its-more.jp">
3 <meta charset="UTF-8">
4 <meta name="viewport" content="width=device-width, initial-scale=1.0">
5 <link rel="icon" id="GshFaviconURL" href=""><!-- place holder -->
6 <span hidden="" id="GshVersion" data-title="GShell-0.4.8--2020-09-21--SatoxITS</span>
7 <header id="GshHeader" data-title="GShell" data-author="sato@its-more.jp">
8 <div align="right"><note><a href="http://archive.gshell.org">GShell</a> version 0.4.8 // 2020-09-21 // 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-WinId" 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="gsh-menuStop" 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 /*
30 <details id="GshStatement" class="gsh-document"><summary>Statement</summary>
31 <h3>Fun to create a shell</h3>
32 <p>For a programmer, it must be far easy and fun to create his own simple shell
33 rightly fitting to his favor and necessities, than learning existing shells with
34 complex full features that he never use.
35 I, as one of programmers, am writing this tiny shell for my own real needs,
36 totally from scratch, with fun.
37 </p><p>
38 For a programmer, it is fun to learn new computer languages. For long years before
39 writing this software, I had been specialized to C and early HTML2 :).
40 Now writing this software, I'm learning Go language, HTML5, JavaScript and CSS
41 on demand as a novice of these, with fun.
42 </p><p>
43 This single file "gsh.go", that is executable by Go, contains all of the code written
44 in Go. Also it can be displayed as "gsh.go.html" by browsers. It is a standalone
45 HTML file that works as the viewer of the code of itself, and as the "home page" of
46 this software.
47 </p><p>
48 Because this HTML file is a Go program, you may run it as a real shell program
49 on your computer.
50 But you must be aware that this program is written under situation like above.
51 Needless to say, there is no warranty for this program in any means.
52 </p>
53 <address>Aug 2020, SatoxITS (sato@its-more.jp)</address>
54 </details>
55 */
56 /*
57 <details id="GshFeatures" class="gsh-document"><summary>Features</summary><p>
58 </p>
59 <h3>Vi compatible command line editor</h3>
60 <p>
61 The command line of GShell can be edited with commands compatible with
62 <a href="https://www.washington.edu/computing/unix/vi.html"><b>vi</b></a>.
63 As in vi, you can enter <i><b>command mode</b></i> by <b>ESC</b> key,
64 then move around in the history by <b><code>j k / ? n N</code></b>,
65 or within the current line by <b><code>l h f w b 0 $ %</code></b> or so.
66 </p>
67 </details>
68 */
69 /*
70 <details id="gsh-gindex">
71 <summary>Index</summary><div class="gsh-src">
72 Documents
73   <span class="gsh-link" onclick="jumpto_JavaScriptView();">Command summary</span>
74 Go lang part<span class="gsh-src" onclick="document.getElementById('gsh-gocode').open=true;">
75   Package structures
76     <a href="#import">import</a>
77     <a href="#struct">struct</a>
78 Main functions
79   <a href="#comexpansion">str-expansion</a> // macro processor
80   <a href="#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 //<details id="gsh-gocode">
113 //<summary>Go Source</summary><div class="gsh-src" onclick="document.getElementById('gsh-gocode').open=false;">
114 // gsh - Go lang based Shell
115 // (c) 2020 ITS more Co., Ltd.
116 // 2020-0807 created by SatoxITS (sato@its-more.jp)
117 //
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 = "0.4.8"
185     DATE = "2020-09-21"
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     }
433     return leng,rstr
434 }
435 func escapeWhiteSP(str string)(string){
436     if len(str) == 0 {
437         return "\z" // empty, to be ignored
438     }
439     rstr := ""
440     for _,ch := range str {
441         switch ch {
442             case '\\': rstr += "\\\\"
443             case '\t': rstr += "\\s"
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 unescapeWhiteSPV(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("--D--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             }
675             line = line[olen:]
676             off += olen
677             remlen -= olen
678         }
679         if strRep { os.Stdout.Write([]byte("\r\n")) }
680     }
681     fmt.Fprintf(os.Stderr,"--I-- lnsp %d to %d\n",ni,toi)
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,npadv,argv)
1032     }
1033 }
1034 }
1035 }
1036 return total
1037 }
1038 func (gsh*GshContext)xxFind(depth int,total *fileSum,dir string,npadv[]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,npadv,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_SIZE) != 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, nlend 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 nlend { 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, nlend bool){
1354     for ai := 1; ai < len(argv); ai++ {
1355         if 1 < ai {
1356             fmt.Printf(" ");
1357         }
1358         arg := argv[ai]
1359         found, val := getval(arg)
1360         if found {
1361             fmt.Printf("%d",val)
1362         }else{
1363             fmt.Printf("%s",arg)
1364         }
1365     }
1366 }

```

```

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 %d-%d %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
1504     gshPA.Files[fdix] = savfd
1505     return pin,pout,false
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     //fmt.Printf("Sleep %v\n",duration)
1579     time.Sleep(d)
1580     if 0 < len(argv[2:]) {
1581         gshCtx.gshellv(argv[2:])
1582     }
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 Done := time.Now()
1760 Elps := float64(Done.Sub(Start))/1000000000 //Seconds
1761 TotalMB := float64(total)/1000000 //MB
1762 MBps := TotalMB / Elps
1763 fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v) %v %.3fMB/s\n",
1764                         what,total,size,nio,absize(total),MBps)
1765 return total
1766 }
1767 func tcpPush(clnt *os.File){
1768     // shrink socket buffer and recover
1769     usleep(100);
1770 }
1771 }
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         }
1868     }
1869 }
```

```

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,"%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     }if 2 < len(argv) && argv[2] != "" {
2075         ofile = argv[2]
2076         //fmt.Printf("(%)v B.ofile=%v\n",len(argv),argv,ofile)
2077     }
2078     //fmt.Printf(Elapsed(Start)+"--I-- Rex out(%v)\n",ofile)
2079     fmt.Printf(Elapsed(Start)+"--In- PUT %v (%v)\n",dszie,bsize)
2080     req = fmt.Sprintf("PUT %v %v \r\n",dszie,ofile)
2081     if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
2082     fmt.Fprintf(serv,"%v",req)
2083     count,err = serv.Read(res)
2084     if debug { fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res[0:count])) }
2085     fileRelay("SendPUT",local,remote,dszie,bsize)
2086 }else{
2087     req = fmt.Sprintf("%v\r\n",strings.Join(argv, " "))
2088     if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
2089     fmt.Fprintf(serv,"%v",req)
2090     //fmt.Printf("--In- sending RexRequest(%v)\n",len(req))
2091 }
2092 //fmt.Printf(Elapsed(Start)+"--In- waiting RexResponse...\r\n")
2093 count,err = serv.Read(res)
2094 ress := ""
2095 if count == 0 {
2096     ress = "(nil)\r\n"
2097 }else{
2098     ress = string(res[:count])
2099 }
2100 if err != nil {
2101     fmt.Printf(Elapsed(Start)+"--En- S: (%d,%v) %v",count,err,ress)
2102 }else{
2103     fmt.Printf(Elapsed(Start)+"--In- S: %v",ress)
2104 }
2105 serv.Close()
2106 //conn.Close()
2107

```

```

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 1 < 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\\t",v.WorkDirX)
2526                 fmt.Printf("%v %11v/t ",start,elps)
2527             }
2528             if isin("-l",argv) && !isin("-lo",argv){
2529                 fmt.Printf("%v",Rusagef("%u\\t/%s",argv,v.Rusage))
2530             }
2531             if isin("-at",argv) { // isin("-ls",argv){
2532                 dhi := v.WorkDirX // workdir history index
2533                 fmt.Printf("%d %v\\t",dhi,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("%d ",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     chdist := 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(ru1, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
2661     TimeValSub(&ru1[0].Utime,&ru2[0].Utime)
2662     TimeValSub(&ru1[0].Stime,&ru2[0].Stime)
2663     TimeValSub(&ru1[1].Utime,&ru2[1].Utime)
2664     TimeValSub(&ru1[1].Stime,&ru2[1].Stime)
2665     return ru1
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(ru1, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
2673     TimeValAdd(ru1[0].Utime,ru2[0].Utime)
2674     TimeValAdd(ru1[0].Stime,ru2[0].Stime)
2675     TimeValAdd(ru1[1].Utime,ru2[1].Utime)
2676     TimeValAdd(ru1[1].Stime,ru2[1].Stime)
2677     return ru1
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 ruv = [2]syscall.Rusage{}
2703     syscall.Getrusage(syscall.RUSAGE_SELF,&ruv[0])
2704     syscall.Getrusage(syscall.RUSAGE_CHILDREN,&ruv[1])
2705     return ruv
2706 }
2707 func showRusage(what string,argv []string, ru *syscall.Rusage){
2708     fmt.Printf(":@: ",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.Printf("--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
2791
2792     ffix := 0
2793     mode := os.O_RDONLY
2794
2795     switch {
2796     case cmd == "-i" || cmd == "<":
2797         ffix = 0
2798         mode = os.O_RDONLY
2799     case cmd == "-o" || cmd == ">":
2800         ffix = 1
2801         mode = os.O_RDWR | os.O_CREATE
2802     case cmd == "-a" || cmd == ">>":
2803         ffix = 1
2804         mode = os.O_RDWR | os.O_CREATE | os.O_APPEND
2805     }
2806     if fname[0] == '#' {
2807         fd, err := strconv.Atoi(fname[1:])
2808         if err != nil {
2809             fmt.Printf("--E-- (%v)\n",err)
2810             return false
2811         }
2812         file = os.NewFile(uintptr(fd),"MaybePipe")
2813     }else{
2814         xfile, err := os.OpenFile(argv[1], mode, 0600)
2815         if err != nil {
2816             fmt.Printf("--E-- (%s)\n",err)
2817             return false
2818         }
2819         file = xfile
2820     }
2821     gshPA := gshCtx.gshPA
2822     savfd := gshPA.Files[ffix]
2823     gshPA.Files[ffix] = file.Fd()
2824     fmt.Printf("--I-- Opened %d %s\n",file.Fd(),argv[1])
2825     gshCtx.gshellv(argv[2:])
2826     gshPA.Files[ffix] = savfd
2827
2828     return false
2829 }
2830
2831 //fmt.Fprintf(res, "GShell Status: %q", html.EscapeString(req.URL.Path))
2832 func httpHandler(res http.ResponseWriter, req *http.Request){
2833     path := req.URL.Path
2834     fmt.Printf("--I-- Got HTTP Request(%s)\n",path)
2835     {
2836         gshCtxBuf, _ := setupGshContext()
2837         gshCtx := &gshCtxBuf
2838         fmt.Printf("--I-- %s\n",path[1:])
2839         gshCtx.tgshelll(path[1:])
2840     }
2841     fmt.Fprintf(res, "Hello(^_~)//\n%s\n",path)
2842 }
2843 func (gshCtx *GshContext) httpServer(argv []string){
2844     http.HandleFunc("/", httpHandler)
2845     accport := "localhost:9999"
2846     fmt.Printf("--I-- HTTP Server Start at [%s]\n",accport)
2847     http.ListenAndServe(accport,nil)
2848 }
2849
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="scanc">Scanc</a> // string decomposer
2930 // scanc [format] [input]
2931 func scanc(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)Scavv(argv[]string){
3030     //fmt.Printf("--D-- Scavv(%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         return
3057     }
3058     //fmt.Printf("--D-- Printv(%v)\n",argv)
3059     //fmt.Printf("%v\n",strings.Join(gsh.iValues,","))
3060     div := gsh.iDelimiter
3061     fmts := ""
3062     argv = argv[1:]
3063     if 0 < len(argv) {
3064         if strBegins(argv[0],"-F") {
3065             div = argv[0][2:]
3066             argv = argv[1:]
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 == "-iu":
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.Printv(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.Scav(argv)
3277 case cmd == "set":
3278     // set name ...
3279 case cmd == "serv":
3280     gshCtx.httpServer(argv)
3281 case cmd == "shift":
3282     gshCtx.Shiftv(argv)
3283 case cmd == "sleep":
3284     gshCtx.sleep(argv)
3285 case cmd == "-sort":
3286     gshCtx.Sortv(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 case cmd == "gj" && l < len(argv) && argv[1] == "listen":
3319     go gj_server(argv[1:]);
3320 case cmd == "gj" && l < len(argv) && argv[1] == "join":
3321     go gj_client(argv[1:]);
3322 case cmd == "gj":
3323     jsend(argv);
3324 case cmd == "jsend":
3325     jsend(argv);
3326 default:
3327     if gshCtx.whichPlugin(cmd,[]string{"-s"}) != nil {
3328         gshCtx.xPlugin(argv)
3329     }else{
3330         notfound,_ := gshCtx.excommand(false,argv)
3331         if notfound {
3332             fmt.Printf("---E--- command not found (%v)\n",cmd)
3333         }
3334     }
3335 }
3336 return fin
3337 }
3338
3339 func (gsh*GshContext)gshell(gline string) (rfin bool) {
3340     argv := strings.Split(string(gline), " ")
3341     fin := gsh.gshellv(argv)
3342     return fin
3343 }
3344 func (gsh*GshContext)tgshell(gline string)(xfin bool){
3345     start := time.Now()
3346     fin := gsh.gshell(gline)
3347     end := time.Now()
3348     elps := end.Sub(start);

```

```

3349     if gsh.CmdTime {
3350         fmt.Printf("--T-- " + time.Now().Format(time.Stamp) + "(%d.%09ds)\n",
3351             elps/1000000000,elps%100000000)
3352     }
3353     return fin
3354 }
3355 func Ttyid() (int) {
3356     fi, err := os.Stdin.Stat()
3357     if err != nil {
3358         return 0;
3359     }
3360     //fmt.Printf("Stdin: %v Dev=%d\n",
3361     // fi.Mode(),fi.Mode()&os.ModeDevice)
3362     if (fi.Mode() & os.ModeDevice) != 0 {
3363         stat := syscall.Stat_t{};
3364         err := syscall.Fstat(0,&stat)
3365         if err != nil {
3366             //fmt.Printf("--I-- Stdin: (%v)\n",err)
3367         }else{
3368             //fmt.Printf("--I-- Stdin: rdev=%d %d\n",
3369             // stat.Rdev&0xFF,stat.Rdev);
3370             //fmt.Printf("--I-- Stdin: tty%d\n",stat.Rdev&0xFF);
3371             return int(stat.Rdev & 0xFF)
3372         }
3373     }
3374     return 0
3375 }
3376 func (gshCtx *GshContext) ttysize() string {
3377     //fmt.Printf("--I-- GSH_HOME=%s\n",gshCtx.GshHomeDir)
3378     ttysize := gshCtx.GshHomeDir + "/" + "gsh-tty" +
3379     fmt.Sprintf("%02d",gshCtx.TerminalId)
3380     //strconv.Itoa(gshCtx.TerminalId)
3381     //fmt.Printf("--I-- ttysize=%s\n",ttysize)
3382     return ttysize
3383 }
3384 func (gshCtx *GshContext) ttysize(*os.File){
3385     file, err := os.OpenFile(gshCtx.ttysize(),os.O_RDWR|os.O_CREATE|os.O_TRUNC,0600)
3386     if err != nil {
3387         fmt.Printf("--F-- cannot open %s (%s)\n",gshCtx.ttysize(),err)
3388         return file;
3389     }
3390     return file
3391 }
3392 func (gshCtx *GshContext)getline(hix int, skipping bool, prevline string) (string) {
3393     if( skipping ){
3394         reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
3395         line, _, _ := reader.ReadLine()
3396         return string(line)
3397     }else
3398     if true {
3399         return xgetline(hix,prevline,gshCtx)
3400     }/*
3401     else
3402     if( with_exgetline && gshCtx.GetLine != "" ){
3403         //var hix int64 = int64(hix); // cast
3404         newenv := os.Environ()
3405         newenv = append(newenv, "GSH_LINENO="+strconv.FormatInt(int64(hix),10) )
3406
3407         tty := gshCtx.ttyline()
3408         tty.WriteString(prevline)
3409         Pa := os.ProcAttr {
3410             "", // start dir
3411             newenv, //os.Environ(),
3412             []*os.File{os.Stdin,os.Stdout,os.Stderr,tty},
3413             nil,
3414         }
3415     }
3416     //fmt.Printf("--I-- getline=%s // %s\n",gsh_getlinev[0],gshCtx.GetLine)
3417     proc, err := os.StartProcess(gsh_getlinev[0],[]string{"getline","getline"},&Pa)
3418     if err != nil {
3419         fmt.Printf("--F-- getline process error (%v)\n",err)
3420         // for ; ; { }
3421         return "exit (getline program failed)"
3422     }
3423     //stat, err := proc.Wait()
3424     proc.Wait()
3425     buff := make([]byte,LINESIZE)
3426     count, err := tty.Read(buff)
3427     // , err = tty.Read(buff)
3428     //fmt.Printf("--D-- getline (%d)\n",count)
3429     if err != nil {
3430         if !(count == 0) { // && err.String() == "EOF" }
3431             fmt.Printf("--E-- getline error (%s)\n",err)
3432         }
3433     }else{
3434         //fmt.Printf("--I-- getline OK \"%s\"\n",buff)
3435     }
3436     tty.Close()
3437     gline := string(buff[0:count])
3438     return gline
3439 }
3440 */
3441 {
3442     // if isatty {
3443         fmt.Printf("!%d",hix)
3444         fmt.Print(PROMPT)
3445     //}
3446     reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
3447     line, _, _ := reader.ReadLine()
3448     return string(line)
3449 }
3450 }
3451 /**
3452 * getline.c
3453 * 2020-0819 extracted from dog.c
3454 * getline.go
3455 * 2020-0822 ported to Go
3456 */
3457 /*
3458 */
3459 package main // getline main
3460 import (
3461     "fmt" // <a href="https://golang.org/pkg/fmt/">fmt</a>
3462     "strings" // <a href="https://golang.org/pkg/strings/">strings</a>
3463     "os" // <a href="https://golang.org/pkg/os/">os</a>
3464     "syscall" // <a href="https://golang.org/pkg/syscall/">syscall</a>
3465     //"bytes" // <a href="https://golang.org/pkg/os/">os</a>
3466     //"os/exec" // <a href="https://golang.org/pkg/os/">os</a>
3467     //<a href="https://golang.org/pkg/os/">os</a>
3468 )
3469 */
3470
3471 // C language compatibility functions
3472 var errno = 0

```

```

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

```

```

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

```

```

3721     if 0 < len(argv) {
3722         arg1 := argv[0]
3723         if arg1[0] == '-' {
3724             switch arg1 {
3725                 default: fmt.Printf("--Ed-- Unknown option(%v)\n",arg1)
3726                 return
3727                 case "-b": builtin = true
3728                 case "-d": debug = true
3729                 case "-s": silent = true
3730                 case "-v": info = true
3731             }
3732             opt = arg1
3733             argv = argv[1:]
3734         }
3735     }
3736 }
3737
3738 dicName := ""
3739 dicURL := ""
3740 if 0 < len(argv) {
3741     arg = argv[0]
3742     dicName = arg
3743     argv = argv[1:]
3744 }
3745 if 0 < len(argv) {
3746     dicURL = argv[0]
3747     argv = argv[1:]
3748 }
3749 if false {
3750     fprintf(stderr,"--Dd-- com(%v) opt(%v) arg(%v)\n",cmd,opt,arg)
3751 }
3752 if cmd == DIC_COM_LOAD {
3753     //dicType := ""
3754     dicBody := ""
3755     if !builtin && dicName != "" && dicURL == "" {
3756         f,err := os.Open(dicName)
3757         if err == nil {
3758             dicURL = dicName
3759         }else{
3760             f,err = os.Open(dicName+".html")
3761             if err == nil {
3762                 dicURL = dicName+".html"
3763             }else{
3764                 f,err = os.Open("gshdic-"+dicName+".html")
3765                 if err == nil {
3766                     dicURL = "gshdic-"+dicName+".html"
3767                 }
3768             }
3769         }
3770         if err == nil {
3771             var buf = make([]byte,128*1024)
3772             count,err := f.Read(buf)
3773             f.Close()
3774             if info {
3775                 fprintf(stderr,"--Id-- ReadDic(%v,%v)\n",count,err)
3776             }
3777             dicBody = string(buf[0:count])
3778         }
3779     }
3780     if dicBody == "" {
3781         switch arg {
3782             default:
3783                 dicName = "WorldDic"
3784                 dicURL = WorldDic
3785                 if info {
3786                     fprintf(stderr,"--Id-- default dictionary \"%v\"\n",
3787                         dicName);
3788             }
3789             case "wnn":
3790                 dicName = "WnnDic"
3791                 dicURL = WnnDic
3792             case "sumomo":
3793                 dicName = "SumomoDic"
3794                 dicURL = SumomoDic
3795             case "sijimi":
3796                 dicName = "SijimiDic"
3797                 dicURL = SijimiDic
3798             case "jkl":
3799                 dicName = "JKLJaDic"
3800                 dicURL = JA_JKLDic
3801         }
3802         if debug {
3803             fprintf(stderr,"--Id-- %v URL=%v\n\n",dicName,dicURL);
3804         }
3805         dicv := strings.Split(dicURL,",")
3806         if debug {
3807             fprintf(stderr,"--Id-- %v encoded data...\n",dicName)
3808             fprintf(stderr,"Type: %v\n",dicv[0])
3809             fprintf(stderr,"Body: %v\n",dicv[1])
3810             fprintf(stderr,"\\n")
3811         }
3812         body,_ := base64.StdEncoding.DecodeString(dicv[1])
3813         dicBody = string(body)
3814     }
3815     if info {
3816         fmt.Printf("--Id-- %v %v\n",dicName,dicURL)
3817         fmt.Printf("%s\n",dicBody)
3818     }
3819     if debug {
3820         fprintf(stderr,"--Id-- dicName %v text...\n",dicName)
3821         fprintf(stderr,"%v\n",string(dicBody))
3822     }
3823     entv := strings.Split(dicBody,"\n");
3824     if info {
3825         fprintf(stderr,"--Id-- %v scan...\n",dicName);
3826     }
3827     var added int = 0
3828     var dup int = 0
3829     for i,v := range entv {
3830         var pat string
3831         var out string
3832         fmt.Sscanf(v,"%s %s",&pat,&out)
3833         if len(pat) <= 0 {
3834            }else{
3835                 if 0 <= isinDic(pat) {
3836                     dup += 1
3837                     continue
3838                 }
3839                 romkana[dicents] = RomKana{dicName,pat,out,0}
3840                 dicents += 1
3841                 added += 1
3842                 Romkan = append(Romkan,RomKana{dicName,pat,out,0})
3843                 if debug {
3844                     fmt.Printf("[%3v]:[%2v]-%v [%2v]%v\n",
3845

```

```

3845             i,len(pat),pat,len(out),out)
3846         }
3847     }
3848   }
3849   if !silent {
3850     url := dicURL
3851     if strBegins(url,"data:") {
3852       url = "builtin"
3853     }
3854     fprintf(stderr,"--Id-- %v scan... %v added, %v dup. / %v total (%v)\n",
3855             dicName,added,dup,len(Romkan),url);
3856   }
3857   // should sort by pattern length for concrete match, for performance
3858   if debug {
3859     arg = "" // search pattern
3860     dump = true
3861   }
3862 }
3863 if cmd == DIC_COM_DUMP || dump {
3864   fprintf(stderr, "--Id-- %v dump... %v entries:\n",dicName,len(Romkan));
3865   var match = 0
3866   for i := 0; i < len(Romkan); i++ {
3867     dic := Romkan[i].dic
3868     pat := Romkan[i].pat
3869     out := Romkan[i].out
3870     if arg == "" || 0 <= strings.Index(pat,arg)||0 <= strings.Index(out,arg) {
3871       fmt.Printf("\\\\%v\\t%v [%2v]@%v\n",
3872             i,dic,len(pat),pat,len(out),out)
3873       match += 1
3874     }
3875   }
3876   fprintf(stderr,"--Id-- %v matched %v / %v entries:\n",arg,match,len(Romkan));
3877 }
3878 }
3879 func loadDefaultDic(dic int){
3880   if( 0 < len(Romkan) ){
3881     return
3882   }
3883   //fprintf(stderr,"\r\n")
3884   xDic([string("dic"),DIC_COM_LOAD]);
3885
3886   var info = false
3887   if info {
3888     fprintf(stderr,"--Id-- Conguratuations!! WorldDic is now activated.\r\n")
3889     fprintf(stderr,"--Id-- enter \"dic\" command for help.\r\n")
3890   }
3891 }
3892 func readDic()(int){
3893   /*
3894   var rk *os.File;
3895   var dic = "MyIME-dic.txt";
3896   //rk = fopen("romkana.txt","r");
3897   //rk = fopen("JK-JA-morse-dic.txt","r");
3898   rk = fopen(dic,"r");
3899   if( rk == NULL_FPP ){
3900     if( true ){
3901       fprintf(stderr,"--%s-- Could not load %s\n",MyIMEVER,dic);
3902     }
3903     return -1;
3904   }
3905   if( true ){
3906     var di int;
3907     var line = make(StrBuff,1024);
3908     var pat string
3909     var out string
3910     for di = 0; di < 1024; di++ {
3911       if( fgets(line,sizeof(line),rk) == NULLSP ){
3912         break;
3913       }
3914       fmt.Sscanf(string(line[0:strlen(line)]),"s s",&pat,&out);
3915       //sscanf(line,"%[^\\r\\n]",&pat,&out);
3916       romkana[di].pat = pat;
3917       romkana[di].out = out;
3918       //fprintf(stderr,"--Dd- %-10s %s\n",pat,out)
3919     }
3920     dicents += di
3921     if( false ){
3922       fprintf(stderr,"--%s-- loaded romkana.txt (%d)\n",MyIMEVER,di);
3923       for di = 0; di < dicents; di++ {
3924         fprintf(stderr,
3925           "%s %s\n",romkana[di].pat,romkana[di].out);
3926       }
3927     }
3928   }
3929   fclose(rk);
3930
3931   //romkana[dicents].pat = "//ddump"
3932   //romkana[dicents].pat = "//ddump" // dump the dic. and clean the command input
3933   */
3934   return 0;
3935 }
3936 func matchlen(stri string, pati string)(int){
3937   if strBegins(stri,pati) {
3938     return len(pati)
3939   }else{
3940     return 0
3941   }
3942 }
3943 func convs(src string)(string){
3944   var si int;
3945   var sx = len(src);
3946   var di int;
3947   var mi int;
3948   var dstb []byte
3949
3950   for si = 0; si < sx; { // search max. match from the position
3951     if strBegins(src[si:], "%x/") {
3952       // %x/integer // s/a/b/
3953       ix := strings.Index(src[si+3:], "/")
3954       if 0 < ix {
3955         var iv int = 0
3956         //fmt.Sscanf(src[si+3:si+3+ix],"d",&iv)
3957         fmt.Sscanf(src[si+3:si+3+ix],"v",&iv)
3958         sval := fmt.Sprintf("%x",iv)
3959         bval := []byte(sval)
3960         dstb = append(dstb,bval...)
3961         si = si+3+ix+1
3962         continue
3963       }
3964     }
3965     if strBegins(src[si:], "%d/") {
3966       // %d/integer // s/a/b/
3967       ix := strings.Index(src[si+3:], "/")
3968       if 0 < ix {
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
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

```

```

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

```

```

4093     for _,a := range argv { // tempo
4094         if strBegins(a,"x") {
4095             fmt.Sscanf(a[1],"%f",&xtempo)
4096             tempo = 1 / xtempo
4097             //fprintf(stderr,"--Dr-- tempo=%v\n",a[2:],tempo);
4098         }else
4099         if strBegins(a,"r") { // repeat
4100             fmt.Sscanf(a[1],"%v",&repeat)
4101         }else
4102         if strBegins(a,"!") {
4103             fmt.Sscanf(a[1],"%d",&hix)
4104         }else{
4105             fmt.Sscanf(a,"%d",&hix)
4106         }
4107     }
4108     if hix == 0 || len(argv) <= 1 {
4109         hix = len(gsh.CommandHistory)-1
4110     }
4111     fmt.Printf("--Ir-- Replay(!v x v r v)\n",hix,xtempo,repeat)
4112     //dumpEvents(hix)
4113     //gsh.xScanReplay(hix,false,repeat,tempo,argv)
4114     go gsh.xScanReplay(hix,true,repeat,tempo,argv)
4115 }
4116 }
4117
4118 // <a href="https://golang.org/pkg/syscall/#FdSet">syscall.Select</a>
4119 // 2020-0827 GShell-0.2.3
4120 /*
4121 func FpollInl(fp *os.File,usec int)(uintptr){
4122     nfd := 1
4123
4124     rdv := syscall.FdSet {}
4125     fd1 := fp.Fd()
4126     bank1 := fd1/32
4127     mask1 := int32(1 << fd1)
4128     rdv.Bits[bank1] = mask1
4129
4130     fd2 := -1
4131     bank2 := -1
4132     var mask2 int32 = 0
4133
4134     if 0 <= EventRecvFd {
4135         fd2 = EventRecvFd
4136         nfd = fd2 + 1
4137         bank2 = fd2/32
4138         mask2 = int32(1 << fd2)
4139         rdv.Bits[bank2] |= mask2
4140         //fmt.Printf("--De-- EventPoll mask added %d[%v][%v]\n",fd2,bank2,mask2)
4141     }
4142
4143     tout := syscall.NsecToTimeval(int64(usec*1000))
4144     //n,err := syscall.Select(nfd,&rdv,nil,nil,&tout) // spec. mismatch
4145     err := syscall.Select(nfd,&rdv,nil,nil,&tout)
4146     if err != nil {
4147         //fmt.Printf("--De-- select() err(%v)\n",err)
4148     }
4149     if err == nil {
4150         if 0 <= fd2 && (rdv.Bits[bank2] & mask2) != 0 {
4151             if false {
4152                 fmt.Printf("--De-- got Event\n")
4153             }
4154             return uintptr(EventFdOffset + fd2)
4155         }else
4156         if (rdv.Bits[bank1] & mask1) != 0 {
4157             return uintptr(NormalFdOffset + fd1)
4158         }else{
4159             return 1
4160         }
4161     }else{
4162         return 0
4163     }
4164 }
4165 */
4166 func fgetctimeoutl(fp *os.File,usec int)(int){
4167     READ1:
4168     //readyFd := FpollInl(fp,usec)
4169     readyFd := CFpollInl(fp,usec)
4170     if readyFd < 100 {
4171         return EV_TIMEOUT
4172     }
4173
4174     var buf [1]byte
4175
4176     if EventFdOffset <= readyFd {
4177         fd := int(readyFd-EventFdOffset)
4178         _,err := syscall.Read(fd,buf[0:1])
4179         if( err != nil ){
4180             return EOF;
4181         }else{
4182             if buf[0] == EV_MODE {
4183                 recvEvent(fd)
4184                 goto READ1
4185             }
4186             return int(buf[0])
4187         }
4188     }
4189
4190     _err := fp.Read(buf[0:1])
4191     if( err != nil ){
4192         return EOF;
4193     }else{
4194         return int(buf[0])
4195     }
4196 }
4197
4198 func visibleChar(ch int)(string){
4199     switch {
4200         case '!' <= ch && ch <= '~':
4201             return string(ch)
4202     }
4203     switch ch {
4204         case '\r': return "\\s"
4205         case '\n': return "\\n"
4206         case '\r': return "\\r"
4207         case '\t': return "\\t"
4208     }
4209     switch ch {
4210         case 0x00: return "NUL"
4211         case 0x07: return "BEL"
4212         case 0x08: return "BS"
4213         case 0x0E: return "SO"
4214         case 0x0F: return "SI"
4215         case 0x1B: return "ESC"
4216         case 0x7F: return "DEL"

```

```

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

```

```

4341     iin.pch = iin.pch[1:];
4342 }else{
4343     ch1 = fgetcTimeout(stdin,timeOutUs);
4344 }
4345 if( ch1 == 033 ){ // escape sequence
4346     ch2 = fgetcTimeout(stdin,EscTimeOut);
4347     if( ch2 == EV_TIMEOUT ){
4348     }else{
4349         ch3 = fgetcTimeout(stdin,EscTimeOut);
4350         if( ch3 == EV_TIMEOUT ){
4351             iin.pch = append(iin.pch,ch2) // enQ
4352         }else{
4353             switch( ch2 ){
4354                 default:
4355                     iin.pch = append(iin.pch,ch2) // enQ
4356                     iin.pch = append(iin.pch,ch3) // enQ
4357                 case '[':
4358                     switch( ch3 ){
4359                         case 'A': ch1 = GO_UP; // ^
4360                         case 'B': ch1 = GO_DOWN; // v
4361                         case 'C': ch1 = GO_RIGHT; // >
4362                         case 'D': ch1 = GO_LEFT; // <
4363                         case '3':
4364                             ch4 := fgetcTimeout(stdin,EscTimeOut);
4365                             if( ch4 == '~' ){
4366 //fprintf(stderr,"%02X %02X %02X %02X\n",ch1,ch2,ch3,ch4);
4367                             ch1 = DEL_RIGHT
4368
4369                         }
4370                         case '\\':
4371                             ch4 := fgetcTimeout(stdin,EscTimeOut);
4372 //fprintf(stderr,"%02X %02X %02X %02X\n",ch1,ch2,ch3,ch4);
4373                             switch( ch3 ){
4374                                 case '~': ch1 = DEL_RIGHT
4375                             }
4376                         }
4377                     }
4378                 }
4379             }
4380         return ch1
4381     }
4382 func (iin*IInput)clearline(){
4383     var i int
4384     fprintf(stderr,"\r");
4385     // should be ANSI ESC sequence
4386     for i = 0; i < TtyMaxCol; i++ { // to the max. position in this input action
4387         fputc(' ',os.Stderr);
4388     }
4389     fprintf(stderr,"\r");
4390 }
4391 func (iin*IInput)Redraw(){
4392     redraw(iin,iin.lno,iin.line,iin.right)
4393 }
4394 func redraw(iin *IInput,lno int,line string,right string){
4395     inMeta := false
4396     showMode := ""
4397     showMeta := "" // visible Meta mode on the cursor position
4398     showLino := fmt.Sprintf("!%d! ",lno)
4399     InsertMark := "" // in visible insert mode
4400
4401     if MODE_VicMode {
4402     }else
4403     if 0 < len(iin.right) {
4404         InsertMark = " "
4405     }
4406
4407     if( 0 < len(iin.waitingMeta) ){
4408         inMeta = true
4409         if iin.waitingMeta[0] != 033 {
4410             showMeta = iin.waitingMeta
4411         }
4412     }
4413     if( romkanmode ){
4414         //romkanmark = " *";
4415     }else{
4416         //romkanmark = "";
4417     }
4418     if MODE_ShowMode {
4419         romkan := "--"
4420         inmeta := "_"
4421         inveri := ""
4422         if MODE_CapsLock {
4423             inmeta = "A"
4424         }
4425         if MODE_LowerLock {
4426             inmeta = "a"
4427         }
4428         if MODE_ViTrace {
4429             inveri = "v"
4430         }
4431         if MODE_VicMode {
4432             inveri = ":"}
4433     }
4434     if romkanmode {
4435         romkan = "\343\201\202"
4436         if MODE_CapsLock {
4437             inmeta = "R"
4438         }else{
4439             inmeta = "r"
4440         }
4441     }
4442     if inMeta {
4443         inmeta = "\\"
4444     }
4445     showMode = "["+romkan+inmeta+inveri+"]";
4446 }
4447 Pre := "\r" + showMode + showLino
4448 Output := ""
4449 Left := ""
4450 Right := ""
4451 if romkanmode {
4452     Left = convs(line)
4453     Right = InsertMark+convs(right)
4454 }else{
4455     Left = line
4456     Right = InsertMark+right
4457 }
4458 Output = Pre+Left
4459 if MODE_ViTrace {
4460     Output += iin.LastCmd
4461 }
4462 Output += showMeta+Right
4463 for len(Output) < TtyMaxCol { // to the max. position that may be dirty
4464     Output += " "

```

```

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

```

```

4589         i -= lastSize
4590         continue
4591     }
4592     break
4593   } for 0 < i {
4594     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4595     if lastSize <= 0 { continue } // not the character top
4596     if !isAlnum(lastRune) { // character, type, or string to be searched
4597       found = i
4598       break
4599     }
4600   } i -= lastSize
4601 } if found < 0 && i == 0 {
4602   found = 0
4603 }
4604 if 0 <= found {
4605   if isAlnum(lastRune) { // or non-kana character
4606     }else{ // when positioning to the top o the word
4607       i += lastSize
4608     }
4609   iin.right = str[i:] + iin.right
4610   if 0 < i {
4611     iin.line = str[0:i]
4612   }else{
4613     iin.line = ""
4614   }
4615 }
4616 //}
4617 //fmt.Printf("\n%d[%s][%s]\n",i,found,iin.line,iin.right)
4618 //fmt.Println("") // set debug messae at the end of line
4619 }
4620 //}
4621 // 0.2.8 2020-0901 created
4622 func (iin*IInput)GotoENDW(){
4623   str := iin.right
4624   if len(str) <= 0 {
4625     return
4626   }
4627   lastSize := 0
4628   var lastRune rune
4629   var lastW = 0
4630   i := 0
4631   inWord := false
4632
4633   lastRune,lastSize = utf8.DecodeRuneInString(str[0:])
4634   if isAlnum(lastRune) {
4635     r,z := utf8.DecodeRuneInString(str[lastSize:])
4636     if 0 < z && isAlnum(r) {
4637       inWord = true
4638     }
4639   }
4640   for i < len(str) {
4641     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4642     if lastSize <= 0 { break } // broken data?
4643     if !isAlnum(lastRune) { // character, type, or string to be searched
4644       break
4645     }
4646     lastW = i // the last alnum if in alnum word
4647     i += lastSize
4648   }
4649   if inWord {
4650     goto DISP
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     i += lastSize
4659   }
4660   for i < len(str) {
4661     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4662     if lastSize <= 0 { break } // broken data?
4663     if !isAlnum(lastRune) { // character, type, or string to be searched
4664       break
4665     }
4666     lastW = i
4667     i += lastSize
4668   }
4669 DISP:
4670   if 0 < lastW {
4671     iin.line = iin.line + str[0:lastW]
4672     iin.right = str[lastW:]
4673   }
4674 //fmt.Printf("\n%d[%s][%s]\n",i,iin.line,iin.right)
4675 //fmt.Println("") // set debug messae at the end of line
4676 }
4677 //}
4678 // 0.2.8 2020-0901 created
4679 func (iin*IInput)GotoNEXTW(){
4680   str := iin.right
4681   if len(str) <= 0 {
4682     return
4683   }
4684   lastSize := 0
4685   var lastRune rune
4686   var found = -1
4687   i := 1
4688   for i < len(str) {
4689     lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4690     if lastSize <= 0 { break } // broken data?
4691     if !isAlnum(lastRune) { // character, type, or string to be searched
4692       found = i
4693       break
4694     }
4695     i += lastSize
4696   }
4697   if 0 < found {
4698     if isAlnum(lastRune) { // or non-kana character
4699     }else{ // when positioning to the top o the word
4700       found += lastSize
4701     }
4702     iin.line = iin.line + str[0:found]
4703     if 0 < found {
4704       iin.right = str[found:]
4705     }else{
4706       iin.right = ""
4707     }
4708 //fmt.Printf("\n%d[%s][%s]\n",i,iin.line,iin.right)
4709 //fmt.Println("") // set debug messae at the end of line
4710 }
4711 //}
4712 func (iin*IInput)GotoPAIRCH(){

```

```

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

```

```

4837     found = iin.SearchBackward(pat)
4838   }
4839   if !found {
4840     found,line := gsh.SearchHistory(pat,false)
4841     if found {
4842       iin.line = line
4843       iin.right = ""
4844     }
4845   }
4846 }
4847 func (iin*IInput)getstring1(prompt string)(string){ // should be editable
4848   iin.clearline();
4849   fprintf(stderr,"\r%v",prompt)
4850   str := ""
4851   for {
4852     ch := iin.Getc(10*1000*1000)
4853     if ch == '\n' || ch == '\r' {
4854       break
4855     }
4856     sch := string(ch)
4857     str += sch
4858     fprintf(stderr,"%s",sch)
4859   }
4860   return str
4861 }
4862
4863 // search pattern must be an array and selectable with ^N/^P
4864 var SearchPat = ""
4865 var SearchForw = true
4866
4867 func (iin*IInput)xgetline1(prevline string, gsh*GshContext)(string){
4868   var ch int;
4869
4870   MODE_ShowMode = false
4871   MODE_VicMode = false
4872   iin.Redraw();
4873   first := true
4874
4875   for cix := 0; ; cix++ {
4876     iin.pinJMode = iin.inJMode
4877     iin.inJMode = false
4878
4879     ch = iin.Getc(1000*1000)
4880
4881     if ch != EV_TIMEOUT && first {
4882       first = false
4883       mode := 0
4884       if romkanmode {
4885         mode = 1
4886       }
4887       now := time.Now()
4888       Events = append(Events,Event{now,EV_MODE,int64(mode),CmdIndex})
4889     }
4890     if ch == 033 {
4891       MODE_ShowMode = true
4892       MODE_VicMode = !MODE_VicMode
4893       iin.Redraw();
4894       continue
4895     }
4896     if MODE_VicMode {
4897       switch ch {
4898         case '0': ch = GO_TOPL
4899         case '$': ch = GO_ENDL
4900         case 'b': ch = GO_TOPW
4901         case 'e': ch = GO_ENDW
4902         case 'w': ch = GO_NEXTW
4903         case '%': ch = GO_PAIRCH
4904
4905         case 'j': ch = GO_DOWN
4906         case 'k': ch = GO_UP
4907         case 'h': ch = GO_LEFT
4908         case 'l': ch = GO_RIGHT
4909         case 'x': ch = DEL_RIGHT
4910         case 'a': MODE_VicMode = !MODE_VicMode
4911           ch = GO_RIGHT
4912         case 'i': MODE_VicMode = !MODE_VicMode
4913           iin.Redraw();
4914           continue
4915         case '-':
4916           right,head := delHeadChar(iin.right)
4917           if len([]byte(head)) == 1 {
4918             ch = int(head[0])
4919             if( 'a' <= ch && ch <= 'z' ){
4920               ch = ch + 'A'-'a'
4921             }else{
4922               if( 'A' <= ch && ch <= 'Z' ){
4923                 ch = ch + 'a'-'A'
4924               }
4925               iin.right = string(ch) + right
4926             }
4927           iin.Redraw();
4928           continue
4929         case 'f': // GO_FORWCH
4930           iin.Redraw();
4931           ch = iin.Getc(3*1000*1000)
4932           if ch == EV_TIMEOUT {
4933             iin.Redraw();
4934             continue
4935           }
4936           SearchPat = string(ch)
4937           SearchForw = true
4938           iin.GotoFORWSTR(SearchPat,gsh)
4939           iin.Redraw();
4940           continue
4941         case '/':
4942           SearchPat = iin.getstring1("//") // should be editable
4943           SearchForw = true
4944           iin.GotoFORWSTR(SearchPat,gsh)
4945           iin.Redraw();
4946           continue
4947         case '?':
4948           SearchPat = iin.getstring1("?) // should be editable
4949           SearchForw = false
4950           iin.GotoBACKSTR(SearchPat,gsh)
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     case 'N':
4962         if !SearchForw {
4963             iin.GotoFORWSTR(SearchPat,gsh)
4964         }else{
4965             iin.GotoBACKSTR(SearchPat,gsh)
4966         }
4967         iin.Redraw();
4968         continue
4969     }
4970 }
4971 switch ch {
4972     case GO_TOPW:
4973         iin.GotoTOPW()
4974         iin.Redraw();
4975         continue
4976     case GO_ENDW:
4977         iin.GotoENDW()
4978         iin.Redraw();
4979         continue
4980     case GO_NEXTW:
4981         // to next space then
4982         iin.GotoNEXTW()
4983         iin.Redraw();
4984         continue
4985     case GO_PAIRCH:
4986         iin.GotoPAIRCH()
4987         iin.Redraw();
4988         continue
4989 }
4990
4991 //fprintf(stderr,"A[%02X]\n",ch);
4992 if( ch == '\\\' || ch == 033 ){
4993     MODE_ShowMode = true
4994     metach := ch
4995     iin.waitingMeta = string(ch)
4996     iin.Redraw();
4997     // set cursor //fprintf(stderr,"???\b\b\b")
4998     ch = fgetTimeout(stdin,2000*1000)
4999     // reset cursor
5000     iin.waitingMeta = ""
5001
5002     cmdch := ch
5003     if( ch == EV_TIMEOUT ){
5004         if metach == 033 {
5005             continue
5006         }
5007         ch = metach
5008     }/*
5009     if( ch == 'm' || ch == 'M' ){
5010         mch := fgetTimeout(stdin,1000*1000)
5011         if mch == 'r' {
5012             romkanmode = true
5013         }else{
5014             romkanmode = false
5015         }
5016         continue
5017     }/*
5018     */
5019     if( ch == 'k' || ch == 'K' ){
5020         MODE_Recursive = !MODE_Recursive
5021         iin.Translate(cmdch);
5022         continue
5023     }else
5024     if( ch == 'j' || ch == 'J' ){
5025         iin.Translate(cmdch);
5026         continue
5027     }else
5028     if( ch == 'i' || ch == 'I' ){
5029         iin.Replace(cmdch);
5030         continue
5031     }else
5032     if( ch == 'l' || ch == 'L' ){
5033         MODE_LowerLock = !MODE_LowerLock
5034         MODE_CapsLock = false
5035         if MODE_ViTrace {
5036             fprintf(stderr,"%v\r\n",string(cmdch));
5037         }
5038         iin.Redraw();
5039         continue
5040     }else
5041     if( ch == 'u' || ch == 'U' ){
5042         MODE_CapsLock = !MODE_CapsLock
5043         MODE_LowerLock = false
5044         if MODE_ViTrace {
5045             fprintf(stderr,"%v\r\n",string(cmdch));
5046         }
5047         iin.Redraw();
5048         continue
5049     }else
5050     if( ch == 'v' || ch == 'V' ){
5051         MODE_ViTrace = !MODE_ViTrace
5052         if MODE_ViTrace {
5053             fprintf(stderr,"%v\r\n",string(cmdch));
5054         }
5055         iin.Redraw();
5056         continue
5057     }else
5058     if( ch == 'c' || ch == 'C' ){
5059         if 0 < len(iin.line) {
5060             xline,tail := delTailChar(iin.line)
5061             if len([byte(tail)]) == 1 {
5062                 ch = int(tail[0])
5063                 if( 'a' <= ch && ch <= 'z' ){
5064                     ch = ch + 'A'-'a'
5065                 }else
5066                 if( 'A' <= ch && ch <= 'Z' ){
5067                     ch = ch + 'a'-'A'
5068                 }
5069                 iin.line = xline + string(ch)
5070             }
5071         }
5072         if MODE_ViTrace {
5073             fprintf(stderr,"%v\r\n",string(cmdch));
5074         }
5075         iin.Redraw();
5076         continue
5077     }else{
5078         iin.pch = append(iin.pch,ch) // push
5079         ch = '\\\'"
5080     }
5081 }
5082 switch( ch ){
5083     case 'P'-0x40: ch = GO_UP

```

```

5085     case 'N'-0x40: ch = GO_DOWN
5086     case 'B'-0x40: ch = GO_LEFT
5087     case 'F'-0x40: ch = GO_RIGHT
5088   }
5089   //fprintf(stderr, "B[%02X]\n", ch);
5090   switch( ch ){
5091     case 0:
5092       continue;
5093
5094     case '\t':
5095       iin.Replace('j');
5096       continue;
5097     case 'X'-0x40:
5098       iin.Replace('j');
5099       continue;
5100
5101     case EV_TIMEOUT:
5102       iin.Redraw();
5103       if iin.pindMode {
5104         fprintf(stderr, "\\J\\r\\n")
5105         iin.inJmode = true
5106       }
5107       continue;
5108     case GO_UP:
5109       if iin.lno == 1 {
5110         continue;
5111       }
5112       cmd.ok := gsh.cmdStringInHistory(iin.lno-1)
5113       if ok {
5114         iin.line = cmd
5115         iin.right = ""
5116         iin.lno = iin.lno - 1
5117       }
5118       iin.Redraw();
5119       continue;
5120     case GO_DOWN:
5121       cmd.ok := gsh.cmdStringInHistory(iin.lno+1)
5122       if ok {
5123         iin.line = cmd
5124         iin.right = ""
5125         iin.lno = iin.lno + 1
5126       }else{
5127         iin.line = ""
5128         iin.right = ""
5129         if iin.lno == iin.lastlno-1 {
5130           iin.lno = iin.lno + 1
5131         }
5132       }
5133       iin.Redraw();
5134       continue;
5135     case GO_LEFT:
5136       if 0 < len(iin.line) {
5137         xline,tail := delTailChar(iin.line)
5138         iin.line = xline
5139         iin.right = tail + iin.right
5140       }
5141       iin.Redraw();
5142       continue;
5143     case GO_RIGHT:
5144       if( 0 < len(iin.right) && iin.right[0] != 0 ){
5145         xright,head := delHeadChar(iin.right)
5146         iin.right = xright
5147         iin.line += head
5148       }
5149       iin.Redraw();
5150       continue;
5151     case EOF:
5152       goto EXIT;
5153     case 'R'-0x40: // replace
5154       dst := convs(iin.line+iin.right);
5155       iin.line = dst
5156       iin.right = ""
5157       iin.Redraw();
5158       continue;
5159     case 'T'-0x40: // just show the result
5160       readDic();
5161       romkanmode = !romkanmode;
5162       iin.Redraw();
5163       continue;
5164     case 'L'-0x40:
5165       iin.Redraw();
5166       continue;
5167     case 'K'-0x40:
5168       iin.right = ""
5169       iin.Redraw();
5170       continue;
5171     case 'E'-0x40:
5172       iin.line += iin.right
5173       iin.right = ""
5174       iin.Redraw();
5175       continue;
5176     case 'A'-0x40:
5177       iin.right = iin.line + iin.right
5178       iin.line = ""
5179       iin.Redraw();
5180       continue;
5181     case 'U'-0x40:
5182       iin.line = ""
5183       iin.right = ""
5184       iin.clearline();
5185       iin.Redraw();
5186       continue;
5187     case DEL_RIGHT:
5188       if( 0 < len(iin.right) ){
5189         iin.right,_ = delHeadChar(iin.right)
5190         iin.Redraw();
5191       }
5192       continue;
5193     case 0xF: // BS? not DEL
5194       if( 0 < len(iin.line) ){
5195         iin.line,_ = delTailChar(iin.line)
5196         iin.Redraw();
5197       }
5198     /*
5199     else
5200       if( 0 < len(iin.right) ){
5201         iin.right,_ = delHeadChar(iin.right)
5202         iin.Redraw();
5203       }
5204     */
5205     continue;
5206     case 'H'-0x40:
5207       if( 0 < len(iin.line) ){
5208         iin.line,_ = delTailChar(iin.line)
5209       }

```

```

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

```

```

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

```



```
5581 "amwJ44KMCmtqa2psCeOCj0pqas2psCeOCjwpramtramwJ44KQCmtqamtrbAnjgpEKa2pqawJ"+  
5582 "44KSCmtqa2prbAnjgpMKa2pqas2psCeODvApla2wJ44KbCmtrampbAnjgpwKa2pramtqbAnj"+  
5583 "gIEK";  
5584 //</span>  
5585  
5586 //</span>  
5587 /*  
5588 <details id="references"><summary>References</summary><div class="gsh-src">  
5589 <p>  
5590 <a href="https://golang.org">The Go Programming Language</a>  
5591 <!--  
5592 <iframe src="https://golang.org" width="100%" height="300"></iframe>  
5593 -->  
5594  
5595 <a href="https://developer.mozilla.org/ja/docs/Web">MDN web docs</a>  
5596 <a href="https://developer.mozilla.org/ja/docs/Web/HTML/Element">HTML</a>  
5597 CSS:  
5598 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors">Selectors</a>  
5599 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/background-repeat">repeat</a>  
5600 HTTP  
5601 JavaScript:  
5602 ...  
5603 </p>  
5604 </div></details>  
5605 /*  
5606 /*  
5607 <details id="html-src" onclick="frame_open();"><summary>Raw Source</summary><div>  
5608 <!-- h2>The full of this HTML including the Go code is here.</h2 -->  
5609 <details id="gsh-whole-view"><summary>Whole file</summary>  
5610 <a name="whole-src-view"></a>  
5611 <span id="src-frame"></span><!-- a window to show source code -->  
5612 </details>  
5613  
5614 <details id="gsh-style-frame" onclick="fill_CSSView()"><summary>CSS part</summary>  
5615 <a name="style-src-view"></a>  
5616 <span id="gsh-style-view"></span>  
5617 </details>  
5618  
5619  
5620 <details id="gsh-script-frame" onclick="fill_JavaScriptView()"><summary>JavaScript part</summary>  
5621 <a name="script-src-view"></a>  
5622 <span id="gsh-script-view"></span>  
5623 </details>  
5624  
5625 <details id="gsh-data-frame" onclick="fill_DataView()"><summary>Builtin data part</summary>  
5626 <a name="gsh-data-frame"></a>  
5627 <span id="gsh-data-view"></span>  
5628 </details>  
5629  
5630 <div id="GshFooter"></div>  
5631 </div></details>  
5632 */  
5633  
5634 /*  
5635 <!-- 2020-09-17 SatoxITS, visible script -->  
5636 <details><summary>GJScrip</summary>  
5637 <style>.gjscript { font-family:Georgia; }</style>  
5638 <pre id="gjscript_1" class="gjscript"> function gjtest1() { alert('Hello GJScrip!'); }  
5639 gjtest1()  
5640 </pre>  
5641 <script>  
5642 gjs = document.getElementById('gjscript_1');  
5643 //eval(gjs.innerHTML);  
5644 //gjs.outerHTML = ""  
5645 </script>  
5646 </details><!-- ----- END-OF-VISIBLE-PART ----- -->  
5647 */  
5648  
5649 /*  
5650 <!--  
5651 // 2020-0906 added,  
5652 https://developer.mozilla.org/en-US/docs/Web/CSS/z-index  
5653 https://developer.mozilla.org/en-US/docs/Web/CSS/position  
5654 -->  
5655 <span id="GshGrid">(^_~)/<small>(Hit j k l h)</small></span>  
5656  
5657 <span id="GStat"><br>  
5658 </span>  
5659 <span id="GMenu" onclick="GShellMenu(this)"></span>  
5660 <span id="GTop"></span>  
5661 <div id="GshellPlane" onclick="showGShellPlane();"></div>  
5662 <div id="RawTextViewer"></div>  
5663 <div id="RawTextViewerClose" onclick="hideRawTextViewer()"> CLOSE </div>  
5664  
5665 <style id="GshStyleDef">  
5666 #LineNumbered table,tr,td {  
5667 margin:0;  
5668 padding:4px;  
5669 spacing:0;  
5670 border:12px;  
5671 }  
5672 textarea.LineNumber {  
5673 font-size:12px;  
5674 font-family:monospace,Courier New;  
5675 color:#282;  
5676 padding:4px;  
5677 text-align:right;  
5678 }  
5679 textarea.LineNumbered {  
5680 font-size:12px;  
5681 font-family:monospace,Courier New;  
5682 padding:4px;  
5683 wrap:off;  
5684 }  
5685 #RawTextViewer{  
5686 z-index:0;  
5687 position:fixed; top:0px; left:0px;  
5688 width:100%; height:50px;  
5689 overflow:auto;  
5690 color:#fff; background-color:rgba(128,128,256,0.2);  
5691 font-size:12px;  
5692 spellcheck:false;  
5693 }  
5694 #RawTextViewerClose{  
5695 z-index:0;  
5696 position:fixed; top:-100px; left:-100px;  
5697 color:#fff; background-color:rgba(128,128,256,0.2);  
5698 font-size:20px; font-family:Georgia;  
5699 white-space:pre;  
5700 }  
5701 #GshellPlane{  
5702 z-index:0;  
5703 position:fixed; top:0px; left:0px;  
5704 width:100%; height:50px;
```

```

5705    overflow:auto;
5706    color:#fff; background-color:rgba(128,128,256,0.3);
5707    font-size:12px;
5708  }
5709  #GTop{
5710    z-index:9;
5711    opacity:1.0;
5712    position:fixed; top:0px; left:0px;
5713    width:320px; height:20px;
5714    color:#fff; background-color:rgba(32,32,160,0.15);
5715    color:#fff; font-size:12px;
5716  }
5717  #GPos{
5718    z-index:12;
5719    position:fixed; top:0px; left:0px;
5720    opacity:1.0;
5721    width:640px; height:30px;
5722    color:#fff; background-color:rgba(0,0,0,0.2);
5723    color:#fff; font-size:12px;
5724  }
5725  #GMenu{
5726    z-index:2000;
5727    position:fixed; top:250px; left:0px;
5728    opacity:1.0;
5729    width:100px; height:100px;
5730    color:#fff;
5731    color:#fff; background-color:rgba(0,0,0,0.0);
5732    color:#fff; font-size:16px; font-family:Georgia;
5733    background-repeat:no-repeat;
5734  }
5735  #GSTat{
5736    z-index:8;
5737    xopacity:0.0;
5738    position:fixed; top:20px; left:0px;
5739    xwidth:640px;
5740    width:100%; height:90px;
5741    color:#fff; background-color:rgba(0,0,128,0.04);
5742    font-size:20px; font-family:Georgia;
5743  }
5744  #GLog{
5745    z-index:10;
5746    position:fixed; top:50px; left:0px;
5747    opacity:1.0;
5748    width:640px; height:60px;
5749    color:#fff; background-color:rgba(0,0,128,0.10);
5750    font-size:12px;
5751  }
5752  #GshGrid {
5753    z-index:11;
5754    xopacity:0.0;
5755    position:fixed; top:0px; left:0px;
5756    width:320px; height:30px;
5757    color:#f9f; font-size:16px;
5758  }
5759  xbody {display:none;}
5760  .gsh-link{color:green;}
5761  #gsh {border-width:1px;margin:0;padding:0;}
5762  #gsh {font-family:monospace,Courier New;color:#ddf;font-size:8px;}
5763  #gsh header{height:100px;}
5764  #xgsh header{height:100px;background-image:url(GShell-Logo00.png);}
5765  #GshMenu{font-size:14pt;color:#c44;}
5766  .GshMenu{font-size:14pt;color:#2a2;padding:4px;}
5767  .GshMenu:hover{font-size:14pt;color:#fff;font-weight:bold;background-color:#2a2;}
5768  #GshFooter{height:100px;background-size:80px;background-repeat:no-repeat;}
5769  #gsh note{color:#000;font-size:10pt;}
5770  #gsh h2{color:#24a;font-family:Georgia;font-size:18pt;}
5771  #gsh h3{color:#24a;font-family:Georgia;font-size:16pt;}
5772  #gsh details{color:#888;background-color:#fff;font-family:monospace;}
5773  #gsh summary{font-size:16pt;color:#fff;background-color:#8af;height:30px;}
5774  #gsh pre{font-size:11pt;color:#223;background-color:#faffff;}
5775  #gsh a{color:#24a;}
5776  #gsh a[name]{color:#24a;font-size:16pt;}
5777  #gsh .gsh-src{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
5778  #gsh .gsh-src{background-color:#faffff;color:#223;}
5779  #gsh-src-src{spellcheck:false}
5780  #src-frame-textarea{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
5781  #src-frame-textarea{background-color:#faffff;color:#223;}
5782  .gsh-code {white-space:pre;font-family:monospace !important;}
5783  .gsh-code {color:#088;font-size:11pt; background-color:#eef;}
5784  .gsh-golang-data {display:none;}
5785  #gsh-WinId {color:#000;font-size:14pt;}
5786
5787  .gsh-document {font-size:11pt;background-color:#fff;font-family:Georgia;}
5788  .gsh-document {color:#000;background-color:#fff !important;}
5789  .gsh-document > h2{color:#000;background-color:#fff !important;}
5790  .gsh-document details{color:#000;background-color:#fff;font-family:Georgia;}
5791  .gsh-document p{max-width:550pt;color:#000;background-color:#fff;font-family:Georgia;}
5792  .gsh-document address{width:500pt;color:#000;background-color:#fff;font-family:Georgia;}
5793
5794  @media print {
5795    #gsh pre{font-size:11pt !important;}
5796  }
5797  </style>
5798
5799 <!--
5800 // Logo image should be drawn by JavaScript from a meta-font.
5801 // CSS seems not follow line-splittered URL
5802 ---
5803 <script id="gsh-data">
5804 //GShellLogo="QR-ITS-more.jp.png"
5805 GShellLogo="data:image/png;base64,
5806 iVBORw0KGgoAAAANSUhEUgAAEAAAABAAQEBAAAUAUAAAABAAAPgEAAMAAAABAAIAIAlpAQAABAB
5807 TUA0KgAAAQABAAFAAAUAAAABAAAPgEAAMAAAABAAIAIAlpAQAABAB
5808 AAAATgAAAABAAAB1AAAQAAEgAAAABAAAQABAACgqgAEAAAAAAQAAQGgAwAB
5809 AAAAQAAAHHAAAAYX1bhgAAAALwSf1AAALEwAACxMBAjqCgAAAF3RJREFUEaHtnQuUFNfZ
5810 x++t7uk23iqgg0/jY6Osbb8WgMzAvn7u4+biSTR7YnQxd0PCkGj2aNu1D2MSlRkeuJpNoOdu
5811 4iuJx7jriiY250DGmP2VqIBElSggCoIMMa+mu+vu/2MD9U1dau62aUbv91GKrq3vdw6/q
5812 fnVdxR8LA8IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES
5813 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES
5814 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES
5815 IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES
5816 Zeks9H9+ftsk6dhxniczcgqdE7Yus+IgaalKfn5YsoKmhwEptK4M0QFz5UeXlbLYsaYU15
5817 npD1LXEZ2C1fIRM53JSUaq9ScqU6i+2kK3StuNy5reEGKJ7Qw7m0vkec2Tcq0izw1jhF8
5818 jbv0LHCstMRB3USXEkjhfu7dsdmB2+xu4vwWFBxbpMezu1AE/hcKoGabe6ekGolnykh5PC
5819 HxH2VVBKoRqk3guEkiYdaOFN56Okd16w5BwomnQOlypziON9DLMxpFK/60p2P/Piyovf
5820 N8mpM+/jWNjnjw9KgO7oLVGSFt2p2Rilng3i0vk7ys0WMzEuVFFplRKydfok2LRSl0d
5821 zrWocCOG6gQhvRaqCj/dktj3g7dXKh4gKnaRS0zPyzergS6RaZDQgfKf79SKTRXHu/e+9Fn
5822 L66as8pU/PN1pL1TLQJKSc73dPXsr20ur7iiwPc8QhbBnCyhU1lrryyO!QyYF5JfvqBl7jx
5823 +cNHjb5j5JyRlJH7y39084d40H2Qtx8THaPefuIOU+w1C+Knhyh5FGeV0WGGaExB83eMoLy
5824 rikbd9gHEP52Vgql489PUA6kJyYFpbobnzbLJg4zfiesndHCwvOeiVQOB/5C9FYDl1ueOH
5825 +zGuh9nSqGqmou#wurk19RpjBD4v6uQcd5TU0W63zD3Mhesyl4V49i9sbdkYxgh1CpF
5826 UJ6toCaCP7F9VFB58NBEDHTOMBaE74En+eWrWr-Lz/QWf60AdB7QJUjs/0A7c0oBNBCeNUZ
5827 ttCu/coG28fLpvKE17PPV9bjuraoEahb1vxaRlquoeBPyfUDo4+ofeBdybSI4t29XeSXFAMo<
5828 bgGov0glzgGGw4jf392xInHhdc+Mwf3JTjint2zyCljBJXNUT5KIKycklsxxRld1d6Bmcenv\
```



```
6077 UEBbDXaB0X2aNjueYDOzNklQassPCKj4c4nW3ElSfwqYk6ju/vAkPhg0AlSFhve8Jt0dkwDMwr\  
6078 yMGSSupWHar19k0tkV2sb3sdW2rUCqW88g4RpIA9s1jPv9Ctp1NRd4xFkin8XaQC1wT6Lzq\  
6079 Z0dHw/4+U2Gzqls8gbqVmkfr1N6YXK80q1D00mlGMvzPERA8AL9vvbOifpSoL33fsVytL\  
6080 S9wiqbzzhuI38v5n783/gBuUs2eLg1c8gAAAABJRU5ErkJgg==";  
6081  
6082 </script>  
6083 <!--  
6084 <div id="GJFactory_1" class="xxxGJFactory" style=""></div>  
6085 -->  
6086 https://developer.mozilla.org/en-US/docs/Web/CSS/line-height  
6087 -->  
6088 <style>  
6089   .GJFactory{  
6090     resize:both; overflow:scroll;  
6091     position:static;  
6092     border:1.2px dashed #282; xborder-radius:2px;  
6093     margin:0px; padding:10px !important;  
6094     width:340px; height:340px;  
6095     flex-wrap: wrap;  
6096     color:#fff; background-color:rgba(0,0,0,0.0);  
6097     line-height:0.0;  
6098     xxxcolor:#22a !important;  
6099     text-shadow:2px 2px #ddf;  
6100   }  
6101   .GJFactory h1,h2,h3,h4 {  
6102     xxxcolor:#22a !important;  
6103   }  
6104   xxxinput {  
6105     border:1px dashed #0f0; border-radius:0px;  
6106   }  
6107   .GJWin:hover{  
6108     color:#df8 !important;  
6109     background-color:rgba(32,32,160,0.8) !important;  
6110     line-height:0.0;  
6111   }  
6112   .GJWin:active{  
6113     color:#df8 !important;  
6114     background-color:rgba(224,32,32,0.8) !important;  
6115     line-height:0.0;  
6116   }  
6117   .GJWin:focus{  
6118     color:#df8 !important;  
6119     background-color:rgba(32,32,32,1.0) !important;  
6120     line-height:0.0;  
6121   }  
6122   .GJWin{  
6123     z-index:10000;  
6124     display:inline;  
6125     position:relative;  
6126     flex-wrap: wrap;  
6127     top:0; left:0px;  
6128     width:285px !important; height:205px !important;  
6129     border:1px solid #ea; border-radius:2px;  
6130     margin:0px; padding:0px;  
6131     font-size:8pt;  
6132     line-height:0.0;  
6133     color:#fff; background-color:rgba(0,0,64,0.1) !important;  
6134   }  
6135   .GJTab{  
6136     display:inline;  
6137     position:relative;  
6138     top:0px; left:0px;  
6139     margin:0px; padding:2px;  
6140     border:0px solid #00f; border-radius:2px;  
6141     width:90px; height:20px;  
6142     font-family:Georgia;  
6143     font-size:9pt;  
6144     line-height:1.0;  
6145     white-space:nowrap;  
6146     color:#fff; background-color:rgba(0,0,64,0.7);  
6147     text-align:center;  
6148     vertical-align:middle;  
6149   }  
6150   .GJStat:focus{  
6151     color:#df8 !important;  
6152     background-color:rgba(32,32,32,1.0) !important;  
6153     line-height:1.0;  
6154   }  
6155   .GJStat{  
6156     display:inline;  
6157     position:relative;  
6158     top:0px; left:0px;  
6159     margin:0px; padding:2px;  
6160     border:0px solid #00f; border-radius:2px;  
6161     width:166px; height:20px;  
6162     font-family:monospace;  
6163     font-size:9pt;  
6164     line-height:1.0;  
6165     color:#fff; background-color:rgba(0,0,64,0.2);  
6166     text-align:center;  
6167     vertical-align:middle;  
6168   }  
6169   .GJIcon{  
6170     display:inline;  
6171     position:relative;  
6172     top:0px; left:1px;  
6173     border:2px solid #44a;  
6174     margin:0px; padding:1px;  
6175     width:13.2; height:13.2px;  
6176     border-radius:2px;  
6177     font-family:Georgia;  
6178     font-size:13.2px;  
6179     line-height:1.0;  
6180     white-space:nowrap;  
6181     color:#fff; background-color:rgba(32,32,160,0.8);  
6182     text-align:center;  
6183     vertical-align:middle;  
6184     text-shadow:0px 0px;  
6185   }  
6186   .GJText:focus{  
6187     color:#fff !important;  
6188     background-color:rgba(32,32,160,0.8) !important;  
6189     line-height:1.0;  
6190   }  
6191   .GJText{  
6192     display:inline;  
6193     position:relative;  
6194     top:0px; left:0px;  
6195     border:0px solid #000; margin:0px; padding:0px;  
6196     width:280px; height:160px;  
6197     border:0px;  
6198     font-family:Courier New,monospace !important;  
6199     font-size:8pt;  
6200     line-height:1.0;
```

```

6201     white-space:pre;
6202     color:#ffff; background-color:rgba(0,0,64,0.5);
6203     background-color:rgba(32,32,128,0.8) !important;
6204   }
6205   .GJMode{
6206     display:inline;
6207     position:relative;
6208     top:0px; left:0px;
6209     border:0px solid #000; border-radius:0px;
6210     margin:0px; padding:0px;
6211     width:280px; height:20px;
6212     font-size:9pt;
6213     line-height:1.0;
6214     white-space:nowrap;
6215     color:#ffff; background-color:rgba(0,0,64,0.7);
6216     text-align:left;
6217     vertical-align:middle;
6218   }
6219 </style>
6220
6221 <script id="gsh-script">
6222 // 2020-0909 added, permanet local storage
6223 // https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
6224 var MyHistory = ""
6225 Permanent = localStorage;
6226 MyHistory = Permanent.getItem('MyHistory')
6227 if( MyHistory == null ){ MyHistory = "" }
6228 d = new Date()
6229 MyHistory = d.getTime() /1000 + " "+document.URL+"\n" + MyHistory
6230 Permanent.setItem('MyHistory',MyHistory)
//Permanent.setItem('MyWindow',window)
6232
6233 var GJLog_Win = null
6234 var GJLog_Tab = null
6235 var GJLog_Stat = null
6236 var GJLog_Text = null
6237 var GJWin_Mode = null
6238 var FProductInterval = 0
6239
6240 var GJ_FactoryID = -1
6241 var GJFactory = null
6242 if( e = document.getElementById('GJFactory_0') ){
6243   GJFactory_1.height = 0
6244   GJFactory = e
6245   e.setAttribute('class','GJFactory')
6246   var GJ_FactoryID = 0
6247 }else{
6248   GJFactory = GJFactory_1
6249   var GJ_FactoryID = 1
6250 }
6251
6252 function GJFactory_Destroy(){
6253   gif = GJFactory
6254   //gif = document.getElementById('GJFactory')
6255   //alert('gif='+gif)
6256   if( gif != null ){
6257     if( gif.childNodes != null ){
6258       for( i = 0; i < gif.childNodes.length; i++ ){
6259         gif.removeChild(gif.childNodes[i])
6260       }
6261     }
6262     gif.innerHTML = ''
6263     gif.style.width = 0
6264     gif.style.height = 0
6265     gif.removeAttribute('style')
6266     GJLog_Win = GJLog_Tab = GJLog_Stat = GJLog_Text = GJWin_Mode = null
6267     window.clearInterval(FProductInterval)
6268     return '-- Destroy: work product destroyed'
6269   }else{
6270     return '-- Destroy: work product not exist'
6271   }
6272 }
6273
6274 var TransMode = false
6275 var OnKeyControl = false
6276 var OnKeyShift = false
6277 var OnKeyAlt = false
6278 var OnKeyJ = false
6279 var OnKeyK = false
6280 var OnKeyL = false
6281
6282 function GJWin_OnKeyUp(ev){
6283   keycode = ev.code;
6284   if( keycode == 'ShiftLeft' ){
6285     OnKeyShift = false
6286   }else
6287   if( keycode == 'ControlLeft' ){
6288     OnKeyControl = false
6289   }else
6290   if( keycode == 'AltLeft' ){
6291     OnKeyAlt = false
6292   }else
6293   if( keycode == 'KeyJ' ){ OnKeyJ = false }else
6294   if( keycode == 'KeyK' ){ OnKeyK = false }else
6295   if( keycode == 'KeyL' ){ OnKeyL = false }else
6296   {
6297   }
6298   ev.preventDefault()
6299 }
6300 function and(a,b){ if(a){ if(b){ return true; } return false; } }
6301 function GJWin_OnKeyDown(ev){
6302   keycode = ev.code;
6303   mode = ''
6304   key = ''
6305   if( keycode == 'ControlLeft' ){
6306     onKeyControl = true
6307     ev.preventDefault()
6308     return;
6309   }else
6310   if( keycode == 'ShiftLeft' ){
6311     OnKeyShift = true
6312     ev.preventDefault()
6313     return;
6314   }else
6315   if( keycode == 'AltLeft' ){
6316     ev.preventDefault()
6317     OnKeyAlt = true
6318     return;
6319   }else
6320   if( keycode == 'Backquote' ){
6321     TransMode = !TransMode
6322     ev.preventDefault()
6323   }else
6324   if( and(keycode == 'Space', OnKeyShift ) ){

```

```

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

```

```

6449 // vertical centering of the last line
6450 sHeight = txa.scrollHeight - 30; // depends on the font-size
6451 tmLap3 = new Date();
6452
6453 txa.scrollTop = sHeight; // depends on the font-size
6454 tmLap4 = new Date();
6455
6456 now = tmLap0.getTime();
6457 if( PrevReport == 0 || 10000 <= now-PrevReport ){
6458   PrevReport = now;
6459   console.log('StatusBarUpdate:'
6460     + 'leng=' + txa.value.length + ' byte,' +
6461     + 'time=' + (tmLap4-tmLap0) + 'ms {' +
6462     + 'tadd=' + (tmLap2-tmLap1) + ', ' +
6463     + 'hcal=' + (tmLap3-tmLap2) + ', ' +
6464     + 'scrl=' + (tmLap4-tmLap3) + '}';
6465   );
6466 }
6467
6468 GJWin_StatusUpdate = GJLog_StatusUpdate;
6469 function GJ_showTime1(wid){
6470   //e = document.getElementById(wid);
6471   //console.log(wid.id+''.value.length='+wid.value.length)
6472   if( e != null ){
6473     //e.value = DateShort();
6474   }else{
6475     // should remove the Listener
6476   }
6477 }
6478 function GJWin_OnResizeTextarea(ev){
6479   this.value += 'resized:' + '\n'
6480 }
6481 function GJ_NewConsole(wname){
6482   wid = wname + '_' + GJ_ConsoleID
6483   GJ_ConsoleID += 1
6484
6485 GJFactory.style.setProperty('width',360+'px'); //GJFsize
6486 GJFactory.style.setProperty('height',320+'px')
6487 e = GJFactory;
6488 console.log('GJFa #' + e.id + ' from w=' + e.style.width + ', h=' + e.style.height)
6489
6490 if( GJFactory.innerHTML == "" ){
6491   GJFactory.innerHTML = '<' + 'H3>GJ Factory_' + GJ_FACTORY_ID + '<' + '/H3><' + 'hr>\n'
6492 }else{
6493   GJFactory.innerHTML += '<' + 'hr>\n'
6494 }
6495
6496 gjwin = GJLog_Win = document.createElement('span')
6497 gjwin.id = wid
6498 gjwin.setAttribute('class','GJWin')
6499 gjwin.setAttribute('draggable','true')
6500 gjwin.addEventListener('dragstart',GJWin_DragStart)
6501 gjwin.addEventListener('drag',GJWin_Drag)
6502 gjwin.addEventListener('dragend',GJWin_Drag)
6503 gjwin.addEventListener('dragover',GJWin_DragIgnore)
6504 gjwin.addEventListener('dragenter',GJWin_DragIgnore)
6505 gjwin.addEventListener('dragleave',GJWin_DragIgnore)
6506 gjwin.addEventListener('dragexit',GJWin_DragIgnore)
6507 gjwin.addEventListener('drop',GJWin_DragIgnore)
6508 gjwin.addEventListener('keydown',GJWin_OnKeyDown)
6509
6510 gjtab = GJLog_Tab = document.createElement('textare')
6511 gjtab.addEventListener('keydown',GJWin_OnKeyDown)
6512 gjtab.style.readonly = true
6513 gjtab.contenteditable = false
6514 gjtab.value = wid
6515 gjtab.id = wid + '_Tab'
6516 gjtab.setAttribute('class','GJTab')
6517 gjtab.setAttribute('spellcheck','false')
6518 gjwin.appendChild(gjtab)
6519
6520 gjstat = GJLog_Status = document.createElement('textare')
6521 gjstat.addEventListener('keydown',GJWin_OnKeyDown)
6522 gjstat.id = wid + '_Stat'
6523 gjstat.value = DateShort()
6524 gjstat.setAttribute('class','GJStat')
6525 gjstat.setAttribute('spellcheck','false')
6526 gjwin.appendChild(gjstat)
6527
6528 gjicon = document.createElement('span')
6529 gjicon.addEventListener('keydown',GJWin_OnKeyDown)
6530 gjicon.id = wid + '_Icon'
6531 gjicon.innerHTML = '<font color="#ff44">J</font>'
6532 gjicon.setAttribute('class','GJIcon')
6533 gjicon.setAttribute('spellcheck','false')
6534 gjwin.appendChild(gjicon)
6535
6536 gjtext = GJLog_Text = document.createElement('textare')
6537 gjtext.addEventListener('keydown',GJWin_OnKeyDown)
6538 gjtext.addEventListener('keyup',GJWin_OnKeyUp)
6539 gjtext.addEventListener('resize',GJWin_OnResizeTextarea)
6540 gjtext.id = wid + '_Text'
6541 gjtext.setAttribute('class','GJText')
6542 gjtext.setAttribute('spellcheck','false')
6543 gjwin.appendChild(gjtext)
6544
6545
6546 // user's mode as of IME
6547 gjmode = GJWin_Mode = document.createElement('textare')
6548 gjmode.addEventListener('keydown',GJWin_OnKeyDown)
6549 gjmode.addEventListener('keydown',GJWin_OnKeyDown)
6550 gjmode.id = wid + '_Mode'
6551 gjmode.setAttribute('class','GJMode')
6552 gjmode.setAttribute('spellcheck','false')
6553 gjmode.innerHTML = '[---]'
6554 gjwin.appendChild(gjmode)
6555
6556 gjwin.zIndex = 30000
6557 GJFactory.appendChild(gjwin)
6558
6559 gjtab.scrollTop = 0
6560 gjstat.scrollTop = 0
6561
6562 //x = gjwin.getBoundingClientRect().left.toFixed(0)
6563 //y = gjwin.getBoundingClientRect().top.toFixed(0)
6564 //gjwin.style.position = 'static'
6565 //gjwin.style.left = 0
6566 //gjwin.style.top = 0
6567
6568 //update = ('+' + wid + '.value=DateShort()');
6569 update = '(GJ_showTime1(' + wid + '))';
6570 // 2020-09-19 this causes memory leaks
6571 //FFProductInterval = window.setInterval(update,200)
6572 //FFProductInterval = window.setInterval(GJWin_StatusUpdate,200)

```

```

6573     //FPProductInterval = window.setInterval(GJ_showTime1,200,wid);
6574     FPProductInterval = window.setInterval(GJ_showTime1,200,gjstat);
6575     return update
6576   }
6577   function xxxGJF_StripClass(){
6578     GJLog_Win.style.removeProperty('width')
6579     GJLog_Tab.style.removeProperty('width')
6580     GJLog_Stat.style.removeProperty('width')
6581     GJLog_Text.style.removeProperty('width')
6582     return "Stripped classes"
6583   }
6584   function isElem(id){
6585     return document.getElementById(id) != null
6586   }
6587   function GJLog_append(...args){
6588     txt = GJLog_Text;
6589     if( txt == null ){
6590       return; // maybe GJLog element is removed
6591     }
6592     logs = args.join(' ')
6593     txt.value += logs + '\n'
6594     txt.scrollTop = txt.scrollHeight
6595     //GJLog_Stat.value = DateShort()
6596   }
6597   //window.addEventListener('time',GJLog_StatUpdate)
6598   function test_GJ_Console(){
6599     window.setInterval(GJLog_StatUpdate,1000);
6600     GJ_NewConsole('GJ_Console')
6601     e = GJFactory;
6602     console.log('GJF0 #' +e.id' from w=' +e.style.width+', h=' +e.style.height)
6603     e.style.width = 360; //GJFsize
6604     e.style.height = 320;
6605     console.log('GJF0 #' +e.id+ ' to w=' +e.style.width+', h=' +e.style.height)
6606   }
6607   /// test_GJ_Console();
6608
6609 var StopConsoleLog = true
6610 // 2020-09-15 added,
6611 // log should be saved to permanent memory
6612 // const px = new Proxy(console.log,{ alert() })
6613 __console_log = console.log
6614 __console_info = console.info
6615 __console_warn = console.warn
6616 __console_error = console.error
6617 __console_exception = console.exception
6618 // should pop callstack info.
6619 console.exception = function(...args){
6620   __console_exception(...args)
6621   alert('-- got console.exception("'+args+'")')
6622 }
6623 console.error = function(...args){
6624   __console_error(...args)
6625   alert('-- got console.error("'+args+'")')
6626 }
6627 console.warn = function(...args){
6628   __console_warn(...args)
6629   alert('-- got console.warn("'+args+'")')
6630 }
6631 console.info = function(...args){
6632   alert('-- got console.info("'+args+'")')
6633   __console_info(...args)
6634 }
6635 console.log = function(...args){
6636   __console_log(...args)
6637   if( StopConsoleLog ){
6638     return;
6639   }
6640   if( 0 <= args[0].indexOf('!') ){
6641     //alert('-- got console.log("'+args+'")')
6642   }
6643   GJLog_append(...args)
6644 }
6645 console.log('Hello, GJShell!')
6646
6647 //document.getElementById('GshFaviconURL').href = GShellFavicon
6648 document.getElementById('GshFaviconURL').href = GShellInsideIcon
6649 //document.getElementById('GshFaviconURL').href = ITSmoreQR
6650 //document.getElementById('GshFaviconURL').href = GSellLogo
6651
6652 // id of Gshell HTML elements
6653 var E_BANNER = "GshBanner" // banner element in HTML
6654 var E_FOOTER = "GshFooter" // footer element in HTML
6655 var E_GINDEX = "gsh-gindex" // index of Golang code of GShell
6656 var E_GOCODE = "gsh-gocode" // Golang code of GShell
6657 var E_TODO = "gsh-todo" // TODO of GShell
6658 var E_DICT = "gsh-dict" // Dictionary of GShell
6659
6660 function bannerElem(){ return document.getElementById(E_BANNER); }
6661 function bannerStyleFunc(){ return bannerElem().style; }
6662 var bannerStyle = bannerStyleFunc()
6663 bannerStyle.backgroundImage = "url("+GSellLogo+"")";
6664 //bannerStyle.backgroundImage = "url("+GShellInsideIcon+"")";
6665 //bannerStyle.backgroundImage = "url("+GShellFavicon+"")";
6666 GMMenu.style.backgroundImage = "url("+GShellInsideIcon+"")";
6667
6668 function footerElem(){ return document.getElementById(E_FOOTER); }
6669 function footerStyle(){ return footerElem().style; }
6670 footerElem().style.backgroundImage="url("+ITSmoreQR+"")";
6671 //footerStyle().backgroundImage = "url("+ITSmoreQR+"")";
6672
6673 function html_fold(e){
6674   if( e.innerHTML == "Fold" ){
6675     e.innerHTML = "Unfold"
6676     document.getElementById('gsh-menu-exit').innerHTML=""
6677     document.getElementById('GshStatement').open=false
6678     GshFeatures.open = false
6679     document.getElementById('html-src').open=false
6680     document.getElementById(E_GINDEX).open=false
6681     document.getElementById(E_GOCODE).open=false
6682     document.getElementById(E_TODO).open=false
6683     document.getElementById('references').open=false
6684   }else{
6685     e.innerHTML = "Fold"
6686     document.getElementById('GshStatement').open=true
6687     GshFeatures.open = true
6688     document.getElementById(E_GINDEX).open=true
6689     document.getElementById(E_GOCODE).open=true
6690     document.getElementById(E_TODO).open=true
6691     document.getElementById('references').open=true
6692   }
6693 }
6694 function html_pure(e){
6695   if( e.innerHTML == "Pure" ){
6696     document.getElementById('gsh').style.display=true

```

```

6697     //document.style.display = false
6698     e.innerHTML = "Unpure"
6699   }else{
6700     document.getElementById('gsh').style.display=false
6701     //document.style.display = true
6702     e.innerHTML = "Pure"
6703   }
6704 }
6705
6706 var bannerIsStopping = false
6707 //NOTE: .com/JSEEF/prop_style_backgroundposition.asp
6708 function shiftBG(){
6709   bannerIsStopping = !bannerIsStopping
6710   bannerStyle.backgroundPosition = "0 0";
6711 }
6712 // status should be inherited on Window Fork(), so use the status in DOM
6713 function html_stop(e,toggle){
6714   if( toggle ){
6715     if( e.innerHTML == "Stop" ){
6716       bannerIsStopping = true
6717       e.innerHTML = "Start"
6718     }else{
6719       bannerIsStopping = false
6720       e.innerHTML = "Stop"
6721     }
6722   }else{
6723     // update JavaScript variable from DOM status
6724     if( e.innerHTML == "Stop" ){ // shown if it's running
6725       bannerIsStopping = false
6726     }else{
6727       bannerIsStopping = true
6728     }
6729   }
6730 }
6731 html_stop(document.getElementById('GshMenuStop'),false) // onInit.
6732 //html_stop(bannerElem(),false) // onInit.
6733
6734 //https://www.w3schools.com/jssref/met_win_setinterval.asp
6735 function shiftBanner(){
6736   var now = new Date().getTime();
6737   //"console.log("now"+(now%10))
6738   if( !bannerIsStopping ){
6739     bannerStyle.backgroundPosition = ((now/10)%100000)+" 0";
6740   }
6741 }
6742 window.setInterval(shiftBanner,10); // onInit.
6743
6744 // <a href="https://developer.mozilla.org/ja/docs/Web/API/Window/open">window.open()</a>
6745 // from embedded html to standalone page
6746 var MyChildren = 0
6747 function html_fork(){
6748   GJFactory_Destroy()
6749   MyChildren += 1
6750   WinId = document.getElementById('gsh-WinId').innerHTML + "." + MyChildren;
6751   newwin = window.open("",WinId,"");
6752   src = document.getElementById('gsh');
6753   srchtml = src.outerHTML
6754   newwin.document.write("/*<"+"html>\n");
6755   newwin.document.write(srchtml);
6756   newwin.document.write("<"/+"html>\n");
6757   newwin.document.getElementById('gsh-menu-exit').innerHTML = "Close";
6758   newwin.document.getElementById('gsh-WinId').innerHTML = WinId;
6759   newwin.document.close();
6760   newwin.focus();
6761 }
6762 function html_close(){
6763   window.close()
6764 }
6765 function win_jump(win){
6766   //win = window.top;
6767   win = window.opener; // https://developer.mozilla.org/ja/docs/Web/API/window.opener
6768   if( win == null ){
6769     console.log("jump to window.opener("+win+")(Error)\n")
6770   }else{
6771     console.log("jump to window.opener("+win+")(n")
6772     win.focus();
6773   }
6774 }
6775
6776 // 0.2.9 2020-0902 created checksum of HTML
6777 CRC32UNIX = 0x04C11DB7 // Unix cksum
6778 function byteCRC32add(bigrcc,octstr,octlen){
6779   var crc = new Int32Array(1)
6780   crc[0] = bigrc
6781
6782   let oi = 0
6783   for( ; oi < octlen; oi++ ){
6784     var oct = new Int8Array(1)
6785     oct[0] = octstr[oi]
6786     for( bi = 0; bi < 8; bi++ ){
6787       //console.log("--CRC32 "+crc[0]+""+oct[0].toString(16)+" ["+oi+"."+bi+"]\n")
6788       ovf1 = crc[0] < 0 ? 1 : 0
6789       ovf2 = oct[0] < 0 ? 1 : 0
6790       ovf = ovf1 ^ ovf2
6791       oct[0] <= 1
6792       crc[0] <= 1
6793       if( ovf ){ crc[0] ^= CRC32UNIX }
6794     }
6795   }
6796   //console.log("--CRC32 byteAdd return crc="+crc[0]+","+oi+"/"+octlen+"\n")
6797   return crc[0];
6798 }
6799 function strCRC32add(bigrcc,str,rlen){
6800   var crc = new Uint32Array(1)
6801   crc[0] = bigrc
6802   var code = new Uint8Array(strlen);
6803   for( i = 0; i < strlen; i++){
6804     code[i] = str.charCodeAt(i) // not charAt() !!!!
6805     //console.log("== "+code[i].toString(16)+" <== "+str[i]+\n")
6806   }
6807   crc[0] = byteCRC32add(crc,code,strlen)
6808   //console.log("--CRC32 strAdd return crc="+crc[0]+\n")
6809   return crc[0]
6810 }
6811 function byteCRC32end(bigrcc,len){
6812   var crc = new Uint32Array(1)
6813   crc[0] = bigrc
6814   var slen = new Uint8Array(4)
6815   let li = 0
6816   for( ; li < 4; ){
6817     selen[li] = len
6818     li += 1
6819     len >= 8
6820     if( len == 0 ){

```

```

6821         break
6822     }
6823   }
6824   crc[0] = byteCRC32add(crc[0],slen,li)
6825   crc[0] ^= 0xFFFFFFFF
6826   return crc[0]
6827 }
6828 function strCRC32(stri,len){
6829   var crc = new Uint32Array(1)
6830   crc[0] = 0
6831   crc[0] = strCRC32add(0,stri,len)
6832   crc[0] = byteCRC32end(crc[0],len)
6833   //console.log("--CRC32 "+crc[0]+" "+len+"\n")
6834   return crc[0]
6835 }
6836
6837 DestroyGJLink = null; // to be replaced
6838
6839 function getSourceText(){
6840   version = document.getElementById('GshVersion').innerHTML
6841   sfavico = document.getElementById('GshFaviconURL').href;
6842   sbanner = document.getElementById('GshBanner').style.backgroundImage;
6843   spositi = document.getElementById('GshBanner').style.backgroundPosition;
6844   sfooter = document.getElementById('GshFooter').style.backgroundImage;
6845
6846   if( document.getElementById('GJC_1') != null ){ GJC_1.remove() }
6847   if( DestroyGJLink != null ) DestroyGJLink();
6848
6849   // these should be removed by CSS selector or class, after seavaed to non-printed attribute
6850   GshBanner.removeAttribute('style');
6851   GshFooter.removeAttribute('style');
6852   document.getElementById('GshMenuSign').removeAttribute("style");
6853   styleGMenu = GMenu.getAttribute("style")
6854   GMenu.removeAttribute("style");
6855   styleGStat = GStat.getAttribute("style")
6856   GStat.removeAttribute("style");
6857   styleGTop = GTop.getAttribute("style")
6858   GTop.removeAttribute("style");
6859   styleGshGrid = GshGrid.getAttribute("style")
6860   GshGrid.removeAttribute("style");
6861   //styleGPos = GPos.getAttribute("style");
6862   //GPos.removeAttribute("style");
6863   //GPos.innerHTML = "";
6864   //styleGLog = GLog.getAttribute("style");
6865   //GLog.removeAttribute("style");
6866   //GLog.innerHTML = "";
6867   styleGshellPlane = GshellPlane.getAttribute("style")
6868   GshellPlane.removeAttribute("style")
6869   styleRawTextViewer = RawTextViewer.getAttribute("style")
6870   RawTextViewer.removeAttribute("style")
6871   styleRawTextViewerClose = RawTextViewerClose.getAttribute("style")
6872   RawTextViewerClose.removeAttribute("style")
6873
6874   GshFaviconURL.href = "";
6875
6876   //it seems that interHTML and outerHTML generate style="" for these (??)
6877   //GshBanner.removeAttribute('style');
6878   //GshFooter.removeAttribute('style');
6879   //GshMenuSign.removeAttribute('style');
6880   GshBanner.style="";
6881   GshFooter.style="";
6882   GshMenuSign.style="";
6883
6884   textarea = document.createElement("textarea")
6885   srchtml = document.getElementById("gsh").outerHTML;
6886   //textarea = document.createElement("textarea")
6887   // 2020-0910 ?? ... this causes inserting style="" to Banner and Footer,
6888   // with Chromium?/ after reloading from file//!
6889   textarea.innerHTML = srchtml
6890   // <a href="https://stackoverflow.com/questions/5796718/html-entity-decode">Thanks</a>
6891   var rawtext = textarea.value
6892   //textarea.destroy()
6893   //rawtext = gsh.textContent // this removes #include <FILENAME> too
6894   var orgtext = ""
6895   + /*<html>\n" // lost preamble text
6896   + rawtext
6897   + "<"/>\n" // lost trail text
6898   ;
6899
6900 tlen = orgtext.length
6901 //console.log("getSourceText: length='"+tlen+"\n")
6902 document.getElementById('GshFaviconURL').href = sfavico;
6903
6904 document.getElementById('GshBanner').style.backgroundImage = sbanner;
6905 document.getElementById('GshBanner').style.backgroundPosition = spositi;
6906 document.getElementById('GshFooter').style.backgroundImage = sfooter;
6907
6908 GStat.setAttribute("style",styleGStat)
6909 GMenu.setAttribute("style",styleGMenu)
6910 GTop.setAttribute("style",styleGTop)
6911 //GLog.setAttribute("style",styleGLog)
6912 //GPos.setAttribute("style",styleGPos)
6913 GshGrid.setAttribute("style",styleGshGrid)
6914 GshellPlane.setAttribute("style",styleGshellPlane)
6915 RawTextViewer.setAttribute("style",styleRawTextViewer)
6916 RawTextViewerClose.setAttribute("style",styleRawTextViewerClose)
6917 canontext = orgtext.replace(' style=""','');
6918 // open="" too
6919 return canontext
6920 }
6921 function getDigest(){
6922   var text = ""
6923   text = getSourceText()
6924   var digest = ""
6925   tlen = text.length
6926   digest = strCRC32(text,tlen) + " " + tlen
6927   return { text, digest }
6928 }
6929 function html_digest(){
6930   version = document.getElementById('GshVersion').innerHTML
6931   let {text, digest} = getDigest()
6932   alert("cksum: " + digest + " " + version)
6933 }
6934 function charsin(stri,char){
6935   ln = 0;
6936   for( i = 0; i < stri.length; i++ ){
6937     if( stri.charCodeAt(i) == char.charCodeAt(0) ){
6938       ln++;
6939     }
6940   }
6941   return ln;
6942 }
6943 //class digestElement extends HTMLElement { }
6944 //< script>customElements.define('digest',digestElement)< /script>

```

```

6945 function showDigest(e){
6946     result = 'version=' + GshVersion.innerHTML + '\n'
6947     result += 'lines=' + e.dataset.lines + '\n'
6948     + 'length=' + e.dataset.length + '\n'
6949     + 'crc32u=' + e.dataset.crc32u + '\n'
6950     + 'time=' + e.dataset.time + '\n';
6951
6952     alert(result)
6953 }
6954
6955 function html_sign(e){
6956     if( RawTextViewer.style.zIndex == 1000 ){
6957         hideRawTextViewer()
6958         return
6959     }
6960     GJFactory_Destroy()
6961     if( DestroyGJLink != null ) DestroyGJLink();
6962     //gsh_digest_.innerHTML = "";
6963     text = getSourceText() // the original text
6964     tlen = text.length
6965     digest = strCRC32(text,tlen)
6966     //gsh_digest_.innerHTML = digest + " " + tlen
6967     //text = getSourceText() // the text with its digest
6968     Lines = charsin(text,'`')
6969
6970     name = "gsh"
6971     sid = name + "-digest"
6972     d = new Date()
6973     signedAt = d.getTime()
6974
6975     sign = '/'+'*'+span\n'
6976     + ' id="'+ sid + '"\n'
6977     + ' class=_digest_"\n'
6978     + ' data-target-id="'+name+'\n'
6979     + ' data-crc32u=' + digest + '\n'
6980     + ' data-length=' + tlen + '\n'
6981     + ' data-lines=' + Lines + '\n'
6982     + ' data-time=' + signedAt + '\n'
6983     + ' ><' + '/span>\n*'+/\n'
6984
6985     text = sign + text
6986
6987     txthtml = '<' + 'table id="LineNumbered"><' + 'tr><' + 'td>
6988     + '<' + 'textarea cols=5 rows=' + Lines + ' class="LineNumber">
6989     for( i = 1; i <= Lines; i++ ){
6990         txthtml += i.toString() + '\n'
6991     }
6992     txthtml += "
6993     + '<' + '/textarea>
6994     + '<' + '/td><' + 'td>
6995     + '<' + 'textarea cols=150 rows=' + Lines + ' spellcheck="false"
6996     + ' class="LineNumbered">
6997     + text + '<' + '/textarea>
6998     + '<' + '/td><' + '/tr><' + '/table>
6999
7000     for( i = 1; i <= 30; i++ ){
7001         txthtml += '<br>\n'
7002     }
7003     RawTextViewer.innerHTML = txthtml
7004
7005     btn = e
7006     e.style.color = "rgba(128,128,255,0.9)";
7007     y = e.getBoundingClientRect().top.toFixed(0)
7008     //h = e.getBoundingClientRect().height.toFixed(0)
7009     RawTextViewer.style.top = Number(y) + 30
7010     RawTextViewer.style.left = 100;
7011     RawTextViewer.style.height = window.innerHeight - 20;
7012     //RawTextViewer.style.opacity = 1.0;
7013     //RawTextViewer.style.backgroundColor = "rgba(0,0,0,0)";
7014     RawTextViewer.style.backgroundColor = "rgba(255,255,255,0.8)";
7015     RawTextViewer.style.zIndex = 1000;
7016     RawTextViewer.style.display = true;
7017
7018     if( RawTextViewerClose.style == null ){
7019         RawTextViewerClose.style = "";
7020     }
7021     RawTextViewerClose.style.top = Number(y) + 10
7022     RawTextViewerClose.style.left = 100;
7023     RawTextViewerClose.style.zIndex = 1001;
7024
7025     ScrollToElement(CurElement,RawTextViewerClose)
7026 }
7027 function hideRawTextViewer(){
7028     RawTextViewer.style.left = 10000;
7029     RawTextViewer.style.zIndex = -100;
7030     RawTextViewer.style.opacity = 0.0;
7031     RawTextViewer.style = null;
7032     RawTextViewer.innerHTML = "";
7033
7034     GshMenuSign.style.color = "rgba(255,128,128,1.0)";
7035     RawTextViewerClose.style.top = 0;
7036     RawTextViewerClose.style = null
7037 }
7038
7039 // source code viewer
7040 function frame_close(){
7041     srcframe = document.getElementById("src-frame");
7042     srcframe.innerHTML = "";
7043     //srcframe.style.cols = 1;
7044     srcframe.style.rows = 1;
7045     srcframe.style.height = 0;
7046     srcframe.style.display = false;
7047     src = document.getElementById("src-frame-textarea");
7048     src.innerHTML = "";
7049     //src.cols = 0
7050     src.rows = 0
7051     src.display = false
7052     //alert("--closed--")
7053 }
7054 //<-- | <span onclick="html_view();">Source</span> -->
7055 //<-- | <span onclick="frame_close();">SourceClose</span> -->
7056 //<--| <span>Download</span> -->
7057 function frame_open(){
7058     document.getElementById('GshFaviconURL').href = "";
7059     oldsrc = document.getElementById("GENSRC");
7060     if( oldsrc != null ){
7061         //alert("--I--(erasing old text)")
7062         oldsrc.innerHTML = "";
7063         return
7064     }else{
7065         //alert("--I--(no old text)")
7066     }
7067     styleBanner = GshBanner.getAttribute("style")
7068     GshBanner.removeAttribute("style")

```

```

7069 styleFooter = GshFooter.getAttribute("style")
7070 GshFooter.removeAttribute("style")
7071 if( document.getElementById('GJC_1') ) { GJC_1.remove() }
7072
7073 GshFaviconURL.href = "";
7074 GStat.removeAttribute('style')
7075 GshGrid.removeAttribute('style')
7076 GshMenuSign.removeAttribute('style')
7077 //GPos.removeAttribute('style')
7078 //GPos.innerHTML = "";
7079 //GLog.removeAttribute('style')
7080 //GLog.innerHTML = "";
7081 GMenu.removeAttribute('style')
7082 GTop.removeAttribute('style')
7083 GShellPlane.removeAttribute('style')
7084 RawTextViewer.removeAttribute('style')
7085 RawTextViewerClose.removeAttribute('style')
7086
7087 if( DestroyGJLink != null ) DestroyGJLink();
7088 GJFactory_Destroy()
7089
7090 src = document.getElementById("gsh");
7091 srchtml = src.outerHTML
7092 srcframe = document.getElementById("src-frame");
7093 srcframe.innerHTML =
7094 + "<+" + "cite id=\"GENSRC\">\n"
7095 + "<+" + "style>\n"
7096 + "#GENSRC textarea(tab-size:4;)\n"
7097 + "#GENSRC textarea(-o-tab-size:4;)\n"
7098 + "#GENSRC textarea(-moz-tab-size:4;)\n"
7099 + "#GENSRC textarea(spellcheck:false;)\n"
7100 + "</" + "style>\n"
7101 + "<+" + "textarea id=\"src-frame-textarea\" cols=100 rows=20 class=\"gsh-code\">"
7102 + "</" + "html>\n" // lost preamble text
7103 + srchtml
7104 + "</" + "html>\n" // lost trail text
7105 + "</" + "textarea>\n"
7106 + "</" + "cite><!-- GENSRC -->\n";
7107
7108 //srcframe.style.cols = 80;
7109 //srcframe.style.rows = 80;
7110
7111 GshBanner.setAttribute('style',styleBanner)
7112 GshFooter.setAttribute('style',styleFooter)
7113 }
7114 function fill_CSSView(){
7115     part = document.getElementById('GshStyleDef')
7116     view = document.getElementById('gsh-style-view')
7117     view.innerHTML =
7118     + "<+" + "textarea cols=100 rows=20 class=\"gsh-code\">"
7119     + part.innerHTML
7120     + "</" + "/textarea"
7121 }
7122 function fill_JavaScriptView(){
7123     jspart = document.getElementById('gsh-script')
7124     view = document.getElementById('gsh-script-view')
7125     view.innerHTML =
7126     + "<+" + "textarea cols=100 rows=20 class=\"gsh-code\">"
7127     + jspart.innerHTML
7128     + "</" + "/textarea"
7129 }
7130 function fill_DataView(){
7131     part = document.getElementById('gsh-data')
7132     view = document.getElementById('gsh-data-view')
7133     view.innerHTML =
7134     + "<+" + "textarea cols=100 rows=20 class=\"gsh-code\">"
7135     + part.innerHTML
7136     + "</" + "/textarea"
7137 }
7138 function jumpTo_StyleView(){
7139     jsvview = document.getElementById('html-src')
7140     jsvview.open = true
7141     jsvview = document.getElementById('gsh-style-frame')
7142     jsvview.open = true
7143     fill_CSSView()
7144 }
7145 function jumpTo_JavaScriptView(){
7146     jsvview = document.getElementById('html-src')
7147     jsvview.open = true
7148     jsvview = document.getElementById('gsh-script-frame')
7149     jsvview.open = true
7150     fill_JavaScriptView()
7151 }
7152 function jumpTo_DataView(){
7153     jsvview = document.getElementById('html-src')
7154     jsvview.open = true
7155     jsvview = document.getElementById('gsh-data-frame')
7156     jsvview.open = true
7157     fill_DataView()
7158 }
7159 function jumpTo_WholeView(){
7160     jsvview = document.getElementById('html-src')
7161     jsvview.open = true
7162     jsvview = document.getElementById('gsh-whole-view')
7163     jsvview.open = true
7164     frame_open()
7165 }
7166 function html_view(){
7167     html_stop();
7168
7169     banner = document.getElementById('GshBanner').style.backgroundImage;
7170     footer = document.getElementById('GshFooter').style.backgroundImage;
7171     document.getElementById('GshBanner').style.backgroundImage = "";
7172     document.getElementById('GshBanner').style.backgroundPosition = "";
7173     document.getElementById('GshFooter').style.backgroundImage = "";
7174
7175     //srcwin = window.open("", "CodeView2", "");
7176     srcwin = window.open("", "", "");
7177     srcwin.document.write("<span id=\"gsh\">\n");
7178
7179     src = document.getElementById("gsh");
7180     srcwin.document.write("<+" + "style>\n");
7181     srcwin.document.write("textarea(tab-size:4;)\n");
7182     srcwin.document.write("textarea(-o-tab-size:4;)\n");
7183     srcwin.document.write("textarea(-moz-tab-size:4;)\n");
7184     srcwin.document.write("</style>\n");
7185     srcwin.document.write("<h2>\n");
7186     srcwin.document.write("<+" + "span onclick=\"window.close();\">Close</span> | \n");
7187     //srcwin.document.write("<+" + "span onclick=\"html_stop();\">Run</span>\n");
7188     srcwin.document.write("</h2>\n");
7189     srcwin.document.write("<+" + "textarea id=\"gsh-src-src\" cols=100 rows=60>");
7190     srcwin.document.write("</" + "html>\n");
7191     srcwin.document.write("<+" + "span id=\"gsh\">");
7192     srcwin.document.write(src.innerHTML);

```

```

7193 srcwin.document.write("<"+"/span><"+"/html>\n");
7194 srcwin.document.write("<"/"+textarea>\n");
7195
7196 document.getElementById('GshBanner').style.backgroundImage = banner;
7197 document.getElementById('GshFooter').style.backgroundImage = footer;
7198
7199 sty = document.getElementById("GshStyleDef");
7200 srcwin.document.write("<"+style>\n");
7201 srcwin.document.write(sty.innerHTML);
7202 srcwin.document.write("<"+style>\n");
7203
7204 run = document.getElementById("gsh-script");
7205 srcwin.document.write("<"+script>\n");
7206 srcwin.document.write(run.innerHTML);
7207 srcwin.document.write("<"+script>\n");
7208
7209 srcwin.document.write("<"+span><"+/html>\n"); // gsh span
7210 srcwin.document.close();
7211 srcwin.focus();
7212 }
7213 GSH = document.getElementById("gsh")
7214
7215 //GSH.onclick = "alert('Ouch!')";
7216 //GSH.css = "{background-color:#eef;}"
7217 //GSH.style = "background-color:#eef;";
7218 //GSH.style.display = false;
7219 //alert('Ouch01')
7220 //GSH.style.display = true;
7221
7222 // 2020-0904 created, tentative
7223 document.addEventListener('keydown',jgshCommand);
7224 //CurElement = GshStatement
7225 CurElement = GshMenu
7226 MemElement = GshMenu
7227
7228 function nextSib(e){
7229   n = e.nextSibling;
7230   for( i = 0; i < 100; i++ ){
7231     if( n == null ){
7232       break;
7233     }
7234     if( n.nodeName == "DETAILS" ){
7235       return n;
7236     }
7237     n = n.nextSibling;
7238   }
7239   return null;
7240 }
7241 function prevSib(e){
7242   n = e.previousSibling;
7243   for( i = 0; i < 100; i++ ){
7244     if( n == null ){
7245       break;
7246     }
7247     if( n.nodeName == "DETAILS" ){
7248       return n;
7249     }
7250     n = n.previousSibling;
7251   }
7252   return null;
7253 }
7254 function setColor(e,eName,eColor){
7255   if( e.childNodes() ){
7256     s = e.childNodes();
7257     if( s != null ){
7258       for( ci = 0; ci < s.length; ci++ ){
7259         if( s[ci].nodeName == eName ){
7260           s[ci].style.color = eColor;
7261           //s[ci].style.backgroundColor = eColor;
7262           break;
7263         }
7264       }
7265     }
7266   }
7267 }
7268
7269 // https://docs.microsoft.com/en-us/previous-versions//hh781509(v=vs.85)
7270 function showCurElementPosition(ev){
7271 //  if( document.getElementById("GPos") == null ){
7272 //    return;
7273 //  }
7274 //  if( GPos == null ){
7275 //    return;
7276 //  }
7277 e = CurElement
7278 y = e.getBoundingClientRect().top.toFixed(0)
7279 x = e.getBoundingClientRect().left.toFixed(0)
7280
7281 h = ev + " "
7282 h += "y='"+y+"', "+ 'x=' +x+ " -- "
7283 h += "w=" + window.innerWidth + ", h=" + window.innerHeight + " -- "
7284 //GPos.test = h
7285 //GPos.innerHTML = h
7286 GPos.innerHTML = h
7287 }
7288
7289 function DateShort(){
7290   d = new Date()
7291   return d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7292   + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7293 }
7294 function DateLong(){
7295   d = new Date()
7296   return d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7297   + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7298   + " " + d.getMilliseconds()
7299   + " " + d.getTimezoneOffset()/60
7300   + " "
7301   + d.getTime() + " " + d.getMilliseconds()
7302 }
7303 }
7304 function GshellMenu(e){
7305   //GLog.innerHTML = "Hello, World! (" + DateLong() + ")"
7306   showGShellPlane()
7307 }
7308 // placements of planes
7309 function GshellResizeX(ev){
7310   //if( document.getElementById("GMENU") != null ){
7311   GMENU.style.left = window.innerWidth - 100
7312   GMENU.style.top = window.innerHeight - 90 - 200
7313   //console.log("place GMENU "+GMENU.style.left+" "+GMENU.style.top)
7314
7315 //}
7316 GStat.style.width = window.innerWidth

```

```

7317 //if( document.getElementById("GPos") != null ){
7318   //GPos.style.width = window.innerWidth
7319   //GPos.style.top = window.innerHeight - 30; //GPos.style.height
7320 //}
7321 //if( document.getElementById("GLog") != null ){
7322   // GLog.style.width = window.innerWidth
7323   //GLog.innerHTML = ""
7324 //}
7325 //if( document.getElementById("GLog") != null ){
7326   //GLog.innerHTML = "Resize: w=" + window.innerWidth +
7327   //", h=" + window.innerHeight
7328 //}
7329 }
7330 showCurElementPosition(ev)
7331 }
7332 function GShellResize(){
7333   GShellResizeX("[RESIZE]")
7334 }
7335 window.onresize = GShellResize
7336 var prevNode = null
7337 function GJSH_OnMouseMove(ev){
7338   x = ev.clientX
7339   y = ev.clientY
7340   d = new Date()
7341   if( document.elementFromPoint ){
7342     e = document.elementFromPoint(x,y)
7343     if( e != null ){
7344       if( e == prevNode ){
7345         }else{
7346           console.log(t+'('+x+','+y+')' +
7347             '+e.nodeType'+ ' '+e.tagName+'#'+e.id)
7348           prevNode = e
7349         }
7350     }else{
7351       console.log(t+'('+x+','+y+') no element')
7352     }
7353   }else{
7354     console.log(t+'('+x+','+y+') no elementFromPoint')
7355   }
7356 }
7357 window.addEventListener('mousemove',GJSH_OnMouseMove);
7358
7359 function GJSH_OnMouseMoveScreen(ev){
7360   x = ev.screenX
7361   y = ev.screenY
7362   d = new Date()
7363   t = d.getTime() / 1000
7364   console.log(t+'('+x+','+y+') no elementFromPoint')
7365 }
7366 //screen.addEventListener('mousemove',GJSH_OnMouseMoveScreen);
7367
7368 function ScrollToElement(oe,ne){
7369   ne.scrollIntoView()
7370   ny = ne.getBoundingClientRect().top.toFixed(0)
7371   nx = ne.getBoundingClientRect().left.toFixed(0)
7372   //GLog.innerHTML = "["+ny+","+nx+"]"
7373   //window.scrollTo(0,0)
7374
7375   GTop.style.backgroundColor = "rgba(0,0,0,0.0)"
7376   GshGrid.style.left = '250px';
7377   GshGrid.style.zIndex = 0
7378   if( false ){
7379     oy = oe.getBoundingClientRect().top.toFixed(0)
7380     ox = oe.getBoundingClientRect().left.toFixed(0)
7381     y = e.getBoundingClientRect().top.toFixed(0)
7382     x = e.getBoundingClientRect().left.toFixed(0)
7383     window.scrollTo(x,y)
7384     ny = e.getBoundingClientRect().top.toFixed(0)
7385     nx = e.getBoundingClientRect().left.toFixed(0)
7386     //GLog.innerHTML = "[ "+oy+", "+ox+" ]->[ "+y+", "+x+" ]->[ "+ny+", "+nx+" ]"
7387   }
7388 }
7389 function showGShellPlane(){
7390   if( GShellPlane.style.zIndex == 0 ){
7391     GShellPlane.style.zIndex = 1000;
7392     GShellPlane.style.left = 30;
7393     GShellPlane.style.height = 320;
7394     GShellPlane.innerHTML = DateLong() + "<br>" +
7395     "-- History --<br>" + MyHistory;
7396   }else{
7397     GShellPlane.style.zIndex = 0;
7398     GShellPlane.style.left = 0;
7399     GShellPlane.style.height = 50;
7400     GShellPlane.innerHTML = "";
7401   }
7402 }
7403 var SuppressGJShell = false
7404 function jgshCommand(keyevent){
7405   if( SuppressGJShell ){
7406     return
7407   }
7408   key = keyevent
7409   keycode = key.code
7410   //GStat.style.width = window.innerWidth
7411   GStat.style.backgroundColor = "rgba(0,0,0,0.4)"
7412
7413   console.log("JSGsh-Key:" +keycode+"(^~)//")
7414   if( keycode == "Slash" ){
7415     console.log('('+x+','+y+')')
7416     e = document.elementFromPoint(x,y)
7417     console.log('('+x+','+y+') '+e.nodeType'+ ' '+e.tagName+'#'+e.id)
7418   }else
7419   if( keycode == "Digit0" ){ // fold side-bar
7420     // "Zero page"
7421     showGShellPlane();
7422   }else
7423   if( keycode == "Digit1" ){ // fold side-bar
7424     primary.style.width = "94%"
7425     secondary.style.width = "0%"
7426     secondary.style.opacity = 0
7427     GStat.innerHTML = "[Single Column View]"
7428   }else
7429   if( keycode == "Digit2" ){ // unfold side-bar
7430     primary.style.width = "58%"
7431     secondary.style.width = "36%"
7432     secondary.style.opacity = 1
7433     GStat.innerHTML = "[Double Column View]"
7434   }else
7435   if( keycode == "KeyU" ){ // fold/unfold all
7436     html_fold(GshMenuFold);
7437     location.href = "#"+CurElement.id;
7438   }else
7439   if( keycode == "KeyO" || keycode == "ArrowRight" ){ // fold the element
7440     CurElement.open = !CurElement.open;

```

```

7441 }else
7442 if( keycode == "ArrowRight" ){ // unfold the element
7443   CurElement.open = true
7444 }else
7445 if( keycode == "ArrowLeft" ){ // unfold the element
7446   CurElement.open = false
7447 }else
7448 if( keycode == "KeyI" ){ // inspect the element
7449   e = CurElement
7450   //GLog.innerHTML =
7451   GJLog_append("Current Element: " + e + "<br>" +
7452     + "name='"+e.nodeName + "', "
7453     + "id='"+e.id + "', "
7454     + "children='"+e.childNodes.length + ", "
7455     + "parent='"+e.parentNode.id + "<br>" +
7456     + "text='"+e.textContent)
7457   GStat.style.backgroundColor = "rgba(0,0,0,0.8)"
7458   return
7459 }else
7460 if( keycode == "KeyM" ){ // memory the position
7461   MemElement = CurElement
7462 }else
7463 if( keycode == "KeyN" || keycode == "ArrowDown" ){ // next element
7464   e = nextSib(CurElement)
7465   if( e != null ){
7466     setColor(CurElement,"SUMMARY","#fff")
7467     setColor(e,"SUMMARY","#8f8") // should be complement ?
7468     oe = CurElement
7469     CurElement = e
7470     //location.href = "#"+e.id;
7471     ScrollToElement(oe,e)
7472   }
7473 }else
7474 if( keycode == "KeyP" || keycode == "ArrowUp" ){ // previous element
7475   oe = CurElement
7476   e = prevSib(CurElement)
7477   if( e != null ){
7478     setColor(CurElement,"SUMMARY","#fff")
7479     setColor(e,"SUMMARY","#8f8") // should be complement ?
7480     CurElement = e
7481     //location.href = "#"+e.id;
7482     ScrollToElement(oe,e)
7483   }else{
7484     e = document.getElementById("GshBanner")
7485     if( e != null ){
7486       setColor(CurElement,"SUMMARY","#fff")
7487       CurElement = e
7488       ScrollToElement(oe,e)
7489     }else{
7490       e = document.getElementById("primary")
7491       if( e != null ){
7492         setColor(CurElement,"SUMMARY","#fff")
7493         CurElement = e
7494         ScrollToElement(oe,e)
7495       }
7496     }
7497   }
7498 }else
7499 if( keycode == "KeyR" ){
7500   location.reload()
7501 }else
7502 if( keycode == "KeyJ" ){
7503   GshGrid.style.top = '120px';
7504   GshGrid.innerHTML = '>_<}{Down}';
7505 }else
7506 if( keycode == "KeyK" ){
7507   GshGrid.style.top = '0px';
7508   GshGrid.innerHTML = '^-^}{Up}';
7509 }else
7510 if( keycode == "KeyH" ){
7511   GshGrid.style.left = '0px';
7512   GshGrid.innerHTML = "'-'}{Left}";
7513 }else
7514 if( keycode == "KeyL" ){
7515   //GLog.innerHTML +=
7516   GJLog_append(
7517     'screen=' + screen.width + 'px' + '<br>' +
7518     'window=' + window.innerWidth + 'px' + '<br>' +
7519   )
7520   GshGrid.style.left = (document.documentElement.clientWidth - 160).toString(10) + 'px';
7521   GshGrid.innerHTML = '(@_@}{Right)';
7522 }else
7523 if( keycode == "KeyS" ){
7524   html_stop(GshMenuStop,true)
7525 }else
7526 if( keycode == "KeyF" ){
7527   html_fork()
7528 }else
7529 if( keycode == "KeyC" ){
7530   window.close()
7531 }else
7532 if( keycode == "KeyD" ){
7533   html_digest()
7534 }else
7535 if( keycode == "KeyV" ){
7536   e = document.getElementById('gsh-digest')
7537   if( e != null ){
7538     showDigest(e)
7539   }
7540 }
7541 showCurElementPosition("[+key.code+] --");
7542 //if( document.getElementById("GPos") != null ){
7543 //  GPos.innerHTML += "[+key.code+] --"
7544 //}
7545 //GShellResizeX("[+key.code+] --");
7546 GShellResizeX("[INIT]");
7547 }
7548 GShellResizeX("DisplaySize = '-- Display: '+ 'screen=' + screen.width + 'px, +' + 'window=' + window.innerWidth + 'px';
7549
7550 let {text, digest} = getDigest()
7551 //GLog.innerHTML +=
7552 GJLog_append(
7553   '-- GShell: ' + GshVersion.innerHTML + '\n' +
7554   '-- Digest: ' + digest + '\n' +
7555   'DisplaySize
7556   //+ "<br>" + "-- LastVisit:<br>" + MyHistory
7557 )
7558 GShellResizeX(null);
7559
7560 // <a href="https://www.w3.org/TR/WebCryptoAPI/">Web Cryptography API</a>
7561 //Convert a string into an ArrayBuffer
7562 //from https://developers.google.com/web/updates/2012/06/How-to-convert-ArrayBuffer-to-and-from-String

```

```

7565 function str2ab(str) {
7566   const buf = new ArrayBuffer(str.length);
7567   const bufView = new Uint8Array(buf);
7568   for (let i = 0, strLen = str.length; i < strLen; i++) {
7569     bufView[i] = str.charCodeAt(i);
7570   }
7571   return buf;
7572 }
7573 function importPrivateKey(pem) {
7574   const binaryDerString = window.atob(pemContents);
7575   const binaryDer = str2ab(binaryDerString);
7576   return window.crypto.subtle.importKey(
7577     "pkcs8",
7578     binaryDer,
7579     {
7580       name: "RSA-PSS",
7581       modulusLength: 2048,
7582       publicExponent: new Uint8Array([1, 0, 1]),
7583       hash: "SHA-256",
7584     },
7585     true,
7586     ["sign"]
7587   );
7588 }
7589 //importPrivateKey(ppem)
7590
7591 //key = {}
7592 //buf = "abc"
7593 //enc = "xyzxxxxx"; //crypto.publicEncrypt(key,buf)
7594 //b64 = btoa(enc)
7595 //dec = atob(b64)
7596 //GLog.innerHTML = "enc:" + b64 + ", dec:" + dec
7597
7598 </script>
7599
7600 <span id="gjc" data-title="GJConsole" data-author="sato@its-more.jp">
7601 <!-- ----- GJConsole BEGIN { ----- -->
7602 <p>
7603 <span id="GJE_RootNode0"></span>
7604 </p>
7605 <style id="GJConsoleStyle">
7606   .GJConsole {
7607     z-index:1000;
7608     width:400; height:200px;
7609     margin:2px;
7610     color:#ffff; background-color:#66a;
7611     font-size:12px; font-family:monospace,Courier New;
7612   }
7613 </style>
7614
7615 <script id="GJConsoleScript" class="GJConsole">
7616   var PS1 = "% "
7617   function GJC_KeyDown(keyevent){
7618     key = keyevent.code
7619     if( key == "Enter" ){
7620       GJC_Command(this)
7621       this.value += "\n" + PS1 // prompt
7622     }else
7623     if( key == "Escape"){
7624       SuppressGJShell = false
7625       GshMenu.focus() // should be previous focus
7626     }
7627   }
7628   var GJC_SessionId
7629   function GJC_SetSessionId(){
7630     var xd = new Date()
7631     GJC_SessionId = xd.getTime() / 1000
7632   }
7633   GJC_SetSessionId()
7634   function GJC_Memory(mem,args,text){
7635     argv = args.split(' ')
7636     cmd = argv[0]
7637     argv.shift()
7638     args = argv.join(' ')
7639     ret = ""
7640
7641     if( cmd == 'clear' ){
7642       Permanent.setItem(mem,'')
7643     }else
7644     if( cmd == 'read' ){
7645       ret = Permanent.getItem(mem)
7646     }else
7647     if( cmd == 'save' ){
7648       val = Permanent.getItem(mem)
7649       if( val == null ){ val = "" }
7650       d = new Date()
7651       val += d.getTime()/1000+" "+GJC_SessionId+" "+document.URL+" "+args+"\n"
7652       val += text.value
7653       Permanent.setItem(mem,val)
7654     }else
7655     if( cmd == 'write' ){
7656       val = Permanent.getItem(mem)
7657       if( val == null ){ val = "" }
7658       d = new Date()
7659       val += d.getTime()/1000+" "+GJC_SessionId+" "+document.URL+" "+args+"\n"
7660       Permanent.setItem(mem,val)
7661     }else{
7662       ret = "Commands: write | read | save | clear"
7663     }
7664   return ret
7665 }
7666 // -- 2020-09-14 added TableEditor
7667 var GJE_CurElement = null; //GJE_RootNode
7668 GJE_NodeSaved = null
7669 GJE_TableNo = 1
7670 function GJE_StyleKeyCommand(kev){
7671   keycode = kev.code
7672   console.log('GJE-Key: '+keycode)
7673   if( keycode == 'Escape' ){
7674     GJE_SetStyle(this);
7675   }
7676   kev.stopPropagation()
7677   // https://developer.mozilla.org/en-US/docs/Web/API/Event/stopPropagation
7678 }
7679 var GJE_CommandMode = false
7680 function GJE_TableKeyCommand(kev,tab){
7681   wasCmdMode = GJE_CommandMode
7682   key = kev.code
7683   if( key == 'Escape' ){
7684     console.log("To command mode: "+tab.nodeName+"#"+tab.id)
7685     //tab.setAttribute('contenteditable','false')
7686     tab.style.caretColor = "blue"
7687     GJE_CommandMode = true
7688   }else

```

```

7689     if( key == "KeyA" ){
7690         tab.style.caretColor = "red"
7691         GJE_CommandMode = false
7692     }else{
7693     if( key == "KeyI" ){
7694         tab.style.caretColor = "red"
7695         GJE_CommandMode = false
7696     }else{
7697     if( key == "KeyO" ){
7698         tab.style.caretColor = "red"
7699         GJE_CommandMode = false
7700     }else{
7701     if( key == "KeyY" ){
7702         console.log("ROW-DOWN")
7703     }else{
7704     if( key == "KeyK" ){
7705         console.log("ROW-UP")
7706     }else{
7707     if( key == "KeyW" ){
7708         console.log("COL-FORW")
7709     }else{
7710     if( key == "KeyB" ){
7711         console.log("COL-BACK")
7712     }
7713
7714     kev.stopPropagation()
7715     if( wasCmdMode ){
7716         kev.preventDefault()
7717     }
7718 }
7719 function GJE_DragEvent(ev,elem){
7720     x = ev.clientX
7721     y = ev.clientY
7722     console.log("Dragged: "+this.nodeName+'#'+this.id+' x='+x+ ' y=' +y)
7723 }
7724 // https://developer.mozilla.org/en-US/docs/Web/API/DragEvent
7725 // https://www.w3.org/TR/uievents/#events-mouseevents
7726 function GJE_DropEvent(ev,elem){
7727     x = ev.clientX
7728     y = ev.clientY
7729     this.style.x = x
7730     this.style.y = y
7731     this.style.position = 'absolute' // 'fixed'
7732     this.parentNode = gsh // just for test
7733     console.log("Dropped: "+this.nodeName+'#'+this.id+' x='+x+ ' y=' +y
7734     +' parent=' +this.parentNode.id)
7735 }
7736 function GJE_SetTableStyle(ev){
7737     this.innerHTML = this.value; // sync. for external representation?
7738     if(false){
7739         stdid = this.parentNode.id+this.id
7740         // and remove "_span" at the end
7741         e = document.getElementById(stdid)
7742         //alert('SetTableStyle #' +e.id+'\n'+this.value)
7743         if( e != null ){
7744             e.innerHTML = this.value
7745         }else{
7746             console.log('Style Not found: '+stdid)
7747         }
7748         //alert('event StopPropagation: '+ev)
7749     }
7750 }
7751 function setCSSofClass(cclass,cstyle){
7752     const ss = document.styleSheets[3]; // 0, 1, 2, 3, ... ?
7753     rlen = ss.cssRules.length;
7754     let tabrule = null;
7755     rulex = -1
7756
7757     // should skip white space at the top of cstyle
7758     sel = cstyle.charAt(0);
7759     selector = sel+cclass;
7760     console.log('-- search style rule for '+selector)
7761
7762     for(let i = 0; i < rlen; i++){
7763         cr = ss.cssRules[i];
7764         console.log('CSS rule ['+i+'/'+rlen+'] '+cr.selectorText);
7765         if( cr.selectorText === selector ) // css class selector
7766             tabrule = ss.cssRules[i];
7767             console.log('CSS rule found for:['+i+'/'+rlen+'] '+selector);
7768             ss.deleteRule(i);
7769             //rlen = ss.cssRules.length;
7770             rulex = i
7771             // should search and replace the property here
7772     }
7773 }
7774 // https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleSheet/insertRule
7775 if( tabrule == null ){
7776     console.log('CSS rule NOT found for:[ '+rlen+'] '+selector);
7777     ss.insertRule(cstyle,rlen);
7778     ss.insertRule(cstyle,0); // override by 0?
7779     console.log('CSS rule inserted:[ '+rlen+']\n'+cstyle);
7780 }else{
7781     ss.insertRule(cstyle,rlen);
7782     ss.insertRule(cstyle,0);
7783     console.log('CSS rule replaced:[ '+rlen+']\n'+cstyle);
7784 }
7785 }
7786 function GJE_SetStyle(te){
7787     console.log('Apply the style to:' +te.id+'\n');
7788     console.log('Apply the style to:' +te.parentNode.id+'\n');
7789     console.log('Apply the style to:' +te.parentNode.className+'\n');
7790     cclass = te.parentNode.className;
7791     setCSSofClass(cclass,te.value); // should get selector part from
7792     // selector { rules }
7793
7794     if(false){
7795         //console.log('Apply the style:')
7796         //stdid = this.parentNode.id+this.id+
7797         //stdid = this.id+" .style"
7798         css = te.value
7799         stdid = te.parentNode.id+".style"
7800         e = document.getElementById(stdid)
7801         if( e != null ){
7802             //console.log('Apply the style:' +e.id+'\n'+te.value);
7803             console.log('Apply the style:' +e.id+'\n'+css);
7804             e.innerHTML = css; //te.value;
7805             //ncss = e.sheet;
7806             //ncss.insertRule(te.value,ncss.cssRules.length);
7807         }else{
7808             console.log('No element to Apply the style: '+stdid)
7809         }
7810         tblid = te.parentNode.id+" .table";
7811         e = document.getElementById(tblid);
7812         if( e != null ){
7813
7814
7815
7816
7817
7818
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
8000
8001
8002
8003
8004
8005
8006
8007
8008
8009
8010
8011
8012
8013
8014
8015
8016
8017
8018
8019
8020
8021
8022
8023
8024
8025
8026
8027
8028
8029
8030
8031
8032
8033
8034
8035
8036
8037
8038
8039
8040
8041
8042
8043
8044
8045
8046
8047
8048
8049
8050
8051
8052
8053
8054
8055
8056
8057
8058
8059
8060
8061
8062
8063
8064
8065
8066
8067
8068
8069
8070
8071
8072
8073
8074
8075
8076
8077
8078
8079
8080
8081
8082
8083
8084
8085
8086
8087
8088
8089
8090
8091
8092
8093
8094
8095
8096
8097
8098
8099
8100
8101
8102
8103
8104
8105
8106
8107
8108
8109
8110
8111
8112
8113
8114
8115
8116
8117
8118
8119
8120
8121
8122
8123
8124
8125
8126
8127
8128
8129
8130
8131
8132
8133
8134
8135
8136
8137
8138
8139
8140
8141
8142
8143
8144
8145
8146
8147
8148
8149
8150
8151
8152
8153
8154
8155
8156
8157
8158
8159
8160
8161
8162
8163
8164
8165
8166
8167
8168
8169
8170
8171
8172
8173
8174
8175
8176
8177
8178
8179
8180
8181
8182
8183
8184
8185
8186
8187
8188
8189
8190
8191
8192
8193
8194
8195
8196
8197
8198
8199
8200
8201
8202
8203
8204
8205
8206
8207
8208
8209
8210
8211
8212
8213
8214
8215
8216
8217
8218
8219
8220
8221
8222
8223
8224
8225
8226
8227
8228
8229
8230
8231
8232
8233
8234
8235
8236
8237
8238
8239
8240
8241
8242
8243
8244
8245
8246
8247
8248
8249
8250
8251
8252
8253
8254
8255
8256
8257
8258
8259
8260
8261
8262
8263
8264
8265
8266
8267
8268
8269
8270
8271
8272
8273
8274
8275
8276
8277
8278
8279
8280
8281
8282
8283
8284
8285
8286
8287
8288
8289
8290
8291
8292
8293
8294
8295
8296
8297
8298
8299
8300
8301
8302
8303
8304
8305
8306
8307
8308
8309
8310
8311
8312
8313
8314
8315
8316
8317
8318
8319
8320
8321
8322
8323
8324
8325
8326
8327
8328
8329
8330
8331
8332
8333
8334
8335
8336
8337
8338
8339
8340
8341
8342
8343
8344
8345
8346
8347
8348
8349
8350
8351
8352
8353
8354
8355
8356
8357
8358
8359
8360
8361
8362
8363
8364
8365
8366
8367
8368
8369
8370
8371
8372
8373
8374
8375
8376
8377
8378
8379
8380
8381
8382
8383
8384
8385
8386
8387
8388
8389
8390
8391
8392
8393
8394
8395
8396
8397
8398
8399
8400
8401
8402
8403
8404
8405
8406
8407
8408
8409
8410
8411
8412
8413
8414
8415
8416
8417
8418
8419
8420
8421
8422
8423
8424
8425
8426
8427
8428
8429
8430
8431
8432
8433
8434
8435
8436
8437
8438
8439
8440
8441
8442
8443
8444
8445
8446
8447
8448
8449
8450
8451
8452
8453
8454
8455
8456
8457
8458
8459
8460
8461
8462
8463
8464
8465
8466
8467
8468
8469
8470
8471
8472
8473
8474
8475
8476
8477
8478
8479
8480
8481
8482
8483
8484
8485
8486
8487
8488
8489
8490
8491
8492
8493
8494
8495
8496
8497
8498
8499
8500
8501
8502
8503
8504
8505
8506
8507
8508
8509
8510
8511
8512
8513
8514
8515
8516
8517
8518
8519
8520
8521
8522
8523
8524
8525
8526
8527
8528
8529
8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8540
8541
8542
8543
8544
8545
8546
8547
8548
8549
8550
8551
8552
8553
8554
8555
8556
8557
8558
8559
8560
8561
8562
8563
8564
8565
8566
8567
8568
8569
8570
8571
8572
8573
8574
8575
8576
8577
8578
8579
8580
8581
8582
8583
8584
8585
8586
8587
8588
8589
8590
8591
8592
8593
8594
8595
8596
8597
8598
8599
8600
8601
8602
8603
8604
8605
8606
8607
8608
8609
8610
8611
8612
8613
8614
8615
8616
8617
8618
8619
8620
8621
8622
8623
8624
8625
8626
8627
8628
8629
8630
8631
8632
8633
8634
8635
8636
8637
8638
8639
8640
8641
8642
8643
8644
8645
8646
8647
8648
8649
8650
8651
8652
8653
8654
8655
8656
8657
8658
8659
8660
8661
8662
8663
8664
8665
8666
8667
8668
8669
8670
8671
8672
8673
8674
8675
8676
8677
8678
8679
8680
8681
8682
8683
8684
8685
8686
8687
8688
8689
8690
8691
8692
8693
8694
8695
8696
8697
8698
8699
8700
8701
8702
8703
8704
8705
8706
8707
8708
8709
8710
8711
8712
8713
8714
8715
8716
8717
8718
8719
8720
8721
8722
8723
8724
8725
8726
8727
8728
8729
8730
8731
8732
8733
8734
8735
8736
8737
8738
8739
8740
8741
8742
8743
8744
8745
8746
8747
8748
8749
8750
8751
8752
8753
8754
8755
8756
8757
8758
8759
8760
8761
8762
8763
8764
8765
8766
8767
8768
8769
8770
8771
8772
8773
8774
8775
8776
8777
8778
8779
8780
8781
8782
8783
8784
8785
8786
8787
8788
8789
8790
8791
8792
8793
8794
8795
8796
8797
8798
8799
8800
8801
8802
8803
8804
8805
8806
8807
8808
8809
8810
8811
8812
8813
8814
8815
8816
8817
8818
8819
8820
8821
8822
8823
8824
8825
8826
8827
8828
8829
8830
8831
8832
8833
8834
8835
8836
8837
8838
8839
8840
8841
8842
8843
8844
8845
8846
8847
8848
8849
8850
8851
8852
8853
8854
8855
8856
8857
8858
8859
8860
8861
8862
8863
8864
8865
8866
8867
8868
8869
8870
8871
8872
8873
8874
8875
8876
8877
8878
8879
8880
8881
8882
8883
8884
8885
8886
8887
8888
8889
8890
8891
8892
8893
8894
8895
8896
8897
8898
8899
8900
8901
8902
8903
8904
8905
8906
8907
8908
8909
8910
8911
8912
8913
8914
8915
8916
8917
8918
8919
8920
8921
8922
8923
8924
8925
8926
8927
8928
8929
8930
8931
8932
8933
8934
8935
8936
8937
8938
8939
8940
8941
8942
8943
8944
8945
8946
8947
8948
8949
8950
8951
8952
8953
8954
8955
8956
8957
8958
8959
8960
8961
8962
8963
8964
8965
8966
8967
8968
8969
8970
8971
8972
8973
8974
8975
8976
8977
8978
8979
8980
8981
8982
8983
8984
8985
8986
8987
8988
8989
8990
8991
8992
8993
8994
8995
8996
8997
8998
8999
9000
9001
9002
9003
9004
9005
9006
9007
9008
9009
9010
9011
9012
9013
9014
9015
9016
9017
9018
9019
9020
9021
9022
9023
9024
9025
9026
9027
9028
9029
9030
9031
9032
9033
9034
9035
9036
9037
9038
9039
9040
9041
9042
9043
9044
9045
9046
9047
9048
9049
9050
9051
9052
9053
9054
9055
9056
9057
9058
9059
9060
9061
9062
9063
9064
9065
9066
9067
9068
9069
9070
9071
9072
9073
9074
9075
9076
9077
9078
9079
9080
9081
9082
9083
9084
9085
9086
9087
9088
9089
9090
9091
9092
9093
9094
9095
9096
9097
9098
9099
9100
9101
9102
9103
9104
9105
9106
9107
9108
9109
9110
9111
9112
9113
9114
9115
9116
9117
9118
9119
9120
9121
9122
9123
9124
9125
9126
9127
9128
9129
9130
9131
9132
9133
9134
9135
9136
9137
9138
9139
9140
9141
9142
9143
9144
9145
9146
9147
9148
9149
9150
9151
9152
9153
9154
9155
9156
9157
9158
9159
9160
9161
9162
9163
9164
9165
9166
9167
9168
9169
9170
9171
9172
9173
9174
9175
9176
9177
9178
9179
9180
9181
9182
9183
9184
9185
9186
9187
9188
9189
9190
9191
9192
9193
9194
9195
9196
9197
9198
9199
9200
9201
9202
9203
9204
9205
9206
9207
9208
9209
9210
9211
9212
9213
9214
9215
9216
9217
9218
9219
9220
9221
9222
9223
9224
9225
9226
9227
9228
9229
9230
9231
9232
9233
9234
9235
9236
9237
9238
9239
9240
9241
9242
9243
9244
9245
9246
9247
9248
9249
9250
9251
9252
9253
9254
9255
9256
9257
9258
9259
9260
9261
9262
9263
9264
9265
9266
9267
9268
9269
9270
9271
9272
9273
9274
9275
9276
9277
9278
9279
9280
9281
9282
9283
9284
9285
9286
9287
9288
9289
9290
9291
9292
9293
9294
9295
9296
9297
9298
9299
9300
9301
9302
9303
9304
9305
9306
9307
9308
9309
9310
9311
9312
9313
9314
9315
9316
9317
9318
9319
9320
9321
9322
9323
9324
9325
9326
9327
9328
9329
9330
9331
9332
9333
9334
9335
9336
9337
9338
9339
9340
9341
9342
9343
9344
9345
9346
9347
9348
9349
9350
9351
9352
9353
9354
9355
9356
9357
9358
9359
9360
9361
9362
9363
9364
9365
9366
9367
9368
9369
9370
9371
9372
9373
9374
9375
9376
9377
9378
9379
9380
9381
9382
9383
9384
9385
9386
9387
9388
9389
9390
9391
9392
9393
9394
9395
9396
9397
9398
9399
9400
9401
9402
9403
9404
9405
9406
9407
9408
9409
9410
9411
9412
9413
9414
9415
9416
9417
9418
9419
9420
9421
9422
9423
9424
9425
9426
9427
9428
9429
9430
9431
9432
9433
9434
9435
9436
9437
9438
9439
9440
9441
9442
9443
9444
9445
9446
9447
9448
9449
9450
9451
9452
9453
9454
9455
9456
9457
9458
9459
9460
9461
9462
9463
9464
9465
9466
9467
9468
9469
9470
9471
9472
9473
9474
9475
9476
9477
9478
9479
9480
9481
9482
9483
9484
9485
9486
9487
9488
9489
9490
9491
9492
9493
9494
9495
9496
9497
9498
9499
9500
9501
9502
9503
9504
9505
9506
9507
9508
9509
9510
9511
9512
9513
9514
9515
9516
9517
9518
9519
9520
9521
9522
9523
9524
9525
9526
9527
9528
9529
9530
9531
9532
9533
9534
9535
9536
9537
9538
9539
9540
9541
9542
9543
9544
9545
9546
9547
9548
9549
9550
9551
9552
9553
9554
9555
9556
9557
9558
9559
9560
9561
9562
9563
9564
9565
9566
9567
9568
9569
9570
9571
9572
9573
9574
9575
9576
9577
9578
9579
9580
9581
9582
9583
9584
9585
9586
9587
9588
9589
9590
9591
9592
9593
9594
9595
9596
9597
9598
9599
9600
9601
9602
9603
9604
9605
9606
9607
9608
9609
9610
9611
9612
9613
9614
9615
9616
9617
9618
9619
9620
9621
9622
9623
9624
9625
9626
9627
9628
9629
9630
9631
9632
9633
9634
9635
9636
9637
9638
9639
9640
9641
9642
9643
9644
9645
9646
9647
9648
9649
9650
9651
9652
9653
9654
9655
9656
9657
9658
9659
9660
9661
9662
9663
9664
9665
9666
9667
9668
9669
9670
9671
9672
9673
9674
9675
9676
9677
9678
9679
9680
9681
9682
9683
9684
9685
9686
9687
9688
9689
9690
9691
9692
9693
9694
9695
9696
9697
9698
9699
9700
9701
9702
9703
9
```

```

7813     //e.setAttribute('style',css);
7814     e.setProperty('style',css,'!important');
7815   }
7816 }
7817 }
7818 function makeTable(argv){
7819   //tid =
7820   cwe = GJE_CurElement
7821   tid = 'table.' + GJE_TableNo
7822
7823   nt = new Text('\n')
7824   cwe.appendChild(nt)
7825
7826   ne = document.createElement('span'); // the container
7827   cwe.appendChild(ne)
7828   ne.id = tid + '-span'
7829   ne.setAttribute('contenteditable',true)
7830
7831   htspan = document.createElement('span'); // html part
7832   //htspan.id = tid + '-html'
7833   //ne.innerHTML = '\n'
7834   nt = new Text('\n')
7835   ne.appendChild(nt)
7836   ne.appendChild(htspan)
7837
7838   htspan.id = tid
7839   htspan.setAttribute('class',tid)
7840
7841   ne.setAttribute('draggable','true')
7842   ne.addEventListener('drag',GJE_DragEvent);
7843   ne.addEventListener('dragend',GJE_DropEvent);
7844
7845   var col = 3
7846   var row = 2
7847   if( argv[0] != null ){
7848     col = argv[0]
7849     argv.shift()
7850   }
7851   if( argv[0] != null ){
7852     row = argv[0]
7853     argv.shift()
7854   }
7855
7856   //ne.setAttribute('class',tid)
7857   ht = "\n"
7858   //ht += '<+'+table' + 'id="'+tid+'" + ' class="'+tid+'"
7859   ht += '<+'+table'
7860   + ' onkeydown="GJE_TableKeyCommand(event,this)"
7861   //+' ondrag="GJE_DragEvent(event,this)\"\n"
7862   //+' ondragend="GJE_DropEvent(event,this)\"\n"
7863   //+' draggable="true"\n"
7864   //+' contenteditable="true"
7865   + '>\n'
7866   ht += '<+'+tbody>\n';
7867   for( r = 0; r < row; r++ ){
7868     ht += '<+'+tr>\n'
7869     for( c = 0; c < col; c++ ){
7870       ht += "<"+td>\n"
7871       ht += "ABCDEFHIJKLMNOPQRSTUVWXYZ".charAt(c) + r
7872       ht += "<"+/td>\n"
7873     }
7874     ht += "<"+/tr>\n"
7875   }
7876   ht += '<+'+tbody>\n';
7877   ht += '<+'+table>\n';
7878   htspan.innerHTML = ht;
7879   nt = new Text('\n')
7880   ne.appendChild(nt)
7881
7882   st = '#'+tid+' *{\n' // # for instance specific
7883   +' '+border:1px solid #aaa;\n'
7884   +' '+background-color:#efef;\n'
7885   +' '+color:#222;\n'
7886   +' '+font-size:#14pt !important;\n'
7887   +' '+font-family:monospace,Courier New !important;\n'
7888   +' } /* hit ESC to apply */\n'
7889
7890   // wish script to be included
7891   //nj = document.createElement('script')
7892   //ne.appendChild(nj)
7893   //ne.innerHTML = 'function SetStyle(e){'
7894
7895   // selector seems lost in dynamic style appending
7896   if(false){
7897     ns = document.createElement('style')
7898     ne.appendChild(ns)
7899     ns.id = tid + '-style'
7900     ns.innerHTML = '\n'+st
7901     nt = new Text('\n')
7902     ne.appendChild(nt)
7903   }
7904   setCSSofClass(tid,st); // should be in JavaScript script?
7905
7906   nx = document.createElement('textarea')
7907   ne.appendChild(nx)
7908   nx.id = tid + '-style_def'
7909   nx.setAttribute('class','GJ_StyleEditor')
7910   nx.spellcheck = false
7911   nx.cols = 60
7912   nx.rows = 10
7913   nx.innerHTML = '\n'+st
7914   nx.addEventListener('change',GJE_SetTableStyle);
7915   nx.addEventListener('keydown',GJE_StyleKeyCommand);
7916   //nx.addEventListener('click',GJE_SetTableStyle);
7917
7918   nt = new Text('\n')
7919   cwe.appendChild(nt)
7920
7921   GJE_TableNo += 1
7922   return 'created TABLE id="'+tid+'"
7923 }
7924 function GJE_NodeEdit(argv){
7925   cwe = GJE_CurElement
7926   cmd = argv[0]
7927   argv.shift()
7928   args = argv.join(' ')
7929   ret = ""
7930
7931   if( cmd == '.u' || cmd == '.un' || cmd == 'undo' ){
7932     if( GJE_NodeSaved != null ){
7933       xn = GJE_RootNode
7934       GJE_RootNode = GJE_NodeSaved
7935       GJE_NodeSaved = xn
7936       ret = '-- did undo'

```

```

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

```

```

8061         text.value += e.outerHTML
8062     }else{
8063         text.value += "Not found: " + id
8064     }
8065   }else{
8066     text.value += GJE_RootNode.outerHTML
8067     //text.value += e.innerHTML
8068   }
8069 }else
8070 if( cmd == 'destroy' ){
8071   text.value += GJFactory_Destroy()
8072 }else
8073 if( cmd == 'save' ){
8074   e = document.getElementById('GJFactory')
8075   Permanent.setItem('GJFactory-1',e.innerHTML)
8076   text.value += "-- Saved GJFactory"
8077 }else
8078 if( cmd == 'load' ){
8079   gjf = Permanent.getItem('GJFactory-1')
8080   e = document.getElementById('GJFactory')
8081   e.innerHTML = gjf
8082   // must restore EventListener
8083   text.value += "-- EventListener was not restored"
8084 }else
8085 if( cmd.charAt(0) == '.' ){
8086   argv0 = args0.split('.')
8087   text.value += GJE_NodeEdit(argv0)
8088 }else
8089 if( cmd == 'cont' ){
8090   bannerIsStopping = false
8091   GshMenuStop.innerHTML = "Stop"
8092 }else
8093 if( cmd == 'date' ){
8094   text.value += DateLong()
8095 }else
8096 if( cmd == 'echo' ){
8097   text.value += args
8098 }else
8099 if( cmd == 'fork' ){
8100   html_fork()
8101 }else
8102 if( cmd == 'last' ){
8103   text.value += MyHistory
8104   //h = document.createElement("span")
8105   //h.innerHTML = MyHistory
8106   //text.value += h.innerHTML
8107   //tx = MyHistory.replace("\n","");
8108   //text.value += tx.replace("<"+br>","\n") + "xxxx<"+br>yyyy"
8109 }else
8110 if( cmd == 'ne' ){
8111   text.value += GJE_NodeEdit(argv)
8112 }else
8113 if( cmd == 'reload' ){
8114   location.reload()
8115 }else
8116 if( cmd == 'mem' ){
8117   text.value += GJC_Memory('GJC_Storage',args,text)
8118 }else
8119 if( cmd == 'stop' ){
8120   bannerIsStopping = true
8121   GshMenuStop.innerHTML = "Start"
8122 }else
8123 if( cmd == 'who' ){
8124   text.value += "SessionId="+GJC_SessionId+" "+document.URL
8125 }else
8126 if( cmd == 'wall' ){
8127   text.value += GJC_Memory('GJC_Wall','write',text)
8128 }else
8129 {
8130   text.value += "Commands: help | echo | date | last \n"
8131   + '          new | save | load | mem \n'
8132   + '          who | wall | fork | nife'
8133 }
8134 }
8135
8136 function GJC_Input(){
8137   if( this.value.endsWith("\n") ){ // remove NL added by textarea
8138     this.value = this.value.slice(0,this.value.length-1)
8139   }
8140 }
8141
8142 var GCJ_Id = null
8143 function GJC_Resize(){
8144   GJC_Id.style.zIndex = 20000
8145   GJC_Id.style.width = window.innerWidth - 16
8146   GJC_Id.style.height = 300
8147   GJC_Id.style.backgroundColor = "rgba(0,64,16,1.0)" // blackboard color
8148   GJC_Id.style.color = "rgba(255,255,255,1.0)"
8149 }
8150 function GJC_FocusIn(){
8151   this.spellcheck = false
8152   SuppressGJShell = true
8153   this.onkeydown = GJC_KeyDown
8154   GJC_Resize()
8155 }
8156 function GJC_FocusOut(){
8157   SuppressGJShell = false
8158   this.removeEventListener('keydown',GJC_KeyDown);
8159 }
8160 window.addEventListener('resize',GJC_Resize);
8161
8162 function GJC_OnStorage(e){
8163   //alert('Got Message')
8164   //GJC.value += "\n((ReceivedMessage))\n"
8165 }
8166 window.addEventListener('storage',GJC_OnStorage);
8167 //window.addEventListener('storage',()=>{alert('GotMessage')})
8168
8169 function GJC_Setup(gjcid){
8170   gjcid.style.width = gsh.getBoundingClientRect().width
8171   gjcid.value = "GJShell Console // " + GshVersion.innerHTML + "\n"
8172   //gjcid.value += "Date:" + DateLong() + "\n"
8173   gjcid.value += PS1
8174   gjcid.onfocus = GJC_FocusIn
8175   gjcid.addEventListener('input',GJC_Input);
8176   gjcid.addEventListener('focusout',GJC_FocusOut);
8177   GJC_Id = gjcid
8178 }
8179 function GJC_Clear(id){
8180 }
8181 if( document.getElementById("GJC_0") != null ){
8182   GJC_Setup(GJC_0)
8183 }else{
8184   document.write('<'+'textarea id="GJC_1" class="GJConsole"><+'/textarea>')

```

```

8185     GJC_Setup(GJC_1)
8186     factory = document.createElement('span');
8187     gsh.appendChild(factory)
8188     GJE_RootNode = factory;
8189     GJE_CurElement = GJE_RootNode;
8190   }
8191
8192   // TODO: focus handling
8193 </script>
8194 <style>
8195   .GJ_StyleEditor {
8196     font-size:9pt !important;
8197     font-family:Courier New, monospace !important;
8198   }
8199 </style>
8200
8201 <!-- ----- GJConsole END } ----- -->
8202 </span>
8203 */
8204
8205 /*
8206 <span id="BlinderText">
8207 <style id="BlinderTextStyle">
8208   .textField {
8209     display:inline;
8210     border:1px solid #000;
8211     color:#000; background-color:#fff;
8212     width:106pt; height:16pt;
8213     padding:2px;
8214     resize:none;
8215     vertical-align:middle;
8216     font-size:10pt; font-family:Courier New;
8217   }
8218   .VisibleText {
8219   }
8220   .BlinderText {
8221     color:#000; background-color:#eee;
8222   }
8223   .joinButton {
8224     font-family:Georgia;
8225     font-size:11pt;
8226     line-height:1.1;
8227     height:16pt;
8228     width:50pt;
8229     padding:3px;
8230     text-align:center !important;
8231     border-color:#aaa !important;
8232     color:#fff; background-color:#4a4 !important;
8233     vertical-align:middle !important;
8234   }
8235   .SendButton {
8236     vertical-align:top;
8237   }
8238   .ws0_log { font-size:9pt; font-family:Courier New,monospace; white-space:pre; }
8239 </style>
8240
8241 <details id="BlinderTextClass" class="gsh-src"><summary>class BlinderText</summary>
8242 <span id="BlinderTextScript">
8243 // https://w3c.github.io/uievents/#event-type-keydown
8244 //
8245 // 2020-09-21 class BlinderText - textarea element not to be readable
8246 //
8247 // BlinderText attributes
8248 // bl_plainText - null
8249 // bl_hideChecksum - [false]
8250 // bl_showLength - [false]
8251 // bl_visible - [false]
8252 // data-bl_config - []
8253 // - min. length
8254 // - max. length
8255 // - acceptable charset in generate text
8256 //
8257 function BlinderChecksum(text){
8258   plain = text.bl_plainText;
8259   return strCRC32(plain,plain.length).toFixed(0);
8260 }
8261 function BlinderKeydown(ev){
8262   pass = ev.target
8263   if( ev.code == 'Enter' ){
8264     ev.preventDefault();
8265   }
8266   ev.stopPropagation()
8267 }
8268 function BlinderKeyup1(ev){
8269   blind = ev.target
8270   if( ev.code == 'Backspace'){
8271     blind.bl_plainText = blind.bl_plainText.slice(0,blind.bl_plainText.length-1)
8272   }else
8273   if( and(ev.code == 'KeyV', ev.ctrlKey) ){
8274     blind.bl_visible = !blind.bl_visible;
8275   }else
8276   if( and(ev.code == 'KeyL', ev.ctrlKey) ){
8277     blind.bl_showLength = !blind.bl_showLength;
8278   }else
8279   if( and(ev.code == 'KeyU', ev.ctrlKey) ){
8280     blind.bl_plainText = "";
8281   }else
8282   if( and(ev.code == 'KeyR', ev.ctrlKey) ){
8283     checksum = BlinderChecksum(blind);
8284     blind.bl_plainText = checksum; //toString(32);
8285   }else
8286   if( ev.code == 'Enter' ){
8287     ev.stopPropagation();
8288     ev.preventDefault();
8289     return;
8290   }else
8291   if( ev.key.length != 1 ){
8292     console.log('KeyUp: '+ev.code+'/'+ev.key);
8293     return;
8294   }else{
8295     blind.bl_plainText += ev.key;
8296   }
8297
8298   leng = blind.bl_plainText.length;
8299   //console.log('KeyUp: '+ev.code+'/'+blind.bl_plainText);
8300   checksum = BlinderChecksum(blind) % 10; // show last one digit only
8301
8302   visual = '';
8303   if( !blind.bl_hideChecksum || blind.bl_showLength ){
8304     visual += '[';
8305   }
8306   if( !blind.bl_hideChecksum ){
8307     visual += '#'+checksum.toString(10);
8308   }

```

```

8309     if( blind.bl_showLength ){
8310         visual += '/' + leng;
8311     }
8312     if( !blind.bl_hideChecksum || blind.bl_showLength ){
8313         visual += ')';
8314     }
8315     if( blind.bl_visible ){
8316         visual += blind.bl_plainText;
8317     }else{
8318         visual += '*'.repeat(leng);
8319     }
8320     blind.value = visual;
8321 }
8322 function BlinderKeyup(ev){
8323     BlinderKeyup(ev);
8324     ev.stopPropagation();
8325 }
8326 // https://w3c.github.io/uievents/#keyboardevent
8327 // https://w3c.github.io/uievents/#uievent
8328 // https://dom.spec.whatwg.org/#event
8329 function BlinderTextEvent(){
8330     ev = event;
8331     blind = ev.target;
8332     console.log('Event '+ev.type+'@'+blind.nodeName+'#'+blind.id)
8333     if( ev.type == 'keyup' ){
8334         BlinderKeyup(ev);
8335     }else
8336     if( ev.type == 'keydown' ){
8337         BlinderKeydown(ev);
8338     }else{
8339         console.log('thru-event '+ev.type+'@'+blind.nodeName+'#'+blind.id)
8340     }
8341 }
8342 < textarea hidden id="BlinderTextClassDef" class="textField"
8343 // onkeydown="BlinderTextEvent()" onkeyup="BlinderTextEvent()"
8344 // spellcheck="false" </textarea>
8345 < textarea hidden id="gj_passi"
8346 // class="textField BlinderText"
8347 // placeholder="PassWord1"
8348 // onkeydown="BlinderTextEvent()"
8349 // onkeyup="BlinderTextEvent()"
8350 // spellcheck="false" </textarea>
8351 function SetupBlinderText(parent,txa,phold){
8352     if( txa == null ){
8353         txa = document.createElement('textarea');
8354         //txa.id = id;
8355     }
8356     txa.setAttribute('class','textField BlinderText');
8357     txa.setAttribute('placeholder',phold);
8358     txa.setAttribute('onkeydown','BlinderTextEvent()');
8359     txa.setAttribute('onkeyup','BlinderTextEvent()');
8360     txa.setAttribute('spellcheck','false');
8361     //txa.setAttribute('bl_plainText','false');
8362     txa.bl_plainText = '';
8363     //parent.appendChild(txa);
8364 }
8365 function DestroyBlinderText(txa){
8366     txa.removeAttribute('class');
8367     txa.removeAttribute('placeholder');
8368     txa.removeAttribute('onkeydown');
8369     txa.removeAttribute('onkeyup');
8370     txa.removeAttribute('spellcheck');
8371     txa.bl_plainText = '';
8372 }
8373 //
8374 // visible textarea like Username
8375 //
8376 function VisibleTextEvent(){
8377     if( event.code == 'Enter' ){
8378         if( event.target.NoEnter ){
8379             event.preventDefault();
8380         }
8381     }
8382     event.stopPropagation();
8383 }
8384 function SetupVisibleText(parent,txa,phold){
8385     txa.setAttribute('class','textField VisibleText');
8386     txa.setAttribute('placeholder',phold);
8387     txa.setAttribute('onkeydown','VisibleTextEvent()');
8388     txa.setAttribute('onkeyup','VisibleTextEvent()');
8389     txa.setAttribute('spellcheck','false');
8390     cols = txa.getAttribute('cols');
8391     if( cols != null ){
8392         txa.style.width = '580px';
8393         console.log(txा.id+' cols='+cols)
8394     }else{
8395         console.log(txा.id+' NO cols')
8396     }
8397     rows = txa.getAttribute('rows');
8398     if( rows != null ){
8399         txa.style.height = '40px';
8400         txa.style.resize = 'both';
8401         txा.NoEnter = false;
8402     }else{
8403         txा.NoEnter = true;
8404     }
8405 }
8406 function DestroyVisibleText(txa){
8407     txa.removeAttribute('class');
8408     txa.removeAttribute('placeholder');
8409     txa.removeAttribute('onkeydown');
8410     txa.removeAttribute('onkeyup');
8411     txa.removeAttribute('spellcheck');
8412     cols = txa.removeAttribute('cols');
8413 }
8414 </span>
8415 <script>
8416 js = document.getElementById('BlinderTextScript');
8417 eval(js.innerHTML);
8418 //js.outerHTML = "";
8419 </script>
8420
8421 </details>
8422 <span>
8423 /*
8424 */
8425 /*
8426 <script id="GJLinkScript">
8427 function addlog(e,msg){
8428     e.value += msg;
8429     e.scrollTop = e.scrollHeight;
8430 }
8431 function gjkey_hash(text){
8432     return strCRC32(text,text.length) % 0x10000;

```

```

8433 }
8434 var GJ_Channel = null;
8435 var GJ_Log = null;
8436 function GJ_Join(){
8437     target = gj_join;
8438     if( target.value == 'Leave' ){
8439         GJ_Channel.close();
8440         GJ_Channel = null;
8441         target.value = 'Join';
8442         return;
8443     }
8444
8445     var ws0;
8446     var ws0_log;
8447     ws0 = new WebSocket("ws://localhost:9999/gshws");
8448     GJ_Channel = ws0;
8449     ws0_log = document.getElementById('ws0_log');
8450     GJ_Log = ws0_log;
8451
8452     now = (new Date().getTime() / 1000).toFixed(3);
8453     addlog(ws0_log,['+now+'] +'opened the channel\n');
8454
8455     ws0.addEventListener('open', function(event){
8456         date1 = new Date().getTime();
8457         date2 = (date1 / 1000).toFixed(3);
8458         seed = date1.toString(16);
8459
8460         // user name and key
8461         user = document.getElementById('gj_user').value;
8462         if( user.length == 0 ){
8463             gj_user.value = 'nemo';
8464             user = 'nemo';
8465         }
8466         key1 = document.getElementById('gj_ukey').bl_plainText;
8467         ukey = gjkey_hash(seed+user+key1).toString(16);
8468
8469         // session name and key
8470         chan = document.getElementById('gj_chan').value;
8471         if( chan.length == 0 ){
8472             gj_chan.value = 'main';
8473             chan = 'main';
8474         }
8475         key2 = document.getElementById('gj_ckey').bl_plainText;
8476         ckey = gjkey_hash(seed+chan+key2).toString(16);
8477
8478         msg = date2 + ' JOIN ' + user + '|' + chan + ' ' + ukey + ':' + ckey;
8479         addlog(ws0_log,['+date2+'] send '+msg+'\n');
8480         ws0.send(msg);
8481
8482         target.value = 'Leave';
8483         //console.log(['+date2+'] #' +target.id+ ' +target.value+\n');
8484         //addlog(ws0_log,['+date2+'] label '+target.value+\n');
8485     });
8486     ws0.addEventListener('message', function(event){
8487         now = (new Date().getTime() / 1000).toFixed(3);
8488         msg = event.data;
8489         addlog(ws0_log,['+now+'] recv '+msg+'\n');
8490
8491         argv = msg.split(' ')
8492         tstamp = argv[0];
8493         argv.shift();
8494         if( argv[0] == 'reload' ){
8495             location.reload()
8496         }
8497         argv.shift(); // command
8498         argv.shift(); // from|to
8499         if( argv[0] == 'auth' ){
8500             // doing authorization required
8501         }
8502         if( argv[0] == 'echo' ){
8503             now = (new Date().getTime() / 1000).toFixed(3);
8504             msg = now' '+RESP ' +argv.join(' ');
8505             addlog(ws0_log,['+now+'] '+send '+msg+'\n');
8506             ws0.send(msg);
8507         }
8508         if( argv[0] == 'eval' ){
8509             argv.shift();
8510             js = argv.join(' ');
8511             ret = eval(js);
8512             addlog(ws0_log,'eval '+js+' = '+ret+'\n');
8513             now = (new Date().getTime() / 1000).toFixed(3);
8514             msg = now' '+RESP ' + ret;
8515             ws0.send(msg);
8516             addlog(ws0_log,['+now+'] send '+msg+'\n')
8517         }
8518     });
8519     ws0.addEventListener('close', function(event){
8520         GJ_Channel.close();
8521         GJ_Channel = null;
8522         target.value = 'Join';
8523         addlog(ws0_log,['+date2+'] closed the channel\n');
8524     });
8525 }
8526 function GJ_Send(){
8527     if( GJ_Channel == null ){
8528         //addlog(GJ_Log,['+now+'] send '+msg+'\n');
8529         return;
8530     }
8531     date2 = (new Date().getTime() / 1000).toFixed(3);
8532     target = event.target;
8533     user = document.getElementById('gj_user').value;
8534     chan = document.getElementById('gj_chan').value;
8535     msg = date2 + ISAY '+user+'|'+chan+'|'+gj_sendText.value;
8536     addlog(GJ_Log,['+date2+'] send '+msg+'\n');
8537     GJ_Channel.send(msg);
8538 }
8539 </script>
8540
8541 <!-- ----- GJLINK ----- -->
8542 <!--
8543     - User can subscribe to a channel
8544     - A channel will be broadcasted
8545     - A channel can be a pattern (regular expression)
8546     - User is like From:(me) and channel is like To: or Recipient:
8547     - like VIABUS
8548         - watch message with SENDME, WATCH, CATCH, HEAR, or so
8549         - routing with path expression or name pattern (with routing with DNS like system)
8550 -->
8551 */
8552
8553 //<span id="GJLinkGolang">
8554 // <details id="GshWebSocket" class="gsh-src"><summary>Golang / JavaScript Link</summary>
8555 // 2020-0920 created
8556 // <a href="https://pkg.go.dev/golang.org/x/net/websocket">WS</a>

```

```

8557 // <a href="https://godoc.org/golang.org/x/net/websocket">WS</a>
8558 // INSTALL: go get golang.org/x/net/websocket
8559 // INSTALL: sudo {apt,yum} install git (if git is not installed yet)
8560 // import "golang.org/x/net/websocket"
8561 const gshws_origin = "http://localhost:9999"
8562 const gshws_port = "localhost:9999"
8563 const gshws_path = "gshws"
8564 const gshws_url = "ws://" +gshws_port+ "/" +gshws_path
8565 const GSHWS_MSGSIZE = (8*1024)
8566 func fmtstring(fmts string, params ...interface{})(string){
8567     return fmt.Sprintf(fmts,params...)
8568 }
8569 func GSHWS_MARK(what string)(string){
8570     now := time.Now()
8571     us := fmtstring("%06d",now.Nanosecond() / 1000)
8572     return "[" +now.Format(time.Stamp)+"."+us+"] --WS-- " + what + ": "
8573 }
8574 func gchk(what string,err error){
8575     if( err != nil){
8576         panic(GSHWS_MARK(what)+err.Error())
8577     }
8578 }
8579 func glog(what string, fmts string, params ...interface{}{
8580     fmt.Println(GSHWS_MARK(what))
8581     fmt.Printf(fmts+"\n",params...)
8582 }
8583
8584 var WSV = []*websocket.Conn{}
8585 func jsend(argv []string{
8586     if len(argv) <= 1 {
8587         fmt.Printf("--Ij %v [-m] command arguments\n",argv[0])
8588         return
8589     }
8590     argv = argv[1:]
8591     if( len(WSV) == 0 ){
8592         fmt.Printf("--Ej-- No link now\n")
8593         return
8594     }
8595     if( 1 < len(WSV) ){
8596         fmt.Printf("--Ij-- multiple links (%v)\n",len(WSV))
8597     }
8598
8599     multicast := false // should be filtered with regexp
8600     if( 0 < len(argv) && argv[0] == "-m" ){
8601         multicast = true
8602         argv = argv[1:]
8603     }
8604     args := strings.Join(argv," ")
8605
8606     now := time.Now()
8607     msec := now.UnixNano() / 1000000;
8608     tstamp := fmtstring("%.3f",float64(msec)/1000.0)
8609     msg := fmtstring("%v SEND gshell|* %v",tstamp,args)
8610
8611     if( multicast ){
8612         for i,ws := range WSV {
8613             wn,werr := ws.Write([]byte(msg))
8614             if( werr != nil){
8615                 fmt.Printf("(%v) wn=%v, werr=%v\n",i,wn,werr)
8616             }
8617             glog("SQ",fmtstring("(%v) %v",wn,msg))
8618         }
8619     }else{
8620         i := 0
8621         ws := WSV[i]
8622         wn,werr := ws.Write([]byte(msg))
8623         if( werr != nil ){
8624             fmt.Printf("(%v) wn=%v, werr=%v\n",i,wn,werr)
8625         }
8626         glog("SQ",fmtstring("(%v) %v",wn,msg))
8627     }
8628 }
8629 func serv1(ws *websocket.Conn {
8630     WSV = append(WSV,ws)
8631     fmt.Println("\n")
8632     fmt.Printf("-- accepted connections(%v)\n",len(WSV))
8633     //remoteAddr := ws.RemoteAddr
8634     //fmt.Printf("-- accepted %v\n",remoteAddr)
8635     //fmt.Printf("-- accepted %v\n",ws.Config())
8636     //fmt.Printf("-- accepted %v\n",ws.Config().Header)
8637     //fmt.Printf("-- accepted %v // %v\n",ws,serv1)
8638
8639     var reqb = make([]byte,GSHWS_MSGSIZE)
8640     for {
8641         rn, rerr := ws.Read(reqb)
8642         if( rerr != nil || rn < 0 ){
8643             glog("SQ",fmtstring("(%v) %v",rn,rerr))
8644             break
8645         }
8646         req := string(reqb[0:rn])
8647         glog("SQ",fmtstring("(%v) %v",rn,req))
8648
8649         argv := strings.Split(req," ");
8650         argv = argv[1:]
8651         if( '0' < len(argv) ){
8652             if( argv[0] == "RESP" ){
8653                 // should forward to the destination
8654                 continue;
8655             }
8656         }
8657         now := time.Now()
8658         msec := now.UnixNano() / 1000000;
8659         tstamp := fmtstring("%.3f",float64(msec)/1000.0)
8660         res := fmtstring("%v "+CAST+" "+%v",tstamp,req)
8661         wn, werr := ws.Write([]byte(res))
8662         gchk("SE",werr)
8663         glog("SR",fmtstring("(%v) %v",wn,string(res)))
8664
8665     glog("SF","WS response finish")
8666
8667     wsv := []*websocket.Conn{
8668         for v := range WSV {
8669             if( v != ws ){
8670                 wsv = append(wsv,v)
8671             }
8672         }
8673     WSV = wsv
8674     fmt.Printf("-- closed %v\n",ws)
8675     ws.Close()
8676 }
8677 func gj_server(argv []string {
8678     port := gshws_port
8679     glog("LS",fmtstring("listening at %v",gshws_url))
8680     http.Handle("/"+gshws_path,websocket.Handler(serv1))

```

```

8681     err := http.ListenAndServe(port,nil)
8682     gchk("LE",err)
8683 }
8684
8685 func gj_client(argv []string) {
8686     glog("CS",fmtstring("connecting to %v",gshws_url))
8687     ws, err := websocket.Dial(gshws_url,"",gshws_origin)
8688     gchk("C",err)
8689
8690     var resb = make([]byte, GSHWS_MSGSIZE)
8691     for qi := 0; qi < 3; qi++ {
8692         req := fmtstring("Hello, GShell! (%v)",qi)
8693         wn, werr := ws.Write([]byte(req))
8694         glog("QM",fmtstring("(%v) %v",wn,req))
8695         gchk("QE",werr)
8696         rn, rerr := ws.Read(resb)
8697         gchk("RE",rerr)
8698         glog("RM",fmtstring("(%v) %v",rn,string(resb)))
8699     }
8700     glog("CF", "WS request finish")
8701 }
8702 //</details></span>
8703 /*
8704 <span id="GJLinkView">
8705 <p>
8706 <note class="gsh-note">Execute command "gsh gj listen" on the localhost and push the Join button:</note>
8707 </p>
8708 </span>
8709 <br>
8710 <span id="GJLink_1">
8711 <input id="gj_join" type="button" class="joinButton" onclick="GJ_Join()" value="Join">
8712 <script id="gj_xxxx_gen">
8713 if( true ){
8714     document.write('<+'+textarea id="gj_user" class="textField"><+'/textarea>');
8715     document.write('<+'+textarea id="gj_ukey" class="textField"><+'/textarea>');
8716     document.write('<+'+textarea id="gj_chan" class="textField"><+'/textarea>');
8717     document.write('<+'+textarea id="gj_ckey" class="textField"><+'/textarea>');
8718 }
8719 </script>
8720 <br>
8721 <input id="gj_sendButton" type="button" class="joinButton SendButton" onclick="GJ_Send()" value="Send" data-bodyid="gj_sendText">
8722 <script id="gj_sendText_gen">
8723 if( true ){
8724     document.write('<+'+textarea id="gj_sendText" class="textField" cols=60 rows=3><+'/textarea>');
8725 }
8726 </script>
8727 </span></p>
8728 <p>
8729 <script id="ws0_log_gen">
8730 if( true ){
8731     document.write('<+'+textarea id="ws0_log" class="ws0_log" cols=90 rows=9><+'/textarea>');
8732 }
8733 </script>
8734 </p>
8735 </span>
8736 <script>
8737 function SetupGJLink(){
8738     SetupVisibleText(GJLink_1,gj_user,'UserName');
8739     SetupBlinderText(GJLink_1,gj_ukey,'UserKey');
8740     SetupVisibleText(GJLink_1,gj_chan,'ChannelName');
8741     SetupBlinderText(GJLink_1,gj_ckey,'ChannelKey');
8742     SetupVisibleText(GJLink_1,gj_sendText,'Message');
8743 }
8744 SetupGJLink();
8745 function iselem(eid){
8746     return document.getElementById(eid);
8747 }
8748 function DestroyGJLink1(){
8749     if( iselem('gj_user') ) gj_user.parentNode.removeChild(gj_user);
8750     if( iselem('gj_ukey') ) gj_ukey.parentNode.removeChild(gj_ukey);
8751     if( iselem('gj_chan') ) gj_chan.parentNode.removeChild(gj_chan);
8752     if( iselem('gj_ckey') ) gj_ckey.parentNode.removeChild(gj_ckey);
8753     if( iselem('gj_sendText') ) gj_sendText.parentNode.removeChild(gj_sendText);
8754     if( iselem('ws0_log') ) ws0_log.parentNode.removeChild(ws0_log);
8755 }
8756 DestroyGJLink = DestroyGJLink1;
8757 </script>
8758 */
8759
8760 /*
8761 *//<br></span></html>
8762

```