```
/*<html>
<span id="gsh">
<link rel="icon" href="GShell-Logo05icon.png">
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>GShell-0.2.1 by SatoxITS</title>
<header id="banner" height="100px" onclick="shiftBG();" style="">
<div align="right"><note>GShell version 0.2.1 // 2020-08-25 // SatoxITS</note></div>
</header>
<h2>GShell // a General purpose Shell built on the top of Golang</h2>
<p>
<note>
It is a shell for myself, by myself, of myself. --SatoxITS(^-^)
</note>
</p>
<span id="gsh-menu">
| <span onclick="html_new();">NewWindow</span>
| <span onclick="html_open();">Unfold</span>
| <span onclick="html_fold();">Fold</span>
| <span onclick="html_stop();">Stop</span>
| <span onclick="html_close();">Close</span>
|</span>
*/
/*
<details id="html-src" onclick="frame_open();"><summary>Total Source of GShell</summary><div>
<h2>The full of this HTML including the Go code is here.</h2>
<span id="src-frame"></span> // a window to show source code
</div></details>
*/
/*
<details id="overview"><summary>Overview</summary><div class="gsh-src">
To be written
</div>
</details>
*/
/*
<details id="index">
<summary>Go Source Code Index</summary><div class="gsh-src" onclick="document.getElementById('gsh-gocode').open=true;">
Implementation
    Structures
        <a href="#import">import</a>
        <a href="#struct">struct</a>
    Main functions
        <a href="#comexpansion">str-expansion</a>   // macro processor
        <a href="#finder">finder</a>        // builtin find + du
        <a href="#grep">grep</a>        // builtin grep + wc + cksum + ...
        <a href="#plugin">plugin</a>        // plugin commands
        <a href="#ex-commands">system</a>       // external commands
        <a href="#builtin">builtin</a>      // builtin commands
        <a href="#network">network</a>      // socket handler
        <a href="#remote-sh">remote-sh</a>  // remote shell
        <a href="#redirect">redirect</a>    // StdIn/Out redireciton
        <a href="#history">history</a>      // command history
        <a href="#rusage">rusage</a>        // resouce usage
        <a href="#encode">encode</a>        // encode / decode
        <a href="#IME">IME</a>      // command line IME
        <a href="#getline">getline</a>      // line editor
        <a href="#scanf">scanf</a>      // string decomposer
        <a href="#interpreter">interpreter</a>  // command interpreter
        <a href="#main">main</a>
</div>
</details>
*/
//<details id="gsh-gocode">
//<summary>Go Source Code</summary><div class="gsh-src" onclick="document.getElementById('gsh-gocode').open=false;">
// gsh - Go lang based Shell
// (c) 2020 ITS more Co., Ltd.
// 2020-0807 created by SatoxITS (sato@its-more.jp)

package main // gsh main
// <a name="import">Imported packages</a> // <a href="https://golang.org/pkg/">Packages</a>
import (
    "fmt"       // <a href="https://golang.org/pkg/fmt/">fmt</a>
    "strings"   // <a href="https://golang.org/pkg/strings/">strings</a>
    "strconv"   // <a href="https://golang.org/pkg/strconv/">strconv</a>
    "sort"      // <a href="https://golang.org/pkg/sort/">sort</a>
    "time"      // <a href="https://golang.org/pkg/time/">time</a>
    "bufio"     // <a href="https://golang.org/pkg/bufio/">bufio</a>
    "io/ioutil" // <a href="https://golang.org/pkg/io/ioutil/">ioutil</a>
    "os"        // <a href="https://golang.org/pkg/os/">os</a>
    "syscall"   // <a href="https://golang.org/pkg/syscall/">syscall</a>
    "plugin"    // <a href="https://golang.org/pkg/plugin/">plugin</a>
    "net"       // <a href="https://golang.org/pkg/net/">net</a>
    "net/http"  // <a href="https://golang.org/pkg/net/http/">http</a>
    //"html"    // <a href="https://golang.org/pkg/html/">html</a>
    "path/filepath" // <a href="https://golang.org/pkg/path/filepath/">filepath</a>
    "go/types"  // <a href="https://golang.org/pkg/go/types/">types</a>
    "go/token"  // <a href="https://golang.org/pkg/go/token/">token</a>
    "encoding/base64"   // <a href="https://golang.org/pkg/encoding/base64/">base64</a>
    "unicode/utf8"  // <a href="https://golang.org/pkg/unicode/utf8/">utf8</a>
    //"gshdata" // gshell's logo and source code
    "hash/crc32"    // <a href="https://golang.org/pkg/unicode/hash/crc32/">crc32</a>
)

var NAME = "gsh"
var AUTHOR = "SatoxITS(^-^)/"
var VERSION = "0.2.1"
var DATE = "2020-08-25"
var LINESIZE = (8*1024)
var PATHSEP = ":"   // should be ";" in Windows
var DIRSEP = "/"    // canbe \ in Windows
var GSH_HOME = ".gsh"   // under home directory
var MaxStreamSize = int64(128*1024*1024*1024) // 128GiB is too large?
var PROMPT = "> "
var GSH_PORT = 9999

// -xX logging control
// --A-- all
// --I-- info.
// --D-- debug
// --T-- time and resource usage
// --W-- warning
// --E-- error
// --F-- fatal error
// --Xn- network

// <a name="struct">Structures</a>
type GCommandHistory struct {
    StartAt     time.Time // command line execution started at
    EndAt       time.Time // command line execution ended at
    ResCode     int       // exit code of (external command)
    CmdError    error     // error string
    OutData     *os.File  // output of the command
    FoundFile   []string  // output - result of ufind
```

```
	Rusagev     [2]syscall.Rusage // Resource consumption, CPU time or so
	CmdId       int       // maybe with identified with arguments or impact
	                  // redireciton commands should not be the CmdId
	WorkDir     string    // working directory at start
	WorkDirX    int       // index in ChdirHistory
	CmdLine     string    // command line
}
type GChdirHistory struct {
	Dir     string
	MovedAt     time.Time
	CmdIndex    int
}
type CmdMode struct {
	BackGround  bool
}
type PluginInfo struct {
	Spec        *plugin.Plugin
	Addr        plugin.Symbol
	Name        string // maybe relative
	Path        string // this is in Plugin but hidden
}
type GServer struct {
	host        string
	port        string
}

// <a href="https://tools.ietf.org/html/rfc3230">Digest</a>
const ( // SumType
	SUM_ITEMS   = 0x000001 // items count
	SUM_SIZE    = 0x000002 // data length (simplly added)
	SUM_SIZEHASH    = 0x000004 // data length (hashed sequence)
	SUM_DATEHASH    = 0x000008 // date of data (hashed sequence)
	// also envelope attributes like time stamp can be a part of digest
	// hashed value of sizes or mod-date of files will be useful to detect changes

	SUM_WORDS   = 0x000010 // word count is a kind of digest
	SUM_LINES   = 0x000020 // line count is a kind of digest
	SUM_SUM64   = 0x000040 // simple add of bytes, useful for human too

	SUM_SUM32_BITS  = 0x000100 // the number of true bits
	SUM_SUM32_2BYTE = 0x000200 // 16bits words
	SUM_SUM32_4BYTE = 0x000400 // 32bits words
	SUM_SUM32_8BYTE = 0x000800 // 64bits words

	SUM_SUM16_BSD   = 0x001000 // UNIXsum -sum -bsd
	SUM_SUM16_SYSV  = 0x002000 // UNIXsum -sum -sysv
	SUM_UNIXFILE    = 0x004000
	SUM_CRCIEEE = 0x008000
)
type CheckSum struct {
	Files       int64   // the number of files (or data)
	Size        int64   // content size
	Words       int64   // word count
	Lines       int64   // line count
	SumType     int
	Sum64       uint64
	Crc32Table  crc32.Table
	Crc32Val    uint32
	Sum16       int
	Ctime       time.Time
	Atime       time.Time
	Mtime       time.Time
	Start       time.Time
	Done        time.Time
	RusgAtStart [2]syscall.Rusage
	RusgAtEnd   [2]syscall.Rusage
}
type ValueStack [][]string
type GshContext struct {
	StartDir    string  // the current directory at the start
	GetLine     string  // gsh-getline command as a input line editor
	ChdirHistory    []GChdirHistory // the 1st entry is wd at the start
	gshPA       syscall.ProcAttr
	CommandHistory  []GCommandHistory
	CmdCurrent  GCommandHistory
	BackGround  bool
	BackGroundJobs  []int
	LastRusage  syscall.Rusage
	GshHomeDir  string
	TerminalId  int
	CmdTrace    bool // should be [map]
	CmdTime     bool // should be [map]
	PluginFuncs []PluginInfo
	iValues     []string
	iDelimiter  string // field sepearater of print out
	iFormat     string // default print format (of integer)
	iValStack   ValueStack
	LastServer  GServer
	RSERV       string // [gsh://]host[:port]
	RWD     string // remote (target, there) working directory
	lastCheckSum    CheckSum
}

func nsleep(ns time.Duration){
	time.Sleep(ns)
}
func usleep(ns time.Duration){
	nsleep(ns*1000)
}
func msleep(ns time.Duration){
	nsleep(ns*1000000)
}
func sleep(ns time.Duration){
	nsleep(ns*1000000000)
}

func strBegins(str, pat string)(bool){
	if len(pat) <= len(str){
		yes := str[0:len(pat)] == pat
		//fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat,yes)
		return yes
	}
	//fmt.Printf("--D-- strBegins(%v,%v)=%v\n",str,pat,false)
	return false
}
func isin(what string, list []string) bool {
	for _, v := range list  {
		if v == what {
			return true
		}
	}
	return false
}
func isinX(what string,list[]string)(int){
	for i,v := range list {
        if v == what {
            return i
        }
    }
    return -1
}

func env(opts []string) {
    env := os.Environ()
    if isin("-s", opts){
        sort.Slice(env, func(i,j int) bool {
            return env[i] < env[j]
        })
    }
    for _, v := range env {
        fmt.Printf("%v\n",v)
    }
}

// - rewriting should be context dependent
// - should postpone until the real point of evaluation
// - should rewrite only known notation of symobl
func scanInt(str string)(val int,leng int){
    leng = -1
    for i,ch := range str {
        if '0' <= ch && ch <= '9' {
            leng = i+1
        }else{
            break
        }
    }
    if 0 < leng {
        ival,_ := strconv.Atoi(str[0:leng])
        return ival,leng
    }else{
        return 0,0
    }
}
func substHistory(gshCtx *GshContext,str string,i int,rstr string)(leng int,rst string){
    if len(str[i+1:]) == 0 {
        return 0,rstr
    }
    hi := 0
    histlen := len(gshCtx.CommandHistory)
    if str[i+1] == '!' {
        hi = histlen - 1
        leng = 1
    }else{
        hi,leng = scanInt(str[i+1:])
        if leng == 0 {
            return 0,rstr
        }
        if hi < 0 {
            hi = histlen + hi
        }
    }
    if 0 <= hi && hi < histlen {
        var ext byte
        if 1 < len(str[i+leng:]) {
            ext = str[i+leng:][1]
        }
        //fmt.Printf("--D-- %v(%c)\n",str[i+leng:],str[i+leng])
        if ext == 'f' {
            leng += 1
            xlist := []string{}
            list := gshCtx.CommandHistory[hi].FoundFile
            for _,v := range list {
                //list[i] = escapeWhiteSP(v)
                xlist = append(xlist,escapeWhiteSP(v))
            }
            //rstr += strings.Join(list," ")
            rstr += strings.Join(xlist," ")
        }else
        if ext == '@' || ext == 'd' {
            // !N@ .. workdir at the start of the command
            leng += 1
            rstr += gshCtx.CommandHistory[hi].WorkDir
        }else{
            rstr += gshCtx.CommandHistory[hi].CmdLine
        }
    }else{
        leng = 0
    }
    return leng,rstr
}
func escapeWhiteSP(str string)(string){
    if len(str) == 0 {
        return "\\z" // empty, to be ignored
    }
    rstr := ""
    for _,ch := range str {
        switch ch {
            case '\\': rstr += "\\\\"
            case ' ': rstr += "\\s"
            case '\t': rstr += "\\t"
            case '\r': rstr += "\\r"
            case '\n': rstr += "\\n"
            default: rstr += string(ch)
        }
    }
    return rstr
}
func unescapeWhiteSP(str string)(string){ // strip original escapes
    rstr := ""
    for i := 0; i < len(str); i++ {
        ch := str[i]
        if ch == '\\' {
            if i+1 < len(str) {
                switch str[i+1] {
                    case 'z':
                        continue;
                }
            }
        }
        rstr += string(ch)
    }
    return rstr
}
func unescapeWhiteSPV(strv []string)([]string){ // strip original escapes
    ustrv := []string{}
    for _,v := range strv {
        ustrv = append(ustrv,unescapeWhiteSP(v))
    }
    return ustrv
}
```

```

// <a name="comexpansion">str-expansion</a>
// - this should be a macro processor
func strsubst(gshCtx *GshContext,str string,histonly bool) string {
	rbuff := []byte{}
	if false {
		//@@U Unicode should be cared as a character
		return str
	}
	//rstr := ""
	inEsc := 0 // escape characer mode
	for i := 0; i < len(str); i++ {
		//fmt.Printf("--D--Subst %v:%v\n",i,str[i:])
		ch := str[i]
		if inEsc == 0 {
			if ch == '!' {
				//leng,xrstr := substHistory(gshCtx,str,i,rstr)
				leng,rs := substHistory(gshCtx,str,i,"")
				if 0 < leng {
	//_,rs := substHistory(gshCtx,str,i,"")
	rbuff = append(rbuff,[]byte(rs)...)
					i += leng
					//rstr = xrstr
					continue
				}
			}
			switch ch {
				case '\\': inEsc = '\\'; continue
				//case '%':  inEsc = '%';  continue
				case '$':
			}
		}
		switch inEsc {
		case '\\':
			switch ch {
				case '\\': ch = '\\'
				case 's': ch = ' '
				case 't': ch = '\t'
				case 'r': ch = '\r'
				case 'n': ch = '\n'
				case 'z': inEsc = 0; continue // empty, to be ignored
			}
			inEsc = 0
		case '%':
			switch {
				case ch == '%': ch = '%'
				case ch == 'T':
					//rstr = rstr + time.Now().Format(time.Stamp)
	rs := time.Now().Format(time.Stamp)
	rbuff = append(rbuff,[]byte(rs)...)
					inEsc = 0
					continue;
				default:
					// postpone the interpretation
					//rstr = rstr + "%" + string(ch)
	rbuff = append(rbuff,ch)
					inEsc = 0
					continue;
			}
			inEsc = 0
		}
		//rstr = rstr + string(ch)
		rbuff = append(rbuff,ch)
	}
	//fmt.Printf("--D--subst(%s)(%s)\n",str,string(rbuff))
	return string(rbuff)
	//return rstr
}
func showFileInfo(path string, opts []string) {
	if isin("-l",opts) || isin("-ls",opts) {
		fi, err := os.Stat(path)
		if err != nil {
			fmt.Printf("---------- ((%v))",err)
		}else{
			mod := fi.ModTime()
			date := mod.Format(time.Stamp)
			fmt.Printf("%v %8v %s ",fi.Mode(),fi.Size(),date)
		}
	}
	fmt.Printf("%s",path)
	if isin("-sp",opts) {
		fmt.Printf(" ")
	}else
	if ! isin("-n",opts) {
		fmt.Printf("\n")
	}
}
func userHomeDir()(string,bool){
	/*
	homedir,_ = os.UserHomeDir() // not implemented in older Golang
	*/
	homedir,found := os.LookupEnv("HOME")
	//fmt.Printf("--I-- HOME=%v(%v)\n",homedir,found)
	if !found {
		return "/tmp",found
	}
	return homedir,found
}

func toFullpath(path string) (fullpath string) {
	if path[0] == '/' {
		return path
	}
	pathv := strings.Split(path,DIRSEP)
	switch {
	case pathv[0] == ".":
		pathv[0], _ = os.Getwd()
	case pathv[0] == "..": // all ones should be interpreted
		cwd, _ := os.Getwd()
		ppathv := strings.Split(cwd,DIRSEP)
		pathv[0] = strings.Join(ppathv,DIRSEP)
	case pathv[0] == "~":
		pathv[0],_ = userHomeDir()
	default:
		cwd, _ := os.Getwd()
		pathv[0] = cwd + DIRSEP + pathv[0]
	}
	return strings.Join(pathv,DIRSEP)
}

func IsRegFile(path string)(bool){
	fi, err := os.Stat(path)
	if err == nil {
		fm := fi.Mode()
		return fm.IsRegular();
	}
	return false
}

// <a name="encode">Encode / Decode</a>
// <a href="https://golang.org/pkg/encoding/base64/#example_NewEncoder">Encoder</a>
func (gshCtx *GshContext)Enc(argv[]string){
	file := os.Stdin
	buff := make([]byte,LINESIZE)
	li := 0
	encoder := base64.NewEncoder(base64.StdEncoding,os.Stdout)
	for li = 0; ; li++ {
		count, err := file.Read(buff)
		if count <= 0 {
			break
		}
		if err != nil {
			break
		}
		encoder.Write(buff[0:count])
	}
	encoder.Close()
}
func (gshCtx *GshContext)Dec(argv[]string){
	decoder := base64.NewDecoder(base64.StdEncoding,os.Stdin)
	li := 0
	buff := make([]byte,LINESIZE)
	for li = 0; ; li++ {
		count, err := decoder.Read(buff)
		if count <= 0 {
			break
		}
		if err != nil {
			break
		}
		os.Stdout.Write(buff[0:count])
	}
}
// lnsp [N] [-crlf][-C \\]
func (gshCtx *GshContext)SplitLine(argv[]string){
	reader := bufio.NewReaderSize(os.Stdin,64*1024)
	ni := 0
	toi := 0
	for ni = 0; ; ni++ {
		line, err := reader.ReadString('\n')
		if len(line) <= 0 {
			if err != nil {
			fmt.Fprintf(os.Stderr,"--I-- lnsp %d to %d (%v)\n",ni,toi,err)
			break
			}
		}
		off := 0
		ilen := len(line)
		remlen := len(line)
		for oi := 0; 0 < remlen; oi++ {
			olen := remlen
			addnl := false
			if 72 < olen {
				olen = 72
				addnl = true
			}
			fmt.Fprintf(os.Stderr,"--D-- write %d [%d.%d] %d %d/%d/%d\n",
				toi,ni,oi,off,olen,remlen,ilen)
			toi += 1
			os.Stdout.Write([]byte(line[0:olen]))
			if addnl {
				//os.Stdout.Write([]byte("\r\n"))
				os.Stdout.Write([]byte("\\"))
				os.Stdout.Write([]byte("\n"))
			}
			line = line[olen:]
			off += olen
			remlen -= olen
		}
	}
	fmt.Fprintf(os.Stderr,"--I-- lnsp %d to %d\n",ni,toi)
}

// CRC32 <a href=http://golang.jp/pkg/hash-crc32>crc32</a>
// 1 0000 0100 1100 0001 0001 1101 1011 0111
var CRC32UNIX uint32 = uint32(0x04C11DB7) // Unix cksum
var CRC32IEEE uint32 = uint32(0xEDB88320)
func byteCRC32add(crc uint32,str[]byte,len uint64)(uint32){
	var i uint64
	for i = 0; i < len; i++ {
		var oct = str[i]
		for bi := 0; bi < 8; bi++ {
			ovf1 := (crc & 0x80000000) != 0
			ovf2 := (oct & 0x80) != 0
			ovf := (ovf1 && !ovf2) || (!ovf1 && ovf2)
			oct <<= 1
			crc <<= 1
			if ovf { crc ^= CRC32UNIX }
		}
	}
	return crc;
}
func byteCRC32end(crc uint32, len uint64)(uint32){
	var slen = make([]byte,4)
	var li = 0
		for li = 0; li < 4; {
				slen[li] = byte(len)
		li += 1
				len >>= 8
				if( len == 0 ){
						break
		}
		}
		crc = byteCRC32add(crc,slen,uint64(li))
		crc ^= 0xFFFFFFFF
		return crc
}
func byteCRC32(str[]byte,len uint64)(crc uint32){
	crc = byteCRC32add(0,str,len)
	crc = byteCRC32end(crc,len)
	return crc
}
func CRC32Finish(crc uint32, table *crc32.Table, len uint64)(uint32){
	var slen = make([]byte,4)
	var li = 0
		for li = 0; li < 4; {
				slen[li] = byte(len & 0xFF)
		li += 1
				len >>= 8
				if( len == 0 ){
```

```
                        break
        }
        }
    crc = crc32.Update(crc,table,slen)
        crc ^= 0xFFFFFFFF
        return crc
}

func (gsh*GshContext)xCksum(path string,argv[]string, sum*CheckSum)(int64){
    if isin("-type/f",argv) && !IsRegFile(path){
        return 0
    }
    if isin("-type/d",argv) && IsRegFile(path){
        return 0
    }
    file, err := os.OpenFile(path,os.O_RDONLY,0)
    if err != nil {
        fmt.Printf("--E-- cksum %v (%v)\n",path,err)
        return -1
    }
    defer file.Close()
    if gsh.CmdTrace { fmt.Printf("--I-- cksum %v %v\n",path,argv) }

    bi := 0
    var buff = make([]byte,32*1024)
    var total int64 = 0
    var initTime = time.Time{}
    if sum.Start == initTime {
        sum.Start = time.Now()
    }
    for bi = 0; ; bi++ {
        count,err := file.Read(buff)
        if count <= 0 || err != nil {
            break
        }
        if (sum.SumType & SUM_SUM64) != 0 {
            s := sum.Sum64
            for _,c := range buff[0:count] {
                s += uint64(c)
            }
            sum.Sum64 = s
        }
        if (sum.SumType & SUM_UNIXFILE) != 0 {
            sum.Crc32Val = byteCRC32add(sum.Crc32Val,buff,uint64(count))
        }
        if (sum.SumType & SUM_CRCIEEE) != 0 {
            sum.Crc32Val = crc32.Update(sum.Crc32Val,&sum.Crc32Table,buff[0:count])
        }
        // <a href="https://en.wikipedia.org/wiki/BSD_checksum">BSD checksum</a>
        if (sum.SumType & SUM_SUM16_BSD) != 0 {
            s := sum.Sum16
            for _,c := range buff[0:count] {
                s = (s >> 1) + ((s & 1) << 15)
                s += int(c)
                s &= 0xFFFF
                //fmt.Printf("BSDsum: %d[%d] %d\n",sum.Size+int64(i),i,s)
            }
            sum.Sum16 = s
        }
        if (sum.SumType & SUM_SUM16_SYSV) != 0 {
            for bj := 0; bj < count; bj++ {
                sum.Sum16 += int(buff[bj])
            }
        }
        total += int64(count)
    }
    sum.Done = time.Now()
    sum.Files += 1
    sum.Size += total
    if !isin("-s",argv) {
        fmt.Printf("%v ",total)
    }
    return 0
}

// <a name="grep">grep</a>
// "lines", "lin" or "lnp" for "(text) line processor" or "scanner"
// a*,!ab,c, ... sequentioal combination of patterns
// what "LINE" is should be definable
// generic line-by-line processing
// grep [-v]
// cat -n -v
// uniq [-c]
// tail -f
// sed s/x/y/ or awk
// grep with line count like wc
// rewrite contents if specified
func (gsh*GshContext)xGrep(path string,rexpv[]string)(int){
    file, err := os.OpenFile(path,os.O_RDONLY,0)
    if err != nil {
        fmt.Printf("--E-- grep %v (%v)\n",path,err)
        return -1
    }
    defer file.Close()
    if gsh.CmdTrace { fmt.Printf("--I-- grep %v %v\n",path,rexpv) }
    //reader := bufio.NewReaderSize(file,LINESIZE)
    reader := bufio.NewReaderSize(file,80)
    li := 0
    found := 0
    for li = 0; ; li++ {
        line, err := reader.ReadString('\n')
        if len(line) <= 0 {
            break
        }
        if 150 < len(line) {
            // maybe binary
            break;
        }
        if err != nil {
            break
        }
        if 0 <= strings.Index(string(line),rexpv[0]) {
            found += 1
            fmt.Printf("%s:%d: %s",path,li,line)
        }
    }
        //fmt.Printf("total %d lines %s\n",li,path)
    //if( 0 < found ){ fmt.Printf("((found %d lines %s))\n",found,path); }
    return found
}

// <a name="finder">Finder</a>
// finding files with it name and contents
// file names are ORed
// show the content with %x fmt list
// ls -R
// tar command by adding output
type fileSum struct {
	Err int64   // access error or so
	Size    int64   // content size
	DupSize int64   // content size from hard links
	Blocks  int64   // number of blocks (of 512 bytes)
	DupBlocks int64 // Blocks pointed from hard links
	HLinks  int64   // hard links
	Words   int64
	Lines   int64
	Files   int64
	Dirs    int64   // the num. of directories
	SymLink int64
	Flats   int64   // the num. of flat files
	MaxDepth    int64
	MaxNamlen   int64   // max. name length
	nextRepo    time.Time
}
func showFusage(dir string,fusage *fileSum){
	bsume := float64(((fusage.Blocks-fusage.DupBlocks)/2)*1024)/1000000.0
	//bsumdup := float64((fusage.Blocks/2)*1024)/1000000.0

	fmt.Printf("%v: %v files (%vd %vs %vh) %.6f MB (%.2f MBK)\n",
		dir,
		fusage.Files,
		fusage.Dirs,
		fusage.SymLink,
		fusage.HLinks,
		float64(fusage.Size)/1000000.0,bsume);
}
const (
	S_IFMT    = 0170000
	S_IFCHR   = 0020000
	S_IFDIR   = 0040000
	S_IFREG   = 0100000
	S_IFLNK   = 0120000
	S_IFSOCK  = 0140000
)
func cumFinfo(fsum *fileSum, path string, staterr error, fstat syscall.Stat_t, argv[]string,verb bool)(*fileSum){
	now := time.Now()
	if time.Second <= now.Sub(fsum.nextRepo) {
		if !fsum.nextRepo.IsZero(){
			tstmp := now.Format(time.Stamp)
			showFusage(tstmp,fsum)
		}
		fsum.nextRepo = now.Add(time.Second)
	}
	if staterr != nil {
		fsum.Err += 1
		return fsum
	}
	fsum.Files += 1
	if 1 < fstat.Nlink {
		// must count only once...
		// at least ignore ones in the same directory
		//if finfo.Mode().IsRegular() {
		if (fstat.Mode & S_IFMT) == S_IFREG {
			fsum.HLinks += 1
			fsum.DupBlocks += int64(fstat.Blocks)
			//fmt.Printf("---Dup HardLink %v %s\n",fstat.Nlink,path)
		}
	}
	//fsum.Size += finfo.Size()
	fsum.Size += fstat.Size
	fsum.Blocks += int64(fstat.Blocks)
	//if verb { fmt.Printf("(%8dBlk) %s",fstat.Blocks/2,path) }
	if isin("-ls",argv){
		//if verb { fmt.Printf("%4d %8d ",fstat.Blksize,fstat.Blocks) }
//		fmt.Printf("%d\t",fstat.Blocks/2)
	}
	//if finfo.IsDir()
	if (fstat.Mode & S_IFMT) == S_IFDIR {
		fsum.Dirs += 1
	}
	//if (finfo.Mode() & os.ModeSymlink) != 0
	if (fstat.Mode & S_IFMT) == S_IFLNK {
		//if verb { fmt.Printf("symlink(%v,%s)\n",fstat.Mode,finfo.Name()) }
		//{ fmt.Printf("symlink(%o,%s)\n",fstat.Mode,finfo.Name()) }
		fsum.SymLink += 1
	}
	return fsum
}
func (gsh*GshContext)xxFindEntv(depth int,total *fileSum,dir string, dstat syscall.Stat_t, ei int, entv []string,npatv[]string,argv[]string)(*fileSum){
	nols := isin("-grep",argv)
	// sort entv
	/*
	if isin("-t",argv){
		sort.Slice(filev, func(i,j int) bool {
			return 0 < filev[i].ModTime().Sub(filev[j].ModTime())
		})
	}
	*/
		/*
		if isin("-u",argv){
			sort.Slice(filev, func(i,j int) bool {
				return 0 < filev[i].AccTime().Sub(filev[j].AccTime())
			})
		}
		if isin("-U",argv){
			sort.Slice(filev, func(i,j int) bool {
				return 0 < filev[i].CreatTime().Sub(filev[j].CreatTime())
			})
		}
		*/
	/*
	if isin("-S",argv){
		sort.Slice(filev, func(i,j int) bool {
			return filev[j].Size() < filev[i].Size()
		})
	}
	*/
	for _,filename := range entv {
		for _,npat := range npatv {
			match := true
			if npat == "*" {
				match = true
			}else{
				match, _ = filepath.Match(npat,filename)
			}
			path := dir + DIRSEP + filename
			if !match {
				continue
			}
			var fstat syscall.Stat_t
```

```
            staterr := syscall.Lstat(path,&fstat)
            if staterr != nil {
                if !isin("-w",argv){fmt.Printf("ufind: %v\n",staterr) }
                continue;
            }
            if isin("-du",argv) && (fstat.Mode & S_IFMT) == S_IFDIR {
                // should not show size of directory in "-du" mode ...
            }else
            if !nols && !isin("-s",argv) && (!isin("-du",argv) || isin("-a",argv)) {
                if isin("-du",argv) {
                    fmt.Printf("%d\t",fstat.Blocks/2)
                }
                showFileInfo(path,argv)
            }
            if true { // && isin("-du",argv)
                total = cumFinfo(total,path,staterr,fstat,argv,false)
            }
            /*
            if isin("-wc",argv) {
            }
            */
            if gsh.lastCheckSum.SumType != 0 {
                gsh.xCksum(path,argv,&gsh.lastCheckSum);
            }
            x := isinX("-grep",argv); // -grep will be convenient like -ls
            if 0 <= x && x+1 <= len(argv) { // -grep will be convenient like -ls
                if IsRegFile(path){
                    found := gsh.xGrep(path,argv[x+1:])
                    if 0 < found {
                        foundv := gsh.CmdCurrent.FoundFile
                        if len(foundv) < 10 {
                            gsh.CmdCurrent.FoundFile =
                            append(gsh.CmdCurrent.FoundFile,path)
                        }
                    }
                }
            }
            if !isin("-r0",argv) { // -d 0 in du, -depth n in find
                //total.Depth += 1
                if (fstat.Mode & S_IFMT) == S_IFLNK {
                    continue
                }
                if dstat.Rdev != fstat.Rdev {
                    fmt.Printf("--I-- don't follow differnet device %v(%v) %v(%v)\n",
                        dir,dstat.Rdev,path,fstat.Rdev)
                }
                if (fstat.Mode & S_IFMT) == S_IFDIR {
                    total = gsh.xxFind(depth+1,total,path,npatv,argv)
                }
            }
        }
    }
    return total
}
func (gsh*GshContext)xxFind(depth int,total *fileSum,dir string,npatv[]string,argv[]string)(*fileSum){
    nols := isin("-grep",argv)
    dirfile,oerr := os.OpenFile(dir,os.O_RDONLY,0)
    if oerr == nil {
        //fmt.Printf("--I-- %v(%v)[%d]\n",dir,dirfile,dirfile.Fd())
        defer dirfile.Close()
    }else{
    }

    prev := *total
    var dstat syscall.Stat_t
    staterr := syscall.Lstat(dir,&dstat) // should be flstat

    if staterr != nil {
        if !isin("-w",argv){ fmt.Printf("ufind: %v\n",staterr) }
        return total
    }
        //filev,err := ioutil.ReadDir(dir)
        //_,err := ioutil.ReadDir(dir) // ReadDir() heavy and bad for huge directory
        /*
        if err != nil {
            if !isin("-w",argv){ fmt.Printf("ufind: %v\n",err) }
            return total
        }
        */
    if depth == 0 {
        total = cumFinfo(total,dir,staterr,dstat,argv,true)
        if !nols && !isin("-s",argv) && (!isin("-du",argv) || isin("-a",argv)) {
            showFileInfo(dir,argv)
        }
    }
    // it it is not a directory, just scan it and finish

    for ei := 0; ; ei++ {
        entv,rderr := dirfile.Readdirnames(8*1024)
        if len(entv) == 0 || rderr != nil {
            //if rderr != nil { fmt.Printf("[%d] len=%d (%v)\n",ei,len(entv),rderr) }
            break
        }
        if 0 < ei {
            fmt.Printf("--I-- xxFind[%d] %d large-dir: %s\n",ei,len(entv),dir)
        }
        total = gsh.xxFindEntv(depth,total,dir,dstat,ei,entv,npatv,argv)
    }
    if isin("-du",argv) {
        // if in "du" mode
        fmt.Printf("%d\t%s\n",(total.Blocks-prev.Blocks)/2,dir)
    }
    return total
}

// {ufind|fu|ls} [Files] [// Names] [-- Expressions]
//  Files is "." by default
//  Names is "*" by default
//  Expressions is "-print" by default for "ufind", or -du for "fu" command
func (gsh*GshContext)xFind(argv[]string){
    if 0 < len(argv) && strBegins(argv[0],"?"){
        showFound(gsh,argv)
        return
    }
    if isin("-cksum",argv) || isin("-sum",argv) {
        gsh.lastCheckSum = CheckSum{}
        if isin("-sum",argv) && isin("-add",argv) {
            gsh.lastCheckSum.SumType |= SUM_SUM64
        }else
        if isin("-sum",argv) && isin("-size",argv) {
            gsh.lastCheckSum.SumType |= SUM_SIZE
        }else
        if isin("-sum",argv) && isin("-bsd",argv) {
            gsh.lastCheckSum.SumType |= SUM_SUM16_BSD
        }else
        if isin("-sum",argv) && isin("-sysv",argv) {
            gsh.lastCheckSum.SumType |= SUM_SUM16_SYSV
        }else
        if isin("-sum",argv) {
            gsh.lastCheckSum.SumType |= SUM_SUM64
        }
        if isin("-unix",argv) {
            gsh.lastCheckSum.SumType |= SUM_UNIXFILE
            gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32UNIX)
        }
        if isin("-ieee",argv){
            gsh.lastCheckSum.SumType |= SUM_CRCIEEE
            gsh.lastCheckSum.Crc32Table = *crc32.MakeTable(CRC32IEEE)
        }
        gsh.lastCheckSum.RusgAtStart = Getrusagev()
    }
    var total = fileSum{}
    npats := []string{}
    for _,v := range argv {
        if 0 < len(v) && v[0] != '-' {
            npats = append(npats,v)
        }
        if v == "//" { break }
        if v == "--" { break }
        if v == "-grep" { break }
        if v == "-ls" { break }
    }
    if len(npats) == 0 {
        npats = []string{"*"}
    }
    cwd := "."
    // if to be fullpath ::: cwd, _ := os.Getwd()
    if len(npats) == 0 { npats = []string{"*"} }
    fusage := gsh.xxFind(0,&total,cwd,npats,argv)
    if gsh.lastCheckSum.SumType != 0 {
        var sumi uint64 = 0
        sum := &gsh.lastCheckSum
        if (sum.SumType & SUM_SIZE) != 0 {
            sumi = uint64(sum.Size)
        }
        if (sum.SumType & SUM_SUM64) != 0 {
            sumi = sum.Sum64
        }
        if (sum.SumType & SUM_SUM16_SYSV) != 0 {
            s := uint32(sum.Sum16)
            r := (s & 0xFFFF) + ((s & 0xFFFFFFFF) >> 16)
            s = (r & 0xFFFF) + (r >> 16)
            sum.Crc32Val = uint32(s)
            sumi = uint64(s)
        }
        if (sum.SumType & SUM_SUM16_BSD) != 0 {
            sum.Crc32Val = uint32(sum.Sum16)
            sumi = uint64(sum.Sum16)
        }
        if (sum.SumType & SUM_UNIXFILE) != 0 {
            sum.Crc32Val = byteCRC32end(sum.Crc32Val,uint64(sum.Size))
            sumi = uint64(byteCRC32end(sum.Crc32Val,uint64(sum.Size)))
        }
        if 1 < sum.Files {
            fmt.Printf("%v %v // %v / %v files, %v/file\r\n",
                sumi,sum.Size,
                abssize(sum.Size),sum.Files,
                abssize(sum.Size/sum.Files))
        }else{
            fmt.Printf("%v %v %v\n",
                sumi,sum.Size,npats[0])
        }
    }
    if !isin("-grep",argv) {
        showFusage("total",fusage)
    }
    if !isin("-s",argv){
        hits := len(gsh.CmdCurrent.FoundFile)
        if 0 < hits {
            fmt.Printf("--I-- %d files hits // can be refered with !%df\n",
                hits,len(gsh.CommandHistory))
        }
    }
    if gsh.lastCheckSum.SumType != 0 {
        if isin("-ru",argv) {
            sum := &gsh.lastCheckSum
            sum.Done = time.Now()
            gsh.lastCheckSum.RusgAtEnd = Getrusagev()
            elps := sum.Done.Sub(sum.Start)
            fmt.Printf("--cksum-size: %v (%v) / %v files, %v/file\r\n",
                sum.Size,abssize(sum.Size),sum.Files,abssize(sum.Size/sum.Files))
            nanos := int64(elps)
            fmt.Printf("--cksum-time: %v/total, %v/file, %.1f files/s, %v\r\n",
                abbtime(nanos),
                abbtime(nanos/sum.Files),
                (float64(sum.Files)*1000000000.0)/float64(nanos),
                abbspeed(sum.Size,nanos))
            diff := RusageSubv(sum.RusgAtEnd,sum.RusgAtStart)
            fmt.Printf("--cksum-rusg: %v\n",sRusagef("",argv,diff))
        }
    }
    return
}

func showFiles(files[]string){
    sp := ""
    for i,file := range files {
        if 0 < i { sp = " " } else { sp = "" }
        fmt.Printf(sp+"%s",escapeWhiteSP(file))
    }
}
func showFound(gshCtx *GshContext, argv[]string){
    for i,v := range gshCtx.CommandHistory {
        if 0 < len(v.FoundFile) {
            fmt.Printf("!%d (%d) ",i,len(v.FoundFile))
            if isin("-ls",argv){
                fmt.Printf("\n")
                for _,file := range v.FoundFile {
                    fmt.Printf("") //sub number?
                    showFileInfo(file,argv)
                }
            }else{
                showFiles(v.FoundFile)
                fmt.Printf("\n")
            }
        }
    }
}

func showMatchFile(filev []os.FileInfo, npat,dir string, argv[]string)(string,bool){
	fname := ""
	found := false
	for _,v := range filev {
		match, _ := filepath.Match(npat,(v.Name()))
		if match {
			fname = v.Name()
			found = true
			//fmt.Printf("[%d] %s\n",i,v.Name())
			showIfExecutable(fname,dir,argv)
		}
	}
	return fname,found
}
func showIfExecutable(name,dir string,argv[]string)(ffullpath string,ffound bool){
	var fullpath string
	if strBegins(name,DIRSEP){
		fullpath = name
	}else{
		fullpath = dir + DIRSEP + name
	}
	fi, err := os.Stat(fullpath)
	if err != nil {
		fullpath = dir + DIRSEP + name + ".go"
		fi, err = os.Stat(fullpath)
	}
	if err == nil {
		fm := fi.Mode()
		if fm.IsRegular() {
		  // R_OK=4, W_OK=2, X_OK=1, F_OK=0
		  if syscall.Access(fullpath,5) == nil {
			ffullpath = fullpath
			ffound = true
			if ! isin("-s", argv) {
				showFileInfo(fullpath,argv)
			}
		  }
		}
	}
	return ffullpath, ffound
}
func which(list string, argv []string) (fullpathv []string, itis bool){
	if len(argv) <= 1 {
		fmt.Printf("Usage: which comand [-s] [-a] [-ls]\n")
		return []string{""}, false
	}
	path := argv[1]
	if strBegins(path,"/") {
		// should check if excecutable?
		_,exOK := showIfExecutable(path,"/",argv)
		fmt.Printf("--D-- %v exOK=%v\n",path,exOK)
		return []string{path},exOK
	}
	pathenv, efound := os.LookupEnv(list)
	if ! efound {
		fmt.Printf("--E-- which: no \"%s\" environment\n",list)
		return []string{""}, false
	}
	showall := isin("-a",argv) || 0 <= strings.Index(path,"*")
	dirv := strings.Split(pathenv,PATHSEP)
	ffound := false
	ffullpath := path
	for _, dir := range dirv {
		if 0 <= strings.Index(path,"*") { // by wild-card
			list,_ := ioutil.ReadDir(dir)
			ffullpath, ffound = showMatchFile(list,path,dir,argv)
		}else{
			ffullpath, ffound = showIfExecutable(path,dir,argv)
		}
		//if ffound && !isin("-a", argv) {
		if ffound && !showall {
			break;
		}
	}
	return []string{ffullpath}, ffound
}

func stripLeadingWSParg(argv[]string)([]string){
	for ; 0 < len(argv); {
		if len(argv[0]) == 0 {
			argv = argv[1:]
		}else{
			break
		}
	}
	return argv
}
func xEval(argv []string, nlend bool){
	argv = stripLeadingWSParg(argv)
	if len(argv) == 0 {
		fmt.Printf("eval [%%format] [Go-expression]\n")
		return
	}
	pfmt := "%v"
	if argv[0][0] == '%' {
		pfmt = argv[0]
		argv = argv[1:]
	}
	if len(argv) == 0 {
		return
	}
	gocode := strings.Join(argv," ");
	//fmt.Printf("eval [%v] [%v]\n",pfmt,gocode)
	fset := token.NewFileSet()
	rval, _ := types.Eval(fset,nil,token.NoPos,gocode)
	fmt.Printf(pfmt,rval.Value)
	if nlend { fmt.Printf("\n") }
}

func getval(name string) (found bool, val int) {
	/* should expand the name here */
	if name == "gsh.pid" {
		return true, os.Getpid()
	}else
	if name == "gsh.ppid" {
		return true, os.Getppid()
	}
	return false, 0
}

func echo(argv []string, nlend bool){
	for ai := 1; ai < len(argv); ai++ {
		if 1 < ai {
			fmt.Printf(" ");
		}
		arg := argv[ai]
```

```
        found, val := getval(arg)
        if found {
            fmt.Printf("%d",val)
        }else{
            fmt.Printf("%s",arg)
        }
    }
    if nlend {
        fmt.Printf("\n");
    }
}

func resfile() string {
    return "gsh.tmp"
}
//var resF *File
func resmap() {
    //_ , err := os.OpenFile(resfile(), os.O_RDWR|os.O_CREATE, os.ModeAppend)
    // https://developpaper.com/solution-to-golang-bad-file-descriptor-problem/
    _ , err := os.OpenFile(resfile(), os.O_RDWR|os.O_CREATE, 0600)
    if err != nil {
        fmt.Printf("refF could not open: %s\n",err)
    }else{
        fmt.Printf("refF opened\n")
    }
}

// @@2020-0821
func gshScanArg(str string,strip int)(argv []string){
    var si = 0
    var sb = 0
    var inBracket = 0
    var arg1 = make([]byte,LINESIZE)
    var ax = 0
    debug := false

    for ; si < len(str); si++ {
        if str[si] != ' ' {
            break
        }
    }
    sb = si
    for ; si < len(str); si++ {
        if sb <= si {
            if debug {
                fmt.Printf("--Da- +%d %2d-%2d %s ... %s\n",
                    inBracket,sb,si,arg1[0:ax],str[si:])
            }
        }
        ch := str[si]
        if ch  == '{' {
            inBracket += 1
            if 0 < strip && inBracket <= strip {
                //fmt.Printf("stripLEV %d <= %d?\n",inBracket,strip)
                continue
            }
        }
        if 0 < inBracket {
            if ch == '}' {
                inBracket -= 1
                if 0 < strip && inBracket < strip {
                    //fmt.Printf("stripLEV %d <  %d?\n",inBracket,strip)
                    continue
                }
            }
            arg1[ax] = ch
            ax += 1
            continue
        }
        if str[si] == ' ' {
            argv = append(argv,string(arg1[0:ax]))
            if debug {
                fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
                    -1+len(argv),sb,si,str[sb:si],string(str[si:]))
            }
            sb = si+1
            ax = 0
            continue
        }
        arg1[ax] = ch
        ax += 1
    }
    if sb < si {
        argv = append(argv,string(arg1[0:ax]))
        if debug {
            fmt.Printf("--Da- [%v][%v-%v] %s ... %s\n",
                -1+len(argv),sb,si,string(arg1[0:ax]),string(str[si:]))
        }
    }
    if debug {
        fmt.Printf("--Da- %d [%s] => [%d]%v\n",strip,str,len(argv),argv)
    }
    return argv
}

// should get stderr (into tmpfile ?) and return
func (gsh*GshContext)Popen(name,mode string)(pin*os.File,pout*os.File,err bool){
    var pv = []int{-1,-1}
    syscall.Pipe(pv)

    xarg := gshScanArg(name,1)
    name = strings.Join(xarg," ")

    pin = os.NewFile(uintptr(pv[0]),"StdoutOf-{"+name+"}")
    pout = os.NewFile(uintptr(pv[1]),"StdinOf-{"+name+"}")
    fdix := 0
    dir := "?"
    if mode == "r" {
        dir = "<"
        fdix = 1 // read from the stdout of the process
    }else{
        dir = ">"
        fdix = 0 // write to the stdin of the process
    }
    gshPA := gsh.gshPA
    savfd := gshPA.Files[fdix]

    var fd uintptr = 0
    if mode == "r" {
        fd = pout.Fd()
        gshPA.Files[fdix] = pout.Fd()
    }else{
        fd = pin.Fd()
        gshPA.Files[fdix] = pin.Fd()
    }
```

```
		// should do this by Goroutine?
		if false {
			fmt.Printf("--Ip- Opened fd[%v] %s %v\n",fd,dir,name)
			fmt.Printf("--RED1 [%d,%d,%d]->[%d,%d,%d]\n",
				os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd(),
				pin.Fd(),pout.Fd(),pout.Fd())
		}
			savi := os.Stdin
			savo := os.Stdout
			save := os.Stderr
			os.Stdin  = pin
			os.Stdout = pout
			os.Stderr = pout
		gsh.BackGround = true
		gsh.gshelllh(name)
		gsh.BackGround = false
			os.Stdin  = savi
			os.Stdout = savo
			os.Stderr = save

	gshPA.Files[fdix] = savfd
	return pin,pout,false
}

// <a name="ex-commands">External commands</a>
func (gsh*GshContext)excommand(exec bool, argv []string) (notf bool,exit bool) {
	if gsh.CmdTrace { fmt.Printf("--I-- excommand[%v](%v)\n",exec,argv) }

	gshPA := gsh.gshPA
	fullpathv, itis := which("PATH",[]string{"which",argv[0],"-s"})
	if itis == false {
		return true,false
	}
	fullpath := fullpathv[0]
	argv = unescapeWhiteSPV(argv)
	if 0 < strings.Index(fullpath,".go") {
		nargv := argv // []string{}
		gofullpathv, itis := which("PATH",[]string{"which","go","-s"})
		if itis == false {
			fmt.Printf("--F-- Go not found\n")
			return false,true
		}
		gofullpath := gofullpathv[0]
		nargv = []string{ gofullpath, "run", fullpath }
		fmt.Printf("--I-- %s {%s %s %s}\n",gofullpath,
			nargv[0],nargv[1],nargv[2])
		if exec {
			syscall.Exec(gofullpath,nargv,os.Environ())
		}else{
			pid, _ := syscall.ForkExec(gofullpath,nargv,&gshPA)
			if gsh.BackGround {
				fmt.Fprintf(stderr,"--Ip- in Background pid[%d]%d(%v)\n",pid,len(argv),nargv)
				gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
			}else{
				rusage := syscall.Rusage {}
				syscall.Wait4(pid,nil,0,&rusage)
				gsh.LastRusage = rusage
				gsh.CmdCurrent.Rusagev[1] = rusage
			}
		}
	}else{
		if exec {
			syscall.Exec(fullpath,argv,os.Environ())
		}else{
			pid, _ := syscall.ForkExec(fullpath,argv,&gshPA)
			//fmt.Printf("[%d]\n",pid); // '&' to be background
			if gsh.BackGround {
				fmt.Fprintf(stderr,"--Ip- in Background pid[%d]%d(%v)\n",pid,len(argv),argv)
				gsh.BackGroundJobs = append(gsh.BackGroundJobs,pid)
			}else{
				rusage := syscall.Rusage {}
				syscall.Wait4(pid,nil,0,&rusage);
				gsh.LastRusage = rusage
				gsh.CmdCurrent.Rusagev[1] = rusage
			}
		}
	}
	return false,false
}

// <a name="builtin">Builtin Commands</a>
func (gshCtx *GshContext) sleep(argv []string) {
	if len(argv) < 2 {
		fmt.Printf("Sleep 100ms, 100us, 100ns, ...\n")
		return
	}
	duration := argv[1];
	d, err := time.ParseDuration(duration)
	if err != nil {
		d, err = time.ParseDuration(duration+"s")
		if err != nil {
			fmt.Printf("duration ? %s (%s)\n",duration,err)
			return
		}
	}
	//fmt.Printf("Sleep %v\n",duration)
	time.Sleep(d)
	if 0 < len(argv[2:]) {
		gshCtx.gshellv(argv[2:])
	}
}
func (gshCtx *GshContext)repeat(argv []string) {
	if len(argv) < 2 {
		return
	}
	start0 := time.Now()
	for ri,_ := strconv.Atoi(argv[1]); 0 < ri; ri-- {
		if 0 < len(argv[2:]) {
			//start := time.Now()
			gshCtx.gshellv(argv[2:])
			end := time.Now()
			elps := end.Sub(start0);
			if( 1000000000 < elps ){
				fmt.Printf("(repeat#%d %v)\n",ri,elps);
			}
		}
	}
}

func (gshCtx *GshContext)gen(argv []string) {
	gshPA := gshCtx.gshPA
	if len(argv) < 2 {
		fmt.Printf("Usage: %s N\n",argv[0])
		return
	}
	// should br repeated by "repeat" command
	count, _ := strconv.Atoi(argv[1])
	fd := gshPA.Files[1] // Stdout
	file := os.NewFile(fd,"internalStdOut")
	fmt.Printf("--I-- Gen. Count=%d to [%d]\n",count,file.Fd())
	//buf := []byte{}
	outdata := "0123 5678 0123 5678 0123 5678 0123 5678\r"
	for gi := 0; gi < count; gi++ {
		file.WriteString(outdata)
	}
	//file.WriteString("\n")
	fmt.Printf("\n(%d B)\n",count*len(outdata));
	//file.Close()
}

// <a name="rexec">Remote Execution</a> // 2020-0820
func Elapsed(from time.Time)(string){
	elps := time.Now().Sub(from)
	if 1000000000 < elps {
		return fmt.Sprintf("[%5d.%02ds]",elps/1000000000,(elps%1000000000)/10000000)
	}else
	if 1000000 < elps {
		return fmt.Sprintf("[%3d.%03dms]",elps/1000000,(elps%1000000)/1000)
	}else{
		return fmt.Sprintf("[%3d.%03dus]",elps/1000,(elps%1000))
	}
}
func abbtime(nanos int64)(string){
	if 1000000000 < nanos {
		return fmt.Sprintf("%d.%02ds",nanos/1000000000,(nanos%1000000000)/10000000)
	}else
	if 1000000 < nanos {
		return fmt.Sprintf("%d.%03dms",nanos/1000000,(nanos%1000000)/1000)
	}else{
		return fmt.Sprintf("%d.%03dus",nanos/1000,(nanos%1000))
	}
}
func abssize(size int64)(string){
	fsize := float64(size)
	if 1024*1024*1024 < size {
		return fmt.Sprintf("%.2fGiB",fsize/(1024*1024*1024))
	}else
	if 1024*1024 < size {
		return fmt.Sprintf("%.3fMiB",fsize/(1024*1024))
	}else{
		return fmt.Sprintf("%.3fKiB",fsize/1024)
	}
}
func absize(size int64)(string){
	fsize := float64(size)
	if 1024*1024*1024 < size {
		return fmt.Sprintf("%8.2fGiB",fsize/(1024*1024*1024))
	}else
	if 1024*1024 < size {
		return fmt.Sprintf("%8.3fMiB",fsize/(1024*1024))
	}else{
		return fmt.Sprintf("%8.3fKiB",fsize/1024)
	}
}
func abbspeed(totalB int64,ns int64)(string){
	MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
	if 1000 <= MBs {
		return fmt.Sprintf("%6.3fGB/s",MBs/1000)
	}
	if 1 <= MBs {
		return fmt.Sprintf("%6.3fMB/s",MBs)
	}else{
		return fmt.Sprintf("%6.3fKB/s",MBs*1000)
	}
}
func abspeed(totalB int64,ns time.Duration)(string){
	MBs := (float64(totalB)/1000000) / (float64(ns)/1000000000)
	if 1000 <= MBs {
		return fmt.Sprintf("%6.3fGBps",MBs/1000)
	}
	if 1 <= MBs {
		return fmt.Sprintf("%6.3fMBps",MBs)
	}else{
		return fmt.Sprintf("%6.3fKBps",MBs*1000)
	}
}
func fileRelay(what string,in*os.File,out*os.File,size int64,bsiz int)(wcount int64){
	Start := time.Now()
	buff := make([]byte,bsiz)
	var total int64 = 0
	var rem int64 = size
	nio := 0
	Prev := time.Now()
	var PrevSize int64 = 0

	fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) START\n",
		what,absize(total),size,nio)

	for i:= 0; ; i++ {
		var len = bsiz
		if int(rem) < len {
			len = int(rem)
		}
		Now := time.Now()
		Elps := Now.Sub(Prev);
		if 1000000000 < Now.Sub(Prev) {
			fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) %s\n",
				what,absize(total),size,nio,
				abspeed((total-PrevSize),Elps))
			Prev = Now;
			PrevSize = total
		}
		rlen := len
		if in != nil {
			// should watch the disconnection of out
			rcc,err := in.Read(buff[0:rlen])
			if err != nil {
				fmt.Printf(Elapsed(Start)+"--En- X: %s read(%v,%v)<%v\n",
					what,rcc,err,in.Name())
				break
			}
			rlen = rcc
			if string(buff[0:10]) == "((SoftEOF " {
				var ecc int64 = 0
				fmt.Sscanf(string(buff),"((SoftEOF %v",&ecc)
				fmt.Printf(Elapsed(Start)+"--En- X: %s Recv ((SoftEOF %v))/%v\n",
					what,ecc,total)
				if ecc == total {
					break
				}
			}
		}

		wlen := rlen
		if out != nil {
			wcc,err := out.Write(buff[0:rlen])
			if err != nil {
				fmt.Printf(Elapsed(Start)+"-En-- X: %s write(%v,%v)>%v\n",
					what,wcc,err,out.Name())
				break
			}
			wlen = wcc
		}
		if wlen < rlen {
			fmt.Printf(Elapsed(Start)+"--En- X: %s incomplete write (%v/%v)\n",
				what,wlen,rlen)
			break;
		}

		nio += 1
		total += int64(rlen)
		rem -= int64(rlen)
		if rem <= 0 {
			break
		}
	}
	Done := time.Now()
	Elps := float64(Done.Sub(Start))/1000000000 //Seconds
	TotalMB := float64(total)/1000000 //MB
	MBps := TotalMB / Elps
	fmt.Printf(Elapsed(Start)+"--In- X: %s (%v/%v/%v) %v %.3fMB/s\n",
		what,total,size,nio,absize(total),MBps)
	return total
}
func tcpPush(clnt *os.File){
	// shrink socket buffer and recover
	usleep(100);
}
func (gsh*GshContext)RexecServer(argv[]string){
	debug := true
	Start0 := time.Now()
	Start := Start0
//	if local == ":" { local = "0.0.0.0:9999" }
	local := "0.0.0.0:9999"

	if 0 < len(argv) {
		if argv[0] == "-s" {
			debug = false
			argv = argv[1:]
		}
	}
	if 0 < len(argv) {
		argv = argv[1:]
	}
	port, err := net.ResolveTCPAddr("tcp",local);
	if err != nil {
		fmt.Printf("--En- S: Address error: %s (%s)\n",local,err)
		return
	}
	fmt.Printf(Elapsed(Start)+"--In- S: Listening at %s...\n",local);
	sconn, err := net.ListenTCP("tcp", port)
	if err != nil {
		fmt.Printf(Elapsed(Start)+"--En- S: Listen error: %s (%s)\n",local,err)
		return
	}

	reqbuf := make([]byte,LINESIZE)
	res := ""
	for {
		fmt.Printf(Elapsed(Start0)+"--In- S: Listening at %s...\n",local);
		aconn, err := sconn.AcceptTCP()
		Start = time.Now()
		if err != nil {
			fmt.Printf(Elapsed(Start)+"--En- S: Accept error: %s (%s)\n",local,err)
			return
		}
		clnt, _ := aconn.File()
		fd := clnt.Fd()
		ar := aconn.RemoteAddr()
		if debug { fmt.Printf(Elapsed(Start0)+"--In- S: Accepted TCP at %s [%d] <- %v\n",
			local,fd,ar) }
		res = fmt.Sprintf("220 GShell/%s Server\r\n",VERSION)
		fmt.Fprintf(clnt,"%s",res)
		if debug { fmt.Printf(Elapsed(Start)+"--In- S: %s",res) }
		count, err := clnt.Read(reqbuf)
		if err != nil {
			fmt.Printf(Elapsed(Start)+"--En- C: (%v %v) %v",
				count,err,string(reqbuf))
		}
		req := string(reqbuf[:count])
		if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",string(req)) }
		reqv := strings.Split(string(req),"\r")
		cmdv := gshScanArg(reqv[0],0)
		//cmdv := strings.Split(reqv[0]," ")
		switch cmdv[0] {
			case "HELO":
				res = fmt.Sprintf("250 %v",req)
			case "GET":
				// download {remotefile|-zN} [localfile]
				var dsize int64 = 32*1024*1024
				var bsize int = 64*1024
				var fname string = ""
				var in *os.File = nil
				var pseudoEOF = false
				if 1 < len(cmdv) {
					fname = cmdv[1]
					if strBegins(fname,"-z") {
						fmt.Sscanf(fname[2:],"%d",&dsize)
					}else
					if strBegins(fname,"{") {
						xin,xout,err := gsh.Popen(fname,"r")
						if err {
						}else{
							xout.Close()
							defer xin.Close()
							in = xin
							dsize = MaxStreamSize
							pseudoEOF = true
						}
					}else{
						xin,err := os.Open(fname)
						if err != nil {
							fmt.Printf("--En- GET (%v)\n",err)
						}else{
							defer xin.Close()
```

```
                                in = xin
                                fi,_ := xin.Stat()
                                dsize = fi.Size()
                            }
                        }
                    }
                    //fmt.Printf(Elapsed(Start)+"--In- GET %v:%v\n",dsize,bsize)
                    res = fmt.Sprintf("200 %v\r\n",dsize)
                    fmt.Fprintf(clnt,"%v",res)
                    tcpPush(clnt); // should be separated as line in receiver
                    fmt.Printf(Elapsed(Start)+"--In- S: %v",res)
                    wcount := fileRelay("SendGET",in,clnt,dsize,bsize)
                    if pseudoEOF {
                        in.Close() // pipe from the command
                        // show end of stream data (its size) by OOB?
                        SoftEOF := fmt.Sprintf("((SoftEOF %v))",wcount)
                        fmt.Printf(Elapsed(Start)+"--In- S: Send %v\n",SoftEOF)

                        tcpPush(clnt); // to let SoftEOF data apper at the top of recevied data
                        fmt.Fprintf(clnt,"%v\r\n",SoftEOF)
                        tcpPush(clnt); // to let SoftEOF alone in a packet (separate with 200 OK)
                            // with client generated random?
                        //fmt.Printf("--In- L: close %v (%v)\n",in.Fd(),in.Name())
                    }
                    res = fmt.Sprintf("200 GET done\r\n")
                case "PUT":
                    // upload {srcfile|-zN} [dstfile]
                    var dsize int64 = 32*1024*1024
                    var bsize int = 64*1024
                    var fname string = ""
                    var out *os.File = nil
                    if 1 < len(cmdv) { // localfile
                        fmt.Sscanf(cmdv[1],"%d",&dsize)
                    }
                    if 2 < len(cmdv) {
                        fname = cmdv[2]
                        if fname == "-" {
                            // nul dev
                        }else
                        if strBegins(fname,"{") {
                            xin,xout,err := gsh.Popen(fname,"w")
                            if err {
                            }else{
                                xin.Close()
                                defer xout.Close()
                                out = xout
                            }
                        }else{
                        // should write to temporary file
                        // should suppress ^C on tty
                xout,err := os.OpenFile(fname,os.O_CREATE|os.O_RDWR|os.O_TRUNC,0600)
                //fmt.Printf("--In- S: open(%v) out(%v) err(%v)\n",fname,xout,err)
                            if err != nil {
                                fmt.Printf("--En- PUT (%v)\n",err)
                            }else{
                                out = xout
                            }
                        }
                    fmt.Printf(Elapsed(Start)+"--In- L: open(%v,w) %v (%v)\n",
                        fname,local,err)
                    }
                    fmt.Printf(Elapsed(Start)+"--In- PUT %v (/%v)\n",dsize,bsize)
                    fmt.Printf(Elapsed(Start)+"--In- S: 200 %v OK\r\n",dsize)
                    fmt.Fprintf(clnt,"200 %v OK\r\n",dsize)
                    fileRelay("RecvPUT",clnt,out,dsize,bsize)
                    res = fmt.Sprintf("200 PUT done\r\n")
                default:
                    res = fmt.Sprintf("400 What? %v",req)
            }
            swcc,serr := clnt.Write([]byte(res))
            if serr != nil {
                fmt.Printf(Elapsed(Start)+"--In- S: (wc=%v er=%v) %v",swcc,serr,res)
            }else{
                fmt.Printf(Elapsed(Start)+"--In- S: %v",res)
            }
            aconn.Close();
            clnt.Close();
        }
        sconn.Close();
}
func (gsh*GshContext)RexecClient(argv[]string)(int,string){
        debug := true
        Start := time.Now()
        if len(argv) == 1 {
            return -1,"EmptyARG"
        }
        argv = argv[1:]
        if argv[0] == "-serv" {
            gsh.RexecServer(argv[1:])
            return 0,"Server"
        }
        remote := "0.0.0.0:9999"
        if argv[0][0] == '@' {
            remote = argv[0][1:]
            argv = argv[1:]
        }
        if argv[0] == "-s" {
            debug = false
            argv = argv[1:]
        }
        dport, err := net.ResolveTCPAddr("tcp",remote);
        if err != nil {
            fmt.Printf(Elapsed(Start)+"Address error: %s (%s)\n",remote,err)
            return -1,"AddressError"
        }
        fmt.Printf(Elapsed(Start)+"--In- C: Connecting to %s\n",remote)
        serv, err := net.DialTCP("tcp",nil,dport)
        if err != nil {
            fmt.Printf(Elapsed(Start)+"Connection error: %s (%s)\n",remote,err)
            return -1,"CannotConnect"
        }
        if debug {
            al := serv.LocalAddr()
            fmt.Printf(Elapsed(Start)+"--In- C: Connected to %v <- %v\n",remote,al)
        }

        req := ""
        res := make([]byte,LINESIZE)
        count,err := serv.Read(res)
        if err != nil {
            fmt.Printf("--En- S: (%3d,%v) %v",count,err,string(res))
        }
        if debug { fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res)) }

        if argv[0] == "GET" {
```

```
		savPA := gsh.gshPA
		var bsize int = 64*1024
		req = fmt.Sprintf("%v\r\n",strings.Join(argv," "))
		fmt.Printf(Elapsed(Start)+"--In- C: %v",req)
		fmt.Fprintf(serv,req)
		count,err = serv.Read(res)
		if err != nil {
		}else{
			var dsize int64 = 0
			var out *os.File = nil
			var out_tobeclosed *os.File = nil
			var fname string = ""
			var rcode int = 0
			var pid int = -1
			fmt.Sscanf(string(res),"%d %d",&rcode,&dsize)
			fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res[0:count]))
			if 3 <= len(argv) {
				fname = argv[2]
				if strBegins(fname,"{") {
					xin,xout,err := gsh.Popen(fname,"w")
					if err {
					}else{
						xin.Close()
						defer xout.Close()
						out = xout
						out_tobeclosed = xout
						pid = 0 // should be its pid
					}
				}else{
					// should write to temporary file
					// should suppress ^C on tty
					xout,err := os.OpenFile(fname,os.O_CREATE|os.O_RDWR|os.O_TRUNC,0600)
					if err != nil {
						fmt.Print("--En- %v\n",err)
					}
					out = xout
					//fmt.Printf("--In-- %d > %s\n",out.Fd(),fname)
				}
			}
			in,_ := serv.File()
			fileRelay("RecvGET",in,out,dsize,bsize)
			if 0 <= pid {
				gsh.gshPA = savPA // recovery of Fd(), and more?
				fmt.Printf(Elapsed(Start)+"--In- L: close Pipe > %v\n",fname)
				out_tobeclosed.Close()
				//syscall.Wait4(pid,nil,0,nil) //@@
			}
		}
	}else
	if argv[0] == "PUT" {
		remote, _ := serv.File()
		var local *os.File = nil
		var dsize int64 = 32*1024*1024
		var bsize int = 64*1024
		var ofile string = "-"
		//fmt.Printf("--I-- Rex %v\n",argv)
		if 1 < len(argv) {
			fname := argv[1]
			if strBegins(fname,"-z") {
				fmt.Sscanf(fname[2:],"%d",&dsize)
			}else
			if strBegins(fname,"{") {
				xin,xout,err := gsh.Popen(fname,"r")
				if err {
				}else{
					xout.Close()
					defer xin.Close()
					//in = xin
					local = xin
					fmt.Printf("--In- [%d] < Upload output of %v\n",
						local.Fd(),fname)
					ofile = "-from."+fname
					dsize = MaxStreamSize
				}
			}else{
				xlocal,err := os.Open(fname)
				if err != nil {
					fmt.Printf("--En- (%s)\n",err)
					local = nil
				}else{
					local = xlocal
					fi,_ := local.Stat()
					dsize = fi.Size()
					defer local.Close()
					//fmt.Printf("--I-- Rex in(%v / %v)\n",ofile,dsize)
				}
				ofile = fname
				fmt.Printf(Elapsed(Start)+"--In- L: open(%v,r)=%v %v (%v)\n",
					fname,dsize,local,err)
			}
		}
		if 2 < len(argv) && argv[2] != "" {
			ofile = argv[2]
			//fmt.Printf("(%d)%v B.ofile=%v\n",len(argv),argv,ofile)
		}
		//fmt.Printf(Elapsed(Start)+"--I-- Rex out(%v)\n",ofile)
		fmt.Printf(Elapsed(Start)+"--In- PUT %v (/%v)\n",dsize,bsize)
		req = fmt.Sprintf("PUT %v %v \r\n",dsize,ofile)
		if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
		fmt.Fprintf(serv,"%v",req)
		count,err = serv.Read(res)
		if debug { fmt.Printf(Elapsed(Start)+"--In- S: %v",string(res[0:count])) }
		fileRelay("SendPUT",local,remote,dsize,bsize)
	}else{
		req = fmt.Sprintf("%v\r\n",strings.Join(argv," "))
		if debug { fmt.Printf(Elapsed(Start)+"--In- C: %v",req) }
		fmt.Fprintf(serv,"%v",req)
		//fmt.Printf("--In- sending RexRequest(%v)\n",len(req))
	}
	//fmt.Printf(Elapsed(Start)+"--In- waiting RexResponse...\n")
	count,err = serv.Read(res)
	ress := ""
	if count == 0 {
		ress = "(nil)\r\n"
	}else{
		ress = string(res[:count])
	}
	if err != nil {
		fmt.Printf(Elapsed(Start)+"--En- S: (%d,%v) %v",count,err,ress)
	}else{
		fmt.Printf(Elapsed(Start)+"--In- S: %v",ress)
	}
	serv.Close()
	//conn.Close()

```

```
    var stat string
    var rcode int
    fmt.Sscanf(ress,"%d %s",&rcode,&stat)
    //fmt.Printf("--D-- Client: %v (%v)",rcode,stat)
    return rcode,ress
}

// <a name="remote-sh">Remote Shell</a>
// gcp file [...] { [host]:[port:][dir] | dir } // -p | -no-p
func (gsh*GshContext)FileCopy(argv[]string){
    var host = ""
    var port = ""
    var upload = false
    var download = false
    var xargv = []string{"rex-gcp"}
    var srcv = []string{}
    var dstv = []string{}
    argv = argv[1:]

    for _,v := range argv {
        /*
        if v[0] == '-' { // might be a pseudo file (generated date)
            continue
        }
        */
        obj := strings.Split(v,":")
        //fmt.Printf("%d %v %v\n",len(obj),v,obj)
        if 1 < len(obj) {
            host = obj[0]
            file := ""
            if 0 < len(host) {
                gsh.LastServer.host = host
            }else{
                host = gsh.LastServer.host
                port = gsh.LastServer.port
            }
            if 2 < len(obj) {
                port = obj[1]
                if 0 < len(port) {
                    gsh.LastServer.port = port
                }else{
                    port = gsh.LastServer.port
                }
                file = obj[2]
            }else{
                file = obj[1]
            }
            if len(srcv) == 0 {
                download = true
                srcv = append(srcv,file)
                continue
            }
            upload = true
            dstv = append(dstv,file)
            continue
        }
        /*
        idx := strings.Index(v,":")
        if 0 <= idx {
            remote = v[0:idx]
            if len(srcv) == 0 {
                download = true
                srcv = append(srcv,v[idx+1:])
                continue
            }
            upload = true
            dstv = append(dstv,v[idx+1:])
            continue
        }
        */
        if download {
            dstv = append(dstv,v)
        }else{
            srcv = append(srcv,v)
        }
    }
    hostport := "@" + host + ":" + port
    if upload {
        if host != "" { xargv = append(xargv,hostport) }
        xargv = append(xargv,"PUT")
        xargv = append(xargv,srcv[0:]...)
        xargv = append(xargv,dstv[0:]...)
    //fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v // %v\n",hostport,dstv,srcv,xargv)
    fmt.Printf("--I-- FileCopy PUT gsh://%s/%v < %v\n",hostport,dstv,srcv)
        gsh.RexecClient(xargv)
    }else
    if download {
        if host != "" { xargv = append(xargv,hostport) }
        xargv = append(xargv,"GET")
        xargv = append(xargv,srcv[0:]...)
        xargv = append(xargv,dstv[0:]...)
    //fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v // %v\n",hostport,srcv,dstv,xargv)
    fmt.Printf("--I-- FileCopy GET gsh://%v/%v > %v\n",hostport,srcv,dstv)
        gsh.RexecClient(xargv)
    }else{
    }
}

// target
func (gsh*GshContext)Trelpath(rloc string)(string){
    cwd, _ := os.Getwd()
    os.Chdir(gsh.RWD)
    os.Chdir(rloc)
    twd, _ := os.Getwd()
    os.Chdir(cwd)

    tpath := twd + "/" + rloc
    return tpath
}
// join to rmote GShell - [user@]host[:port] or cd host:[port]:path
func (gsh*GshContext)Rjoin(argv[]string){
    if len(argv) <= 1 {
        fmt.Printf("--I-- current server = %v\n",gsh.RSERV)
        return
    }
    serv := argv[1]
    servv := strings.Split(serv,":")
    if 1 <= len(servv) {
        if servv[0] == "lo" {
            servv[0] = "localhost"
        }
    }
    switch len(servv) {
        case 1:
            //if strings.Index(serv,":") < 0 {
```

```
			serv = servv[0] + ":" + fmt.Sprintf("%d",GSH_PORT)
			//}
		case 2: // host:port
			serv = strings.Join(servv,":")
	}
	xargv := []string{"rex-join","@"+serv,"HELO"}
	rcode,stat := gsh.RexecClient(xargv)
	if (rcode / 100) == 2 {
		fmt.Printf("--I-- OK Joined (%v) %v\n",rcode,stat)
		gsh.RSERV = serv
	}else{
		fmt.Printf("--I-- NG, could not joined (%v) %v\n",rcode,stat)
	}
}
func (gsh*GshContext)Rexec(argv[]string){
	if len(argv) <= 1 {
		fmt.Printf("--I-- rexec command [ | {file || {command} ]\n",gsh.RSERV)
		return
	}

	/*
	nargv := gshScanArg(strings.Join(argv," "),0)
	fmt.Printf("--D-- nargc=%d [%v]\n",len(nargv),nargv)
	if nargv[1][0] != '{' {
		nargv[1] = "{" + nargv[1] + "}"
		fmt.Printf("--D-- nargc=%d [%v]\n",len(nargv),nargv)
	}
	argv = nargv
	*/
	nargv := []string{}
	nargv = append(nargv,"{"+strings.Join(argv[1:]," ")+"}")
	fmt.Printf("--D-- nargc=%d %v\n",len(nargv),nargv)
	argv = nargv

	xargv := []string{"rex-exec","@"+gsh.RSERV,"GET"}
	xargv = append(xargv,argv...)
	xargv = append(xargv,"/dev/tty")
	rcode,stat := gsh.RexecClient(xargv)
	if (rcode / 100) == 2 {
		fmt.Printf("--I-- OK Rexec (%v) %v\n",rcode,stat)
	}else{
		fmt.Printf("--I-- NG Rexec (%v) %v\n",rcode,stat)
	}
}
func (gsh*GshContext)Rchdir(argv[]string){
	if len(argv) <= 1 {
		return
	}
	cwd, _ := os.Getwd()
	os.Chdir(gsh.RWD)
	os.Chdir(argv[1])
	twd, _ := os.Getwd()
	gsh.RWD = twd
	fmt.Printf("--I-- JWD=%v\n",twd)
	os.Chdir(cwd)
}
func (gsh*GshContext)Rpwd(argv[]string){
	fmt.Printf("%v\n",gsh.RWD)
}
func (gsh*GshContext)Rls(argv[]string){
	cwd, _ := os.Getwd()
	os.Chdir(gsh.RWD)
	argv[0] = "-ls"
	gsh.xFind(argv)
	os.Chdir(cwd)
}
func (gsh*GshContext)Rput(argv[]string){
	var local string = ""
	var remote string = ""
	if 1 < len(argv) {
		local = argv[1]
		remote = local // base name
	}
	if 2 < len(argv) {
		remote = argv[2]
	}
	fmt.Printf("--I-- jput from=%v to=%v\n",local,gsh.Trelpath(remote))
}
func (gsh*GshContext)Rget(argv[]string){
	var remote string = ""
	var local string = ""
	if 1 < len(argv) {
		remote = argv[1]
		local = remote // base name
	}
	if 2 < len(argv) {
		local = argv[2]
	}
	fmt.Printf("--I-- jget from=%v to=%v\n",gsh.Trelpath(remote),local)
}

// <a name="network">network</a>
// -s, -si, -so // bi-directional, source, sync (maybe socket)
func (gshCtx*GshContext)sconnect(inTCP bool, argv []string) {
	gshPA := gshCtx.gshPA
	if len(argv) < 2 {
		fmt.Printf("Usage: -s [host]:[port[.udp]]\n")
		return
	}
	remote := argv[1]
	if remote == ":" { remote = "0.0.0.0:9999" }

	if inTCP { // TCP
		dport, err := net.ResolveTCPAddr("tcp",remote);
		if err != nil {
			fmt.Printf("Address error: %s (%s)\n",remote,err)
			return
		}
		conn, err := net.DialTCP("tcp",nil,dport)
		if err != nil {
			fmt.Printf("Connection error: %s (%s)\n",remote,err)
			return
		}
		file, _ := conn.File();
		fd := file.Fd()
		fmt.Printf("Socket: connected to %s, socket[%d]\n",remote,fd)

		savfd := gshPA.Files[1]
		gshPA.Files[1] = fd;
		gshCtx.gshellv(argv[2:])
		gshPA.Files[1] = savfd
		file.Close()
		conn.Close()
	}else{
		//dport, err := net.ResolveUDPAddr("udp4",remote);
```

```
        dport, err := net.ResolveUDPAddr("udp",remote);
        if err != nil {
            fmt.Printf("Address error: %s (%s)\n",remote,err)
            return
        }
        //conn, err := net.DialUDP("udp4",nil,dport)
        conn, err := net.DialUDP("udp",nil,dport)
        if err != nil {
            fmt.Printf("Connection error: %s (%s)\n",remote,err)
            return
        }
        file, _ := conn.File();
        fd := file.Fd()

        ar := conn.RemoteAddr()
        //al := conn.LocalAddr()
        fmt.Printf("Socket: connected to %s [%s], socket[%d]\n",
            remote,ar.String(),fd)

        savfd := gshPA.Files[1]
        gshPA.Files[1] = fd;
        gshCtx.gshellv(argv[2:])
        gshPA.Files[1] = savfd
        file.Close()
        conn.Close()
    }
}
func (gshCtx*GshContext)saccept(inTCP bool, argv []string) {
    gshPA := gshCtx.gshPA
    if len(argv) < 2 {
        fmt.Printf("Usage: -ac [host]:[port[.udp]]\n")
        return
    }
    local := argv[1]
    if local == ":" { local = "0.0.0.0:9999" }
    if inTCP { // TCP
        port, err := net.ResolveTCPAddr("tcp",local);
        if err != nil {
            fmt.Printf("Address error: %s (%s)\n",local,err)
            return
        }
        //fmt.Printf("Listen at %s...\n",local);
        sconn, err := net.ListenTCP("tcp", port)
        if err != nil {
            fmt.Printf("Listen error: %s (%s)\n",local,err)
            return
        }
        //fmt.Printf("Accepting at %s...\n",local);
        aconn, err := sconn.AcceptTCP()
        if err != nil {
            fmt.Printf("Accept error: %s (%s)\n",local,err)
            return
        }
        file, _ := aconn.File()
        fd := file.Fd()
        fmt.Printf("Accepted TCP at %s [%d]\n",local,fd)

        savfd := gshPA.Files[0]
        gshPA.Files[0] = fd;
        gshCtx.gshellv(argv[2:])
        gshPA.Files[0] = savfd

        sconn.Close();
        aconn.Close();
        file.Close();
    }else{
        //port, err := net.ResolveUDPAddr("udp4",local);
        port, err := net.ResolveUDPAddr("udp",local);
        if err != nil {
            fmt.Printf("Address error: %s (%s)\n",local,err)
            return
        }
        fmt.Printf("Listen UDP at %s...\n",local);
        //uconn, err := net.ListenUDP("udp4", port)
        uconn, err := net.ListenUDP("udp", port)
        if err != nil {
            fmt.Printf("Listen error: %s (%s)\n",local,err)
            return
        }
        file, _ := uconn.File()
        fd := file.Fd()
        ar := uconn.RemoteAddr()
        remote := ""
        if ar != nil { remote = ar.String() }
        if remote == "" { remote = "?" }

        // not yet received
        //fmt.Printf("Accepted at %s [%d] <- %s\n",local,fd,"")

        savfd := gshPA.Files[0]
        gshPA.Files[0] = fd;
        savenv := gshPA.Env
        gshPA.Env = append(savenv, "REMOTE_HOST="+remote)
        gshCtx.gshellv(argv[2:])
        gshPA.Env = savenv
        gshPA.Files[0] = savfd

        uconn.Close();
        file.Close();
    }
}

// empty line command
func (gshCtx*GshContext)xPwd(argv[]string){
    // execute context command, pwd + date
    // context notation, representation scheme, to be resumed at re-login
    cwd, _ := os.Getwd()
    switch {
    case isin("-a",argv):
        gshCtx.ShowChdirHistory(argv)
    case isin("-ls",argv):
        showFileInfo(cwd,argv)
    default:
        fmt.Printf("%s\n",cwd)
    case isin("-v",argv): // obsolete emtpy command
        t := time.Now()
        date := t.Format(time.UnixDate)
        exe, _ := os.Executable()
        host, _ := os.Hostname()
        fmt.Printf("{PWD=\"%s\"",cwd)
        fmt.Printf(" HOST=\"%s\"",host)
        fmt.Printf(" DATE=\"%s\"",date)
        fmt.Printf(" TIME=\"%s\"",t.String())
        fmt.Printf(" PID=\"%d\"",os.Getpid())
        fmt.Printf(" EXE=\"%s\"",exe)
        fmt.Printf("}\n")
    }
}

// <a name="history">History</a>
// these should be browsed and edited by HTTP browser
// show the time of command with -t and direcotry with -ls
// openfile-history, sort by -a -m -c
// sort by elapsed time by -t -s
// search by "more" like interface
// edit history
// sort history, and wc or uniq
// CPU and other resource consumptions
// limit showing range (by time or so)
// export / import history
func (gshCtx *GshContext)xHistory(argv []string){
    atWorkDirX := -1
    if 1 < len(argv) && strBegins(argv[1],"@") {
        atWorkDirX,_ = strconv.Atoi(argv[1][1:])
    }
    //fmt.Printf("--D-- showHistory(%v)\n",argv)
    for i, v := range gshCtx.CommandHistory {
        // exclude commands not to be listed by default
        // internal commands may be suppressed by default
        if v.CmdLine == "" && !isin("-a",argv) {
            continue;
        }
        if 0 <= atWorkDirX {
            if v.WorkDirX != atWorkDirX {
                continue
            }
        }
        if !isin("-n",argv){ // like "fc"
            fmt.Printf("!%-2d ",i)
        }
        if isin("-v",argv){
            fmt.Println(v) // should be with it date
        }else{
            if isin("-l",argv) || isin("-l0",argv) {
                elps := v.EndAt.Sub(v.StartAt);
                start := v.StartAt.Format(time.Stamp)
                fmt.Printf("@%d ",v.WorkDirX)
                fmt.Printf("[%v] %11v/t ",start,elps)
            }
            if isin("-l",argv) && !isin("-l0",argv){
                fmt.Printf("%v",Rusagef("%t %u\t// %s",argv,v.Rusagev))
            }
            if isin("-at",argv) { // isin("-ls",argv){
                dhi := v.WorkDirX // workdir history index
                fmt.Printf("@%d %s\t",dhi,v.WorkDir)
                // show the FileInfo of the output command??
            }
            fmt.Printf("%s",v.CmdLine)
            fmt.Printf("\n")
        }
    }
}
// !n - history index
func searchHistory(gshCtx GshContext, gline string) (string, bool, bool){
    if gline[0] == '!' {
        hix, err := strconv.Atoi(gline[1:])
        if err != nil {
            fmt.Printf("--E-- (%s : range)\n",hix)
            return "", false, true
        }
        if hix < 0 || len(gshCtx.CommandHistory) <= hix {
            fmt.Printf("--E-- (%d : out of range)\n",hix)
            return "", false, true
        }
        return gshCtx.CommandHistory[hix].CmdLine, false, false
    }
    // search
    //for i, v := range gshCtx.CommandHistory {
    //}
    return gline, false, false
}
func (gsh*GshContext)cmdStringInHistory(hix int)(cmd string, ok bool){
    if 0 <= hix && hix < len(gsh.CommandHistory) {
        return gsh.CommandHistory[hix].CmdLine,true
    }
    return "",false
}

// temporary adding to PATH environment
// cd name -lib for LD_LIBRARY_PATH
// chdir with directory history (date + full-path)
// -s for sort option (by visit date or so)
func (gsh*GshContext)ShowChdirHistory1(i int,v GChdirHistory, argv []string){
    fmt.Printf("!%-2d ",v.CmdIndex) // the first command at this WorkDir
    fmt.Printf("@%d ",i)
    fmt.Printf("[%v] ",v.MovedAt.Format(time.Stamp))
    showFileInfo(v.Dir,argv)
}
func (gsh*GshContext)ShowChdirHistory(argv []string){
    for i, v := range gsh.ChdirHistory {
        gsh.ShowChdirHistory1(i,v,argv)
    }
}
func skipOpts(argv[]string)(int){
    for i,v := range argv {
        if strBegins(v,"-") {
        }else{
            return i
        }
    }
    return -1
}
func (gshCtx*GshContext)xChdir(argv []string){
    cdhist := gshCtx.ChdirHistory
    if isin("?",argv ) || isin("-t",argv) || isin("-a",argv) {
        gshCtx.ShowChdirHistory(argv)
        return
    }
    pwd, _ := os.Getwd()
    dir := ""
    if len(argv) <= 1 {
        dir = toFullpath("~")
    }else{
        i := skipOpts(argv[1:])
        if i < 0 {
            dir = toFullpath("~")
        }else{
            dir = argv[1+i]
        }
    }
```

```
	if strBegins(dir,"@") {
		if dir == "@0" { // obsolete
			dir = gshCtx.StartDir
		}else
		if dir == "@!" {
			index := len(cdhist) - 1
			if 0 < index { index -= 1 }
			dir = cdhist[index].Dir
		}else{
			index, err := strconv.Atoi(dir[1:])
			if err != nil {
				fmt.Printf("--E-- xChdir(%v)\n",err)
				dir = "?"
			}else
			if len(gshCtx.ChdirHistory) <= index {
				fmt.Printf("--E-- xChdir(history range error)\n")
				dir = "?"
			}else{
				dir = cdhist[index].Dir
			}
		}
	}
	if dir != "?" {
		err := os.Chdir(dir)
		if err != nil {
			fmt.Printf("--E-- xChdir(%s)(%v)\n",argv[1],err)
		}else{
			cwd, _ := os.Getwd()
			if cwd != pwd {
				hist1 := GChdirHistory { }
				hist1.Dir = cwd
				hist1.MovedAt = time.Now()
				hist1.CmdIndex = len(gshCtx.CommandHistory)+1
				gshCtx.ChdirHistory = append(cdhist,hist1)
				if !isin("-s",argv){
					//cwd, _ := os.Getwd()
					//fmt.Printf("%s\n",cwd)
					ix := len(gshCtx.ChdirHistory)-1
					gshCtx.ShowChdirHistory1(ix,hist1,argv)
				}
			}
		}
	}
	if isin("-ls",argv){
		cwd, _ := os.Getwd()
		showFileInfo(cwd,argv);
	}
}
func TimeValSub(tv1 *syscall.Timeval, tv2 *syscall.Timeval){
	*tv1 = syscall.NsecToTimeval(tv1.Nano() - tv2.Nano())
}
func RusageSubv(ru1, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
	TimeValSub(&ru1[0].Utime,&ru2[0].Utime)
	TimeValSub(&ru1[0].Stime,&ru2[0].Stime)
	TimeValSub(&ru1[1].Utime,&ru2[1].Utime)
	TimeValSub(&ru1[1].Stime,&ru2[1].Stime)
	return ru1
}
func TimeValAdd(tv1 syscall.Timeval, tv2 syscall.Timeval)(syscall.Timeval){
	tvs := syscall.NsecToTimeval(tv1.Nano() + tv2.Nano())
	return tvs
}
/*
func RusageAddv(ru1, ru2 [2]syscall.Rusage)([2]syscall.Rusage){
	TimeValAdd(ru1[0].Utime,ru2[0].Utime)
	TimeValAdd(ru1[0].Stime,ru2[0].Stime)
	TimeValAdd(ru1[1].Utime,ru2[1].Utime)
	TimeValAdd(ru1[1].Stime,ru2[1].Stime)
	return ru1
}
*/

// <a name="rusage">Resource Usage</a>
func sRusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
	// ru[0] self , ru[1] children
	ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
	st := TimeValAdd(ru[0].Stime,ru[1].Stime)
	uu := (ut.Sec*1000000 + int64(ut.Usec)) * 1000
	su := (st.Sec*1000000 + int64(st.Usec)) * 1000
	tu := uu + su
	ret := fmt.Sprintf("%v/sum",abbtime(tu))
	ret += fmt.Sprintf(", %v/usr",abbtime(uu))
	ret += fmt.Sprintf(", %v/sys",abbtime(su))
	return ret
}
func Rusagef(fmtspec string, argv []string, ru [2]syscall.Rusage)(string){
	ut := TimeValAdd(ru[0].Utime,ru[1].Utime)
	st := TimeValAdd(ru[0].Stime,ru[1].Stime)
	fmt.Printf("%d.%06ds/u ",ut.Sec,ut.Usec) //ru[1].Utime.Sec,ru[1].Utime.Usec)
	fmt.Printf("%d.%06ds/s ",st.Sec,st.Usec) //ru[1].Stime.Sec,ru[1].Stime.Usec)
	return ""
}
func Getrusagev()([2]syscall.Rusage){
	var ruv = [2]syscall.Rusage{}
	syscall.Getrusage(syscall.RUSAGE_SELF,&ruv[0])
	syscall.Getrusage(syscall.RUSAGE_CHILDREN,&ruv[1])
	return ruv
}
func showRusage(what string,argv []string, ru *syscall.Rusage){
	fmt.Printf("%s: ",what);
	fmt.Printf("Usr=%d.%06ds",ru.Utime.Sec,ru.Utime.Usec)
	fmt.Printf(" Sys=%d.%06ds",ru.Stime.Sec,ru.Stime.Usec)
	fmt.Printf(" Rss=%vB",ru.Maxrss)
	if isin("-l",argv) {
		fmt.Printf(" MinFlt=%v",ru.Minflt)
		fmt.Printf(" MajFlt=%v",ru.Majflt)
		fmt.Printf(" IxRSS=%vB",ru.Ixrss)
		fmt.Printf(" IdRSS=%vB",ru.Idrss)
		fmt.Printf(" Nswap=%vB",ru.Nswap)
	fmt.Printf(" Read=%v",ru.Inblock)
	fmt.Printf(" Write=%v",ru.Oublock)
	}
	fmt.Printf(" Snd=%v",ru.Msgsnd)
	fmt.Printf(" Rcv=%v",ru.Msgrcv)
	//if isin("-l",argv) {
		fmt.Printf(" Sig=%v",ru.Nsignals)
	//}
	fmt.Printf("\n");
}
func (gshCtx *GshContext)xTime(argv[]string)(bool){
	if 2 <= len(argv){
		gshCtx.LastRusage = syscall.Rusage{}
		rusagev1 := Getrusagev()
		fin := gshCtx.gshellv(argv[1:])
		rusagev2 := Getrusagev()
		showRusage(argv[1],argv,&gshCtx.LastRusage)
		rusagev := RusageSubv(rusagev2,rusagev1)
		showRusage("self",argv,&rusagev[0])
		showRusage("chld",argv,&rusagev[1])
		return fin
	}else{
		rusage:= syscall.Rusage {}
		syscall.Getrusage(syscall.RUSAGE_SELF,&rusage)
		showRusage("self",argv, &rusage)
		syscall.Getrusage(syscall.RUSAGE_CHILDREN,&rusage)
		showRusage("chld",argv, &rusage)
		return false
	}
}
func (gshCtx *GshContext)xJobs(argv[]string){
	fmt.Printf("%d Jobs\n",len(gshCtx.BackGroundJobs))
	for ji, pid := range gshCtx.BackGroundJobs {
		//wstat := syscall.WaitStatus {0}
		rusage := syscall.Rusage {}
		//wpid, err := syscall.Wait4(pid,&wstat,syscall.WNOHANG,&rusage);
		wpid, err := syscall.Wait4(pid,nil,syscall.WNOHANG,&rusage);
		if err != nil {
			fmt.Printf("--E-- %%%d [%d] (%v)\n",ji,pid,err)
		}else{
			fmt.Printf("%%%d[%d](%d)\n",ji,pid,wpid)
			showRusage("chld",argv,&rusage)
		}
	}
}
func (gsh*GshContext)inBackground(argv[]string)(bool){
	if gsh.CmdTrace { fmt.Printf("--I-- inBackground(%v)\n",argv) }
	gsh.BackGround = true // set background option
	xfin := false
	xfin = gsh.gshellv(argv)
	gsh.BackGround = false
	return xfin
}
// -o file without command means just opening it and refer by #N
// should be listed by "files" comnmand
func (gshCtx*GshContext)xOpen(argv[]string){
	var pv = []int{-1,-1}
	err := syscall.Pipe(pv)
	fmt.Printf("--I-- pipe()=[#%d,#%d](%v)\n",pv[0],pv[1],err)
}
func (gshCtx*GshContext)fromPipe(argv[]string){
}
func (gshCtx*GshContext)xClose(argv[]string){
}

// <a name="redirect">redirect</a>
func (gshCtx*GshContext)redirect(argv[]string)(bool){
	if len(argv) < 2 {
		return false
	}

	cmd := argv[0]
	fname := argv[1]
	var file *os.File = nil

	fdix := 0
	mode := os.O_RDONLY

	switch {
	case cmd == "-i" || cmd == "<":
		fdix = 0
		mode = os.O_RDONLY
	case cmd == "-o" || cmd == ">":
		fdix = 1
		mode = os.O_RDWR | os.O_CREATE
	case cmd == "-a" || cmd == ">>":
		fdix = 1
		mode = os.O_RDWR | os.O_CREATE | os.O_APPEND
	}
	if fname[0] == '#' {
		fd, err := strconv.Atoi(fname[1:])
		if err != nil {
			fmt.Printf("--E-- (%v)\n",err)
			return false
		}
		file = os.NewFile(uintptr(fd),"MaybePipe")
	}else{
		xfile, err := os.OpenFile(argv[1], mode, 0600)
		if err != nil {
			fmt.Printf("--E-- (%s)\n",err)
			return false
		}
		file = xfile
	}
	gshPA := gshCtx.gshPA
	savfd := gshPA.Files[fdix]
	gshPA.Files[fdix] = file.Fd()
	fmt.Printf("--I-- Opened [%d] %s\n",file.Fd(),argv[1])
	gshCtx.gshellv(argv[2:])
	gshPA.Files[fdix] = savfd

	return false
}

//fmt.Fprintf(res, "GShell Status: %q", html.EscapeString(req.URL.Path))
func httpHandler(res http.ResponseWriter, req *http.Request){
	path := req.URL.Path
	fmt.Printf("--I-- Got HTTP Request(%s)\n",path)
	{
		gshCtxBuf, _ :=  setupGshContext()
		gshCtx := &gshCtxBuf
		fmt.Printf("--I-- %s\n",path[1:])
		gshCtx.tgshelll(path[1:])
	}
	fmt.Fprintf(res, "Hello(^-^)/\n%s\n",path)
}
func (gshCtx *GshContext) httpServer(argv []string){
	http.HandleFunc("/", httpHandler)
	accport := "localhost:9999"
	fmt.Printf("--I-- HTTP Server Start at [%s]\n",accport)
	http.ListenAndServe(accport,nil)
}
func (gshCtx *GshContext)xGo(argv[]string){
	go gshCtx.gshellv(argv[1:]);
}
func (gshCtx *GshContext) xPs(argv[]string)(){
}

// <a name="plugin">Plugin</a>
// plugin [-ls [names]] to list plugins
// Reference: <a href="https://golang.org/src/plugin/">plugin</a> source code
```

```
func (gshCtx *GshContext) whichPlugin(name string,argv[]string)(pi *PluginInfo){
	pi = nil
	for _,p := range gshCtx.PluginFuncs {
		if p.Name == name && pi == nil {
			pi = &p
		}
		if !isin("-s",argv){
			//fmt.Printf("%v %v ",i,p)
			if isin("-ls",argv){
				showFileInfo(p.Path,argv)
			}else{
				fmt.Printf("%s\n",p.Name)
			}
		}
	}
	return pi
}
func (gshCtx *GshContext) xPlugin(argv[]string) (error) {
	if len(argv) == 0 || argv[0] == "-ls" {
		gshCtx.whichPlugin("",argv)
		return  nil
	}
	name := argv[0]
	Pin := gshCtx.whichPlugin(name,[]string{"-s"})
	if Pin != nil {
		os.Args = argv // should be recovered?
		Pin.Addr.(func())()
		return nil
	}
	sofile := toFullpath(argv[0] + ".so") // or find it by which($PATH)

	p, err := plugin.Open(sofile)
	if err != nil {
		fmt.Printf("--E-- plugin.Open(%s)(%v)\n",sofile,err)
		return err
	}
	fname := "Main"
	f, err := p.Lookup(fname)
	if( err != nil ){
		fmt.Printf("--E-- plugin.Lookup(%s)(%v)\n",fname,err)
		return err
	}
	pin := PluginInfo {p,f,name,sofile}
	gshCtx.PluginFuncs = append(gshCtx.PluginFuncs,pin)
	fmt.Printf("--I-- added (%d)\n",len(gshCtx.PluginFuncs))

	//fmt.Printf("--I-- first call(%s:%s)%v\n",sofile,fname,argv)
	os.Args = argv
	f.(func())()
	return err
}
func (gshCtx*GshContext)Args(argv[]string){
	for i,v := range os.Args {
		fmt.Printf("[%v] %v\n",i,v)
	}
}
func (gshCtx *GshContext) showVersion(argv[]string){
	if isin("-l",argv) {
		fmt.Printf("%v/%v (%v)",NAME,VERSION,DATE);
	}else{
		fmt.Printf("%v",VERSION);
	}
	if isin("-a",argv) {
		fmt.Printf(" %s",AUTHOR)
	}
	if !isin("-n",argv) {
		fmt.Printf("\n")
	}
}

// <a name="scanf">Scanf</a> // string decomposer
// scanf [format] [input]
func scanv(sstr string)(strv[]string){
	strv = strings.Split(sstr," ")
	return strv
}
func scanUntil(src,end string)(rstr string,leng int){
	idx := strings.Index(src,end)
	if 0 <= idx {
		rstr = src[0:idx]
		return rstr,idx+len(end)
	}
	return src,0
}

// -bn -- display base-name part only // can be in some %fmt, for sed rewriting
func (gsh*GshContext)printVal(fmts string, vstr string, optv[]string){
	//vint,err := strconv.Atoi(vstr)
	var ival int64 = 0
	n := 0
	err := error(nil)
	if strBegins(vstr,"_") {
		vx,_ := strconv.Atoi(vstr[1:])
		if vx < len(gsh.iValues) {
			vstr = gsh.iValues[vx]
		}else{
		}
	}
	// should use Eval()
	if strBegins(vstr,"0x") {
		n,err = fmt.Sscanf(vstr[2:],"%x",&ival)
	}else{
		n,err = fmt.Sscanf(vstr,"%d",&ival)
//fmt.Printf("--D-- n=%d err=(%v) {%s}=%v\n",n,err,vstr, ival)
	}
	if n == 1 && err == nil {
		//fmt.Printf("--D-- formatn(%v) ival(%v)\n",fmts,ival)
		fmt.Printf("%"+fmts,ival)
	}else{
		if isin("-bn",optv){
			fmt.Printf("%"+fmts,filepath.Base(vstr))
		}else{
			fmt.Printf("%"+fmts,vstr)
		}
	}
}
func (gsh*GshContext)printfv(fmts,div string,argv[]string,optv[]string,list[]string){
	//fmt.Printf("{%d}",len(list))
	//curfmt := "v"
	outlen := 0
	curfmt := gsh.iFormat

	if 0 < len(fmts) {
		for xi := 0; xi < len(fmts); xi++ {
			fch := fmts[xi]
```

```
            if fch == '%' {
                if xi+1 < len(fmts) {
                    curfmt = string(fmts[xi+1])
 gsh.iFormat = curfmt
                    xi += 1
    if xi+1 < len(fmts) && fmts[xi+1] == '(' {
        vals,leng := scanUntil(fmts[xi+2:],")")
        //fmt.Printf("--D-- show fmt(%v) val(%v) next(%v)\n",curfmt,vals,leng)
        gsh.printVal(curfmt,vals,optv)
        xi += 2+leng-1
        outlen += 1
    }
                    continue
                }
            }
            if fch == '_' {
                hi,leng := scanInt(fmts[xi+1:])
                if 0 < leng {
                    if hi < len(gsh.iValues) {
                        gsh.printVal(curfmt,gsh.iValues[hi],optv)
                        outlen += 1 // should be the real length
                    }else{
                        fmt.Printf("((out-range))")
                    }
                    xi += leng
                    continue;
                }
            }
            fmt.Printf("%c",fch)
            outlen += 1
        }
    }else{
        //fmt.Printf("--D-- print {%s}\n")
        for i,v := range list {
            if 0 < i {
                fmt.Printf(div)
            }
            gsh.printVal(curfmt,v,optv)
            outlen += 1
        }
    }
    if 0 < outlen {
        fmt.Printf("\n")
    }
}
func (gsh*GshContext)Scanv(argv[]string){
    //fmt.Printf("--D-- Scanv(%v)\n",argv)
    if len(argv) == 1 {
        return
    }
    argv = argv[1:]
    fmts := ""
    if strBegins(argv[0],"-F") {
        fmts = argv[0]
        gsh.iDelimiter = fmts
        argv = argv[1:]
    }
    input := strings.Join(argv," ")
    if fmts == "" { // simple decomposition
        v := scanv(input)
        gsh.iValues = v
        //fmt.Printf("%v\n",strings.Join(v,","))
    }else{
        v := make([]string,8)
        n,err := fmt.Sscanf(input,fmts,&v[0],&v[1],&v[2],&v[3])
        fmt.Printf("--D-- Scanf ->(%v) n=%d err=(%v)\n",v,n,err)
        gsh.iValues = v
    }
}
func (gsh*GshContext)Printv(argv[]string){
    if false { //@@U
        fmt.Printf("%v\n",strings.Join(argv[1:]," "))
        return
    }
    //fmt.Printf("--D-- Printv(%v)\n",argv)
    //fmt.Printf("%v\n",strings.Join(gsh.iValues,","))
    div := gsh.iDelimiter
    fmts := ""
    argv = argv[1:]
    if 0 < len(argv) {
        if strBegins(argv[0],"-F") {
            div = argv[0][2:]
            argv = argv[1:]
        }
    }

    optv := []string{}
    for _,v := range argv {
        if strBegins(v,"-"){
            optv = append(optv,v)
            argv = argv[1:]
        }else{
            break;
        }
    }
    if 0 < len(argv) {
        fmts = strings.Join(argv," ")
    }
    gsh.printfv(fmts,div,argv,optv,gsh.iValues)
}
func (gsh*GshContext)Basename(argv[]string){
    for i,v := range gsh.iValues {
        gsh.iValues[i] = filepath.Base(v)
    }
}
func (gsh*GshContext)Sortv(argv[]string){
    sv := gsh.iValues
    sort.Slice(sv , func(i,j int) bool {
        return sv[i] < sv[j]
    })
}
func (gsh*GshContext)Shiftv(argv[]string){
    vi := len(gsh.iValues)
    if 0 < vi {
        if isin("-r",argv) {
            top := gsh.iValues[0]
            gsh.iValues = append(gsh.iValues[1:],top)
        }else{
            gsh.iValues = gsh.iValues[1:]
        }
    }
}

func (gsh*GshContext)Enq(argv[]string){
}
```

```
func (gsh*GshContext)Deq(argv[]string){
}
func (gsh*GshContext)Push(argv[]string){
    gsh.iValStack = append(gsh.iValStack,argv[1:])
    fmt.Printf("depth=%d\n",len(gsh.iValStack))
}
func (gsh*GshContext)Dump(argv[]string){
    for i,v := range gsh.iValStack {
        fmt.Printf("%d %v\n",i,v)
    }
}
func (gsh*GshContext)Pop(argv[]string){
    depth := len(gsh.iValStack)
    if 0 < depth {
        v := gsh.iValStack[depth-1]
        if isin("-cat",argv){
            gsh.iValues = append(gsh.iValues,v...)
        }else{
            gsh.iValues = v
        }
        gsh.iValStack = gsh.iValStack[0:depth-1]
        fmt.Printf("depth=%d %s\n",len(gsh.iValStack),gsh.iValues)
    }else{
        fmt.Printf("depth=%d\n",depth)
    }
}

// <a name="interpreter">Command Interpreter</a>
func (gshCtx*GshContext)gshellv(argv []string) (fin bool) {
    fin = false

    if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr,"--I-- gshellv((%d))\n",len(argv)) }
    if len(argv) <= 0 {
        return false
    }
    xargv := []string{}
    for ai := 0; ai < len(argv); ai++ {
        xargv = append(xargv,strsubst(gshCtx,argv[ai],false))
    }
    argv = xargv
    if false {
        for ai := 0; ai < len(argv); ai++ {
            fmt.Printf("[%d] %s [%d]%T\n",
                ai,argv[ai],len(argv[ai]),argv[ai])
        }
    }
    cmd := argv[0]
    if gshCtx.CmdTrace { fmt.Fprintf(os.Stderr,"--I-- gshellv(%d)%v\n",len(argv),argv) }
    switch { // https://tour.golang.org/flowcontrol/11
    case cmd == "":
        gshCtx.xPwd([]string{}); // emtpy command
    case cmd == "-x":
        gshCtx.CmdTrace = ! gshCtx.CmdTrace
    case cmd == "-xt":
        gshCtx.CmdTime = ! gshCtx.CmdTime
    case cmd == "-ot":
        gshCtx.sconnect(true, argv)
    case cmd == "-ou":
        gshCtx.sconnect(false, argv)
    case cmd == "-it":
        gshCtx.saccept(true , argv)
    case cmd == "-iu":
        gshCtx.saccept(false, argv)
    case cmd == "-i" || cmd == "<" || cmd == "-o" || cmd == ">" || cmd == "-a" || cmd == ">>" || cmd == "-s" || cmd == "><":
        gshCtx.redirect(argv)
    case cmd == "|":
        gshCtx.fromPipe(argv)
    case cmd == "args":
        gshCtx.Args(argv)
    case cmd == "bg" || cmd == "-bg":
        rfin := gshCtx.inBackground(argv[1:])
        return rfin
    case cmd == "-bn":
        gshCtx.Basename(argv)
    case cmd == "call":
        _,_ = gshCtx.excommand(false,argv[1:])
    case cmd == "cd" || cmd == "chdir":
        gshCtx.xChdir(argv);
    case cmd == "-cksum":
        gshCtx.xFind(argv)
    case cmd == "-sum":
        gshCtx.xFind(argv)
    case cmd == "close":
        gshCtx.xClose(argv)
    case cmd == "gcp":
        gshCtx.FileCopy(argv)
    case cmd == "dec" || cmd == "decode":
        gshCtx.Dec(argv)
    case cmd == "#define":
    case cmd == "dump":
        gshCtx.Dump(argv)
    case cmd == "echo":
        echo(argv,true)
    case cmd == "enc" || cmd == "encode":
        gshCtx.Enc(argv)
    case cmd == "env":
        env(argv)
    case cmd == "eval":
        xEval(argv[1:],true)
    case cmd == "exec":
        _,_ = gshCtx.excommand(true,argv[1:])
        // should not return here
    case cmd == "exit" || cmd == "quit":
        // write Result code EXIT to 3>
        return true
    case cmd == "fdls":
        // dump the attributes of fds (of other process)
    case cmd == "-find" || cmd == "fin" || cmd == "ufind" || cmd == "uf":
        gshCtx.xFind(argv[1:])
    case cmd == "fu":
        gshCtx.xFind(argv[1:])
    case cmd == "fork":
        // mainly for a server
    case cmd == "-gen":
        gshCtx.gen(argv)
    case cmd == "-go":
        gshCtx.xGo(argv)
    case cmd == "-grep":
        gshCtx.xFind(argv)
    case cmd == "gdeq":
        gshCtx.Deq(argv)
    case cmd == "genq":
        gshCtx.Enq(argv)
    case cmd == "gpop":
        gshCtx.Pop(argv)
    case cmd == "gpush":
        gshCtx.Push(argv)
    case cmd == "history" || cmd == "hi": // hi should be alias
        gshCtx.xHistory(argv)
    case cmd == "jobs":
        gshCtx.xJobs(argv)
    case cmd == "lnsp":
        gshCtx.SplitLine(argv)
    case cmd == "-ls":
        gshCtx.xFind(argv)
    case cmd == "nop":
        // do nothing
    case cmd == "pipe":
        gshCtx.xOpen(argv)
    case cmd == "plug" || cmd == "plugin" || cmd == "pin":
        gshCtx.xPlugin(argv[1:])
    case cmd == "print" || cmd == "-pr":
        // output internal slice // also sprintf should be
        gshCtx.Printv(argv)
    case cmd == "ps":
        gshCtx.xPs(argv)
    case cmd == "pstitle":
        // to be gsh.title
    case cmd == "rexecd" || cmd == "rexd":
        gshCtx.RexecServer(argv)
    case cmd == "rexec" || cmd == "rex":
        gshCtx.RexecClient(argv)
    case cmd == "repeat" || cmd == "rep": // repeat cond command
        gshCtx.repeat(argv)
    case cmd == "scan":
        // scan input (or so in fscanf) to internal slice (like Files or map)
        gshCtx.Scanv(argv)
    case cmd == "set":
        // set name ...
    case cmd == "serv":
        gshCtx.httpServer(argv)
    case cmd == "shift":
        gshCtx.Shiftv(argv)
    case cmd == "sleep":
        gshCtx.sleep(argv)
    case cmd == "-sort":
        gshCtx.Sortv(argv)

    case cmd == "j" || cmd == "join":
        gshCtx.Rjoin(argv)
    case cmd == "a" || cmd == "alpa":
        gshCtx.Rexec(argv)
    case cmd == "jcd" || cmd == "jchdir":
        gshCtx.Rchdir(argv)
    case cmd == "jget":
        gshCtx.Rget(argv)
    case cmd == "jls":
        gshCtx.Rls(argv)
    case cmd == "jput":
        gshCtx.Rput(argv)
    case cmd == "jpwd":
        gshCtx.Rpwd(argv)

    case cmd == "time":
        fin = gshCtx.xTime(argv)
    case cmd == "pwd":
        gshCtx.xPwd(argv);
    case cmd == "ver" || cmd == "-ver" || cmd == "version":
        gshCtx.showVersion(argv)
    case cmd == "where":
        // data file or so?
    case cmd == "which":
        which("PATH",argv);
    default:
        if gshCtx.whichPlugin(cmd,[]string{"-s"}) != nil {
            gshCtx.xPlugin(argv)
        }else{
            notfound,_ := gshCtx.excommand(false,argv)
            if notfound {
                fmt.Printf("--E-- command not found (%v)\n",cmd)
            }
        }
    }
    return fin
}

func (gsh*GshContext)gshelll(gline string) (rfin bool) {
    argv := strings.Split(string(gline)," ")
    fin := gsh.gshellv(argv)
    return fin
}
func (gsh*GshContext)tgshelll(gline string)(xfin bool){
    start := time.Now()
    fin := gsh.gshelll(gline)
    end := time.Now()
    elps := end.Sub(start);
    if gsh.CmdTime {
        fmt.Printf("--T-- " + time.Now().Format(time.Stamp) + "(%d.%09ds)\n",
            elps/1000000000,elps%1000000000)
    }
    return fin
}
func Ttyid() (int) {
    fi, err := os.Stdin.Stat()
    if err != nil {
        return 0;
    }
    //fmt.Printf("Stdin: %v Dev=%d\n",
    //  fi.Mode(),fi.Mode()&os.ModeDevice)
    if (fi.Mode() & os.ModeDevice) != 0 {
        stat := syscall.Stat_t{};
        err := syscall.Fstat(0,&stat)
        if err != nil {
            //fmt.Printf("--I-- Stdin: (%v)\n",err)
        }else{
            //fmt.Printf("--I-- Stdin: rdev=%d %d\n",
            //  stat.Rdev&0xFF,stat.Rdev);
            //fmt.Printf("--I-- Stdin: tty%d\n",stat.Rdev&0xFF);
            return int(stat.Rdev & 0xFF)
        }
    }
    return 0
}
func (gshCtx *GshContext) ttyfile() string {
    //fmt.Printf("--I-- GSH_HOME=%s\n",gshCtx.GshHomeDir)
    ttyfile := gshCtx.GshHomeDir + "/" + "gsh-tty" +
         fmt.Sprintf("%02d",gshCtx.TerminalId)
         //strconv.Itoa(gshCtx.TerminalId)
    //fmt.Printf("--I-- ttyfile=%s\n",ttyfile)
    return ttyfile
```

```
}
func (gshCtx *GshContext) ttyline()(*os.File){
	file, err := os.OpenFile(gshCtx.ttyfile(),os.O_RDWR|os.O_CREATE|os.O_TRUNC,0600)
	if err != nil {
		fmt.Printf("--F-- cannot open %s (%s)\n",gshCtx.ttyfile(),err)
		return file;
	}
	return file
}
func (gshCtx *GshContext)getline(hix int, skipping bool, prevline string) (string) {
	if( skipping ){
		reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
		line, _, _ := reader.ReadLine()
		return string(line)
	}else
	if true {
		return xgetline(hix,prevline,gshCtx)
	}
	/*
	else
	if( with_exgetline && gshCtx.GetLine != "" ){
		//var xhix int64 = int64(hix); // cast
		newenv := os.Environ()
		newenv = append(newenv, "GSH_LINENO="+strconv.FormatInt(int64(hix),10) )

		tty := gshCtx.ttyline()
		tty.WriteString(prevline)
		Pa := os.ProcAttr {
			"", // start dir
			newenv, //os.Environ(),
			[]*os.File{os.Stdin,os.Stdout,os.Stderr,tty},
			nil,
		}
//fmt.Printf("--I-- getline=%s // %s\n",gsh_getlinev[0],gshCtx.GetLine)
proc, err := os.StartProcess(gsh_getlinev[0],[]string{"getline","getline"},&Pa)
		if err != nil {
			fmt.Printf("--F-- getline process error (%v)\n",err)
			// for ; ; { }
			return "exit (getline program failed)"
		}
		//stat, err := proc.Wait()
		proc.Wait()
		buff := make([]byte,LINESIZE)
		count, err := tty.Read(buff)
		//_, err = tty.Read(buff)
		//fmt.Printf("--D-- getline (%d)\n",count)
		if err != nil {
			if ! (count == 0) { // && err.String() == "EOF" ) {
				fmt.Printf("--E-- getline error (%s)\n",err)
			}
		}else{
			//fmt.Printf("--I-- getline OK \"%s\"\n",buff)
		}
		tty.Close()
		gline := string(buff[0:count])
		return gline
	}else
	*/
	{
		// if isatty {
			fmt.Printf("!%d",hix)
			fmt.Print(PROMPT)
		// }
		reader := bufio.NewReaderSize(os.Stdin,LINESIZE)
		line, _, _ := reader.ReadLine()
		return string(line)
	}
}

//== begin ======================================================= getline
/*
 * getline.c
 * 2020-0819 extracted from dog.c
 * getline.go
 * 2020-0822 ported to Go
 */
/*
package main // getline main
import (
	"fmt"       // <a href="https://golang.org/pkg/fmt/">fmt</a>
	"strings"   // <a href="https://golang.org/pkg/strings/">strings</a>
	"os"        // <a href="https://golang.org/pkg/os/">os</a>
	"syscall"   // <a href="https://golang.org/pkg/syscall/">syscall</a>
	//"bytes"       // <a href="https://golang.org/pkg/os/">os</a>
	//"os/exec" // <a href="https://golang.org/pkg/os/">os</a>
)
*/

// C language compatibility functions
var errno = 0
var stdin  *os.File = os.Stdin
var stdout *os.File = os.Stdout
var stderr *os.File = os.Stderr
var EOF = -1
var NULL = 0
type FILE os.File
type StrBuff []byte
var NULL_FP *os.File = nil
var NULLSP = 0
//var LINESIZE = 1024

func system(cmdstr string)(int){
	PA := syscall.ProcAttr {
		"", // the starting directory
		os.Environ(),
		[]uintptr{os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd()},
		nil,
	}
	argv := strings.Split(cmdstr," ")
	pid,err := syscall.ForkExec(argv[0],argv,&PA)
	if( err != nil ){
		fmt.Printf("--E-- syscall(%v) err(%v)\n",cmdstr,err)
	}
	syscall.Wait4(pid,nil,0,nil)

	/*
	argv := strings.Split(cmdstr," ")
	fmt.Fprintf(os.Stderr,"--I-- system(%v)\n",argv)
	//cmd := exec.Command(argv[0:]...)
	cmd := exec.Command(argv[0],argv[1],argv[2])
	cmd.Stdin = strings.NewReader("output of system")
	var out bytes.Buffer
	cmd.Stdout = &out
	var serr bytes.Buffer
	cmd.Stderr = &serr
	err := cmd.Run()
	if err != nil {
		fmt.Fprintf(os.Stderr,"--E-- system(%v)err(%v)\n",argv,err)
		fmt.Printf("ERR:%s\n",serr.String())
	}else{
		fmt.Printf("%s",out.String())
	}
	*/
	return 0
}
func atoi(str string)(ret int){
	ret,err := fmt.Sscanf(str,"%d",ret)
	if err == nil {
		return ret
	}else{
		// should set errno
		return 0
	}
}
func getenv(name string)(string){
	val,got := os.LookupEnv(name)
	if got {
		return val
	}else{
		return "?"
	}
}
func strcpy(dst StrBuff, src string){
	var i int
	srcb := []byte(src)
	for i = 0; i < len(src) && srcb[i] != 0; i++ {
		dst[i] = srcb[i]
	}
	dst[i] = 0
}
func xstrcpy(dst StrBuff, src StrBuff){
	dst = src
}
func strcat(dst StrBuff, src StrBuff){
	dst = append(dst,src...)
}
func strdup(str StrBuff)(string){
	return string(str[0:strlen(str)])
}
func sstrlen(str string)(int){
	return len(str)
}
func strlen(str StrBuff)(int){
	var i int
	for i = 0; i < len(str) && str[i] != 0; i++ {
	}
	return i
}
func sizeof(data StrBuff)(int){
	return len(data)
}
func isatty(fd int)(ret int){
	return 1
}

func fopen(file string,mode string)(fp*os.File){
	if mode == "r" {
		fp,err := os.Open(file)
		if( err != nil ){
			fmt.Printf("--E-- fopen(%s,%s)=(%v)\n",file,mode,err)
			return NULL_FP;
		}
		return fp;
	}else{
		fp,err := os.OpenFile(file,os.O_RDWR|os.O_CREATE|os.O_TRUNC,0600)
		if( err != nil ){
			return NULL_FP;
		}
		return fp;
	}
}
func fclose(fp*os.File){
	fp.Close()
}
func fflush(fp *os.File)(int){
	return 0
}
func fgetc(fp*os.File)(int){
	var buf [1]byte
	_,err := fp.Read(buf[0:1])
	if( err != nil ){
		return EOF;
	}else{
		return int(buf[0])
	}
}
func sfgets(str*string, size int, fp*os.File)(int){
	buf := make(StrBuff,size)
	var ch int
	var i int
	for i = 0; i < len(buf)-1; i++ {
		ch = fgetc(fp)
		//fprintf(stderr,"--fgets %d/%d %X\n",i,len(buf),ch)
		if( ch == EOF ){
			break;
		}
		buf[i] = byte(ch);
		if( ch == '\n' ){
			break;
		}
	}
	buf[i] = 0
	//fprintf(stderr,"--fgets %d/%d (%s)\n",i,len(buf),buf[0:i])
	return i
}
func fgets(buf StrBuff, size int, fp*os.File)(int){
	var ch int
	var i int
	for i = 0; i < len(buf)-1; i++ {
		ch = fgetc(fp)
		//fprintf(stderr,"--fgets %d/%d %X\n",i,len(buf),ch)
		if( ch == EOF ){
			break;
		}
		buf[i] = byte(ch);
		if( ch == '\n' ){
			break;
		}
	}
	buf[i] = 0
```

```
    //fprintf(stderr,"--fgets %d/%d (%s)\n",i,len(buf),buf[0:i])
    return i
}
func fputc(ch int , fp*os.File)(int){
    var buf [1]byte
    buf[0] = byte(ch)
    fp.Write(buf[0:1])
    return 0
}
func fputs(buf StrBuff, fp*os.File)(int){
    fp.Write(buf)
    return 0
}
func xfputss(str string, fp*os.File)(int){
    return fputs([]byte(str),fp)
}
func sscanf(str StrBuff,fmts string, params ...interface{})(int){
    fmt.Sscanf(string(str[0:strlen(str)]),fmts,params...)
    return 0
}
func fprintf(fp*os.File,fmts string, params ...interface{})(int){
    fmt.Fprintf(fp,fmts,params...)
    return 0
}

// <a name="IME">Command Line IME</a>
//---------------------------------------------------------------------- MyIME
var MyIMEVER = "MyIME/0.0.2";
type RomKana struct {
    pat string;
    out string;
}
var dicents = 0
var romkana [1024]RomKana
func readDic()(int){
    var rk *os.File;
    var dic = "MyIME-dic.txt";
    //rk = fopen("romkana.txt","r");
    //rk = fopen("JK-JA-morse-dic.txt","r");
    rk = fopen(dic,"r");
    if( rk == NULL_FP ){
        if( true ){
            fprintf(stderr,"--%s-- Could not load %s\n",MyIMEVER,dic);
        }
        return -1;
    }
    if( true ){
        var di int;
        var line = make(StrBuff,1024);
        var pat string
        var out string
        for di = 0; di < 1024; di++ {
            if( fgets(line,sizeof(line),rk) == NULLSP ){
                break;
            }
            fmt.Sscanf(string(line[0:strlen(line)]),"%s %s",&pat,&out);
            //sscanf(line,"%s %[^\r\n]",&pat,&out);
            romkana[di].pat = pat;
            romkana[di].out = out;
            //fprintf(stderr,"--Dd- %-10s %s\n",pat,out)
        }
        dicents += di
        if( false ){
            fprintf(stderr,"--%s-- loaded romkana.txt [%d]\n",MyIMEVER,di);
            for di = 0; di < dicents; di++ {
                fprintf(stderr,
                    "%s %s\n",romkana[di].pat,romkana[di].out);
            }
        }
    }
    fclose(rk);

    //romkana[dicents].pat = "//ddump"
    //romkana[dicents].pat = "//ddump" // dump the dic. and clean the command input
    return 0;
}
func matchlen(stri string, pati string)(int){
    if strBegins(stri,pati) {
        return len(pati)
    }else{
        return 0
    }
}
func convs(src string)(string){
    var si int;
    var sx = len(src);
    var di int;
    var mi int;
    var dstb []byte

    for si = 0; si < sx; { // search max. match from the position
        if strBegins(src[si:],"%x/") {
            // %x/integer/ // s/a/b/
            ix := strings.Index(src[si+3:],"/")
            if 0 < ix {
                var iv int = 0
                //fmt.Sscanf(src[si+3:si+3+ix],"%d",&iv)
                fmt.Sscanf(src[si+3:si+3+ix],"%v",&iv)
                sval := fmt.Sprintf("%x",iv)
                bval := []byte(sval)
                dstb = append(dstb,bval...)
                si = si+3+ix+1
                continue
            }
        }
        if strBegins(src[si:],"%d/") {
            // %d/integer/ // s/a/b/
            ix := strings.Index(src[si+3:],"/")
            if 0 < ix {
                var iv int = 0
                fmt.Sscanf(src[si+3:si+3+ix],"%v",&iv)
                sval := fmt.Sprintf("%d",iv)
                bval := []byte(sval)
                dstb = append(dstb,bval...)
                si = si+3+ix+1
                continue
            }
        }
        var maxlen int = 0;
        var len int;
        mi = -1;
        for di = 0; di < dicents; di++ {
            len = matchlen(src[si:],romkana[di].pat);
            if( maxlen < len ){
                maxlen = len;
				mi = di;
			}
		}
		if( 0 < maxlen ){
			out := romkana[mi].out;
			dstb = append(dstb,[]byte(out)...);
			si += maxlen;
		}else{
			dstb = append(dstb,src[si])
			si += 1;
		}
	}
	return string(dstb)
}
func trans(src string)(int){
	dst := convs(src);
	xfputss(dst,stderr);
	return 0;
}

//---------------------------------------------------------------- LINEEDIT
// "?" at the top of the line means searching history

var GO_UP = 201
var GO_DOWN = 202
var GO_RIGHT = 203
var GO_LEFT = 204

func getesc(in *os.File)(int){
	var ch1 int
	var ch2 int
	ch1 = fgetc(in);
	ch2 = fgetc(in);
	if false {
		fprintf(stderr,"(%c/%X %c/%X)",ch1,ch1,ch2,ch2);
	}
	switch( ch1 ){
		case '[':
			switch( ch2 ){
				case 'A': return GO_UP; // ^
				case 'B': return GO_DOWN; // v
				case 'C': return GO_RIGHT; // >
				case 'D': return GO_LEFT; // <
			}
			break;
	}
	return 0;
}
func clearline(){
	var i int
	fprintf(stderr,"\r");
	for i = 0; i < 80; i++ {
		fputc(' ',os.Stderr);
	}
	fprintf(stderr,"\r");
}
var romkanmode bool;
var insertmode int;
func redraw(lno int,line string,right string){
	var bsi int
	var rlen int
	var romkanmark string

	if( romkanmode ){
		//romkanmark = " *";
	}else{
		romkanmark = "";
	}
	clearline();
	xfputss("\r",stderr);
	if( romkanmode ){
		fprintf(stderr,"[\343\201\202r]");
		//fprintf(stderr,"[R]");
	}
	fprintf(stderr,"!%d! ",lno);
	if( romkanmode ){
		trans(line);
		//fputs(romkanmark,stderr);
		trans(right);
	}else{
		xfputss(line,stderr);
		//fputs(romkanmark,stderr);
		xfputss(right,stderr);
	}
	if true { //romkanmode {
		fprintf(stderr,"\r")
		if romkanmode {
			fprintf(stderr,"[\343\201\202r]");
			fprintf(stderr,"!%d! ",lno);
			trans(line);
		}else{
			fprintf(stderr,"!%d! ",lno);
			xfputss(line,stderr);
		}
	}else{
		rlen = len(right) + len(romkanmark);
		if true {
			for bsi = 0; bsi < rlen; bsi++ {
				fputc('\b',stderr);
			}
		}
	}
}
func delHeadChar(str string)(rline string,head string){
	_,clen := utf8.DecodeRune([]byte(str))
	head = string(str[0:clen])
	return str[clen:],head
}
func delTailChar(str string)(rline string, last string){
	var i = 0
	var clen = 0
	for {
		_,siz := utf8.DecodeRune([]byte(str)[i:])
		if siz <= 0 { break }
		clen = siz
		i += siz
	}
	last = str[len(str)-clen:]
	return str[0:len(str)-clen],last
}

// 3> for output and history
// 4> for keylog?
// <a name="getline">Command Line Editor</a>
func xgetline(lno int, prevline string, gsh*GshContext)(string){
	lastlno := lno;
	line := ""
	right := ""

	//readDic();
	if( isatty(0) == 0 ){
		if( sfgets(&line,LINESIZE,stdin) == NULL ){
			line = "exit\n";
		}else{
		}
		goto EXIT_GOT;
	}
	if( true ){
		//var pts string;
		//pts = ptsname(0);
		//pts = ttyname(0);
		//fprintf(stderr,"--pts[0] = %s\n",pts?pts:"?");
	}
	if( false ){
		fprintf(stderr,"! ");
		fflush(stderr);
		sfgets(&line,LINESIZE,stdin);
	}else{
		var ch int;

		system("/bin/stty -echo -icanon");
		redraw(lno,line,right);
		line = ""
		right = ""
		pch := -1
		for {
			if( pch != -1 ){
				ch = pch
				pch = -1
			}else{
				ch = fgetc(stdin);
			}
			if( ch == 033 ){
				ch = getesc(stdin);
			}
			if( ch == '\\' ){
				fputc(ch,stderr)
				ch = fgetc(stdin)
				if( ch == 'j' || ch == 'J' ){
					readDic();
					romkanmode = !romkanmode;
					if( ch == 'J' ){
						fprintf(stderr,"J\r\n");
					}
					redraw(lno,line,right);
					continue
				}else
				if( ch == 'i' || ch == 'I' ){
					dst := convs(line+right);
					line = dst
					right = ""
					if( ch == 'I' ){
						fprintf(stderr,"I\r\n");
					}
					redraw(lno,line,right);
					continue
				}else{
					pch = ch
					ch = '\\'
				}
			}
			switch( ch ){
				case 0:
					continue;
				case GO_UP:
					if lno == 1 {
						continue
					}
					cmd,ok := gsh.cmdStringInHistory(lno-1)
					if ok {
						line = cmd
						right = ""
						lno = lno - 1
					}
					redraw(lno,line,right);
					continue
				case GO_DOWN:
					cmd,ok := gsh.cmdStringInHistory(lno+1)
					if ok {
						line = cmd
						right = ""
						lno = lno + 1
					}else{
						line = ""
						right = ""
						if lno == lastlno-1 {
							lno = lno + 1
						}
					}
					redraw(lno,line,right);
					continue
				case GO_LEFT:
					if 0 < len(line) {
						xline,tail := delTailChar(line)
						line = xline
						right = tail + right
					}
					redraw(lno,line,right);
					continue;
				case GO_RIGHT:
					if( 0 < len(right) && right[0] != 0 ){
						xright,head := delHeadChar(right)
						right = xright
						line += head
					}
					redraw(lno,line,right);
					continue;
				case EOF:
					goto EXIT;
				case 'R'-0x40: // replace
					dst := convs(line+right);
					line = dst
					right = ""
					redraw(lno,line,right);
					continue;
				case 'T'-0x40: // just show the result
					readDic();
					romkanmode = !romkanmode;
					redraw(lno,line,right);
					continue;
				case 'L'-0x40:
					redraw(lno,line,right);
					continue
				case 'K'-0x40:
					right = ""
					redraw(lno,line,right);
					continue
				case 'E'-0x40:
					line += right
					right = ""
					redraw(lno,line,right);
					continue
				case 'A'-0x40:
					right = line + right
					line = ""
					redraw(lno,line,right);
					continue
				case 'U'-0x40:
					line = ""
					right = ""
					clearline();
					redraw(lno,line,right);
					continue;
				case 0x7F: // DEL
					if( 0 < len(line) ){
						line,_ = delTailChar(line)
						redraw(lno,line,right);
					}
					continue;
				case 'H'-0x40:
					if( 0 < len(line) ){
						line,_ = delTailChar(line)
						redraw(lno,line,right);
					}
					continue;
			}
			if( ch == '\n' || ch == '\r' ){
				fputc(ch,stderr);
				break;
			}
			line += string(ch);
			redraw(lno,line,right);
		}
		EXIT:
		system("/bin/stty echo sane");
	}
	//fprintf(stderr,"\r\nLINE:%s\r\n",line);

EXIT_GOT:
	return line + right;
}

func getline_main(){
	line := xgetline(0,"",nil)
	fprintf(stderr,"%s\n",line);
/*
	dp = strpbrk(line,"\r\n");
	if( dp != NULL ){
		*dp = 0;
	}

	if( 0 ){
		fprintf(stderr,"\n(%d)\n",int(strlen(line)));
	}
	if( lseek(3,0,0) == 0 ){
		if( romkanmode ){
			var buf [8*1024]byte;
			convs(line,buff);
			strcpy(line,buff);
		}
		write(3,line,strlen(line));
		ftruncate(3,lseek(3,0,SEEK_CUR));
		//fprintf(stderr,"outsize=%d\n",(int)lseek(3,0,SEEK_END));
		lseek(3,0,SEEK_SET);
		close(3);
	}else{
		fprintf(stderr,"\r\ngotline: ");
		trans(line);
		//printf("%s\n",line);
		printf("\n");
	}
*/
}
//== end ========================================================= getline

//
// $USERHOME/.gsh/
//		gsh-rc.txt, or gsh-configure.txt
//			gsh-history.txt
//			gsh-aliases.txt // should be conditional?
//
func (gshCtx *GshContext)gshSetupHomedir()(bool) {
	homedir,found := userHomeDir()
	if !found {
		fmt.Printf("--E-- You have no UserHomeDir\n")
		return true
	}
	gshhome := homedir + "/" + GSH_HOME
	_, err2 := os.Stat(gshhome)
	if err2 != nil {
		err3 := os.Mkdir(gshhome,0700)
		if err3 != nil {
			fmt.Printf("--E-- Could not Create %s (%s)\n",
				gshhome,err3)
			return true
		}
		fmt.Printf("--I-- Created %s\n",gshhome)
	}
	gshCtx.GshHomeDir = gshhome
	return false
}
func setupGshContext()(GshContext,bool){
	gshPA := syscall.ProcAttr {
		"", // the staring directory
		os.Environ(), // environ[]
		[]uintptr{os.Stdin.Fd(),os.Stdout.Fd(),os.Stderr.Fd()},
		nil, // OS specific
	}
	cwd, _ := os.Getwd()
	gshCtx := GshContext {
		cwd, // StartDir
		"", // GetLine
		[]GChdirHistory { {cwd,time.Now(),0} }, // ChdirHistory
		gshPA,
		[]GCommandHistory{}, //something for invokation?
		GCommandHistory{}, // CmdCurrent
		false,
		[]int{},
		syscall.Rusage{},
		"", // GshHomeDir
		Ttyid(),
		false,
		false,
		[]PluginInfo{},
		[]string{},
		" ",
		"v",
		ValueStack{},
		GServer{"",""}, // LastServer
		"", // RSERV
		cwd, // RWD
		CheckSum{},
	}
	err := gshCtx.gshSetupHomedir()
	return gshCtx, err
}
func (gsh*GshContext)gshelllh(gline string)(bool){
	ghist := gsh.CmdCurrent
	ghist.WorkDir,_ = os.Getwd()
	ghist.WorkDirX = len(gsh.ChdirHistory)-1
	//fmt.Printf("--D--ChdirHistory(@%d)\n",len(gsh.ChdirHistory))
	ghist.StartAt = time.Now()
	rusagev1 := Getrusagev()
	gsh.CmdCurrent.FoundFile = []string{}
	fin := gsh.tgshelll(gline)
	rusagev2 := Getrusagev()
	ghist.Rusagev = RusageSubv(rusagev2,rusagev1)
	ghist.EndAt = time.Now()
	ghist.CmdLine = gline
	ghist.FoundFile = gsh.CmdCurrent.FoundFile

	/* record it but not show in list by default
	if len(gline) == 0 {
		continue
	}
	if gline == "hi" || gline == "history" { // don't record it
		continue
	}
	*/
	gsh.CommandHistory = append(gsh.CommandHistory, ghist)
	return fin
}
// <a name="main">Main loop</a>
func script(gshCtxGiven *GshContext) (_ GshContext) {
	gshCtxBuf,err0 := setupGshContext()
	if err0 {
		return gshCtxBuf;
	}
	gshCtx := &gshCtxBuf

	//fmt.Printf("--I-- GSH_HOME=%s\n",gshCtx.GshHomeDir)
	//resmap()

	/*
	if false {
		gsh_getlinev, with_exgetline :=
		     which("PATH",[]string{"which","gsh-getline","-s"})
		if with_exgetline {
			gsh_getlinev[0] = toFullpath(gsh_getlinev[0])
			gshCtx.GetLine = toFullpath(gsh_getlinev[0])
		}else{
		fmt.Printf("--W-- No gsh-getline found. Using internal getline.\n");
		}
	}
	*/

	ghist0 := gshCtx.CmdCurrent // something special, or gshrc script, or permanent history
	gshCtx.CommandHistory = append(gshCtx.CommandHistory,ghist0)

	prevline := ""
	skipping := false
	for hix := len(gshCtx.CommandHistory); ; {
		gline := gshCtx.getline(hix,skipping,prevline)
		if skipping {
			if strings.Index(gline,"fi") == 0 {
				fmt.Printf("fi\n");
				skipping = false;
			}else{
				//fmt.Printf("%s\n",gline);
			}
			continue
		}
		if strings.Index(gline,"if") == 0 {
			//fmt.Printf("--D-- if start: %s\n",gline);
			skipping = true;
			continue
		}
		if false {
			os.Stdout.Write([]byte("gotline:"))
			os.Stdout.Write([]byte(gline))
			os.Stdout.Write([]byte("\n"))
		}
		gline = strsubst(gshCtx,gline,true)
		if false {
			fmt.Printf("fmt.Printf %%v - %v\n",gline)
			fmt.Printf("fmt.Printf %%s - %s\n",gline)
			fmt.Printf("fmt.Printf %%x - %s\n",gline)
			fmt.Printf("fmt.Printf %%U - %s\n",gline)
			fmt.Printf("Stouut.Write -")
			os.Stdout.Write([]byte(gline))
			fmt.Printf("\n")
		}
		/*
		// should be cared in substitution ?
		if 0 < len(gline) && gline[0] == '!' {
			xgline, set, err := searchHistory(gshCtx,gline)
			if err {
				continue
			}
			if set {
				// set the line in command line editor
			}
			gline = xgline
		}
		*/
		fin := gshCtx.gshelllh(gline)
		if fin {
			break;
		}
		prevline = gline;
```

```
        hix++;
    }
    return *gshCtx
}
func main() {
    gshCtxBuf := GshContext{}
    gsh := &gshCtxBuf
    argv := os.Args
    if 1 < len(argv) {
        if isin("version",argv){
            gsh.showVersion(argv)
            return
        }
        comx := isinX("-c",argv)
        if 0 < comx {
            gshCtxBuf,err := setupGshContext()
            gsh := &gshCtxBuf
            if !err {
                gsh.gshellv(argv[comx+1:])
            }
            return
        }
    }
    if 1 < len(argv) && isin("-s",argv) {
    }else{
        gsh.showVersion(append(argv,[]string{"-l","-a"}...))
    }
    script(nil)
    //gshCtx := script(nil)
    //gshell(gshCtx,"time")
}
//</div></details>
//<details id="todo"><summary>Consideration</summary><div class="gsh-src">
// - inter gsh communication, possibly running in remote hosts -- to be remote shell
// - merged histories of multiple parallel gsh sessions
// - alias as a function or macro
// - instant alias end environ export to the permanent > ~/.gsh/gsh-alias and gsh-environ
// - retrieval PATH of files by its type
// - gsh as an IME with completion using history and file names as dictionaies
// - gsh a scheduler in precise time of within a millisecond
// - all commands have its subucomand after "---" symbol
// - filename expansion by "-find" command
// - history of ext code and output of each commoand
// - "script" output for each command by pty-tee or telnet-tee
// - $BUILTIN command in PATH to show the priority
// - "?" symbol in the command (not as in arguments) shows help request
// - searching command with wild card like: which ssh-*
// - longformat prompt after long idle time (should dismiss by BS)
// - customizing by building plugin and dynamically linking it
// - generating syntactic element like "if" by macro expansion (like CPP) >> alias
// - "!" symbol should be used for negation, don't wast it just for job control
// - don't put too long output to tty, record it into GSH_HOME/session-id/comand-id.log
// - making canonical form of command at the start adding quatation or white spaces
// - name(a,b,c) ... use "(" and ")" to show both delimiter and realm
// - name? or name! might be useful
// - htar format - packing directory contents into a single html file using data scheme
// - filepath substitution shold be done by each command, expecially in case of builtins
// - @N substition for the history of working directory, and @spec for more generic ones
// - @dir prefix to do the command at there, that means like (chdir @dir; command)
// - GSH_PATH for plugins
// - standard command output: list of data with name, size, resouce usage, modified time
// - generic sort key option -nm name, -sz size, -ru rusage, -ts start-time, -tm mod-time
//   -wc word-count, grep match line count, ...
// - standard command execution result: a list of string, -tm, -ts, -ru, -sz, ...
// - -tailf-filename like tail -f filename, repeat close and open before read
// - max. size and max. duration and timeout of (generated) data transfer
// - auto. numbering, aliasing, IME completion of file name (especially rm of quieer name)
// - IME "?" at the top of the command line means searching history
// - IME %d/0x10000/ %x/ffff/
// - IME ESC to go the edit mode like in vi, and use :command as :s/x/y/g to edit history
// - gsh in WebAssembly
// - gsh as a HTTP server of online-manual
//---END--- (^-^)/ITS more</div></details>
/*
<details id="references"><summary>References</summary><div class="gsh-src">
<p>
<a href="https://golang.org">The Go Programming Language</a>
<iframe src="https://golang.org" width="100%" height="300"></iframe>

<a href="https://developer.mozilla.org/ja/docs/Web">MDN web docs</a>
 <a href="https://developer.mozilla.org/ja/docs/Web/HTML/Element">HTML</a>
 CSS:
   <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors">Selectors</a>
   <a href="https://developer.mozilla.org/en-US/docs/Web/CSS/background-repeat">repeat</a>
 HTTP
 JavaScript:
 ...
</p>
</div></details>
<div id="gsh-footer" style="">Fin.</div>
<style id="gsh-style">
 #gsh {border-width:1;margin:0;padding:0;}
 #gsh {font-family:monospace,Courier New;color:#ddf;font-size:8px;}
 #gsh header{height:100px;}
 #xgsh header{height:100px;background-image:url(GShell-Logo00.png);}
 #gsh-menu{font-size:14pt;color:#f88;}
 #gsh-footer{height:100px;background-size:80px;background-repeat:no-repeat;}
 #gsh note{color:#000;font-size:10pt;}
 #gsh h2{color:#24a;font-family:Georgia;font-size:18pt;}
 #gsh details{color:#888;background-color:#aaa;font-family:monospace;}
 #gsh summary{font-size:16pt;color:#24a;background-color:#eef;height:30px;}
 #gsh pre{font-size:11pt;color:#223;background-color:#faffff;}
 #gsh a{color:#24a;}
 #gsh a[name]{color:#24a;font-size:16pt;}
 #gsh .gsh-src{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
 #gsh .gsh-src{background-color:#faffff;color:#223;}
 #gsh-src-src{spellcheck:false}
 #src-frame-textarea{white-space:pre;font-family:monospace,Courier New;font-size:11pt;}
 #src-frame-textarea{background-color:#faffff;color:#223;}
 @media print {
  #gsh pre{font-size:11pt !import;}
 }
</style>
<!--
// Logo image should be drawn by JavaScript from a meta-font.
// CSS seems not follow line-splitted URL
-->
<script id="gsh-run">
GshLogo="data:image/png;base64,\
```

```
iVBORw0KGgoAAAANSUhEUgAAAQEAAAB/CAYAAADvs3f4AAAAAXNSR0IArs4c6QAAAHhlWElm\
TU0AKgAAAAgABAEaAAUAAAABAAAAPgEbAAUAAAABAAAARgEoAAMAAAABAAIAAIdpAAQAAAAB\
AAAATgAAAAAAAABIAAAAAQAAAEgAAAABAAOgAQADAAAAAQABAACgAgAEAAAAAQAAAQGgAwAE\
AAAAAQAAAH8AAAAAYx1BhgAAAAlwSFlzAAALEwAACxMBAJqcGAAAF3RJREFUeAHtnQuUFNWZ\
x++t7ukZ3iCggO/jY6Osb8WgMzAvn7uG4+bISTR7YnQXdQPCkGj2aNwlD2MSlRkeUaPnoCdu\
4iuJx7jriYZ50DOGmF2VqIBEiSggCoiMMA+mu+vu//dx6/q\
fnXvdx8tBA8SIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES\
IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES\
IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAES\
IAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIAESIIFDl4A8dLP2\
2eXs9H9+ftSkSdHxsic2qqdE7YusS+1qaalKfnY5YsokMHwEPtdK4MQFz5UeExlbLYSaYUl5\
npDiLKXEZClFiRM53JSUaq9ScqcU6i+2kK3StuONy5reEGKJ7Qw7mOvKec2ToqOiZwoljhFS\
jbOVHCstMRb3USXEJ8hFu7DsdmFb2+xU4vWWFVXbBpMeZUlAE/hcKoGab66eKGOlNykh56PC\
HxH2VVBKoRKqh3qUeKi1YdaOfONJ56OkdI6w5BwomnOQlyPzi0N9DLmXpFK/60p2P/Piyovf\
N8mfM+/nJWNGnjw9KqOToLVGSFt2p2Ril1gn3ij0Vk7YsoWVMzEuVPfPlRKYdfOak2LRSB0q\
zrWocCOG6qEhvgRaCj/dktj3g7dXXH4gKN6aRS0zpYzerqS6RAoZDQqfk79SKTRXHu/e+9FN\
L66as88pU/PN1pNlTLQJKSc73dPXSr20ur7iiwPcC8QhbNnCyhUIlryyOTQvYF5JfvqBL7jx\
+cNHjBj5gJRyDlJHy39o84D40H2Qtx8THaPeFuIOU+w1C+KnyhK5FGEv0WGgAExB83eXMoLY\
rikbd9gHEP52VgQl4h89FUA6kJyYFbbQbnzLJg4zFiesnDHCwvUoeiVQOb/5C9FY9DlUueOH\
+zGhUh9nSqOqrm0uWgurkI9RpjBD4Y6uQcQdD5TUOW63zD3MHesy14V49isbdKyxbGHlCpFR\
UJ6toACF7F9VF58NBfDHT0MBaE74Ent+eWrrWr+Lz/QTw60AdB7QJUjps/OA7cOoBNBCeMUZ\
ttCu/coG28fLpvKElTPFV8juRasEahbHvxaR1guoeBPyfUDo4+OfeBdyb8L4tz9XeSXFAMOc\
bgGgov0g1zgGGw4jF392xnHhdc+Mwf3JTjfntZ2yC1YJBJXNUt5KIKyck1sxXRdld6BmcevN\
aJovy/VBacMevqEP46/ZlnJjt9jx17VL53Zl5Mtvap1QGlNHw5pQDqXyNTQlZ2b8nGcMG2ZV\
qOoFjSdYvV0AZzDfayidv6FJ35CS4jXZk9hir7e27zm6p3T8hLJpkYicJpV1HtK/DJFU4Jw1\
lImhxM5IR9fzzgRKx4w/C+HQSPE+krbIyrN3qEPTNahsHaLDs2xh5Q5NCoPPVdEpgcqbm/8e\
7/zdOaHptag/mlKJ77U0VG0xybTdX/Ex/PTfa/i7r7Ku+cSoiCxUwrohUxF16wEV9H+ccVgl\
pd/CfU42AK2IUPlvTK1L/sJjyE5PVHqr728NzvfUzvvDODGy9GoopuuhmNLNfcTx48YHL2qH\
f/8hpXVu/43rQg9xtq6YtcvlXDC3fmWDQn9nbf2le7wKE1bOK65icBu0Eqhd3IaW82dwKPUw\
hrauc6ZcWdkcjUZK8EUXMae71zUqwCu2nbi6eVn1Ji9/P7eW+ioMAogF+NI3iJLSf8dn9ipA\
WNW4rPy9jJxuPeDL/HXzNzgTsveslD2vsWHWI9mu5rvVvZX9foS4v/LfmqdEIpHDGlfM2uCW\
gJIy2wOENPaZ3fEcivd+ZYNCNJYtrNyhyGAo8jRoJTAUmRiqOCJnRW5FpTN++frTwdh4SiUv\
bVlWvbffLcRF04qazRD7176/rBjKylD5pBiZ5Wi4wQu7tikPBeCOpuW+Kj0sqP8GHNoAZuwL\
iOzuywDhQ9zBr2xoDRqVQFi5QxxH6OwVjRKAAW46pvT+RxAJVLjW7vY9/+CeUBMkl68/rPQn\
mCufKzaldFN/yI8gA5iwC3dkIKhsyvZuCYSVG/KHcwhFWDRKAMMcD8EKX+rHFl2A9bt2d172\
2qNzOvzCDYmfEtNy7QogXDXWIKAIQ7cOQZchyADWnerqN5xVXttcJsdGp2OtwqmWJU7A+Eh7\
yhYbUgm1IX7f7K1DwaRyUfN42FIuxNDdVEtamL6sYC9R26VtbZaW2px8Nfmehz3EM+mgsolk\
d3/ZnBGElXPGUWzXg1YCq5eW5/zBGy54aWOgwWKfnWbqptcevWT4FUBvov32gew8DLzDTMaj\
aupq7t/bMXX+yw/egJGKoTksy2d+gFBb9VoDvX5BlZTOR+Wfjyb0pP6U0XGOYNqR/quta3vB\
Fgeua6qv2d7vn8dFdV3rldBw34GSPg9i0DG9h5XWknh9kAaMmyJ6dklPzZmtD3cnu77vtw5C\
h/YrG1p7Wxp/VvuRDuc+wsq54ymm+8zzKOgyRSPRa4IKoGz1i8b6ytagcEPmb9v/m09cUATz\
Jow6tVnPcMxHzj+sNNpHsCJyja6csrRsMyrGkiwF4I5UiouliL1RW7fmNLeX3z2+/GfW1LU2\
Y572b6EAzkfYoPctJi15Q1nJyLdrFrUZp1/3pmkuG/yN9gAoGyMTf7neVIvx/6CHUghlluh/\
f9Uvo+gG7O3q7rzFL8xQ+zW+/8F6PW6fV7xSXhiNlayvWdz2X1ULm/4uL1mPwNoA5uGcdoL9\
ZFa6cgoxzhTG6Q4lNR5Doj9xuvlcy+rFbcujVsnLkKv0CefphUbICLRMvl+9KP4vngHg6Fc2\
NCqMSiCsnCkfxeD+mTflBwuxdmFbOZqT/l94225Y3TCrzpQWhthG2zHraJO/yb0kkdhpanZq\
GXwFf66/8Cb5AHcbzdpnhUjeG6YFow1gZeMmtqNCDekzTiXVuc3LK4yVTJepuq5tqSWFkXdA\
ufu9MfWiG3sqnNtcX76+3xEXQWWzVeqSpvrZmC2afYSVy46l+O4KvyVgicCugG2rp0yPTveJ\
o2Ulm2JWZEO+f6K0dFtNXfw2U9x7O/bqZct5z0Poi0+vdpyDJcdxrD34U9XCeHrloSktt3ug\
AcwtkOO9FZFn+gWtWdS6ODcFoDrAxneOCfRXWUSoK93pBZXN7vAe+gwr506/2O4LXgngLbrC\
76HgRdvetHz2WlMYVVqqm5zTTP5+7volRR/zJlOYlx+8ohOzEb+CV/0TU5ic3NGfjkSs30MZ\
tFUtil+Yi4yfAcwkjzqpZyb6HlgJebwpgLYxoO9/j8k//WW3xS32gQPHrV5aMTp1IDFN2Op6\
fz5ywF4HfmXD+/Buy4NVu73yEFbOK65icot+ZjP+8qf4JkYiTnGKTb/qST0zMKACq18jjPGL\
A4PCxYNpMKOtjREv84HpyOsws/BsqyT2RGZ6rzl0gA9sBhEp46hsP2ratmOJeGrugBWDB2Pw\
NYD1B4OSTMBmcmdS2E/GG2ZvrF7Uejsqyw/7A7guEH6Kyyl9q3fpQQvgXtx4dz+Ueg+Lmy5v\
bjjYtO+b5LSqpq5Nz6nwbFFhUdaYgemZy4ap1z5dlbByA3NQTC4F3RKYfOTkaUF9Xry0LwU8\
sDMC/H29oV0GTNV1C+iZhTu27rgAebkb4+8H3P553qOOyu/WHj21ZWbd7z2XLuv4fA1gmQSV\
2GML+6KmhorvaQWgne11yZ/glLX+IBNcn2FQ7F9Y5XQfN/qUa+Hr3UrAGg1MTLrG3bfPyEtp\
m6d5oyCZcJmzX9nQ2jAqgbBymXSL9VzQSgBfxUBjHpbXbzM+vKueRBRiotE/Bw8ogf/LIZhY\
/9Tcnsb68lt7DtgnQRE8lEvT2z9eWT5SjF7lFSZoVlyfTLvqUTOb62etccbRO1lHeS68SYeT\
2OzUdegWmRTW7S7ng7dKrVi9rLztoMPBK73nA4YrdZfM+5DZsymDymaHnClokvPOVHG5FrQS\
wCY6RwU9Dkx5MU9wQXMaX+ePguLw8/dvfg6U1LPvsPBpXspOniQwagElsm9gqNxctOEQlvj5\
7tBBBjAdHkMPdY0/q/irW1bf44t5cNKQKwAq7DsuJzHl6Clz8bk+1u2u78FXYWfklQ4/qY2x\
tYvjX8boyWN6zwc9/Ojwz7pUtv1Lp0NQ2UxLo8PKOdMu1uvooTDjLyxcrNWHEhjQWsyKrkPs\
2JHl4LpJicQXoyp6nMs5fYsKeile0G95+WXcEj3m5mcmjNe5b+lyHZYELxGjRmDnY/HtMK0S\
aPE7Md34PueUYz8DWDovSjzXVF/xsFe+Lpz/wjQQ9eiH94ZWqVS62+CUhV3lMtNjSHfXorHf\
wKgZg9FwIrTCRJwjWh5+/ocSLzQ1zG52BvItG+wOpqXRYeWcaRfrdbSgC5bD/PySxBHakPWO\
qZx9y4L10uABB4xk5we8qDsHO6++b0nwjzFXYaUViy6Ece0O1I7SAZkxOUgxtmZB9RcaVyxx\
2CbMBjAdTcruWWyKriwy4myTH9zt3R93/8Xlj0ESWetyy7qFIj1odwkAmhFEA2KD6DlwNe6h\
H52HuWwIaLQHQOUYZwr6yznTLs7rgu4OYBJq4JBWJCayRhTyeYx4X8/xCw+rus9L5yc50A+W\
8v0w0N2ZxAw7VADPZcEDpXpdsLXoDKefrwEM+yj47aEAa7yxzMjXm+61FzUL46ch7cOd6Q/m\
Wncf9BTvXbs6Z3hNxPIvm1kJhJUbTFkKRbaglQCWiwbuiiPtyKlhHwZaq8YKoeMcji9Iy9Ly\
Pwk79U/55Bk75fSXMchwhj79Y35xY7qu8YspvTbqSG+55hdjjn6YS6ErfyqVOL2xoeLrbmWj\
YwkqG5S2p1IOK5djzgs+2LB1B4Z6/gG+uosa6yuWOYljzcCuoG4llqxVQOYep1wu1xUL4pPR\
zD3GL6wlVE4jA35xePk1NlSuBb/34RcwB6JXGgz6rflBBjBbJH7tlWbGDRVdb4bieXgpPbhN\
NQT3iqMHz7ETHvuRxnv45r8FpfQWRnDiqVfV2qBlxEFl6+rqDLV82CTnVYBidBs2JfBpwMJP\
aW3rXYbqm9qXMLnmChjCnvUN5fKMRc2LbzJBk8mU55cn4x/2rLdJQzNjtKkyuuO1pdqccfMz\
gKGp/aHfXooVi+JTofimZuJyn8F7QHmhAMxdAaUeTX6c7F07sUUkgyq5Oz33vV/Z0C7b+scH\
LtnpltH3YeW84ipGt4JWAnu7Pn5xwqjxB4IMabBc3Q8rfLzPCJfTc0SF0b8NaDzSFWqYfhBU\
nm1djITHGhN3eSRt+42Mk5KWcTsxFMe35RJTvorP3rmn49VMOgfP8oiD19lX6IdvbXmkqjvb\
NfydX9m8WimZlMLKZeSL/VzQSkDPzcdYcyte7lq/B4XKfKQaNeK3mL47r29fQL/gaT+/vrEO\
gDTTX0U9UWbKUVMfh9MYuLZjVPzxxu0fPO0/pTedhOd/1XXxGZawfuXp6eGIlz+eme2X9lbo\
0xuUl19F0bLaKGgQhafa5NVPhxjK7X0gLuOMRm+JAFefsnnaKzLRhZXLyBf5ediUwKc1/wD7\
fD+JL72vEtDPEIqgWkZj6zFP/d5duzt+ZHihxfkLnhs7umT0l1AjKkyVScenpJ1WAlAACzAE\
dqV2Sx/S+nLN0dPelXVtD/SkUr+JL5/9VsbL75z+bYNS8Q2EuQN/Oa3x1/FJZS/VZ30EGcBg\
ePdtCYCR0RCKr3q6vL0pOf7XfXvDAaVzcGjQECZX56CyYcmxZ/7CyuWar2IIN2xK4NOC075/\
4yMTRk3XuwyfGJgmxt/xdbpt8uSRi7Fl1luoFJtQm3Ul7cKXfyqMVsfDvwpVq9RPAeh07FRv\
hUL4693pwu1YyN+FX0C+Cy0VrIWXzylh/w3n7fiibreUtTsVURMitjpKWRYmPKkZmHDzFciM\
dMflf6+eWl0/65lMmCDD2YFEl2dFycgj38aRAbQSPGX1sCGUcCaKrDOUyszauvgcZx6zAvTf\
LLGqFlXPjFjyIthCkphR+cN+r76LoLJ1d3d45i+snDv9Yr4veCWg9+SrXtx6G/arezLXB4WX\
tgzv7Wk4n+Z8f/FFzzUKIa3ky5ULmo9CE8N3HgLinI5IsRNy32hsXxoRnTBmBvWmiP9zT7o3\
j0q8vnN35zecGfY1gCm1w2/fviCjoJXytieolL0xvRGhMyNZ1/IJtL6Ww3j5y8j+7i1dyU57\
xLjDJmM+xOFQgtrucgEUTDVIpFcnovWAf2KAEvArG5T3tjBGQT+5rCIU+U1BzxPIPJumpRVP\
4YEuz9wP9xlfvw/0ppuyxDp9uNPyih9l/XNXovNSd5dGG8C8wms31CzfrkCQUTCZSHj+wm8q\
JV7XE3xM6WqjLSr6LVB668ToEXtHjJ/4Cdw24+uzFvsJrsT11RkFoOOALtznFZdf2SDl2QrQ\
8YSV88pDsboVhRLQD6exvrEOj9y4g9DQPkC5Zmjjyz021LdV7yb3zfL8qmsDmOFARTVWFC3i\
N1NQGwX1jEavqOMrZ78D2ZVefmHcdPfCU86nbFBB5rKFlfPMRHE6Fo0S0AtoVm/d8VV8km7D\
C58YrseFuLvspLpbx79z64erdZNyuNLKileJdalUak7j0orr315x+YA9CbQBDF/ck7JkHDdB\
E5sg69OKMH9pdRJd6v3vgEvYbdQcucSlVM9nO/QaPP3KZlve8zWCmJjk3OkX+30RKQE8KiwN\
blxafhe29JqBL8of8GKam6n5P9mdGP5bmUikpmc22tR7BHSKjjP0kmCktCf/KAMlsOJXtejK\
v7q+/OzmZbN/Z5IoHT3+NPgZn2eyx7uiZOJDM9xoyTcZBTOya+vndqW3URPijYxbmDOe1/au\
zq4BrYqgslmphGdLIKxcmLwXskzBGwa94OstveB+sf714IiK3oiO5mXod+r9/I2VxB0P9Ec3\
xp7XQYu8JGTqmcatO+NeY/99v3xbh+21bh03cnotljdfCZnzkeapSDN/vjDg4XP4Cnb8+W9p\
9zzduKz2Q3fevO5lytqtomo30pzk9Ec5sHOY+FXfVl5Or6xr5HkDFMGAadKQ3yAO9DydFdjj\
ppf5kjNq6qrnYi3DfyKI5h14oOKj1aZehBJ9NtWTfBAGvv1uIawS2xVTahfsB5OdfrpseEaP\
mRiFlXOm8Xm4xnP/fBy6aVg2fty5SkWno2mMPfSF3sgCf3o4UGGSj/wI548wVLfbVvab7Z0b\
Xx/MrwGlf9ZrXPQMbMx5CiAfjiHIyXjhsR7BKkMfG8mLT+D3CdJF2qod1vNN3V3d60xW7hyf\
koSVf0pEpkZFeqJWQtld70c6dnp1H7zi0z933hOLHWYJu1REhZ7ptxeVe69XWH+3Jdasm6tO\
iEWsY1G5j8Eaj2NR0adga7IeVOR2LBSCcVC8Z0u5Ue1JbspxVqHEcusjRKkYLW0VSSUinTmW\
LaycfxHpSwIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIk\
QAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIk\
QAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIk\
QAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIk\
QAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIk\
QAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAIkQAJ5Evh/ikTb\
m38w0ncAAAAASUVORK5CYII=";

ITSmoreQR="data:image/png;base64,\
iVBORw0KGgoAAAANSUhEUgAAAG8AAABvAQMAAADYCwwjAAAABlBMVEX///9BaeFHqDaJAAAB\
HklEQVQ4jdXTsa2EMAwGYCMX7sICkVgjXVaCBe7CArASXdaIlAWgS4HwM5zEVS+mvSgS+ZBQ\
8gcb4BdHyzwv8szMSaUBHNm+KAd4QC8LDpDn8ogT4UpPGci2jI8IGFx3eLwPWaHknVyWecev\
UEbDXaB0X2aNjueYDOzNklQassPCkjc4nW3E1SfwqYk6jU/vAkPhg0AlSFhve8Jt0dkwDMwr\
yMGSSuPyWHAr19k0tkV2sb3sdW2rUCqW88g4Rp1A9s1JPv9cTp1NRD4XFkin8XaQCIwT6Lzq\
ZO8dHw/4+U2GzqlS8gbqVmkfr1N6YXK8OqlD0OmlGTMvzPERA8AL9vvbOifpSoL33fsVytrL\
S9wiqDzznhUI38v5n783/gBuUs2eLg1c8gAAAABJRU5ErkJggg==";
```

```
document.getElementById('banner').style.backgroundImage="url("+GshLogo+")";
//document.getElementById('gsh-footer').style.backgroundImage="url("+"QR-ITS-more.jp.png"+")";
document.getElementById('gsh-footer').style.backgroundImage="url("+ITSmoreQR+")";
//https://www.w3schools.com/JSREF/prop_style_backgroundposition.asp
var bannerStop = false
```

```
function shiftBG(){
 bannerStop = !bannerStop
 document.getElementById('banner').style.backgroundPosition = "0 0";
}

function html_fold(){
 document.getElementById('index').open=false
 document.getElementById('gsh-gocode').open=false
 document.getElementById('todo').open=false
 document.getElementById('reference').open=false
}
function html_open(){
 document.getElementById('index').open=true
 document.getElementById('gsh-gocode').open=true
 document.getElementById('todo').open=true
 document.getElementById('reference').open=true
}
function html_stop(){
 bannerStop = !bannerStop
}

//https://www.w3schools.com/jsref/met_win_setinterval.asp
function shiftBanner(){
 var now = new Date().getTime();
 //"console.log("now="+(now%10))
 if( !bannerStop ){
  document.getElementById('banner').style.backgroundPosition = ((now/10)%100000)+" 0";
 }
}
setInterval(shiftBanner,10);

// from embedded html to standalone page
function html_close(){
    window.close()
}

// from embedded html to standalone page
function html_new(){
    newwin = window.open("","","");
    src = document.getElementById("gsh");
    newwin.document.write("/*<"+"html>\n");
    newwin.document.write("<"+"span id=\"gsh\">");
    newwin.document.write(src.innerHTML);
    newwin.document.write("<"+"/span><"+"/html>\n"); // gsh span
    newwin.document.close();
    newwin.focus();
}

// source code viewr
function frame_close(){
    srcframe = document.getElementById("src-frame");
    srcframe.innterHTML = "";
    //srcframe.style.cols = 1;
    srcframe.style.rows = 1;
    srcframe.style.height = 0;
    srcframe.style.display = false;
    src = document.getElementById("src-frame-textarea");
    src.innerHML = ""
    //src.cols = 0
    src.rows = 0
    src.display = false
    //alert("--closed--")
}
//<!-- | <span onclick="html_view();">Source</span> -->
//<!-- | <span onclick="frame_close();">SourceClose</span> -->
//<!--| <span>Download</span> -->
function frame_open(){
    oldsrc = document.getElementById("GENSRC");
    if( oldsrc != null ){
        //alert("--I--(erasing old text)")
        oldsrc.innterHTML = "";
        return
    }else{
        //alert("--I--(no old text)")
    }
    banner = document.getElementById('banner').style.backgroundImage;
    footer = document.getElementById('gsh-footer').style.backgroundImage;
    document.getElementById('banner').style.backgroundImage = "";
    document.getElementById('banner').style.backgroundPosition = "";
    document.getElementById('gsh-footer').style.backgroundImage = "";

    src = document.getElementById("gsh");
    srcframe = document.getElementById("src-frame");
    srcframe.innerHTML = ""
     + "<"+"cite id=\"GENSRC\">\n"
     + "<"+"style>\n"
     + "#GENSRC textarea{tab-size:4;}\n"
     + "#GENSRC textarea{-o-tab-size:4;}\n"
     + "#GENSRC textarea{-moz-tab-size:4;}\n"
     + "#GENSRC textarea{spellcheck:false;}\n"
     + "</"+"style>\n"
     + "<h2>\n"
     //+ "<"+"span onclick=\"frame_close();\">Close</"+"span>\n"
     //+ " | <"+"span onclick=\"html_stop();\">Run</"+"span>\n"
     + "</h2>\n"
     + "<"+"textarea id=\"src-frame-textarea\" cols=100 rows=40>"
     + "/*<"+"html>\n"
     + "<"+"span id=\"gsh\">"
     + src.innerHTML
     + "<"+"/span><"+"/html>\n"
     + "</"+"textarea>\n"
     + "</"+"cite><!-- GENSRC -->\n";

    //srcframe.style.cols = 80;
    //srcframe.style.rows = 80;

    document.getElementById('banner').style.backgroundImage = banner;
    document.getElementById('gsh-footer').style.backgroundImage = footer
}
function html_view(){
    html_stop();

    banner = document.getElementById('banner').style.backgroundImage;
    footer = document.getElementById('gsh-footer').style.backgroundImage;
    document.getElementById('banner').style.backgroundImage = "";
    document.getElementById('banner').style.backgroundPosition = "";
    document.getElementById('gsh-footer').style.backgroundImage = "";

    //srcwin = window.open("","CodeView2","");
    srcwin = window.open("","","");
    srcwin.document.write("<span id=\"gsh\">\n");

    src = document.getElementById("gsh");
    srcwin.document.write("<style>\n");
    srcwin.document.write("textarea{tab-size:4;}\n");
    srcwin.document.write("textarea{-o-tab-size:4;}\n");
    srcwin.document.write("textarea{-moz-tab-size:4;}\n");
    srcwin.document.write("</style>\n");
    srcwin.document.write("<h2>\n");
    srcwin.document.write("<"+"span onclick=\"window.close();\">Close</span> | \n");
    //srcwin.document.write("<"+"span onclick=\"html_stop();\">Run</span>\n");
    srcwin.document.write("</h2>\n");
    srcwin.document.write("<textarea id=\"gsh-src-src\" cols=100 rows=60>");
    srcwin.document.write("/*<"+"html>\n");
    srcwin.document.write("<"+"span id=\"gsh\">");
    srcwin.document.write(src.innerHTML);
    srcwin.document.write("<"+"/span><"+"/html>\n");
    srcwin.document.write("</"+"textarea>\n");

    document.getElementById('banner').style.backgroundImage = banner;
    document.getElementById('gsh-footer').style.backgroundImage = footer

    sty = document.getElementById("gsh-style");
    srcwin.document.write("<"+"style>\n");
    srcwin.document.write(sty.innerHTML);
    srcwin.document.write("<"+"/style>\n");

    run = document.getElementById("gsh-run");
    srcwin.document.write("<"+"script>\n");
    srcwin.document.write(run.innerHTML);
    srcwin.document.write("<"+"/script>\n");

    srcwin.document.write("<"+"/span><"+"/html>\n"); // gsh span
    srcwin.document.close();
    srcwin.focus();
}
</script>
-->
*/ //</span></html>

```