

```

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 id="GshVersion" hidden="">gsh--0.5.4--2020-09-28--SatoxITS</span>
7  <title>GShell-0.5.4 by SatoxITS</title>
8  <header id="GshBanner" height="100px" onclick="shiftBG();">
9  <div align="right"><note><a href="http://archive.gshell.org">GShell</a> version 0.5.4 // 2020-09-28 // SatoxITS</note></div>
10 </header>
11 <h2>GShell // a General purpose Shell built on the top of Golang</h2>
12 <p>
13 <note>
14 It is a shell for myself, by myself, of myself. --SatoxITS(^-^ )
15 </note>
16 </p>
17 <div id="GJFactory_x"></div>
18
19 <div>
20 <span id="GshMenu">
21 <span class="GshMenu" id="GshMenuEdit" onclick="html_edit();">Edit</span>
22 <span class="GshMenu" id="GshMenuSave" onclick="html_save();">Save</span>
23 <span class="GshMenu" id="GshMenuLoad" onclick="html_load();">Load</span>
24 <span class="GshMenu" id="GshMenuVers" onclick="html_ver();">Vers</span>
25 <span id="gsh-WinId" onclick="win_jump('0.1');">0</span>
26 <span class="GshMenu" id="gsh-menu-exit" onclick="html_close();"></span>
27 <span class="GshMenu" id="gsh-menu-fork" onclick="html_fork();">Fork</span>
28 <span class="GshMenu" id="GshMenuStop" onclick="html_stop(this,true);">Stop</span>
29 <span class="GshMenu" id="GshMenuFold" onclick="html_fold(this);">Unfold</span>
30 <span class="GshMenu" id="gsh-menu-cksum" onclick="html_digest();">Digest</span>
31 <span class="GshMenu" id="GshMenuSign" onclick="html_sign(this);" style="">Source</span>
32 <!-- / <span id="gsh-menu-pure" onclick="html_pure(this);">Pure</span> -->
33 </span>
34 </div>
35 */
36 /*
37 <details id="GshStatement" class="gsh-document"><summary>Statement</summary>
38 <h3>Fun to create a shell</h3>
39 <p>For a programmer, it must be far easy and fun to create his own simple shell
40 rightly fitting to his favor and necessities, than learning existing shells with
41 complex full features that he never use.
42 I, as one of programmers, am writing this tiny shell for my own real needs,
43 totally from scratch, with fun.
44 </p><p>
45 For a programmer, it is fun to learn new computer languages. For long years before
46 writing this software, I had been specialized to C and early HTML2 :-).
47 Now writing this software, I'm learning Go language, HTML5, JavaScript and CSS
48 on demand as a novice of these, with fun.
49 </p><p>
50 This single file "gsh.go", that is executable by Go, contains all of the code written
51 in Go. Also it can be displayed as "gsh.go.html" by browsers. It is a standalone
52 HTML file that works as the viewer of the code of itself, and as the "home page" of
53 this software.
54 </p><p>
55 Because this HTML file is a Go program, you may run it as a real shell program
56 on your computer.
57 But you must be aware that this program is written under situation like above.
58 Needless to say, there is no warranty for this program in any means.
59 </p>
60 <address>Aug 2020, SatoxITS (sato@its-more.jp)</address>
61 </details>
62 */
63 /*
64 <details id="GshFeatures" class="gsh-document"><summary>Features</summary><p>
65 </p>
66 <h3>Cross-browser communication</h3>
67 <p>
68 ... to be written ...
69 </p>
70 <h3>Vi compatible command line editor</h3>
71 <p>
72 The command line of GShell can be edited with commands compatible with
73 <a href="https://www.washington.edu/computing/unix/vi.html"><b>vi</b></a>.
74 As in vi, you can enter <b>command mode</b> by <b>ESC</b> key,
75 then move around in the history by <b>code>j k / ? n N</code>,
76 or within the current line by <b>code>l h f w b 0 $ %</code> or so.
77 </p>
78 </details>
79 */
80 /*
81 <details id="gsh-gindex">
82 <summary>Index</summary><div class="gsh-src">
83 Documents
84 <span class="gsh-link" onclick="jumpto_JavaScriptView();">Command summary</span>
85 Go lang part<span class="gsh-src" onclick="document.getElementById('gsh-gocode').open=true;">
86 Package structures
87 <a href="#import">import</a>
88 <a href="#struct">struct</a>
89 Main functions
90 <a href="#comexpansion">str-expansion</a> // macro processor
91 <a href="#finder">finder</a> // builtin find + du
92 <a href="#grep">grep</a> // builtin grep + wc + cksum + ...
93 <a href="#plugin">plugin</a> // plugin commands
94 <a href="#ex-commands">system</a> // external commands
95 <a href="#builtin">builtin</a> // builtin commands
96 <a href="#network">network</a> // socket handler
97 <a href="#remote-sh">remote-sh</a> // remote shell
98 <a href="#redirect">redirect</a> // StdIn/Out redirection
99 <a href="#history">history</a> // command history
100 <a href="#rusage">rusage</a> // resource usage
101 <a href="#encode">encode</a> // encode / decode
102 <a href="#IME">IME</a> // command line IME
103 <a href="#getline">getline</a> // line editor
104 <a href="#scanf">scanf</a> // string decomposer
105 <a href="#interpreter">interpreter</a> // command interpreter
106 <a href="#main">main</a>
107 </span>
108 JavaScript part
109 <a href="#script-src-view" class="gsh-link" onclick="jumpto_JavaScriptView();">Source</a>
110 <a href="#gsh-data-frame" class="gsh-link" onclick="jumpto_DataView();">BuiltIn data</a>
111 CSS part
112 <a href="#style-src-view" class="gsh-link" onclick="jumpto_StyleView();">Source</a>
113 References
114 <a href="#" class="gsh-link" onclick="jumpto_WholeView();">Internal</a>
115 <a href="#gsh-reference" class="gsh-link" onclick="jumpto_ReferenceView();">External</a>
116 Whole parts
117 <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Source</a>
118 <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Download</a>
119 <a href="#whole-src-view" class="gsh-link" onclick="jumpto_WholeView();">Dump</a>
120
121 </div>
122 </details>
123 */
124 </details id="gsh-gocode">

```

```

125 //<summary>Go Source</summary><div class="gsh-src" onclick="document.getElementById('gsh-gocode').open=false;">
126 // gsh - Go lang based Shell
127 // (c) 2020 ITS more Co., Ltd.
128 // 2020-0807 created by SatoxITS (sato@its-more.jp)
129
130 package main // gsh main
131
132 // <a name="import">Imported packages</a> // <a href="https://golang.org/pkg/">Packages</a>
133 import (
134     "fmt" // <a href="https://golang.org/pkg/fmt/">fmt</a>
135     "strings" // <a href="https://golang.org/pkg/strings/">strings</a>
136     "strconv" // <a href="https://golang.org/pkg/strconv/">strconv</a>
137     "sort" // <a href="https://golang.org/pkg/sort/">sort</a>
138     "time" // <a href="https://golang.org/pkg/time/">time</a>
139     "bufio" // <a href="https://golang.org/pkg/bufio/">bufio</a>
140     "io/ioutil" // <a href="https://golang.org/pkg/io/ioutil/">ioutil</a>
141     "os" // <a href="https://golang.org/pkg/os/">os</a>
142     "syscall" // <a href="https://golang.org/pkg/syscall/">syscall</a>
143     "plugin" // <a href="https://golang.org/pkg/plugin/">plugin</a>
144     "net" // <a href="https://golang.org/pkg/net/">net</a>
145     "net/http" // <a href="https://golang.org/pkg/net/http/">http</a>
146     "html" // <a href="https://golang.org/pkg/html/">html</a>
147     "path/filepath" // <a href="https://golang.org/pkg/path/filepath/">filepath</a>
148     "go/types" // <a href="https://golang.org/pkg/go/types/">types</a>
149     "go/token" // <a href="https://golang.org/pkg/go/token/">token</a>
150     "encoding/base64" // <a href="https://golang.org/pkg/encoding/base64/">base64</a>
151     "unicode/utf8" // <a href="https://golang.org/pkg/unicode/utf8/">utf8</a>
152     //gshdata // gshell's logo and source code
153     "hash/crc32" // <a href="https://golang.org/pkg/unicode/hash/crc32/">crc32</a>
154     "golang.org/x/net/websocket"
155 )
156
157 // // 2020-0906 added,
158 // // <a href="https://golang.org/cmd/cgo/">CGo</a>
159 // #include "poll.h" // <poll.h> // </poll.h> to be closed as HTML tag :-p
160 // typedef struct { struct pollfd fdv[8]; } pollFdv;
161 // int poll(struct pollfd *fdv, int nfd, int timeout){
162 //     return poll(fdv->fdv,nfds,timeout);
163 // }
164 import "C"
165
166 // // 2020-0906 added,
167 func CFPollInl(fp*os.File, timeoutUs int)(ready uintptr){
168     var fdv = C.pollFdv{}
169     var nfds = 1
170     var timeout = timeoutUs/1000
171
172     fdv.fdv[0].fd = C.int(fp.Fd())
173     fdv.fdv[0].events = C.POLLIN
174     if ( 0 < EventRecvFd ) {
175         fdv.fdv[1].fd = C.int(EventRecvFd)
176         fdv.fdv[1].events = C.POLLIN
177         nfds += 1
178     }
179     r := C.pollx(&fdv,C.int(nfds),C.int(timeout))
180     if( r <= 0 ){
181         return 0
182     }
183     if (int(fdv.fdv[1].revents) & int(C.POLLIN)) != 0 {
184         //fprintf(stderr,"--De-- got Event\n");
185         return uintptr(EventFdOffset + fdv.fdv[1].fd)
186     }
187     if (int(fdv.fdv[0].revents) & int(C.POLLIN)) != 0 {
188         return uintptr(NormalFdOffset + fdv.fdv[0].fd)
189     }
190     return 0
191 }
192
193 const (
194     NAME = "gsh"
195     VERSION = "0.5.4"
196     DATE = "2020-09-28"
197     AUTHOR = "SatoxITS(^-^)"
198 )
199 var (
200     GSH_HOME = ".gsh" // under home directory
201     GSH_PORT = 9999
202     MaxStreamSize = int64(128*1024*1024*1024) // 128GiB is too large?
203     PROMPT = "> "
204     LINESIZE = (8*1024)
205     PATHSEP = ":" // should be ";" in Windows
206     DIRSEP = "/" // canbe \ in Windows
207 )
208
209 // -xX logging control
210 // --A-- all
211 // --I-- info.
212 // --D-- debug
213 // --T-- time and resource usage
214 // --W-- warning
215 // --E-- error
216 // --F-- fatal error
217 // --Xn- network
218
219 // <a name="struct">Structures</a>
220 type GCommandHistory struct {
221     StartAt time.Time // command line execution started at
222     EndAt time.Time // command line execution ended at
223     ResCode int // exit code of (external command)
224     CmdError error // error string
225     OutData *os.File // output of the command
226     FoundFile []string // output - result of ufind
227     Rusagev [2]syscall.Rusage // Resource consumption, CPU time or so
228     CmdId int // maybe with identified with arguments or impact
229     // redirection commands should not be the CmdId
230     WorkDir string // working directory at start
231     WorkDirX int // index in ChdirHistory
232     CmdLine string // command line
233 }
234 type GChdirHistory struct {
235     Dir string
236     MovedAt time.Time
237     CmdIndex int
238 }
239 type CmdMode struct {
240     Background bool
241 }
242 type Event struct {
243     when time.Time
244     event int
245     evarg int64
246     CmdIndex int
247 }
248 var CmdIndex int

```

```

249 var Events []Event
250 type PluginInfo struct {
251     Spec      *Plugin.Plugin
252     Addr      Plugin.Symbol
253     Name      string // maybe relative
254     Path      string // this is in Plugin but hidden
255 }
256 type GServer struct {
257     host      string
258     port      string
259 }
260
261 // <a href="https://tools.ietf.org/html/rfc3230">Digest</a>
262 const ( // SumType
263     SUM_ITEMS   = 0x000001 // items count
264     SUM_SIZE    = 0x000002 // data length (simply added)
265     SUM_SIZEHASH = 0x000004 // data length (hashed sequence)
266     SUM_DATEHASH = 0x000008 // date of data (hashed sequence)
267     // also envelope attributes like time stamp can be a part of digest
268     // hashed value of sizes or mod-date of files will be useful to detect changes
269
270     SUM_WORDS   = 0x000010 // word count is a kind of digest
271     SUM_LINES   = 0x000020 // line count is a kind of digest
272     SUM_SUM64   = 0x000040 // simple add of bytes, useful for human too
273
274     SUM_SUM32_BITS = 0x000100 // the number of true bits
275     SUM_SUM32_2BYTE = 0x000200 // 16bits words
276     SUM_SUM32_4BYTE = 0x000400 // 32bits words
277     SUM_SUM32_8BYTE = 0x000800 // 64bits words
278
279     SUM_SUM16_BSD = 0x001000 // UNIXsum -sum -bsd
280     SUM_SUM16_SYSV = 0x002000 // UNIXsum -sum -sysv
281     SUM_UNIXFILE  = 0x004000
282     SUM_CRCIEEE   = 0x008000
283 )
284 type CheckSum struct {
285     Files      int64 // the number of files (or data)
286     Size       int64 // content size
287     Words      int64 // word count
288     Lines      int64 // line count
289     SumType    int
290     Sum64      uint64
291     Crc32Table crc32.Table
292     Crc32Val   uint32
293     Sum16      int
294     Ctime      time.Time
295     Atime      time.Time
296     Mtime      time.Time
297     Start      time.Time
298     Done       time.Time
299     RusgAtStart [2]syscall.Rusage
300     RusgAtEnd   [2]syscall.Rusage
301 }
302 type ValueStack [][]string
303 type GshContext struct {
304     StartDir      string // the current directory at the start
305     GetLine       string // gsh-getline command as a input line editor
306     ChdirHistory  []GChdirHistory // the 1st entry is wd at the start
307     gshPA         syscall.ProcAttr
308     CommandHistory []GCommandHistory
309     CmdCurrent    GCommandHistory
310     Background    bool
311     BackgroundJobs []int
312     LastRusage    syscall.Rusage
313     GshHomeDir    string
314     TerminalId    int
315     CmdTrace      bool // should be [map]
316     CmdTime       bool // should be [map]
317     PluginFuncs  []PluginInfo
318     iValues       []string
319     iDelimiter    string // field separator of print out
320     iFormat       string // default print format (of integer)
321     iValStack     ValueStack
322     LastServer    GServer
323     RSERV         string // [gsh://]host[:port]
324     RWD           string // remote (target, there) working directory
325     lastCheckSum CheckSum
326 }
327
328 func nsleep(ns time.Duration){
329     time.Sleep(ns)
330 }
331 func usleep(ns time.Duration){
332     nsleep(ns*1000)
333 }
334 func msleep(ns time.Duration){
335     nsleep(ns*1000000)
336 }
337 func sleep(ns time.Duration){
338     nsleep(ns*1000000000)
339 }
340
341 func strBegins(str, pat string)(bool){
342     if len(pat) <= len(str){
343         yes := str[0:len(pat)] == pat
344         //fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat, yes)
345         return yes
346     }
347     //fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat,false)
348     return false
349 }
350 func isin(what string, list []string) bool {
351     for _, v := range list {
352         if v == what {
353             return true
354         }
355     }
356     return false
357 }
358 func isinX(what string,list[]string)(int){
359     for i,v := range list {
360         if v == what {
361             return i
362         }
363     }
364     return -1
365 }
366
367 func env(opts []string) {
368     env := os.Environ()
369     if isin("-s", opts){
370         sort.Slice(env, func(i,j int) bool {
371             return env[i] < env[j]
372         })
373     }
374 }

```

```

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

```

```

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

```

```

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

```

```

745         break
746     }
747 }
748 crc = crc32.Update(crc,table,slen)
749 crc ^= 0xFFFFFFFF
750 return crc
751 }
752
753 func (gsh*GshContext)xCKsum(path string,argv[]string, sum*Checksum)(int64){
754     if isin("-type/f",argv) && !IsRegFile(path){
755         return 0
756     }
757     if isin("-type/d",argv) && IsRegFile(path){
758         return 0
759     }
760     file, err := os.OpenFile(path,os.O_RDONLY,0)
761     if err != nil {
762         fmt.Printf("--E-- cksum %v (%v)\n",path,err)
763         return -1
764     }
765     defer file.Close()
766     if gsh.CmdTrace { fmt.Printf("--I-- cksum %v %v\n",path,argv) }
767
768     bi := 0
769     var buff = make([]byte,32*1024)
770     var total int64 = 0
771     var initTime = time.Time{}
772     if sum.Start == initTime {
773         sum.Start = time.Now()
774     }
775     for bi = 0; ; bi++ {
776         count,err := file.Read(buff)
777         if count <= 0 || err != nil {
778             break
779         }
780         if (sum.SumType & SUM_SUM64) != 0 {
781             s := sum.Sum64
782             for _,c := range buff[0:count] {
783                 s += uint64(c)
784             }
785             sum.Sum64 = s
786         }
787         if (sum.SumType & SUM_UNIXFILE) != 0 {
788             sum.Crc32Val = byteCRC32add(sum.Crc32Val,buff,uint64(count))
789         }
790         if (sum.SumType & SUM_CRCIEEE) != 0 {
791             sum.Crc32Val = crc32.Update(sum.Crc32Val,&sum.Crc32Table,buff[0:count])
792         }
793         // <a href="https://en.wikipedia.org/wiki/BSD_checksum">BSD checksum</a>
794         if (sum.SumType & SUM_SUM16_BSD) != 0 {
795             s := sum.Sum16
796             for _,c := range buff[0:count] {
797                 s = (s >> 1) + ((s & 1) << 15)
798                 s += int(c)
799                 s ^= 0xFFFF
800                 //fmt.Printf("BSDsum: %d[%d] %d\n",sum.Size+int64(i),i,s)
801             }
802             sum.Sum16 = s
803         }
804         if (sum.SumType & SUM_SUM16_SYSV) != 0 {
805             for bj := 0; bj < count; bj++ {
806                 sum.Sum16 += int(buff[bj])
807             }
808         }
809         total += int64(count)
810     }
811     sum.Done = time.Now()
812     sum.Files += 1
813     sum.Size += total
814     if !isin("-s",argv) {
815         fmt.Printf("%v ",total)
816     }
817     return 0
818 }
819
820 // <a name="grep">grep</a>
821 // "lines", "lin" or "lnp" for "(text) line processor" or "scanner"
822 // a*,lab,c, ... sequential combination of patterns
823 // what "LINE" is should be definable
824 // generic line-by-line processing
825 // grep [-v]
826 // cat -n -v
827 // uniq [-c]
828 // tail -f
829 // sed s/x/y/ or awk
830 // grep with line count like wc
831 // rewrite contents if specified
832 func (gsh*GshContext)xGrep(path string,regexp[]string)(int){
833     file, err := os.OpenFile(path,os.O_RDONLY,0)
834     if err != nil {
835         fmt.Printf("--E-- grep %v (%v)\n",path,err)
836         return -1
837     }
838     defer file.Close()
839     if gsh.CmdTrace { fmt.Printf("--I-- grep %v %v\n",path,regexp) }
840     //reader := bufio.NewReaderSize(file,LINESIZE)
841     reader := bufio.NewReaderSize(file,80)
842     li := 0
843     found := 0
844     for li = 0; ; li++ {
845         line, err := reader.ReadString('\n')
846         if len(line) <= 0 {
847             break
848         }
849         if 150 < len(line) {
850             // maybe binary
851             break;
852         }
853         if err != nil {
854             break
855         }
856         if 0 <= strings.Index(string(line),regexp[0]) {
857             found += 1
858             fmt.Printf("%s:%d: %s",path,li,line)
859         }
860     }
861     //fmt.Printf("total %d lines %s\n",li,path)
862     //if( 0 < found ){ fmt.Printf("(found %d lines %s)\n",found,path); }
863     return found
864 }
865
866 // <a name="finder">Finder</a>
867 // finding files with it name and contents
868 // file names are Ored

```

```

869 // show the content with %x fmt list
870 // ls -R
871 // tar command by adding output
872 type fileSum struct {
873     Err int64 // access error or so
874     Size int64 // content size
875     DupSize int64 // content size from hard links
876     Blocks int64 // number of blocks (of 512 bytes)
877     DupBlocks int64 // Blocks pointed from hard links
878     HLinks int64 // hard links
879     Words int64
880     Lines int64
881     Files int64
882     Dirs int64 // the num. of directories
883     SymLink int64
884     Flats int64 // the num. of flat files
885     MaxDepth int64
886     MaxNamlen int64 // max. name length
887     nextRepo time.Time
888 }
889 func showFusage(dir string, fusage *fileSum){
890     bsume := float64(((fusage.Blocks-fusage.DupBlocks)/2)*1024)/1000000.0
891     //bsumdup := float64((fusage.Blocks/2)*1024)/1000000.0
892
893     fmt.Printf("%v: %v files (%vd %vs %vh) %.6f MB (%.2f MBK)\n",
894         dir,
895         fusage.Files,
896         fusage.Dirs,
897         fusage.SymLink,
898         fusage.HLinks,
899         float64(fusage.Size)/1000000.0, bsume);
900 }
901 const (
902     S_IFMT = 0170000
903     S_IFCHR = 0020000
904     S_IFDIR = 0040000
905     S_IFREG = 0100000
906     S_IFLNK = 0120000
907     S_IFSOCK = 0140000
908 )
909 func cumPinfo(fsum *fileSum, path string, staterr error, fstat syscall.Stat_t, argv []string, verb bool)(*fileSum){
910     now := time.Now()
911     if time.Second <= now.Sub(fsum.nextRepo) {
912         if !fsum.nextRepo.IsZero(){
913             tstamp := now.Format(time.Stamp)
914             showFusage(tstamp, fsum)
915         }
916         fsum.nextRepo = now.Add(time.Second)
917     }
918     if staterr != nil {
919         fsum.Err += 1
920         return fsum
921     }
922     fsum.Files += 1
923     if 1 < fstat.Nlink {
924         // must count only once...
925         // at least ignore ones in the same directory
926         //if finfo.Mode().IsRegular() {
927         if (fstat.Mode & S_IFMT) == S_IFREG {
928             fsum.HLinks += 1
929             fsum.DupBlocks += int64(fstat.Blocks)
930             //fmt.Printf("---Dup HardLink %v %s\n", fstat.Nlink, path)
931         }
932     }
933     //fsum.Size += finfo.Size()
934     fsum.Size += fstat.Size
935     fsum.Blocks += int64(fstat.Blocks)
936     //if verb { fmt.Printf("%8dBk %s", fstat.Blocks/2, path) }
937     if isin("-ls", argv){
938         //if verb { fmt.Printf("%4d %8d ", fstat.Blksize, fstat.Blocks) }
939     //     fmt.Printf("%d\t", fstat.Blocks/2)
940     }
941     //if finfo.IsDir()
942     if (fstat.Mode & S_IFMT) == S_IFDIR {
943         fsum.Dirs += 1
944     }
945     //if (finfo.Mode() & os.ModeSymlink) != 0
946     if (fstat.Mode & S_IFMT) == S_IFLNK {
947         //if verb { fmt.Printf("symlink(%v,%s)\n", fstat.Mode, finfo.Name()) }
948         //if verb { fmt.Printf("symlink(%o,%s)\n", fstat.Mode, finfo.Name()) }
949         fsum.SymLink += 1
950     }
951     return fsum
952 }
953 func (gsh*GshContext)xxFindEntv(depth int, total *fileSum, dir string, dstat syscall.Stat_t, ei int, entv []string, npatv []string, argv []string)(*fileSum){
954     nols := isin("-grep", argv)
955     // sort entv
956     /*
957     if isin("-t", argv){
958         sort.Slice(filev, func(i, j int) bool {
959             return 0 < filev[i].ModTime().Sub(filev[j].ModTime())
960         })
961     }
962     */
963     /*
964     if isin("-u", argv){
965         sort.Slice(filev, func(i, j int) bool {
966             return 0 < filev[i].AccTime().Sub(filev[j].AccTime())
967         })
968     }
969     if isin("-U", argv){
970         sort.Slice(filev, func(i, j int) bool {
971             return 0 < filev[i].CreateTime().Sub(filev[j].CreateTime())
972         })
973     }
974     */
975     /*
976     if isin("-s", argv){
977         sort.Slice(filev, func(i, j int) bool {
978             return filev[j].Size() < filev[i].Size()
979         })
980     }
981     */
982     for _, filename := range entv {
983         for _, npat := range npatv {
984             match := true
985             if npat == "*" {
986                 match = true
987             }else{
988                 match, _ = filepath.Match(npat, filename)
989             }
990             path := dir + DIRSEP + filename
991             if !match {
992                 continue

```



```

993     }
994     var fstat syscall.Stat_t
995     staterr := syscall.Lstat(path, &fstat)
996     if staterr != nil {
997         if !isin("-w", argv) {fmt.Printf("ufind: %v\n", staterr) }
998         continue;
999     }
1000     if isin("-du", argv) && (fstat.Mode & S_IFMT) == S_IFDIR {
1001         // should not show size of directory in "-du" mode ...
1002     }else
1003     if !nols && !isin("-s", argv) && (!isin("-du", argv) || isin("-a", argv)) {
1004         if isin("-du", argv) {
1005             fmt.Printf("%d\t", fstat.Blocks/2)
1006         }
1007         showFileInfo(path, argv)
1008     }
1009     if true { // && isin("-du", argv)
1010         total = cumFinfo(total, path, staterr, fstat, argv, false)
1011     }
1012     /*
1013     if isin("-wc", argv) {
1014     }
1015     */
1016     if gsh.lastCheckSum.SumType != 0 {
1017         gsh.xCksum(path, argv, &gsh.lastCheckSum);
1018     }
1019     x := isinX("-grep", argv); // -grep will be convenient like -ls
1020     if 0 <= x && x+1 <= len(argv) { // -grep will be convenient like -ls
1021         if IsRegFile(path){
1022             found := gsh.xGrep(path, argv[x+1:])
1023             if 0 < found {
1024                 foundv := gsh.CmdCurrent.FoundFile
1025                 if len(foundv) < 10 {
1026                     gsh.CmdCurrent.FoundFile =
1027                         append(gsh.CmdCurrent.FoundFile, path)
1028                 }
1029             }
1030         }
1031     }
1032     if !isin("-r0", argv) { // -d 0 in du, -depth n in find
1033         //total.Depth += 1
1034         if (fstat.Mode & S_IFMT) == S_IFLNK {
1035             continue
1036         }
1037         if dstat.Rdev != fstat.Rdev {
1038             fmt.Printf("--I-- don't follow differnet device %v(%v) %v(%v)\n",
1039                 dir, dstat.Rdev, path, fstat.Rdev)
1040         }
1041         if (fstat.Mode & S_IFMT) == S_IFDIR {
1042             total = gsh.xxFind(depth+1, total, path, npatv, argv)
1043         }
1044     }
1045     }
1046     }
1047     return total
1048 }
1049 func (gsh*GshContext)xxFind(depth int, total *fileSum, dir string, npatv[]string, argv[]string)(*fileSum){
1050     nols := isin("-grep", argv)
1051     dirfile, oerr := os.OpenFile(dir, os.O_RDONLY, 0)
1052     if oerr == nil {
1053         //fmt.Printf("--I-- %v(%v)[%d]\n", dir, dirfile, dirfile.Fd())
1054         defer dirfile.Close()
1055     }else{
1056     }
1057
1058     prev := *total
1059     var dstat syscall.Stat_t
1060     staterr := syscall.Lstat(dir, &dstat) // should be flstat
1061
1062     if staterr != nil {
1063         if !isin("-w", argv){ fmt.Printf("ufind: %v\n", staterr) }
1064         return total
1065     }
1066     //filev, err := ioutil.ReadDir(dir)
1067     //_, err := ioutil.ReadDir(dir) // ReadDir() heavy and bad for huge directory
1068     /*
1069     if err != nil {
1070         if !isin("-w", argv){ fmt.Printf("ufind: %v\n", err) }
1071         return total
1072     }
1073     */
1074     if depth == 0 {
1075         total = cumFinfo(total, dir, staterr, dstat, argv, true)
1076         if !nols && !isin("-s", argv) && (!isin("-du", argv) || isin("-a", argv)) {
1077             showFileInfo(dir, argv)
1078         }
1079     }
1080     // it it is not a directory, just scan it and finish
1081
1082     for ei := 0; ; ei++ {
1083         entv, rderr := dirfile.Readdirnames(8*1024)
1084         if len(entv) == 0 || rderr != nil {
1085             //if rderr != nil { fmt.Printf("[%d] len=%d (%v)\n", ei, len(entv), rderr) }
1086             break
1087         }
1088         if 0 < ei {
1089             fmt.Printf("--I-- xxFind[%d] %d large-dir: %s\n", ei, len(entv), dir)
1090         }
1091         total = gsh.xxFindEntv(depth, total, dir, dstat, ei, entv, npatv, argv)
1092     }
1093     if isin("-du", argv) {
1094         // if in "du" mode
1095         fmt.Printf("%d\t%s\n", (total.Blocks-prev.Blocks)/2, dir)
1096     }
1097     return total
1098 }
1099
1100 // {ufind|fu|ls} [Files] [// Names] [-- Expressions]
1101 // Files is "." by default
1102 // Names is "*" by default
1103 // Expressions is "-print" by default for "ufind", or -du for "fu" command
1104 func (gsh*GshContext)xFind(argv[]string){
1105     if 0 < len(argv) && strBegins(argv[0], "?"){
1106         showFound(gsh, argv)
1107         return
1108     }
1109     if isin("-cksum", argv) || isin("-sum", argv) {
1110         gsh.lastCheckSum = CheckSum{}
1111         if isin("-sum", argv) && isin("-add", argv) {
1112             gsh.lastCheckSum.SumType |= SUM_SUM64
1113         }else
1114         if isin("-sum", argv) && isin("-size", argv) {
1115             gsh.lastCheckSum.SumType |= SUM_SIZE
1116         }else

```

```

1117     if isin("-sum",argv) && isin("-bsd",argv) {
1118         gsh.lastCheckSum.SumType |= SUM_SUM16_BSD
1119     }else
1120     if isin("-sum",argv) && isin("-sysv",argv) {
1121         gsh.lastCheckSum.SumType |= SUM_SUM16_SYSV
1122     }else
1123     if isin("-sum",argv) {
1124         gsh.lastCheckSum.SumType |= SUM_SUM64
1125     }
1126     if isin("-unix",argv) {
1127         gsh.lastCheckSum.SumType |= SUM_UNIXFILE
1128         gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32UNIX)
1129     }
1130     if isin("-ieee",argv){
1131         gsh.lastCheckSum.SumType |= SUM_CRCIEEE
1132         gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32IEEE)
1133     }
1134     gsh.lastCheckSum.RusgAtStart = Getrusagev()
1135 }
1136 var total = fileSum()
1137 npats := []string{}
1138 for _,v := range argv {
1139     if 0 < len(v) && v[0] != '-' {
1140         npats = append(npats,v)
1141     }
1142     if v == "/" { break }
1143     if v == "--" { break }
1144     if v == "-grep" { break }
1145     if v == "-ls" { break }
1146 }
1147 if len(npats) == 0 {
1148     npats = []string{"*"}
1149 }
1150 cwd := "."
1151 // if to be fullpath ::: cwd, _ := os.Getwd()
1152 if len(npats) == 0 { npats = []string{"*"} }
1153 fusage := gsh.xxFind(0,total,cwd,npats,argv)
1154 if gsh.lastCheckSum.SumType != 0 {
1155     var sumi uint64 = 0
1156     sum := &gsh.lastCheckSum
1157     if (sum.SumType & SUM_SIZE) != 0 {
1158         sumi = uint64(sum.Size)
1159     }
1160     if (sum.SumType & SUM_SUM64) != 0 {
1161         sumi = sum.Sum64
1162     }
1163     if (sum.SumType & SUM_SUM16_SYSV) != 0 {
1164         s := uint32(sum.Sum16)
1165         r := (s & 0xFFFF) + ((s & 0xFFFFFFFF) >> 16)
1166         s = (r & 0xFFFF) + (r >> 16)
1167         sum.Crc32Val = uint32(s)
1168         sumi = uint64(s)
1169     }
1170     if (sum.SumType & SUM_SUM16_BSD) != 0 {
1171         sum.Crc32Val = uint32(sum.Sum16)
1172         sumi = uint64(sum.Sum16)
1173     }
1174     if (sum.SumType & SUM_UNIXFILE) != 0 {
1175         sum.Crc32Val = byteCRC32end(sum.Crc32Val,uint64(sum.Size))
1176         sumi = uint64(byteCRC32end(sum.Crc32Val,uint64(sum.Size)))
1177     }
1178     if 1 < sum.Files {
1179         fmt.Printf("%v %v // %v / %v files, %v/file\r\n",
1180             sumi,sum.Size,
1181             abssize(sum.Size),sum.Files,
1182             abssize(sum.Size/sum.Files))
1183     }else{
1184         fmt.Printf("%v %v %v\n",
1185             sumi,sum.Size,npats[0])
1186     }
1187 }
1188 if !isin("-grep",argv) {
1189     showFusage("total",fusage)
1190 }
1191 if !isin("-s",argv){
1192     hits := len(gsh.CmdCurrent.FoundFile)
1193     if 0 < hits {
1194         fmt.Printf("--I-- %d files hits // can be refered with !&df\n",
1195             hits,len(gsh.CommandHistory))
1196     }
1197 }
1198 if gsh.lastCheckSum.SumType != 0 {
1199     if isin("-ru",argv) {
1200         sum := &gsh.lastCheckSum
1201         sum.Done = time.Now()
1202         gsh.lastCheckSum.RusgAtEnd = Getrusagev()
1203         elps := sum.Done.Sub(sum.Start)
1204         fmt.Printf("--cksum-size: %v (%v) / %v files, %v/file\r\n",
1205             sum.Size,abssize(sum.Size),sum.Files,abssize(sum.Size/sum.Files))
1206         nanos := int64(elps)
1207         fmt.Printf("--cksum-time: %v/total, %v/file, %.1f files/s, %v\r\n",
1208             abftime(nanos),
1209             abftime(nanos/sum.Files),
1210             (float64(sum.Files)*1000000000.0)/float64(nanos),
1211             abspeed(sum.Size,nanos))
1212         diff := RusageSubv(sum.RusgAtEnd,sum.RusgAtStart)
1213         fmt.Printf("--cksum-rusg: %v\n",sRusagef("",argv,diff))
1214     }
1215 }
1216 return
1217 }
1218
1219 func showFiles(files[]string){
1220     sp := ""
1221     for i,file := range files {
1222         if 0 < i { sp = " " } else { sp = "" }
1223         fmt.Printf(sp+"%s",escapeWhiteSP(file))
1224     }
1225 }
1226 func showFound(gshCtx *GshContext, argv[]string){
1227     for i,v := range gshCtx.CommandHistory {
1228         if 0 < len(v.FoundFile) {
1229             fmt.Printf("!%d (%d) ",i,len(v.FoundFile))
1230             if isin("-ls",argv){
1231                 fmt.Printf("\n")
1232                 for _,file := range v.FoundFile {
1233                     fmt.Printf(" ") //sub number?
1234                     showFileInfo(file,argv)
1235                 }
1236             }else{
1237                 showFiles(v.FoundFile)
1238                 fmt.Printf("\n")
1239             }
1240         }
1241     }
1242 }

```

```

1241     }
1242 }
1243
1244 func showMatchFile(filev []os.FileInfo, npat,dir string, argv[]string)(string,bool){
1245     fname := ""
1246     found := false
1247     for _,v := range filev {
1248         match, _ := filepath.Match(npat,(v.Name()))
1249         if match {
1250             fname = v.Name()
1251             found = true
1252             //fmt.Printf("[%d] %s\n",i,v.Name())
1253             showIfExecutable(fname,dir,argv)
1254         }
1255     }
1256     return fname,found
1257 }
1258 func showIfExecutable(name,dir string,argv[]string)(ffullpath string,ffound bool){
1259     var fullpath string
1260     if strBegins(name,DIRSEP){
1261         fullpath = name
1262     }else{
1263         fullpath = dir + DIRSEP + name
1264     }
1265     fi, err := os.Stat(fullpath)
1266     if err != nil {
1267         fullpath = dir + DIRSEP + name + ".go"
1268         fi, err = os.Stat(fullpath)
1269     }
1270     if err == nil {
1271         fm := fi.Mode()
1272         if fm.IsRegular() {
1273             // R_OK=4, W_OK=2, X_OK=1, F_OK=0
1274             if syscall.Access(fullpath,5) == nil {
1275                 ffullpath = fullpath
1276                 ffound = true
1277                 if ! isin("-s", argv) {
1278                     showFileInfo(fullpath,argv)
1279                 }
1280             }
1281         }
1282     }
1283     return ffullpath, ffound
1284 }
1285 func which(list string, argv []string) (fullpathv []string, itis bool){
1286     if len(argv) <= 1 {
1287         fmt.Printf("Usage: which comand [-s] [-a] [-ls]\n")
1288         return []string(""), false
1289     }
1290     path := argv[1]
1291     if strBegins(path,"/") {
1292         // should check if executable?
1293         _,exOK := showIfExecutable(path,"/",argv)
1294         fmt.Printf("--D-- %v exOK=%v\n",path,exOK)
1295         return []string(path),exOK
1296     }
1297     pathenv, efound := os.LookupEnv(list)
1298     if ! efound {
1299         fmt.Printf("--E-- which: no \"%s\" environment\n",list)
1300         return []string(""), false
1301     }
1302     showall := isin("-a",argv) || 0 <= strings.Index(path,"*")
1303     dirv := strings.Split(pathenv,PATHSEP)
1304     ffound := false
1305     ffullpath := path
1306     for _, dir := range dirv {
1307         if 0 <= strings.Index(path,"*") { // by wild-card
1308             list,_ := ioutil.ReadDir(dir)
1309             ffullpath, ffound = showMatchFile(list,path,dir,argv)
1310         }else{
1311             ffullpath, ffound = showIfExecutable(path,dir,argv)
1312         }
1313         //if ffound && !isin("-a", argv) {
1314         if ffound && !showall {
1315             break;
1316         }
1317     }
1318     return []string(ffullpath), ffound
1319 }
1320
1321 func stripLeadingWSParg(argv[]string)([]string){
1322     for ; 0 < len(argv); {
1323         if len(argv[0]) == 0 {
1324             argv = argv[1:]
1325         }else{
1326             break
1327         }
1328     }
1329     return argv
1330 }
1331 func xEval(argv []string, nlend bool){
1332     argv = stripLeadingWSParg(argv)
1333     if len(argv) == 0 {
1334         fmt.Printf("eval [%%format] [Go-expression]\n")
1335         return
1336     }
1337     pfmt := "%v"
1338     if argv[0][0] == '%' {
1339         pfmt = argv[0]
1340         argv = argv[1:]
1341     }
1342     if len(argv) == 0 {
1343         return
1344     }
1345     gocode := strings.Join(argv, " ");
1346     //fmt.Printf("eval [%v] [%v]\n",pfmt,gocode)
1347     fset := token.NewFileSet()
1348     rval, _ := types.Eval(fset,nil,token.NoPos,gocode)
1349     fmt.Printf(pfmt,rval.Value)
1350     if nlend { fmt.Printf("\n") }
1351 }
1352
1353 func getval(name string) (found bool, val int) {
1354     /* should expand the name here */
1355     if name == "gsh.pid" {
1356         return true, os.Getpid()
1357     }else
1358     if name == "gsh.ppid" {
1359         return true, os.Getppid()
1360     }
1361     return false, 0
1362 }
1363
1364 func echo(argv []string, nlend bool){

```

```

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

```

```

1489     fd = pout.Fd()
1490     gshPA.Files[fdix] = pout.Fd()
1491 }else{
1492     fd = pin.Fd()
1493     gshPA.Files[fdix] = pin.Fd()
1494 }
1495 // should do this by Goroutine?
1496 if false {
1497     fmt.Printf("--Ip- Opened fd[%v] %s %v\n",fd,dir,name)
1498     fmt.Printf("--RED1 [%d,%d,%d]->[%d,%d,%d]\n",
1499         os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd(),
1500         pin.Fd(),pout.Fd(),pout.Fd())
1501 }
1502     savi := os.Stdin
1503     savo := os.Stdout
1504     save := os.Stderr
1505     os.Stdin = pin
1506     os.Stdout = pout
1507     os.Stderr = pout
1508     gsh.BackGround = true
1509     gsh.gshellh(name)
1510     gsh.BackGround = false
1511     os.Stdin = savi
1512     os.Stdout = savo
1513     os.Stderr = save
1514
1515     gshPA.Files[fdix] = savfd
1516     return pin,pout,false
1517 }
1518
1519 // <a name="ex-commands">External commands</a>
1520 func (gsh*GshContext)excommand(exec bool, argv []string) (notf bool,exit bool) {
1521     if gsh.CmdTrace { fmt.Printf("--I-- excommand[%v](%v)\n",exec,argv) }
1522
1523     gshPA := gsh.gshPA
1524     fullpathv, itis := which("PATH",[]string{"which",argv[0],"-s"})
1525     if itis == false {
1526         return true,false
1527     }
1528     fullpath := fullpathv[0]
1529     argv = unescapeWhiteSPV(argv)
1530     if 0 < strings.Index(fullpath,".go") {
1531         nargv := argv // []string{}
1532         gofullpathv, itis := which("PATH",[]string{"which","go","-s"})
1533         if itis == false {
1534             fmt.Printf("--F-- Go not found\n")
1535             return false,true
1536         }
1537         gofullpath := gofullpathv[0]
1538         nargv = []string{ gofullpath, "run", fullpath }
1539         fmt.Printf("--I-- %s {%s %s %s}\n",gofullpath,
1540             nargv[0],nargv[1],nargv[2])
1541         if exec {
1542             syscall.Exec(gofullpath,nargv,os.Environ())
1543         }else{
1544             pid, _ := syscall.ForkExec(gofullpath,nargv,&gshPA)
1545             if gsh.BackGround {
1546                 fmt.Fprintf(stderr,"--Ip- in Background pid[%d]%(%v)\n",pid,len(argv),nargv)
1547                 gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
1548             }else{
1549                 rusage := syscall.Rusage {}
1550                 syscall.Wait4(pid,nil,0,&rusage)
1551                 gsh.LastRusage = rusage
1552                 gsh.CmdCurrent.Rusagev[1] = rusage
1553             }
1554         }
1555     }else{
1556         if exec {
1557             syscall.Exec(fullpath,argv,os.Environ())
1558         }else{
1559             pid, _ := syscall.ForkExec(fullpath,argv,&gshPA)
1560             //fmt.Printf("[%d]\n",pid); // "&" to be background
1561             if gsh.BackGround {
1562                 fmt.Fprintf(stderr,"--Ip- in Background pid[%d]%(%v)\n",pid,len(argv),argv)
1563                 gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
1564             }else{
1565                 rusage := syscall.Rusage {}
1566                 syscall.Wait4(pid,nil,0,&rusage);
1567                 gsh.LastRusage = rusage
1568                 gsh.CmdCurrent.Rusagev[1] = rusage
1569             }
1570         }
1571     }
1572     return false,false
1573 }
1574
1575 // <a name="builtin">Builtin Commands</a>
1576 func (gshCtx *GshContext) sleep(argv []string) {
1577     if len(argv) < 2 {
1578         fmt.Printf("Sleep 100ms, 100us, 100ns, ... \n")
1579         return
1580     }
1581     duration := argv[1];
1582     d, err := time.ParseDuration(duration)
1583     if err != nil {
1584         d, err = time.ParseDuration(duration+"s")
1585         if err != nil {
1586             fmt.Printf("duration ? %s (%s)\n",duration,err)
1587             return
1588         }
1589     }
1590     //fmt.Printf("Sleep %v\n",duration)
1591     time.Sleep(d)
1592     if 0 < len(argv[2:]) {
1593         gshCtx.gshellv(argv[2:])
1594     }
1595 }
1596 func (gshCtx *GshContext)repeat(argv []string) {
1597     if len(argv) < 2 {
1598         return
1599     }
1600     start0 := time.Now()
1601     for ri, _ := strconv.Atoi(argv[1]); 0 < ri; ri-- {
1602         if 0 < len(argv[2:]) {
1603             //start := time.Now()
1604             gshCtx.gshellv(argv[2:])
1605             end := time.Now()
1606             elps := end.Sub(start0);
1607             if( 1000000000 < elps ){
1608                 fmt.Printf("(repeat#%d %v)\n",ri,elps);
1609             }
1610         }
1611     }
1612 }

```

```

1613
1614 func (gshCtx *GshContext)gen(argv []string) {
1615     gshPA := gshCtx.gshPA
1616     if len(argv) < 2 {
1617         fmt.Printf("Usage: %s N\n",argv[0])
1618         return
1619     }
1620     // should br repeated by "repeat" command
1621     count, _ := strconv.Atoi(argv[1])
1622     fd := gshPA.Files[1] // Stdout
1623     file := os.NewFile(fd,"internalStdOut")
1624     fmt.Printf("--I-- Gen. Count=%d to [%d]\n",count,file.Fd())
1625     //buf := []byte{}
1626     outdata := "0123 5678 0123 5678 0123 5678 0123 5678\r"
1627     for gi := 0; gi < count; gi++ {
1628         file.WriteString(outdata)
1629     }
1630     //file.WriteString("\n")
1631     fmt.Printf("\n(%d B)\n",count*len(outdata));
1632     //file.Close()
1633 }
1634
1635 // <a name="rexec">Remote Execution</a> // 2020-0820
1636 func Elapsed(from time.Time)(string){
1637     elps := time.Now().Sub(from)
1638     if 1000000000 < elps {
1639         return fmt.Sprintf("[%5d.%02ds]",elps/1000000000,(elps%1000000000)/1000000)
1640     }else
1641     if 1000000 < elps {
1642         return fmt.Sprintf("[%3d.%03dms]",elps/1000000,(elps%1000000)/1000)
1643     }else{
1644         return fmt.Sprintf("[%3d.%03dus]",elps/1000,(elps%1000))
1645     }
1646 }
1647 func abftime(nanos int64)(string){
1648     if 1000000000 < nanos {
1649         return fmt.Sprintf("%d.%02ds",nanos/1000000000,(nanos%1000000000)/1000000)
1650     }else
1651     if 1000000 < nanos {
1652         return fmt.Sprintf("%d.%03dms",nanos/1000000,(nanos%1000000)/1000)
1653     }else{
1654         return fmt.Sprintf("%d.%03dus",nanos/1000,(nanos%1000))
1655     }
1656 }
1657 func absbsize(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 absz(size int64)(string){
1669     fsize := float64(size)
1670     if 1024*1024*1024 < size {
1671         return fmt.Sprintf("%.2fGiB",fsize/(1024*1024*1024))
1672     }else
1673     if 1024*1024 < size {
1674         return fmt.Sprintf("%.3fMiB",fsize/(1024*1024))
1675     }else{
1676         return fmt.Sprintf("%.3fKiB",fsize/1024)
1677     }
1678 }
1679 func abspspeed(totalB int64,ns int64)(string){
1680     MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
1681     if 1000 <= MBs {
1682         return fmt.Sprintf("%.3fGB/s",MBs/1000)
1683     }
1684     if 1 <= MBs {
1685         return fmt.Sprintf("%.3fMB/s",MBs)
1686     }else{
1687         return fmt.Sprintf("%.3fKB/s",MBs*1000)
1688     }
1689 }
1690 func abspspeed(totalB int64,ns time.Duration)(string){
1691     MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
1692     if 1000 <= MBs {
1693         return fmt.Sprintf("%.3fGBps",MBs/1000)
1694     }
1695     if 1 <= MBs {
1696         return fmt.Sprintf("%.3fMBps",MBs)
1697     }else{
1698         return fmt.Sprintf("%.3fKBps",MBs*1000)
1699     }
1700 }
1701 func fileRelay(what string,in*os.File,out*os.File,size int64,bsiz int)(wcount int64){
1702     Start := time.Now()
1703     buff := make([]byte,bsiz)
1704     var total int64 = 0
1705     var rem int64 = size
1706     nio := 0
1707     Prev := time.Now()
1708     var PrevSize int64 = 0
1709
1710     fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) START\n",
1711         what,absz(total),size,nio)
1712
1713     for i:= 0; ; i++ {
1714         var len = bsiz
1715         if int(rem) < len {
1716             len = int(rem)
1717         }
1718         Now := time.Now()
1719         Elps := Now.Sub(Prev);
1720         if 1000000000 < Now.Sub(Prev) {
1721             fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) %s\n",
1722                 what,absz(total),size,nio,
1723                 abspspeed((total-PrevSize),Elps))
1724             Prev = Now;
1725             PrevSize = total
1726         }
1727         rlen := len
1728         if in != nil {
1729             // should watch the disconnection of out
1730             rcc,err := in.Read(buff[0:rlen])
1731             if err != nil {
1732                 fmt.Printf(Elapsed(Start)+"--En- X: %s read(%v,%v)<%v\n",
1733                     what,rcc,err,in.Name())
1734                 break
1735             }
1736             rlen = rcc

```

```

1737         if string(buff[0:10]) == "(SoftEOF " {
1738             var ecc int64 = 0
1739             fmt.Sscanf(string(buff), "(SoftEOF %v", &ecc)
1740             fmt.Printf(Elapsed(Start)+"--En- X: %s Recv ((SoftEOF %v))/%v\n",
1741                 what, ecc, total)
1742             if ecc == total {
1743                 break
1744             }
1745         }
1746     }
1747
1748     wlen := rlen
1749     if out != nil {
1750         wcc, err := out.Write(buff[0:rlen])
1751         if err != nil {
1752             fmt.Printf(Elapsed(Start)+"--En-- X: %s write(%v,%v)>%v\n",
1753                 what, wcc, err, out.Name())
1754             break
1755         }
1756         wlen = wcc
1757     }
1758     if wlen < rlen {
1759         fmt.Printf(Elapsed(Start)+"--En- X: %s incomplete write (%v/%v)\n",
1760             what, wlen, rlen)
1761         break;
1762     }
1763
1764     nio += 1
1765     total += int64(rlen)
1766     rem -= int64(rlen)
1767     if rem <= 0 {
1768         break
1769     }
1770 }
1771 Done := time.Now()
1772 Elps := float64(Done.Sub(Start))/1000000000 //Seconds
1773 TotalMB := float64(total)/1000000 //MB
1774 MBps := TotalMB / Elps
1775 fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) %v %.3fMB/s\n",
1776     what, total, size, nio, abs(size), MBps)
1777 return total
1778 }
1779 func tcpPush(clnt *os.File){
1780     // shrink socket buffer and recover
1781     usleep(100);
1782 }
1783 func (gsh*GshContext)RexecServer(argv []string){
1784     debug := true
1785     Start0 := time.Now()
1786     Start := Start0
1787     // if local == ":" { local = "0.0.0.0:9999" }
1788     local := "0.0.0.0:9999"
1789
1790     if 0 < len(argv) {
1791         if argv[0] == "-s" {
1792             debug = false
1793             argv = argv[1:]
1794         }
1795     }
1796     if 0 < len(argv) {
1797         argv = argv[1:]
1798     }
1799     port, err := net.ResolveTCPAddr("tcp", local);
1800     if err != nil {
1801         fmt.Printf("--En- S: Address error: %s (%s)\n", local, err)
1802         return
1803     }
1804     fmt.Printf(Elapsed(Start)+"--In- S: Listening at %s...\n", local);
1805     sconn, err := net.ListenTCP("tcp", port)
1806     if err != nil {
1807         fmt.Printf(Elapsed(Start)+"--En- S: Listen error: %s (%s)\n", local, err)
1808         return
1809     }
1810
1811     reqbuf := make([]byte, LINESIZE)
1812     res := ""
1813     for {
1814         fmt.Printf(Elapsed(Start0)+"--In- S: Listening at %s...\n", local);
1815         aconn, err := sconn.AcceptTCP()
1816         Start = time.Now()
1817         if err != nil {
1818             fmt.Printf(Elapsed(Start)+"--En- S: Accept error: %s (%s)\n", local, err)
1819             return
1820         }
1821         clnt, _ := aconn.File()
1822         fd := clnt.Fd()
1823         ar := aconn.RemoteAddr()
1824         if debug { fmt.Printf(Elapsed(Start0)+"--In- S: Accepted TCP at %s [%d] <- %v\n",
1825             local, fd, ar) }
1826         res = fmt.Sprintf("220 GShell/%s Server\r\n", VERSION)
1827         fmt.Pprintf(clnt, "%s", res)
1828         if debug { fmt.Printf(Elapsed(Start)+"--In- S: %s", res) }
1829         count, err := clnt.Read(reqbuf)
1830         if err != nil {
1831             fmt.Printf(Elapsed(Start)+"--En- C: (%v %v) %v",
1832                 count, err, string(reqbuf))
1833         }
1834         req := string(reqbuf[:count])
1835         if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v", string(req)) }
1836         reqv := strings.Split(string(req), "\r")
1837         cmdv := gshScanArg(reqv[0], 0)
1838         //cmdv := strings.Split(reqv[0], " ")
1839         switch cmdv[0] {
1840             case "HELO":
1841                 res = fmt.Sprintf("250 %v", req)
1842             case "GET":
1843                 // download {remotefile|-zN} [localfile]
1844                 var dsize int64 = 32*1024*1024
1845                 var bsize int = 64*1024
1846                 var fname string = ""
1847                 var in *os.File = nil
1848                 var pseudoEOF = false
1849                 if 1 < len(cmdv) {
1850                     fname = cmdv[1]
1851                     if strBegins(fname, "-z") {
1852                         fmt.Sscanf(fname[2:], "%d", &dsize)
1853                     } else
1854                     if strBegins(fname, "{") {
1855                         xin, xout, err := gsh.Popen(fname, "r")
1856                         if err {
1857                             } else {
1858                                 xout.Close()
1859                                 defer xin.Close()
1860                                 in = xin

```

```

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

```



```

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

```

```

2109 }else{
2110     ress = string(res[:count])
2111 }
2112 if err != nil {
2113     fmt.Printf(Elapsed(Start)+"--En- S: (%d,%v) %v",count,err,ress)
2114 }else{
2115     fmt.Printf(Elapsed(Start)+"--In- S: %v",ress)
2116 }
2117 serv.Close()
2118 //conn.Close()
2119
2120 var stat string
2121 var rcode int
2122 fmt.Sscanf(ress,"%d %s",&rcode,&stat)
2123 //fmt.Printf("--D-- Client: %v (%v)",rcode,stat)
2124 return rcode,ress
2125 }
2126
2127 // <a name="remote-sh">Remote Shell</a>
2128 // gcp file [...] { host:[port]:[dir] | dir } // -p | -no-p
2129 func (gsh*GshContext)FileCopy(argv []string){
2130     var host = ""
2131     var port = ""
2132     var upload = false
2133     var download = false
2134     var xargv = []string{"rex-gcp"}
2135     var srcv = []string{}
2136     var dstv = []string{}
2137     argv = argv[1:]
2138
2139     for _,v := range argv {
2140         /*
2141         if v[0] == '-' { // might be a pseudo file (generated date)
2142             continue
2143         }
2144         */
2145         obj := strings.Split(v,":")
2146         //fmt.Printf("%d %v %v\n",len(obj),v,obj)
2147         if 1 < len(obj) {
2148             host = obj[0]
2149             file := ""
2150             if 0 < len(host) {
2151                 gsh.LastServer.host = host
2152             }else{
2153                 host = gsh.LastServer.host
2154                 port = gsh.LastServer.port
2155             }
2156             if 2 < len(obj) {
2157                 port = obj[1]
2158                 if 0 < len(port) {
2159                     gsh.LastServer.port = port
2160                 }else{
2161                     port = gsh.LastServer.port
2162                 }
2163                 file = obj[2]
2164             }else{
2165                 file = obj[1]
2166             }
2167             if len(srcv) == 0 {
2168                 download = true
2169                 srcv = append(srcv,file)
2170                 continue
2171             }
2172             upload = true
2173             dstv = append(dstv,file)
2174             continue
2175         }
2176         /*
2177         idx := strings.Index(v,":")
2178         if 0 <= idx {
2179             remote = v[0:idx]
2180             if len(srcv) == 0 {
2181                 download = true
2182                 srcv = append(srcv,v[idx+1:])
2183                 continue
2184             }
2185             upload = true
2186             dstv = append(dstv,v[idx+1:])
2187             continue
2188         }
2189         */
2190         if download {
2191             dstv = append(dstv,v)
2192         }else{
2193             srcv = append(srcv,v)
2194         }
2195     }
2196     hostport := "@" + host + ":" + port
2197     if upload {
2198         if host != "" { xargv = append(xargv,hostport) }
2199         xargv = append(xargv,"PUT")
2200         xargv = append(xargv,srcv[0]...)
2201         xargv = append(xargv,dstv[0]...)
2202         //fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v // %v\n",hostport,dstv,srcv,xargv)
2203         fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v\n",hostport,dstv,srcv)
2204         gsh.RexecClient(xargv)
2205     }else
2206     if download {
2207         if host != "" { xargv = append(xargv,hostport) }
2208         xargv = append(xargv,"GET")
2209         xargv = append(xargv,srcv[0]...)
2210         xargv = append(xargv,dstv[0]...)
2211         //fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v // %v\n",hostport,srcv,dstv,xargv)
2212         fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v\n",hostport,srcv,dstv)
2213         gsh.RexecClient(xargv)
2214     }else{
2215     }
2216 }
2217
2218 // target
2219 func (gsh*GshContext)Trelpath(rloc string)(string){
2220     cwd, _ := os.Getwd()
2221     os.Chdir(gsh.RWD)
2222     os.Chdir(rloc)
2223     twd, _ := os.Getwd()
2224     os.Chdir(cwd)
2225
2226     tpath := twd + "/" + rloc
2227     return tpath
2228 }
2229 // join to rmote GShell - [user@]host[:port] or cd host[:port]:path
2230 func (gsh*GshContext)Rjoin(argv []string){
2231     if len(argv) <= 1 {
2232         fmt.Printf("--I-- current server = %v\n",gsh.RSERV)

```

```

2233     return
2234 }
2235 serv := argv[1]
2236 servv := strings.Split(serv, ":")
2237 if 1 <= len(servv) {
2238     if servv[0] == "lo" {
2239         servv[0] = "localhost"
2240     }
2241 }
2242 switch len(servv) {
2243 case 1:
2244     //if strings.Index(serv, ":") < 0 {
2245     serv = servv[0] + ":" + fmt.Sprintf("%d", GSH_PORT)
2246     //}
2247 case 2: // host:port
2248     serv = strings.Join(servv, ":")
2249 }
2250 xargv := []string{"rex-join", "@"+serv, "HELO"}
2251 rcode, stat := gsh.RexecClient(xargv)
2252 if (rcode / 100) == 2 {
2253     fmt.Printf("--I-- OK Joined (%v) %v\n", rcode, stat)
2254     gsh.RSERV = serv
2255 }else{
2256     fmt.Printf("--I-- NG, could not joined (%v) %v\n", rcode, stat)
2257 }
2258 }
2259 func (gsh*GshContext)Rexec(argv []string){
2260     if len(argv) <= 1 {
2261         fmt.Printf("--I-- rexec command [ | {file || {command} ]\n", gsh.RSERV)
2262         return
2263     }
2264 }
2265 /*
2266 nargv := gsh.ScanArg(strings.Join(argv, " "), 0)
2267 fmt.Printf("--D-- nargc=%d [%v]\n", len(nargv), nargv)
2268 if nargv[1][0] != '{' {
2269     nargv[1] = "{" + nargv[1] + "}"
2270     fmt.Printf("--D-- nargc=%d [%v]\n", len(nargv), nargv)
2271 }
2272 argv = nargv
2273 */
2274 nargv := []string{}
2275 nargv = append(nargv, "{"+strings.Join(argv[1:], " ")+"}")
2276 fmt.Printf("--D-- nargc=%d [%v]\n", len(nargv), nargv)
2277 argv = nargv
2278
2279 xargv := []string{"rex-exec", "@"+gsh.RSERV, "GET"}
2280 xargv = append(xargv, argv...)
2281 xargv = append(xargv, "/dev/tty")
2282 rcode, stat := gsh.RexecClient(xargv)
2283 if (rcode / 100) == 2 {
2284     fmt.Printf("--I-- OK Rexec (%v) %v\n", rcode, stat)
2285 }else{
2286     fmt.Printf("--I-- NG Rexec (%v) %v\n", rcode, stat)
2287 }
2288 }
2289 func (gsh*GshContext)Rchdir(argv []string){
2290     if len(argv) <= 1 {
2291         return
2292     }
2293     cwd, _ := os.Getwd()
2294     os.Chdir(gsh.RWD)
2295     os.Chdir(argv[1])
2296     twd, _ := os.Getwd()
2297     gsh.RWD = twd
2298     fmt.Printf("--I-- JWD=%v\n", twd)
2299     os.Chdir(cwd)
2300 }
2301 func (gsh*GshContext)Rpwd(argv []string){
2302     fmt.Printf("%v\n", gsh.RWD)
2303 }
2304 func (gsh*GshContext)Rls(argv []string){
2305     cwd, _ := os.Getwd()
2306     os.Chdir(gsh.RWD)
2307     argv[0] = "-ls"
2308     gsh.xFind(argv)
2309     os.Chdir(cwd)
2310 }
2311 func (gsh*GshContext)Rput(argv []string){
2312     var local string = ""
2313     var remote string = ""
2314     if 1 < len(argv) {
2315         local = argv[1]
2316         remote = local // base name
2317     }
2318     if 2 < len(argv) {
2319         remote = argv[2]
2320     }
2321     fmt.Printf("--I-- jput from=%v to=%v\n", local, gsh.Trepath(remote))
2322 }
2323 func (gsh*GshContext)Rget(argv []string){
2324     var remote string = ""
2325     var local string = ""
2326     if 1 < len(argv) {
2327         remote = argv[1]
2328         local = remote // base name
2329     }
2330     if 2 < len(argv) {
2331         local = argv[2]
2332     }
2333     fmt.Printf("--I-- jget from=%v to=%v\n", gsh.Trepath(remote), local)
2334 }
2335 }
2336 // <a name="network">network</a>
2337 // -s, -si, -so // bi-directional, source, sync (maybe socket)
2338 func (gshCtx*GshContext)sconnect(inTCP bool, argv []string) {
2339     gshPA := gshCtx.gshPA
2340     if len(argv) < 2 {
2341         fmt.Printf("Usage: -s [host]:[port.udp]\n")
2342         return
2343     }
2344     remote := argv[1]
2345     if remote == ":" { remote = "0.0.0.0:9999" }
2346
2347     if inTCP { // TCP
2348         dport, err := net.ResolveTCPAddr("tcp", remote);
2349         if err != nil {
2350             fmt.Printf("Address error: %s (%s)\n", remote, err)
2351             return
2352         }
2353         conn, err := net.DialTCP("tcp", nil, dport)
2354         if err != nil {
2355             fmt.Printf("Connection error: %s (%s)\n", remote, err)
2356             return
2357         }
2358     }

```

```

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

```

```

2481     showFileInfo(cwd,argv)
2482     default:
2483         fmt.Printf("%s\n",cwd)
2484     case isin("v",argv): // obsolete empty command
2485         t := time.Now()
2486         date := t.Format(time.UnixDate)
2487         exe, _ := os.Executable()
2488         host, _ := os.Hostname()
2489         fmt.Printf("PWD=\"%s\"",cwd)
2490         fmt.Printf("HOST=\"%s\"",host)
2491         fmt.Printf("DATE=\"%s\"",date)
2492         fmt.Printf("TIME=\"%s\"",t.String())
2493         fmt.Printf("PID=\"%d\"",os.Getpid())
2494         fmt.Printf("EXE=\"%s\"",exe)
2495         fmt.Printf("\n")
2496     }
2497 }
2498
2499 // <a name="history">History</a>
2500 // these should be browsed and edited by HTTP browser
2501 // show the time of command with -t and direcotry with -ls
2502 // openfile-history, sort by -a -m -c
2503 // sort by elapsed time by -t -s
2504 // search by "more" like interface
2505 // edit history
2506 // sort history, and wc or uniq
2507 // CPU and other resource consumptions
2508 // limit showing range (by time or so)
2509 // export / import history
2510 func (gshCtx *GshContext)xHistory(argv []string){
2511     atWorkDirX := -1
2512     if 1 < len(argv) && strBegins(argv[1],"@") {
2513         atWorkDirX,_ = strconv.Atoi(argv[1][1:])
2514     }
2515     //fmt.Printf("--D-- showHistory(%v)\n",argv)
2516     for i, v := range gshCtx.CommandHistory {
2517         // exclude commands not to be listed by default
2518         // internal commands may be suppressed by default
2519         if v.CmdLine == "" && !isin("-a",argv) {
2520             continue;
2521         }
2522         if 0 <= atWorkDirX {
2523             if v.WorkDirX != atWorkDirX {
2524                 continue
2525             }
2526         }
2527         if !isin("-n",argv){ // like "fc"
2528             fmt.Printf("!\%2d ",i)
2529         }
2530         if isin("-v",argv){
2531             fmt.Println(v) // should be with it date
2532         }else{
2533             if isin("-l",argv) || isin("-l0",argv) {
2534                 elps := v.EndAt.Sub(v.StartAt);
2535                 start := v.StartAt.Format(time.Stamp)
2536                 fmt.Printf("!\%d ",v.WorkDirX)
2537                 fmt.Printf("[%v] %11v/t ",start,elps)
2538             }
2539             if isin("-l",argv) && !isin("-l0",argv){
2540                 fmt.Printf("%v",Rusagef("%t %u\t// %s",argv,v.Rusagev))
2541             }
2542             if isin("-at",argv) { // isin("-ls",argv){
2543                 dhi := v.WorkDirX // workdir history index
2544                 fmt.Printf("!\%d %s\t",dhi,v.WorkDir)
2545                 // show the FileInfo of the output command??
2546             }
2547             fmt.Printf("%s",v.CmdLine)
2548             fmt.Printf("\n")
2549         }
2550     }
2551 }
2552 // In - history index
2553 func searchHistory(gshCtx GshContext, gline string) (string, bool, bool){
2554     if gline[0] == '!' {
2555         hix, err := strconv.Atoi(gline[1:])
2556         if err != nil {
2557             fmt.Printf("--E-- (%s : range)\n",hix)
2558             return "", false, true
2559         }
2560         if hix < 0 || len(gshCtx.CommandHistory) <= hix {
2561             fmt.Printf("--E-- (%d : out of range)\n",hix)
2562             return "", false, true
2563         }
2564         return gshCtx.CommandHistory[hix].CmdLine, false, false
2565     }
2566     // search
2567     //for i, v := range gshCtx.CommandHistory {
2568     //}
2569     //}
2570 }
2571 func (gsh*GshContext)cmdStringInHistory(hix int)(cmd string, ok bool){
2572     if 0 <= hix && hix < len(gsh.CommandHistory) {
2573         return gsh.CommandHistory[hix].CmdLine,true
2574     }
2575     return "",false
2576 }
2577
2578 // temporary adding to PATH environment
2579 // cd name -lib for LD_LIBRARY_PATH
2580 // chdir with directory history (date + full-path)
2581 // -s for sort option (by visit date or so)
2582 func (gsh*GshContext)ShowChdirHistory(i int,v GChdirHistory, argv []string){
2583     fmt.Printf("!\%2d ",v.CmdIndex) // the first command at this WorkDir
2584     fmt.Printf("!\%d ",i)
2585     fmt.Printf("[%v] ",v.MovedAt.Format(time.Stamp))
2586     showFileInfo(v.Dir,argv)
2587 }
2588 func (gsh*GshContext)ShowChdirHistory(argv []string){
2589     for i, v := range gsh.ChdirHistory {
2590         gsh.ShowChdirHistory1(i,v,argv)
2591     }
2592 }
2593 func skipOpts(argv[]string)(int){
2594     for i,v := range argv {
2595         if strBegins(v,"-") {
2596             }else{
2597                 return i
2598             }
2599     }
2600     return -1
2601 }
2602 func (gshCtx*GshContext)xChdir(argv []string){
2603     cdhist := gshCtx.ChdirHistory
2604     if isin("?",argv) || isin("-t",argv) || isin("-a",argv) {

```

```

2605     gshCtx.ShowChdirHistory(argv)
2606     return
2607 }
2608 pwd, _ := os.Getwd()
2609 dir := ""
2610 if len(argv) <= 1 {
2611     dir = toFullpath("~/")
2612 }else{
2613     i := skipOpts(argv[1:])
2614     if i < 0 {
2615         dir = toFullpath("~/")
2616     }else{
2617         dir = argv[1+i]
2618     }
2619 }
2620 if strBegins(dir, "@") {
2621     if dir == "@0" { // obsolete
2622         dir = gshCtx.StartDir
2623     }else
2624     if dir == "@!" {
2625         index := len(cdhist) - 1
2626         if 0 < index { index -= 1 }
2627         dir = cdhist[index].Dir
2628     }else{
2629         index, err := stroconv.Atoi(dir[1:])
2630         if err != nil {
2631             fmt.Printf("--E-- xChdir(%v)\n",err)
2632             dir = "?"
2633         }else
2634         if len(gshCtx.ChdirHistory) <= index {
2635             fmt.Printf("--E-- xChdir(history range error)\n")
2636             dir = "?"
2637         }else{
2638             dir = cdhist[index].Dir
2639         }
2640     }
2641 }
2642 if dir != "?" {
2643     err := os.Chdir(dir)
2644     if err != nil {
2645         fmt.Printf("--E-- xChdir(%s)(%v)\n",argv[1],err)
2646     }else{
2647         cwd, _ := os.Getwd()
2648         if cwd != pwd {
2649             hist1 := GChdirHistory { }
2650             hist1.Dir = cwd
2651             hist1.MovedAt = time.Now()
2652             hist1.CmdIndex = len(gshCtx.CommandHistory)+1
2653             gshCtx.ChdirHistory = append(cdhist,hist1)
2654             if !isin("-s",argv){
2655                 //cwd, _ := os.Getwd()
2656                 //fmt.Printf("%s\n",cwd)
2657                 ix := len(gshCtx.ChdirHistory)-1
2658                 gshCtx.ShowChdirHistory1(ix,hist1,argv)
2659             }
2660         }
2661     }
2662 }
2663 if isin("-ls",argv){
2664     cwd, _ := os.Getwd()
2665     showFileInfo(cwd,argv);
2666 }
2667 }
2668 func TimeValSub(tv1 *syscall.Timeval, tv2 *syscall.Timeval){
2669     *tv1 = syscall.NsecToTimeval(tv1.Nano() - tv2.Nano())
2670 }
2671 func RusageSubv(rul, ru2 [2]syscall.Rusage){[2]syscall.Rusage){
2672     TimeValSub(&rul[0].Utime,&ru2[0].Utime)
2673     TimeValSub(&rul[0].Stime,&ru2[0].Stime)
2674     TimeValSub(&rul[1].Utime,&ru2[1].Utime)
2675     TimeValSub(&rul[1].Stime,&ru2[1].Stime)
2676     return rul
2677 }
2678 func TimeValAdd(tv1 syscall.Timeval, tv2 syscall.Timeval)(syscall.Timeval){
2679     tvs := syscall.NsecToTimeval(tv1.Nano() + tv2.Nano())
2680     return tvs
2681 }
2682 /*
2683 func RusageAddv(rul, ru2 [2]syscall.Rusage){[2]syscall.Rusage){
2684     TimeValAdd(rul[0].Utime,ru2[0].Utime)
2685     TimeValAdd(rul[0].Stime,ru2[0].Stime)
2686     TimeValAdd(rul[1].Utime,ru2[1].Utime)
2687     TimeValAdd(rul[1].Stime,ru2[1].Stime)
2688     return rul
2689 }
2690 */
2691 // <a name="rusage">Resource Usage</a>
2692 func sRusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2693     // ru[0] self, ru[1] children
2694     ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
2695     st := TimeValAdd(ru[0].Stime,ru[1].Stime)
2696     uu := (ut.Sec*1000000 + int64(ut.Usec)) * 1000
2697     su := (st.Sec*1000000 + int64(st.Usec)) * 1000
2698     tu := uu + su
2699     ret := fmt.Sprintf("%v/sum",abtime(tu))
2700     ret += fmt.Sprintf(", %v/usr",abtime(uu))
2701     ret += fmt.Sprintf(", %v/sys",abtime(su))
2702     return ret
2703 }
2704 }
2705 func Rusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
2706     ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
2707     st := TimeValAdd(ru[0].Stime,ru[1].Stime)
2708     fmt.Printf("%d.%06ds/u ",ut.Sec,ut.Usec) //ru[1].Utime.Sec,ru[1].Utime.Usec)
2709     fmt.Printf("%d.%06ds/s ",st.Sec,st.Usec) //ru[1].Stime.Sec,ru[1].Stime.Usec)
2710     return ""
2711 }
2712 func Getrusagev(){[2]syscall.Rusage){
2713     var ruv = [2]syscall.Rusage{
2714         syscall.Getrusage(syscall.RUSAGE_SELF,&ruv[0])
2715         syscall.Getrusage(syscall.RUSAGE_CHILDREN,&ruv[1])
2716     }
2717     return ruv
2718 }
2719 func showRusage(what string,argv []string, ru *syscall.Rusage){
2720     fmt.Printf("%s: ",what);
2721     fmt.Printf(" Utr=%d.%06ds",ru.Utime.Sec,ru.Utime.Usec)
2722     fmt.Printf(" Sst=%d.%06ds",ru.Stime.Sec,ru.Stime.Usec)
2723     fmt.Printf(" Rss=%vB",ru.Maxrss)
2724     if isin("-l",argv) {
2725         fmt.Printf(" MinFlt=%v",ru.Minflt)
2726         fmt.Printf(" MajFlt=%v",ru.Majflt)
2727         fmt.Printf(" Ixrss=%vB",ru.Ixrss)
2728         fmt.Printf(" IdRSS=%vB",ru.Idrss)
2729         fmt.Printf(" Nswap=%vB",ru.Nswap)

```

```

2729     fmt.Printf(" Read=%v",ru.Inblock)
2730     fmt.Printf(" Write=%v",ru.Outblock)
2731 }
2732     fmt.Printf(" Snd=%v",ru.Msgsnd)
2733     fmt.Printf(" Rcv=%v",ru.Msgrcv)
2734 //if isin("-l",argv) {
2735     fmt.Printf(" Sig=%v",ru.Nsignals)
2736 //}
2737     fmt.Printf("\n");
2738 }
2739 func (gshCtx *GshContext)xTime(argv[]string)(bool){
2740     if 2 <= len(argv){
2741         gshCtx.LastRusage = syscall.Rusage{
2742             rusagev1 := Getrusagev()
2743             fin := gshCtx.gshellv(argv[1])
2744             rusagev2 := Getrusagev()
2745             showRusage(argv[1],argv,&gshCtx.LastRusage)
2746             rusagev := RusageSubv(rusagev2,rusagev1)
2747             showRusage("self",argv,&rusagev[0])
2748             showRusage("chld",argv,&rusagev[1])
2749             return fin
2750         }else{
2751             rusage:= syscall.Rusage {
2752                 syscall.Getrusage(syscall.RUSAGE_SELF,&rusage)
2753                 showRusage("self",argv, &rusage)
2754                 syscall.Getrusage(syscall.RUSAGE_CHILDREN,&rusage)
2755                 showRusage("chld",argv, &rusage)
2756                 return false
2757             }
2758         }
2759     func (gshCtx *GshContext)xJobs(argv[]string){
2760         fmt.Printf("%d Jobs\n",len(gshCtx.BackGroundJobs))
2761         for ji, pid := range gshCtx.BackGroundJobs {
2762             //wstat := syscall.WaitStatus {0}
2763             rusage := syscall.Rusage {}
2764             //wpid, err := syscall.Wait4(pid,&wstat,syscall.WNOHANG,&rusage);
2765             wpid, err := syscall.Wait4(pid,nil,syscall.WNOHANG,&rusage);
2766             if err != nil {
2767                 fmt.Printf("--E-- %%%d [%d] (%v)\n",ji,pid,err)
2768             }else{
2769                 fmt.Printf("%%d[%d](%d)\n",ji,pid,wpid)
2770                 showRusage("chld",argv,&rusage)
2771             }
2772         }
2773     }
2774     func (gsh*GshContext)inBackground(argv[]string)(bool){
2775         if gsh.CmdTrace { fmt.Printf("--I-- inBackground(%v)\n",argv) }
2776         gsh.BackGround = true // set background option
2777         xfin := false
2778         xfin = gsh.gshellv(argv)
2779         gsh.BackGround = false
2780         return xfin
2781     }
2782     // -o file without command means just opening it and refer by #N
2783     // should be listed by "files" command
2784     func (gshCtx*GshContext)xOpen(argv[]string){
2785         var pv = [int(-1,-1)]
2786         err := syscall.Pipe(pv)
2787         fmt.Printf("--I-- pipe()=[#%d,#%d](%v)\n",pv[0],pv[1],err)
2788     }
2789     func (gshCtx*GshContext)fromPipe(argv[]string){
2790     }
2791     func (gshCtx*GshContext)xClose(argv[]string){
2792     }
2793 }
2794 // <a name="redirect">redirect</a>
2795 func (gshCtx*GshContext)redirect(argv[]string)(bool){
2796     if len(argv) < 2 {
2797         return false
2798     }
2799     cmd := argv[0]
2800     fname := argv[1]
2801     var file *os.File = nil
2802     fdix := 0
2803     mode := os.O_RDONLY
2804     switch {
2805     case cmd == "-i" || cmd == "<":
2806         fdix = 0
2807         mode = os.O_RDONLY
2808     case cmd == "-o" || cmd == ">":
2809         fdix = 1
2810         mode = os.O_RDWR | os.O_CREATE
2811     case cmd == "-a" || cmd == ">>":
2812         fdix = 1
2813         mode = os.O_RDWR | os.O_CREATE | os.O_APPEND
2814     }
2815     if fname[0] == '#' {
2816         fd, err := strconv.Atoi(fname[1:])
2817         if err != nil {
2818             fmt.Printf("--E-- (%v)\n",err)
2819             return false
2820         }
2821         file = os.NewFile(uintptr(fd),"MaybePipe")
2822     }else{
2823         xfile, err := os.OpenFile(argv[1], mode, 0600)
2824         if err != nil {
2825             fmt.Printf("--E-- (%s)\n",err)
2826             return false
2827         }
2828         file = xfile
2829     }
2830     gshPA := gshCtx.gshPA
2831     savfd := gshPA.Files[fdix]
2832     gshPA.Files[fdix] = file.Fd()
2833     fmt.Printf("--I-- Opened [%d] %s\n",file.Fd(),argv[1])
2834     gshCtx.gshellv(argv[2:])
2835     gshPA.Files[fdix] = savfd
2836 }
2837     return false
2838 }
2839 //fmt.Fprintf(res, "GShell Status: %q", html.EscapeString(req.URL.Path))
2840 func httpHandler(res http.ResponseWriter, req *http.Request){
2841     path := req.URL.Path
2842     fmt.Printf("--I-- Got HTTP Request(%s)\n",path)
2843     {
2844         gshCtxBuf, _ := setupGshContext()
2845         gshCtx := &gshCtxBuf
2846         fmt.Printf("--I-- %s\n",path[1:])
2847         gshCtx.tgshellv(path[1:])
2848     }
2849 }

```

```

2853     fmt.Fprintf(res, "Hello(^-^)/\n%s\n",path)
2854 }
2855 func (gshCtx *GshContext) httpServer(argv []string){
2856     http.HandleFunc("/", httpHandler)
2857     accept := "localhost:9999"
2858     fmt.Printf("--I-- HTTP Server Start at [%s]\n",accept)
2859     http.ListenAndServe(accept,nil)
2860 }
2861 func (gshCtx *GshContext)xGo(argv[]string){
2862     go gshCtx.gshellv(argv[1:]);
2863 }
2864 func (gshCtx *GshContext) xPs(argv[]string){}
2865 }
2866
2867 // <a name="plugin">Plugin</a>
2868 // plugin [-ls [names]] to list plugins
2869 // Reference: <a href="https://golang.org/src/plugin/">plugin</a> source code
2870 func (gshCtx *GshContext) whichPlugin(name string,argv[] string)(pi *PluginInfo){
2871     pi = nil
2872     for _,p := range gshCtx.PluginFuncs {
2873         if p.Name == name && pi == nil {
2874             pi = &p
2875         }
2876         if !isin("-s",argv){
2877             //fmt.Printf("%v %v ".i,p)
2878             if isin("-ls",argv){
2879                 showFileInfo(p.Path,argv)
2880             }else{
2881                 fmt.Printf("%s\n",p.Name)
2882             }
2883         }
2884     }
2885     return pi
2886 }
2887 func (gshCtx *GshContext) xPlugin(argv[]string) (error) {
2888     if len(argv) == 0 || argv[0] == "-ls" {
2889         gshCtx.whichPlugin("",argv)
2890         return nil
2891     }
2892     name := argv[0]
2893     Pin := gshCtx.whichPlugin(name,[]string{"-s"})
2894     if Pin != nil {
2895         os.Args = argv // should be recovered?
2896         Pin.Addr.(func())()
2897         return nil
2898     }
2899     sofile := toFullpath(argv[0] + ".so") // or find it by which($PATH)
2900
2901     p, err := plugin.Open(sofile)
2902     if err != nil {
2903         fmt.Printf("--E-- plugin.Open(%s)(%v)\n",sofile,err)
2904         return err
2905     }
2906     fname := "Main"
2907     f, err := p.Lookup(fname)
2908     if( err != nil ){
2909         fmt.Printf("--E-- plugin.Lookup(%s)(%v)\n",fname,err)
2910         return err
2911     }
2912     pin := PluginInfo {p,f,name,sofile}
2913     gshCtx.PluginFuncs = append(gshCtx.PluginFuncs,pin)
2914     fmt.Printf("--I-- added (%d)\n",len(gshCtx.PluginFuncs))
2915
2916     //fmt.Printf("--I-- first call(%s:%s)%v\n",sofile,fname,argv)
2917     os.Args = argv
2918     f.(func())()
2919     return err
2920 }
2921 func (gshCtx*GshContext)Args(argv[]string){
2922     for i,v := range os.Args {
2923         fmt.Printf("[%v] %v\n",i,v)
2924     }
2925 }
2926 func (gshCtx *GshContext) showVersion(argv[]string){
2927     if isin("-l",argv) {
2928         fmt.Printf("%v/%v (%v)",NAME,VERSION,DATE);
2929     }else{
2930         fmt.Printf("%v",VERSION);
2931     }
2932     if isin("-a",argv) {
2933         fmt.Printf(" %s",AUTHOR)
2934     }
2935     if !isin("-n",argv) {
2936         fmt.Printf("\n")
2937     }
2938 }
2939
2940 // <a name="scanf">Scanf</a> // string decomposer
2941 // scanf [format] [input]
2942 func scanf(str string)(strv[]string){
2943     strv = strings.Split(str," ")
2944     return strv
2945 }
2946 func scanUntil(src,end string)(rstr string, leng int){
2947     idx := strings.Index(src,end)
2948     if 0 <= idx {
2949         rstr = src[0:idx]
2950         return rstr,idx+len(end)
2951     }
2952     return src,0
2953 }
2954
2955 // -bn -- display base-name part only // can be in some %fmt, for sed rewriting
2956 func (gsh*GshContext)printVal(fmts string, vstr string, optv[]string){
2957     //vint,err := strconv.Atoi(vstr)
2958     var ival int64 = 0
2959     n := 0
2960     err := error(nil)
2961     if strBegins(vstr," ") {
2962         vx, _ := strconv.Atoi(vstr[1:])
2963         if vx < len(gsh.iValues) {
2964             vstr = gsh.iValues[vx]
2965         }else{
2966         }
2967     }
2968     // should use Eval()
2969     if strBegins(vstr,"0x") {
2970         n,err = fmt.Sscanf(vstr[2:], "%x", &ival)
2971     }else{
2972         n,err = fmt.Sscanf(vstr, "%d", &ival)
2973     }
2974     //fmt.Printf("--D-- n=%d err=(%v) (%s)=%v\n",n,err,vstr, ival)
2975     if n == 1 && err == nil {
2976         //fmt.Printf("--D-- formatn(%v) ival(%v)\n",fmts,ival)

```



```

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

```

```

3101 sv := gsh.iValues
3102 sort.Slice(sv, func(i, j int) bool {
3103     return sv[i] < sv[j]
3104 })
3105 }
3106 func (gsh*GshContext)Shiftv(argv []string){
3107     vi := len(gsh.iValues)
3108     if 0 < vi {
3109         if isin("-r", argv) {
3110             top := gsh.iValues[0]
3111             gsh.iValues = append(gsh.iValues[1:], top)
3112         }else{
3113             gsh.iValues = gsh.iValues[1:]
3114         }
3115     }
3116 }
3117
3118 func (gsh*GshContext)Enq(argv []string){
3119 }
3120 func (gsh*GshContext)Deq(argv []string){
3121 }
3122 func (gsh*GshContext)Push(argv []string){
3123     gsh.iValStack = append(gsh.iValStack, argv[1:])
3124     fmt.Printf("depth=%d\n", len(gsh.iValStack))
3125 }
3126 func (gsh*GshContext)Dump(argv []string){
3127     for i, v := range gsh.iValStack {
3128         fmt.Printf("%d %v\n", i, v)
3129     }
3130 }
3131 func (gsh*GshContext)Pop(argv []string){
3132     depth := len(gsh.iValStack)
3133     if 0 < depth {
3134         v := gsh.iValStack[depth-1]
3135         if isin("-cat", argv){
3136             gsh.iValues = append(gsh.iValues, v...)
3137         }else{
3138             gsh.iValues = v
3139         }
3140         gsh.iValStack = gsh.iValStack[0:depth-1]
3141         fmt.Printf("depth=%d %s\n", len(gsh.iValStack), gsh.iValues)
3142     }else{
3143         fmt.Printf("depth=%d\n", depth)
3144     }
3145 }
3146
3147 // <a name="interpreter">Command Interpreter</a>
3148 func (gshCtx*GshContext)gshellv(argv []string) (fin bool) {
3149     fin = false
3150
3151     if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr, "--I-- gshellv(%d)\n", len(argv)) }
3152     if len(argv) <= 0 {
3153         return false
3154     }
3155     xargv := []string{}
3156     for ai := 0; ai < len(argv); ai++ {
3157         xargv = append(xargv, strsubst(gshCtx, argv[ai], false))
3158     }
3159     argv = xargv
3160     if false {
3161         for ai := 0; ai < len(argv); ai++ {
3162             fmt.Printf("[%d] %s [%d]T\n",
3163                 ai, argv[ai], len(argv[ai]), argv[ai])
3164         }
3165     }
3166     cmd := argv[0]
3167     if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr, "--I-- gshellv(%d)%v\n", len(argv), argv) }
3168     switch { // https://tour.golang.org/flowcontrol/11
3169     case cmd == "":
3170         gshCtx.xPwd([]string{}); // empty command
3171     case cmd == "-x":
3172         gshCtx.CmdTrace = ! gshCtx.CmdTrace
3173     case cmd == "-xt":
3174         gshCtx.CmdTime = ! gshCtx.CmdTime
3175     case cmd == "-ot":
3176         gshCtx.sconnect(true, argv)
3177     case cmd == "-ou":
3178         gshCtx.sconnect(false, argv)
3179     case cmd == "-it":
3180         gshCtx.saccept(true, argv)
3181     case cmd == "-iu":
3182         gshCtx.saccept(false, argv)
3183     case cmd == "-i" || cmd == "<" || cmd == "-o" || cmd == ">" || cmd == "-a" || cmd == ">>" || cmd == "-s" || cmd == "><":
3184         gshCtx.redirect(argv)
3185     case cmd == "|":
3186         gshCtx.fromPipe(argv)
3187     case cmd == "args":
3188         gshCtx.Args(argv)
3189     case cmd == "bg" || cmd == "-bg":
3190         rfin := gshCtx.inBackground(argv[1:])
3191         return rfin
3192     case cmd == "bn":
3193         gshCtx.Basename(argv)
3194     case cmd == "call":
3195         _, _ = gshCtx.excommand(false, argv[1:])
3196     case cmd == "cd" || cmd == "chdir":
3197         gshCtx.xChdir(argv);
3198     case cmd == "-cksum":
3199         gshCtx.xFind(argv)
3200     case cmd == "-sum":
3201         gshCtx.xFind(argv)
3202     case cmd == "-sumtest":
3203         str := ""
3204         if 1 < len(argv) { str = argv[1] }
3205         crc := strCRC32(str, uint64(len(str)))
3206         fprintf(stderr, "%v %v\n", crc, len(str))
3207     case cmd == "close":
3208         gshCtx.xClose(argv)
3209     case cmd == "gcp":
3210         gshCtx.FileCopy(argv)
3211     case cmd == "dec" || cmd == "decode":
3212         gshCtx.Dec(argv)
3213     case cmd == "#define":
3214     case cmd == "dic" || cmd == "d":
3215         xDic(argv)
3216     case cmd == "dump":
3217         gshCtx.Dump(argv)
3218     case cmd == "echo" || cmd == "e":
3219         echo(argv, true)
3220     case cmd == "enc" || cmd == "encode":
3221         gshCtx.Enc(argv)
3222     case cmd == "env":
3223         env(argv)
3224     case cmd == "eval":

```

```

3225     xEval(argv[1:],true)
3226 case cmd == "ev" || cmd == "events":
3227     dumpEvents(argv)
3228 case cmd == "exec":
3229     _ = gshCtx.excommand(true,argv[1:])
3230     // should not return here
3231 case cmd == "exit" || cmd == "quit":
3232     // write Result code EXIT to 3>
3233     return true
3234 case cmd == "fdls":
3235     // dump the attributes of fds (of other process)
3236 case cmd == "-find" || cmd == "fin" || cmd == "ufind" || cmd == "uf":
3237     gshCtx.xFind(argv[1:])
3238 case cmd == "fu":
3239     gshCtx.xFind(argv[1:])
3240 case cmd == "fork":
3241     // mainly for a server
3242 case cmd == "-gen":
3243     gshCtx.gen(argv)
3244 case cmd == "-go":
3245     gshCtx.xGo(argv)
3246 case cmd == "grep":
3247     gshCtx.xFind(argv)
3248 case cmd == "gdeg":
3249     gshCtx.Deg(argv)
3250 case cmd == "genq":
3251     gshCtx.Enq(argv)
3252 case cmd == "gpop":
3253     gshCtx.Pop(argv)
3254 case cmd == "gpush":
3255     gshCtx.Push(argv)
3256 case cmd == "history" || cmd == "hi": // hi should be alias
3257     gshCtx.xHistory(argv)
3258 case cmd == "jobs":
3259     gshCtx.xJobs(argv)
3260 case cmd == "lisp" || cmd == "nlsp":
3261     gshCtx.SplitLine(argv)
3262 case cmd == "ls":
3263     gshCtx.xFind(argv)
3264 case cmd == "nop":
3265     // do nothing
3266 case cmd == "pipe":
3267     gshCtx.xOpen(argv)
3268 case cmd == "plug" || cmd == "plugin" || cmd == "pin":
3269     gshCtx.xPlugin(argv[1:])
3270 case cmd == "print" || cmd == "-pr":
3271     // output internal slice // also sprintf should be
3272     gshCtx.Printv(argv)
3273 case cmd == "ps":
3274     gshCtx.xPs(argv)
3275 case cmd == "pstitle":
3276     // to be gsh.title
3277 case cmd == "rexeed" || cmd == "rexd":
3278     gshCtx.RexeServer(argv)
3279 case cmd == "rexe" || cmd == "rex":
3280     gshCtx.RexeClient(argv)
3281 case cmd == "repeat" || cmd == "rep": // repeat cond command
3282     gshCtx.repeat(argv)
3283 case cmd == "replay":
3284     gshCtx.xReplay(argv)
3285 case cmd == "scan":
3286     // scan input (or so in fscanf) to internal slice (like Files or map)
3287     gshCtx.Scanv(argv)
3288 case cmd == "set":
3289     // set name ...
3290 case cmd == "serv":
3291     gshCtx.httpServer(argv)
3292 case cmd == "shift":
3293     gshCtx.Shiftv(argv)
3294 case cmd == "sleep":
3295     gshCtx.sleep(argv)
3296 case cmd == "-sort":
3297     gshCtx.Sortv(argv)
3298
3299 case cmd == "j" || cmd == "join":
3300     gshCtx.Rjoin(argv)
3301 case cmd == "a" || cmd == "alpa":
3302     gshCtx.Rexe(argv)
3303 case cmd == "jcd" || cmd == "jchdir":
3304     gshCtx.Rchdir(argv)
3305 case cmd == "jget":
3306     gshCtx.Rget(argv)
3307 case cmd == "jls":
3308     gshCtx.Rls(argv)
3309 case cmd == "jput":
3310     gshCtx.Rput(argv)
3311 case cmd == "jpwd":
3312     gshCtx.Rpwd(argv)
3313
3314 case cmd == "time":
3315     fin = gshCtx.xTime(argv)
3316 case cmd == "ungets":
3317     if 1 < len(argv) {
3318         ungets(argv[1]+\n")
3319     }else{
3320     }
3321 case cmd == "pwd":
3322     gshCtx.xPwd(argv);
3323 case cmd == "ver" || cmd == "-ver" || cmd == "version":
3324     gshCtx.showVersion(argv)
3325 case cmd == "where":
3326     // data file or so?
3327 case cmd == "which":
3328     which("PATH",argv);
3329 case cmd == "gj" && 1 < len(argv) && argv[1] == "listen":
3330     go gj_server(argv[1:]);
3331 case cmd == "gj" && 1 < len(argv) && argv[1] == "serve":
3332     go gj_server(argv[1:]);
3333 case cmd == "gj" && 1 < len(argv) && argv[1] == "join":
3334     go gj_client(argv[1:]);
3335 case cmd == "gj":
3336     jsend(argv);
3337 case cmd == "jsend":
3338     jsend(argv);
3339 default:
3340     if gshCtx.whichPlugin(cmd,[]string{"-s"}) != nil {
3341         gshCtx.xPlugin(argv)
3342     }else{
3343         notfound,_ := gshCtx.excommand(false,argv)
3344         if notfound {
3345             fmt.Printf("--E-- command not found (%v)\n",cmd)
3346         }
3347     }
3348 }

```

```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

```

4093     fmt.Printf("--De-- putEvent[%v] (%3v) (%v %v)\n",EventSendFd,event,n,err)
4094 }
4095 }
4096 func ungets(str string){
4097     for _,ch := range str {
4098         putEvent(int(ch),0)
4099     }
4100 }
4101 func (gsh*GshContext)xReplay(argv[]string){
4102     hix := 0
4103     tempo := 1.0
4104     xtempo := 1.0
4105     repeat := 1
4106
4107     for _,a := range argv { // tempo
4108         if strBegins(a,"x") {
4109             fmt.Sscanf(a[1:],"%f",&xtempo)
4110             tempo = 1 / xtempo
4111             //fprintf(stderr,"--Dr-- tempo=[%v]%v\n",a[2:],tempo);
4112         }else
4113         if strBegins(a,"r") { // repeat
4114             fmt.Sscanf(a[1:],"%v",&repeat)
4115         }else
4116         if strBegins(a,"!") {
4117             fmt.Sscanf(a[1:],"%d",&hix)
4118         }else{
4119             fmt.Sscanf(a,"%d",&hix)
4120         }
4121     }
4122     if hix == 0 || len(argv) <= 1 {
4123         hix = len(gsh.CommandHistory)-1
4124     }
4125     fmt.Printf("--Ir-- Replay(!%v x%v r%v)\n",hix,xtempo,repeat)
4126     //dumpEvents(hix)
4127     //gsh.xScanReplay(hix,false,repeat,tempo,argv)
4128     go gsh.xScanReplay(hix,true,repeat,tempo,argv)
4129 }
4130
4131 // <a href="https://golang.org/pkg/syscall/#FdSet">syscall.Select</a>
4132 // 2020-0827 GShell-0.2.3
4133 /*
4134 func FpollIn1(fp *os.File,usec int)(uintptr){
4135     nfd := 1
4136
4137     rdv := syscall.FdSet {}
4138     fd1 = fp.Fd()
4139     bank1 := fd1/32
4140     mask1 := int32(1 << fd1)
4141     rdv.Bits[bank1] = mask1
4142
4143     fd2 := -1
4144     bank2 := -1
4145     var mask2 int32 = 0
4146
4147     if 0 <= EventRecvFd {
4148         fd2 = EventRecvFd
4149         nfd = fd2 + 1
4150         bank2 = fd2/32
4151         mask2 = int32(1 << fd2)
4152         rdv.Bits[bank2] |= mask2
4153         //fmt.Printf("--De-- EventPoll mask added [%d][%v][%v]\n",fd2,bank2,mask2)
4154     }
4155
4156     tout := syscall.NsecToTimeval(int64(usec*1000))
4157     //n,err := syscall.Select(nfd,&rdv,nil,nil,&tout) // spec. mismatch
4158     err := syscall.Select(nfd,&rdv,nil,nil,&tout)
4159     if err != nil {
4160         //fmt.Printf("--De-- select() err(%v)\n",err)
4161     }
4162     if err == nil {
4163         if 0 <= fd2 && (rdv.Bits[bank2] & mask2) != 0 {
4164             if false {
4165                 fmt.Printf("--De-- got Event\n")
4166             }
4167             return uintptr(EventFdOffset + fd2)
4168         }else
4169         if (rdv.Bits[bank1] & mask1) != 0 {
4170             return uintptr(NormalFdOffset + fd1)
4171         }else{
4172             return 1
4173         }
4174     }else{
4175         return 0
4176     }
4177 }
4178 */
4179 func fgetcTimeout1(fp *os.File,usec int)(int){
4180     READ1:
4181     //readyFd := FpollIn1(fp,usec)
4182     readyFd := CFpollIn1(fp,usec)
4183     if readyFd < 100 {
4184         return EV_TIMEOUT
4185     }
4186
4187     var buf [1]byte
4188
4189     if EventFdOffset <= readyFd {
4190         fd := int(readyFd-EventFdOffset)
4191         _,err := syscall.Read(fd,buf[0:1])
4192         if( err != nil ){
4193             return EOF;
4194         }else{
4195             if buf[0] == EV_MODE {
4196                 recvEvent(fd)
4197                 goto READ1
4198             }
4199             return int(buf[0])
4200         }
4201     }
4202
4203     _,err := fp.Read(buf[0:1])
4204     if( err != nil ){
4205         return EOF;
4206     }else{
4207         return int(buf[0])
4208     }
4209 }
4210
4211 func visibleChar(ch int)(string){
4212     switch {
4213     case '!' <= ch && ch <= '~':
4214         return string(ch)
4215     }
4216     switch ch {

```

```

4217     case ' ': return "\\s"
4218     case '\n': return "\\n"
4219     case '\r': return "\\r"
4220     case '\t': return "\\t"
4221 }
4222 switch ch {
4223     case 0x00: return "NUL"
4224     case 0x07: return "BEL"
4225     case 0x08: return "BS"
4226     case 0x0E: return "SO"
4227     case 0x0F: return "SI"
4228     case 0x1B: return "ESC"
4229     case 0x7F: return "DEL"
4230 }
4231 switch ch {
4232     case EV_IDLE: return fmt.Sprintf("IDLE")
4233     case EV_MODE: return fmt.Sprintf("MODE")
4234 }
4235 return fmt.Sprintf("%X",ch)
4236 }
4237 func recvEvent(fd int){
4238     var buf = make([]byte,1)
4239     _,_ = syscall.Read(fd,buf[0:1])
4240     if( buf[0] != 0 ){
4241         romkanmode = true
4242     }else{
4243         romkanmode = false
4244     }
4245 }
4246 func (gsh*GshContext)xScanReplay(hix int,replay bool,repeat int,tempo float64,argv[]string){
4247     var Start time.Time
4248     var events = []Event{}
4249     for _,e := range Events {
4250         if hix == 0 || e.CmdIndex == hix {
4251             events = append(events,e)
4252         }
4253     }
4254     elen := len(events)
4255     if 0 < elen {
4256         if events[elen-1].event == EV_IDLE {
4257             events = events[0:elen-1]
4258         }
4259     }
4260     for r := 0; r < repeat; r++ {
4261         for i,e := range events {
4262             nano := e.when.Nanosecond()
4263             micro := nano / 1000
4264             if Start.Second() == 0 {
4265                 Start = time.Now()
4266             }
4267             diff := time.Now().Sub(Start)
4268             if replay {
4269                 if e.event != EV_IDLE {
4270                     putEvent(e.event,0)
4271                     if e.event == EV_MODE { // event with arg
4272                         putEvent(int(e.evarg),0)
4273                     }
4274                 }
4275             }else{
4276                 fmt.Printf("%7.3fms #%-3v !%-3v [%v.%06d] %3v %02X %-4v %10.3fms\n",
4277                     float64(diff)/1000000.0,
4278                     i,
4279                     e.CmdIndex,
4280                     e.when.Format(time.Stamp),micro,
4281                     e.event,e.event,visibleChar(e.event),
4282                     float64(e.evarg)/1000000.0)
4283             }
4284             if e.event == EV_IDLE {
4285                 d := time.Duration(float64(time.Duration(e.evarg)) * tempo)
4286                 //nsleep(time.Duration(e.evarg))
4287                 nsleep(d)
4288             }
4289         }
4290     }
4291 }
4292 func dumpEvents(arg[]string){
4293     hix := 0
4294     if 1 < len(arg) {
4295         fmt.Sscanf(arg[1],"%d",&hix)
4296     }
4297     for i,e := range Events {
4298         nano := e.when.Nanosecond()
4299         micro := nano / 1000
4300         //if e.event != EV_TIMEOUT {
4301         if hix == 0 || e.CmdIndex == hix {
4302             fmt.Printf("#%-3v !%-3v [%v.%06d] %3v %02X %-4v %10.3fms\n",i,
4303                 e.CmdIndex,
4304                 e.when.Format(time.Stamp),micro,
4305                 e.event,e.event,visibleChar(e.event),float64(e.evarg)/1000000.0)
4306         }
4307         //}
4308     }
4309 }
4310 func fgetcTimeout(fp *os.File,usec int)(int){
4311     ch := fgetcTimeout1(fp,usec)
4312     if ch != EV_TIMEOUT {
4313         now := time.Now()
4314         if 0 < len(Events) {
4315             last := Events[len(Events)-1]
4316             dura := int64(now.Sub(last.when))
4317             Events = append(Events,Event{last.when,EV_IDLE,dura,last.CmdIndex})
4318         }
4319         Events = append(Events,Event{time.Now(),ch,0,CmdIndex})
4320     }
4321     return ch
4322 }
4323 }
4324 var AtConsoleLineTop = true
4325 var TtyMaxCol = 72 // to be obtained by ioctl?
4326 var EscTimeout = (100*1000)
4327 var (
4328     MODE_VicMode bool // vi compatible command mode
4329     MODE_ShowMode bool
4330     romkanmode bool // shown translation mode, the mode to be retained
4331     MODE_Recursive bool // recursive translation
4332     MODE_CapsLock bool // software CapsLock
4333     MODE_LowerLock bool // force lower-case character lock
4334     MODE_ViInsert int // visible insert mode, should be like "I" icon in X Window
4335     MODE_ViTrace bool // output newline before translation
4336 )
4337 type IInput struct {
4338     lno int
4339     lastlno int
4340     pch []int // input queue

```

```

4341 prompt string
4342 line string
4343 right string
4344 inMode bool
4345 pinMode bool
4346 waitingMeta string // waiting meta character
4347 lastCmd string
4348 }
4349 func (iin*IInput)Getc(timeoutUs int)(int){
4350 ch1 := EOF
4351 ch2 := EOF
4352 ch3 := EOF
4353 if( 0 < len(iin.pch) ){ // deQ
4354 ch1 = iin.pch[0]
4355 iin.pch = iin.pch[1:]
4356 }else{
4357 ch1 = fgetcTimeout(stdin,timeoutUs);
4358 }
4359 if( ch1 == 033 ){ // escape sequence
4360 ch2 = fgetcTimeout(stdin,EscTimeout);
4361 if( ch2 == EV_TIMEOUT ){
4362 }else{
4363 ch3 = fgetcTimeout(stdin,EscTimeout);
4364 if( ch3 == EV_TIMEOUT ){
4365 iin.pch = append(iin.pch,ch2) // enQ
4366 }else{
4367 switch( ch2 ){
4368 default:
4369 iin.pch = append(iin.pch,ch2) // enQ
4370 iin.pch = append(iin.pch,ch3) // enQ
4371 case '[':
4372 switch( ch3 ){
4373 case 'A': ch1 = GO_UP; // ^
4374 case 'B': ch1 = GO_DOWN; // v
4375 case 'C': ch1 = GO_RIGHT; // >
4376 case 'D': ch1 = GO_LEFT; // <
4377 case '3':
4378 ch4 := fgetcTimeout(stdin,EscTimeout);
4379 if( ch4 == '-' ){
4380 //fprintf(stderr,"%x%02X %02X %02X\n",ch1,ch2,ch3,ch4);
4381 ch1 = DEL_RIGHT
4382 }
4383 }
4384 case '\\':
4385 //ch4 := fgetcTimeout(stdin,EscTimeout);
4386 //fprintf(stderr,"%y%02X %02X %02X %02X\n",ch1,ch2,ch3,ch4);
4387 switch( ch3 ){
4388 case '-': ch1 = DEL_RIGHT
4389 }
4390 }
4391 }
4392 }
4393 }
4394 return ch1
4395 }
4396 func (iin*IInput)clearline(){
4397 var i int
4398 fprintf(stderr,"\r");
4399 // should be ANSI ESC sequence
4400 for i = 0; i < TtyMaxCol; i++ { // to the max. position in this input action
4401 fputc(' ',os.Stderr);
4402 }
4403 fprintf(stderr,"\r");
4404 }
4405 func (iin*IInput)Redraw(){
4406 redraw(iin,iin.lno,iin.line,iin.right)
4407 }
4408 func redraw(iin *IInput,lno int,line string,right string){
4409 inMeta := false
4410 showMode := ""
4411 showMeta := "" // visible Meta_mode on the cursor position
4412 showLino := fmt.Sprintf("%d1",lno)
4413 insertMark := "" // in visible insert mode
4414 }
4415 if MODE_VicMode {
4416 }else
4417 if 0 < len(iin.right) {
4418 insertMark = " "
4419 }
4420 }
4421 if( 0 < len(iin.waitingMeta) ){
4422 inMeta = true
4423 if iin.waitingMeta[0] != 033 {
4424 showMeta = iin.waitingMeta
4425 }
4426 }
4427 if( romkanmode ){
4428 //romkanmark = " *";
4429 }else{
4430 //romkanmark = "";
4431 }
4432 if MODE_ShowMode {
4433 romkan := "-";
4434 inmeta := "-";
4435 inveri := ""
4436 if MODE_CapsLock {
4437 inmeta = "A"
4438 }
4439 if MODE_LowerLock {
4440 inmeta = "a"
4441 }
4442 if MODE_ViTrace {
4443 inveri = "v"
4444 }
4445 if MODE_VicMode {
4446 inveri = ":"
4447 }
4448 if romkanmode {
4449 romkan = "\343\201\202"
4450 if MODE_CapsLock {
4451 inmeta = "R"
4452 }else{
4453 inmeta = "r"
4454 }
4455 }
4456 if inMeta {
4457 inmeta = "\\ "
4458 }
4459 showMode = "["+romkan+inmeta+inveri+"]";
4460 }
4461 Pre := "\r" + showMode + showLino
4462 Output := ""
4463 Left := ""
4464 Right := ""

```

```

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

```

```

4589 // <a href="https://golang.org/pkg/unicode/utf8/#DecodeRuneInString">DecodeRuneInString</a>
4590 func (iin*Input)GotoTOPW(){
4591     str := iin.line
4592     i := len(str)
4593     if i <= 0 {
4594         return
4595     }
4596     //i0 := i
4597     i -= 1
4598     lastSize := 0
4599     var lastRune rune
4600     var found = -1
4601     for 0 < i { // skip preamble spaces
4602         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4603         if !isAlnum(lastRune) { // character, type, or string to be searched
4604             i -= lastSize
4605             continue
4606         }
4607         break
4608     }
4609     for 0 < i {
4610         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4611         if lastSize <= 0 { continue } // not the character top
4612         if !isAlnum(lastRune) { // character, type, or string to be searched
4613             found = i
4614             break
4615         }
4616         i -= lastSize
4617     }
4618     if found < 0 && i == 0 {
4619         found = 0
4620     }
4621     if 0 <= found {
4622         if isAlnum(lastRune) { // or non-kana character
4623             }else{ // when positioning to the top o the word
4624                 i += lastSize
4625             }
4626             iin.right = str[i:] + iin.right
4627             if 0 < i {
4628                 iin.line = str[0:i]
4629             }else{
4630                 iin.line = ""
4631             }
4632         }
4633         //fmt.Printf("\n(%d,%d,%d)[%s][%s]\n",i0,i,found,iin.line,iin.right)
4634         //fmt.Printf("") // set debug messae at the end of line
4635     }
4636     // 0.2.8 2020-0901 created
4637     func (iin*Input)GotoENDW(){
4638         str := iin.right
4639         if len(str) <= 0 {
4640             return
4641         }
4642         lastSize := 0
4643         var lastRune rune
4644         var lastW = 0
4645         i := 0
4646         inWord := false
4647         lastRune,lastSize = utf8.DecodeRuneInString(str[0:])
4648         if isAlnum(lastRune) {
4649             r,z := utf8.DecodeRuneInString(str[lastSize:])
4650             if 0 < z && isAlnum(r) {
4651                 inWord = true
4652             }
4653         }
4654         for i < len(str) {
4655             lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4656             if lastSize <= 0 { break } // broken data?
4657             if !isAlnum(lastRune) { // character, type, or string to be searched
4658                 break
4659             }
4660             lastW = i // the last alnum if in alnum word
4661             i += lastSize
4662         }
4663         if inWord {
4664             goto DISP
4665         }
4666         for i < len(str) {
4667             lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4668             if lastSize <= 0 { break } // broken data?
4669             if isAlnum(lastRune) { // character, type, or string to be searched
4670                 break
4671             }
4672             i += lastSize
4673         }
4674         for i < len(str) {
4675             lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4676             if lastSize <= 0 { break } // broken data?
4677             if !isAlnum(lastRune) { // character, type, or string to be searched
4678                 break
4679             }
4680             lastW = i
4681             i += lastSize
4682         }
4683     }
4684     DISP:
4685     if 0 < lastW {
4686         iin.line = iin.line + str[0:lastW]
4687         iin.right = str[lastW:]
4688     }
4689     //fmt.Printf("\n(%d)[%s][%s]\n",i,iin.line,iin.right)
4690     //fmt.Printf("") // set debug messae at the end of line
4691 }
4692 // 0.2.8 2020-0901 created
4693 func (iin*Input)GotoNEXTW(){
4694     str := iin.right
4695     if len(str) <= 0 {
4696         return
4697     }
4698     lastSize := 0
4699     var lastRune rune
4700     var found = -1
4701     i := 1
4702     for i < len(str) {
4703         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4704         if lastSize <= 0 { break } // broken data?
4705         if !isAlnum(lastRune) { // character, type, or string to be searched
4706             found = i
4707             break
4708         }
4709         i += lastSize
4710     }
4711     if 0 < found {
4712         if isAlnum(lastRune) { // or non-kana character

```

```

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

```

```

4837     found := false
4838     if 0 < len(iin.right) {
4839         found = iin.SearchForward(pat)
4840     }
4841     if !found {
4842         found, line := gsh.SearchHistory(pat, true)
4843         if found {
4844             iin.line = line
4845             iin.right = ""
4846         }
4847     }
4848 }
4849 func (iin*IInput)GotoBACKSTR(pat string, gsh*GshContext){
4850     found := false
4851     if 0 < len(iin.line) {
4852         found = iin.SearchBackward(pat)
4853     }
4854     if !found {
4855         found, line := gsh.SearchHistory(pat, false)
4856         if found {
4857             iin.line = line
4858             iin.right = ""
4859         }
4860     }
4861 }
4862 func (iin*IInput)getString1(prompt string)(string){ // should be editable
4863     iin.clearline();
4864     fprintf(stderr, "\r%v", prompt)
4865     str := ""
4866     for {
4867         ch := iin.Getc(10*1000*1000)
4868         if ch == '\n' || ch == '\r' {
4869             break
4870         }
4871         sch := string(ch)
4872         str += sch
4873         fprintf(stderr, "%s", sch)
4874     }
4875     return str
4876 }
4877
4878 // search pattern must be an array and selectable with ^N/^P
4879 var SearchPat = ""
4880 var SearchForw = true
4881
4882 func (iin*IInput)xgetline1(prevline string, gsh*GshContext)(string){
4883     var ch int;
4884
4885     MODE_ShowMode = false
4886     MODE_VicMode = false
4887     iin.Redraw();
4888     first := true
4889
4890     for cix := 0; ; cix++ {
4891         iin.pinJmode = iin.inJmode
4892         iin.inJmode = false
4893
4894         ch = iin.Getc(1000*1000)
4895
4896         if ch != EV_TIMEOUT && first {
4897             first = false
4898             mode := 0
4899             if romkanmode {
4900                 mode = 1
4901             }
4902             now := time.Now()
4903             Events = append(Events, Event{now, EV_MODE, int64(mode), CmdIndex})
4904         }
4905         if ch == 033 {
4906             MODE_ShowMode = true
4907             MODE_VicMode = !MODE_VicMode
4908             iin.Redraw();
4909             continue
4910         }
4911         if MODE_VicMode {
4912             switch ch {
4913                 case '0': ch = GO_TOPL
4914                 case '$': ch = GO_ENDL
4915                 case 'b': ch = GO_TOPW
4916                 case 'e': ch = GO_ENDW
4917                 case 'w': ch = GO_NEXTW
4918                 case '^': ch = GO_PAIRCH
4919
4920                 case 'j': ch = GO_DOWN
4921                 case 'k': ch = GO_UP
4922                 case 'h': ch = GO_LEFT
4923                 case 'l': ch = GO_RIGHT
4924                 case 'x': ch = DEL_RIGHT
4925                 case 'a': MODE_VicMode = !MODE_VicMode
4926                     ch = GO_RIGHT
4927                 case 'i': MODE_VicMode = !MODE_VicMode
4928                     iin.Redraw();
4929                     continue
4930                 case '-':
4931                     right, head := delHeadChar(iin.right)
4932                     if len([]byte(head)) == 1 {
4933                         ch = int(head[0])
4934                         if( 'a' <= ch && ch <= 'z' ){
4935                             ch = ch + 'A'-'a'
4936                         }else
4937                         if( 'A' <= ch && ch <= 'Z' ){
4938                             ch = ch + 'a'-'A'
4939                         }
4940                         iin.right = string(ch) + right
4941                     }
4942                     iin.Redraw();
4943                     continue
4944                 case '^': // GO_FORWCH
4945                     iin.Redraw();
4946                     ch = iin.Getc(3*1000*1000)
4947                     if ch == EV_TIMEOUT {
4948                         iin.Redraw();
4949                         continue
4950                     }
4951                     SearchPat = string(ch)
4952                     SearchForw = true
4953                     iin.GotoFORWSTR(SearchPat, gsh)
4954                     iin.Redraw();
4955                     continue
4956                 case '/':
4957                     SearchPat = iin.getString1("/") // should be editable
4958                     SearchForw = true
4959                     iin.GotoFORWSTR(SearchPat, gsh)
4960                     iin.Redraw();

```



```

4961         continue
4962     case '?':
4963         SearchPat = iin.getstringl("?") // should be editable
4964         SearchForw = false
4965         iin.GotoBACKSTR(SearchPat,gsh)
4966         iin.Redraw();
4967         continue
4968     case 'n':
4969         if SearchForw {
4970             iin.GotoFORWSTR(SearchPat,gsh)
4971         }else{
4972             iin.GotoBACKSTR(SearchPat,gsh)
4973         }
4974         iin.Redraw();
4975         continue
4976     case 'N':
4977         if !SearchForw {
4978             iin.GotoFORWSTR(SearchPat,gsh)
4979         }else{
4980             iin.GotoBACKSTR(SearchPat,gsh)
4981         }
4982         iin.Redraw();
4983         continue
4984     }
4985 }
4986 switch ch {
4987 case GO_TOPW:
4988     iin.GotoTOPW()
4989     iin.Redraw();
4990     continue
4991 case GO_ENDW:
4992     iin.GotoENDW()
4993     iin.Redraw();
4994     continue
4995 case GO_NEXTW:
4996     // to next space then
4997     iin.GotoNEXTW()
4998     iin.Redraw();
4999     continue
5000 case GO_PAIRCH:
5001     iin.GotoPAIRCH()
5002     iin.Redraw();
5003     continue
5004 }
5005
5006 //fprintf(stderr,"A[%02X]\n",ch);
5007 if( ch == '\\ ' || ch == 033 ){
5008     MODE_ShowMode = true
5009     metach := ch
5010     iin.waitingMeta = string(ch)
5011     iin.Redraw();
5012     // set cursor //fprintf(stderr,"???\b\b\b")
5013     ch = fgetcTimeout(stdin,2000*1000)
5014     // reset cursor
5015     iin.waitingMeta = ""
5016
5017     cmdch := ch
5018     if( ch == EV_TIMEOUT ){
5019         if metach == 033 {
5020             continue
5021         }
5022         ch = metach
5023     }else
5024     /*
5025     if( ch == 'm' || ch == 'M' ){
5026         mch := fgetcTimeout(stdin,1000*1000)
5027         if mch == 'r' {
5028             romkanmode = true
5029         }else{
5030             romkanmode = false
5031         }
5032         continue
5033     }else
5034     /*
5035     if( ch == 'k' || ch == 'K' ){
5036         MODE_Recursive = !MODE_Recursive
5037         iin.Translate(cmdch);
5038         continue
5039     }else
5040     if( ch == 'j' || ch == 'J' ){
5041         iin.Translate(cmdch);
5042         continue
5043     }else
5044     if( ch == 'i' || ch == 'I' ){
5045         iin.Replace(cmdch);
5046         continue
5047     }else
5048     if( ch == 'l' || ch == 'L' ){
5049         MODE_LowerLock = !MODE_LowerLock
5050         MODE_CapsLock = false
5051         if MODE_ViTrace {
5052             fprintf(stderr,"%v\r\n",string(cmdch));
5053         }
5054         iin.Redraw();
5055         continue
5056     }else
5057     if( ch == 'u' || ch == 'U' ){
5058         MODE_CapsLock = !MODE_CapsLock
5059         MODE_LowerLock = false
5060         if MODE_ViTrace {
5061             fprintf(stderr,"%v\r\n",string(cmdch));
5062         }
5063         iin.Redraw();
5064         continue
5065     }else
5066     if( ch == 'v' || ch == 'V' ){
5067         MODE_ViTrace = !MODE_ViTrace
5068         if MODE_ViTrace {
5069             fprintf(stderr,"%v\r\n",string(cmdch));
5070         }
5071         iin.Redraw();
5072         continue
5073     }else
5074     if( ch == 'c' || ch == 'C' ){
5075         if 0 < len(iin.line) {
5076             xline,tail := delTailChar(iin.line)
5077             if len([]byte(tail)) == 1 {
5078                 ch = int(tail[0])
5079                 if( 'a' <= ch && ch <= 'z' ){
5080                     ch = ch + 'A'-'a'
5081                 }else
5082                 if( 'A' <= ch && ch <= 'Z' ){
5083                     ch = ch + 'a'-'A'
5084                 }

```

```

5085         iin.line = xline + string(ch)
5086     }
5087     }
5088     if MODE_ViTrace {
5089         fprintf(stderr, "%v\r\n", string(cmdch));
5090     }
5091     iin.Redraw();
5092     continue;
5093 }else{
5094     iin.pch = append(iin.pch, ch) // push
5095     ch = '\\'
5096 }
5097 }
5098 switch( ch ){
5099     case 'P'-0x40: ch = GO_UP
5100     case 'N'-0x40: ch = GO_DOWN
5101     case 'B'-0x40: ch = GO_LEFT
5102     case 'F'-0x40: ch = GO_RIGHT
5103 }
5104 //fprintf(stderr, "B[%02X]\n", ch);
5105 switch( ch ){
5106     case 0:
5107         continue;
5108     case '\t':
5109         iin.Replace('j');
5110         continue
5111     case 'X'-0x40:
5112         iin.Replace('j');
5113         continue
5114     case EV_TIMEOUT:
5115         iin.Redraw();
5116         if iin.pinJmode {
5117             fprintf(stderr, "\\j\r\n")
5118             iin.inJmode = true
5119         }
5120         continue
5121     case GO_UP:
5122         if iin.lno == 1 {
5123             continue
5124         }
5125         cmd,ok := gsh.cmdStringInHistory(iin.lno-1)
5126         if ok {
5127             iin.line = cmd
5128             iin.right = ""
5129             iin.lno = iin.lno - 1
5130         }
5131         iin.Redraw();
5132         continue
5133     case GO_DOWN:
5134         cmd,ok := gsh.cmdStringInHistory(iin.lno+1)
5135         if ok {
5136             iin.line = cmd
5137             iin.right = ""
5138             iin.lno = iin.lno + 1
5139         }else{
5140             iin.line = ""
5141             iin.right = ""
5142             if iin.lno == iin.lastlno-1 {
5143                 iin.lno = iin.lno + 1
5144             }
5145         }
5146         iin.Redraw();
5147         continue
5148     case GO_LEFT:
5149         if 0 < len(iin.line) {
5150             xline,tail := delTailChar(iin.line)
5151             iin.line = xline
5152             iin.right = tail + iin.right
5153         }
5154         iin.Redraw();
5155         continue;
5156     case GO_RIGHT:
5157         if( 0 < len(iin.right) && iin.right[0] != 0 ){
5158             xright,head := delHeadChar(iin.right)
5159             iin.right = xright
5160             iin.line += head
5161         }
5162         iin.Redraw();
5163         continue;
5164     case EOF:
5165         goto EXIT;
5166     case 'R'-0x40: // replace
5167         dst := convs(iin.line+iin.right);
5168         iin.line = dst
5169         iin.right = ""
5170         iin.Redraw();
5171         continue;
5172     case 'n'-0x40: // just show the result
5173         readDic();
5174         romkanmode = !romkanmode;
5175         iin.Redraw();
5176         continue;
5177     case 'L'-0x40:
5178         iin.Redraw();
5179         continue
5180     case 'K'-0x40:
5181         iin.right = ""
5182         iin.Redraw();
5183         continue
5184     case 'E'-0x40:
5185         iin.line += iin.right
5186         iin.right = ""
5187         iin.Redraw();
5188         continue
5189     case 'A'-0x40:
5190         iin.right = iin.line + iin.right
5191         iin.line = ""
5192         iin.Redraw();
5193         continue
5194     case 'U'-0x40:
5195         iin.line = ""
5196         iin.right = ""
5197         iin.clearline();
5198         iin.Redraw();
5199         continue;
5200     case DEL_RIGHT:
5201         if( 0 < len(iin.right) ){
5202             iin.right,_ = delHeadChar(iin.right)
5203             iin.Redraw();
5204         }
5205         continue;
5206     case 0x7F: // BS? not DEL

```

```

5209         if( 0 < len(iin.line) ){
5210             iin.line,_ = delTailChar(iin.line)
5211             iin.Redraw();
5212         }
5213         /*
5214         else
5215         if( 0 < len(iin.right) ){
5216             iin.right,_ = delHeadChar(iin.right)
5217             iin.Redraw();
5218         }
5219         */
5220         continue;
5221     case 'H'-0x40:
5222         if( 0 < len(iin.line) ){
5223             iin.line,_ = delTailChar(iin.line)
5224             iin.Redraw();
5225         }
5226         continue;
5227     }
5228     if( ch == '\n' || ch == '\r' ){
5229         iin.line += iin.right;
5230         iin.right = ""
5231         iin.Redraw();
5232         fputc(ch,stderr);
5233         AtConsoleLineTop = true
5234         break;
5235     }
5236     if MODE_CapsLock {
5237         if 'a' <= ch && ch <= 'z' {
5238             ch = ch+'A'-'a'
5239         }
5240     }
5241     if MODE_LowerLock {
5242         if 'A' <= ch && ch <= 'Z' {
5243             ch = ch+'a'-'A'
5244         }
5245     }
5246     iin.line += string(ch);
5247     iin.Redraw();
5248 }
5249 EXIT:
5250 return iin.line + iin.right;
5251 }
5252
5253 func getline_main(){
5254     line := xgetline(0,"",nil)
5255     fprintf(stderr,"%s\n",line);
5256     /*
5257     dp = strpbrk(line,"\r\n");
5258     if( dp != NULL ){
5259         *dp = 0;
5260     }
5261
5262     if( 0 ){
5263         fprintf(stderr,"\n%d\n",int(strlen(line)));
5264     }
5265     if( lseek(3,0,0) == 0 ){
5266         if( romkanmode ){
5267             var buf [8*1024]byte;
5268             convs(line,buf);
5269             strcpy(line,buf);
5270         }
5271         write(3,line,strlen(line));
5272         ftruncate(3,lseek(3,0,SEEK_CUR));
5273         //fprintf(stderr,"outsize=%d\n", (int)lseek(3,0,SEEK_END));
5274         lseek(3,0,SEEK_SET);
5275         close(3);
5276     }else{
5277         fprintf(stderr,"\r\ngetline: ");
5278         trans(line);
5279         //printf("%s\n",line);
5280         printf("\n");
5281     }
5282     */
5283 }
5284 //== end ====== getline
5285
5286 //
5287 // $USERHOME/.gsh/
5288 //     gsh-rc.txt, or gsh-configure.txt
5289 //     gsh-history.txt
5290 //     gsh-aliases.txt // should be conditional?
5291 //
5292 func (gshCtx *GshContext)gshSetupHomedir()(bool) {
5293     homedir,found := userHomeDir()
5294     if !found {
5295         fmt.Printf("--E-- You have no UserHomeDir\n")
5296         return true
5297     }
5298     gshhome := homedir + "/" + GSH_HOME
5299     _, err2 := os.Stat(gshhome)
5300     if err2 != nil {
5301         err3 := os.Mkdir(gshhome,0700)
5302         if err3 != nil {
5303             fmt.Printf("--E-- Could not Create %s (%s)\n",
5304                 gshhome,err3)
5305             return true
5306         }
5307         fmt.Printf("--I-- Created %s\n",gshhome)
5308     }
5309     gshCtx.GshHomeDir = gshhome
5310     return false
5311 }
5312 func setupGshContext()(GshContext,bool){
5313     gshPA := syscall.ProcAttr {
5314         "", // the starting directory
5315         os.Environ(), // environ[]
5316         []uintptr{os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd()},
5317         nil, // OS specific
5318     }
5319     cwd,_ := os.Getwd()
5320     gshCtx := GshContext {
5321         cwd, // StartDir
5322         "", // GetLine
5323         []GchdirHistory { {cwd,time.Now(),0} }, // ChdirHistory
5324         gshPA,
5325         []GCommandHistory(), //something for invokation?
5326         GCommandHistory(), // CmdCurrent
5327         false,
5328         []int{},
5329         syscall.Rusage{},
5330         "", // GshHomeDir
5331         Ttyid(),
5332         false,

```

```

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

```



```

5581 "77yX77yScM5hbmFqdXVuaVgJ77yX77yScuS4g+WNgeS6jHgJnzIKa29idW5uCeWai+WIhgp0"+
5582 "aWthcmFxCe0Boe0Bi+OC1Qp0aWthcmEJ5YqbCmNoaWthcmEJ5YqbCjvvdGV4dGFyZWE+CG="
5583 //</span>
5584
5585 var JA_JKLDic = //<span id="gsh-ja-jkl-dic">
5586 "data:text/dic;base64, "+
5587 "Iy92Z2JscU15SUIFamRpy2ptb3ZzZwpKQWpKS0wMjAyMGowODE5KS5KSheLV4pL1NhdG94SVRT"+
5588 "CmtqamprbGtqa2tsa2psIOS4lueVjApqamgtamwJ44GCmtqbAnjgYQKa2tqbAnjgYYKamtq"+
5589 "amwJ44GCmtqa2trbAnjgYoKa2pra2wJ44GLCmpramrbAnjgY0Ka2tramwJ44GCMpramps"+
5590 "Ce0BkQpampqbAnjgZMKamtqa2psCe0B1Qpamtqa2wJ44GCMpamtqbAnjgZKka2pqamts"+
5591 "Ce0BmpqamprbAnjgZ0KamtsCe0Bnwpra2prbAnjgEka2pqa2wJ44GCMtqa2pqbAnjgAYK"+
5592 "a2tqa2tsCe0BqpramtsCe0Bggpqa2prbAnjgKa2tra2wJ44GCMpqa2psCe0BrQpra2pq"+
5593 "bAnjga4Kamtra2wJ44GvCmpqa2tqbAnjgIKampra2wJ44G1CmtsCe0BuApga2tsCe0Buwpg"+
5594 "a2tqbAnjg4Ka2tqa2psCe0BvwpqbAnjgAKamtra2psCe0CgQpqa2tqa2wJ44KCCmtqamwJ"+
5595 "44KCMpra2pqbAnjgYKampsCe0CiApra2tsCe0CiQpamtsCe0CiQpqa2pqa2wJ44KLCmpq"+
5596 "amwJ44KCCmtqa2psCe0CjQpqa2psCe0CjwpramramwJ44KQCMtqamrbAnjgEka2pqaamwJ"+
5597 "44KCCmtqa2prbAnjgpMka2pqa2psCe0DvApra2wJ44KbCmtrampbrbAnjgpwKa2pramtqbAnj"+
5598 "gLEK";
5599 //</span>
5600
5601 //</span>
5602 /*
5603 <style id="gsh-references-style">
5604 #references details { font-family:Georgia; }
5605 #references a { font-family:Georgia; }
5606 .wrap { white-space:normal; }
5607 </style>
5608 <details id="references"><summary>References</summary><div class="gsh-src">
5609 Web technology
5610 <a href="https://html.spec.whatwg.org">HTML: The Living Standard</a> (September 2020)
5611 <a href="https://html.spec.whatwg.org/dev/">Developer Version</a>
5612
5613 <a href="https://drafts.csswg.org">CSS Working Group Editor Drafts</a> (September 2020)
5614
5615 <a href="https://developer.mozilla.org/ja/docs/Web">MDN web docs</a>
5616 <a href="https://developer.mozilla.org/ja/docs/Web/HTML/Element">HTML</a>
5617 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS">CSS</a> : <span class="wrap">
5618 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/Selectors">selectors</a>
5619 <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/background-repeat">repeat</a></span>
5620 <a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript">JavaScript</a>
5621 <a href="https://developer.mozilla.org/en-US/docs/Web/HTTP">HTTP</a>
5622
5623 Go language (August 2020 / Go 1.15)
5624 <a href="https://golang.org">The Go Programming Language</a>
5625 <a href="https://golang.org/pkg/">Packages</a>
5626 <a href="https://godoc.org/golang.org/x/net/websocket">WebSocket</a>
5627
5628 Stackoverflow
5629 <!--
5630 <iframe src="https://golang.org" width="100%" height="300"></iframe>
5631 -->
5632 </div></details>
5633 */
5634
5635 <details id="html-src" onclick="frame_open();"><summary>Raw Source</summary><div>
5636
5637 <!-- h2>The full of this HTML including the Go code is here.</h2 -->
5638 <details id="gsh-whole-view"><summary>Whole file</summary>
5639 <a name="whole-src-view"></a>
5640 <span id="src-frame"></span><!-- a window to show source code -->
5641 </details>
5642
5643 <details id="gsh-style-frame" onclick="fill_CSSView()"><summary>CSS part</summary>
5644 <a name="style-src-view"></a>
5645 <span id="gsh-style-view"></span>
5646 </details>
5647
5648 <details id="gsh-script-frame" onclick="fill_JavaScriptView()"><summary>JavaScript part</summary>
5649 <a name="script-src-view"></a>
5650 <span id="gsh-script-view"></span>
5651 </details>
5652
5653 <details id="gsh-data-frame" onclick="fill_DataView()"><summary>Builtin data part</summary>
5654 <a name="gsh-data-frame"></a>
5655 <span id="gsh-data-view"></span>
5656 </details>
5657
5658 </div></details>
5659 */
5660
5661 /*
5662 <div id="GshFooter0"></div>
5663 <!-- 2020-09-17 SatoxITS, visible script { -- >
5664 <details><summary>GJScript</summary>
5665 <style>.gjscript { font-family:Georgia; }</style>
5666 <pre id="gjscript_1" class="gjscript"> function gjtest1(){ alert('Hello GJScript!'); }
5667 gjtest1()
5668 </pre>
5669 <script>
5670 gjs = document.getElementById('gjscript_1');
5671 //eval(gjs.inerHTML);
5672 //gjs.outerHTML = "";
5673 </script>
5674 </details><!-- ----- END-OF-VISIBLE-PART ----- } -->
5675
5676 <!--
5677 // 2020-0906 added,
5678 https://developer.mozilla.org/en-US/docs/Web/CSS/z-index
5679 https://developer.mozilla.org/en-US/docs/Web/CSS/position
5680 -->
5681 <span id="GshGrid">(^_^)</small>(Hit j k l h)</small></span>
5682
5683 <span id="GStat"><br>
5684 </span>
5685 <span id="GMenu" onclick="GShellMenu(this)"></span>
5686 <span id="GTop"></span>
5687 <div id="GShellPlane" onclick="showGShellPlane();"></div>
5688 <div id="RawTextViewer"></div>
5689 <div id="RawTextViewerClose" onclick="hideRawTextViewer()"> CLOSE </div>
5690
5691 <style id="GshStyleDef">
5692 #LineNumbered table,tr,td {
5693 margin:0;
5694 padding:4px;
5695 spacing:0;
5696 border:12px;
5697 }
5698 textarea.LineNumber {
5699 font-size:12px;
5700 font-family:monospace,Courier New;
5701 color:#282;
5702 padding:4px;
5703 text-align:right;
5704 }

```

```

5705 textarea.LineNumbered {
5706     font-size:12px;
5707     font-family:monospace,Courier New;
5708     padding:4px;
5709     wrap:off;
5710 }
5711 #RawTextViewer{
5712     z-index:0;
5713     position:fixed; top:0px; left:0px;
5714     width:100%; xxxheight:50px; xheight:0px;
5715     overflow:auto;
5716     color:#fff; background-color:rgba(128,128,256,0.2);
5717     font-size:12px;
5718     spellcheck:false;
5719 }
5720 #RawTextViewerClose{
5721     z-index:0;
5722     position:fixed; top:-100px; left:-100px;
5723     color:#fff; background-color:rgba(128,128,256,0.2);
5724     font-size:20px; font-family:Georgia;
5725     white-space:pre;
5726 }
5727 #xxxGShellPlane{
5728     z-index:0;
5729     position:fixed; top:0px; left:0px;
5730     width:100%; height:50px;
5731     overflow:auto;
5732     color:#fff; background-color:rgba(128,128,256,0.3);
5733     font-size:12px;
5734 }
5735 #xxxGTop{
5736     z-index:9;
5737     opacity:1.0;
5738     position:fixed; top:0px; left:0px;
5739     width:320px; height:20px;
5740     color:#fff; background-color:rgba(32,32,160,0.15);
5741     color:#fff; font-size:12px;
5742 }
5743 #xxxGPos{
5744     z-index:12;
5745     position:fixed; top:0px; left:0px;
5746     opacity:1.0;
5747     width:640px; height:30px;
5748     color:#fff; background-color:rgba(0,0,0,0.2);
5749     color:#fff; font-size:12px;
5750 }
5751 #GMenu{
5752     z-index:2000;
5753     position:fixed; top:250px; left:0px;
5754     opacity:1.0;
5755     width:100px; height:100px;
5756     color:#fff;
5757     color:#fff; background-color:rgba(0,0,0,0.0);
5758     color:#fff; font-size:16px; font-family:Georgia;
5759     background-repeat:no-repeat;
5760 }
5761 #xxxGStat{
5762     z-index:8;
5763     xopacity:0.0;
5764     position:fixed; top:20px; left:0px;
5765     xwidth:640px;
5766     width:100%; height:90px;
5767     color:#fff; background-color:rgba(0,0,128,0.04);
5768     font-size:20px; font-family:Georgia;
5769 }
5770 #GLog{
5771     z-index:10;
5772     position:fixed; top:50px; left:0px;
5773     opacity:1.0;
5774     width:640px; height:60px;
5775     color:#fff; background-color:rgba(0,0,128,0.10);
5776     font-size:12px;
5777 }
5778 #GshGrid {
5779     z-index:11;
5780     xopacity:0.0;
5781     position:fixed; top:0px; left:0px;
5782     width:320px; height:30px;
5783     color:#9f9; font-size:16px;
5784 }
5785 xbody {display:none;}
5786 .gsh-link{color:green;}
5787 #gsh {border-width:1;margin:0;padding:0;}
5788 #gsh {font-family:monospace,Courier New;color:#ddf;font-size:8px;}
5789 #gsh header{height:100px;}
5790 #xgsh header{height:100px;background-image:url(GShell-Logo00.png);}
5791 #GshMenu{font-size:14pt;color:#c44;}
5792 .GshMenu{
5793     font-size:14pt;color:#2a2;padding:4px; text-align:right;
5794 }
5795 .GshMenu1:hover{
5796     font-size:14pt;color:#fff;font-weight:bold;background-color:#2a2;
5797 }
5798 #GshFooter{height:100px;background-size:80px;background-repeat:no-repeat;}
5799 #gsh note{color:#000;font-size:10pt;}
5800 #gsh h2{color:#24a;font-family:Georgia;font-size:18pt;}
5801 #gsh h3{color:#24a;font-family:Georgia;font-size:16pt;}
5802 #gsh details{color:#888;background-color:#fff;font-family:monospace;}
5803 #gsh summary{font-size:16pt;color:#fff;background-color:#8af;height:30px;}
5804 #gsh pre{font-size:11pt;color:#223;background-color:#ffffff;}
5805 #gsh a{color:#24a;}
5806 #gsh a{name}{color:#24a;font-size:16pt;}
5807 #gsh .gsh-src{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
5808 #gsh .gsh-src{background-color:#ffffff;color:#223;}
5809 #gsh-src-src{spellcheck:false}
5810 #SrcTextarea{background-color:#ffffff;color:#223;}
5811 #SrcTextarea{background-color:#ffffff;color:#223;}
5812 .gsh-code {white-space:pre;font-family:Courier New !important;}
5813 .gsh-code {color:#024;font-size:11pt; background-color:#ffffff;}
5814 .gsh-golang-data {display:none;}
5815 #gsh-WinId {color:#000;font-size:14pt;}
5816
5817 .gsh-document {font-size:11pt;background-color:#fff;font-family:Georgia;}
5818 .gsh-document {color:#000;background-color:#fff !important;}
5819 .gsh-document > h2{color:#000;background-color:#fff !important;}
5820 .gsh-document details{color:#000;background-color:#fff;font-family:Georgia;}
5821 .gsh-document p{max-width:550pt;color:#000;background-color:#fff;font-family:Georgia;}
5822 .gsh-document address{width:500pt;color:#000;background-color:#fff;font-family:Georgia;}
5823
5824 @media print {
5825     #gsh pre{font-size:11pt !important;}
5826 }
5827 </style>
5828

```



```

6077 AjzxxVwCLD+BYJ07RA6IHTuc8jclEpNNZ25q24P88xK09H9p55d2S3TmJngu8zUPTN+OL/PC\
6078 dc2P4UFahmsfn47H6RP12Vnwjzr25LuFlwSLBs0Yf72KosQJyIzN2fL00aGkG4U6b8+BxYQ\
6079 TkKwWenYgeupTgZ2fth6ghg26/jB8aPknob059jzZLh9L+084E59USUQhki65Vwg6P3jyDW\
6080 85ziR5001+qAbR67So+rzwbQxw2xR0caSSmQI/fcFY7LPdZ21Zz5Kk+C5dEL6ixTfWLA\
6081 V1/nm4/cmbCw+nXmNwee48EznelwaOf+EKyIrciIDOGPL3PavpZRGFPInhtCOXQ5CLgPQW\
6082 RGJ4fbl+LbuY3VncC8zF4Kvlseze2FVNV6s6qjsQv++Y0t29BUzqRnjgW2DI0bJWxzE18vIX\
6083 KEPL9f9dsBxp/2Xs6sgXXM3dfCLatfd8adBN1u0UNhLafwg6Bw93sevgj1HRMLPw/bl14a6npg\
6084 pksWlw06fYm+nV3MLU6MwfbD3KwyWstXwj2zUcu04jSj+6WUyJBTLLpSvlokE327S/NJwX\
6085 Mqj+21W6vtW1Ti4TY/bUnDxms7iu9baga2OYX5DbUX1z9BRpGevdrDHJ51k3m3z394Vgd5Yp\
6086 qZbnk1kQbbVhbtEhI61/0vu/2gszaeFLR+TNObCXY90Xa7q7BbtQ6tbuV/oYiPhu8xhzA4R7\
6087 o1Maou3Q4pYZWHWct1a17ndi3bXo2P7v2p70cmmEpycew84LQ6770Ev+ht2PnED+mNPy/W2\
6088 9LRATHr5EJ/vQ5fllw7StTREMd4gAu5rQ3T6aRSqdmx7z+/GB47ui290UW9J4XAnJ711IS\
6089 167+xi8smf6YcrRQxul4KlYsH6Be9110YHs79i/4cxbvgnH2jWB1j1XXxecYQuZ00g5WDLuq\
6090 Y6xMfg2Xrcb6wYrTjP5N07Zu3v9iRmdNn4F5eSbKohotab6stQot7/beUgSubPmhrC27B1\
6091 OYQhS1OJvclko4fxKoz+kgv+oajDVEsgVER9PehP+SrXwnkMNLm6VpQnUiKXzm+0PveQgf\
6092 h4F8U1j9Wort+64Stf500WEz2G5tCd/F2S/VXH3nagrQU1+4B2j8m8Ss/FmI9D3McBjwo\
6093 kRn3kXzuzqpszkZUicOcrjMPhWQ1aBxM1gsdui315d3J1R9ywoVm9Np1ktIE/COYIDtWZV2\
6094 8/KpQxkKkvc1bdvEDDtU0sc+RmxsD1pnxmw178tYMHZGdCt2fgznJ+8xzuSRBvQ4zrhEw9\
6095 H92688VJ50HFzRGI17AAPowk17LsplurBj0QpXybyb/8dmn2//11/gqnaGo2Awqf/38+WE1\
6096 14af+5Q5EXEMARKAo12CC2xvJNV+1Mz78LkH3V27LWVtzn9w4/+6JgdKxJPLgd7b1TADkw1p\
6097 n1iS+QSI+H1Ew5RPUvengY20d6mF7K0tloF1dj/1kuisatCEBEABEABEABEABEABEABEAB\
6098 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6099 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6100 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6101 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6102
6103 ITSmoreQR="data:image/png;base64,\
6104 iVBORw0KGgoAAAANSUHEUgAAAG8AAABvQAAMAAADYCWwjAAAAB1BMVEX///9BaeFHqDaJAAB\
6105 HkLEQVQ4jdTsa2EMawGYCMX7s1CkVgjXVACbe7CarASXda1lAWgS4HwM5zEVs+mvSgs+ZBQ\
6106 8gcb4Bdhzyvw8szMSAuBHNm+KAd4Qc8LDpDn8ogT4UpFGci2jI8IGF3eLwPwahnVyWecev\
6107 UEbDXa0X2anJueYDzNkLQassPckjC4nW3E1Sfwqk6jU/vAkPhg0AlSfVhe8Jt0dkwDMw\
6108 YMGSSuPyWhar19k0tkv2sb3sdw2rUCqW88g4RplA9s1JPv9cTPlNRD4XFkn8XaQC1w6Lzq\
6109 Z08dHw4+U2GzqlS8gbgVmkfr1N6YXK8OqLD00mlGTMvzPERA8AL9vnbOifpS0L33fsVyrTrL\
6110 S9wiqDzznhUI38v5n783/gBuUs2eLg1c8gAAAABRU5ErkJggg==";
6111
6112 </script>
6113
6114 <div id="GJFactory_1" class="xxxGJFactory" style=""></div>
6115 <!--
6116 https://developer.mozilla.org/en-US/docs/Web/CSS/line-height
6117 -->
6118 <style>
6119 .GJFactory{
6120   resize:both; overflow:scroll;
6121   position:static;
6122   border:1.2px dashed #282; xborder-radius:2px;
6123   margin:0px; padding:10px !important;
6124   width:340px; height:340px;
6125   flex-wrap: wrap;
6126   color:#fff; background-color:rgba(0,0,0,0.0);
6127   line-height:0.0;
6128   xxxcolor:#22a !important;
6129   text-shadow:2px 2px #ddf;
6130 }
6131 .GJFactory h1,h2,h3,h4 {
6132   xxxcolor:#22a !important;
6133 }
6134 xxxinput {
6135   border:1px dashed #0f0; border-radius:0px;
6136 }
6137 .GJWin:hover{
6138   color:#df8 !important;
6139   background-color:rgba(32,32,160,0.8) !important;
6140   line-height:0.0;
6141 }
6142 .GJWin:active{
6143   color:#df8 !important;
6144   background-color:rgba(224,32,32,0.8) !important;
6145   line-height:0.0;
6146 }
6147 .GJWin:focus{
6148   color:#df8 !important;
6149   background-color:rgba(32,32,32,1.0) !important;
6150   line-height:0.0;
6151 }
6152 .GJWin{
6153   z-index:10000;
6154   display:inline;
6155   position:relative;
6156   flex-wrap: wrap;
6157   top:0; left:0px;
6158   width:285px !important; height:205px !important;
6159   border:1px solid #eea; border-radius:2px;
6160   margin:0px; padding:0px;
6161   font-size:8pt;
6162   line-height:0.0;
6163   color:#fff; background-color:rgba(0,0,64,0.1) !important;
6164 }
6165 .GJTab{
6166   display:inline;
6167   position:relative;
6168   top:0px; left:0px;
6169   margin:0px; padding:2px;
6170   border:0px solid #000; border-radius:2px;
6171   width:90px; height:20px;
6172   font-family:Georgia;
6173   font-size:9pt;
6174   line-height:1.0;
6175   white-space:nowrap;
6176   color:#fff; background-color:rgba(0,0,64,0.7);
6177   text-align:center;
6178   vertical-align:middle;
6179 }
6180 .GJStat:focus{
6181   color:#df8 !important;
6182   background-color:rgba(32,32,32,1.0) !important;
6183   line-height:1.0;
6184 }
6185 .GJStat{
6186   display:inline;
6187   position:relative;
6188   top:0px; left:0px;
6189   margin:0px; padding:2px;
6190   border:0px solid #00f; border-radius:2px;
6191   width:166px; height:20px;
6192   font-family:monospace;
6193   font-size:9pt;
6194   line-height:1.0;
6195   color:#fff; background-color:rgba(0,0,64,0.2);
6196   text-align:center;
6197   vertical-align:middle;
6198 }
6199 .GJIcon{
6200   display:inline;

```

```

6201 position:relative;
6202 top:0px; left:1px;
6203 border:2px solid #44a;
6204 margin:0px; padding:1px;
6205 width:13.2; height:13.2px;
6206 border-radius:2px;
6207 font-family:Georgia;
6208 font-size:13.2px;
6209 line-height:1.0;
6210 white-space:nowrap;
6211 color:#fff; background-color:rgba(32,32,160,0.8);
6212 text-align:center;
6213 vertical-align:middle;
6214 text-shadow:0px 0px;
6215 }
6216 .GJText:focus{
6217 color:#fff !important;
6218 background-color:rgba(32,32,160,0.8) !important;
6219 line-height:1.0;
6220 }
6221 .GJText{
6222 display:inline;
6223 position:relative;
6224 top:0px; left:0px;
6225 border:0px solid #000; margin:0px; padding:0px;
6226 width:280px; height:160px;
6227 border:0px;
6228 font-family:Courier New,monospace !important;
6229 font-size:8pt;
6230 line-height:1.0;
6231 white-space:pre;
6232 color:#fff; xbackground-color:rgba(0,0,64,0.5);
6233 background-color:rgba(32,32,128,0.8) !important;
6234 }
6235 .GJMode{
6236 display:inline;
6237 position:relative;
6238 top:0px; left:0px;
6239 border:0px solid #000; border-radius:0px;
6240 margin:0px; padding:0px;
6241 width:280px; height:20px;
6242 font-size:9pt;
6243 line-height:1.0;
6244 white-space:nowrap;
6245 color:#fff; background-color:rgba(0,0,64,0.7);
6246 text-align:left;
6247 vertical-align:middle;
6248 }
6249 </style>
6250
6251 <script id="gsh-script">
6252 // 2020-0909 added, permanent local storage
6253 // https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
6254 var MyHistory = ""
6255 Permanent = localStorage;
6256 MyHistory = Permanent.getItem('MyHistory')
6257 if( MyHistory == null ){ MyHistory = "" }
6258 d = new Date()
6259 MyHistory = d.getTime()/1000+ " +document.URL+"\n" + MyHistory
6260 Permanent.setItem('MyHistory',MyHistory)
6261 //Permanent.setItem('MyWindow',window)
6262
6263 var GJLog_Win = null
6264 var GJLog_Tab = null
6265 var GJLog_Stat = null
6266 var GJLog_Text = null
6267 var GJWin_Mode = null
6268 var FProductInterval = 0
6269
6270 var GJ_FactoryID = -1
6271 var GJFactory = null
6272 if( e = document.getElementById('GJFactory_0') ){
6273 GJFactory_1.height = 0
6274 GJFactory = e
6275 e.setAttribute('class','GJFactory')
6276 var GJ_FactoryID = 0
6277 }else{
6278 GJFactory = GJFactory_1
6279 var GJ_FactoryID = 1
6280 }
6281
6282 function GJFactory_Destroy(){
6283 gjf = GJFactory
6284 //gjf = document.getElementById('GJFactory')
6285 //alert('gjf='+gjf)
6286 if( gjf != null ){
6287 if( gjf.childNodes != null ){
6288 for( i = 0; i < gjf.childNodes.length; i++ ){
6289 gjf.removeChild(gjf.childNodes[i])
6290 }
6291 }
6292 gjf.innerHTML = ''
6293 gjf.style.width = 0
6294 gjf.style.height = 0
6295 gjf.removeAttribute('style')
6296 GJLog_Win = GJLog_Tab = GJLog_Stat = GJLog_Text = GJWin_Mode = null
6297 window.clearInterval(FProductInterval)
6298 return '-- Destroy: work product destroyed'
6299 }else{
6300 return '-- Destroy: work product not exist'
6301 }
6302 }
6303
6304 var TransMode = false
6305 var OnKeyControl = false
6306 var OnKeyShift = false
6307 var OnKeyAlt = false
6308 var OnKeyJ = false
6309 var OnKeyK = false
6310 var OnKeyL = false
6311
6312 function GJWin_OnKeyUp(ev){
6313 keycode = ev.code;
6314 if( keycode == 'ShiftLeft' ){
6315 OnKeyShift = false
6316 }else
6317 if( keycode == 'ControlLeft' ){
6318 OnKeyControl = false
6319 }else
6320 if( keycode == 'AltLeft' ){
6321 OnKeyAlt = false
6322 }else
6323 if( keycode == 'KeyJ' ){ OnKeyJ = false }else
6324 if( keycode == 'KeyK' ){ OnKeyK = false }else

```

```

6325     if( keycode == 'KeyL' ){ OnKeyL = false }else
6326     {
6327     }
6328     ev.preventDefault()
6329 }
6330 function and(a,b){ if(a){ if(b){ return true; } return false; } }
6331 function GJWin_OnKeyDown(ev){
6332     keycode = ev.code;
6333     mode = ''
6334     key = ''
6335     if( keycode == 'ControlLeft' ){
6336         onKeyControl = true
6337         ev.preventDefault()
6338         return;
6339     }else
6340     if( keycode == 'ShiftLeft' ){
6341         OnKeyShift = true
6342         ev.preventDefault()
6343         return;
6344     }else
6345     if( keycode == 'AltLeft' ){
6346         ev.preventDefault()
6347         OnKeyAlt = true
6348         return;
6349     }else
6350     if( keycode == 'Backquote' ){
6351         TransMode = !TransMode
6352         ev.preventDefault()
6353     }else
6354     if( and(keycode == 'Space', OnKeyShift) ){
6355         TransMode = !TransMode
6356         ev.preventDefault()
6357     }else
6358     if( keycode == 'ShiftRight' ){
6359         TransMode = !TransMode
6360     }else
6361     if( keycode == 'Escape' ){
6362         TransMode = true
6363         ev.preventDefault()
6364     }else
6365     if( keycode == 'Enter' ){
6366         TransMode = false
6367         //ev.preventDefault()
6368     }
6369     if( keycode == 'KeyJ' ){ OnKeyJ = true }else
6370     if( keycode == 'KeyK' ){ OnKeyK = true }else
6371     if( keycode == 'KeyL' ){ OnKeyL = true }else
6372     {
6373     }
6374
6375     if( ev.altKey ){ key += 'Alt+' }
6376     if( onKeyControl ){ key += 'Ctrl+' }
6377     if( OnKeyShift ){ key += 'Shift+' }
6378     if( and(keycode != 'KeyJ', OnKeyJ) ){ key += 'J+' }
6379     if( and(keycode != 'KeyK', OnKeyK) ){ key += 'K+' }
6380     if( and(keycode != 'KeyL', OnKeyL) ){ key += 'L+' }
6381     key += keycode
6382
6383     if( TransMode ){
6384         //mode = "{\343\201\202r}"
6385         mode = "[\u2020]"
6386     }else{
6387         mode = '[---]'
6388     }
6389     //GJWin_Mode.innerHTML = "[---]"
6390     GJWin_Mode.innerHTML = mode + ' ' + key
6391     //alert('Key:'+keycode)
6392     ev.stopPropagation()
6393     //ev.preventDefault()
6394 }
6395 function GJWin_OnScroll(ev){
6396     x = DragStartX = gsh.getBoundingClientRect().left.toFixed(0)
6397     y = DragStartY = gsh.getBoundingClientRect().top.toFixed(0)
6398     GJLog_append('OnScroll: x='+x+',y='+y)
6399 }
6400 document.addEventListener('scroll',GJWin_OnScroll)
6401 function GJWin_OnResize(ev){
6402     w = window.innerWidth
6403     h = window.innerHeight
6404     GJLog_append('OnResize: w='+w+',h='+h)
6405 }
6406 window.addEventListener('resize',GJWin_OnResize)
6407
6408 var DragStartX = 0
6409 var DragStartY = 0
6410 function GJWin_DragStart(ev){
6411     // maybe this is the grabbing position
6412     this.style.position = 'fixed'
6413     x = DragStartX = this.getBoundingClientRect().left.toFixed(0)
6414     y = DragStartY = this.getBoundingClientRect().top.toFixed(0)
6415     GJLog_Stat.value = 'DragStart: x='+x+',y='+y
6416 }
6417 function GJWin_Drag(ev){
6418     x = ev.clientX; y = ev.clientY // x = ev.pageX; y = ev.pageY
6419     this.style.left = x - DragStartX
6420     this.style.top = y - DragStartY
6421     this.style.zIndex = '30000'
6422     this.style.position = 'fixed'
6423     x = this.getBoundingClientRect().left.toFixed(0)
6424     y = this.getBoundingClientRect().top.toFixed(0)
6425     GJLog_Stat.value = 'x='+x+',y='+y
6426     ev.preventDefault()
6427     ev.stopPropagation()
6428 }
6429 function GJWin_DragEnd(ev){
6430     x = ev.clientX; y = ev.clientY
6431     //x = ev.pageX; y = ev.pageY
6432     this.style.left = x - DragStartX
6433     this.style.top = y - DragStartY
6434     this.style.zIndex = '30000'
6435     this.style.position = 'fixed'
6436     if( true ){
6437         console.log("Dropped: "+this.nodeName+'#'+this.id+' x='+x+' y='+y
6438             +' parent='+this.parentNode.id)
6439     }
6440     x = this.getBoundingClientRect().left.toFixed(0)
6441     y = this.getBoundingClientRect().top.toFixed(0)
6442     GJLog_Stat.value = 'x='+x+',y='+y
6443     ev.preventDefault()
6444     ev.stopPropagation()
6445 }
6446 function GJWin_DragIgnore(ev){
6447     ev.preventDefault()
6448     ev.stopPropagation()

```

```

6449 }
6450 // 2020-09-15 let every object have console view!
6451 var GJ_ConsoleID = 0
6452 var PrevReport = new Date()
6453 function GJLog_StatUpdate(){
6454   txa = GJLog_Stat;
6455   if( txa == null ){
6456     return;
6457   }
6458   tmLap0 = new Date();
6459   p = txa.parentNode;
6460   pw = txa.getBoundingClientRect().width;
6461   ph = txa.getBoundingClientRect().height;
6462   //txa.value += '#'+p.id+' pw='+pw+' ph='+ph+'\n';
6463   tx1 = '#'+p.id+' pw='+pw+' ph='+ph+'\n';
6464
6465   w = txa.getBoundingClientRect().width;
6466   h = txa.getBoundingClientRect().height;
6467   //txa.value += 'w'+w+' h'+h+'\n';
6468   tx1 += 'w'+w+' h'+h+'\n';
6469
6470   //txa.value += '\n';
6471   //txa.value += DateShort() + '\n';
6472   tx1 += '\n';
6473   tx1 += DateShort() + '\n';
6474   tmLap1 = new Date();
6475
6476   txa.value += tx1;
6477   tmLap2 = new Date();
6478
6479   // vertical centering of the last line
6480   sHeight = txa.scrollHeight - 30; // depends on the font-size
6481   tmLap3 = new Date();
6482
6483   txa.scrollTop = sHeight; // depends on the font-size
6484   tmLap4 = new Date();
6485
6486   now = tmLap0.getTime();
6487   if( PrevReport == 0 || 10000 <= now-PrevReport ){
6488     PrevReport = now;
6489     console.log('StatBarUpdate:
6490       + 'lenge=' + txa.value.length + ' byte, '
6491       + 'time=' + (tmLap4 -tmLap0) + 'ms {'
6492       + 'tadd=' + (tmLap2 -tmLap1) + ', '
6493       + 'hcal=' + (tmLap3 -tmLap2) + ', '
6494       + 'sctl=' + (tmLap4 -tmLap3) + '}'
6495     );
6496   }
6497 }
6498 GJWin_StatUpdate = GJLog_StatUpdate;
6499 function GJ_showTime1(wid){
6500   //e = document.getElementById(wid);
6501   //console.log(wid.id+'.value.length='+wid.value.length)
6502   if( e != null ){
6503     //e.value = DateShort();
6504   }else{
6505     // should remove the Listener
6506   }
6507 }
6508 function GJWin_OnResizeTextarea(ev){
6509   this.value += 'resized: ' + '\n'
6510 }
6511 function GJ_NewConsole(wname){
6512   wid = wname + '_' + GJ_ConsoleID
6513   GJ_ConsoleID += 1
6514
6515   GJFactory.style.setProperty('width',360+'px'); //GJFSIZE
6516   GJFactory.style.setProperty('height',320+'px')
6517   e = GJFactory;
6518   console.log('GJFa #' +e.id+' from w='+e.style.width+', h='+e.style.height)
6519
6520   if( GJFactory.innerHTML == "" ){
6521     GJFactory.innerHTML = '<'+H3>GJ_Factory_' + GJ_FactoryID +'<'+H3><'+hr>\n'
6522   }else{
6523     GJFactory.innerHTML += '<'+hr>\n'
6524   }
6525
6526   gjwin = GJLog_Win = document.createElement('span')
6527   gjwin.id = wid
6528   gjwin.setAttribute('class','GJWin')
6529   gjwin.setAttribute('draggable','true')
6530   gjwin.addEventListener('dragstart',GJWin_DragStart)
6531   gjwin.addEventListener('drag',GJWin_Drag)
6532   gjwin.addEventListener('dragend',GJWin_Drag)
6533   gjwin.addEventListener('dragover',GJWin_DragIgnore)
6534   gjwin.addEventListener('dragenter',GJWin_DragIgnore)
6535   gjwin.addEventListener('dragleave',GJWin_DragIgnore)
6536   gjwin.addEventListener('dragexit',GJWin_DragIgnore)
6537   gjwin.addEventListener('drop',GJWin_DragIgnore)
6538   gjwin.addEventListener('keydown',GJWin_OnKeyDown)
6539
6540   gjtab = GJLog_Tab = document.createElement('textarea')
6541   gjtab.addEventListener('keydown',GJWin_OnKeyDown)
6542   gjtab.style.readonly = true
6543   gjtab.contenteditable = false
6544   gjtab.value = wid
6545   gjtab.id = wid + '_Tab'
6546   gjtab.setAttribute('class','GJTab')
6547   gjtab.setAttribute('spellcheck','false')
6548   gjwin.appendChild(gjtab)
6549
6550   gjstat = GJLog_Stat = document.createElement('textarea')
6551   gjstat.addEventListener('keydown',GJWin_OnKeyDown)
6552   gjstat.id = wid + '_Stat'
6553   gjstat.value = DateShort()
6554   gjstat.setAttribute('class','GJStat')
6555   gjstat.setAttribute('spellcheck','false')
6556   gjwin.appendChild(gjstat)
6557
6558   gjicon = document.createElement('span')
6559   gjicon.addEventListener('keydown',GJWin_OnKeyDown)
6560   gjicon.id = wid + '_Icon'
6561   gjicon.innerHTML = '<G<font color="#F44">J</font>'
6562   gjicon.setAttribute('class','GJIcon')
6563   gjicon.setAttribute('spellcheck','false')
6564   gjwin.appendChild(gjicon)
6565
6566   gjtext = GJLog_Text = document.createElement('textarea')
6567   gjtext.addEventListener('keydown',GJWin_OnKeyDown)
6568   gjtext.addEventListener('keyup',GJWin_OnKeyUp)
6569   gjtext.addEventListener('resize',GJWin_OnResizeTextarea)
6570   gjtext.id = wid + '_Text'
6571   gjtext.setAttribute('class','GJText')
6572   gjtext.setAttribute('spellcheck','false')

```

```

6573     gjwin.appendChild(gjtext)
6574
6575
6576     // user's mode as of IME
6577     gjmode = GJWin_Mode = document.createElement('textarea')
6578     gjmode.addEventListener('keydown',GJWin_OnKeyDown)
6579     gjmode.addEventListener('keydown',GJWin_OnKeyDown)
6580     gjmode.id = wid + '_Mode'
6581     gjmode.setAttribute('class','GJMode')
6582     gjmode.setAttribute('spellcheck','false')
6583     gjmode.innerHTML = '[---]'
6584     gjwin.appendChild(gjmode)
6585
6586     gjwin.zIndex = 30000
6587     GJFactory.appendChild(gjwin)
6588
6589     gjtab.scrollTop = 0
6590     gjstat.scrollTop = 0
6591
6592     //x = gjwin.getBoundingClientRect().left.toFixed(0)
6593     //y = gjwin.getBoundingClientRect().top.toFixed(0)
6594     //gjwin.style.position = 'static'
6595     //gjwin.style.left = 0
6596     //gjwin.style.top = 0
6597
6598     //update = '{'+wid+',value=DateShort()}',
6599     update = '{GJ_showTime('+wid+')}',
6600     // 2020-09-19 this causes memory leaks
6601     //FProductInterval = window.setInterval(update,200)
6602     //FProductInterval = window.setInterval(GJWin_StatUpdate,200)
6603     //FProductInterval = window.setInterval(GJ_showTime,200,wid);
6604     FProductInterval = window.setInterval(GJ_showTime,200,gjstat);
6605     return update
6606 }
6607 function xxxGJF_StripClass(){
6608     GJLog_Win.style.removeProperty('width')
6609     GJLog_Tab.style.removeProperty('width')
6610     GJLog_Stat.style.removeProperty('width')
6611     GJLog_Text.style.removeProperty('width')
6612     return "Stripped classes"
6613 }
6614 function isElem(id){
6615     return document.getElementById(id) != null
6616 }
6617 function GJLog_append(...args){
6618     txt = GJLog_Text;
6619     if( txt == null ){
6620         return; // maybe GJLog element is removed
6621     }
6622     logs = args.join(' ')
6623     txt.value += logs + '\n'
6624     txt.scrollTop = txt.scrollHeight
6625     //GJLog_Stat.value = DateShort()
6626 }
6627 //window.addEventListener('time',GJLog_StatUpdate)
6628 function test_GJ_Console(){
6629     window.setInterval(GJLog_StatUpdate,1000);
6630     GJ_NewConsole('GJ_Console')
6631     e = GJFactory;
6632     console.log('GJF0 #' + e.id + ' from w=' + e.style.width + ', h=' + e.style.height)
6633     e.style.width = 360; //GJFsize
6634     e.style.height = 320;
6635     console.log('GJF0 #' + e.id + ' to w=' + e.style.width + ', h=' + e.style.height)
6636 }
6637 // test_GJ_Console();
6638
6639 var StopConsoleLog = true
6640 // 2020-09-15 added,
6641 // log should be saved to permanent memory
6642 // const px = new Proxy(console.log,{ alert() })
6643 __console_log = console.log
6644 __console_info = console.info
6645 __console_warn = console.warn
6646 __console_error = console.error
6647 __console_exception = console.exception
6648 // should pop callstack info.
6649 console.exception = function(...args){
6650     __console_exception(...args)
6651     alert('-- got console.exception(""+args+"")')
6652 }
6653 console.error = function(...args){
6654     __console_error(...args)
6655     alert('-- got console.error(""+args+"")')
6656 }
6657 console.warn = function(...args){
6658     __console_warn(...args)
6659     alert('-- got console.warn(""+args+"")')
6660 }
6661 console.info = function(...args){
6662     alert('-- got console.info(""+args+"")')
6663     __console_info(...args)
6664 }
6665 console.log = function(...args){
6666     __console_log(...args)
6667     if( StopConsoleLog ){
6668         return;
6669     }
6670     if( 0 <= args[0].indexOf('!') ){
6671         //alert('-- got console.log(""+args+"")')
6672     }
6673     GJLog_append(...args)
6674 }
6675
6676 //document.getElementById('GshFaviconURL').href = GShellFavicon
6677 document.getElementById('GshFaviconURL').href = GShellInsideIcon
6678 //document.getElementById('GshFaviconURL').href = ITSmoreQR
6679 //document.getElementById('GshFaviconURL').href = GShellLogo
6680
6681 // id of GShell HTML elemets
6682 var E_BANNER = "GshBanner" // banner element in HTML
6683 var E_FOOTER = "GshFooter" // footer element in HTML
6684 var E_GINDEX = "gsh-gindex" // index of Golang code of GShell
6685 var E_GOCODE = "gsh-gocode" // Golang code of GShell
6686 var E_TODO = "gsh-todo" // TODO of GShell
6687 var E_DICT = "gsh-dict" // Dictionary of GShell
6688
6689 function bannerElem(){ return document.getElementById(E_BANNER); }
6690 function bannerStyleFunc(){ return bannerElem().style; }
6691 var bannerStyle = bannerStyleFunc()
6692 bannerStyle.backgroundImage = "url("+GShellLogo+")";
6693 //bannerStyle.backgroundImage = "url("+GShellInsideIcon+")";
6694 //bannerStyle.backgroundImage = "url("+GShellFavicon+")";
6695 GMenu.style.backgroundImage = "url("+GShellInsideIcon+")";
6696

```

```

6697 function footerElem(){ return document.getElementById(E_FOOTER); }
6698 function footerStyle(){ return footerElem().style; }
6699 //footerElem().style.backgroundImage="url("+ITSmoreQR+")";
6700 //footerStyle().backgroundImage = "url("+ITSmoreQR+")";
6701
6702 function html_fold(e){
6703     if( e.innerHTML == "Fold" ){
6704         e.innerHTML = "Unfold"
6705         document.getElementById('gsh-menu-exit').innerHTML=""
6706         document.getElementById('GshStatement').open=false
6707         GshFeatures.open = false
6708         document.getElementById('html-src').open=false
6709         document.getElementById(E_GINDEX).open=false
6710         document.getElementById(E_GOCODE).open=false
6711         document.getElementById(E_TODO).open=false
6712         document.getElementById('References').open=false
6713     }else{
6714         e.innerHTML = "Fold"
6715         document.getElementById('GshStatement').open=true
6716         GshFeatures.open = true
6717         document.getElementById(E_GINDEX).open=true
6718         document.getElementById(E_GOCODE).open=true
6719         document.getElementById(E_TODO).open=true
6720         document.getElementById('References').open=true
6721     }
6722 }
6723 function html_pure(e){
6724     if( e.innerHTML == "Pure" ){
6725         document.getElementById('gsh').style.display=true
6726         //document.style.display = false
6727         e.innerHTML = "Unpure"
6728     }else{
6729         document.getElementById('gsh').style.display=false
6730         //document.style.display = true
6731         e.innerHTML = "Pure"
6732     }
6733 }
6734
6735 var bannerIsStopping = false
6736 //NOTE: .com/JSREF/prop_style_backgroundposition.asp
6737 function shiftBG(){
6738     bannerIsStopping = !bannerIsStopping
6739     bannerStyle.backgroundPosition = "0 0";
6740 }
6741 // status should be inherited on Window Fork(), so use the status in DOM
6742 function html_stop(e,toggle){
6743     if( toggle ){
6744         if( e.innerHTML == "Stop" ){
6745             bannerIsStopping = true
6746             e.innerHTML = "Start"
6747         }else{
6748             bannerIsStopping = false
6749             e.innerHTML = "Stop"
6750         }
6751     }else{
6752         // update JavaScript variable from DOM status
6753         if( e.innerHTML == "Stop" ){ // shown if it's running
6754             bannerIsStopping = false
6755         }else{
6756             bannerIsStopping = true
6757         }
6758     }
6759 }
6760 html_stop(document.getElementById('GshMenuStop'),false) // onInit.
6761 //html_stop(bannerElem(),false) // onInit.
6762
6763 //https://www.w3schools.com/jsref/met_win_setinterval.asp
6764 function shiftBanner(){
6765     var now = new Date().getTime();
6766     //console.log("now="+now%10)
6767     if( !bannerIsStopping ){
6768         bannerStyle.backgroundPosition = ((now/10)%100000)+" 0";
6769     }
6770 }
6771 window.setInterval(shiftBanner,10); // onInit.
6772
6773 // <a href="https://developer.mozilla.org/ja/docs/Web/API/Window/open">window.open()</a>
6774 // from embedded html to standalone page
6775 var MyChildren = 0
6776 function html_fork(){
6777     GJFactory_Destroy()
6778     MyChildren += 1
6779     WinId = document.getElementById('gsh-WinId').innerHTML + "." + MyChildren;
6780     newwin = window.open("",WinId,"");
6781     src = document.getElementById("gsh");
6782     srhtml = src.outerHTML
6783     newwin.document.write("/<*>"+html>\n");
6784     newwin.document.write(srhtml);
6785     newwin.document.write("<*>"+html>\n");
6786     newwin.document.getElementById('gsh-menu-exit').innerHTML = "Close";
6787     newwin.document.getElementById('gsh-WinId').innerHTML = WinId;
6788     newwin.document.close();
6789     newwin.focus();
6790 }
6791 function html_close(){
6792     window.close()
6793 }
6794 function win_jump(win){
6795     //win = window.top;
6796     win = window.opener; // https://developer.mozilla.org/ja/docs/Web/API/window/opener
6797     if( win == null ){
6798         console.log("jump to window.opener("+win+") (Error)\n")
6799     }else{
6800         console.log("jump to window.opener("+win+")\n")
6801         win.focus();
6802     }
6803 }
6804
6805 // 0.2.9 2020-0902 created chekcsun of HTML
6806 CRC32UNIX = 0x04C11DB7 // Unix cksum
6807 function byteCRC32add(bigorc,octstr,octlen){
6808     var crc = new Int32Array(1)
6809     crc[0] = bigorc
6810
6811     let oi = 0
6812     for( ; oi < octlen; oi++ ){
6813         var oct = new Int8Array(1)
6814         oct[0] = octstr[oi]
6815         for( bi = 0; bi < 8; bi++ ){
6816             //console.log("--CRC32 "+crc[0]+" "+oct[0].toString(16)+" ["+oi+"."+bi+"]\n")
6817             ovf1 = crc[0] < 0 ? 1 : 0
6818             ovf2 = oct[0] < 0 ? 1 : 0
6819             ovf = ovf1 ^ ovf2
6820             oct[0] <<= 1

```

```

6821         crc[0] <= 1
6822         if( ovf ){ crc[0] ^= CRC32UNIX }
6823     }
6824 }
6825 //console.log("--CRC32 byteAdd return crc="+crc[0]+","+oi+"/"+"octlen+"\n")
6826 return crc[0];
6827 }
6828 function strCRC32add(bigcrc,stri,strlen){
6829     var crc = new Uint32Array(1)
6830     crc[0] = bigcrc
6831     var code = new Uint8Array(strlen);
6832     for( i = 0; i < strlen; i++){
6833         code[i] = stri.charCodeAtAt(i) // not charAt() !!!!
6834         //console.log("=== "+code[i].toString(16)+" <== "+stri[i)+"\n")
6835     }
6836     crc[0] = byteCRC32add(crc,code,strlen)
6837     //console.log("--CRC32 strAdd return crc="+crc[0)+"\n")
6838     return crc[0]
6839 }
6840 function byteCRC32end(bigcrc,len){
6841     var crc = new Uint32Array(1)
6842     crc[0] = bigcrc
6843     var slen = new Uint8Array(4)
6844     let li = 0
6845     for( ; li < 4; ){
6846         slen[li] = len
6847         li += 1
6848         len >>= 8
6849         if( len == 0 ){
6850             break
6851         }
6852     }
6853     crc[0] = byteCRC32add(crc[0],slen,li)
6854     crc[0] ^= 0xFFFFFFFF
6855     return crc[0]
6856 }
6857 function strCRC32(stri,len){
6858     var crc = new Uint32Array(1)
6859     crc[0] = 0
6860     crc[0] = strCRC32add(0,stri,len)
6861     crc[0] = byteCRC32end(crc[0],len)
6862     //console.log("--CRC32 "+crc[0]+" "+len+"\n")
6863     return crc[0]
6864 }
6865 }
6866 DestroyGJLink = null; // to be replaced
6867 DestroyFooter = null; // to be defined
6868 DestroyEventSharingCodeview = function dummy(){}
6869 Destroy_WirtualDesktop = function(){}
6870 }
6871 function getSourceText(){
6872     if( DestroyFooter != null ) DestroyFooter();
6873     version = document.getElementById('GshVersion').innerHTML
6874     sfavico = document.getElementById('GshFaviconURL').href;
6875     sbanner = document.getElementById('GshBanner').style.backgroundImage;
6876     spositi = document.getElementById('GshBanner').style.backgroundPosition;
6877 }
6878 if( document.getElementById('GJC_1') != null ){ GJC_1.remove() }
6879 if( DestroyGJLink != null ) DestroyGJLink();
6880 DestroyEventSharingCodeview();
6881 Destroy_WirtualDesktop();
6882 }
6883 // these should be removed by CSS selector or class, after sevaed to non-printed attribute
6884 GshBanner.removeAttribute('style');
6885 document.getElementById('GshMenuSign').removeAttribute("style");
6886 styleGMenu = GMenu.getAttribute("style")
6887 GMenu.removeAttribute("style");
6888 styleGStat = GStat.getAttribute("style")
6889 GStat.removeAttribute("style");
6890 styleGTop = GTop.getAttribute("style")
6891 GTop.removeAttribute("style");
6892 styleGshGrid = GshGrid.getAttribute("style")
6893 GshGrid.removeAttribute("style");
6894 //styleGPos = GPos.getAttribute("style");
6895 //GPos.removeAttribute("style");
6896 //GPos.innerHTML = "";
6897 //styleGLog = GLog.getAttribute("style");
6898 //GLog.removeAttribute("style");
6899 //GLog.innerHTML = "";
6900 styleGShellPlane = GShellPlane.getAttribute("style")
6901 GShellPlane.removeAttribute("style")
6902 styleRawTextViewer = RawTextViewer.getAttribute("style")
6903 RawTextViewer.removeAttribute("style")
6904 styleRawTextViewerClose = RawTextViewerClose.getAttribute("style")
6905 RawTextViewerClose.removeAttribute("style")
6906 }
6907 GshFaviconURL.href = "";
6908 }
6909 //it seems that interHTML and outerHTML generate style="" for these (??)
6910 //GshBanner.removeAttribute('style');
6911 //GshFooter.removeAttribute('style');
6912 //GshMenuSign.removeAttribute('style');
6913 GshBanner.style=""
6914 GshMenuSign.style=""
6915 }
6916 textarea = document.createElement("textarea")
6917 srchtml = document.getElementById("gsh").outerHTML;
6918 //textarea = document.createElement("textarea")
6919 // 2020-0910 ?? ... this causes inserting style="" to Banner and Footer,
6920 // with Chromium?/ after reloading from file:///
6921 textarea.innerHTML = srchtml
6922 // <a href="https://stackoverflow.com/questions/5796718/html-entity-decode">Thanks</a>
6923 var rawtext = textarea.value
6924 //textarea.destroy()
6925 //rawtext = gsh.textContent // this removes #include <FILENAME> too
6926 var orgtext =
6927     +"/<"+html>\n" // lost preamble text
6928     + rawtext
6929     + "<"+/html>\n" // lost trail text
6930 ;
6931 }
6932 tlen = orgtext.length
6933 //console.log("getSourceText: length="+tlen+"\n")
6934 document.getElementById('GshFaviconURL').href = sfavico;
6935 }
6936 document.getElementById('GshBanner').style.backgroundImage = sbanner;
6937 document.getElementById('GshBanner').style.backgroundPosition = spositi;
6938 }
6939 GStat.setAttribute("style",styleGStat)
6940 GMenu.setAttribute("style",styleGMenu)
6941 GTop.setAttribute("style",styleGTop)
6942 //GLog.setAttribute("style",styleGLog)
6943 //GPos.setAttribute("style",styleGPos)
6944 GshGrid.setAttribute("style",styleGshGrid)

```



```

6945 GShellPlane.setAttribute("style",styleGShellPlane)
6946 RawTextViewer.setAttribute("style",styleRawTextViewer)
6947 RawTextViewerClose.setAttribute("style",styleRawTextViewerClose)
6948 canontext = origtext.replace(' style=""','')
6949 // open="" too
6950 return canontext
6951 }
6952 function getDigest(){
6953   var text = ""
6954   text = getSourceText()
6955   var digest = ""
6956   tlen = text.length
6957   digest = strCRC32(text,tlen) + " " + tlen
6958   return { text, digest }
6959 }
6960 function html_digest(){
6961   version = document.getElementById('GshVersion').innerHTML
6962   let {text, digest} = getDigest()
6963   alert("cksum: " + digest + " " + version)
6964 }
6965 function charsin(stri,char){
6966   ln = 0;
6967   for( i = 0; i < stri.length; i++){
6968     if( stri.charCodeAt(i) == char.charCodeAt(0) )
6969       ln++;
6970   }
6971   return ln;
6972 }
6973
6974 //class digestElement extends HTMLElement { }
6975 //< script>customElements.define('digest',digestElement)< /script>
6976 function showDigest(e){
6977   result = 'version=' + GshVersion.innerHTML + '\n'
6978   result += 'lines=' + e.dataset.lines + '\n'
6979   result += 'length=' + e.dataset.length + '\n'
6980   result += 'crc32u=' + e.dataset.crc32u + '\n'
6981   result += 'time=' + e.dataset.time + '\n';
6982
6983   alert(result)
6984 }
6985
6986 function html_sign(e){
6987   if( RawTextViewer.style.zIndex == 1000 ){
6988     hideRawTextViewer()
6989     return
6990   }
6991   DestroyEventSharingCodeview();
6992   Destroy_WirtualDesktop();
6993   GJFactory_Destroy()
6994   if( DestroyGJLink != null ) DestroyGJLink();
6995   //gsh_digest_.innerHTML = "";
6996   text = getSourceText() // the original text
6997   tlen = text.length
6998   digest = strCRC32(text,tlen)
6999   //gsh_digest_.innerHTML = digest + " " + tlen
7000   //text = getSourceText() // the text with its digest
7001   Lines = charsin(text,'\n')
7002
7003   name = "gsh"
7004   sid = name + "--digest"
7005   d = new Date()
7006   signedAt = d.getTime()
7007
7008   sign = '/'+'*'+<'span'\n'
7009     + ' id="' + sid + '"\n'
7010     + ' class=" digest_"\n'
7011     + ' data-target-id="'+name+'"\n'
7012     + ' data-crc32u="' + digest + '"\n'
7013     + ' data-length="' + tlen + '"\n'
7014     + ' data-lines="' + Lines + '"\n'
7015     + ' data-time="' + signedAt + '"\n'
7016     + '>' + '/span>\n*'+'/\n'
7017
7018   text = sign + text
7019
7020   txhtml = '<' + 'table id="LineNumbered"><' + 'tr><' + 'td'
7021     + '<' + 'textarea cols=5 rows=' + Lines + ' class="LineNumber">'
7022   for( i = 1; i <= Lines; i++){
7023     txhtml += i.toString() + '\n'
7024   }
7025   txhtml += ""
7026     + '<' + '/textarea>'
7027     + '<' + '/td><' + 'td'
7028     + '<' + 'textarea cols=150 rows=' + Lines + ' spellcheck="false"'
7029     + ' class="LineNumbered">'
7030     + text + '<'+'/textarea>'
7031     + '<' + '/td><' + '/tr><' + '/table>'
7032
7033   for( i = 1; i <= 30; i++){
7034     txhtml += '<br>\n'
7035   }
7036   RawTextViewer.innerHTML = txhtml
7037   RawTextViewer.spellcheck = false // (spellcheck above seems ineffective)
7038
7039   btn = e
7040   e.style.color = "rgba(128,128,255,0.9)";
7041   y = e.getBoudingClientRect().top.toFixed(0)
7042   //h = e.getBoudingClientRect().height.toFixed(0)
7043   RawTextViewer.style.top = Number(y) + 30
7044   RawTextViewer.style.left = 100;
7045   RawTextViewer.style.height = window.innerHeight - 20;
7046   //RawTextViewer.style.Opacity = 1.0;
7047   //RawTextViewer.style.backgroundColor = "rgba(0,0,0,0)";
7048   RawTextViewer.style.backgroundColor = "rgba(255,255,255,0.8)";
7049   RawTextViewer.style.zIndex = 1000;
7050   RawTextViewer.style.display = true;
7051
7052   if( RawTextViewerClose.style == null ){
7053     RawTextViewerClose.style = "";
7054   }
7055   RawTextViewerClose.style.top = Number(y) + 10
7056   RawTextViewerClose.style.left = 100;
7057   RawTextViewerClose.style.zIndex = 1001;
7058
7059   ScrollToElement(CurElement,RawTextViewerClose)
7060 }
7061 function hideRawTextViewer(){
7062   RawTextViewer.style.left = 10000;
7063   RawTextViewer.style.zIndex = -100;
7064   RawTextViewer.style.Opacity = 0.0;
7065   RawTextViewer.style = null
7066   RawTextViewer.innerHTML = "";
7067
7068   GshMenuSign.style.color = "rgba(255,128,128,1.0)";

```

```

7069 RawTextViewerClose.style.top = 0;
7070 RawTextViewerClose.style = null;
7071 }
7072
7073 // source code view
7074 function frame_close(){
7075   srcframe = document.getElementById("src-frame");
7076   srcframe.innterHTML = "";
7077   //srcframe.style.cols = 1;
7078   srcframe.style.rows = 1;
7079   srcframe.style.height = 0;
7080   srcframe.style.display = false;
7081   src = document.getElementById("SrcTextarea");
7082   src.innerHTML = ""
7083   //src.cols = 0
7084   src.rows = 0
7085   src.display = false
7086   //alert("--closed--")
7087 }
7088 //<!-- | <span onclick="html_view();">Source</span> -->
7089 //<!-- | <span onclick="frame_close();">SourceClose</span> -->
7090 //<!-- | <span>Download</span> -->
7091 function frame_open(){
7092   if( DestroyFooter != null ) DestroyFooter();
7093   document.getElementById('GshFaviconURL').href = "";
7094   oldsrc = document.getElementById("GENSRC");
7095   if( oldsrc != null ){
7096     //alert("--I--(erasing old text)")
7097     oldsrc.innterHTML = "";
7098     return
7099   }else{
7100     //alert("--I--(no old text)")
7101   }
7102   styleBanner = GshBanner.getAttribute("style")
7103   GshBanner.removeAttribute("style")
7104   if( document.getElementById('GJC_1') ){ GJC_1.remove() }
7105
7106   GshFaviconURL.href = "";
7107   GStat.removeAttribute('style')
7108   GshGrid.removeAttribute('style')
7109   GshMenuSign.removeAttribute('style')
7110   //GPos.removeAttribute('style')
7111   //GPos.innerHTML = "";
7112   //GLog.removeAttribute('style')
7113   //GLog.innerHTML = "";
7114   GMenu.removeAttribute('style')
7115   GTop.removeAttribute('style')
7116   GShellPlane.removeAttribute('style')
7117   RawTextViewer.removeAttribute('style')
7118   RawTextViewerClose.removeAttribute('style')
7119
7120   if( DestroyGJLink != null ) DestroyGJLink();
7121   GJFactory_Destroy()
7122   Destroy_WirtualDesktop();
7123   DestroyEventSharingCodeview();
7124
7125   src = document.getElementById("gsh");
7126   srchtml = src.outerHTML
7127   srcframe = document.getElementById("src-frame");
7128   srcframe.innerHTML = ""
7129   + "<"+>cite id="GENSRC">\n"
7130   + "<"+>style>\n"
7131   + "#GENSRC textarea{tab-size:4;}\n"
7132   + "#GENSRC textarea{-o-tab-size:4;}\n"
7133   + "#GENSRC textarea{-moz-tab-size:4;}\n"
7134   + "#GENSRC textarea{spellcheck:false;}\n"
7135   + "</"+>style>\n"
7136   + "<"+>textarea id="SrcTextarea" cols=100 rows=20 class="gsh-code" spellcheck="false">'
7137   + /*<"+>html>\n" // lost preamble text
7138   + srchtml
7139   + "<"+>/html>\n" // lost trail text
7140   + "<"+>textarea>\n"
7141   + "</"+>cite><!-- GENSRC -->\n";
7142
7143   //srcframe.style.cols = 80;
7144   //srcframe.style.rows = 80;
7145
7146   GshBanner.setAttribute('style',styleBanner)
7147 }
7148 function fill_CSSView(){
7149   part = document.getElementById('GshStyleDef')
7150   view = document.getElementById('gsh-style-view')
7151   view.innerHTML = ""
7152   + "<"+>textarea cols=100 rows=20 class="gsh-code">'
7153   + part.innerHTML
7154   + "<"+>/textarea>"
7155 }
7156 function fill_JavaScriptView(){
7157   jspart = document.getElementById('gsh-script')
7158   view = document.getElementById('gsh-script-view')
7159   view.innerHTML = ""
7160   + "<"+>textarea cols=100 rows=20 class="gsh-code">'
7161   + jspart.innerHTML
7162   + "<"+>/textarea>"
7163 }
7164 function fill_DataView(){
7165   part = document.getElementById('gsh-data')
7166   view = document.getElementById('gsh-data-view')
7167   view.innerHTML = ""
7168   + "<"+>textarea cols=100 rows=20 class="gsh-code">'
7169   + part.innerHTML
7170   + "<"+>/textarea>"
7171 }
7172 function jumpto_StyleView(){
7173   jsview = document.getElementById('html-src')
7174   jsview.open = true
7175   jsview = document.getElementById('gsh-style-frame')
7176   jsview.open = true
7177   fill_CSSView()
7178 }
7179 function jumpto_JavaScriptView(){
7180   jsview = document.getElementById('html-src')
7181   jsview.open = true
7182   jsview = document.getElementById('gsh-script-frame')
7183   jsview.open = true
7184   fill_JavaScriptView()
7185 }
7186 function jumpto_DataView(){
7187   jsview = document.getElementById('html-src')
7188   jsview.open = true
7189   jsview = document.getElementById('gsh-data-frame')
7190   jsview.open = true
7191   fill_DataView()
7192 }

```

```

7193 function jumpto_WholeView(){
7194     jsview = document.getElementById('html-src')
7195     jsview.open = true
7196     jsview = document.getElementById('gsh-whole-view')
7197     jsview.open = true
7198     frame_open()
7199 }
7200 function html_view(){
7201     html_stop();
7202
7203     banner = document.getElementById('GshBanner').style.backgroundImage;
7204     footer = document.getElementById('GshFooter').style.backgroundImage;
7205     document.getElementById('GshBanner').style.backgroundImage = "";
7206     document.getElementById('GshBanner').style.backgroundPosition = "";
7207     document.getElementById('GshFooter').style.backgroundImage = "";
7208
7209     //srcwin = window.open("", "CodeView2", "");
7210     srcwin = window.open("", "", "");
7211     srcwin.document.write("<span id='gsh'>\n");
7212
7213     src = document.getElementById("gsh");
7214     srcwin.document.write("<"+style>\n");
7215     srcwin.document.write("textareat{tab-size:4;}\n");
7216     srcwin.document.write("textareat{-o-tab-size:4;}\n");
7217     srcwin.document.write("textareat{-moz-tab-size:4;}\n");
7218     srcwin.document.write("</style>\n");
7219     srcwin.document.write("<h2>\n");
7220     srcwin.document.write("<"+span onclick='window.close();\n">Close</span> | \n");
7221     //srcwin.document.write("<"+span onclick='html_stop();\n">Run</span>\n");
7222     srcwin.document.write("</h2>\n");
7223     srcwin.document.write("<"+textareat id='gsh-src-src'\n cols=100 rows=60>\n");
7224     srcwin.document.write("/*<"+html>\n");
7225     srcwin.document.write("<"+span id='gsh'\n");
7226     srcwin.document.write(src.innerHTML);
7227     srcwin.document.write("</span><"+html>\n");
7228     srcwin.document.write("</"+textareat>\n");
7229
7230     document.getElementById('GshBanner').style.backgroundImage = banner;
7231     document.getElementById('GshFooter').style.backgroundImage = footer
7232
7233     sty = document.getElementById("GshStyleDef");
7234     srcwin.document.write("<"+style>\n");
7235     srcwin.document.write(sty.innerHTML);
7236     srcwin.document.write("</"+style>\n");
7237
7238     run = document.getElementById("gsh-script");
7239     srcwin.document.write("<"+script>\n");
7240     srcwin.document.write(run.innerHTML);
7241     srcwin.document.write("</"+script>\n");
7242
7243     srcwin.document.write("</span><"+html>\n"); // gsh span
7244     srcwin.document.close();
7245     srcwin.focus();
7246 }
7247 GSH = document.getElementById("gsh")
7248
7249 //GSH.onclick = "alert('Ouch!')"
7250 //GSH.css = "{background-color:#eef;}"
7251 //GSH.style = "background-color:#eef;"
7252 //GSH.style.display = false;
7253 //alert('Ouch0!')
7254 //GSH.style.display = true;
7255
7256 // 2020-0904 created, tentative
7257 document.addEventListener('keydown', jgshCommand);
7258 //CurElement = GshStatement
7259 CurElement = GshMenu
7260 MemElement = GshMenu
7261
7262 function nextSib(e){
7263     n = e.nextSibling;
7264     for( i = 0; i < 100; i++){
7265         if( n == null ){
7266             break;
7267         }
7268         if( n.nodeName == "DETAILS" ){
7269             return n;
7270         }
7271         n = n.nextSibling;
7272     }
7273     return null;
7274 }
7275 function prevSib(e){
7276     n = e.previousSibling;
7277     for( i = 0; i < 100; i++){
7278         if( n == null ){
7279             break;
7280         }
7281         if( n.nodeName == "DETAILS" ){
7282             return n;
7283         }
7284         n = n.previousSibling;
7285     }
7286     return null;
7287 }
7288 function setColor(e, eName, eColor){
7289     if( e.hasChildNodes() ){
7290         s = e.childNodes;
7291         if( s != null ){
7292             for( ci = 0; ci < s.length; ci++){
7293                 if( s[ci].nodeName == eName ){
7294                     s[ci].style.color = eColor;
7295                     //s[ci].style.backgroundColor = eColor;
7296                     break;
7297                 }
7298             }
7299         }
7300     }
7301 }
7302
7303 // https://docs.microsoft.com/en-us/previous-versions//hh781509(v=vs.85)
7304 function showCurElementPosition(ev){
7305     // if( document.getElementById("GPos") == null ){
7306     //     return;
7307     // }
7308     // if( GPos == null ){
7309     //     return;
7310     // }
7311     e = CurElement
7312     y = e.getBoundingClientRect().top.toFixed(0)
7313     x = e.getBoundingClientRect().left.toFixed(0)
7314
7315     h = ev + " "
7316     h += 'y'+y+', '+ 'x'+x+' -- "

```

```

7317 h += "w=" + window.innerWidth + ", h=" + window.innerHeight + " -- "
7318 //GPos.test = h
7319 //GPos.innerHTML = h
7320 // GPos.innerHTML = h
7321 }
7322 }
7323 function zero2(n){
7324   if( n < 10 ){
7325     return '0' + n;
7326   }else{
7327     return n;
7328   }
7329 }
7330 function DateHourMin(){
7331   d = new Date();
7332   //return '%02d:%02d'.sprintf(d.getHours(),d.getMinutes())
7333   return zero2(d.getHours()) + ":" + zero2(d.getMinutes());
7334 }
7335 function DateShort(){
7336   d = new Date()
7337   return d.getFullYear()
7338     + "/" + zero2(d.getMonth())
7339     + "/" + zero2(d.getDate())
7340     + " " + zero2(d.getHours())
7341     + ":" + zero2(d.getMinutes())
7342     + ":" + zero2(d.getSeconds())
7343 }
7344 function DateLong0(ms){
7345   d = new Date();
7346   d.setTime(ms);
7347   return
7348     //d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7349     //+ d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7350     DateShort()
7351     + "." + d.getMilliseconds()
7352     + " " + d.getTimezoneOffset()/60
7353     + " "
7354     + d.getTime() + "." + d.getMilliseconds()
7355 }
7356 }
7357 function DateLong(){
7358   return DateLong0(new Date());
7359 }
7360 function GShellMenu(e){
7361   //GLog.innerHTML = "Hello, World! (" + DateLong() + ")"
7362   showGShellPlane()
7363 }
7364 // placements of planes
7365 function GShellResizeX(ev){
7366   //if( document.getElementById("GMenu") != null ){
7367     GMenu.style.left = window.innerWidth - 100
7368     GMenu.style.top = window.innerHeight - 90 - 200
7369     //console.log("place GMENU "+GMenu.style.left+" "+GMenu.style.top)
7370   }
7371   //}
7372   GStat.style.width = window.innerWidth
7373   //if( document.getElementById("GPos") != null ){
7374     //GPos.style.width = window.innerWidth
7375     //GPos.style.top = window.innerHeight - 30; //GPos.style.height
7376   //}
7377   //if( document.getElementById("GLog") != null ){
7378     // GLog.style.width = window.innerWidth
7379     //GLog.innerHTML = ""
7380   //}
7381   //if( document.getElementById("GLog") != null ){
7382     //GLog.innerHTML = "Resize: w=" + window.innerWidth +
7383     //", h=" + window.innerHeight
7384   //}
7385   showCurElementPosition(ev)
7386 }
7387 function GShellResize(){
7388   GShellResizeX("[RESIZE]")
7389 }
7390 window.onresize = GShellResize
7391 var prevNode = null
7392 var LogMouseMoveOverElement = false;
7393 function GJSH_OnMouseMove(ev){
7394   if( LogMouseMoveOverElement == false ){
7395     return;
7396   }
7397   x = ev.clientX
7398   y = ev.clientY
7399   d = new Date()
7400   t = d.getTime() / 1000
7401   if( document.elementFromPoint ){
7402     e = document.elementFromPoint(x,y)
7403     if( e != null ){
7404       if( e == prevNode ){
7405         }else{
7406           console.log('Mo-' + t + '+' + x + '+' + y + ' '
7407             + e.nodeType + ' ' + e.tagName + '#' + e.id)
7408           prevNode = e
7409         }
7410       }else{
7411         console.log(t + '+' + x + '+' + y + ' no element')
7412       }
7413     }else{
7414       console.log(t + '+' + x + '+' + y + ' no elementFromPoint')
7415     }
7416   }
7417 window.addEventListener('mousemove',GJSH_OnMouseMove);
7418 }
7419 function GJSH_OnMouseMoveScreen(ev){
7420   x = ev.screenX
7421   y = ev.screenY
7422   d = new Date()
7423   t = d.getTime() / 1000
7424   console.log(t + '+' + x + '+' + y + ' no elementFromPoint')
7425 }
7426 //screen.addEventListener('mousemove',GJSH_OnMouseMoveScreen);
7427 }
7428 function ScrollToElement(oe,ne){
7429   ne.scrollIntoView()
7430   ny = ne.getBoundingClientRect().top.toFixed(0)
7431   nx = ne.getBoundingClientRect().left.toFixed(0)
7432   //GLog.innerHTML = "[" + ny + ", " + nx + "]"
7433   //window.scrollTo(0,0)
7434 }
7435 GTop.style.backgroundColor = "rgba(0,0,0,0.0)"
7436 GshGrid.style.left = '250px';
7437 GshGrid.style.zIndex = 0
7438 if( false ){
7439   oy = oe.getBoundingClientRect().top.toFixed(0)
7440   ox = oe.getBoundingClientRect().left.toFixed(0)

```

```

7441     y = e.getBoundingClientRect().top.toFixed(0)
7442     x = e.getBoundingClientRect().left.toFixed(0)
7443     window.scrollTo(x,y)
7444     my = e.getBoundingClientRect().top.toFixed(0)
7445     mx = e.getBoundingClientRect().left.toFixed(0)
7446     //GLog.innerHTML = ["+oy+", "+ox+"->["+y+", "+x+"->["+ny+", "+nx+"]"
7447   }
7448 }
7449 function showGShellPlane(){
7450   if( GShellPlane.style.zIndex == 0 ){
7451     GShellPlane.style.zIndex = 1000;
7452     GShellPlane.style.left = 30;
7453     GShellPlane.style.height = 320;
7454     GShellPlane.innerHTML = DateLong() + "<br>" +
7455       "-- History --<br>" + MyHistory;
7456   }else{
7457     GShellPlane.style.zIndex = 0;
7458     GShellPlane.style.left = 0;
7459     GShellPlane.style.height = 50;
7460     GShellPlane.innerHTML = "";
7461   }
7462 }
7463 var SuppressGJShell = false
7464 function jgshCommand(kevent){
7465   if( SuppressGJShell ){
7466     return
7467   }
7468   key = kevent
7469   keycode = key.code
7470   //GStat.style.width = window.innerWidth
7471   GStat.style.backgroundColor = "rgba(0,0,0,0.4)"
7472
7473   console.log("JSGsh-Key:"+keycode+"(^-^)/")
7474   if( keycode == "Slash" ){
7475     console.log('( '+x+', '+y+' )')
7476     e = document.elementFromPoint(x,y)
7477     console.log('( '+x+', '+y+' ) '+e.nodeType+' '+e.tagName+'#'+e.id)
7478   }else
7479   if( keycode == "Digit0" ){ // fold side-bar
7480     // "Zero page"
7481     showGShellPlane();
7482   }else
7483   if( keycode == "Digit1" ){ // fold side-bar
7484     primary.style.width = "94%"
7485     secondary.style.width = "0%"
7486     secondary.style.opacity = 0
7487     GStat.innerHTML = "[Single Column View]"
7488   }else
7489   if( keycode == "Digit2" ){ // unfold side-bar
7490     primary.style.width = "58%"
7491     secondary.style.width = "36%"
7492     secondary.style.opacity = 1
7493     GStat.innerHTML = "[Double Column View]"
7494   }else
7495   if( keycode == "KeyU" ){ // fold/unfold all
7496     html_fold(GshMenuFold);
7497     location.href = "#"+CurElement.id;
7498   }else
7499   if( keycode == "KeyO" || keycode == "ArrowRight" ){ // fold the element
7500     CurElement.open = !CurElement.open;
7501   }else
7502   if( keycode == "ArrowRight" ){ // unfold the element
7503     CurElement.open = true
7504   }else
7505   if( keycode == "ArrowLeft" ){ // unfold the element
7506     CurElement.open = false
7507   }else
7508   if( keycode == "KeyI" ){ // inspect the element
7509     e = CurElement
7510     //GLog.innerHTML =
7511     GJLog_append("Current Element: " + e + "<br>"
7512       + "name="+e.nodeName + ", "
7513       + "id="+e.id + " "
7514       + "children="+e.childNodes.length + ", "
7515       + "parent="+e.parentNode.id + "<br>"
7516       + "text="+e.textContent)
7517     GStat.style.backgroundColor = "rgba(0,0,0,0.8)"
7518     return
7519   }else
7520   if( keycode == "KeyM" ){ // memory the position
7521     MemElement = CurElement
7522   }else
7523   if( keycode == "KeyN" || keycode == "ArrowDown" ){ // next element
7524     e = nextSib(CurElement)
7525     if( e != null ){
7526       setColor(CurElement,"SUMMARY","#fff")
7527       setColor(e,"SUMMARY","#8f8") // should be complement ?
7528       oe = CurElement
7529       CurElement = e
7530       //location.href = "#"+e.id;
7531       ScrollToElement(oe,e)
7532     }
7533   }else
7534   if( keycode == "KeyP" || keycode == "ArrowUp" ){ // previous element
7535     oe = CurElement
7536     e = prevSib(CurElement)
7537     if( e != null ){
7538       setColor(CurElement,"SUMMARY","#fff")
7539       setColor(e,"SUMMARY","#8f8") // should be complement ?
7540       CurElement = e
7541       //location.href = "#"+e.id;
7542       ScrollToElement(oe,e)
7543     }else{
7544       e = document.getElementById("GshBanner")
7545       if( e != null ){
7546         setColor(CurElement,"SUMMARY","#fff")
7547         CurElement = e
7548         ScrollToElement(oe,e)
7549       }else{
7550         e = document.getElementById("primary")
7551         if( e != null ){
7552           setColor(CurElement,"SUMMARY","#fff")
7553           CurElement = e
7554           ScrollToElement(oe,e)
7555         }
7556       }
7557     }
7558   }else
7559   if( keycode == "KeyR" ){
7560     location.reload()
7561   }else
7562   if( keycode == "KeyJ" ){
7563     GshGrid.style.top = '120px';
7564     GshGrid.innerHTML = '( > < ) {Down}';

```

```

7565     }else
7566     if( keycode == "KeyK" ){
7567         GshGrid.style.top = '0px';
7568         GshGrid.innerHTML = '(-~){Up}';
7569     }else
7570     if( keycode == "KeyH" ){
7571         GshGrid.style.left = '0px';
7572         GshGrid.innerHTML = "('_~){Left}";
7573     }else
7574     if( keycode == "KeyL" ){
7575         //GLog.innerHTML +=
7576         GJLog_append(
7577             'screen='+screen.width+'px'+<br>'+
7578             'window='+window.innerWidth+'px'+<br>'
7579         )
7580         GshGrid.style.left = (document.documentElement.clientWidth-160).toString(10)+'px';
7581         GshGrid.innerHTML = '(@_@){Right}';
7582     }else
7583     if( keycode == "KeyS" ){
7584         html_stop(GshMenuStop,true)
7585     }else
7586     if( keycode == "KeyF" ){
7587         html_fork()
7588     }else
7589     if( keycode == "KeyC" ){
7590         window.close()
7591     }else
7592     if( keycode == "KeyD" ){
7593         html_digest()
7594     }else
7595     if( keycode == "KeyV" ){
7596         e = document.getElementById('gsh-digest')
7597         if( e != null ){
7598             showDigest(e)
7599         }
7600     }
7601 }
7602 showCurElementPosition(["+key.code+" --]);
7603 //if( document.getElementById("GPos") != null ){
7604 //GPos.innerHTML += ["+key.code+" --"]
7605 //}
7606 //GShellResizeX(["+key.code+" --"]);
7607 }
7608 GShellResizeX(["INIT"]);
7609
7610 DisplaySize = '-- Display: ' + 'screen='+screen.width+'px, ' + 'window='+window.innerWidth+'px';
7611
7612 let {text, digest} = getDigest()
7613 //GLog.innerHTML +=
7614 GJLog_append(
7615     '-- GShell: ' + GshVersion.innerHTML + '\n' +
7616     '-- Digest: ' + digest + '\n' +
7617     DisplaySize
7618     //+ "<br>" + "-- LastVisit:<br>" + MyHistory
7619 )
7620 GShellResizeX(null);
7621
7622 // <a href="https://www.w3.org/TR/WebCryptoAPI/">Web Cryptography API</a>
7623 //Convert a string into an ArrayBuffer
7624 //from https://developers.google.com/web/updates/2012/06/how-to-convert-ArrayBuffer-to-and-from-String
7625 function str2ab(str) {
7626     const buf = new ArrayBuffer(str.length);
7627     const bufView = new Uint8Array(buf);
7628     for (let i = 0, strLen = str.length; i < strLen; i++) {
7629         bufView[i] = str.charCodeAt(i);
7630     }
7631     return buf;
7632 }
7633 function importPrivateKey(pem) {
7634     const binaryDerString = window.atob(pemContents);
7635     const binaryDer = str2ab(binaryDerString);
7636     return window.crypto.subtle.importKey(
7637         "pkcs8",
7638         binaryDer,
7639         {
7640             name: "RSA-PSS",
7641             modulusLength: 2048,
7642             publicExponent: new Uint8Array([1, 0, 1]),
7643             hash: "SHA-256",
7644         },
7645         true,
7646         ["sign"]
7647     );
7648 }
7649 //importPrivateKey(ppem)
7650
7651 //key = {}
7652 //buf = "abc"
7653 //enc = "xyzxxxxxx"; //crypto.publicEncrypt(key,buf)
7654 //b64 = btoa(enc)
7655 //dec = atob(b64)
7656 //GLog.innerHTML = "enc:" + b64 + ", dec:" + dec
7657 </script>
7658 */
7659
7660 /*
7661 <!-- ----- GJConsole BEGIN { ----- -->
7662 <span id="gjc" data-title="GJConsole" data-author="sato@its-more.jp">
7663 <details><summary>GJ Console</summary>
7664 <p>
7665 <span id="GJE_RootNode0"></span>
7666 </p>
7667 <style id="GJConsoleStyle">
7668 .GJConsole {
7669     z-index:1000;
7670     width:400; height:200px;
7671     margin:2px;
7672     color:#fff; background-color:#66a;
7673     font-size:12px; font-family:monospace,Courier New;
7674 }
7675 </style>
7676
7677 <script id="GJConsoleScript" class="GJConsole">
7678     var PS1 = "% "
7679     function GJC_Keydown(keyevent){
7680         key = keyevent.code
7681         if( key == "Enter" ){
7682             GJC_Command(this)
7683             this.value += "\n" + PS1 // prompt
7684         }else
7685         if( key == "Escape"){
7686             SuppressGJShell = false
7687             GshMenu.focus() // should be previous focus
7688         }

```

```

7689 }
7690 var GJC_SessionId
7691 function GJC_SetSessionId(){
7692     var xd = new Date()
7693     GJC_SessionId = xd.getTime() / 1000
7694 }
7695 GJC_SetSessionId()
7696 function GJC_Memory(mem,args,text){
7697     argv = args.split(' ')
7698     cmd = argv[0]
7699     argv.shift()
7700     args = argv.join(' ')
7701     ret = ""
7702
7703     if( cmd == 'clear' ){
7704         Permanent.setItem(mem,'')
7705     }else
7706     if( cmd == 'read' ){
7707         ret = Permanent.getItem(mem)
7708     }else
7709     if( cmd == 'save' ){
7710         val = Permanent.getItem(mem)
7711         if( val == null ){ val = "" }
7712         d = new Date()
7713         val += d.getTime()/1000+ " "+GJC_SessionId+ " "+document.URL+ " "+args+"\n"
7714         val += text.value
7715         Permanent.setItem(mem,val)
7716     }else
7717     if( cmd == 'write' ){
7718         val = Permanent.getItem(mem)
7719         if( val == null ){ val = "" }
7720         d = new Date()
7721         val += d.getTime()/1000+ " "+GJC_SessionId+ " "+document.URL+ " "+args+"\n"
7722         Permanent.setItem(mem,val)
7723     }else{
7724         ret = "Commands: write | read | save | clear"
7725     }
7726     return ret
7727 }
7728 // -- 2020-09-14 added TableEditor
7729 var GJE_CurElement = null; //GJE_RootNode
7730 GJE_NodeSaved = null
7731 GJE_TableNo = 1
7732 function GJE_StyleKeyCommand(kev){
7733     keycode = kev.code
7734     console.log('GJE-Key: '+keycode)
7735     if( keycode == 'Escape' ){
7736         GJE_SetStyle(this);
7737     }
7738     kev.stopPropagation()
7739     // https://developer.mozilla.org/en-US/docs/Web/API/Event/stopPropagation
7740 }
7741 var GJE_CommandMode = false
7742 function GJE_TableKeyCommand(kev,tab){
7743     wasCmdMode = GJE_CommandMode
7744     key = kev.code
7745     if( key == 'Escape' ){
7746         console.log("To command mode: "+tab.nodeName+"#"+tab.id)
7747         //tab.setAttribute('contenteditable','false')
7748         tab.style.caretColor = "blue"
7749         GJE_CommandMode = true
7750     }else
7751     if( key == "KeyA" ){
7752         tab.style.caretColor = "red"
7753         GJE_CommandMode = false
7754     }else
7755     if( key == "KeyI" ){
7756         tab.style.caretColor = "red"
7757         GJE_CommandMode = false
7758     }else
7759     if( key == "KeyO" ){
7760         tab.style.caretColor = "red"
7761         GJE_CommandMode = false
7762     }else
7763     if( key == "KeyJ" ){
7764         console.log("ROW-DOWN")
7765     }else
7766     if( key == "KeyK" ){
7767         console.log("ROW-UP")
7768     }else
7769     if( key == "Keyw" ){
7770         console.log("COL-FORW")
7771     }else
7772     if( key == "Keyb" ){
7773         console.log("COL-BACK")
7774     }
7775     kev.stopPropagation()
7776     if( wasCmdMode ){
7777         kev.preventDefault()
7778     }
7779 }
7780 function GJE_DragEvent(ev,elem){
7781     x = ev.clientX
7782     y = ev.clientY
7783     console.log("Dragged: "+this.nodeName+'#'+this.id+' x='+x+' y='+y)
7784 }
7785 // https://developer.mozilla.org/en-US/docs/Web/API/DragEvent
7786 // https://www.w3.org/TR/uievents/#events-mouseevents
7787 function GJE_DropEvent(ev,elem){
7788     x = ev.clientX
7789     y = ev.clientY
7790     this.style.x = x
7791     this.style.y = y
7792     this.style.position = 'absolute' // 'fixed'
7793     this.parentNode = gsh // just for test
7794     console.log("Dropped: "+this.nodeName+'#'+this.id+' x='+x+' y='+y
7795     + " parent="+this.parentNode.id)
7796 }
7797 function GJE_SetTableStyle(ev){
7798     this.innerHTML = this.value; // sync. for external representation?
7799     if(false){
7800         stid = this.parentNode.id+this.id
7801         // and remove " span" at the end
7802         e = document.getElementById(stid)
7803         //alert('SetTableStyle #' +e.id+'\n'+this.value)
7804         if( e != null ){
7805             e.innerHTML = this.value
7806         }else{
7807             console.log('Style Not found: '+stid)
7808         }
7809     }
7810     //alert('event StopPropagatation: '+ev)
7811 }
7812 }

```

```

7813 function setCSSofClass(cclass,cstyle){
7814     const ss = document.styleSheets[3]; // 0, 1, 2, 3, ... ?
7815     rlen = ss.cssRules.length;
7816     let tabrule = null;
7817     rulex = -1
7818
7819     // should skip white space at the top of cstyle
7820     sel = cstyle.charAt(0);
7821     selector = sel+cclass;
7822     console.log('-- search style rule for '+selector)
7823
7824     for(let i = 0; i < rlen; i++){
7825         cr = ss.cssRules[i];
7826         console.log('CSS rule ['+'i'+rlen+''] '+cr.selectorText);
7827         if( cr.selectorText === selector ){ // css class selector
7828             tabrule = ss.cssRules[i];
7829             console.log('CSS rule found for:['+'i'+rlen+''] '+selector);
7830             ss.deleteRule(i);
7831             //rlen = ss.cssRules.length;
7832             rulex = i
7833             // should search and replace the property here
7834         }
7835     }
7836     // https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleSheet/insertRule
7837     if( tabrule == null ){
7838         console.log('CSS rule NOT found for:['+rlen+''] '+selector);
7839         ss.insertRule(cstyle,rlen);
7840         ss.insertRule(cstyle,0); // override by 0?
7841         console.log('CSS rule inserted:['+(rlen+1)+'']\n'+cstyle);
7842     }else{
7843         ss.insertRule(cstyle,rlen);
7844         ss.insertRule(cstyle,0);
7845         console.log('CSS rule replaced:['+(rlen+1)+'']\n'+cstyle);
7846     }
7847 }
7848 function GJE_SetStyle(te){
7849     console.log('Apply the style to:'+te.id+'\n');
7850     console.log('Apply the style to:'+te.parentNode.id+'\n');
7851     console.log('Apply the style to:'+te.parentNode.class+'\n');
7852     cclass = te.parentNode.class;
7853     setCSSofClass(cclass,te.value); // should get selector part from
7854     // selector { rules }
7855
7856     if(false){
7857         //console.log('Apply the style:')
7858         //stid = this.parentNode.id+this.id+"
7859         //stid = this.id+".style"
7860         css = te.value
7861         stid = te.parentNode.id+".style"
7862         e = document.getElementById(stid)
7863         if( e != null ){
7864             //console.log('Apply the style:'+e.id+'\n'+te.value);
7865             console.log('Apply the style:'+e.id+'\n'+css);
7866             // e.innerHTML = css; //te.value;
7867             //ncss = e.sheet;
7868             //ncss.insertRule(te.value,ncss.cssRules.length);
7869         }else{
7870             console.log('No element to Apply the style: '+stid)
7871         }
7872         tblid = te.parentNode.id+".table";
7873         e = document.getElementById(tblid);
7874         if( e != null ){
7875             //e.setAttribute('style',css);
7876             e.setProperty('style',css,'!important');
7877         }
7878     }
7879 }
7880 function makeTable(argv){
7881     //tid = ''
7882     cwe = GJE_CurElement
7883     tid = 'table_' + GJE_TableNo
7884
7885     nt = new Text('\n')
7886     cwe.appendChild(nt)
7887
7888     ne = document.createElement('span'); // the container
7889     cwe.appendChild(ne)
7890     ne.id = tid + '-span'
7891     ne.setAttribute('contenteditable',true)
7892
7893     htspan = document.createElement('span'); // html part
7894     //htspan.id = tid + '-html'
7895     //ne.innerHTML = '\n'
7896     nt = new Text('\n')
7897     ne.appendChild(nt)
7898     ne.appendChild(htspan)
7899
7900     htspan.id = tid
7901     htspan.setAttribute('class',tid)
7902
7903     ne.setAttribute('draggable','true')
7904     ne.addEventListener('drag',GJE_DragEvent);
7905     ne.addEventListener('dragend',GJE_DropEvent);
7906
7907     var col = 3
7908     var row = 2
7909     if( argv[0] != null ){
7910         col = argv[0]
7911         argv.shift()
7912     }
7913     if( argv[0] != null ){
7914         row = argv[0]
7915         argv.shift()
7916     }
7917
7918     //ne.setAttribute('class',tid)
7919     ht = '\n'
7920     //ht += '<'+table ' + 'id="'+tid+'"' + ' class="'+tid+'"'
7921     ht += '<'+table '
7922     + ' onkeydown="GJE_TableKeyCommand(event,this)"'
7923     //+ ' ondrag="GJE_DragEvent(event,this)"\n'
7924     //+ ' ondragend="GJE_DropEvent(event,this)"\n'
7925     //+ ' draggable="true"\n'
7926     //+ ' contenteditable="true"'
7927     + '>\n'
7928     ht += '<'+tbody>\n';
7929     for( r = 0; r < row; r++ ){
7930         ht += "<"+tr>\n"
7931         for( c = 0; c < col; c++ ){
7932             ht += "<"+td>"
7933             ht += "ABCDEFGHIJKLMNOPQRSTUVWXYZ".charAt(c) + r
7934             ht += "<"+td>\n"
7935         }
7936         ht += "<"+tr>\n"

```



```

7937     }
7938     ht += '<'+'/tbody>\n';
7939     ht += '<'+'/table>\n';
7940     htspan.innerHTML = ht;
7941     nt = new Text('\n')
7942     ne.appendChild(nt)
7943
7944     st = '#'+tid+' *{\n' // # for instanse specific
7945     + 'border:1px solid #aaa;\n'
7946     + 'background-color:#efe;\n'
7947     + 'color:#222;\n'
7948     + 'font-size:#14pt !important;\n'
7949     + 'font-family:monospace,Courier New !important;\n'
7950     +'} /* hit ESC to apply */'\n'
7951
7952     // wish script to be included
7953     //nj = document.createElement('script')
7954     //ne.appendChild(nj)
7955     //ne.innerHTML = 'function SetStyle(e){'
7956
7957     // selector seems lost in dynamic style appending
7958     if(false){
7959         ns = document.createElement('style')
7960         ne.appendChild(ns)
7961         ns.id = tid + '.style'
7962         ns.innerHTML = '\n'+st
7963         nt = new Text('\n')
7964         ne.appendChild(nt)
7965     }
7966     setCSSofClass(tid,st); // should be in JavaScript script?
7967
7968     nx = document.createElement('textarea')
7969     ne.appendChild(nx)
7970     nx.id = tid + '-style_def'
7971     nx.setAttribute('class','GJ_StyleEditor')
7972     nx.spellcheck = false
7973     nx.cols = 60
7974     nx.rows = 10
7975     nx.innerHTML = '\n'+st
7976     nx.addEventListener('change',GJE_SetTableStyle);
7977     nx.addEventListener('keydown',GJE_StyleKeyCommand);
7978     //nx.addEventListener('click',GJE_SetTableStyle);
7979
7980     nt = new Text('\n')
7981     cwe.appendChild(nt)
7982
7983     GJE_TableNo += 1
7984     return 'created TABLE id="'+tid+'"'
7985 }
7986 function GJE_NodeEdit(argv){
7987     cwe = GJE_CurElement
7988     cmd = argv[0]
7989     argv.shift()
7990     args = argv.join(' ')
7991     ret = ""
7992
7993     if( cmd == '.u' || cmd == '.un' || cmd == 'undo' ){
7994         if( GJE_NodeSaved != null ){
7995             xn = GJE_RootNode
7996             GJE_RootNode = GJE_NodeSaved
7997             GJE_NodeSaved = xn
7998             ret = '-- did undo'
7999         }else{
8000             ret = '-- could not undo'
8001         }
8002         return ret
8003     }
8004     GJE_NodeSaved = GJE_RootNode.cloneNode()
8005     if( cmd == '.c' || cmd == '.cd' || cmd == 'cd' ){
8006         if( argv[0] == null ){
8007             ne = GJE_RootNode
8008         }else
8009         if( argv[0] == '..' ){
8010             ne = cwe.parentNode
8011         }else{
8012             ne = document.getElementById(argv[0])
8013         }
8014         if( ne != null ){
8015             GJE_CurElement = ne
8016             ret = "-- current node: " + ne.id
8017         }else{
8018             ret = "-- not found: " + argv[0]
8019         }
8020     }else
8021     if( cmd == '.mkt' || cmd == '.mktable' ){
8022         makeTable(argv)
8023     }else
8024     if( cmd == '.m' || cmd == '.mk' || cmd == 'mk' ){
8025         ne = document.createElement(argv[0])
8026         //ne.id = argv[0]
8027         ret = "-- created " + ne + " under " + cwe.tagName + "#" + cwe.id
8028         cwe.appendChild(ne)
8029         if( cmd == '.m' || cmd == '.mk' ){
8030             GJE_CurElement = ne
8031         }
8032     }else
8033     if( cmd == '.n' || cmd == '.nm' || cmd == 'nm' ){
8034         cwe.id = argv[0]
8035     }else
8036     if( cmd == '.r' || cmd == '.rm' || cmd == 'rm' ){
8037     }else
8038     if( cmd == '.h' || cmd == '.sh' || cmd == 'sh' ){
8039         s = argv.join(' ')
8040         cwe.innerHTML = s
8041     }else
8042     if( cmd == '.a' || cmd == '.sa' || cmd == 'sa' ){
8043         cwe.setAttribute(argv[0],argv[1])
8044     }else
8045     if( cmd == '.l' ){
8046     }else
8047     if( cmd == '.i' || cmd == '.ih' || cmd == 'ih' ){
8048         ret = cwe.innerHTML
8049     }else
8050     if( cmd == '.p' || cmd == '.pw' || cmd == 'pw' ){
8051         ret = cwe.nodeType + " " + cwe.tagName + " " + cwe.id
8052         for( we = cwe.parentNode; we != null; ){
8053             ret += "\n" + " " + we.nodeType + " " + we.tagName + " " + we.id
8054             we = we.parentNode
8055         }
8056     }else
8057     {
8058         ret = "Command: mk | rm \n"
8059         ret += " pw -- print current node\n"
8060         ret += " mk type -- make node with name and type\n"

```

```

8061     ret += "    nm name -- set the id #name of current node\n"
8062     ret += "    rm name -- remove named node\n"
8063     ret += "    cd name -- change current node\n"
8064 }
8065 //alert(ret)
8066 return ret
8067 }
8068 function GJC_Command(text){
8069     lines = text.value.split('\n')
8070     line = lines[lines.length-1]
8071     argv = line.split(' ')
8072     text.value += '\n'
8073     if( argv[0] == '#' ){ argv.shift() }
8074     args0 = argv.join(' ')
8075     cmd = argv[0]
8076     argv.shift()
8077     args = argv.join(' ')
8078
8079     if( cmd == 'nolog' ){
8080         StopConsoleLog = true
8081     }else
8082     if( cmd == 'new' ){
8083         if( argv[0] == 'table' ){
8084             argv.shift()
8085             console.log('argv'+argv)
8086             text.value += makeTable(argv)
8087         }else
8088         if( argv[0] == 'console' ){
8089             text.value += GJ_NewConsole('GJ_Console')
8090         }else{
8091             text.value += '-- new { console | table }'
8092         }
8093     }else
8094     if( cmd == 'strip' ){
8095         //text.value += GJF_StripClass()
8096     }else
8097     if( cmd == 'css' ){
8098         sel = '#table_1'
8099         if(argv[0] == '0'){
8100             rule1 = sel+'{color:#000 !important; background-color:#fff !important;}';
8101         }else
8102         rule1 = sel+'{color:#f00 !important; background-color:#eef !important;}';
8103         document.styleSheets[3].deleteRule(0);
8104         document.styleSheets[3].insertRule(rule1,0);
8105         text.value += 'CSS rule added: '+rule1
8106     }else
8107     if( cmd == 'print' ){
8108         e = null;
8109         if( e == null ){
8110             e = document.getElementById('GJFactory_0')
8111         }
8112         if( e == null ){
8113             e = document.getElementById('GJFactory_1')
8114         }
8115         if( argv[0] != null ){
8116             id = argv[0]
8117             if( id == 'f' ){
8118                 //e = document.getElementById('GJE_RootNode');
8119             }else{
8120                 e = document.getElementById(id)
8121             }
8122             if( e != null ){
8123                 text.value += e.outerHTML
8124             }else{
8125                 text.value += "Not found: " + id
8126             }
8127         }else{
8128             text.value += GJE_RootNode.outerHTML
8129             //text.value += e.innerHTML
8130         }
8131     }else
8132     if( cmd == 'destroy' ){
8133         text.value += GJFactory_Destroy()
8134     }else
8135     if( cmd == 'save' ){
8136         e = document.getElementById('GJFactory')
8137         Permanent.setItem('GJFactory-1',e.innerHTML)
8138         text.value += "-- Saved GJFactory"
8139     }else
8140     if( cmd == 'load' ){
8141         gjf = Permanent.getItem('GJFactory-1')
8142         e = document.getElementById('GJFactory')
8143         e.innerHTML = gjf
8144         // must restore EventListener
8145         text.value += "-- EventListener was not restored"
8146     }else
8147     if( cmd.charAt(0) == '.' ){
8148         argv0 = args0.split(' ')
8149         text.value += GJE_NodeEdit(argv0)
8150     }else
8151     if( cmd == 'cont' ){
8152         bannerIsStopping = false
8153         GshMenuStop.innerHTML = "Stop"
8154     }else
8155     if( cmd == 'date' ){
8156         text.value += DateLong()
8157     }else
8158     if( cmd == 'echo' ){
8159         text.value += args
8160     }else
8161     if( cmd == 'fork' ){
8162         html_fork()
8163     }else
8164     if( cmd == 'last' ){
8165         text.value += MyHistory
8166         //h = document.createElement("span")
8167         //h.innerHTML = MyHistory
8168         //text.value += h.innerHTML
8169         //tx = MyHistory.replace("\n","")
8170         //text.value += tx.replace("<+>br>","\n") + "xxxx<+>br>yyyy"
8171     }else
8172     if( cmd == 'ne' ){
8173         text.value += GJE_NodeEdit(argv)
8174     }else
8175     if( cmd == 'reload' ){
8176         location.reload()
8177     }else
8178     if( cmd == 'mem' ){
8179         text.value += GJC_Memory('GJC_Storage',args,text)
8180     }else
8181     if( cmd == 'stop' ){
8182         bannerIsStopping = true
8183         GshMenuStop.innerHTML = "Start"
8184     }else

```

```

8185     if( cmd == 'who' ){
8186         text.value += "SessionId="+GJC_SessionId+" "+document.URL
8187     }else
8188     if( cmd == 'wall' ){
8189         text.value += GJC_Memory('GJC_Wall','write',text)
8190     }else
8191     {
8192         text.value += "Commands: help | echo | date | last \n"
8193         + '      new | save | load | mem \n'
8194         + '      who | wall | fork | nife'
8195     }
8196 }
8197
8198 function GJC_Input(){
8199     if( this.value.endsWith("\n") ){ // remove NL added by textarea
8200         this.value = this.value.slice(0,this.value.length-1)
8201     }
8202 }
8203
8204 var GJC_Id = null
8205 function GJC_Resize(){
8206     GJC_Id.style.zindex = 20000
8207     GJC_Id.style.width = window.innerWidth - 16
8208     GJC_Id.style.height = 300
8209     GJC_Id.style.backgroundColor = "rgba(0,64,16,1.0)" // blackboard color
8210     GJC_Id.style.color = "rgba(255,255,255,1.0)"
8211 }
8212 function GJC_FocusIn(){
8213     this.spellcheck = false
8214     SuppressGJShell = true
8215     this.onkeydown = GJC_Keydown
8216     GJC_Resize()
8217 }
8218 function GJC_FocusOut(){
8219     SuppressGJShell = false
8220     this.removeEventListener('keydown',GJC_Keydown);
8221 }
8222 window.addEventListener('resize',GJC_Resize);
8223
8224 function GJC_OnStorage(e){
8225     //alert('Got Message')
8226     //GJC.value += "\n((ReceivedMessage))\n"
8227 }
8228 window.addEventListener('storage',GJC_OnStorage);
8229 //window.addEventListener('storage',()=>{alert('GotMessage')})
8230
8231 function GJC_Setup(gjcId){
8232     gjcId.style.width = gsh.getBoundingClientRect().width
8233     gjcId.value = "GJShell Console // " + GshVersion.innerHTML + "\n"
8234     //gjcId.value += "Date: " + DateLong() + "\n"
8235     gjcId.value += PS1
8236     gjcId.onfocus = GJC_FocusIn
8237     gjcId.addEventListener('input',GJC_Input);
8238     gjcId.addEventListener('focusout',GJC_FocusOut);
8239     GJC_Id = gjcId
8240 }
8241 function GJC_Clear(id){
8242 }
8243 if( document.getElementById("GJC_0") != null ){
8244     GJC_Setup(GJC_0)
8245 }else{
8246     document.write('<'+`textarea id="GJC_1" class="GJConsole"><'+`/textarea>')
8247     GJC_Setup(GJC_1)
8248     factory = document.createElement('span');
8249     gsh.appendChild(factory)
8250     GJE_RootNode = factory;
8251     GJE_CurElement = GJE_RootNode;
8252 }
8253
8254 // TODO: focus handling
8255 </script>
8256 <style>
8257 .GJ_StyleEditor {
8258     font-size:9pt !important;
8259     font-family:Courier New, monospace !important;
8260 }
8261 </style>
8262
8263 </details>
8264 </span>
8265 <!-- ----- GJConsole END ] ----- -->
8266 */
8267
8268 /*
8269 <span id="BlinderText">
8270 <style id="BlinderTextStyle">
8271 #GJLinkView {
8272     xposition:absolute; z-index:5000;
8273     position:relative;
8274     display:block;
8275     left:8px;
8276     color:#fff;
8277     width:800px; height:300px; resize:both;
8278     margin:0px; padding:4px;
8279     background-color:rgba(200,200,200,0.5) !important;
8280 }
8281 .MssgText {
8282     width:578px !important;
8283     resize:both !important;
8284     color:#000 !important;
8285 }
8286 .GJNote {
8287     font-family:Georgia !important;
8288     font-size:13pt !important;
8289     color:#22a !important;
8290 }
8291 .textField {
8292     display:inline;
8293     border:0.5px solid #444;
8294     border-radius:3px;
8295     color:#000; background-color:#fff;
8296     width:106pt; height:18pt;
8297     margin:2px;
8298     padding:2px;
8299     resize:none;
8300     vertical-align:middle;
8301     font-size:10pt; font-family:Courier New;
8302 }
8303 .textLabel {
8304     border:0px solid #000 !important;
8305     background-color:rgba(0,0,0,0);
8306 }
8307 .textURL {
8308     width:300pt !important;

```

```

8309     border:0px solid #000 !important;
8310     background-color:rgba(0,0,0,0);
8311 }
8312 .VisibleText {
8313 }
8314 .BlinderText {
8315     color:#000; background-color:#eee;
8316 }
8317 .joinButton {
8318     font-family:Georgia !important;
8319     font-size:11pt;
8320     line-height:1.1;
8321     height:18pt;
8322     width:50pt;
8323     padding:3px !important;
8324     text-align:center !important;
8325     border-color:#aaa !important;
8326     border-radius:5px;
8327     color:#fff; background-color:#4a4 !important;
8328     vertical-align:middle !important;
8329 }
8330 .SendButton {
8331     vertical-align:top;
8332 }
8333 .ws0_log {
8334     font-size:10pt;
8335     color:#000 !important;
8336     line-height:1.0;
8337     background-color:rgba(255,255,255,0.7) !important;
8338     font-family:Courier New,monospace !important;
8339     width:99.3%;
8340     white-space:pre;
8341 }
8342 </style>
8343
8344 <!-- Form autofill test
8345 Location: <input id="xxserv" type="text" value="https://192.168.10.1/boafirm/formLogin" size="80">
8346 <form method="POST" id="xxform" action="https://192.168.10.1/boafirm/formLogin">
8347 dest? <input id="XDS" name="dest" type="text" size="80" value="/index_contents.html">
8348 -->
8349 <details><summary>Form Auto. Filling</summary>
8350 <style>
8351 .xxinput { width:260pt !important; line-height:1.1 !important; margin:1px;
8352     display:inline !important; font-size:10pt !important; padding:1px !important;
8353 }
8354 </style>
8355 <span style="font-family:Courier New;color:black;font-size:12pt;" onactive="">
8356 <form method="POST" id="xxform" action="https://192.168.10.1/" style="white-space:pre;">
8357 Location: <input id="xxserv" class="xxinput" type="text" value="https://192.168.10.1/" size="80">
8358 Username: <input id="xxuser" class="xxinput" name="user" type="text" autocomplete="on">
8359 Password: <input id="xxpass" class="xxinput" name="pass" type="password" autocomplete="on">
8360 SessionId: <input id="xxssid" class="xxinput" name="SESSION_ID" type="text" size="80">
8361 <input id="xxsubm" class="xxinput" type="submit" value="Submit"></form>
8362 </span>
8363 <script>
8364     function XXSetFormAction(){
8365         xxform.setAttribute('action',xxserv.value);
8366     }
8367     xxform.setAttribute('action',xxserv.value);
8368     xxserv.addEventListener('change',XXSetFormAction);
8369     //xxserv.value = location.href;
8370 </script>
8371 </details>
8372 */
8373
8374 /*
8375 <details id="BlinderTextClass" class="gsh-src"><summary>BlinderText</summary>
8376 <span id="BlinderTextScript">
8377 // https://w3c.github.io/uevents/#event-type-keydown
8378 //
8379 // 2020-09-21 class BlinderText - textarea element not to be readable
8380 //
8381 // BlinderText attributes
8382 // bl_plainText - null
8383 // bl_hideChecksum - [false]
8384 // bl_showLength - [false]
8385 // bl_visible - [false]
8386 // data-bl_config - []
8387 // - min. length
8388 // - max. length
8389 // - acceptable charset in generate text
8390 //
8391 function BlinderChecksum(text){
8392     plain = text.bl_plainText;
8393     return strCRC32(plain,plain.length).toFixed(0);
8394 }
8395 function BlinderKeydown(ev){
8396     pass = ev.target
8397     if( ev.code == 'Enter' ){
8398         ev.preventDefault();
8399     }
8400     ev.stopPropagation()
8401 }
8402 function BlinderKeyUp1(ev){
8403     blind = ev.target
8404     if( ev.code == 'Backspace' ){
8405         blind.bl_plainText = blind.bl_plainText.slice(0,blind.bl_plainText.length-1)
8406     }else
8407     if( and(ev.code == 'KeyV', ev.ctrlKey) ){
8408         blind.bl_visible = !blind.bl_visible;
8409     }else
8410     if( and(ev.code == 'KeyL', ev.ctrlKey) ){
8411         blind.bl_showLength = !blind.bl_showLength;
8412     }else
8413     if( and(ev.code == 'KeyU', ev.ctrlKey) ){
8414         blind.bl_plainText = "";
8415     }else
8416     if( and(ev.code == 'KeyR', ev.ctrlKey) ){
8417         checksum = BlinderChecksum(blind);
8418         blind.bl_plainText = checksum; //.toString(32);
8419     }else
8420     if( ev.code == 'Enter' ){
8421         ev.stopPropagation();
8422         ev.preventDefault();
8423         return;
8424     }else
8425     if( ev.key.length != 1 ){
8426         console.log('KeyUp: '+ev.code+' '+ev.key);
8427         return;
8428     }else{
8429         blind.bl_plainText += ev.key;
8430     }
8431     leng = blind.bl_plainText.length;
8432

```

```

8433 //console.log('KeyUp: '+ev.code+'/'+blind.bl_plainText);
8434 checksum = BlinderChecksum(blind) % 10; // show last one digit only
8435
8436 visual = '';
8437 if( !blind.bl_hideChecksum || blind.bl_showLength ){
8438     visual += '[';
8439 }
8440 if( !blind.bl_hideChecksum ){
8441     visual += '#'+checksum.toString(10);
8442 }
8443 if( blind.bl_showLength ){
8444     visual += '/' + leng;
8445 }
8446 if( !blind.bl_hideChecksum || blind.bl_showLength ){
8447     visual += ']' ;
8448 }
8449 if( blind.bl_visible ){
8450     visual += blind.bl_plainText;
8451 }else{
8452     visual += '*'.repeat(leng);
8453 }
8454 blind.value = visual;
8455 }
8456 function BlinderKeyUp(ev){
8457     BlinderKeyUp1(ev);
8458     ev.stopPropagation();
8459 }
8460 // https://w3c.github.io/uievents/#keyboardevent
8461 // https://w3c.github.io/uievents/#uievent
8462 // https://dom.spec.whatwg.org/#event
8463 function BlinderTextEvent(){
8464     ev = event;
8465     blind = ev.target;
8466     console.log('Event '+ev.type+'@'+blind.nodeName+'#'+blind.id)
8467     if( ev.type == 'keyup' ){
8468         BlinderKeyUp(ev);
8469     }else
8470     if( ev.type == 'keydown' ){
8471         BlinderKeyDown(ev);
8472     }else{
8473         console.log('thru-event '+ev.type+'@'+blind.nodeName+'#'+blind.id)
8474     }
8475 }
8476 << textarea hidden id="BlinderTextClassDef" class="textField"
8477 // onkeydown="BlinderTextEvent()" onkeyup="BlinderTextEvent()"
8478 // spellcheck="false"></textarea>
8479 << textarea hidden id="gj_pass1"
8480 // class="textField BlinderText"
8481 // placeholder="PassWord1"
8482 // onkeydown="BlinderTextEvent()"
8483 // onkeyup="BlinderTextEvent()"
8484 // spellcheck="false"</textarea>
8485 function SetupBlinderText(parent,txa,phold){
8486     if( txa == null ){
8487         txa = document.createElement('textarea');
8488         //txa.id = id;
8489     }
8490     txa.setAttribute('class','textField BlinderText');
8491     txa.setAttribute('placeholder',phold);
8492     txa.setAttribute('onkeydown','BlinderTextEvent()');
8493     txa.setAttribute('onkeyup','BlinderTextEvent()');
8494     txa.setAttribute('spellcheck','false');
8495     //txa.setAttribute('bl_plainText','false');
8496     txa.bl_plainText = '';
8497     //parent.appendChild(txa);
8498 }
8499 function DestroyBlinderText(txa){
8500     txa.removeAttribute('class');
8501     txa.removeAttribute('placeholder');
8502     txa.removeAttribute('onkeydown');
8503     txa.removeAttribute('onkeyup');
8504     txa.removeAttribute('spellcheck');
8505     txa.bl_plainText = '';
8506 }
8507 //
8508 // visible textarea like Username
8509 //
8510 function VisibleTextEvent(){
8511     if( event.code == 'Enter' ){
8512         if( event.target.NoEnter ){
8513             event.preventDefault();
8514         }
8515     }
8516     event.stopPropagation();
8517 }
8518 function SetupVisibleText(parent,txa,phold){
8519     if( false ){
8520         txa.setAttribute('class','textField VisibleText');
8521     }else{
8522         newclass = txa.getAttribute('class');
8523         if( and(newclass != null, newclass != '') ){
8524             newclass += ' ';
8525         }
8526         newclass += 'VisibleText';
8527         txa.setAttribute('class',newclass);
8528     }
8529     //console.log('SetupVisibleText class='+txa.class);
8530     txa.setAttribute('placeholder',phold);
8531     txa.setAttribute('onkeydown','VisibleTextEvent()');
8532     txa.setAttribute('onkeyup','VisibleTextEvent()');
8533     txa.setAttribute('spellcheck','false');
8534     cols = txa.getAttribute('cols');
8535     if( cols != null ){
8536         txa.style.width = '580px';
8537     //console.log('VisualText#'+txa.id+' cols='+cols)
8538     }else{
8539     //console.log('VisualText#'+txa.id+' NO cols')
8540     }
8541     rows = txa.getAttribute('rows');
8542     if( rows != null ){
8543         txa.style.height = '30px';
8544         txa.style.resize = 'both';
8545         txa.NoEnter = false;
8546     }else{
8547         txa.NoEnter = true;
8548     }
8549 }
8550 function DestroyVisibleText(txa){
8551     txa.removeAttribute('class');
8552     txa.removeAttribute('placeholder');
8553     txa.removeAttribute('onkeydown');
8554     txa.removeAttribute('onkeyup');
8555     txa.removeAttribute('spellcheck');
8556     cols = txa.removeAttribute('cols');

```

```

8557 }
8558 </span>
8559 <script>
8560 js = document.getElementById('BlinderTextScript');
8561 eval(js.innerHTML);
8562 //js.outerHTML = ""
8563 </script>
8564
8565 </details>
8566 </span>
8567 */
8568
8569 /*
8570 <script id="GJLinkScript">
8571 function gjkey_hash(text){
8572     return strCRC32(text,text.length) % 0x10000;
8573 }
8574 function gj_addlog(e,msg){
8575     now = (new Date().getTime() / 1000).toFixed(3);
8576     tstp = '['+now+' ]
8577     e.value += tstp + msg;
8578     e.scrollTop = e.scrollHeight;
8579 }
8580 function gj_addlog_cl(msg){
8581     ws0_log.value += '(console.log) ' + msg + '\n';
8582 }
8583 var GJ_Channel = null;
8584 var GJ_Log = null;
8585 var gjx; // the global variable
8586 function GJ_Join(){
8587     target = gj_join;
8588     if( target.value == 'Leave' ){
8589         GJ_Channel.close();
8590         GJ_Channel = null;
8591         target.value = 'Join';
8592     }
8593 }
8594
8595 var ws0;
8596 var ws0_log;
8597
8598 sav_console_log = console.error
8599 console.error = gj_addlog_cl
8600 ws0 = new WebSocket(gj_serv.innerHTML);
8601 console.error = sav_console_log
8602
8603 GJ_Channel = ws0;
8604 ws0_log = document.getElementById('ws0_log');
8605 GJ_Log = ws0_log;
8606
8607 now = (new Date().getTime() / 1000).toFixed(3);
8608 const wsstats = ["CONNECTING","OPEN","CLOSING","CLOSED"];
8609 cst = wsstats[ws0.readyState];
8610 gj_addlog(ws0_log,'stat '+ws0.readyState+'('+cst+') : GJ Linked\n');
8611
8612 ws0.addEventListener('error', function(event){
8613     gj_addlog(ws0_log,'stat error : transport error?\n');
8614 });
8615 ws0.addEventListener('open', function(event){
8616     GJLinkView.style.zIndex = 10000;
8617     //console.log('#'+GJLinkView.id+'.zIndex='+GJLinkView.style.zIndex);
8618     date1 = new Date().getTime();
8619     date2 = (date1 / 1000).toFixed(3);
8620     seed = date1.toString(16);
8621
8622     // user name and key
8623     user = document.getElementById('gj_user').value;
8624     if( user.length == 0 ){
8625         gj_user.value = 'nemo';
8626         user = 'nemo';
8627     }
8628     key1 = document.getElementById('gj_ukey').bl_plainText;
8629     ukey = gjkey_hash(seed+user+key1).toString(16);
8630
8631     // session name and key
8632     chan = document.getElementById('gj_chan').value;
8633     if( chan.length == 0 ){
8634         gj_chan.value = 'main';
8635         chan = 'main';
8636     }
8637     key2 = document.getElementById('gj_ckey').bl_plainText;
8638     ckey = gjkey_hash(seed+chan+key2).toString(16);
8639
8640     msg = date2 + ' JOIN ' + user + '|' + chan + ' ' + ukey + ':' + ckey;
8641     gj_addlog(ws0_log,'send '+msg+'\n');
8642     ws0.send(msg);
8643
8644     target.value = 'Leave';
8645     //console.log(['+date2+'] '#'+target.id+' '+target.value+'\n');
8646     //gj_addlog(ws0_log,'label '+target.value+'\n');
8647 });
8648 ws0.addEventListener('message', function(event){
8649     now = (new Date().getTime() / 1000).toFixed(3);
8650     msg = event.data;
8651     gj_addlog(ws0_log,'recv '+msg+'\n');
8652
8653     argv = msg.split(' ');
8654     tstamp = argv[0];
8655     argv.shift();
8656     if( argv[0] == 'reload' ){
8657         location.reload()
8658     }
8659     argv.shift(); // command
8660     argv.shift(); // from|to
8661     if( argv[0] == 'auth' ){
8662         // doing authorization required
8663     }
8664     if( argv[0] == 'echo' ){
8665         now = (new Date().getTime() / 1000).toFixed(3);
8666         msg = now+' '+RESP '+argv.join(' ');
8667         gj_addlog(ws0_log,'send '+msg+'\n');
8668         ws0.send(msg);
8669     }
8670     if( argv[0] == 'eval' ){
8671         argv.shift();
8672         js = argv.join(' ');
8673         ret = eval(js); // <----- eval()
8674         gj_addlog(ws0_log,'eval '+js+' = '+ret+'\n');
8675         now = (new Date().getTime() / 1000).toFixed(3);
8676         msg = now + ' ' + RESP + ret;
8677         ws0.send(msg);
8678         gj_addlog(ws0_log,'send '+msg+'\n')
8679     }
8680 });

```

```

8681 ws0.addEventListener('close', function(event){
8682     if( GJ_Channel == null ){
8683         gj_addlog(ws0_log,'stat OK : GJ UnLinked\n');
8684         return;
8685     }
8686     GJ_Channel.close();
8687     GJ_Channel = null;
8688     target.value = 'Join';
8689     gj_addlog(ws0_log,'stat error : close : GJ UnLinked unexpectedly\n');
8690 });
8691 }
8692 function GJ_Send(){
8693     if( GJ_Channel == null ){
8694         gj_addlog(ws0_log,'stat error : send : GJ not Linked\n');
8695         return;
8696     }
8697     target = event.target;
8698     user = document.getElementById('gj_user').value;
8699     chan = document.getElementById('gj_chan').value;
8700     now = (new Date()).getTime() / 1000).toFixed(3);
8701     msg = now + " ISAY "+user+" | "+chan+" "+gj_sendText.value;
8702     gj_addlog(GJ_Log,'send '+msg+'\n');
8703     GJ_Channel.send(msg);
8704 }
8705 </script>
8706
8707 <!-- ----- GJLINK ----- -->
8708 <!--
8709 - User can subscribe to a channel
8710 - A channel will be broadcasted
8711 - A channel can be a pattern (regular expression)
8712 - User is like From:(me) and channel is like To: or Recipient:
8713 - like VIABUS
8714 - watch message with SENDME, WATCH, CATCH, HEAR, or so
8715 - routing with path expression or name pattern (with routing with DNS like system)
8716 -->
8717 */
8718
8719 <<span id="GJLinkGolang">
8720 <<details id="GshWebSocket" class="gsh-src"><summary>Golang / JavaScript Link</summary>
8721 // 2020-0920 created
8722 <<a href="https://pkg.go.dev/golang.org/x/net/websocket">WS</a>
8723 <<a href="https://gdoc.org/golang.org/x/net/websocket">WS</a>
8724 // INSTALL: go get golang.org/x/net/websocket
8725 // INSTALL: sudo {apt,yum} install git (if git is not installed yet)
8726 // import "golang.org/x/net/websocket"
8727 const gshws_origin = "http://localhost:9999"
8728 const gshws_server = "localhost:9999"
8729 const gshws_port = 9999
8730 const gshws_path = "gjlink1"
8731 const gshws_url = "ws://" + gshws_server + "/" + gshws_path
8732 const GSHWS_MSGSIZE = (8*1024)
8733 func fmtstring(fmts string, params ...interface{})(string){
8734     return fmt.Sprintf(fmts,params...)
8735 }
8736 func GSHWS_MARK(what string)(string){
8737     now := time.Now()
8738     us := fmtstring("%06d",now.Nanosecond() / 1000)
8739     mark := ""
8740     if( !AtConsoleLineTop ){
8741         mark += "\n"
8742         AtConsoleLineTop = true
8743     }
8744     mark += "[" + now.Format(time.Stamp) + "." + us + "] -GJ- " + what + ": "
8745     return mark
8746 }
8747 func gchk(what string,err error){
8748     if( err != nil ){
8749         panic(GSHWS_MARK(what)+err.Error())
8750     }
8751 }
8752 func glog(what string, fmts string, params ...interface{}){
8753     fmt.Print(GSHWS_MARK(what))
8754     fmt.Printf(fmts+"\n",params...)
8755 }
8756
8757 var WSV = []*websocket.Conn{}
8758 func jsend(argv []string){
8759     if len(argv) <= 1 {
8760         fmt.Printf("--Ij %v [-m] command arguments\n",argv[0])
8761         return
8762     }
8763     argv = argv[1:]
8764     if( len(WSV) == 0 ){
8765         fmt.Printf("--Ej-- No link now\n")
8766         return
8767     }
8768     if( 1 < len(WSV) ){
8769         fmt.Printf("--Ij-- multiple links (%v)\n",len(WSV))
8770     }
8771
8772     multicast := false // should be filtered with regexp
8773     if( 0 < len(argv) && argv[0] == "-m" ){
8774         multicast = true
8775         argv = argv[1:]
8776     }
8777     args := strings.Join(argv, " ")
8778
8779     now := time.Now()
8780     msec := now.UnixNano() / 1000000;
8781     tstamp := fmtstring("%.3f",float64(msec)/1000.0)
8782     msg := fmtstring("%v SEND gshell|* %v",tstamp,args)
8783
8784     if( multicast ){
8785         for i,ws := range WSV {
8786             wn,werr := ws.Write([]byte(msg))
8787             if( werr != nil ){
8788                 fmt.Printf("[%v] wn=%v, werr=%v\n",i,wn,werr)
8789             }
8790             glog("SQ",fmtstring("(%v) %v",wn,msg))
8791         }
8792     }else{
8793         i := 0
8794         ws := WSV[i]
8795         wn,werr := ws.Write([]byte(msg))
8796         if( werr != nil ){
8797             fmt.Printf("[%v] wn=%v, werr=%v\n",i,wn,werr)
8798         }
8799         glog("SQ",fmtstring("(%v) %v",wn,msg))
8800     }
8801 }
8802 func serv1(ws *websocket.Conn) {
8803     WSV = append(WSV,ws)
8804     //fmt.Print("\n")

```

```

8805 glog("CO", "accepted connections[%v]", len(WSV))
8806 //remoteAddr := ws.RemoteAddr
8807 //fmt.Printf("--- accepted %v\n", remoteAddr)
8808 //fmt.Printf("--- accepted %v\n", ws.Config())
8809 //fmt.Printf("--- accepted %v\n", ws.Config().Header)
8810 //fmt.Printf("--- accepted %v // %v\n", ws, serv1)
8811
8812 var reqb = make([]byte, GSHWS_MSGSIZE)
8813 for {
8814     rn, rerr := ws.Read(reqb)
8815     if( rerr != nil || rn < 0 ){
8816         glog("SQ", fmtstring("(%v,%v)", rn, rerr))
8817         break
8818     }
8819     req := string(reqb[0:rn])
8820     glog("SQ", fmtstring("(%v) %v", rn, req))
8821
8822     margv := strings.Split(req, " ");
8823     margv = margv[1:]
8824     if( 0 < len(margv) ){
8825         if( margv[0] == "RESP" ){
8826             // should forward to the destination
8827             continue;
8828         }
8829     }
8830     now := time.Now()
8831     msec := now.UnixNano() / 1000000;
8832     tstamp := fmtstring("%.3f", float64(msec)/1000.0)
8833     res := fmtstring("%v "+ "CAST"+ " %v", tstamp, req)
8834     wn, werr := ws.Write([]byte(res))
8835     gchk("SE", werr)
8836     glog("SR", fmtstring("(%v) %v", wn, string(res)))
8837 }
8838 glog("SF", "WS response finish")
8839
8840 wsv := []*websocket.Conn{}
8841 wsv = 0
8842 for i, v := range WSV {
8843     if( v != ws ){
8844         wsx = i
8845         wsv = append(wsv, v)
8846     }
8847 }
8848 WSV = wsv
8849 //glog("CO", "closed %v", ws)
8850 glog("CO", "closed connection [%v/%v]", wsx+1, len(WSV)+1)
8851 ws.Close()
8852 }
8853 // url := [scheme://]host[:port]/[path]
8854 func decomp_URL(url string){
8855 }
8856 func full_wsURL(){
8857 }
8858 func gj_server(argv []string) {
8859     gjserv := gshws_url
8860     gjport := gshws_server
8861     gjpath := gshws_path
8862     gjscheme := "ws"
8863
8864     //cmd := argv[0]
8865     argv = argv[1:]
8866     if( 1 <= len(argv) ){
8867         serv := argv[0]
8868         if( 0 < strings.Index(serv, "://") ){
8869             schemev := strings.Split(serv, "://")
8870             gjscheme = schemev[0]
8871             serv = schemev[1]
8872         }
8873         if( 0 < strings.Index(serv, "/") ){
8874             pathv := strings.Split(serv, "/" )
8875             serv = pathv[0]
8876             gjpath = pathv[1]
8877         }
8878         servv := strings.Split(serv, ":")
8879         host := "localhost"
8880         port := 9999
8881         if( servv[0] != "" ){
8882             host = servv[0]
8883         }
8884         if( len(servv) == 2 ){
8885             fmt.Sscanf(servv[1], "%d", &port)
8886         }
8887         //glog("LC", "hostport=%v (%v : %v)", servv, host, port)
8888         gjport = fmt.Sprintf("%v:%v", host, port)
8889         gjserv = gjscheme + "://" + gjport + "/" + gjpath
8890     }
8891     glog("LS", fmtstring("listening at %v", gjserv))
8892     http.Handle("/"+gjpath, websocket.Handler(serv1))
8893     err := error(nil)
8894     if( gjscheme == "wss" ){
8895         // https://golang.org/pkg/net/http/#ListenAndServeTLS
8896         //err = http.ListenAndServeTLS(gjport, nil)
8897     }else{
8898         err = http.ListenAndServe(gjport, nil)
8899     }
8900     gchk("LE", err)
8901 }
8902
8903 func gj_client(argv []string) {
8904     glog("CS", fmtstring("connecting to %v", gshws_url))
8905     ws, err := websocket.Dial(gshws_url, "", gshws_origin)
8906     gchk("C", err)
8907
8908     var resb = make([]byte, GSHWS_MSGSIZE)
8909     for qi := 0; qi < 3; qi++ {
8910         req := fmtstring("Hello, GShell! (%v)", qi)
8911         wn, werr := ws.Write([]byte(req))
8912         glog("QM", fmtstring("(%v) %v", wn, req))
8913         gchk("QE", werr)
8914         rn, rerr := ws.Read(resb)
8915         gchk("RE", rerr)
8916         glog("RM", fmtstring("(%v) %v", rn, string(resb)))
8917     }
8918     glog("CF", "WS request finish")
8919 }
8920 //</details></span>
8921
8922 /*
8923 <details><summary>GJ Link</summary>
8924 <span id="GJLinkView" class="GJLinkView">
8925 <p>
8926 <note class="GJnote">Execute command "gsh gj server" on the localhost and push the Join button:</note>
8927 </p>
8928 <p>

```



```

8929 <span id="GJLink_1">
8930 <script id="gj_xxx1_gen">
8931 if( document.getElementById('gj_serv')== null ){ // executed twice??
8932 document.write('<'+span id="gj_serv_label" class="textField textLabel">Server: <'+/span>');
8933 document.write('<'+span id="gj_serv" class="textField textURL" contenteditable><'+/span>');
8934 }
8935 </script>
8936 <br>
8937 <input id="gj_join" type="button" class="joinButton" onclick="GJ_Join()" value="Join">
8938 <script id="gj_xxx2_gen">
8939 if( true ){
8940 document.write('<'+textarea id="gj_user" class="textField"><'+/textarea>');
8941 document.write('<'+textarea id="gj_ukey" class="textField"><'+/textarea>');
8942 document.write('<'+textarea id="gj_chan" class="textField"><'+/textarea>');
8943 document.write('<'+textarea id="gj_ckey" class="textField"><'+/textarea>');
8944 }
8945 </script>
8946 <br>
8947 <input id="gj_sendButton" type="button" class="joinButton SendButton" onclick="GJ_Send()" value="Send" data-bodyid="gj_sendText">
8948 <script id="gj_sendText_gen">
8949 if( true ){
8950 document.write('<'+textarea id="gj_sendText" class="textField MssgText" cols=60 rows=2><'+/textarea>');
8951 }
8952 </script>
8953 </span></p>
8954 <p>
8955 <script id="ws0_log_gen">
8956 if( true ){
8957 document.write('<'+textarea id="ws0_log" class="ws0_log"
8958 + ' cols=100 rows=10 spellcheck="false"><'+/textarea>');
8959 }
8960 </script>
8961 </p>
8962 </span>
8963 <script>
8964 function SetupGJLink(){
8965 SetupVisibleText(GJLink_1,gj_serv,'GJLinkSv');
8966 SetupVisibleText(GJLink_1,gj_user,'UserName');
8967 SetupBlinderText(GJLink_1,gj_ukey,'UserKey');
8968 SetupVisibleText(GJLink_1,gj_chan,'ChannelName');
8969 SetupBlinderText(GJLink_1,gj_ckey,'ChannelKey');
8970 SetupVisibleText(GJLink_1,gj_sendText,'Message');
8971 gj_serv.innerHTML = 'ws://localhost:9999/gjlink1'
8972 }
8973 SetupGJLink();
8974 function iselem(eid){
8975 return document.getElementById(eid);
8976 }
8977 function DestroyGJLink1(){
8978 if( iselem('gj_serv_label') ) gj_user.parentNode.removeChild(gj_serv_label);
8979 if( iselem('gj_serv') ) gj_user.parentNode.removeChild(gj_serv);
8980 if( iselem('gj_user') ) gj_user.parentNode.removeChild(gj_user);
8981 if( iselem('gj_ukey') ) gj_ukey.parentNode.removeChild(gj_ukey);
8982 if( iselem('gj_chan') ) gj_chan.parentNode.removeChild(gj_chan);
8983 if( iselem('gj_ckey') ) gj_ckey.parentNode.removeChild(gj_ckey);
8984 if( iselem('gj_sendText') ) gj_sendText.parentNode.removeChild(gj_sendText);
8985 if( iselem('ws0_log') ) ws0_log.parentNode.removeChild(ws0_log);
8986 }
8987 DestroyGJLink = DestroyGJLink1;
8988 </script>
8989 </details>
8990 */
8991
8992 /*
8993 <script id="HtmlCodeview-script">
8994 function showHtmlCode(otxa,code){
8995 if( event.target.value == 'ShowCode' ){
8996 txa = document.createElement('textarea');
8997 txa.id = otxa.id;
8998 txa.setAttribute('class','HtmlCodeviewText');
8999 otxa.parentNode.replaceChild(txa,otxa);
9000 txa.setAttribute('spellcheck','false');
9001 txa.value = code.innerHTML;
9002 txa.style.display = "block";
9003 txa.style.width = "100%";
9004 txa.style.height = "300px";
9005 event.target.value = 'HideCode';
9006 }else{
9007 txa.style.display = "none";
9008 event.target.value = 'ShowCode';
9009 }
9010 }
9011 </script>
9012 <style id="HtmlCodeview-style">
9013 .HtmlCodeviewText {
9014 font-size:10pt;
9015 font-family:Courier New;
9016 white-space:pre;
9017 }
9018 .HtmlCodeviewButton {
9019 font-size:11pt;
9020 line-height:1.2;
9021 font-family:Georgia;
9022 border-radius:3px;
9023 color:#ddd; background-color:#333;
9024 }
9025 </style>
9026 */
9027
9028 /*
9029 <details><summary>Live HTML Snapshot</summary>
9030 <span id="LiveHTML">
9031 <!-- ----- HTML Snapshot: Edit, save and load // 2020-0924 SatoxITS { -->
9032 <input class="HtmlCodeviewButton" type="button" value="ShowCode" onclick="showLiveHTMLCode()">
9033 <span id="LiveHTML_Codeview"></span>
9034 <script id="LiveHTMLScript">
9035 function showLiveHTMLCode(){
9036 showHtmlCode(LiveHTML_Codeview,LiveHTML);
9037 }
9038 var _editable = false;
9039 var savSuppressGJShell = false;
9040 function ToggleEditMode(){
9041 _editable = !_editable;
9042 if( _editable ){
9043 savSuppressGJShell = SuppressGJShell;
9044 SuppressGJShell = true;
9045 gsh.setAttribute('contenteditable','true');
9046 GshMenuEdit.innerHTML = 'Lock';
9047 GshMenuEdit.style.color = 'rgba(255,0,0,1)';
9048 GshMenuEdit.style.backgroundColor = 'rgba(255,255,255,1)';
9049 }else{
9050 SuppressGJShell = savSuppressGJShell;
9051 gsh.setAttribute('contenteditable','false');
9052 GshMenuEdit.innerHTML = 'Edit';

```

```

9053     GshMenuEdit.style.color = 'rgba(16,160,16,1)';
9054     GshMenuEdit.style.backgroundColor = 'rgba(255,255,255,1)';
9055 }
9056 }
9057 function html_edit(){
9058     ToggleEditMode();
9059 }
9060
9061 // Live HTML (DOM) Snapshot onto browser's localStorage
9062 // 2020-0923 SatoxITS
9063 var htRoot = gsh // -- Element-ID, should be selectable
9064 const snappedHTML = 'SnappedHTML'; // Item-ID of the HTML data in localStogate
9065 // -- should be a [map] of URL
9066 // -- should be with CSSOM as inline script
9067 const htVersionTag = 'VersionTag'; // VesionTag Eelment-ID in the HTML (in DOM)
9068 function showVersion(note,w,v,u,t){
9069     w.alert(note+' : ' + v + '\n'
9070         + '-- URL: ' + u + '\n'
9071         + '-- Time: ' + DateLong0(t*1000)
9072     );
9073 }
9074 function html_save(){
9075     u = document.URL;
9076     t = new Date().getTime() / 1000;
9077     v = '<'+span id="+htVersionTag+" data-url="+u+" data-time="+t+">";
9078     v += '<'+/span>\n';
9079     h += v + htRoot.outerHTML;
9080     localStorage.setItem(snappedHTML,h);
9081     showVersion("Saved",window,v,u,t);
9082 }
9083 function html_load(){
9084     h = localStorage.getItem(snappedHTML);
9085     if( h == null ){
9086         alert('No snapshot taken yet');
9087         return;
9088     }
9089     w = window.open('','');
9090     d = w.document;
9091     d.write(h);
9092     w.focus();
9093     html_ver1("Loaded",w,d);
9094 }
9095 function html_ver1(note,w,d){
9096     if( (v = d.getElementById(htVersionTag)) != null ){
9097         h = v.outerHTML;
9098         u = v.getAttribute('data-url');
9099         t = v.getAttribute('data-time');
9100     }else{
9101         h = 'No version info. in the page';
9102         u = '';
9103         t = 0;
9104     }
9105     showVersion(note,w,v,u,t);
9106 }
9107 function html_ver0(){
9108     html_ver1("Version",window,document);
9109 }
9110 </script>
9111 <!-- LiveHTML -->
9112 </span>
9113 </details>
9114 */
9115 /*
9116 <details><summary>Event sharing</summary>
9117 <span id="EventSharingCodeSpan">
9118
9119
9120 <!-- ----- Event sharing // 2020-0925 SatoxITS { -->
9121
9122 <div id="iftestTemplate" class="iftest" hidden="">
9123 <style>.iftestbody{ color:#f22;font-family:Georgia;font-size:10pt;} </style>
9124 <span id="frameBody" class="iftestbody" onclick="frameClick()"><script>
9125     function docadd(txt){
9126         document.body.append(txt);
9127         window.scrollTo(0,100000);
9128     }
9129     function frameClick(){
9130         xy = '(x='+event.x + ' y='+event.y+')';
9131         //docadd('Got Click on #' +event.target.id+ '+xy+ '\n');
9132         docadd('Got Click on #' +Fid.value+ ', '+xy+ '\n');
9133         window.scrollTo(0,100000);
9134         window.parent.postMessage('OnClick: '+xy,'*');
9135     }
9136     function frameMousemove(){
9137         if( false ){
9138             document.body.append('Mousemove on #' +event.target.id+ ' '
9139                 + 'x='+event.x + ' y='+event.y + '\n');
9140             peerWin = window.frames.iframe1;
9141             document.body.append('Send to peer #' +peerWin+ ' ' + '\n');
9142             window.scrollTo(0,100000);
9143             peerWin.postMessage('Hi!', '*');
9144         }
9145     }
9146     function frameKeydown(){
9147         msg = 'Got Keydown: #' +Fid.value+ ', (' +event.code+')';
9148         docadd(msg+'\n');
9149         window.parent.postMessage(msg, '*');
9150     }
9151     function frameOnMessage(){
9152         docadd('Message ' + event.data + '\n');
9153         window.scrollTo(0,100000);
9154     }
9155     if( document.getElementById('Fid') ){
9156         frameBody.id = Fid.value;
9157         h = '';
9158         h += '<'+style>*<';
9159         h += 'font-size:10pt;white-space:pre-wrap;';
9160         h += 'font-family:Courier New;';
9161         h += '<'+/style>';
9162         h += 'I am '+Fid.value+'\n';
9163         document.write(h);
9164         window.addEventListener('click',frameClick);
9165         window.addEventListener('keydown',frameKeydown);
9166         window.addEventListener('message',frameOnMessage);
9167         window.addEventListener('mousemove',frameMousemove);
9168         window.parent.postMessage('Hi parent, I am '+Fid.value,'*');
9169     }
9170 </script></span></div>
9171
9172 <style>.iframeTest{margin:3px;resize:both;width:370px;height:120px;}</style>
9173 <h2>Inter-window communicaiton</h2>
9174 <note>
9175 frame0 >>> frame1 and frame2<br>
9176 frame1 >>> frame0 and frame2<br>

```

```

9177 frame2 >>> frame0 and frame1<br>
9178 </note>
9179 <div id="iframe-test">
9180 <pre id="iframeHost" style="border:1px;font-family:Courier New;font-size:10pt;"></pre>
9181 <iframe id="iframe0" title="iframe0" class="iframeTest" style=""></iframe>
9182 <iframe id="iframe1" title="iframe1" class="iframeTest"></iframe>
9183 <iframe id="iframe2" title="iframe2" class="iframeTest"></iframe>
9184 </div>
9185
9186 <script id="if0-test-script">
9187 setupFrames0();
9188 setupFrames12();
9189
9190 function setFrameSrcdoc(dst,src){
9191   if( true ){
9192     dst.contentWindow.document.write(src);
9193     // this makes browser waite close, and crash if accumulated !?
9194     // so it should be closed after write
9195     dst.contentWindow.document.close();
9196   }else{
9197     // to be erased before source dump
9198     // but should be set for live snapshot
9199     dst.srcdoc = src;
9200   }
9201 }
9202 function setupFrames0(){
9203   ibody = iframe0.contentWindow.document.body;
9204   iframe0.style.width = "755px"
9205   //iframeHost.innerHTML = "Message exchange at iframes' host:\n";
9206   window.addEventListener('message',messageFromChild);
9207
9208   if0 = '';
9209   if0 += '<'+pre style="font-family:Courier New;">';
9210   if0 += '<input id="Fid" value="iframe0">';
9211   if0 += iftestTemplate.innerHTML;
9212   setFrameSrcdoc(iframe0,if0);
9213
9214   function clickOnChild(){
9215     console.log('clickOn #' +this.id);
9216   }
9217   function moveOnChild(){
9218     console.log('moveOn #' +this.id);
9219   }
9220   iframe0.contentWindow.document.body.style.setProperty('white-space','pre');
9221   iframe0.contentWindow.document.body.style.setProperty('font-size','9pt');
9222 }
9223 function setupFrames12(){
9224   if1 = '<input id="Fid" value="iframe1">';
9225   if1 += iftestTemplate.innerHTML;
9226   setFrameSrcdoc(iframe1,if1);
9227   //iframe1.name = 'iframe1'; // this seems break contentWindow
9228
9229   if2 = '<input id="Fid" value="iframe2">';
9230   if2 += iftestTemplate.innerHTML;
9231   setFrameSrcdoc(iframe2,if2);
9232
9233   iframe1.addEventListener('message',messageFromChild);
9234   //iframe1.addEventListener('mouseover',moveOnChild);
9235   iframe2.addEventListener('message',messageFromChild);
9236   //iframe2.addEventListener('mouseover',moveOnChild);
9237   iframe1.contentWindow.postMessage(['parent0] Hi iframe1 -- from parent.','*');
9238   //iframe1.contentWindow.postMessage('Your peer is '+iframe2.contentWindow,'*');
9239   iframe2.contentWindow.postMessage(['parent0] Hi iframe2 -- from parent.','*');
9240   //iframe2.contentWindow.postMessage('Your peer is '+iframe1.contentWindow,'*');
9241 }
9242 function messageFromChild(){
9243   from = null;
9244   forw = null;
9245   if( event.source == iframe0.contentWindow ){
9246     from = 'iframe0'
9247     forw = 'iframe12';
9248   }else
9249   if( event.source == iframe1.contentWindow ){
9250     from = 'iframe1'
9251     forw = 'iframe2';
9252   }else
9253   if( event.source == iframe2.contentWindow ){
9254     from = 'iframe2'
9255     forw = 'iframe1';
9256   }else
9257   {
9258     iframeHost.innerHTML += 'Message [unknown] '
9259     + ' orig=' + event.origin
9260     + ' data=' + event.data
9261     //+ ' from=' + event.source
9262     ;
9263   }
9264   msglog1 = from + event.data + ' -- '
9265   + ' from=' + event.source
9266   + ' orig=' + event.origin
9267   + ' name=' + event.source.name
9268   //+ ' port=' + event.ports
9269   //+ ' evid=' + event.lastEventId
9270   + '\n'
9271   ;
9272   if( true ){
9273     if( forw == 'iframe1' || forw == 'iframe12' ){
9274       iframe1.contentWindow.postMessage(from+event.data);
9275     }
9276     if( forw == 'iframe2' || forw == 'iframe12' ){
9277       iframe2.contentWindow.postMessage(from+event.data);
9278     }
9279   }
9280   txtadd0(msglog1);
9281
9282   function txtadd0(txt){
9283     iframe0.contentWindow.document.body.append(txt);
9284     iframe0.contentWindow.scrollTo(0,100000);
9285   }
9286 }
9287 function es_ShowSelf(){
9288   iframe1.setAttribute('src',document.URL);
9289   iframe2.setAttribute('src',document.URL);
9290 }
9291 </script>
9292
9293 <input class="HtmlCodeviewButton" type="button" value="ShowSelf" onclick="es_ShowSelf()">
9294 <input class="HtmlCodeviewButton" type="button" value="ShowCode" onclick="es_showHtmlCode()">
9295 <span id="EventSharingCodeview"></span>
9296 <script id="EventSharingScript">
9297 function es_showHtmlCode(){
9298   showHtmlCode(EventSharingCodeview,EventSharingCodeSpan);
9299 }
9300 DestroyEventSharingCodeview = function(){

```

```

9301 //EventSharingCodeview.parentNode.removeChild(EventSharingCodeview);
9302 EventSharingCodeview.innerHTML = "";
9303 iframe0.style = "";
9304 //iframe0.srcdoc = "erased";
9305 //iframe1.srcdoc = "erased";
9306 //iframe2.srcdoc = "erased";
9307 }
9308 </script>
9309 <!-- EventSharing } -->
9310 </span>
9311 </details>
9312 */
9313
9314 /*
9315 <!-- ----- "GShell Inside" Notifitaction { -->
9316 <script id="script-gshell-inside">
9317 var notices = 0;
9318 function noticeGShellInside(){
9319     ver = '';
9320     if( ver = document.getElementById('GshVersion') ){
9321         ver = ver.innerHTML;
9322     }
9323     console.log('GJShell Inside (~)/'+ver);
9324     notices += 1;
9325     if( 2 <= notices ){
9326         document.removeEventListener('mousemove',noticeGShellInside);
9327     }
9328 }
9329 document.addEventListener('mousemove',noticeGShellInside);
9330 noticeGShellInside();
9331
9332 const FooterName = 'GshFooter'
9333 function DestroyFooter(){
9334     if( (footer = document.getElementById(FooterName)) != null ){
9335         //footer.parentNode.removeChild(footer);
9336         empty = document.createElement('div');
9337         empty.id = 'GshFooter0';
9338         footer.parentNode.replaceChild(empty,footer);
9339     }
9340 }
9341
9342 footer = document.createElement('div');
9343 footer.id = FooterName;
9344 footer.style.backgroundImage = "url("+ITSmoreQR+)";
9345 //GshFooter0.parentNode.appendChild(footer);
9346 GshFooter0.parentNode.replaceChild(footer,GshFooter0);
9347 </script>
9348 <!-- } -->
9349
9350 <!--
9351     border:20px inset #888;
9352 -->
9353
9354 //<span id="WirtualDesktopCodeSpan">
9355 /*
9356 <details open=""><summary>Wirtual Desktop</summary>
9357 <!-- ----- Web Wirtual Desktop // 2020-0927 SatoxITS { -->
9358 <style>
9359 .WirtualDesktop {
9360     position:relative;
9361     resize:both;
9362     display:block;
9363     width:800px;
9364     height:480;
9365     border:30px solid #222;
9366     border-radius:20px;
9367     background-color:rgba(32,32,160,0.8);
9368     background-image:url("WD-WallPaper.jpeg");
9369     background-size:100% 100%;}
9370     color:#22a;font-family:Georgia;font-size:10pt;
9371     overflow:scroll;
9372 }
9373 // specifying overflow seems fix the position of the element
9374 .WirtualBrowserSpan {
9375     position:relative;
9376     xxxposition:absolute;
9377     left:100px;
9378     top:100px;
9379     z-index:10;
9380     display:block;
9381     resize:both;
9382     width:500px;
9383     height:270px;
9384     background-color:rgba(255,200,160,1.0);
9385     overflow:scroll;
9386 }
9387 .xWirtualBrowserLocationBar:focus {
9388     color:#f00;
9389     background-color:rgba(255,128,128,0.2);
9390 }
9391 .xWirtualBrowserLocationBar:active {
9392     color:#f00;
9393     background-color:rgba(128,255,128,0.2);
9394 }
9395 a.WirtualBrowserLocation {
9396     color:#ccc !important;
9397     text-decoration:none;
9398 }
9399 a.WirtualBrowserLocation:hover {
9400     color:#fff !important;
9401     text-decoration:underline;
9402 }
9403 .WirtualBrowserLocationBar {
9404     position:absolute;
9405     z-index:100000;
9406     xxxdisplay:inline;
9407     display:block;
9408     width:400px;
9409     height:20px;
9410     padding-left:2px;
9411     line-height:1.1;
9412     vertical-align:middle;
9413     font-size:14px;
9414     color:#fff;
9415     background-color:rgba(128,128,128,0.1);
9416     font-family:Georgia;
9417 }
9418 .WirtualBrowserCommandBar {
9419     position:absolute;
9420     z-index:200000;
9421     xxxdisplay:inline;
9422     display:block;
9423     width:60px;
9424     height:20px;

```

```

9425     line-height:1.1;
9426     vertical-align:middle;
9427     font-size:14px;
9428     color:#f44;
9429     background-color:rgba(128,128,128,0.1);
9430     font-family:Georgia;
9431     text-align:left;
9432     left:404px;
9433 }
9434 .VirtualBrowserFrame {
9435     position:relative;
9436     display:inline;
9437     z-index:10;
9438     resize:both;
9439     display:block;
9440     width:480px;
9441     height:240px;
9442     border-radius:6px;
9443     background-color:rgba(255,255,255,0.9);
9444     border-top:20px solid;
9445     border-right:4px solid;
9446     border-bottom:10px solid;
9447 }
9448 .WinFavicon {
9449     width:16px;
9450     height:16px;
9451     margin:1px;
9452     margin-right:3px;
9453     vertical-align:middle;
9454     background-color:rgba(255,255,255,1.0);
9455 }
9456 .VirtualDesktopMenuBar {
9457     color:#fff;
9458     font-size:7pt;
9459     text-align:right;
9460     padding-right:4px;
9461     background-color:rgba(255,255,255,0.2);
9462 }
9463 .VirtualDesktopCalender {
9464     color:#fff;
9465     font-size:22pt;
9466     text-align:right;
9467     padding-right:4px;
9468     xbackground-color:rgba(255,255,255,0.2);
9469 }
9470 </style>
9471 <h2>Virtual Desktop</h2>
9472 <div id="VirtualDesktop_1" class="VirtualDesktop" xdraggable="true" contenteditable="true">
9473   <div id="VirtualDesktop_1_MenuBar" class="VirtualDesktopMenuBar" spellcheck="false">
9474     <i>Virtual Desktop 0.0.2</i>
9475     <span id="VirtualDesktop_1_Clock"></span>
9476   </div>
9477   <div id="VirtualDesktop_1_Calender" class="VirtualDesktopCalender" >00:00</div>
9478
9479   <div id="VirtualBrowser_1" class="VirtualBrowserSpan" spellcheck="false" draggable="true">
9480     <div id="VirtualBrowser_1_Location" class="VirtualBrowserLocationBar"></div>
9481     <span id="VirtualBrowser_1_Command" class="VirtualBrowserCommandBar">Reload</span>
9482     <iframe id="VirtualBrowser_1_Frame" class="VirtualBrowserFrame" style=""></iframe>
9483   </div>
9484
9485   <div id="VirtualBrowser_2" class="VirtualBrowserSpan" spellcheck="false" draggable="true">
9486     <div id="VirtualBrowser_2_Location" class="VirtualBrowserLocationBar"></div>
9487     <span id="VirtualBrowser_2_Command" class="VirtualBrowserCommandBar">Reload</span>
9488     <iframe id="VirtualBrowser_2_Frame" class="VirtualBrowserFrame" style=""></iframe>
9489   </div>
9490
9491   <div id="VirtualBrowser_3" class="VirtualBrowserSpan" spellcheck="false" draggable="true">
9492     <div id="VirtualBrowser_3_Location" class="VirtualBrowserLocationBar"></div>
9493     <span id="VirtualBrowser_3_Command" class="VirtualBrowserCommandBar">Reload</span>
9494     <iframe id="VirtualBrowser_3_Frame" class="VirtualBrowserFrame" style=""></iframe>
9495   </div>
9496
9497 </div>
9498 <input class="HtmlCodeviewButton" type="button" value="ShowCode" onclick="vd_showHtmlCode()">
9499 <span id="VirtualDesktopCodeview"></span>
9500 <script id="WirutalDesktopScript">
9501 function vd_showHtmlCode(){
9502     showHtmlCode(VirtualDesktopCodeview,VirtualDesktopCodeSpan);
9503 }
9504 DestroyEventSharingCodeview = function(){
9505     VirtualDesktopCodeview.innerHTML = "";
9506 }
9507
9508 var topMostWin = 10000;
9509 function onEnterWin(e){
9510     t = e.target;
9511     oindex = t.style.zIndex;
9512     //if( oindex == '' ) oindex = 0;
9513     //t.saved_zIndex = oindex;
9514     //t.style.zIndex = 10000;
9515     topMostWin += 1;
9516     t.style.zIndex = topMostWin;
9517     nindex = t.style.zIndex;
9518     console.log('Enter '+t.id+'('+oindex+'->'+nindex+')');
9519     e.stopPropagation();
9520     e.preventDefault();
9521 }
9522 function onClickWin(e){ // can detect click on the thick border?
9523     t = e.target;
9524     oindex = t.style.zIndex;
9525     topMostWin += 1;
9526     t.style.zIndex = topMostWin;
9527     nindex = t.style.zIndex;
9528     console.log('Click '+t.id+'('+oindex+'->'+nindex+')');
9529     //e.stopPropagation();
9530     //e.preventDefault();
9531 }
9532 function onLeaveWin(e){
9533     t = e.target;
9534     //oindex = t.style.zIndex;
9535     //nindex = t.saved_zIndex;
9536     //t.style.zIndex = nindex;
9537     //console.log('Leave '+e.target.id+'('+oindex+'->'+nindex+')');
9538     e.stopPropagation();
9539     e.preventDefault();
9540 }
9541
9542 var WinDragstartX; // event
9543 var WinDragstartY;
9544 var WinDragstartTX; // target
9545 var WinDragstartTY;
9546
9547 function onWinDragstart(e){
9548     t = e.target;

```

```

9549 //WinDragstartTX = t.getBoundingClientRect().left.toFixed(0)
9550 //WinDragstartTY = t.getBoundingClientRect().top.toFixed(0)
9551 WinDragstartTX = x0 = t.style.x;
9552 WinDragstartTY = y0 = t.style.y;
9553 console.log('DragSTA #' + t.id + '(' + e.x + ', ' + e.y + ') --- init(' + x0 + ', ' + y0 + ')');
9554 WinDragstartX = e.x;
9555 WinDragstartY = e.y;
9556 e.stopPropagation();
9557 //e.preventDefault();
9558 }
9559 function onWinDrag(e){
9560 t = e.target;
9561 //console.log('DragGIN #' + t.id + '(' + e.x + ', ' + e.y + ')(' + xx + ', ' + yy + ')');
9562 dx = e.x - WinDragstartX;
9563 dy = e.y - WinDragstartY;
9564 t.style.left = WinDragstartTX + dx;
9565 t.style.top = WinDragstartTY + dy;
9566 console.log('DragGIN #' + t.id + '(' + e.x + ', ' + e.y + ') set (' + dx + ', ' + dy + ')');
9567 e.stopPropagation();
9568 e.preventDefault();
9569 }
9570 function onWinDragend(e){
9571 t = e.target;
9572 //console.log('DragEND #' + t.id + '(' + e.x + ', ' + e.y + ')');
9573 e.stopPropagation();
9574 e.preventDefault();
9575 }
9576 function onFaviconChange(e){
9577 console.log('--Favicon #' + e.target.id + ' href=' + e.details.href);
9578 }
9579 var savedSuppressGJShell = false;
9580 function stopGJShell(e){
9581 console.log('enter Gsh STOP\n');
9582 savedSuppressGJShell = SuppressGJShell;
9583 SuppressGJShell = true;
9584 e.stopPropagation();
9585 e.preventDefault();
9586 }
9587 function contGJShell(e){
9588 console.log('leave Gsh STOP\n');
9589 SuppressGJShell = savedSuppressGJShell;
9590 e.stopPropagation();
9591 e.preventDefault();
9592 }
9593 function settleWin(s,l,cmd,f,u,x,y,c){
9594 function VirtualBrowserCommand(e,s,l,cmd,f){
9595 command = cmd.innerHTML
9596 if( command == "Reload" ){
9597 href_id = e.target.href_id;
9598 d = document.getElementById(href_id);
9599 url = d.innerHTML;
9600 //alert('Reload '+f.id+' with '+href_id+'\nhref='+d+'\nurl='+url);
9601 f.src = url;
9602 }else{
9603 alert('unknown command'+command+' '+e.target.id+' '+l.id+' '+f.id);
9604 }
9605 }
9606 function onKeyDown(e){
9607 if( e.code == 'Enter' ){
9608 e.stopPropagation();
9609 e.preventDefault();
9610 }
9611 }
9612 function onKeyUp(e){
9613 if( e.code == 'Enter' ){
9614 e.stopPropagation();
9615 e.preventDefault();
9616 // should reload immediately ?
9617 }
9618 }
9619 console.log('start settle VirtualBrowser url='+u +'\n'
9620 + 'id=' + s.id + '\n'
9621 + 'width=' + s.style.width + '\n'
9622 + 'height=' + s.style.height + '\n'
9623 );
9624 // very important for WordPress ??
9625 s.style.width = f.style.width = 501; // for WordPress ...??
9626 s.style.height = f.style.height = 271; // for WordPress ...??
9627 console.log('midway settle VirtualBrowser url='+u +'\n'
9628 + 'id=' + s.id + '\n'
9629 + 'width=' + s.style.width + '\n'
9630 + 'height=' + s.style.height + '\n'
9631 );
9632 s.width = 502; // for WordPress ...??
9633 s.height = 272; // for WordPress ...??
9634 console.log('midway-2 settle VirtualBrowser url='+u +'\n'
9635 + 'id=' + s.id + '\n'
9636 + 'span-width=' + s.width + '\n'
9637 + 'span-height=' + s.height + '\n'
9638 );
9639 );
9640 //console.log('--x1-- u='+u+' width s='+s.style.width+',f='+f.style.width);
9641 //console.log('--x2-- u='+u+' width s='+s.style.width+',f='+f.style.width);
9642 s.setAttribute('draggable','true')
9643 s.addEventListener('dragstart', e => { onWinDragstart(e); });
9644 s.addEventListener('drag', e => { onWinDrag(e); });
9645 s.addEventListener('dragend', e => { onWinDragend(e); });
9646
9647 if( false ){
9648 s.style.position = "absolute";
9649 s.style.x = x+'px';
9650 s.style.left = x+'px';
9651 s.style.y = y+'px';
9652 s.style.top = y+'px';
9653 }else{
9654 s.style.setProperty('position','absolute','important');
9655 s.style.setProperty('x',x+'px','important');
9656 s.style.setProperty('left',x+'px','important');
9657 s.style.setProperty('y',y+'px','important');
9658 s.style.setProperty('top',y+'px','important');
9659 }
9660 }
9661
9662 favicon = './favicon.ico';
9663 uv1 = u.split('/');
9664 if( 2 <= uv1.length ){
9665 uv2 = uv1[1].split('/');
9666 if( 2 <= uv2.length ){
9667 if( uv1[0] == 'file' ){
9668 //favicon = 'file://' + uv2.slice(0,uv2.length-1).join('/');
9669 // + '/favicon.ico';
9670 favicon = './favicon.ico';
9671 }else{
9672 favicon = uv1[0] + '://' + uv2[0] + '/favicon.ico';

```

```

9673     }
9674   }
9675 }
9676 //console.log("----- favicon-url="+favicon);
9677 href_id = l.id + ' href';
9678 l.innerHTML += '+u+'</a>';
9680 //l.addEventListener('click', e => { onClickWin(e); });
9681 l.addEventListener('mouseenter',e => { stopGShell(e); });
9682 l.addEventListener('mouseleave',e => { contGShell(e); });
9683 l.addEventListener('keydown', e => { onKeyDown(e); });
9684 l.addEventListener('keyup', e => { onKeyUp(e); });
9685
9686 cmd.href_id = href_id;
9687 cmd.addEventListener('click', e => { WirtualBrowserCommand(e,s,l,cmd,f); });
9688
9689 f.style.borderColor = c;
9690 f.src = u;
9691 f.addEventListener('mouseenter',e => { onEnterWin(e); });
9692 f.addEventListener('mouseleave',e => { onLeaveWin(e); });
9693
9694 //s.addEventListener('click', e => { onClickWin(e); });
9695 //f.addEventListener('click', e => { console.log('click wbl'); });
9696 f.addEventListener('mozbrowsericonchange',onFaviconChange);
9697
9698 console.log('done settle WirtualBrowser url='+u +'\n'
9699   + 'id=' + s.id + '\n'
9700   + 'width=' + s.style.width + '\n'
9701   + 'height=' + s.style.height + '\n'
9702 );
9703 }
9704
9705 WirtualDesktop_1.style.width = "800px";
9706 WirtualDesktop_1.style.height = "500px";
9707
9708 settleWin(
9709   WirtualBrowser_1,
9710   WirtualBrowser_1_Location,
9711   WirtualBrowser_1_Command,
9712   WirtualBrowser_1_Frame,
9713   document.URL,
9714   50,20,'#262');
9715 settleWin(
9716   WirtualBrowser_2,
9717   WirtualBrowser_2_Location,
9718   WirtualBrowser_2_Command,
9719   WirtualBrowser_2_Frame,
9720   'https://its-more.jp/ja_jp/',
9721   150,100,'#448');
9722 settleWin(
9723   WirtualBrowser_3,
9724   WirtualBrowser_3_Location,
9725   WirtualBrowser_3_Command,
9726   WirtualBrowser_3_Frame,
9727   './gshell/gsh.go.html',
9728   '//http://gshell.org/gshell/gsh.go.html',
9729   250,180,'#444');
9730
9731 function WD_ClockUpdate(e){
9732   WirtualDesktop_1_Clock.innerHTML = DateShort();
9733   WirtualDesktop_1_Calender.innerHTML = DateHourMin();
9734 }
9735 window.setInterval(WD_ClockUpdate,500);
9736
9737 Destroy_WirtualDesktop = function(){
9738   WirtualBrowser_1.removeAttribute('style');
9739   WirtualBrowser_1_Location.innerHTML = '';
9740   WirtualBrowser_1_Frame.removeAttribute('src');
9741   WirtualBrowser_1_Frame.removeAttribute('style');
9742   WirtualBrowser_1_Frame.style="";
9743
9744   WirtualBrowser_2.removeAttribute('style');
9745   WirtualBrowser_2_Location.innerHTML = '';
9746   WirtualBrowser_2_Frame.removeAttribute('src');
9747   WirtualBrowser_2_Frame.style="";
9748
9749   WirtualBrowser_3.removeAttribute('style');
9750   WirtualBrowser_3_Location.innerHTML = '';
9751   WirtualBrowser_3_Frame.removeAttribute('src');
9752   WirtualBrowser_3_Frame.style="";
9753
9754   GJFactory_1.style = "";
9755   iframe0.style = "";
9756   WirtualDesktop_1.style = "";
9757 }
9758
9759 </script>
9760 <!-- WirtualDesktop } -->
9761 </details>
9762 */
9763 </span>
9764
9765 <!-- ----- Work { ----- -->
9766 <span id="Template_WorkCodeSpan">
9767 /*
9768 <details open=""><summary>Work Template</summary>
9769 <!-- ----- Template of Work// 2020-0928 SatoxITS { -->
9770 <h2>Template of Work</h2>
9771 <input id="Template_WorkCodeViewOpen" class="HtmlCodeViewButton" type="button" value="ShowCode">
9772 <span id="Template_WorkCodeView"></span>
9773 <script id="Template_WorkScript">
9774 function Tmplate_openWorkCodeView(){
9775   function Template_showWorkCode(){
9776     showHtmlCode(Template_WorkCodeView,Template_WorkCodeSpan);
9777   }
9778   Template_WorkCodeViewOpen.addEventListener('click',Template_showWorkCode);
9779 }
9780 Template_openWorkCodeView(); // should be invoked by an event
9781 </script>
9782 </details>
9783 <!-- Template_WorkCodeSpan } -->
9784 */ </span>
9785 <!-- ----- Work } ----- -->
9786
9787 /*
9788 */ </br></span></html>
9789
9790

```