

```

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--0.5.6--2020-09-30--SatoxITS</span>
7  <title>GShell-0.5.6 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 0.5.6 // 2020-09-30 // 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();">Buildin 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 pollx(pollFdv *fdv, int nfds, int timeout){
162 //     return poll(fdv->fdv,nfds,timeout);
163 // }
164 import "C"
165
166 // // 2020-0906 added,
167 func CFPollIn1(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 = "0.5.6"
196     DATE = "2020-09-30"
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(%)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 arg1 = 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,arg1[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                 arg1[ax] = ch
1435                 ax += 1
1436                 continue
1437             }
1438         }
1439         if str[si] == ' ' {
1440             argv = append(argv,string(arg1[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         arg1[ax] = ch
1450         ax += 1
1451     }
1452     if sb < si {
1453         argv = append(argv,string(arg1[0:ax]))
1454         if debug {
1455             fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
1456                 -1+len(argv),sb,si,string(arg1[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]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]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 absi(size int64)(string){
1669     fsize := float64(size)
1670     if 1024*1024*1024 < size {
1671         return fmt.Sprintf("%8.2fGiB",fsize/(1024*1024*1024))
1672     }else
1673     if 1024*1024 < size {
1674         return fmt.Sprintf("%8.3fMiB",fsize/(1024*1024))
1675     }else{
1676         return fmt.Sprintf("%8.3fKiB",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("%6.3fGB/s",MBs/1000)
1683     }
1684     if 1 <= MBs {
1685         return fmt.Sprintf("%6.3fMB/s",MBs)
1686     }else{
1687         return fmt.Sprintf("%6.3fKB/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("%6.3fGBps",MBs/1000)
1694     }
1695     if 1 <= MBs {
1696         return fmt.Sprintf("%6.3fMBps",MBs)
1697     }else{
1698         return fmt.Sprintf("%6.3fKBps",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,absize(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,absize(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, absize(total), 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     }
1949     }
1950 }
1951 func (gsh*GshContext)RexecClient(argv []string)(int,string){
1952     debug := true
1953     Start := time.Now()
1954     if len(argv) == 1 {
1955         return -1,"EmptyARG"
1956     }
1957     argv = argv[1:]
1958     if argv[0] == "-serv" {
1959         gsh.RexecServer(argv[1:])
1960         return 0,"Server"
1961     }
1962     remote := "0.0.0.0:9999"
1963     if argv[0][0] == '@' {
1964         remote = argv[0][1:]
1965         argv = argv[1:]
1966     }
1967     if argv[0] == "-s" {
1968         debug = false
1969         argv = argv[1:]
1970     }
1971     dport, err := net.ResolveTCPAddr("tcp",remote);
1972     if err != nil {
1973         fmt.Printf(Elapsed(Start)+"Address error: %s (%s)\n",remote,err)
1974         return -1,"AddressError"
1975     }
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 (/%)\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.Trelpath(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.Trelpath(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     return gline, false, false
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", abftime(tu))
2700     ret += fmt.Sprintf(", %v/usr", abftime(uu))
2701     ret += fmt.Sprintf(", %v/sys", abftime(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(" Ssr=%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
2803     fdix := 0
2804     mode := os.O_RDONLY
2805
2806     switch {
2807     case cmd == "-i" || cmd == "<":
2808         fdix = 0
2809         mode = os.O_RDONLY
2810     case cmd == "-o" || cmd == ">":
2811         fdix = 1
2812         mode = os.O_RDWR | os.O_CREATE
2813     case cmd == "-a" || cmd == ">>":
2814         fdix = 1
2815         mode = os.O_RDWR | os.O_CREATE | os.O_APPEND
2816     }
2817     if fname[0] == '#' {
2818         fd, err := strconv.Atoi(fname[1:])
2819         if err != nil {
2820             fmt.Printf("--E-- (%v)\n",err)
2821             return false
2822         }
2823         file = os.NewFile(uintptr(fd),"MaybePipe")
2824     }else{
2825         xfile, err := os.OpenFile(argv[1], mode, 0600)
2826         if err != nil {
2827             fmt.Printf("--E-- (%s)\n",err)
2828             return false
2829         }
2830         file = xfile
2831     }
2832     gshPA := gshCtx.gshPA
2833     savfd := gshPA.Files[fdix]
2834     gshPA.Files[fdix] = file.Fd()
2835     fmt.Printf("--I-- Opened [%d] %s\n",file.Fd(),argv[1])
2836     gshCtx.gshellv(argv[2:])
2837     gshPA.Files[fdix] = savfd
2838     return false
2839 }
2840
2841 //fmt.Fprintf(res, "GShell Status: %q", html.EscapeString(req.URL.Path))
2842 func httpHandler(res http.ResponseWriter, req *http.Request){
2843     path := req.URL.Path
2844     fmt.Printf("--I-- Got HTTP Request(%s)\n",path)
2845     {
2846         gshCtxBuf, _ := setupGshContext()
2847         gshCtx := &gshCtxBuf
2848         fmt.Printf("--I-- %s\n",path[1:])
2849         gshCtx.tgshellv(path[1:])
2850     }
2851 }

```

```

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 }
3156 xargv := []string{}
3157 for ai := 0; ai < len(argv); ai++ {
3158     xargv = append(xargv, strsubst(gshCtx, argv[ai], false))
3159 }
3160 argv = xargv
3161 if false {
3162     for ai := 0; ai < len(argv); ai++ {
3163         fmt.Printf("[%d] %s [%d]T\n",
3164             ai, argv[ai], len(argv[ai]), argv[ai])
3165     }
3166 }
3167 cmd := argv[0]
3168 if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr, "--I-- gshellv(%d)%v\n", len(argv), argv) }
3169 switch { // https://tour.golang.org/flowcontrol/11
3170 case cmd == "":
3171     gshCtx.xPwd([]string{}); // empty command
3172 case cmd == "-x":
3173     gshCtx.CmdTrace = ! gshCtx.CmdTrace
3174 case cmd == "-xt":
3175     gshCtx.CmdTime = ! gshCtx.CmdTime
3176 case cmd == "-ot":
3177     gshCtx.sconnect(true, argv)
3178 case cmd == "-ou":
3179     gshCtx.sconnect(false, argv)
3180 case cmd == "-it":
3181     gshCtx.saccept(true, argv)
3182 case cmd == "-iu":
3183     gshCtx.saccept(false, argv)
3184 case cmd == "-i" || cmd == "<" || cmd == "-o" || cmd == ">" || cmd == "-a" || cmd == ">>" || cmd == "-s" || cmd == "><":
3185     gshCtx.redirect(argv)
3186 case cmd == "|":
3187     gshCtx.fromPipe(argv)
3188 case cmd == "args":
3189     gshCtx.Args(argv)
3190 case cmd == "bg" || cmd == "-bg":
3191     rfin := gshCtx.inBackground(argv[1:])
3192     return rfin
3193 case cmd == "bn":
3194     gshCtx.Basename(argv)
3195 case cmd == "call":
3196     _, _ = gshCtx.excommand(false, argv[1:])
3197 case cmd == "cd" || cmd == "chdir":
3198     gshCtx.xChdir(argv);
3199 case cmd == "-cksum":
3200     gshCtx.xFind(argv)
3201 case cmd == "-sum":
3202     gshCtx.xFind(argv)
3203 case cmd == "-sumtest":
3204     str := ""
3205     if 1 < len(argv) { str = argv[1] }
3206     crc := strCRC32(str, uint64(len(str)))
3207     fmt.Fprintf(stderr, "%v %v\n", crc, len(str))
3208 case cmd == "close":
3209     gshCtx.xClose(argv)
3210 case cmd == "gcp":
3211     gshCtx.FileCopy(argv)
3212 case cmd == "dec" || cmd == "decode":
3213     gshCtx.Dec(argv)
3214 case cmd == "#define":
3215     gshCtx.xDic(argv)
3216 case cmd == "dic" || cmd == "d":
3217     xDic(argv)
3218 case cmd == "dump":
3219     gshCtx.Dump(argv)
3220 case cmd == "echo" || cmd == "e":
3221     echo(argv, true)
3222 case cmd == "enc" || cmd == "encode":
3223     gshCtx.Enc(argv)
3224 case cmd == "env":
3225     env(argv)
3226 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 == "rexecc" || cmd == "rexd":
3278     gshCtx.RexeccServer(argv)
3279 case cmd == "rexece" || cmd == "rex":
3280     gshCtx.RexeccClient(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.Rexec(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],argv[1],argv[2])
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 // dictionaly 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     xfprintf(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, &stout) // spec. mismatch
4158     err := syscall.Select(nfd, &rdv, nil, nil, &stout)
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         }
4395     }
4396     return ch1
4397 }
4398 func (iin*IInput)clearline(){
4399     var i int
4400     fprintf(stderr,"\r");
4401     // should be ANSI ESC sequence
4402     for i = 0; i < TtyMaxCol; i++ { // to the max. position in this input action
4403         fputc(' ',os.Stderr);
4404     }
4405     fprintf(stderr,"\r");
4406 }
4407 func (iin*IInput)Redraw(){
4408     redraw(iin,iin.lno,iin.line,iin.right)
4409 }
4410 func redraw(iin *IInput,lno int,line string,right string){
4411     inMeta := false
4412     showMode := ""
4413     showMeta := "" // visible Meta_mode on the cursor position
4414     showLino := fmt.Sprintf("%d",lno)
4415     insertMark := "" // in visible insert mode
4416 }
4417 if MODE_VicMode {
4418     }else
4419     if 0 < len(iin.right) {
4420         insertMark = " "
4421     }
4422     if( 0 < len(iin.waitingMeta) ){
4423         inMeta = true
4424         if iin.waitingMeta[0] != 033 {
4425             showMeta = iin.waitingMeta
4426         }
4427     }
4428     if( romkanmode ){
4429         //romkanmark = " *";
4430     }else{
4431         //romkanmark = "";
4432     }
4433     if MODE_ShowMode {
4434         romkan := "- "
4435         inmeta := "- "
4436         inveri := ""
4437         if MODE_CapsLock {
4438             inmeta = "A"
4439         }
4440         if MODE_LowerLock {
4441             inmeta = "a"
4442         }
4443         if MODE_ViTrace {
4444             inveri = "v"
4445         }
4446         if MODE_VicMode {
4447             inveri = ":"
4448         }
4449         if romkanmode {
4450             romkan = "\343\201\202"
4451             if MODE_CapsLock {
4452                 inmeta = "R"
4453             }else{
4454                 inmeta = "r"
4455             }
4456         }
4457         if inMeta {
4458             inmeta = "\\ "
4459         }
4460         showMode = "["+romkan+inmeta+inveri+"]";
4461     }
4462     Pre := "\r" + showMode + showLino
4463     Output := ""
4464     Left := ""
4465     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     }
4655     for i < len(str) {
4656         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4657         if lastSize <= 0 { break } // broken data?
4658         if !isAlnum(lastRune) { // character, type, or string to be searched
4659             break
4660         }
4661         lastW = i // the last alnum if in alnum word
4662         i += lastSize
4663     }
4664     if inWord {
4665         goto DISP
4666     }
4667     for i < len(str) {
4668         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4669         if lastSize <= 0 { break } // broken data?
4670         if isAlnum(lastRune) { // character, type, or string to be searched
4671             break
4672         }
4673         i += lastSize
4674     }
4675     for i < len(str) {
4676         lastRune,lastSize = utf8.DecodeRuneInString(str[i:])
4677         if lastSize <= 0 { break } // broken data?
4678         if !isAlnum(lastRune) { // character, type, or string to be searched
4679             break
4680         }
4681         lastW = i
4682         i += lastSize
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
5109     case '\t':
5110         iin.Replace('j');
5111         continue
5112     case 'X'-0x40:
5113         iin.Replace('j');
5114         continue
5115
5116     case EV_TIMEOUT:
5117         iin.Redraw();
5118         if iin.pinJmode {
5119             fprintf(stderr, "\\j\r\n")
5120             iin.inJmode = true
5121         }
5122         continue
5123     case GO_UP:
5124         if iin.lno == 1 {
5125             continue
5126         }
5127         cmd,ok := gsh.cmdStringInHistory(iin.lno-1)
5128         if ok {
5129             iin.line = cmd
5130             iin.right = ""
5131             iin.lno = iin.lno - 1
5132         }
5133         iin.Redraw();
5134         continue
5135     case GO_DOWN:
5136         cmd,ok := gsh.cmdStringInHistory(iin.lno+1)
5137         if ok {
5138             iin.line = cmd
5139             iin.right = ""
5140             iin.lno = iin.lno + 1
5141         }else{
5142             iin.line = ""
5143             iin.right = ""
5144             if iin.lno == iin.lastlno-1 {
5145                 iin.lno = iin.lno + 1
5146             }
5147         }
5148         iin.Redraw();
5149         continue
5150     case GO_LEFT:
5151         if 0 < len(iin.line) {
5152             xline,tail := delTailChar(iin.line)
5153             iin.line = xline
5154             iin.right = tail + iin.right
5155         }
5156         iin.Redraw();
5157         continue;
5158     case GO_RIGHT:
5159         if( 0 < len(iin.right) && iin.right[0] != 0 ){
5160             xright,head := delHeadChar(iin.right)
5161             iin.right = xright
5162             iin.line += head
5163         }
5164         iin.Redraw();
5165         continue;
5166     case EOF:
5167         goto EXIT;
5168     case 'R'-0x40: // replace
5169         dst := convs(iin.line+iin.right);
5170         iin.line = dst
5171         iin.right = ""
5172         iin.Redraw();
5173         continue;
5174     case 'n'-0x40: // just show the result
5175         readDic();
5176         romkanmode = !romkanmode;
5177         iin.Redraw();
5178         continue;
5179     case 'L'-0x40:
5180         iin.Redraw();
5181         continue
5182     case 'K'-0x40:
5183         iin.right = ""
5184         iin.Redraw();
5185         continue
5186     case 'E'-0x40:
5187         iin.line += iin.right
5188         iin.right = ""
5189         iin.Redraw();
5190         continue
5191     case 'A'-0x40:
5192         iin.right = iin.line + iin.right
5193         iin.line = ""
5194         iin.Redraw();
5195         continue
5196     case 'U'-0x40:
5197         iin.line = ""
5198         iin.right = ""
5199         iin.clearline();
5200         iin.Redraw();
5201         continue;
5202     case DEL_RIGHT:
5203         if( 0 < len(iin.right) ){
5204             iin.right,_ = delHeadChar(iin.right)
5205             iin.Redraw();
5206         }
5207         continue;
5208     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 "aWthcmFxeOBoe0Bi+OCi0p0aWthcmEJ5YqbCmNoaWthcmEJ5YqbCjwvdGV4dGFyZWE+CG="
5583 //</span>
5584
5585 var JA_JKLDic = //<span id="gsh-ja-jkl-dic">
5586 "data:text/dic;base64, "+
5587 "Iy92Z2JscU15SUI1FamRpy2ptb3JzZWRpK0WpKS0wMjAyMGowODE5K5KSheLV4pL1NhdG94SVRT"+
5588 "CmtqamprbGtqa2tsa2psIOS4lueVjApqamgtamwJ44GCmtqbAnjgYkKa2tqbAnjgYkYkamtq"+
5589 "amwJ44GCmtqa2trbAnjgYkKa2pra2wJ44GLCmpramrbAnjgYkKa2tramwJ44GCmpramps"+
5590 "CeObkQppqamqbAnjgZMKamtqa2psCeOb1Qpamtqa2wJ44GCmpqamtqbAnjgZkKa2pqamts"+
5591 "CeO8mpqamprbAnjgZ0KamtsCeObNwpra2prbAnjgEka2pqa2wJ44GkCmtqa2pqbAnjgYkYk"+
5592 "a2tqa2tsCeObqApramtsCeObGppqa2prbAnjgKa2tra2wJ44GkCmpqa2psCeObRQpra2pq"+
5593 "bAnjg4Kamtra2wJ44GvCmpqa2tqbAnjgIKampra2wJ44G1CmtsCeObuApga2tsCeObuwpq"+
5594 "a2tqbAnjg4Ka2tqa2psCeObvwpqbAnjgAKamtra2psCeOCgOpqa2tqa2wJ44KCmtqamwJ"+
5595 "44KCmprap2pqbAnjgYkampsCeOCiApra2tsCeOCiOpqamtsCeOCiGppqa2pqa2wJ44KLCmpq"+
5596 "amwJ44KCmtqa2psCeOCjOpqa2psCeOCjwpramramwJ44KQcmtqamrbAnjgEka2pqa2wJ"+
5597 "44KSCmtqa2prbAnjgpmKa2pqa2psCeObvApra2wJ44KbCmtrampbrbAnjgpmKa2pramtqbAnj"+
5598 "gIEK";
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 <a href="https://mdn-web-dna.s3-us-west-2.amazonaws.com/MDN-Browser-Compatibility-Report-2020.pdf">MDN Browser Compatibility Report (2020)</a>
5623
5624 Go language (August 2020 / Go 1.15)
5625 <a href="https://golang.org">The Go Programming Language</a>
5626 <a href="https://golang.org/pkg/">Packages</a>
5627 <a href="https://godoc.org/golang.org/x/net/websocket">WebSocket</a>
5628
5629 <a href="https://stackoverflow.com/">Stackoverflow</a>
5630 <!--
5631 <iframe src="https://golang.org" width="100%" height="300"></iframe>
5632 -->
5633 </div></details>
5634 */
5635
5636 <details id="html-src" onclick="frame_open();"><summary>Raw Source</summary><div>
5637
5638 <!-- h2>The full of this HTML including the Go code is here.</h2 -->
5639 <details id="gsh-whole-view"><summary>Whole file</summary>
5640 <a name="whole-src-view"></a>
5641 <span id="src-frame"></span><!-- a window to show source code -->
5642 </details>
5643
5644 <details id="gsh-style-frame" onclick="fill_CSSView()"><summary>CSS part</summary>
5645 <a name="style-src-view"></a>
5646 <span id="gsh-style-view"></span>
5647 </details>
5648
5649 <details id="gsh-script-frame" onclick="fill_JavaScriptView()"><summary>JavaScript part</summary>
5650 <a name="script-src-view"></a>
5651 <span id="gsh-script-view"></span>
5652 </details>
5653
5654 <details id="gsh-data-frame" onclick="fill_DataView()"><summary>Builtin data part</summary>
5655 <a name="gsh-data-frame"></a>
5656 <span id="gsh-data-view"></span>
5657 </details>
5658
5659 </div></details>
5660 */
5661
5662 /*
5663 <div id="GshFooter0"></div>
5664 <!-- 2020-09-17 SatoxITS, visible script { -- >
5665 <details><summary>GJScript</summary>
5666 <style>gjscript { font-family:Georgia; }</style>
5667 <pre id="gjscript_1" class="gjscript"> function gjtest1(){ alert('Hello GJScript!'); }
5668 gjtest1()
5669 </pre>
5670 <script>
5671 gjs = document.getElementById('gjscript_1');
5672 //eval(gjs.innerHTML);
5673 //gjs.outerHTML = " "
5674 </script>
5675 </details><!-- ----- END-OF-VISIBLE-PART ----- } -->
5676
5677 <!--
5678 // 2020-0906 added,
5679 https://developer.mozilla.org/en-US/docs/Web/CSS/z-index
5680 https://developer.mozilla.org/en-US/docs/Web/CSS/position
5681 -->
5682 <span id="GshGrid">(^_^)</small>(Hit j k l h)</small></span>
5683
5684 <span id="GStat"><br>
5685 </span>
5686 <span id="GMenu" onclick="GShellMenu(this)"></span>
5687 <span id="Gmp"></span>
5688 <div id="GShellPlane" onclick="showGShellPlane();"></div>
5689 <div id="RawTextViewer"></div>
5690 <div id="RawTextViewerClose" onclick="hideRawTextViewer()"> CLOSE </div>
5691
5692 <style id="GshStyleDef">
5693 #LineNumbered table,tr,td {
5694 margin:0;
5695 padding:4px;
5696 spacing:0;
5697 border:12px;
5698 }
5699 textarea.LineNumber {
5700 font-size:12px;
5701 font-family:monospace,Courier New;
5702 color:#282;
5703 padding:4px;
5704 text-align:right;

```

```

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

```



```

6077 BnasW4zhsv+2rDyr5qzQAvdp5Welit/GQSum2YW6HgmgfFJR0/y4aaU+7Gffy1+L80KKwC+3\
6078 AjzxxVwCLD+BJJ07RA61HTuc8jclEpNNZ25qz4P58xK09H9p55d2S3TmJngu8zUPTN+OL/PC\
6079 dc2P4UFahmsfn47H6RP12VhwjzrZ5LuFLwSLBs0F772KosQJyTaN2fL0OaGkG4U6b8+BXyQ\
6080 TKVWenKpupTqZ2ftH6ghz6/j88aPKnoB59jZLh9L+084E59SUQhki65Vw6P3njjyDW\
6081 85ziR500l+qabR6Tso+rrzbQxwv2xr0csSSmQI/fcFV7LPdZ2LJzr5KK+C5dELh6ixYTW\
6082 v1/nm4/cmbCW+nXmNwee48Z2nelWaoF+EKyUroI1DOGL3PawpZRGFP1nThtCOXQ5CLQPQ\
6083 RGJ4fbi+LEuY3VncC8bZFKV1szszfVNVs6qjsQw++Y0t29BUzqrJgWZDIoBJWxzE18vIx\
6084 KEPL9fdsBxp/2Xs6sgXKM3dfCLatfd8adBN1u0UNh1Afwg6Bw93sevqj1HMULPw/b14a6npg\
6085 pksWlwr06ffYm+v3M1U6MwfbD3KwyW5TWxj2zUcu04jSj+6WUyJbTLlpSv1okE327S/NwX\
6086 Mg+21W6vtW1Ti4TY/bUnDxms7iu9baga20YX5dbUX1z9BRpGEvdRDHJ51k3m3z394VgdSYp\
6087 qZbnk1kQbbVhteH161/Ovu/ZgszaeFLR+LN0BCXy90Ka7q7BBt6tbuV/oYiPh8xhzA4R7\
6088 o1MaOu3Q4pYZWHWct1a17Ndi3bXo2P7v2p70cmmeEYpewc8L4Q6770Ev+htZPnED+mNPy/W2\
6089 9LRAHr5EJj/v05gf1lw7StTREMd4gAu5rQ3T6aRSqdmx7z+/GB47ui290UWv9JX4AnJ711IS\
6090 16V+xi8fmsYcrQxul4K1ysH6Be9110YHs79i/4cxvgnH2jWB1j1XXxYQuZU0g5WDluq\
6091 Y6xMfG2XRCb6YrTjP5N07ZuP3v9irmNn4F5eSbKoHotab6aStQ07/beUgSubPmhrC27B1\
6092 0YQhS10JvcYk04fxKoZ+kgw+oaJdVEsagVEr9PehP+SrXwnKNNLm6VpQnUiKxIzm+0PveQqf\
6093 h4F81j9Wwot+64Stf500WBzd2G5tCd/FZS/VXH3nagrU1+4B2j8m8SsS/FmI9D3McBjwo\
6094 kRn3kXzuzqzskZkU1cbOCRjpmWhQ1aBxM1gsdu1315d3JL9ryw0Vm9Np1kT1E/CQYIdtWZV2\
6095 8/KpXgnKvkc1bdveIDDT0usc+RmxsD1pnxmw178tYm6HZGdCt2fgznJ+8xzuSRBvQ4zrhEw9\
6096 H9268VJSOHFzRdI17AAPQwzK17LsplurBj0QpxTbyb/8dmn2//11/gnago2AwqF/38+WE1\
6097 I4ae5Q5EXMARkAo2CCP2xwJW+1Mz78LkH3V2LWVt2n9w4/+6JgdXJFJgq7b1fADkwiP\
6098 nHS+Qe+H1dY5RPUvengZ0d6NF7K0t1oF1dJ/1kUtsatCEBE1ABE1ABE1ABE1ABE1AB\
6099 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6100 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6101 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6102 E1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1ABE1AB\
6103
6104 ITSMOREQR="data:image/png;base64,\
6105 iVBORw0KG0AAAAANSUHEUGAAAG8AAABvQMAAADYCWjAAAABLMVEX///9BaeFhgDAAAB\
6106 HkLEQVQ4jdTsa2EMawGYCM7sICkVgjXvACBe7CarASXda1LAWgS4HwM5zEVS+mvSgS+ZBQ\
6107 8gcB4BdHyZv8wS2MSaUBHNm+Kad4QC8LDpDn8ogT4UpFGci2j8IGF3eLWPaHknVyWecev\
6108 UebDXab0ZANjueYDzNklQassPckjc4nW3E1SfWqYk6jU/vAkPhg0ALSPhve8J0dkwDMwr\
6109 YMGSSuYWHAr19k0tkV2sb3sdw2RUCqW88g4Rp1A9s1JpV9tP1NRD4XFkin8XaQCITw6Lzq\
6110 Z08dW/4+U2GzqlS8gbqVmkFr1N6YXK8OqLD00mlGTmVzPERA8AL9vVbOifpSOL33fsYrTL\
6111 89wiqDznhUI38v5n783/gBuUs2eLg1c8gAAABJRU5ErkJggg==";
6112
6113 </script>
6114
6115 <div id="GJFactory_1" class="xxxGJFactory" style=""></div>
6116 <!--
6117 https://developer.mozilla.org/en-US/docs/Web/CSS/line-height
6118 -->
6119 <style>
6120 .GJFactory{
6121   resize:both; overflow:scroll;
6122   position:static;
6123   border:1.2px dashed #282; xborder-radius:2px;
6124   margin:0px; padding:10px !important;
6125   width:340px; height:340px;
6126   flex-wrap: wrap;
6127   color:#fff; background-color:rgba(0,0,0,0.0);
6128   line-height:0.0;
6129   xxxcolor:#22a !important;
6130   text-shadow:2px 2px #ddf;
6131 }
6132 .GJFactory h1,h2,h3,h4 {
6133   xxxcolor:#22a !important;
6134 }
6135 xxxinput {
6136   border:1px dashed #0f0; border-radius:0px;
6137 }
6138 .GJWin:hover{
6139   color:#df8 !important;
6140   background-color:rgba(32,32,160,0.8) !important;
6141   line-height:0.0;
6142 }
6143 .GJWin:active{
6144   color:#df8 !important;
6145   background-color:rgba(224,32,32,0.8) !important;
6146   line-height:0.0;
6147 }
6148 .GJWin:focus{
6149   color:#df8 !important;
6150   background-color:rgba(32,32,32,1.0) !important;
6151   line-height:0.0;
6152 }
6153 .GJWin{
6154   z-index:10000;
6155   display:inline;
6156   position:relative;
6157   flex-wrap: wrap;
6158   top:0; left:0px;
6159   width:285px !important; height:205px !important;
6160   border:1px solid #eea; border-radius:2px;
6161   margin:0px; padding:0px;
6162   font-size:8pt;
6163   line-height:0.0;
6164   color:#fff; background-color:rgba(0,0,64,0.1) !important;
6165 }
6166 .GJTab{
6167   display:inline;
6168   position:relative;
6169   top:0px; left:0px;
6170   margin:0px; padding:2px;
6171   border:0px solid #000; border-radius:2px;
6172   width:90px; height:20px;
6173   font-family:Georgia;
6174   font-size:9pt;
6175   line-height:1.0;
6176   white-space:nowrap;
6177   color:#fff; background-color:rgba(0,0,64,0.7);
6178   text-align:center;
6179   vertical-align:middle;
6180 }
6181 .GJStat:focus{
6182   color:#df8 !important;
6183   background-color:rgba(32,32,32,1.0) !important;
6184   line-height:1.0;
6185 }
6186 .GJStat{
6187   display:inline;
6188   position:relative;
6189   top:0px; left:0px;
6190   margin:0px; padding:2px;
6191   border:0px solid #00f; border-radius:2px;
6192   width:166px; height:20px;
6193   font-family:monospace;
6194   font-size:9pt;
6195   line-height:1.0;
6196   color:#fff; background-color:rgba(0,0,64,0.2);
6197   text-align:center;
6198   vertical-align:middle;
6199 }
6200 .GJIcon{

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

```

7069 GshMenuSign.style.color = "rgba(255,128,128,1.0)";
7070 RawTextViewerClose.style.top = 0;
7071 RawTextViewerClose.style = null
7072 }
7073 }
7074 // source code view
7075 function frame_close(){
7076   srcframe = document.getElementById("src-frame");
7077   srcframe.innterHTML = "";
7078   //srcframe.style.cols = 1;
7079   srcframe.style.rows = 1;
7080   srcframe.style.height = 0;
7081   srcframe.style.display = false;
7082   src = document.getElementById("SrcTextarea");
7083   src.innerHTML = ""
7084   //src.cols = 0
7085   src.rows = 0
7086   src.display = false
7087   //alert("--closed--")
7088 }
7089 //<!-- | <span onclick="html_view();">Source</span> -->
7090 //<!-- | <span onclick="frame_close();">SourceClose</span> -->
7091 //<!--| <span>Download</span> -->
7092 function frame_open(){
7093   if( DestroyFooter != null ) DestroyFooter();
7094   document.getElementById('GshFaviconURL').href = "";
7095   oldsrc = document.getElementById("GENSRC");
7096   if( oldsrc != null ){
7097     //alert("--I--(erasing old text)")
7098     oldsrc.innterHTML = "";
7099     return
7100   }else{
7101     //alert("--I--(no old text)")
7102   }
7103   styleBanner = GshBanner.getAttribute("style")
7104   GshBanner.removeAttribute("style")
7105   if( document.getElementById('GJC_1') ){ GJC_1.remove() }
7106 }
7107 GshFaviconURL.href = "";
7108 GStat.removeAttribute('style')
7109 GshGrid.removeAttribute('style')
7110 GshMenuSign.removeAttribute('style')
7111 //GPos.removeAttribute('style')
7112 //GPos.innerHTML = "";
7113 //GLog.removeAttribute('style')
7114 //GLog.innerHTML = "";
7115 GMenu.removeAttribute('style')
7116 GTop.removeAttribute('style')
7117 GShellPlane.removeAttribute('style')
7118 RawTextViewer.removeAttribute('style')
7119 RawTextViewerClose.removeAttribute('style')
7120 }
7121 if( DestroyGJLink != null ) DestroyGJLink();
7122 GJFactory_Destroy()
7123 Destroy_WirtualDesktop();
7124 DestroyEventSharingCodeview();
7125 }
7126 src = document.getElementById("gsh");
7127 srcthtml = src.outerHTML
7128 srcframe = document.getElementById("src-frame");
7129 srcframe.innerHTML = ""
7130 + "<"+<cite id="GENSRC">\n"
7131 + "<"+<style>\n"
7132 + "#GENSRC textarea{tab-size:4;}\n"
7133 + "#GENSRC textarea{-o-tab-size:4;}\n"
7134 + "#GENSRC textarea{-moz-tab-size:4;}\n"
7135 + "#GENSRC textarea{spellcheck:false;}\n"
7136 + "</"+<style>\n"
7137 + "<"+<textarea id="SrcTextarea" cols=100 rows=20 class="gsh-code" spellcheck="false">'
7138 + /*<"+<html>\n" // lost preamble text
7139 + srcthtml
7140 + "<"+</html>\n" // lost trail text
7141 + "<"+<textarea>\n"
7142 + "</"+<cite><!-- GENSRC -->\n";
7143 }
7144 //srcframe.style.cols = 80;
7145 //srcframe.style.rows = 80;
7146 }
7147 GshBanner.setAttribute('style',styleBanner)
7148 }
7149 function fill_CSSView(){
7150   part = document.getElementById('GshStyleDef')
7151   view = document.getElementById('gsh-style-view')
7152   view.innerHTML = ""
7153   + "<"+<textarea cols=100 rows=20 class="gsh-code">'
7154   + part.innerHTML
7155   + "<"+</textarea>"
7156 }
7157 function fill_JavaScriptView(){
7158   jspart = document.getElementById('gsh-script')
7159   view = document.getElementById('gsh-script-view')
7160   view.innerHTML = ""
7161   + "<"+<textarea cols=100 rows=20 class="gsh-code">'
7162   + jspart.innerHTML
7163   + "<"+</textarea>"
7164 }
7165 function fill_DataView(){
7166   part = document.getElementById('gsh-data')
7167   view = document.getElementById('gsh-data-view')
7168   view.innerHTML = ""
7169   + "<"+<textarea cols=100 rows=20 class="gsh-code">'
7170   + part.innerHTML
7171   + "<"+</textarea>"
7172 }
7173 function jumpto_StyleView(){
7174   jsview = document.getElementById('html-src')
7175   jsview.open = true
7176   jsview = document.getElementById('gsh-style-frame')
7177   jsview.open = true
7178   fill_CSSView()
7179 }
7180 function jumpto_JavaScriptView(){
7181   jsview = document.getElementById('html-src')
7182   jsview.open = true
7183   jsview = document.getElementById('gsh-script-frame')
7184   jsview.open = true
7185   fill_JavaScriptView()
7186 }
7187 function jumpto_DataView(){
7188   jsview = document.getElementById('html-src')
7189   jsview.open = true
7190   jsview = document.getElementById('gsh-data-frame')
7191   jsview.open = true
7192   fill_DataView()

```

```

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

```

```

7317 h += 'y'+y+", "+ 'x'+x+" -- "
7318 h += "w=" + window.innerWidth + ", h=" + window.innerHeight + " -- "
7319 //GPos.test = h
7320 //GPos.innerHTML = h
7321 // GPos.innerHTML = h
7322 }
7323
7324 function zero2(n){
7325   if( n < 10 ){
7326     return '0' + n;
7327   }else{
7328     return n;
7329   }
7330 }
7331 function DateHourMin(){
7332   d = new Date();
7333   //return '%02d:%02d'.sprintf(d.getHours(),d.getMinutes())
7334   return zero2(d.getHours()) + ":" + zero2(d.getMinutes());
7335 }
7336 function DateShort(){
7337   d = new Date()
7338   return d.getFullYear()
7339   + "/" + zero2(d.getMonth())
7340   + "/" + zero2(d.getDate())
7341   + " " + zero2(d.getHours())
7342   + ":" + zero2(d.getMinutes())
7343   + ":" + zero2(d.getSeconds())
7344 }
7345 function DateLong0(ms){
7346   d = new Date();
7347   d.setTime(ms);
7348   return
7349   //d.getFullYear() + "/" + d.getMonth() + "/" + d.getDate() + " "
7350   //+ d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
7351   DateShort()
7352   + "." + d.getMilliseconds()
7353   + " " + d.getTimezoneOffset()/60
7354   + " "
7355   + d.getTime() + "." + d.getMilliseconds()
7356 }
7357 }
7358 function DateLong(){
7359   return DateLong0(new Date());
7360 }
7361 function GShellMenu(e){
7362   //GLog.innerHTML = "Hello, World! (" + DateLong() + ")"
7363   showGShellPlane()
7364 }
7365 // placements of planes
7366 function GShellResizeX(ev){
7367   //if( document.getElementById("GMenu") != null ){
7368     GMenu.style.left = window.innerWidth - 100
7369     GMenu.style.top = window.innerHeight - 90 - 200
7370     //console.log("place GMENU "+GMenu.style.left+" "+GMenu.style.top)
7371   }
7372   //}
7373   GStat.style.width = window.innerWidth
7374   //if( document.getElementById("GPos") != null ){
7375     //GPos.style.width = window.innerWidth
7376     //GPos.style.top = window.innerHeight - 30; //GPos.style.height
7377   //}
7378   //if( document.getElementById("GLog") != null ){
7379     // GLog.style.width = window.innerWidth
7380     //GLog.innerHTML = ""
7381   //}
7382   //if( document.getElementById("GLog") != null ){
7383     //GLog.innerHTML = "Resize: w=" + window.innerWidth +
7384     //", h=" + window.innerHeight
7385   //}
7386   showCurElementPosition(ev)
7387 }
7388 }
7389 function GShellResize(){
7390   GShellResizeX("[RESIZE]")
7391 }
7392 }
7393 window.onresize = GShellResize
7394 var prevNode = null
7395 var LogMouseMoveOverElement = false;
7396 function GJSH_OnMouseMove(ev){
7397   if( LogMouseMoveOverElement == false ){
7398     return;
7399   }
7400   x = ev.clientX
7401   y = ev.clientY
7402   d = new Date()
7403   t = d.getTime() / 1000
7404   if( document.elementFromPoint ){
7405     e = document.elementFromPoint(x,y)
7406     if( e != null ){
7407       if( e == prevNode ){
7408         console.log('Mo-'+t+'('+x+', '+y+') '
7409           +e.nodeType+' '+e.tagName+'#'+e.id)
7410         prevNode = e
7411       }
7412     }else{
7413       console.log(t+'('+x+', '+y+') no element')
7414     }
7415   }else{
7416     console.log(t+'('+x+', '+y+') no elementFromPoint')
7417   }
7418 }
7419 window.addEventListener('mousemove',GJSH_OnMouseMove);
7420 }
7421 function GJSH_OnMouseMoveScreen(ev){
7422   x = ev.screenX
7423   y = ev.screenY
7424   d = new Date()
7425   t = d.getTime() / 1000
7426   console.log(t+'('+x+', '+y+') no elementFromPoint')
7427 }
7428 //screen.addEventListener('mousemove',GJSH_OnMouseMoveScreen);
7429 }
7430 function ScrollToElement(oe,ne){
7431   ne.scrollIntoView()
7432   ny = ne.getBoudingClientRect().top.toFixed(0)
7433   nx = ne.getBoudingClientRect().left.toFixed(0)
7434   //GLog.innerHTML = "["+ny+", "+nx+"]"
7435   //window.scrollTo(0,0)
7436 }
7437 }
7438 GTop.style.backgroundColor = "rgba(0,0,0,0.0)"
7439 GshGrid.style.left = '250px';
7440 GshGrid.style.zIndex = 0
7441 if( false ){
7442   oy = oe.getBoudingClientRect().top.toFixed(0)

```

```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

8309 width:300pt !important;
8310 border:0px solid #000 !important;
8311 background-color:rgba(0,0,0,0);
8312 }
8313 .VisibleText {
8314 }
8315 .BlinderText {
8316 color:#000; background-color:#eee;
8317 }
8318 .joinButton {
8319 font-family:Georgia !important;
8320 font-size:11pt;
8321 line-height:1.1;
8322 height:18pt;
8323 width:50pt;
8324 padding:3px !important;
8325 text-align:center !important;
8326 border-color:#aaa !important;
8327 border-radius:5px;
8328 color:#fff; background-color:#4a4 !important;
8329 vertical-align:middle !important;
8330 }
8331 .SendButton {
8332 vertical-align:top;
8333 }
8334 .ws0_log {
8335 font-size:10pt;
8336 color:#000 !important;
8337 line-height:1.0;
8338 background-color:rgba(255,255,255,0.7) !important;
8339 font-family:Courier New,monospace !important;
8340 width:99.3%;
8341 white-space:pre;
8342 }
8343 </style>
8344
8345 <!-- Form autofill test
8346 Location: <input id="xxserv" type="text" value="https://192.168.10.1/boafirm/formLogin" size="80">
8347 <form method="POST" id="xxform" action="https://192.168.10.1/boafirm/formLogin">
8348 dest? <input id="XDS" name="dest" type="text" size="80" value="/index_contents.html">
8349 -->
8350 <details><summary>Form Auto. Filling</summary>
8351 <style>
8352 .xxinput { width:260pt !important; line-height:1.1 !important; margin:1px;
8353 display:inline !important; font-size:10pt !important; padding:1px !important;
8354 }
8355 </style>
8356 <span style="font-family:Courier New;color:black;font-size:12pt;" onactive="">
8357 <form method="POST" id="xxform" action="https://192.168.10.1/" style="white-space:pre;">
8358 Location: <input id="xxserv" class="xxinput" type="text" value="https://192.168.10.1/" size="80">
8359 Username: <input id="xxuser" class="xxinput" name="user" type="text" autocomplete="on">
8360 Password: <input id="xxpass" class="xxinput" name="pass" type="password" autocomplete="on">
8361 SessionId:<input id="xxssid" class="xxinput" name="SESSION_ID" type="text" size="80">
8362 <input id="xxsubm" class="xxinput" type="submit" value="Submit"></form>
8363 </span>
8364 <script>
8365 function XXSetFormAction(){
8366 xxform.setAttribute('action',xxserv.value);
8367 }
8368 xxform.setAttribute('action',xxserv.value);
8369 xxserv.addEventListener('change',XXSetFormAction);
8370 //xxserv.value = location.href;
8371 </script>
8372 </details>
8373 */
8374
8375 /*
8376 <details id="BlinderTextClass" class="gsh-src"><summary>BlinderText</summary>
8377 <span id="BlinderTextScript">
8378 // https://w3c.github.io/uievents/#event-type-keydown
8379 //
8380 // 2020-09-21 class BlinderText - textarea element not to be readable
8381 //
8382 // BlinderText attributes
8383 // bl_plainText - null
8384 // bl_hideChecksum - [false]
8385 // bl_showLength - [false]
8386 // bl_visible - [false]
8387 // data-bl_config - []
8388 // - min. length
8389 // - max. length
8390 // - acceptable charset in generate text
8391 //
8392 function BlinderChecksum(text){
8393 plain = text.bl_plainText;
8394 return strCRC32(plain,plain.length).toFixed(0);
8395 }
8396 function BlinderKeydown(ev){
8397 pass = ev.target
8398 if( ev.code == 'Enter' ){
8399 ev.preventDefault();
8400 }
8401 ev.stopPropagation()
8402 }
8403 function BlinderKeyupl(ev){
8404 blind = ev.target
8405 if( ev.code == 'Backspace'){
8406 blind.bl_plainText = blind.bl_plainText.slice(0,blind.bl_plainText.length-1)
8407 }else
8408 if( and(ev.code == 'KeyV', ev.ctrlKey) ){
8409 blind.bl_visible = !blind.bl_visible;
8410 }else
8411 if( and(ev.code == 'KeyL', ev.ctrlKey) ){
8412 blind.bl_showLength = !blind.bl_showLength;
8413 }else
8414 if( and(ev.code == 'KeyU', ev.ctrlKey) ){
8415 blind.bl_plainText = "";
8416 }else
8417 if( and(ev.code == 'KeyR', ev.ctrlKey) ){
8418 checksum = BlinderChecksum(blind);
8419 blind.bl_plainText = checksum; //.toString(32);
8420 }else
8421 if( ev.code == 'Enter' ){
8422 ev.stopPropagation();
8423 ev.preventDefault();
8424 return;
8425 }else
8426 if( ev.key.length != 1 ){
8427 console.log('KeyUp: '+ev.code+'/' +ev.key);
8428 return;
8429 }else{
8430 blind.bl_plainText += ev.key;
8431 }
8432 }

```

```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

9177     }
9178 </script></span></div>
9179
9180 <style>.iframeTest{margin:3px;resize:both;width:370px;height:120px;}</style>
9181 <h2>Inter-window communicaiton</h2>
9182 <note>
9183 frame0 >>> frame1 and frame2<br>
9184 frame1 >>> frame0 and frame2<br>
9185 frame2 >>> frame0 and frame1<br>
9186 </note>
9187 <div id="iframe-test">
9188 <pre id="iframeHost" style="border:1px;font-family:Courier New;font-size:10pt;"></pre>
9189 <iframe id="iframe0" title="iframe0" class="iframeTest" style=""></iframe>
9190 <iframe id="iframe1" title="iframe1" class="iframeTest"></iframe>
9191 <iframe id="iframe2" title="iframe2" class="iframeTest"></iframe>
9192 </div>
9193
9194 <script id="if0-test-script">
9195 setupFrames0();
9196 setupFrames12();
9197
9198 function setFrameSrcdoc(dst,src){
9199     if( true ){
9200         dst.contentWindow.document.write(src);
9201         // this makes browser waite close, and crash if accumulated !?
9202         // so it should be closed after write
9203         dst.contentWindow.document.close();
9204     }else{
9205         // to be erased before source dump
9206         // but should be set for live snapshot
9207         dst.srcdoc = src;
9208     }
9209 }
9210 function setupFrames0(){
9211     ibody = iframe0.contentWindow.document.body;
9212     iframe0.style.width = "755px"
9213     //iframeHost.innerHTML = "Message exchange at iframes' host:\n";
9214     window.addEventListener('message',messageFromChild);
9215
9216     if0 = '';
9217     if0 += '<'+pre style="font-family:Courier New;">';
9218     if0 += '<input id="Fid" value="iframe0">';
9219     if0 += iftestTemplate.innerHTML;
9220     setFrameSrcdoc(iframe0,if0);
9221
9222     function clickOnChild(){
9223         console.log('clickOn #' +this.id);
9224     }
9225     function moveOnChild(){
9226         console.log('moveOn #' +this.id);
9227     }
9228     iframe0.contentWindow.document.body.style.setProperty('white-space','pre');
9229     iframe0.contentWindow.document.body.style.setProperty('font-size','9pt');
9230 }
9231 function setupFrames12(){
9232     if1 = '<input id="Fid" value="iframe1">';
9233     if1 += iftestTemplate.innerHTML;
9234     setFrameSrcdoc(iframe1,if1);
9235     //iframe1.name = 'iframe1'; // this seems break contentWindow
9236
9237     if2 = '<input id="Fid" value="iframe2">';
9238     if2 += iftestTemplate.innerHTML;
9239     setFrameSrcdoc(iframe2,if2);
9240
9241     iframe1.addEventListener('message',messageFromChild);
9242     //iframe1.addEventListener('mouseover',moveOnChild);
9243     iframe2.addEventListener('message',messageFromChild);
9244     //iframe2.addEventListener('mouseover',moveOnChild);
9245     iframe1.contentWindow.postMessage('[parent0] Hi iframe1 -- from parent.','*');
9246     //iframe1.contentWindow.postMessage('Your peer is '+iframe2.contentWindow,'*');
9247     iframe2.contentWindow.postMessage('[parent0] Hi iframe2 -- from parent.','*');
9248     //iframe2.contentWindow.postMessage('Your peer is '+iframe1.contentWindow,'*');
9249 }
9250 function messageFromChild(){
9251     from = null;
9252     forw = null;
9253     if( event.source == iframe0.contentWindow ){
9254         from = '[iframe0] '
9255         forw = 'iframe12';
9256     }else
9257     if( event.source == iframe1.contentWindow ){
9258         from = '[iframe1] '
9259         forw = 'iframe2';
9260     }else
9261     if( event.source == iframe2.contentWindow ){
9262         from = '[iframe2] '
9263         forw = 'iframe1';
9264     }else
9265     {
9266         iframeHost.innerHTML += 'Message [unknown] '
9267         + ' orig=' + event.origin
9268         + ' data=' + event.data
9269         //+ ' from=' + event.source
9270         ;
9271     }
9272     msglog1 = from + event.data + ' -- '
9273     + ' from=' + event.source
9274     + ' orig=' + event.origin
9275     + ' name=' + event.source.name
9276     //+ ' port=' + event.ports
9277     //+ ' evid=' + event.lastEventId
9278     + '\n'
9279     ;
9280     if( true ){
9281         if( forw == 'iframe1' || forw == 'iframe12' ){
9282             iframe1.contentWindow.postMessage(from+event.data);
9283         }
9284         if( forw == 'iframe2' || forw == 'iframe12' ){
9285             iframe2.contentWindow.postMessage(from+event.data);
9286         }
9287     }
9288     txtadd0(msglog1);
9289
9290     function txtadd0(txt){
9291         iframe0.contentWindow.document.body.append(txt);
9292         iframe0.contentWindow.scrollTo(0,100000);
9293     }
9294 }
9295 function es_ShowSelf(){
9296     iframe1.setAttribute('src',document.URL);
9297     iframe2.setAttribute('src',document.URL);
9298 }
9299 </script>
9300

```

```

9301 <input class="HtmlCodeviewButton" type="button" value="ShowSelf" onclick="es_ShowSelf()">
9302 <input class="HtmlCodeviewButton" type="button" value="ShowCode" onclick="es_showHtmlCode()">
9303 <span id="EventSharingCodeview"></span>
9304 <script id="EventSharingScript">
9305 function es_showHtmlCode(){
9306     showHtmlCode(EventSharingCodeview,EventSharingCodeSpan);
9307 }
9308 DestroyEventSharingCodeview = function(){
9309     //EventSharingCodeview.parentNode.removeChild(EventSharingCodeview);
9310     EventSharingCodeview.innerHTML = "";
9311     iframe0.style = "";
9312     //iframe0.srcdoc = "erased";
9313     //iframe1.srcdoc = "erased";
9314     //iframe2.srcdoc = "erased";
9315 }
9316 </script>
9317 <!-- EventSharing } -->
9318 </span>
9319 </details>
9320 */
9321 */
9322 /*
9323 <!-- ----- "GShell Inside" Notifitacion { -->
9324 <script id="script-gshell-inside">
9325 var notices = 0;
9326 function noticeGShellInside(){
9327     ver = '';
9328     if( ver = document.getElementById('GshVersion') ){
9329         ver = ver.innerHTML;
9330     }
9331     console.log('GJShell Inside (^~^)' + ver);
9332     notices += 1;
9333     if( 2 <= notices ){
9334         document.removeEventListener('mousemove',noticeGShellInside);
9335     }
9336 }
9337 document.addEventListener('mousemove',noticeGShellInside);
9338 noticeGShellInside();
9339
9340 const FooterName = 'GshFooter'
9341 function DestroyFooter(){
9342     if( (footer = document.getElementById(FooterName)) != null ){
9343         //footer.parentNode.removeChild(footer);
9344         empty = document.createElement('div');
9345         empty.id = 'GshFooter0';
9346         footer.parentNode.replaceChild(empty,footer);
9347     }
9348 }
9349
9350 footer = document.createElement('div');
9351 footer.id = FooterName;
9352 footer.style.backgroundImage = "url("+ITSmoreQR+)";
9353 //GshFooter0.parentNode.appendChild(footer);
9354 GshFooter0.parentNode.replaceChild(footer,GshFooter0);
9355 </script>
9356 <!-- } -->
9357
9358 <!--
9359     border:20px inset #888;
9360 -->
9361
9362 //<span id="WirtualDesktopCodeSpan">
9363 /*
9364 <details open=""><summary>Wirtual Desktop</summary>
9365 <!-- ----- Web Wirtual Desktop // 2020-0927 SatoxITS { -->
9366 <style>
9367 .WirtualDesktop {
9368     position:relative;
9369     resize:both !important;
9370     overflow:scroll;
9371     display:block;
9372     min-width:120px !important; min-height:60px !important;
9373     width:800px;
9374     height:480px;
9375     border:30px solid #222;
9376     border-radius:20px;
9377     background-color:rgba(32,32,160,0.8);
9378     background-image:url("WD-WallPaper03.png");
9379     background-size:100% 100%;}
9380 color:#22a;font-family:Georgia;font-size:10pt;
9381 }
9382 // specifying overflow seems fix the position of the element
9383 .WirtualBrowserSpan {
9384     position:relative;
9385     xxxposition:absolute;
9386     left:100px;
9387     top:100px;
9388     z-index:10;
9389     display:block;
9390     resize:both;
9391     width:500px;
9392     height:270px;
9393     min-width:40px; min-height:20px;
9394     background-color:rgba(255,200,160,1.0);
9395     overflow:scroll;
9396 }
9397 .xWirtualBrowserLocationBar:focus {
9398     color:#f00;
9399     background-color:rgba(255,128,128,0.2);
9400 }
9401 .xWirtualBrowserLocationBar:active {
9402     color:#f00;
9403     background-color:rgba(128,255,128,0.2);
9404 }
9405 a.WirtualBrowserLocation {
9406     color:#ccc !important;
9407     text-decoration:none;
9408 }
9409 a.WirtualBrowserLocation:hover {
9410     color:#fff !important;
9411     text-decoration:underline;
9412 }
9413 .WirtualBrowserLocationBar {
9414     position:absolute;
9415     z-index:100000;
9416     xxxdisplay:inline;
9417     display:block;
9418     width:400px;
9419     height:20px;
9420     padding-left:2px;
9421     line-height:1.1;
9422     vertical-align:middle;
9423     font-size:14px;
9424     color:#fff;

```

```

9425     background-color:rgba(128,128,128,0.2);
9426     font-family:Georgia;
9427 }
9428 .VirtualBrowserCommandBar {
9429     position:absolute;
9430     z-index:200000;
9431     xxxdisplay:inline;
9432     display:block;
9433     width:60px;
9434     height:20px;
9435     line-height:1.1;
9436     vertical-align:middle;
9437     font-size:14px;
9438     color:#fe4;
9439     background-color:rgba(128,128,128,0.1);
9440     font-family:Georgia;
9441     text-align:left;
9442     left:404px;
9443 }
9444 .VirtualBrowserFrame {
9445     position:relative;
9446     display:inline;
9447     z-index:10;
9448     resize:both;
9449     display:block;
9450     width:480px; height:240px;
9451     min-width:60px; min-height:30px;
9452     border-radius:6px;
9453     background-color:rgba(255,255,255,0.9);
9454     border-top:20px solid;
9455     border-right:4px solid;
9456     border-bottom:10px solid;
9457 }
9458 .WinFavicon {
9459     width:16px;
9460     height:16px;
9461     margin:1px;
9462     margin-right:3px;
9463     vertical-align:middle;
9464     background-color:rgba(255,255,255,1.0);
9465 }
9466 .VirtualDesktopMenuBar {
9467     color:#fff;
9468     font-size:7pt;
9469     text-align:right;
9470     padding-right:4px;
9471     background-color:rgba(128,128,128,0.7);
9472 }
9473 .VirtualDesktopCalender {
9474     color:#fff;
9475     font-size:22pt;
9476     text-align:right;
9477     padding-right:4px;
9478     xbackground-color:rgba(255,255,255,0.2);
9479 }
9480 .LiveHtmlCodeviewText {
9481     font-size:10pt;
9482     font-family:Courier New;
9483     xwhite-space:pre;
9484 }
9485 </style>
9486 <h2>Window Surfer 0.0.4</h2>
9487 <p>
9488 <a href="WD-WallPaper03.png">WD-WallPaler03.png</a>
9489 </p>
9490 <div id="VirtualDesktop_1" class="VirtualDesktop" draggable="true" contenteditable="true">
9491 <div id="VirtualDesktop_1_MenuBar" class="VirtualDesktopMenuBar" spellcheck="false">
9492 <i>Window Surfer 0.0.4</i>
9493 <span id="VirtualDesktop_1_Clock"></span>
9494 </div>
9495 <div id="VirtualDesktop_1_Calender" class="VirtualDesktopCalender" >00:00</div>
9496
9497 <div id="VirtualBrowser_1" class="VirtualBrowserSpan" spellcheck="false" draggable="true">
9498 <div id="VirtualBrowser_1_Location" class="VirtualBrowserLocationBar"></div>
9499 <span id="VirtualBrowser_1_Command" class="VirtualBrowserCommandBar">Reload</span>
9500 <iframe id="VirtualBrowser_1_Frame" class="VirtualBrowserFrame" style=""></iframe>
9501 </div>
9502
9503 <div id="VirtualBrowser_2" class="VirtualBrowserSpan" spellcheck="false" draggable="true">
9504 <div id="VirtualBrowser_2_Location" class="VirtualBrowserLocationBar"></div>
9505 <span id="VirtualBrowser_2_Command" class="VirtualBrowserCommandBar">Reload</span>
9506 <iframe id="VirtualBrowser_2_Frame" class="VirtualBrowserFrame" style=""></iframe>
9507 </div>
9508
9509 <div id="VirtualBrowser_3" class="VirtualBrowserSpan" spellcheck="false" draggable="true">
9510 <div id="VirtualBrowser_3_Location" class="VirtualBrowserLocationBar"></div>
9511 <span id="VirtualBrowser_3_Command" class="VirtualBrowserCommandBar">Reload</span>
9512 <iframe id="VirtualBrowser_3_Frame" class="VirtualBrowserFrame" style=""></iframe>
9513 </div>
9514
9515 </div>
9516 <input class="HtmlCodeviewButton" type="button" value="ShowCode" onclick="vd_showHtmlCode()">
9517 <span id="VirtualDesktopCodeview"></span>
9518 <script id="WirutalDesktopScript">
9519 function vd_showHtmlCode(){
9520     codespan = document.getElementById('VirtualDesktopCodeSpan');
9521     showHtmlCode(VirtualDesktopCodeview,codespan);
9522     WirtualDesktopCodeview.setAttribute('class','LiveHtmlCodeviewText');
9523 }
9524 DestroyEventSharingCodeview = function(){
9525     WirtualDesktopCodeview.innerHTML = "";
9526 }
9527
9528 function wdlog(log){
9529     GJ_SendMessage('WD '+log);
9530     console.log(log);
9531 }
9532 var topMostWin = 10000;
9533 function onEnterWin(e){
9534     t = e.target;
9535     oindex = t.style.zIndex;
9536     //if( oindex == '' ) oindex = 0;
9537     //t.saved_zIndex = oindex;
9538     //t.style.zIndex = 10000;
9539     topMostWin += 1;
9540     t.style.zIndex = topMostWin;
9541     nindex = t.style.zIndex;
9542     wdlog('Enter '+t+' #' + t.id + '(' + oindex + '-' + nindex + ')');
9543     e.stopPropagation();
9544     e.preventDefault();
9545 }
9546 function onClickWin(e){ // can detect click on the thick border?
9547     t = e.target;
9548     oindex = t.style.zIndex;

```

```

9549     topMostWin += 1;
9550     t.style.zIndex = topMostWin;
9551     nIndex = t.style.zIndex;
9552     wdlog('Click '+t+' #' + t.id + ('+oindex+'->'+nindex+''));
9553     //e.stopPropagation();
9554     //e.preventDefault();
9555 }
9556 function onLeaveWin(e){
9557     t = e.target;
9558     //oindex = t.style.zIndex;
9559     //nindex = t.saved_zIndex;
9560     //t.style.zIndex = nIndex;
9561     //wdlog('Leave '+e.target+' #' + e.target.id + ('+oindex+'->'+nindex+''));
9562     e.stopPropagation();
9563     e.preventDefault();
9564 }
9565
9566 var WinDragstartX; // event
9567 var WinDragstartY;
9568 var WinDragstartTX; // target
9569 var WinDragstartTY;
9570
9571 function onWinDragstart(e){
9572     WinDragstartX = e.x;
9573     WinDragstartY = e.y;
9574
9575     t = e.target;
9576
9577     //WinDragstartTX = t.getBoundingClientRect().left.toFixed(0)
9578     //WinDragstartTY = t.getBoundingClientRect().top.toFixed(0)
9579     if( t.style.left == '' ){
9580         WinDragstartTX = x0 = 0;
9581         t.style.left = '0px';
9582     }else{
9583         //WinDragstartTX = x0 = Number(t.style.left);
9584         WinDragstartTX = x0 = parseInt(t.style.left);
9585     }
9586     if( t.style.top == '' ){
9587         WinDragstartTY = y0 = 0;
9588         t.style.top = '0px';
9589     }else{
9590         //WinDragstartTY = y0 = Number(t.style.top);
9591         WinDragstartTY = y0 = parseInt(t.style.top);
9592     }
9593     if( true ){ // to be undo
9594         t.wasAtX = WinDragstartTX;
9595         t.wasAtY = WinDragstartTY;
9596     }
9597     wdlog('DragSTA #' + t.id
9598         + ' event('+e.x+', '+e.y+')'
9599         + ' position=' + t.style.position
9600         + ' style left,top('+t.style.left+', '+t.style.top+')'
9601     );
9602     e.stopPropagation();
9603     //e.preventDefault();
9604     return true;
9605 }
9606 function onWinDragEvent(wh,e,set,dolog){
9607     t = e.target;
9608     dx = e.x - WinDragstartX;
9609     dy = e.y - WinDragstartY;
9610     nx = WinDragstartTX + dx;
9611     ny = WinDragstartTY + dy;
9612     log = 'Drag'+wh+' #' + t.id
9613         + ' event0('+WinDragstartX+', '+WinDragstartY+')'
9614         + ' event('+e.x+', '+e.y+')'
9615         + ' diff('+dx+', '+dy+')'
9616         + ' (' + nx + ', ' + ny + '),'
9617         + ' (' + t.style.left + ', ' + t.style.top + '),'
9618         + ' wasAt('+ t.wasAtX + ', ' + t.wasAtY + '),'
9619     ;
9620     if( e.x != 0 || e.y != 0 ){
9621         if( set == true ){
9622             //t.style.x = nx + 'px'; // not effective
9623             //t.style.y = ny + 'px'; // not effective
9624             t.style.left = nx + 'px';
9625             t.style.top = ny + 'px';
9626             log += ' Set';
9627         }else{
9628             log += ' NotSet';
9629             if( !dolog ){
9630                 log = '';
9631             }
9632         }
9633     }else{
9634         log += ' What?'; // the type is event start?
9635         if( !dolog ){
9636             log = '';
9637         }
9638     }
9639     if( and(dolog, log != '' ) ){
9640         wdlog(log);
9641     }
9642     if( true ){
9643         // should be propargeted to parent in FireFox ?
9644         e.stopPropagation();
9645     }
9646     e.preventDefault();
9647     return false;
9648 }
9649 function onWinDrag(e){
9650     return onWinDragEvent('Ing',e,true,false);
9651 }
9652 function onWinDragend(e){
9653     return onWinDragEvent('End',e,false,true);
9654 }
9655 function onWinDragexit(e){
9656     return onWinDragEvent('Exit',e,false,true);
9657 }
9658 function onWinDragover(e){
9659     return onWinDragEvent('Over',e,false,true);
9660 }
9661 function onWinDragenter(e){
9662     return onWinDragEvent('Enter',e,false,true);
9663 }
9664 function onWinDragleave(e){
9665     return onWinDragEvent('Leave',e,false,true);
9666 }
9667 function onWinDragdrop(e){
9668     return onWinDragEvent('Drop',e,false,true);
9669 }
9670 function onFaviconChange(e){
9671     wdlog('--Favicon #' + e.target.id + ' href='+e.details.href);
9672 }

```

```

9673 var savedSuppressGJShell = false;
9674 function stopGShell(e){
9675     //wlog('enter Gsh STOP');
9676     savedSuppressGJShell = SuppressGJShell;
9677     SuppressGJShell = true;
9678     e.stopPropagation();
9679     e.preventDefault();
9680 }
9681 function contGShell(e){
9682     //wlog('leave Gsh STOP');
9683     SuppressGJShell = savedSuppressGJShell;
9684     e.stopPropagation();
9685     e.preventDefault();
9686 }
9687 function settleWin(s,l,cmd,f,u,x,y,c){
9688     function VirtualBrowserCommand(e,s,l,cmd,f){
9689         command = cmd.innerHTML
9690         if( command == "Reload" ){
9691             href_id = e.target.href_id;
9692             d = document.getElementById(href_id);
9693             wlog('href_tag=#'+href_id+'\n elem=#'+href_id+'\n href='+d);
9694             url = d.innerHTML;
9695             wlog('---- Load href_tag=#'+href_id+'\n elem=#'+href_id+'\n href='+d
9696                 +'\n url='+url);
9697             wlog('---- Load target #' + f.id + ' with url=' + url;
9698             f.src = url;
9699         }else{
9700             alert('unknown command "' + command + '" '+ e.target.id + ', '+ l.id + ', '+ f.id);
9701         }
9702     }
9703     function onKeyDown(e){
9704         if( e.code == 'Enter' ){
9705             e.stopPropagation();
9706             e.preventDefault();
9707         }
9708     }
9709     function onKeyUp(e){
9710         if( e.code == 'Enter' ){
9711             e.stopPropagation();
9712             e.preventDefault();
9713             // should reload immediately ?
9714         }
9715     }
9716     if( false ){
9717         wlog('start settle VirtualBrowser url='+u+'\n'
9718             + 'id=' + s.id + '\n'
9719             + 'width=' + s.style.width + '\n'
9720             + 'height=' + s.style.height
9721         );
9722     }
9723     // very important for WordPress ??
9724     s.style.width = f.style.width = 501; // for WordPress ...??
9725     s.style.height = f.style.height = 271; // for WordPress ...??
9726     if( false ){
9727         wlog('midway settle VirtualBrowser url='+u+'\n'
9728             + 'id=' + s.id + '\n'
9729             + 'width=' + s.style.width + '\n'
9730             + 'height=' + s.style.height
9731         );
9732     }
9733     s.width = 502; // for WordPress ...??
9734     s.height = 272; // for WordPress ...??
9735     if( false ){
9736         wlog('midway-2 settle VirtualBrowser url='+u+'\n'
9737             + 'id=' + s.id + '\n'
9738             + 'span-width=' + s.width + '\n'
9739             + 'span-height=' + s.height
9740         );
9741     }
9742     //wlog('--x1-- u'+u+' width s='+s.style.width+'f='+f.style.width);
9743     //wlog('--x2-- u'+u+' width s='+s.style.width+'f='+f.style.width);
9744     s.setAttribute('draggable','true');
9745     f.setAttribute('draggable','false'); // why necessary?
9746     l.setAttribute('draggable','false'); // why necessary?
9747     cmd.setAttribute('draggable','false'); // why necessary?
9748     s.addEventListener('dragstart', e => { onWinDragstart(e); });
9749     s.addEventListener('drag', e => { onWinDrag(e); });
9750     s.addEventListener('exit', e => { onWinDragexit(e); });
9751     s.addEventListener('dragend', e => { onWinDragend(e); });
9752     s.addEventListener('dragexit', e => { onWinDragexit(e); });
9753     s.addEventListener('dragenter', e => { onWinDragenter(e); });
9754     s.addEventListener('dragover', e => { onWinDragover(e); });
9755     s.addEventListener('dragleave', e => { onWinDragleave(e); });
9756     s.addEventListener('drop', e => { onWinDragdrop(e); });
9757     if( false ){
9758         s.style.position = "absolute";
9759         s.style.x = x+'px';
9760         s.style.left = x+'px';
9761         s.style.y = y+'px';
9762         s.style.top = y+'px';
9763     }else{
9764         s.style.setProperty('position','absolute','important');
9765         s.style.setProperty('x',x+'px','important');
9766         s.style.setProperty('left',x+'px','important');
9767         s.style.setProperty('y',y+'px','important');
9768         s.style.setProperty('top',y+'px','important');
9769     }
9770     favicon = './favicon.ico';
9771     uv1 = u.split('/:');
9772     if( 2 <= uv1.length ){
9773         uv2 = uv1[1].split('/');
9774         if( 2 <= uv2.length ){
9775             if( uv1[0] == 'file' ){
9776                 //favicon = 'file://' + uv2.slice(0,uv2.length-1).join('/');
9777                 // + './favicon.ico';
9778                 favicon = './favicon.ico';
9779             }else{
9780                 favicon = uv1[0] + '://' + uv2[0] + '/favicon.ico';
9781             }
9782         }
9783     }
9784     //wlog("---- favicon-url="+favicon);
9785     href_id = l.id + '_href';
9786     l.innerHTML = '>'+u+'</a>';
9788     //l.addEventListener('click', e => { onClickWin(e); });
9789     l.addEventListener('mouseenter',e => { stopGShell(e); });
9790     l.addEventListener('mouseleave',e => { contGShell(e); });
9791     l.addEventListener('keydown', e => { onKeyDown(e); });
9792     l.addEventListener('keyup', e => { onKeyUp(e); });

```

```

9797
9798     cmd.href_id = href_id;
9799     wdllog('(0)cmd=#'+cmd.id);
9800     wdllog('(1)href_id=#'+href_id);
9801     wdllog('(2)href_id=#'+cmd.href_id);
9802     cmd.addEventListener('click', e => { VirtualBrowserCommand(e,s,l,cmd,f); });
9803
9804     f.style.borderColor = c;
9805     f.src = u;
9806     f.addEventListener('mouseenter',e => { onEnterWin(e); });
9807     f.addEventListener('mouseleave',e => { onLeaveWin(e); });
9808
9809     //s.addEventListener('click', e => { onClickWin(e); });
9810     //f.addEventListener('click', e => { wdllog('click wbl'); });
9811     f.addEventListener('mozbrowsericonchange',onFaviconChange);
9812
9813     wdllog('done settle VirtualBrowser url='+u+'\n'
9814           + 'id=' + s.id + ' '
9815           + 'width=' + s.style.width + ' '
9816           + 'height=' + s.style.height + ' '
9817           + 'cmd=' + cmd.id
9818           );
9819 }
9820
9821 //VirtualDesktop_1.style.width = "800px";
9822 //VirtualDesktop_1.style.height = "500px";
9823 dt = VirtualDesktop_1;
9824 dt.style.width = "800px";
9825 dt.style.height = "500px";
9826 dt.addEventListener('dragstart',e => { onWinDragstart(e); });
9827 dt.addEventListener('drag', e => { onWinDrag(e); });
9828 dt.addEventListener('exit', e => { onWinDragexit(e); });
9829
9830 settleWin(
9831     VirtualBrowser_1,
9832     VirtualBrowser_1_Location,
9833     VirtualBrowser_1_Command,
9834     VirtualBrowser_1_Frame,
9835     document.URL,
9836     50,20,'#262');
9837 settleWin(
9838     VirtualBrowser_2,
9839     VirtualBrowser_2_Location,
9840     VirtualBrowser_2_Command,
9841     VirtualBrowser_2_Frame,
9842     'https://its-more.jp/ja_jp/',
9843     150,100,'#448');
9844 settleWin(
9845     VirtualBrowser_3,
9846     VirtualBrowser_3_Location,
9847     VirtualBrowser_3_Command,
9848     VirtualBrowser_3_Frame,
9849     '../gshell/gsh.go.html',
9850     // 'http://gshell.org/gshell/gsh.go.html',
9851     'https://golang.org',
9852     250,180,'#444');
9853
9854 function WD_ClockUpdate(e){
9855     VirtualDesktop_1_Clock.innerHTML = DateShort();
9856     VirtualDesktop_1_Calender.innerHTML = DateHourMin();
9857 }
9858 window.setInterval(WD_ClockUpdate,500);
9859 GJ_Join();
9860
9861 Destroy_VirtualDesktop = function(){
9862     VirtualBrowser_1.removeAttribute('style');
9863     VirtualBrowser_1_Location.innerHTML = '';
9864     VirtualBrowser_1_Frame.removeAttribute('src');
9865     VirtualBrowser_1_Frame.removeAttribute('style');
9866     VirtualBrowser_1_Frame.style="";
9867
9868     VirtualBrowser_2.removeAttribute('style');
9869     VirtualBrowser_2_Location.innerHTML = '';
9870     VirtualBrowser_2_Frame.removeAttribute('src');
9871     VirtualBrowser_2_Frame.style="";
9872
9873     VirtualBrowser_3.removeAttribute('style');
9874     VirtualBrowser_3_Location.innerHTML = '';
9875     VirtualBrowser_3_Frame.removeAttribute('src');
9876     VirtualBrowser_3_Frame.style="";
9877
9878     GJFactory_1.style = "";
9879     iframe0.style = "";
9880     VirtualDesktop_1.style = "";
9881 }
9882
9883 </script>
9884 <!-- VirtualDesktop } -->
9885 </details>
9886 */ </span>
9887
9888 <!-- ----- Work { ----- -->
9889 <span id="Template_WorkCodeSpan">
9890 /*
9891 <details><summary>Work Template</summary>
9892 <!-- ----- Template of Work// 2020-0928 SatoxITS { -->
9893 <h2>Template of Work</h2>
9894 <input id="Template_WorkCodeViewOpen" class="HtmlCodeViewButton" type="button" value="ShowCode">
9895 <span id="Template_WorkCodeView"></span>
9896 <script id="Template_WorkScript">
9897     function Tmplate_openWorkCodeView(){
9898         function Template_showWorkCode(){
9899             showHtmlCode(Template_WorkCodeView,Template_WorkCodeSpan);
9900         }
9901         Template_WorkCodeViewOpen.addEventListener('click',Template_showWorkCode);
9902     }
9903     Template_openWorkCodeView(); // should be invoked by an event
9904 </script>
9905 </details>
9906 <!-- Template_WorkCodeSpan } -->
9907 */ </span>
9908 <!-- ----- Work } ----- -->
9909
9910 <<br></span></html>
9911

```