1.0.23 JOD Source Code

https://github.com/bakerjd99/jod/tree/master/jodijs

John D. Baker

January 27, 2023

Contents

JOD Overview

JOD (J Object Dictionary) is a J addon.

See the following for details:

1. The JOD Page. `https://analyzethedatanotthedrivel.org/the-jod-page/`
2. The JOD manual `jod.pdf`. `https://github.com/bakerjd99/joddoc/blob/master/jod.pdf`

JOD User Interface Words

Some of the interface words listed here are not documented in the JOD manual. Hey, sprinkling source code with “undocumented features” and “Easter Eggs” for attentive readers is a longstanding software development *enticement*.

`abv`	[21]	*all backup version names*
`addgrp`	[286]	*add words/tests to group/suite*
`allnames`	[290]	*all names from uses: allnames 31 uses 'name'*
`allrefs`	[290]	*all nonlocale name references: allrefs ;:'return my references'*
`bget`	[24]	*retrieves objects from put dictionary backups*
`bnl`	[29]	*list objects in put dictionary database backup files*
`compj`	[253]	*compresses nonnouns by removing white space and shortening local identifiers*
`de`	[258]	*display JOD result without return code*
`del`	[43]	*deletes objects in dictionary database files*
`delgrp`	[294]	*remove words/tests from groups/suites*
`did`	[45]	*dictionary identification and statistics*
`disp`	[259]	*display dictionary objects as text*

dnl	[46]	*list objects in dictionary database files*
doc	[260]	*formats document text using the conventions of the (docct) verb*
dpset	[47]	*set dictionary parameters*
ed	[266]	*edit dictionary objects*
et	[268]	*edit text*
fsen	[297]	*first document sentence*
gdeps	[52]	*group and suite dependents*
get	[54]	*retrieves objects from dictionary database files*
getrx	[297]	*get required to execute*
globs	[58]	*analyze, report and store global names*
grp	[59]	*create and modify groups*
gt	[270]	*get J script text from J temp directory*
hlpnl	[298]	*displays short descriptions of objects on (y)*
jodage	[299]	*days since last change and creation of JOD objects*
jodhelp	[271]	*display PDF JOD help*
lg	[300]	*make and load JOD group*
locgrp	[301]	*list groups and suites with name*
make	[65]	*makes J scripts*
mls	[302]	*make load script*
mnl	[68]	*list objects in all registered dictionaries*
newd	[70]	*creates a new dictionary*
noexp	[305]	*returns a list of objects with no explanations*
notgrp	[306]	*words or tests from (y) that are not in groups or suites*
nt	[307]	*edit a new test script using JOD conventions*
nw	[308]	*edit a new explicit word using JOD conventions*
obnames	[309]	*object/locale names from uses: allnames 31 uses 'name'*
od	[72]	*opens and closes dictionaries*

`packd`	[74]	*backs up and recovers wasted space in dictionary files*
`pr`	[310]	*put and cross reference word*
`put`	[75]	*stores objects in dictionary database files*
`refnames`	[310]	*referenced nonlocale names from uses: allnames 31 uses 'name'*
`regd`	[80]	*register and unregister JOD dictionaries*
`restd`	[82]	*restores the most recent backup created by (packd)*
`revo`	[275]	*recently revised objects*
`revonex`	[310]	*returns a list of put dictionary objects with no explanations*
`rm`	[276]	*runs J macro scripts*
`rtt`	[277]	*runs J test scripts*
`rxs`	[83]	*regular expression search*
`swex`	[311]	*extract single line explanation from word header comment and save*
`usedby`	[314]	*returns a list of words from (y) that DIRECTLY call words on (x)*
`uses`	[91]	*returns word references*

jodon Source Code

```
NB.*jodon s-- places (jodon) and (jodoff) in z locale.

cocurrent 'z'

NB.*end-header


jodoff=: 3 : 0

NB.*jodoff v-- turns JOD off  result is  1  if  successful and  0
NB. otherwise.
NB.
NB. Destroys dictionary objects, clears JOD classes and drops the
NB. (ijod) interface. This  verb plus (jodon) and (jodsystempath)
NB. remain in the (z) locale after off'ing JOD and can be used to
NB. reload the system.
NB.
NB. monad:  jodoff uuIgnore

NB. HARDCODE: JODobj_ijod_ ajod ijod base
try.
  jo=. <'JODobj_ijod_'
  if. 0 = (4!:0) jo do. (4!:55) jo [ (18!:55) destroyjod__JODobj 0 end.

  NB. erase jod classes
  (18!:55) w #~ 'ajod'&-:@:(4&{.)&> w=. 18!:1 ] 0
  NB. erase (ijod) interface and clear base path
  ((18!:2<'base')-.<'ijod') 18!:2 <'base'
  (18!:55)<'ijod'

  _1=(4!:0) jo
catchd.
   0
end.
)

jodon=: 3 : 0

NB.*jodon v--  turn  JOD on  result  is 1  if  successful  and  0
NB. otherwise.
NB.
NB. Tests the  current J environment and creates or activates JOD
NB. objects.
NB.
NB. monad:  paRc =. jodon uuIgnore

NB. format of (9!:14) has changed without warning in the past
jvn=. 9!:14 ''

NB. first value before '/' is the version number (we hope).
jvn=. , (jvn i. '/') {. jvn
if. #jvn do. jvn=. 0 ". jvn #~ jvn e. '0123456789' else. jvn=. 0 end.
```

```
NB. allow older system to run but nag the user to upgrade
if. jvn < 801 do.
  0 0 $ (1!:2&2) 'WARNING: JOD works best with current J 8.0x-9.0x systems - upgrade!'
end.

sp=. ] [ 1!:2&2
if. jvn < 602 do.
  NB. J's before 6.02 are beyond the pale
  msg=. 'JOD requires J 8.01 or later.'
  msg=. msg,LF, 'J is freely available at www.jsoftware.com'
  0 [ sp msg,LF, 'Download and install J 8.0x-9.0x and then reinstall JOD.'
  return.
end.

nc=. (4!:0)@<
ex=. (4!:55)@<

NB. spot check of J environment - we need core J utilities
NB. if the following are not present JOD will not work
if. _1 e. (4!:0);:'load conew coclass coerase coinsert cocurrent copath jpath UNAME IFWIN' do.
  msg=. 'JOD depends on core J load and class utilities.'
  0 [ sp msg=. msg,LF,'Load J with a standard profile to use JOD.'
  return.
end.

NB. HARDCODE: JODobj_ijod_ ijod ajod
jodob=. nc 'JODobj_ijod_'          NB. name class of JOD object pointer
jodco=. (<'ajod') e. 18!:1 ] 0     NB. JOD class status

if. (0=jodob) *. jodco      do. 1  NB. JOD is loaded
elseif. (_1=jodob) *. jodco do.
  NB. jod is off and classes are loaded - create objects !(*)=. conew
  JODobj_ijod_=:  conew 'ajod'
  if. jodcs=. createjod__JODobj JODobj_ijod_ do. 1 else. 0 [ ex 'JODobj_ijod_' end.
elseif. -. jodco do.
  NB. JOD classes absent load and start system !(*)=. load
  ex 'JODobj_ijod_'
  NB. JOD load now requires addon path
  load 'general/jod'
  0 = nc 'JODobj_ijod_'
elseif.do. 0 NB. utterly screwed up system state
end.
)
```

jod Source Code

```
NB. *jod c-- main JOD dictionary class.
NB.
NB. All other dictionary classes are extensions of the dictionary class.
NB. They all use standard constants and verbs defined in this class.
NB.
NB. Creating a JOD object defines a (ijod) locale interface.
NB. Destroying a JOD object erases the (ijod) locale interface.
NB.
NB. Contains: dictionary utilities, constants, interface verbs
NB.
NB. Interface: (verbs made available by ijod locale)
NB.   abv     all backup version names
NB.   bget    get objects from put dictionary backups
NB.   bnl     backup name lists from patterns
NB.   del     delete words, tests, groups, macros, et cetera
NB.   did     dictionary identification
NB.   dnl     dictionary name lists from patterns
NB.   dpset   sets dictionary parameters
NB.   gdeps   list group and suite dependents
NB.   get     get words, tests, macros, et cetera from dictionary
NB.   globs   word and test global name references
NB.   grp     create and query groups and suites
NB.   make    generate J scripts and database dumps
NB.   mnl     many dictionary name lists from patterns
NB.   newd    create new dictionary
NB.   od       opens and closes dictionaries
NB.   packd    pack dictionaries
NB.   put      put words, tests, macros, et cetera into dictionary
NB.   regd     register/unregister a dictionary
NB.   restd    restore last backup created by (packd)
NB.   rxs      regular expression search
NB.   uses     words used by words and tests
NB.
NB. Notes:
NB.   Error messages (JOD errors 000-049)

coclass  'ajod'
coinsert 'ijod'

NB. task addon loaded first for J 9.01
require 'jfiles regex'

NB.*dependents x-- words defined in this section have related definitions

NB. host specific z locale nouns set during J profile loading
NB. (*)=: IFWIN UNAME IFIOS

NB. line feed, carriage return, tab and line ends
LF=: 10{a.
CR=: 13{a.
TAB=: 9{a.
CRLF=: CR,LF
```

```
NB. macro script option codes - to add more add a new object code
NB. and modify the following definition of MACROTYPE
JSCRIPT=: 21
LATEX=: 22
HTML=: 23
XML=: 24
TEXT=: 25
BYTE=: 26
MARKDOWN=: 27
UTF8=: 28
PYTHON=: 29
SQL=: 30
JSON=: 31

NB. macro text types
MACROTYPE=: JSCRIPT,LATEX,HTML,XML,TEXT,BYTE,MARKDOWN,UTF8,PYTHON,SQL,JSON

NB. object codes
WORD=: 0
TEST=: 1
GROUP=: 2
SUITE=: 3
MACRO=: 4

NB. dictionary self reference
DICTIONARY=: 5
```

```
NB. object name class, depends: WORD,TEST,GROUP,SUITE,MACRO
OBJECTNC=: WORD,TEST,GROUP,SUITE,MACRO

NB. bad object code, depends: OBJECTNC
badobj=: [: -. [: *./ [: , ] e. OBJECTNC"_

NB. path delimiter character & path punctuation characters
PATHDEL=:  IFWIN { '/\'
PATHCHRS=: ' :.-',PATHDEL

NB. path verbs - embed /\ chars depending on host OS
hostsep=:  (IFWIN{'/\')&(((IFWIN{'\/') I.@:= ])})

NB. extracts only the path from qualified file names
justpath=: [: }: ] #~ ([: -. [: +./\. ':'&=) *. [: +./\. PATHDEL&=

NB. default master profile user locations
NB. jodsystempath is left global here as this
NB. verb is defined in jodon.ijs
JMASTER=: jodsystempath 'jmaster'
JODPROF=: jodsystempath 'jodprofile.ijs'
JODUSER=: jodsystempath 'joduserconfig.ijs'

NB.*enddependents

NB.*end-header
NB. valid characters in file and path names
ALPHA=: 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'

NB. master file cn: dictionary number log - see long documentation
CNMFDLOG=: 10

NB. master file cn: in use bit
CNMFMARK=: 0

NB. master file cn: dictionary parameter defaults
CNMFPARMDEFS=: 9

NB. master file cn: dictionary parameters - see long documentation
CNMFPARMS=: 7

NB. master file cn: main dictionary table - see long documentation
CNMFTAB=: 2

NB. master file cn: main dictionary table backup
CNMFTABBCK=: 3

NB. default option code
DEFAULT=: 7

NB. comment tag marking end of dependents section
DEPENDENTSEND=: 'enddependents'
```

```
NB. comment tag marking start of dependents section
DEPENDENTSSTART=: 'dependents'

NB. numeral characters
DIGITS=: '0123456789'

NB. document option code
DOCUMENT=: 9

NB. controls dependent block processing - (1) process (0) do not process
DODEPENDENTS=: 1

NB. dictionary path table - see long documentation
DPATH=: 0 4$00

NB. maximum dictionary path length
DPLIMIT=: 32

ERR001=: 'invalid option(s)'

ERR002=: 'invalid name(s)'

ERR003=: 'name(s) to long'

ERR004=: 'invalid or missing locale'
```

```
ERR005=: 'invalid or missing dictionary name(s)'

ERR006=: 'cannot read master'

ERR007=: 'cannot read master documentation'

ERR008=: 'invalid names(s) - embedded locale references'

ERR009=: 'no documentation text for ->'

ERR010=: 'invalid name pattern(s)'

ERR011=: 'error(s) creating dictionary master file'

ERR012=: 'master in use - wait or try (dpset)'

ERR013=: 'cannot mark master'

ERR014=: 'invalid name and text'

ERR015=: 'invalid name, class and text'

ERR016=: 'definition failure among ->'

ERR017=: 'jfile replace error'
```

```
ERR018=: 'dictionary in use - cannot unregister'

ERR019=: 'invalid parameter or value'

ERR020=: 'table name(s) are not unique'

ERR021=: 'dll error generating GUID'

ERR022=: 'JOD z interface clashes with current z locale names. JOD load aborted'

ERR023=: 'white space preservation is off - turn on to put'

ERR024=: 'dependent section unbalanced'

ERR025=: 'only one balanced dependent section allowed'

ERR026=: 'error in joduserconfig.ijs - last J error ->'

ERR027=: 'unable to set master parameters ->'

ERR028=: 'not supported on this environment ->'

ERR029=: 'regex pattern error ->'

ERR030=: 'binary version conflict - dictionary -> '
```

```
NB. explain option code
EXPLAIN=: 8

NB. space in bytes required to create dictionary (0 turns off volume sizing)
FREESPACE=: 0

NB. group and suite header code
HEADER=: 1

NB. database file extension (it's changed in the past)
IJF=: '.ijf'

NB. J script file extension
IJS=: '.ijs'

NB. inverted data code: name classes and macro types
INCLASS=: 12

NB. inverted data code: object creation time
INCREATE=: 13

NB. inverted data code: last object put time
INPUT=: 14

NB. inverted data code: object size in bytes
INSIZE=: 15
```

```
NB. core JOD interface - loaded into (ijod) - see (setjodinterface)
IzJODinterface=: <;._1 ' abv bnl bget del did dnl dpset gdeps get globs grp make mnl newd od packd put regd
>..> restd rxs uses'


NB. standard dictionary file names - order matters
JDFILES=: <;._1 ' jwords jtests jgroups jsuites jmacros juses'

NB. standard dictionary subdirectory names - order matters
JDSDIRS=: <;._1 ' script suite document dump alien backup'

JEPOCHVER=: 9.0399999999999915

NB. default JOD user directory
JJODDIR=: 'joddicts\'

NB. regular expression matching valid J names
JNAME=: '[[:alpha:]][[:alnum:]_]*'

NB. version, make and date
JODVMD=: '1.0.23';30;'26 Jan 2023 15:15:15'

NB. base J version - prior versions not supported by JOD
JVERSION=: ,6.0199999999999957
```

```
NB. default master file parameters
MASTERPARMS=: 6 3$'PUTFACTOR';'(+integer) words stored in one loop pass';100;'GETFACTOR';'(+integer) words
>..>retrieved in one loop pass (<2048)';250;'COPYFACTOR';'(+integer) components copied in one loop pass';100;'
>..>DUMPFACTOR';'(+integer) objects dumped in one loop pass (<240)';50;'DOCUMENTWIDTH';'(+integer) width of ju
>..>stified document text';61;'WWWBROWSER';'(character) browser command line - used for jod help';' "C:\Progra
>..>m Files\Internet Explorer\IEXPLORE.EXE"'


NB. maximum length of short explanation text
MAXEXPLAIN=: 80

NB. maximum length of dictionary names
MAXNAME=: 128

NB. (name,[class],value) option code
NVTABLE=: 10

NB. successful return
OK=: 1;1

OK001=: 'dictionary unregistered ->'

OK002=: ' is a noun - no references'

OK003=: 'defaults restored for ->'
```

```
OK004=: 'master file reset'

OK005=: 'path cleared ->'

OK006=: 'parameter set ->'

OK007=: 'put dictionary is now a read/only library ->'

OK008=: 'put dictionary read/write status restored ->'

OK009=: 'put dictionary references deleted ->'

NB. indexes of dictionary subdirectories in dictionary parameter list
PARMDIRS=: 4 5 6 7 8 9

NB. parameter file - extension is required
PARMFILE=: 'jodparms.ijs'

NB. displayed path delimiter character
PATHSHOWDEL=: '/'

NB. search pattern option codes
PATOPS=: 1 2 3 _1 _2 _3

NB. controls whether words are saved when whitespace is discarded
PUTBLACK=: 0
```

```
NB. reference option code
REFERENCE=: 11

NB. maximum number of words per locale
SYMBOLLIM=: 2048

NB. uses union option code
UNION=: 31

abv=: 3 : 0

NB.*abv v-- all backup version names.
NB.
NB. Returns all  valid backup  names  matching  name prefix  (y).
NB. Names are listed from most recent backups to older backups.
NB.
NB. monad:  (paRc ; blclBNames) =. abv zl|clPfx
NB.
NB.   abv 'ch'  NB. all words in all backups starting with 'ch'
NB.   abv ''    NB. all words in all backups
NB.
NB. dyad:   (paRc ; blclBNames) =. il abv zl|clPfx
NB.
NB.   2 abv 'jod'  NB. all group names in all backups starting with 'jod'
NB.   4 abv ''     NB. all macros in all backups

0 abv y NB. word default
```

```
:
if. badcl y do. jderr ERR002 return. end. NB. errmsg: invalid name(s)
if. (1 < #,x) +. badil x do. jderr ERR001 return. end. NB. errmsg: invalid option(s)
if. -.isempty y do. if. badrc uv=. checknames y do. uv return. else. y=. rv uv end. end.
if. badrc uv=. x bnl '.' do. uv return. else. bn=. }.uv end.

NB. names matching prefix in all backups
pfx=. (<a:) -.&.>~ }.@(x&bnl)&.> (<y) ,&.> bn
b=. 0 < #&> pfx

NB. return backup names from most recent to older backups
ok \:~ ;<"1@;"1&.> (b # pfx) ,"0&.> <"0 b # bn
)


NB. retains string after first occurrence of (x)
afterstr=: ] }.~ #@[ + 1&(i.~)@([ E. ])


NB. trims all leading and trailing blanks
alltrim=: ] #~ [: -. [: (*./\. +. *./\) ' '&=


NB. test for jfile append errors
badappend=: 0: > {.


badblia=: 4 : 0

NB.*badblia v-- returns 0 if (y) is a boxed list of integer atoms
NB. or singleton codes from (x)
```

```
if. _1 -: dat=. , (; :: _1:) y do. 1
elseif. (#y) ~: #dat do. 1
elseif. badil dat do. 1
elseif.do. -. *./ dat e. x
end.
)

NB. 1 if (y) is not boxed
badbu=: [: 32&~: 3!:0

NB. 1 if (y) is not a character list or atom
badcl=: -.@(2&=@(3!:0)) (+.) 1: < [: # $

NB. 1 if (y) is not floating
badfl=: [: (-.) 8"_ = 3!:0

NB. 1 if (y) is not a list of non-extended integers
badil=: -.@((([: # $) (e.) 0 1"_) (*.) 3!:0 (e.) 1 4"_)

NB. bad jfile operation
badjr=: [: +./ _1 _2&e.

NB. bad locale name
badlocn=: [ >: [: 18!:0 ::(_2:) [: < ]
```

```
NB. bad return code
badrc=: [: (-.) 1: -: [: > {.

NB. test for jfile replacement errors
badreps=: 0: > <./

NB. 1 if any of shape, type or sign differ
badsts=: 0:

NB. 1 if items are not unique 0 otherwise
badunique=: # ~: [: # ~.

NB. retains string before first occurrence of (x)
beforestr=: ] {.~ 1&(i.~)@([ E. ])

bget=: 3 : 0

NB.*bget v-- retrieves objects from put dictionary backups.
NB.
NB. (bget) implements  a subset of (get). (bget) fetches  objects
NB. from either the last backup or particular backups.
NB.
NB. OBJECTS ARE NOT DEFINED IN LOCALES for the simple reason that
NB. backup fetches may return many versions of the same object.
NB.
NB. Only put  dictionary backups  are searched there is no backup
NB. path.  Also,  there is no corresponding  (bput)  because  the
NB. files  read  by (bget)  are backups that, once  created,  are
NB. never altered by JOD.
NB.
NB. Also,   certain  objects  are  not  fetched,  name   classes,
NB. timestamps and sizes.
NB.
NB. monad:  bget cl/blcl
NB.
NB.   NB. get last word backup
NB.   bget 'oops'
NB.
NB.   NB. collect from most current backup
NB.   bget ;: 'shawn of the dead'
NB.
NB.   NB. collect objects from particular put dictionary backups
NB.   bget <;._1 ' us.12 poor.10 little.08 words.08 lastback'
NB.
NB.   NB. get many versions of a word
NB.   bget <;._1 ' me.12 me.09 me.08 me.02'
NB.
NB. dyad:  ilCodes bget cl/bluu
NB.
NB.   5 bget ''     NB. dictionary document from last backup
NB.   5 bget '.12'  NB. dictionary document from particular backup
NB.   5 bget }. bnl '.'  NB. dictionary document versions in all backups
NB.
NB.   NB. get a suite header from particular backup
NB.   3 bget 'sweet.04'
NB.
NB.   NB. get long documents of an object
NB.   2 9 bget <;._1 ' gfoo.12 gfoo.05 gfoo.00'
NB.
NB.   NB. all short explanations of words in last backup
NB.   0 8 get }. revo ''
NB.
NB.   NB. three versions of a group's header - similar to (get) where
NB.   NB. (2 get 'group') returns the group header
NB.   2   bget <;._1 ' gfoo.12 gfoo.05 gfoo.00'
NB.
NB.   2 1 bget <;._1 ' gfoo.12 gfoo.05 gfoo.00'  NB. three versions of a group's word list

WORD bget y
:
msg=. ERR001

if. (2<#x) +. badil x do. jderr msg return. end.

NB. do we have a dictionary open?
if. badrc uv=. checkopen__ST 0 do. uv return. end.

NB. are backups present?
if. badrc uv=. checkback__ST {:0{DPATH__ST do. uv return. else. bn=. rv uv end.
```

```
NB. format standard (x) options
x=. x , (-2-#x) {. DEFAULT

NB. are backup names and numbers valid?
if. badrc bnm=. (({.x),bn) bchecknames__ST ,boxopen y do. bnm return. else. bnm=. rv bnm end.

select. {. x
case. WORD do.
  select. second x
    case. DEFAULT  do. (WORD,0) bgetobjects__ST bnm
    case. EXPLAIN  do. WORD bgetexplain__ST bnm
    case. DOCUMENT do. (WORD,1) bgetobjects__ST bnm
    case.do. jderr msg
  end.
case. TEST do.
  select. second x
    case. DEFAULT  do. (TEST,0) bgetobjects__ST bnm
    case. EXPLAIN  do. TEST bgetexplain__ST bnm
    case. DOCUMENT do. (TEST,1) bgetobjects__ST bnm
    case.do. jderr msg
  end.
case. GROUP do.
  select. second x
    case. HEADER   do. (GROUP,2) bgetobjects__ST bnm
    case. DEFAULT  do. GROUP bgetgstext__ST bnm
    case. EXPLAIN  do. GROUP bgetexplain__ST bnm
    case. DOCUMENT do. (GROUP,1) bgetobjects__ST bnm
    case.do. jderr msg
  end.
case. SUITE do.
  select. second x
    case. HEADER   do. (SUITE,2) bgetobjects__ST bnm
    case. DEFAULT  do. SUITE bgetgstext__ST bnm
    case. EXPLAIN  do. SUITE bgetexplain__ST bnm
    case. DOCUMENT do. (SUITE,1) bgetobjects__ST bnm
    case.do. jderr msg
  end.
case. MACRO do.
  select. second x
    case. DEFAULT  do. (MACRO,0) bgetobjects__ST bnm
    case. EXPLAIN  do. MACRO bgetexplain__ST bnm
    case. DOCUMENT do. (MACRO,1) bgetobjects__ST bnm
    case.do. jderr msg
  end.
case. DICTIONARY do.
  select. second x
    case. DEFAULT  do. bgetdicdoc__ST bnm
    case.do. jderr msg
  end.
case.do. jderr msg
end.
)

binverchk=: 3 : 0
```

```
NB.*binverchk v-- check binary compatibility of dictionary with directory object (y).
NB.
NB. monad:  blRc =. binverchk baObj

if. JEPOCHVER <: JVERSION do.
  if. JCREATEVER__y < JEPOCHVER do.
    (jderr ERR030),<'(',DNAME__y,') created with ',(":JCREATEVER__y),' rebuild as ',":JVERSION
    return.
  end.
end.
OK
)

bnl=: 3 : 0

NB.*bnl v-- list objects in put dictionary database backup files.
NB.
NB. monad:  dnl clStr | zlStr
NB.
NB.   bnl ''        NB. list all words in last backup
NB.   bnl '.'       NB. list backup suffixes
NB.   bnl 'pfx'     NB. list all words in last backup starting with 'pfx'
NB.   bnl 're.12'   NB. list all words in backup 12 starting with 're'
NB.
NB. dyad:  ilCodes bnl clStr | zlStr
NB.
NB.   4 2  bnl 'ex'     NB. macros with names containing 'ex' in last backup
NB.   2 3  bnl 'et.13'  NB. groups with names ending with 'et' in backup 13
NB.
NB.   14 bnl '.'  NB. display pack/backup dates

WORD bnl y
:
if. badrc msg=.x nlargs y do. msg return. end.

NB. format standard (bnl) (x) options and search
x=.  x , (<:#x)}. 1 , DEFAULT

NB. special list backup dates case first
if. (INPUT=0{x) *. (,NDOT__ST)-:alltrim y do. x bnlsearch__ST y return. end.

if. ((0{x) e. WORD,MACRO) *. -.(2{x) e. DEFAULT,MACROTYPE,i. 4 do. jderr ERR001
elseif. ({. x) e. OBJECTNC do. x bnlsearch__ST y
elseif.do. jderr ERR001
end.
)

NB. boxes open nouns
boxopen=: <^:(L. = 0:)

catrefs=: 3 : 0

NB.*catrefs v-- split into nonlocale and locale names.
NB.
NB. monad:  catrefs blcl
```

```
if. (,a:)-:,y do. ''
else.
  r=. islocref&> y  NB. insure 2 item result
  s=. <(-.r) # y
  l=. <r # y
  s,l
end.
)

NB. call dll
cd=: 15!:0

changestr=: 4 : 0

NB.*changestr v-- replaces substrings - see long documentation.
NB.
NB. dyad:  clReps changestr cl
NB.
NB.   NB. first character delimits replacements
NB.   '/change/becomes/me/ehh' changestr 'blah blah ...'

pairs=. 2 {."(1) _2 [\ <;._1 x      NB. change table
cnt=._1 [ lim=. # pairs
while. lim > cnt=.>:cnt do.         NB. process each change pair
  't c'=. cnt { pairs               NB. /target/change
  if. +./b=. t E. y do.             NB. next if no target
```

```
    r=. I. b                           NB. target starts
    'l q'=. #&> cnt { pairs            NB. lengths
    p=. r + 0,+/\(<:# r)$ d=. q - l NB. change starts
    s=. * d                            NB. reduce < and > to =
    if. s = _1 do.
      b=. 1 #~ # b
      b=. ((l * # r)$ 1 0 #~ q,l-q) (,r +/ i. l)} b
      y=. b # y
      if. q = 0 do. continue. end.  NB. next for deletions
    elseif. s = 1 do.
      y=. y #~ >: d r} b             NB. first target char replicated
    end.
    y=.(c $~ q *# r) (,p +/i. q)} y  NB. insert replacements
  end.
end. y                                 NB. altered string
)

checknames=: 3 : 0

NB.*checknames v-- tests alleged  boxed lists of J names. Accepts
NB. all valid J  names. When (x-:1)  names  with embedded locale
NB. references  are  rejected  otherwise  embedded   locales  are
NB. accepted.
NB.
NB. monad:  checknames cl/blcl
NB.
NB.   checknames 'we';'check';'out'
NB.
NB. dyad:  pa checknames cl/blcl
NB.
NB.   0 checknames ;:'accept our_poor_ locale__NAMES'

1 checknames y
:
msg=. ERR002  NB. errmsg: invalid name(s)
if. 1<#$ y        do. jderr msg return. end.
y=. ,&.> boxopen y   NB. allow char lists
if. +./ badcl&> y do. jderr msg return. end.

if. x do.
  NB. restrict embedded locales
  msg2=. ERR008  NB. errmsg: invalid names(s) - embedded locale references
  if. '_' e. , _1&{.&> y  do. jderr msg2 return. end.
  if. +./ +./@:('__'&E.)&> y do. jderr msg2 return. end.
  if. _2 e. nc y     do. jderr msg return. end.
else.
  NB. permit embedded locales - test must eschew class tests
  NB. to avoid evaluation of indirect locale references
  if. (#jnfrblcl y)~:#y do. jderr msg return. end.
end.

if. MAXNAME < >./ #&> y do. jderr ERR003 return. end. NB. errmsg: name(s) to long
ok trimnl y  NB. return deblanked name list
)

checknttab=: 3 : 0
```

```
NB.*checknttab v--  checks (name,text)  tables. A name text table
NB. is a two column boxed table. Column 0  contains valid  names.
NB. Column 1 contains character  lists representing various texts
NB. like J scripts, LaTeX or HTML code.
NB.
NB. monad:  checknttab btcl
NB.
NB.  checknttab (;:'n1 n2 n3') ,. 'blah blah..';'more ehh..';'stuff ...'

msg =. ERR014 NB. errmsg: invalid name and text
if. badbu y do. jderr msg
elseif. -. 1 2 e.~ #$ y do. jderr msg
elseif. 2 ~: {: $ y=. plt y do. jderr msg
elseif. +./badcl&> 1 {"1 y do. jderr msg
elseif. badrc uv=.checknames (<a:;0){y do. jderr msg
elseif. badunique uv=. }.uv do. jderr ERR020
elseif.do. ok <y=. uv (<a:;0)} y  NB. insures deblanked names
end.
)

checknttab2=: 4 : 0

NB.*checknttab2 v-- checks (name,class,text) tables.  Similar  to
NB. (checknttab)  except the  additional column is a numeric name
NB. class or type code.
NB.
NB. dyad:  ilCodes checknttab2 btcl
NB.
NB.  (i.4) checknttab2 'name';3;'verb=: ...'

msg=. ERR015 NB. errmsg: invalid name, class and text
if. badbu y do. jderr msg
elseif. -. 1 2 e.~ #$ y do. jderr msg
elseif. 3 ~: {: $ y=. plt y do. jderr msg
elseif. +./badcl&> {:"1 y do. jderr msg
elseif. x badblia 1 {"1 y do. jderr msg
elseif. badrc uv=.checknames (<a:;0){y do. jderr msg
elseif. badunique uv=. }.uv do. jderr ERR020
elseif.do. ok <y=. uv (<a:;0)} y  NB. insures deblanked names
end.
)

checknttab3=: 3 : 0

NB.*checknttab3 v-- checks all three (name,[class],text) tables.
NB.
NB. monad:  checknttab3 bt

if. 3 = cols=. {:$y do.
  NB. there are two species of three column tables - words
  NB. and macros - distingquished by the codes in column 1
  if. ((i. 4), MACROTYPE) badblia 1 {"1 y do. jderr ERR014
  NB. macro codes start at 21 much higher than J name class codes
  elseif. 3 < <./ ;1 {"1 y do.
    MACROTYPE checknttab2 y
  elseif. do.
    (i. 4) checknttab2 y
  end.
elseif. 2 = cols do.
  NB. two column tables
  checknttab y
elseif.do. jderr ERR014
end.
)

createjod=: 3 : 0

NB.*createjod  v--  dictionary  object  creation verb. (y)  is  a
NB. dictionary object locale reference. This verb initializes  an
NB. (ijod) locale user interface for the JOD system.  and creates
NB. all necessary subobjects.
NB.
NB. monad:  paRc =. createjod ba
NB.
NB.   JD=: conew 'ajod'
NB.   createjod__JD JD

NB. set default master, profile and user if they don't exist
if. -.wex <'JMASTER' do. JMASTER=: jodsystempath 'jmaster' end.
if. -.wex <'JODPROF' do. JODPROF=: jodsystempath 'jodprofile.ijs' end.
if. -.wex <'JODUSER' do. JODUSER=: jodsystempath 'joduserconfig.ijs' end.

NB. set J version number
```

```
JVERSION_ajod_=:  (jvn :: _9:) ''

NB. create master file if necessary
if. -. fex <JMASTER,IJF do.
  if. badrc mdat=. createmast JMASTER do. mdat return. end.
end.

NB. execute any user script - allows for redefintions of various
NB. class nouns before JOD objects are created - joduserconfig.ijs
NB. is not installed and must be created by users
if. fex <JODUSER do.
  NB. attempt execution of script - obfuscate names (/:)=:
  if. (_9 -: ((0!:0) :: _9:) <JODUSER){0 1 do. (jderr ERR026),<13!:12 '' return. end.
end.

NB. initialize master dictionary parameters - used when
NB. creating directory objects to insure that all master
NB. parameters are set in directory objects - this amends
NB. the "jod" class to exploit inheritance in all derived classes
if. badjr mdat=. jread JMASTER;CNMFPARMS do. jderr ERR006 return. end.
MASTERPARMS_ajod_=:  > mdat

NB. extension objects and complete (ijod) interface (*)=. JODEXT IZJODALL
JODEXT=:  0$a:
IZJODALL=:  IzJODinterface,<'JODobj'

NB. create storage, scratch, maker and utility objects !(*)=. JOD ST SO MK UT
```

```
JOD=:  y
ST=:  conew 'ajodstore'
MK=:  conew 'ajodmake'
UT=:  conew 'ajodutil'

NB. empty classless object - must see ijod
SO=:  cocreate ''
('ijod';'z') copath ;SO

obs=. JOD;ST;MK;UT;<SO

NB. initialize objects - they need to know each other
createst__ST obs
createmk__MK obs
createut__UT obs

NB. create direct _n_ (ijod) locale interface - if the (ijod)
NB. trap word (jodsf) exists define an error trapping interface
".&.> y defzface IzJODinterface

NB. attempt to create J temp directory ignoring errors
NB. required for JOD edit utilities and not always present on J systems
makedir <jpath '~temp/'

NB. execute any master dictionary profile script
if. fex <JODPROF do. (_9 -: ((0!:0) :: _9:) <JODPROF){1 0 else. 1 end.
)
```

```
createmast=: 3 : 0

NB.*createmast v-- creates the dictionary master file.
NB.
NB. The master  file holds the  master  dictionary directory  and
NB. dictionary parameters.  The master  file  tracks the state of
NB. dictionaries.  In  this  system  only  one task  can  open  a
NB. dictionary read/write. When opening a  dictionary  the master
NB. file is  checked  to determine if  the  dictionary  has  been
NB. opened read/write by another task. Dictionaries can be opened
NB. read/only by any number of tasks.
NB.
NB. monad:  createmast clFile
NB.
NB.   createmast_ajod_  JMASTER_ajod_  NB. recreate master

fn=.  hostsep y
if. IFWIN do.
  syp=. PATHDEL ,~ (justdrv , ':'"_ , justpath) fn
else.
  syp=. PATHDEL ,~ justpath fn
end.

if. badappend jcreate fn do.
  jderr ERR011 NB. errmsg: error(s) creating dictionary master file
  return.
end.
```

```
fn=. jopen_jfiles_ fn
cn=. (<0;now '') jappend fn     NB. c0 use bit and last change
'jodversion jodbuildcnt'=. 2{.JODVMD
cn=. cn, (<jodversion;jodbuildcnt,didnum 0) jappend fn  NB. c1 version, build #, unique id
cn=. cn, (4 0$'') jappend fn    NB. c2 dictionary directory
cn=. cn, (4 0$'') jappend fn    NB. c3 directory backup
cn=. cn, (3#<'') jappend fn     NB. c4,c5,c6 RESERVED

NB. parse parameter settings --- sets (MASTERPARMS)
try.
  0!:0 <syp,PARMFILE
  parms=. <dptable MASTERPARMS    NB. created by 0!:0 !(*)=. MASTERPARMS
catchd.
  jclose_jfiles_ fn
  (jderr ERR027),<syp,PARMFILE return.
end.

cn=. cn, parms jappend fn       NB. c7 active dictionary parameters
cn=. cn, parms jappend fn       NB. c8 active parameter backup
cn=. cn, parms jappend fn       NB. c9 default parameters
cn=. cn, (i.0) jappend fn       NB. c10 dictionary log
jclose_jfiles_ fn
if. 0 > <./cn do.
  jderr ERR011
else.
  ok {: cn  NB. return last component
end.
)

NB. character table to newline delimited list
ctl=: }.@(,@(1&(,"1)@(-.@(*./\."1@(=&' '@])))) # ,@((10{a.)&(,"1)@]))

NB. YYYYMMDD to YYYY MM DD - see long document
datefrnum=: 0 100 100&#:@<.

NB. enclose all character lists in blcl in " quotes
dblquote=: '"'&,@:(,&'"')&.>

decomm=: 3 : 0

NB.*decomm v--  removes comments  from j words. The (x) argument
NB. specifies whether all blank lines are removed or retained.
NB.
NB. monad:  decomm ctWord
NB.
NB.   decomm jcr 'decomm'  NB. decomment self
NB.
NB. dyad:  pa decomm ctWord
NB.
NB.   1 decomm jcr 'decomm'  NB. remove blanks (default)
NB.   0 decomm jcr 'decomm'  NB. retain all blank lines

1 decomm y
```

```
:
NB. mask of unquoted comment starts
c=. ($y)$'NB.' E. ,y
c=. +./\"1 c > ~:/\"1 y e. ''''

NB. ,, work around for j8.05 bug - remove when fixed
NB. y=. ,,y

NB. blank out comments
y=. ' ' (I. ,c)} ,y
y=. y $~ $c

NB. remove blank lines - default
if. x do. y #~ y +./ . ~: ' ' end.
)

defzface=: 4 : 0

NB.*defzface v-- define (ijod) interface from word list.
NB.
NB. dyad:  blcl =. clSuffix defzface blclWords

NB. if the top level error trap word exists
NB. define an error trapping interface
if. 3 = (4!:0) <'jodsf_ijod_' do.
  iface=. (y ,&.> locsfx x) ,&.> <' :: jodsf'
else.
  iface=. y ,&.> locsfx x
```

```
end.
(y ,&.> <'_ijod_=:') ,&.> iface
)

del=: 3 : 0

NB.*del v-- deletes objects in dictionary database  files. Result
NB. is  a return code and message. The deletion only modifies the
NB. object's directory. The  actual data  remains in the file  as
NB. "dead"  components  until a (packd)  operation reclaims  file
NB. space.
NB.
NB. monad:  del blclWords
NB.
NB.   del ;: 'we are toast'
NB.
NB. dyad:  iaObject del blclName
NB.
NB.   1 del 'toast these tests'

WORD del y
:
msg=. ERR001
if. badil x do. jderr msg return. end.

NB. do we have a put dictionary open?
if. badrc uv=. checkput__ST 0 do. uv return. end.
DL=. 1 { uv
```

```
select. x
case. WORD do.
  (WORD;INVWORDS__ST;<DL) delstuff__ST y
case. TEST do.
  (TEST;INVTESTS__ST;<DL) delstuff__ST y
case. GROUP do.
  (GROUP;INVGROUPS__ST;<DL) delstuff__ST y
case. SUITE do.
  (SUITE;INVSUITES__ST;<DL) delstuff__ST y
case. MACRO do.
  (MACRO;INVMACROS__ST;<DL) delstuff__ST y
case. REFERENCE do.
  if. badrc y=. checknames y do. y
  elseif. badrc msg=. DL delwordrefs__ST }. y do. msg
  elseif.do. (ok OK009),<DNAME__DL
  end.
case. do. jderr msg
end.
)

destroyjod=: 3 : 0

NB.*destroyjod v--  dictionary  object  destroy  verb.  This  verb
NB. erases the JOD (ijod) locale user interface.
NB.
NB. monad:  destroyjd uuIgnore
NB. close any open dictionaries
3 od ''

NB. erase current direct _n_ ijod locale references
NB. (*)=. IZJODALL JODEXT
(4!:55) IZJODALL ,&.> locsfx 'z'

NB. destroy sub-objects !(*)=. ST MK UT SO
coerase ST,MK,UT,SO

NB. destroy any JOD class extension objects
coerase JODEXT

NB. return self reference
18!:5 ''
)

did=: 3 : 0

NB.*did v-- dictionary identification and statistics
NB.
NB. monad:  did uuIgnore
NB. dyad:  uuIgnore did uuIgnore

if. badrc msg=. checkopen__ST 0 do. msg else. ok {."1 DPATH__ST end.
:
0 didstats__ST 0
)
```

```
didnum=: 3 : 0

NB.*didnum  v--  generates  a  unique extended  precision integer
NB. based GUID.  The GUID is designed to produce a unique  global
NB. identifier every time it's called.
NB.
NB. monad:  didnum uuIgnore

NB. Original Windows only code
NB. call dll to get GUID
NB. guid=. genguid <16#' '
NB. if. 0 ~: >{. guid do. jderr ERR021
NB. else.
NB.  NB. guid as 128 bit mask
NB.  guid=. , (a. i. >{: guid){ truth 8
NB.
NB.  NB. convert mask to an integer computing
NB.  NB. only required extended powers of 2
NB.  pos=. I. guid
NB.  +/(2x ^ pos) pos} guid
NB. end.

NB. More general Win/Linux/Mac code
guidsx i.0
)

dnl=: 3 : 0
NB.*dnl v-- list objects in dictionary database files.
NB.
NB. monad:  dnl clStr|zlStr
NB.
NB.   dnl ''        NB. list all words on path
NB.   dnl 'pfx'     NB. list all words on path begining with 'pfx'
NB.
NB. dyad:  ilCodes dnl clStr|zlStr
NB.
NB.   4 2  dnl 'ex'  NB. macros with names containing 'ex'
NB.   0 _3 dnl 'ugh' NB. path order listing of words ending with 'ugh'

WORD dnl y
:
if. badrc msg=.x nlargs y do. msg return. end.

NB. format standard (dnl) (x) options and search
x=.  x , (<:#x)}. 1 , DEFAULT
if. ({. x) e. OBJECTNC do. x dnlsearch__ST y else. jderr ERR001 end.
)

dpset=: 3 : 0

NB.*dpset v-- set dictionary parameters.
NB.
NB. monad:  dpset zl | clCommand | (cllParm ; uuValue)
NB. dyad:  iaCode dpset (clParm ; uuValue)
NB. objects !(*)=. DL ST
NB. allow mixed assignments (<:)=:

NB. resets should always work - close any open dictionaries
if. y -: 'RESETME' do.
  3 od '' NB. HARDCODE 3 close code
  if. badrc msg=. markmast~0 do. msg else. ok OK004 [ remast 1 end.
elseif. y -: 'RESETALL' do.
  3 od '' NB. HARDCODE 3 close code
  if. badrc msg=. markmast~0 do. msg else. ok OK004 [ remast 0 end.
elseif.do.
  NB. other options require an open dictionary
  if. badrc msg=.checkopen__ST 0 do. msg return. end.
  DL=.  {:{.DPATH__ST

  if. isempty y do.
    NB. display settable parameters of put/first with current values
    ok <|:>{:>jread WF__DL;CNPARMS__ST

  elseif. -.badcl y do.

    NB. cannot change dictionary parameters for older dictionaries
    NB. that are not fully binary compatible with J 9.04+
    if. badrc msg=. binverchk DL do. msg return. end.

    NB. if we are resetting READWRITE status dictionary need only be open
    if. 'READWRITE'-:y do.
  NB. check attributes of READONLY dictionary to insure
  NB. that it will allow read/write operations on all files
  dcfiles=. (WF__DL;TF__DL;GF__DL;SF__DL;MF__DL;UF__DL) ,&.> <IJF
  NB. err msg (JODstore errors): dictionary file attributes do not allow read/write
  if. 0 e. iswriteable__ST dcfiles do. (jderr ERR095__ST),<DNAME__DL return. end.

  if. badrc msg=.libstatus__DL 0 do. msg
  else.
    RW__DL=:  -. LIBSTATUS__DL=:  0  NB. library off/read write
    ok OK008;DNAME__DL
  end.
  return.
end.
NB. other changes of dictionary parameters require a put dictionary
if. badrc msg=. checkput__ST 0 do. msg return. end.
select. y
case. 'DEFAULTS' do.
  if. badjr dat=. jread JMASTER;CNMFPARMDEFS   do. jderr ERR088
  elseif. badjr dpt=. jread WF__DL;CNPARMS__ST do. jderr ERR088
  elseif. dpt=. <(}:>dpt),<|: 1 0 1#"1 dat=. >dat
          badreps dpt jreplace WF__DL;CNPARMS__ST  do. jderr ERR017
  elseif.do.
    NB. reset live object parameters
    (({."1 dat) ,&.> locsfx DL)=:  {:"1 dat
    ok OK003;DNAME__DL
  end.
```

```
    case. 'CLEARPATH' do.
      RPATH__DL=. i.0
      if. badreps (i.0) jreplace UF__DL;CNRPATH__ST do.
        jderr ERR017
      else.
        ok OK005;DNAME__DL
      end.
    case. 'READONLY' do.
      if. badrc msg=.libstatus__DL 1 do. msg
      else.
        RW__DL=:  -. LIBSTATUS__DL=:  1  NB. library on/read only
        ok OK007;DNAME__DL
      end.
    case.do. jderr ERR001
    end.

  elseif. -.badbu y do.
    NB. parameter changes only allowed for put dictionaries
    if. badrc msg=. checkput__ST 0 do. msg return. end.
    msg=. ERR019  NB. errmsg: invalid name/parameter
    if. -. (1=#$ y) *. 2=#y do. jderr msg return. end.
    if. badjr dpt=. jread WF__DL;CNPARMS__ST do. jderr ERR088 return. end.
    usp=. >{:dpt=. >dpt
    if. ({:$usp) = pos=. ({.usp) i. {.y do. jderr msg return. end.
    if. (>pos{{:usp) badsts >{:y do. jderr msg return. end.
    NB. reset live object
    ('__DL' ,~ >pos{{.usp)=:  >{:y
    dpt=. (}:dpt),<usp=. ({:y) (<1;pos)} usp
    if. badreps (<dpt) jreplace WF__DL;CNPARMS__ST do. jderr ERR017 else. ok OK006;y end.

  elseif.do. jderr ERR001
  end.
end.
)

dptable=: 3 : 0

NB.*dptable  v--  parses MASTERPARMS.
NB.
NB. (MASTERPARMS) is set by running the script (jodparms.ijs).
NB.
NB. monad:  dptable clParms
NB.
NB.   0!:0 <jpath '~addons\general\jodparms.ijs'
NB.   dptable__JODobj  MASTERPARMS

NB. parse MASTERPARMS table - remove J comments
y=. (<;._1)"1  ';' ,. decomm ];._2 y -. CR

NB. remove extra blanks
y=. (alltrim&.>(<a:;0 1){y) (<a:;0 1)} y

NB. handle parm types currently only (+integer) and (character)
NB. NIMP - there is no error checking for dictionary parameters
ptype=. > 1{"1 y
```

```
pint=. I. (,:'(+integer)') ({."1)@E. ptype

NB. character and other types left as is
NB. char=. I. (,:'(character)') ({."1)@E. ptype
y=. (".&.> (<pint;2){y) (<pint;2)} y
)

NB. 1 if empty dictionary name list 0 otherwise
empdnl=: (,<0$0) -: ]

NB. test boxed list of path\file names for existence (0 some bad, 1 all ok)
fex=: *./@:(1:@(1!:4) ::0:)

NB. 0's all but the first 1 in runs of 1's
firstone=: ] > [: }: 0: , ]

NB. first of doubles
fod=: ] #~ 1 0"_ $~ #

NB. first on path order list index - see long documentation
fopix=: 1: i.~ [ +/@:e.&> [: < [: < ]

gdeps=: 3 : 0

NB.*gdeps v-- group and suite dependents.
NB.
NB. Dependents are global J assignments between the dependents tags:
NB.
NB. verbatim:
NB.
NB.    NB.*dependents
NB.    NB.*enddependents
NB.
NB. monad:  gdeps clGroup
NB.
NB.   gdeps 'jod'
NB.
NB. dyad:   iaGScode gdeps clGroupSuite
NB.
NB.   3 gdeps

GROUP gdeps y
:
if. badil x     do. jderr ERR001 NB. errmsg: invalid options
elseif. badcl y do. jderr ERR002 NB. errmsg: invalid name(s)
elseif.  x=. {.x
         -. x e. GROUP,SUITE do. jderr ERR001
elseif.  badrc uv0=. (x,1) obtext__UT y do. uv0
elseif.do.
  uv0=. ,>2{uv0
  NB. hides tags from searches
  beg=. 'NB.*',DEPENDENTSSTART
  fin=. 'NB.*',DEPENDENTSEND
  tcnt=. (+/beg E. uv0),+/fin E. uv0
  select. tcnt
  case. 0 0 do. ok ''
  case. 0 1 do. jderr ERR024  NB.errmsg: dependent block unbalanced
  case. 1 0 do. jderr ERR024
  case. 1 1 do.
     uv0=. ];._1 LF,fin beforestr uv0 -. CR
     0 namecats__MK uv0 }.~ I. (,:beg) +./"1@E. uv0
  case.do.
    jderr ERR025 NB. errmsg: only one balanced dependent block allowed
  end.
end.
)

get=: 3 : 0

NB.*get v-- retrieves objects from dictionary database files.
NB.
NB. monad:  get blcl
NB.
NB.   get ;: 'us poor little words'
NB.
NB. dyad:  ilCodes get bluu
NB.
NB.   2 8 put 'GroupName';'Group documentation text ....'
NB.   2 8 get 'GroupName'
NB.   4 get 'MacroText'
```

```
WORD get y
:
msg=. ERR001 [ loc =. <'base' NB. errmsg: invalid option(s)

if. badil x do.
  NB. errmsg: invalid or missing locale
  if. _2&badlocn x do. jderr ERR004 return. else. x=. WORD [ loc=. <x-.' ' end.
end.

NB. do we have a dictionary open?
if. badrc uv=. checkopen__ST 0 do. uv return. end.

NB. format standard (x) options
x=. x , (-3-#x) {. DEFAULT , 0
if. -. 0 1 e.~ {: x do. jderr msg return. end.

select. {. x
case. WORD do.
  select. second x
    case. DEFAULT  do. loc defwords__ST y
    case. EXPLAIN  do. WORD getexplain__ST y
    case. DOCUMENT do. WORD getdocument__ST y
    case. NVTABLE  do. (WORD,0) getobjects__ST y
    case. INCLASS;INCREATE;INPUT;INSIZE do. (2{.x) invfetch__ST y
    case. -INPUT   do. WORD getntstamp__ST y
    case.do. jderr msg
  end.
```

```
case. TEST do.
  select. second x
    case. DEFAULT  do. (TEST,0) getobjects__ST y
    case. EXPLAIN  do. TEST getexplain__ST y
    case. DOCUMENT do. TEST getdocument__ST y
    case. INCREATE;INPUT;INSIZE do. (2{.x) invfetch__ST y
    case. -INPUT   do. TEST getntstamp__ST y
    case.do. jderr msg
  end.
case. GROUP do.
  select. second x
    case. DEFAULT  do. GROUP getgstext__ST y
    case. EXPLAIN  do. GROUP getexplain__ST y
    case. DOCUMENT do. GROUP getdocument__ST y
    case. INCREATE;INPUT do. (2{.x) invfetch__ST y
    case. -INPUT   do. GROUP getntstamp__ST y
    case.do. jderr msg
  end.
case. SUITE do.
  select. second x
    case. DEFAULT  do. SUITE getgstext__ST y
    case. EXPLAIN  do. SUITE getexplain__ST y
    case. DOCUMENT do. SUITE getdocument__ST y
    case. INCREATE;INPUT do. (2{.x) invfetch__ST y
    case. -INPUT   do. SUITE getntstamp__ST y
    case.do. jderr msg
  end.
```

```
case. MACRO do.
  select. second x
    case. DEFAULT  do. (MACRO,0) getobjects__ST y
    case. EXPLAIN  do. MACRO getexplain__ST y
    case. DOCUMENT do. MACRO getdocument__ST y
    case. INCLASS;INCREATE;INPUT;INSIZE do. (2{.x) invfetch__ST y
    case. -INPUT   do. MACRO getntstamp__ST y
    case.do. jderr msg
  end.
case. DICTIONARY do.
  select. second x
    case. DEFAULT  do. getdicdoc__ST 0
    case.do. jderr msg
  end.
case.do. jderr msg
end.
)

globals=: 4 : 0
if. badcl y do. jderr ERR002 return. end. NB. errmsg: invalid name(s)
if. badrc y=. 0 checknames y do. y
else.
  y =.>1{y
  NB. use base locale if no locale reference
  if. -.islocref y do. y=. y,'_base_' end.
  x wrdglobals__MK y
end.
)
```

```
globs=: 3 : 0

NB.*globs v-- analyze, report and store global names
NB.
NB. monad:  globs clName
NB.
NB.   globs 'word'    NB. list globals in locale word
NB.
NB. dyad:   iaCode globs clName
NB.
NB.   NB. stores global references of word in dictionary
NB.   0 globs 'word'
NB.
NB.   1 globs 'test'  NB. list globals in test

0 globals y
:
if. (,x)-:,REFERENCE do. 1 globals y
elseif. badcl y do. jderr ERR002 NB. errmsg: invalid name(s)
elseif.do.
  select. x
  case. WORD do.
    if. badrc uv=. checkput__ST 0  do. uv return. else. DL=. 1 { uv end.
    if. badrc uv=. binverchk DL do. uv return. end.
    if. badrc y=. checknames__ST y do. y return. else. y=. ,>}.y end.
    if. badrc uv=. (WORD;<DL) inputdict__ST <y  do. uv return. end.
    if. badrc uv=. WORD getobjects__ST y do. uv return. else. uv=. ,1 {:: uv end.
    if. 0=>1{uv do. ok '<',y,'>',OK002 return. end.
    if. badrc uv=. 0 namecats__MK ];._2 (>2{uv),LF do. uv return. end.
    (y;<DL) putwordxrs__ST }.uv
  case. TEST do.
    if. badrc uv=. TEST get y do. uv return. else. uv=. ,1 {:: uv end.
    NB. return references in stored test text
    0 namecats__MK ];._2 (>1{uv),LF
  case.do. jderr ERR001  NB. errmsg: invalid option(s)
  end.
end.
)

grp=: 3 : 0

NB.*grp v-- create and modify groups.
NB.
NB. monad:  grp blcl
NB. dyad: ia grp ?

GROUP grp y
:
select. x
  case. GROUP do. (GROUP,WORD) gsmakeq y
  case. SUITE do. (SUITE,TEST) gsmakeq y
  case.do. jderr ERR001 NB. errmsg: invalid option(s)
end.
)
```

```
gsmakeq=: 4 : 0

NB.*gsmakeq v-- make or query groups and suites.
NB.
NB. dyad:  ilCodes gsmakeq blcl

'gscode obcode'=. x

if. isempty y do. gscode dnl ''
else.
  if. badcl y do.

    NB. create/modify group
    if. badrc mdl=. checkput__ST 0  do. mdl return. end.
    if. badrc msg=. checkpath__ST {: mdl do. msg return. end.
    NB. remove empties from name list allows (grp 'name';'') to create null groups
    if. badrc y=.  checknames y -. a: do. y return. end.
    (({:mdl);obcode;gscode) putgs__ST }. y

  else.

    NB. query group contents
    if. badrc msg=. checkopen__ST 0 do. msg return. end.
    if. badrc y=. checknames y do. y return. end.
    gscode gslistnl__ST rv y
  end.
end.
```

```
)

guids=: 3 : 0

NB.*guids v-- create guids as 16 byte strings on supported J systems.
NB.
NB. This verb taken from ~addons/general/misc/guids.ijs returns guids
NB. on Windows, Linux and Mac systems.
NB.
NB. monad:  guids zl | ilShape
NB.
NB.   guids ''    NB. create guid as a 16-byte character string
NB.   guids $0
NB.   guids 3 4   NB. create 3x4 array of 16-byte strings

if. IFWIN do.
  cmd=. 'ole32 CoCreateGuid i *c'
else.
  cmd=. ((UNAME-:'Darwin'){::'libuuid.so.1';'libSystem.B.dylib'),' uuid_generate n *c'
end.
>{:"1 cmd 15!:0"1 0 <"1 (y,16)$' '
)

NB. guids as extended precision integers: guidsx i.0 [ guidsx 3 5
guidsx=: 256 #. [: x: a. i. guids

NB. returns result of linux/unix commands as text string
host=: [: 2!:0 '('"_ , ] , ' || true)'"_
```

```
NB. 1 if noun is empty on any axis and 0 otherwise
isempty=: 0: e. $

NB. 1 if name is a locale reference 0 otherwise
islocref=: ('_'"_ = {:) +. [: +./ '__'"_ E. ]

NB. error trapped call to jappend_jfiles_
jappend=: jappend_jfiles_ ::(_2:)

NB. character table representation of j words, call:  jcr 'verb'
jcr=: [: ];._1 (10{a.) , [: 5!:5 <

NB. error trapped call to jcreate_jfiles_
jcreate=: jcreate_jfiles_ ::0:

NB. format error message
jderr=: 0: ; '!JOD error: '"_ , ]

jdmasterr=: 3 : 0

NB.*jdmasterr v-- master error handling.
NB.
NB. Use when the master file is set otherwise the master will not
NB. be  properly reset. Because of the file overhead I decided to
NB. use a second error  handler  instead  of  burdening the  very
NB. frequently called  (jderr)  with this  often unnecessary file
NB. access function.

if. badrc msg=.markmast~0 do. msg else. jderr y end.
)

jnfrblcl=: 3 : 0

NB.*jnfrblcl  v-- extracts valid  J names  from  boxed  lists  of
NB. character lists. Only  proper  unquoted,  inflection free (no
NB. trailing .'s)  names are  returned. This  verb extracts names
NB. without using name class tests. Class tests cannot be used on
NB. indirect locale names, eg. (BOO__HOO)  as the noun (HOO) must
NB. exist.
NB.
NB. monad: jnfrblcl  blcl
NB.
NB.   jnfrblcl 'good';' '' bad';'888';'ok';'notok.';'3r7'

NB. trim end blanks and eliminate any empties
y=. y #~ 0 < #&> y=.alltrim&.> y

NB. remove all lists containing invalid name characters
y=. y #~ y *./@:e.&> <ALPHA,'_'

NB. remove all lists beginning with numerals and _
y=. y #~ -.({.&> y) e. '_0123456789'

NB. extract any remaing names with regular expression
```

```
if. #y do.
  NB. NOTE: workaround for J 9.04 PCRE2 changes
  NB. turn of utf8 support for J 9.04+ !(*)=. rxutf8
  if. b903=. 9.03 < jvn'' do. rgs=. rxutf8 0 end.
  ejn=. JNAME rxall ; y ,&.> ' '
  if. b903 do. rgs=. rxutf8 rgs end.
  ejn
else. ''
end.
)

NB. standarizes J path delimiter to unix/linux forward slash
jpathsep=: '/'&(('\' I.@:= ])} )

NB. error trapped call to jread_jfiles_
jread=: jread_jfiles_ ::(_2:)

NB. error trapped call to jreplace_jfiles_
jreplace=: jreplace_jfiles_ ::(_2:)

NB. extracts the drive from qualified file names
justdrv=: [: }: ] #~ [: +./\. ':'&=

jvn=: 3 : 0

NB.*jvn-- J version number.
NB.
NB. monad:  fa =. jvn uuIgnore
NB. dyad: fa =. cl jvn uuIgnore

(9!:14 '') jvn y
:
NB. for empty version strings return
NB. 0 we don't know the version
if. 0=#x do. 0
else.
  NB. extract J version number from (9!;14) string
  ver=. '0/' ,~ (x e. '0123456789/')#x

  NB. return version 0 if string is not numeric
  100 %~ , 0 ". (ver i. '/') {. ver
end.
)

NB. removes all leading and trailing CR and LF characters
lfcrtrim=: ] #~ [: -. [: (*./\. +. *./\) ] e. (10 13{a.)"_

NB. surround names with locale delimiters, eg: _name_
locsfx=: '_'&,@,&'_'&.>

make=: 3 : 0

NB.*make v-- makes J scripts.
NB.
NB. monad:  make zl|cl
NB.
NB.   make ''  NB. basic put dump
NB.
NB. dyad:  ilObjOpt make cl|blcl
NB.
NB.   0 make ;:'an arbitrary list of words into a script and file it'
NB.   0 2 make ;: 'a list of words returned as a character list'
NB.
NB.   3 make 'suite'    NB. make suite write to script subdirectory
NB.   2 2 make 'group'  NB. make group return character list
NB.
NB.   NB. make groups that are not in put dictionary
NB.   NB. file is written to put dictionary script directory
NB.   2 _1 make 'deepgroup'

makedump__MK y
:
msg=. ERR001  NB. errmsg: invalid option(s)
if. badil x do. jderr msg return. end.

NB. do we have a dictionary open?
if. badrc uv=. checkopen__ST 0 do. uv return. end.

NB. format standard (x) options HARDCODE
x=. 2 {. x , 1 2
```

```
if. -.({: x) e. _2 _1 1 2 do. jderr msg return. end.

if. ({. x) e. GROUP,SUITE do. x makegs__MK y
elseif.  ({. x)=WORD do.
  if. badrc uv=.WORD obtext__UT y do. uv
  elseif. 1={: x do. (WORD;1{uv) writeijs__MK >{:uv
  elseif.do. ok >{: uv
  end.
elseif.do. jderr msg
end.
)

NB. make a directory (1 success, 0 failure)
makedir=: 1!:5 ::0:


markmast=: 3 : 0

NB.*markmast v-- marks the master dictionary file. This system is
NB. is primarily  a single  writer  system. Many dictionary tasks
NB. can read data but only one  task can  change  it.  The master
NB. file  is used to enforce this  protocol. (markmast)  sets and
NB. unsets a use bit. When the bit is set the master file  itself
NB. cannot be changed.
NB.
NB. monad:  markmast uuIgnore
NB. dyad:   uuIgnore markmast uuIgnore
NB. set the use bit/timestamp in the master file
if. badjr ub=. jread JMASTER;CNMFMARK do. jderr ERR006  NB. errmsg: cannot read master
elseif. >{.>ub do. jderr ERR012  NB. errmsg: master in use - wait or try (dpset)
elseif. badreps (mubmark y) jreplace JMASTER;CNMFMARK do.
  jderr ERR013 NB. errmsg: cannot mark master
elseif.do. ok y
end.
:
NB. dyad resets the master
if. badreps (mubmark 0) jreplace JMASTER;CNMFMARK do. jderr ERR013 else. ok 0 end.
)

mnl=: 3 : 0

NB.*mnl v-- list objects in all registered dictionaries.
NB.
NB. monad:  mnl clStr | zlStr
NB.
NB.   mnl ''     NB. list all words in all registered dictionaries
NB.   mnl 'pfx'  NB. list all words in all registered dictionaries starting with 'pfx'
NB.
NB. dyad:  ilCodes mnl clStr | zlStr
NB.
NB.   4 2  mnl 'ex'  NB. macros with names containing 'ex' in all registered dictionaries
NB.   2 3  mnl 'et'  NB. groups with names ending with 'et' in all registered dictionaries
NB.   4 3 25 mnl '_sql' NB. text macros with names ending '_sql'
NB.   0 _1 mnl 'se'  NB. duplicate words starting with 'se'
```

```
WORD mnl y
:

NB. (mnl) does not require open dictionaries
if.     badcl y do. jderr ERR010  NB. errmsg: invalid name pattern
elseif. badil x do. jderr ERR001  NB. errmsg: invalid option(s)
elseif. do.

  NB. format standard (mnl) (x) options and search
  x=. 3 {. x , (<:#x)}. 1 , DEFAULT

  NB. validate options
  if. -.((1{x) e. PATOPS) *. (0{x) e. OBJECTNC do. jderr ERR001 return. end.


  if. WORD = 0{x do.

    if. -.(2{x) e. (i. 4),DEFAULT     do. jderr ERR001 return. end.

  elseif. (0{x) e. TEST,GROUP,SUITE do.

    if. DEFAULT ~: 2{x                do. jderr ERR001 return. end.

  elseif. MACRO = 0{x  do.

    if. -.(2{x) e. MACROTYPE,DEFAULT  do. jderr ERR001 return. end.
  elseif. do. jderr ERR001 return.

  end.

  x mnlsearch__ST y
end.
)

NB. master use bit mark
mubmark=: ] ; [: (6!:0) 0: $ ]

NB. J name class override - traps limit error for very long names
nc=: 4!:0 ::(_2:)

newd=: 3 : 0

NB.*newd v-- creates a new dictionary
NB.
NB. monad:  newd clName | (clName ; clPath)
NB.
NB.   newd 'New0Dict'              NB. store in default J user directory
NB.   newd 'New1Dict';'c:\put\it\here'            NB. windows drives
NB.   newd 'New2Dict';'\\shared\netdrive\new2'   NB. windows UNC shares
NB.   newd 'New3Dict';'/home/john/temp/new3'     NB. linux rooted paths

if. badcl y do.
  1 newregdict__ST y
```

```
else.
  drn=. y -. y -. ALPHA  NB. safe directory chars only
  1 newregdict__ST y;hostsep (jpath '~user\'),JJODDIR,(255<.#drn){.drn
end.
)

nlargs=: 4 : 0

NB.*nlargs v-- test basic name list arguments
NB.
NB. dyad:  il nlargs cl

if.     badcl y do. jderr ERR010  NB. errmsg: invalid name pattern
elseif. badil x do. jderr ERR001  NB. errmsg: invalid option(s)
NB. do we have a dictionary open?
elseif.do. checkopen__ST 0
end.
)

NB. numeric list timestamp
now=: 6!:0

NB. convert timestamp to yyyymmdd
nowfd=: ([: (0 100 100&#.) 3: {. ]) + ([: (24 60 60&#.) 3: }. ]) % 86400"_

obidfile=: 3 : 0
```

```
NB.*obidfile v-- location of jod object id history file.
NB.
NB. monad: obidfile uuIgnore

(jodsystempath ''),'jod.ijn'
)

od=: 3 : 0

NB.*od v-- opens and closes dictionaries.
NB.
NB. monad: od clDictionary|blclDictionary
NB.
NB. dyad:  iaOption od clDictionary|blclDictionary
NB.
NB.   od 'test dictionary';'another test dictionary'  NB. open r/w
NB.   3 od 'test dictionary'                          NB. close

1 od y
:
msg=. ERR005 NB. errmsg: invalid or missing dictionary names

NB. list all registered dictionaries (short form)
if. badjr mdt=. jread JMASTER;CNMFTAB do.
  jderr ERR006 return.
end.
dl=. 0{>mdt
```

```
select. x
case. 1 do.   NB. HARDCODE: magic numbers read/write codes

  if. isempty y do. ok /:~ dl
  else.
    NB. open read/write
    y=. boxopen ,y
    NB. all dictionary names must be on master list
    if. *./y e. dl do. y opendict__ST 1;mdt else. jderr msg end.
  end.

case. 2 do.

  NB. open read/only
  y=. boxopen ,y
  if. *./y e. dl do. y opendict__ST 2;mdt else. jderr msg end.

case. 3 do.

  NB. close dictionaries
  if. badrc msg1=. checkopen__ST 0 do. msg1 return. end.
  if. isempty y do. y=. {."1 DPATH__ST else. y=.boxopen ,y end.
  if. *./y e. dl do. mdt closedict__ST y else. jderr msg end.

case. 4 do.

  NB. HARDCODE (mdt rows) display dictionary names and source paths
  mdt=. jpathsep&.> 0 2{>mdt
  ok <(/:0{mdt){ |: mdt

case. 5 do.

  NB. return the currently registered dictionaries as a sorted (regd) script
  if. 0 e. $mdt=. >mdt do.
    ok 'NB. No current JOD registrations: ',tstamp ''
  else.
    mdt=. quote&.> 0 2{mdt {"1~ /: 0{mdt
    mdt=. ctl ;"1 (<'regd ') ,"1 |: 1 0 2{ (<';'),mdt
    NB. prefix command to close and unregister all current dictionaries
    mdt=. 'NB. require ''general/jod''',LF,'3 regd&> }. od'''' [ 3 od ''''',LF,mdt
    ok 'NB. JOD registrations: ',(tstamp ''),LF,jpathsep mdt
  end.

case.do. jderr ERR001  NB. errmsg: invalid option(s)
end.
)

NB. format normal return
ok=: 1: ; ]

packd=: 3 : 0

NB.*packd  v-- backs up and recovers wasted  space in  dictionary
NB. files.   Backups  are  stored  in  the   dictionary's  backup
NB. directory. Sets  of backup  files are prefixed  with an  ever
NB. increasing backup number, e.g: 13jwords.ijf. Dictionary files
NB. are NEVER deleted by JOD commands.
NB.
NB. monad:  packd clName
NB.
NB.   packd 'dictionary'

NB. NIMP: packd/restd not supported on iOS/Android devices for now
NB. if. badrc uv=. checksup 'packd' do. uv return. end.

NB. only put dictionaries can be packed
if. badrc uv=. checkput__ST 0 do. uv return. end.
DL=. 1 { uv  NB. directory object !(*)=. DL

NB. is there enough space on the backup volume?
if. badrc uv=. packspace__DL 0 do. uv return. end.

packdict__DL y
)


NB. promote lists to tables - other ranks unchanged
plt=: ]`,:@.(1&=@:(#@:$))


put=: 3 : 0

NB.*put v-- stores objects in dictionary database files.
NB.
NB. monad:  put blclWords
NB.
NB.   put ;: 'it where the sun dont shine'
NB.
NB. dyad:  ilCodes put bluu
NB.
NB.   2 7 put 'GroupName';'Group documentation text ....'

WORD put y
:
msg=. ERR001 [ loc=. <'base'  NB. errmsg: invalid option(s)

NB. do not save decommented words - set PUTBLACK to 1 to override
if. -. PUTBLACK +. 9!:40'' do.
  NB. errmsg: white space preservation is off - turn on to put
  jderr ERR023 return.
end.

if. badil x do.
  NB. errmsg: invalid or missing locale
  if. _1&badlocn x do. jderr ERR004 return. else. x=. WORD [ loc =. <x-.' ' end.
end.

NB. do we have a put dictionary open?
if. badrc uv=. checkput__ST 0 do. uv return. end.
DL=. 1 { uv  NB. directory object !(*)=. DL
```

```
NB. NOTE: j 9.04 introduced a new binary format for extended precision
NB. integers that is not backward compatible with prior versions of j.
NB. While it ok to read jod binary files created in older versions it's
NB. not ok to write to them. JOD uses extended precision integers to encode
NB. GUIDs. In retrospect it would have been a better choice to encode
NB. GUIDS as plain old character data. HARDCODE:
if. badrc msg=. binverchk DL do. msg return. end.

NB. format standard (x) options
x=. 2 {. x , DEFAULT

select. {. x
case. WORD do.
  select. second x
    case. DEFAULT  do. (loc;<DL) putwords__ST y
    case. EXPLAIN  do. (WORD;<DL) putexplain__ST y
    case. DOCUMENT do. (WORD;1;<DL) puttexts__ST y
    case. NVTABLE  do.
      if. badrc y=. (i. 4) checknttab2 y do. y else. (WORD;<DL) puttable__ST y end.
    case. -INPUT   do. (WORD;<DL) putntstamp__ST y
    case.do. jderr msg
  end.
case. TEST do.
  select. second x
    case. DEFAULT  do.
      if. badrc y=. checknttab y do. y else. (TEST;<DL) puttable__ST y end.
    case. EXPLAIN  do. (TEST;<DL) putexplain__ST y
    case. DOCUMENT do. (TEST;1;<DL) puttexts__ST y
    case. -INPUT   do. (TEST;<DL) putntstamp__ST y
    case.do. jderr msg
  end.
case. GROUP do.
  select. second x
    case. DEFAULT  do. (GROUP;0;<DL) puttexts__ST y
    case. EXPLAIN  do. (GROUP;<DL) putexplain__ST y
    case. DOCUMENT do. (GROUP;1;<DL) puttexts__ST y
    NB. HARDCODE - lines inserted to maintain put/get symmetry for
    NB. the frequent argument cases 2 1 and 3 1
    case. 1 do. (GROUP;0;<DL) puttexts__ST y
    case. -INPUT   do. (GROUP;<DL) putntstamp__ST y
    case.do. jderr msg
  end.
case. SUITE do.
  select. second x
    case. DEFAULT  do. (SUITE;0;<DL) puttexts__ST y
    case. EXPLAIN  do. (SUITE;<DL) putexplain__ST y
    case. DOCUMENT do. (SUITE;1;<DL) puttexts__ST y
    case. 1 do. (SUITE;0;<DL) puttexts__ST y  NB. HARDCODE
    case. -INPUT   do. (SUITE;<DL) putntstamp__ST y
    case.do. jderr msg
  end.
case. MACRO do.
  select. second x
```

```
    case. DEFAULT  do.
      if. badrc y=. MACROTYPE checknttab2 y do. y else. (MACRO;<DL) puttable__ST y end.
    case. EXPLAIN  do. (MACRO;<DL) putexplain__ST y
    case. DOCUMENT do. (MACRO;1;<DL) puttexts__ST y
    case. -INPUT   do. (MACRO;<DL) putntstamp__ST y
    case.do. jderr msg
  end.
case. DICTIONARY do.
  select. second x
    case. DEFAULT  do. putdicdoc__ST y
    case.do. jderr msg
  end.
case.do. jderr msg
end.
)

NB. quotes character lists for execution
quote=: ''''&,@(,&'''')@(#~ >:@(=&''''))

NB. reads a file as a list of bytes
read=: 1!:1&(]`<@.(32&>@(3!:0)))

NB. reads a J binary noun file
readnoun=: 3!:2@(1!:1&(]`<@.(32&>@(3!:0))))

readobid=: 3 : 0
```

```
NB.*readobid v-- unique object ids that opened dictionaries
NB. read/write on this machine.
NB.
NB. monad: readobid uuIngnore

(readnoun :: ((i.0)"_)) obidfile 0
)

regd=: 3 : 0

NB.*regd v-- register and unregister JOD dictionaries.
NB.
NB. monad:  regd blcl
NB.
NB.    regd 'name';'c:\location\of\files';'documentation...'
NB.
NB. dyad:  iaOption make cl
NB.
NB.    3 regd 'name'  NB. unregister dictionary

0 newregdict__ST y
:
if. x-:3 do. NB. HARDCODE option
  NB. errmsg: invalid or missing dictionary name(s)
  if. badcl y do. jderr ERR005 return. end.
  NB. errmsg: dictionary in use - cannot unregister
  if. (<,y) e. {."1 DPATH__ST do. jderr ERR018 return. end.
  NB. errmsg: cannot read master
  if. badjr mdt=. jread JMASTER;CNMFTAB do. jderr ERR006 return. end.
  mdt=.>mdt
  mu=. (0{mdt)=<,y
  if. +./mu do.
    'path inuse'=. 2 3{mu #"1 mdt
    NB. errmsg: dictionary in use - cannot unregister
    if. inuse do. jderr ERR018 return. end.
    newmdt=. (-.mu)#"1 mdt
    if. badrc msg=. markmast 1 do. msg return. end.
    if. badreps ((<newmdt),<mdt) jreplace JMASTER;CNMFTAB,CNMFTABBCK do.
      jdmasterr ERR017 return.  NB. errmsg: jfile replace error
    end.
    if. badrc msg=. markmast~0 do. msg return. end.
    (ok OK001),y;jpathsep path
  else.
    jderr ERR005
  end.
else.
  jderr ERR001
end.
)

remast=: 3 : 0

NB.*remast v-- clears all  in use bits  in the  master file. When
NB. JOD  opens a dictionary  an in  use bit is set in the  master
NB. file. When the dictionary  is closed the bit is cleared. When
NB. the  in  use  bit is set  the  dictionary  cannot  be  opened
NB. read/write by other dictionary tasks.
NB.
NB. monad:  remast paMeAll
NB.
NB.   remast 0  NB. reset all
NB.   remast 1  NB. reset me

mdt=. > jread JMASTER;CNMFTAB
if. 0=y do.
  NB. reset all
  mdt=. (<"0 ({:$mdt)#0) 3} mdt
else.
  NB. reset me
  mdt=. (<0) (<3;I. (;3{mdt) e. readobid obidfile 0)}mdt
end.
(<mdt) jreplace JMASTER;CNMFTAB
)

restd=: 3 : 0

NB.*restd v-- restores the most recent backup created by (packd).
NB.
NB. monad:  restd cl
NB.
NB.   restd 'backup'

NB. NIMP: packd/restd not supported on iOS/Android devices for now
NB. if. badrc uv=. checksup 'restd' do. uv return. end.
NB. only put dictionaries can be restored
if. badrc uv=. checkput__ST 0 do. uv return. end.
DL=. 1 { uv  NB. directory object !(*)=. DL

NB. is there enough space on the dictionary volume?
if. badrc uv=. restspace__DL 0 do. uv else. (}. uv) restdict__DL y end.
)

NB. ok return value
rv=: >@(1&({ ))


rxs=: ''&$: :(4 : 0)

NB.*rxs v-- regular expression search.
NB.
NB. monad:  rxs blclNames
NB.
NB.   NB. display all WORD regx search text
NB.   NB.  '' rxs }. dnl 're'
NB.
NB.   rxs }. dnl 're'
NB.
NB. dyad:  (clPatten ; ilCodes) rxs blclNames
NB.        clPattern rxs blclNames
NB. do we have a dictionary open?
if. badrc uv=. checkopen__ST 0 do. uv return. end.

NB. (x) is either cl or (cl ; il)  errmsg: invalid option(s)
msg=. ERR001
if. 1 < L. x do. jderr msg return. end.
if. 0 = L. x do. x=. x ; WORD,DEFAULT,1
else.
  if. (1 ~: $$,x) *. 2 ~: #,x do. jderr msg return. end.
end.

NB. regular expression and object options
'pat opts'=. x
if. badcl pat  do. jderr msg return. end.
if. badil opts do. jderr msg return. end.

NB. format options HARDCODE: codes and positions
opts=. opts , (-3-#opts) {. DEFAULT , 1
if. -. 1 2 3 e.~ {: opts do. jderr msg return. end.

if. DICTIONARY=0{opts do.
  NB. no short and long texts for dictionary documents
  if. DEFAULT ~: 1{opts do. jderr msg return. end.
  NB. tolerate any (y) for dictionary text case
  uv=. opts rxsget 0
else.
```

```
  NB. are names valid?
  if. badrc y=.checknames y do. y return. else. y=. }.y end.

  NB. remove nouns - they are not searched for patterns
  NB. return nothing found if all names are nouns
  if. WORD = 0{opts do.
    if. badrc uv=. (WORD,INCLASS) invfetch__ST y do. uv return. end.
    if. 0 = #y=. y #~ 0 ~: >1{uv do. ok <0 2$<'' return. end.
  end.

  if. badrc uv=. opts rxsget y do. uv return. end.
end.

NB. empty patterns mean return all nonempty text to be searched
NB. handy for complex pattern debugging and verification
if. #pat do. (pat;opts) rxssearch >1{uv else. uv  end.
)

rxsget=: 4 : 0

NB.*rxsget v-- retrieves  text  objects from dictionary  database
NB. files.
NB.
NB. A  variation of (get) that only retrieves  text  objects from
NB. dictionary  database  files. (rxsget)  returns the texts that
NB. are searched for regular expression patterns by (rxs).
NB.
NB. Note:  binary objects (nouns)  are  eliminated from the  name
NB. list (y) by the caller of this verb.
NB.
NB. dyad:  ilCodes rxsget bluu
NB.
NB.   2 8 1 rxsget 'GroupName'
NB.   4 7 1 rxsget 'MacroText'

msg=. ERR001 NB. errmsg: invalid option(s)

select. {. x
case. WORD do.
  select. second x
    case. DEFAULT  do. txt=. (WORD,0) getobjects__ST y
    case. EXPLAIN  do. txt=. WORD getexplain__ST y
    case. DOCUMENT do. txt=. WORD getdocument__ST y
    case.do. jderr msg return.
  end.
case. TEST do.
  select. second x
    case. DEFAULT  do. txt=. (TEST,0) getobjects__ST y
    case. EXPLAIN  do. txt=. TEST getexplain__ST y
    case. DOCUMENT do. txt=. TEST getdocument__ST y
    case.do. jderr msg  return.
  end.
case. GROUP do.
  select. second x
    case. DEFAULT  do. txt=. GROUP getgstext__ST y
    case. EXPLAIN  do. txt=. GROUP getexplain__ST y
    case. DOCUMENT do. txt=. GROUP getdocument__ST y
    case.do. jderr msg return.
  end.
case. SUITE do.
  select. second x
    case. DEFAULT  do. txt=. SUITE getgstext__ST y
    case. EXPLAIN  do. txt=. SUITE getexplain__ST y
    case. DOCUMENT do. txt=. SUITE getdocument__ST y
    case.do. jderr msg return.
  end.
case. MACRO do.
  select. second x
    case. DEFAULT  do. txt=. (MACRO,0) getobjects__ST y
    case. EXPLAIN  do. txt=. MACRO getexplain__ST y
    case. DOCUMENT do. txt=. MACRO getdocument__ST y
    case.do. jderr msg return.
  end.
case. DICTIONARY do.
  select. second x
    case. DEFAULT  do. txt=. getdicdoc__ST 0
    case.do. jderr msg return.
  end.
case.do. jderr msg return.
end.

if. badrc txt do. txt
```

```
else.
  NB. form two column (name,text) table remove 0 length texts
  if. badcl txt=. >1{txt do.
    txt=. (0,<:{:$txt) {"1 txt
    ok <txt #~ 0 < #&> 1 {"1 txt
  else.
    NB. dictionary documentation case often empty - only unnamed text
    ok <((0<#txt),2)$'';txt
  end.
end.
)

rxssearch=: 4 : 0

NB.*rxssearch v-- search (name, text) table for regex matches.
NB.
NB. dyad:  (clPat ; ilOpts) rxssearch btNameText

NB. all arguments validated by callers
'pat opts'=. x

NB. require 'regex' !(*)=. rxfirst rxall rxmatches rxutf8
NB. NOTE: workaround for J 9.04 PCRE2 changes
NB. turn of utf8 support for J 9.04+ !(*)=. rxutf8
if. b903=. 9.03 < jvn'' do. rgs=. rxutf8 0 end.

NB. HARDCODE: option codes
try.
```

```
  select. {:opts
  case. 1 do.
    h=. pat&rxfirst&.> 1 {"1 y
    if. b903 do. rgs=. rxutf8 rgs end.
    ok  <((0 {"1 y) ,. h) #~ 0 < #&> h
  case. 2 do.
    h=. pat&rxall&.> 1 {"1 y
    if. b903 do. rgs=. rxutf8 rgs end.
    ok  <((0 {"1 y) ,. h) #~ 0 < #&> h
  case. 3 do.
    h=. pat&rxmatches&.> 1 {"1 y
    if. b903 do. rgs=. rxutf8 rgs end.
    b=. 0 < #&> h
    ok  <(b # 0 {"1 y) ,. (b # h) ,. b # 1 {"1 y
  case.do.
    if. b903 do. rgs=. rxutf8 rgs end.
    jderr ERR001
  end.
catchd.
  if. b903 do. rgs=. rxutf8 rgs end.
  NB. errmsg: regex pattern error ->
  (jderr ERR029),<13!:12''
end.
)

saveobid=: 3 : 0

NB.*saveobid v--  saves the last  n JOD object ids  in the \jnxxx
NB. directory. These globally unique values are used to reset any
NB. dictionaries left open by JOD tasks spawned from  the current
NB. machine.
NB.
NB. monad:  saveobid xiObid
NB.
NB.   saveobid JODOBID

id=. ~. y , readobid file=.obidfile 0

NB. HARDCODE up to 30 last object ids spawned on this machine
NB. NOTE: if you run more than 30 JOD tasks on the current
NB. machine you will lose object id's which cause the RESETME
NB. option of (dpset) to not reset all dictionaries recently opened -
NB. but never closed - on this machine. JUST INCREASE THE NUMBER EHHH!!
((30<.#id) {. id) (writenoun :: _1:) file
)


NB. second list item
second=: 1&({ )


NB. J type code
tc=: 3!:0


NB. removes blanks from items on blcl
trimnl=: -.&' '&.>
```

```
NB. appends trailing / iff last character is not \ or /
tslash2=: ([: - '\/' e.~ {:) }. '/' ,~ ]

tstamp=: 3 : 0

NB.*tstamp v-- standard j 8_07 library timestamp.
NB.
NB. A renamed version of the standard  J 8.07 era  timestamp. JOD
NB. used an earlier version of this verb, see (tstamp2), that did
NB. not handle all zero timestamps.
NB.
NB. monad:  clDate =. tstamp il | fl
NB.
NB.   tstamp '' NB. now timestamp
NB.   tstamp 0 0 0 0 0 0  NB. zero stamp

if. 0 = #y do. w=. 6!:0'' else. w=. y end.
r=. }: $ w
t=. 2 1 0 3 4 5 {"1 [ _6 [\ , 6 {."1 <. w
d=. '+++::' 2 6 11 14 17 }"1 [ 2 4 5 3 3 3 ": t
mth=. _3[\'   JanFebMarAprMayJunJulAugSepOctNovDec'
d=. ,((1 {"1 t) { mth) 3 4 5 }"1 d
d=. '0' (I. d=' ') } d
d=. ' ' (I. d='+') } d
(r,20) $ d
)

uses=: 3 : 0
NB.*uses v-- returns word references.
NB.
NB. monad:  uses blclName
NB.
NB.   NB. non-locale global word references
NB.   uses ;:'out global references please'
NB.
NB. dyad:   ilObjOpt uses clName
NB.
NB.   NB. global locale word references
NB.   11 uses ;:'out locale references'
NB.
NB.   0 31 uses 'wordname'  NB. uses union of word
NB.   0 32 uses '

0 uses y
:

if. badrc uv=. checkopen__ST 0 do. uv return. end.
if. badrc y=.checknames y do. y return. else. y=. }.y end.

msg=. ERR001 NB. errmsg: invalid option(s)
if. badil x do. jderr msg return. else. x=. ''$x end.

if. x-:WORD do.
  if. badrc dat=.WORD getrefs__ST y do. dat return. end.
  dat=. rv dat
  dat=. (uv=. {."1 dat) ,. > {:"1 dat
  NB. return in order requested
  ok <(({."1 dat) i. y){dat
elseif. x-:UNION do.

  NB. word uses unions
  uv=. i. 0 0
  for_wrd. y do.
    srch=.'' [ refs=.wrd
    loc=.'' [ self=.0
    while.do.
      if. badrc dat=.WORD getrefs__ST refs do. dat return. end.
      srch=. ~. srch , {."1 dat=. rv dat
      NB. only non-locale names are searched
      self=. self+.wrd e. new=. ~. ; {.&> {:"1 dat
      new=. new -. srch
      loc=. ~. loc , (; {:&> {:"1 dat) -. loc
      if. isempty new do. break. end.
      refs=. new
    end.
    srch=. /:~ srch -. self}.wrd
    uv=. uv,wrd,srch;<loc
  end.
  ok <uv

elseif.do. jderr msg
end.
)

valdate=: 3 : 0

NB.*valdate v-- validates lists or tables of YYYY MM DD Gregorian
NB. calendar dates.
NB.
NB. monad:  valdate il|it
NB.
NB.   valdate 1953 7 2
NB.   valdate 1953 2 29 ,: 1953 2 28  NB. not a leap year

s=. }:$y
'w m d'=. t=. |:((*/s),3)$,y
b=. *./(t=<.t),(_1 0 0<t),12>:m
day=. (13|m){0 31 28 31 30 31 30 31 31 30 31 30 31
day=. day+(m=2)*-/0=4 100 400|/w
s$b*d<:day
)

NB. 1 when word with name exists 0 otherwise
wex=: 0&<:@:nc

NB. word storage representation - nouns binary others linear
wrep=: 5!:5@<`(3!:1@:".)@.(0&=@(nc@<))
```

```
NB. writes a list of bytes to file
write=: 1!:2 ]`<@.(32&>@(3!:0))

NB. writes a J noun file
writenoun=: ([: 3!:1 [) (1!:2 ]`<@.(32&>@(3!:0))) ]
```

jodstore Source Code

```
NB.*jodstore c-- storage object class: extension of (jod).
NB.
NB. Hides the underlying database/file system used to store
NB. dictionary objects. Replacing this class is all that's
NB. required to change the dictionary storage system.
NB.
NB. Verb interface:
NB.   bchecknames  checks backup name patterns
NB.   bgetdicdoc   get backup versions of the dictionary document
NB.   bgetexplain  get backup versions of short object explanations
NB.   bgetgstext   get backup versions of group/suite headers
NB.   bgetobjects  get objects from backups
NB.   bnlsearch    searches put dictionary backup name lists for simple character list patterns
NB.   bnums        returns unique backup ordered list of dictionary backup numbers
NB.   checkopen    checks if any dictionary is open
NB.   checkpath    checks current path against dictionary path
NB.   checkput     checks if first path dictionary is a put dictionary
NB.   closedict    closes dictionaries
NB.   createst     initializes storage objects
NB.   defwords     define words
NB.   delstuff     delete objects
NB.   didstats     dictionary statistics and path information
NB.   dnlsearch    search for name patterns
NB.   getdocument  get object documentation
NB.   getexplain   get short object explanations
NB.   getgstext    get group and suite script text
NB.   getntstamp   get name, creation and last put timestamps
NB.   getobjects   get objects
NB.   getrefs      get references
NB.   gslistnl     group and suite name lists
NB.   inputdict    test for objects in put dictionary
NB.   invappend    append inverted data
NB.   invdelete    delete inverted data
NB.   invfetch     fetch inverted data
NB.   invreplace   update inverted data
NB.   newregdict   create new or register dictionary
NB.   opendict     open a dictionary
NB.   pathnl       path name lists
NB.   putexplain   store short object explanations
NB.   putgs        store groups and suites
NB.   putntstamp   store name, creation and last put timestamps
NB.   puttable     store (name,text) and (name,class,text) tables
NB.   puttexts     store object documentation and group/suite texts
NB.   putwords     store words
NB.   putwordxrs   store word global references
NB.
NB. Notes:
NB.   Error messages (jodstore range 050-149)

coclass  'ajodstore'
coinsert 'ajod'
```

```
NB.*dependents x-- JODstore dependent defintions

CNMARK=: 0      NB. file component: count and timestamp mark
CNLIST=: 4      NB. file component: main object index list
CNCOMPS=: 5     NB. file component: main object component list

NB. main directory file component list
CNDIR=: CNMARK,CNLIST,CNCOMPS

CNCLASS=: 6     NB. file component: word name class or macro type
CNCREATION=: 8  NB. file component: when object was first created
CNDICDOC=: 2    NB. file component: dictionary documentation - (regd)
CNEXPLAIN=: 11  NB. file component: short explanations
CNPARMS=: 3     NB. file component: dictionary parameters
CNPUTDATE=: 7   NB. file component: last time object was (put)
CNREF=: 5+i.2 2 NB. reference component table
CNRPATH=: 19    NB. file component: reference path - (didnum) list
CNSIZE=: 9      NB. file component: size of object in bytes

NB. inverted group and suite data file components
INVGROUPS=:  CNPUTDATE,CNCREATION,CNEXPLAIN
INVSUITES=:  INVGROUPS

NB. inverted macro and word data file components
INVMACROS=:  CNCLASS,CNPUTDATE,CNCREATION,CNSIZE,CNEXPLAIN
INVWORDS=:   INVMACROS
```

```
NB. inverted test data
INVTESTS=:  CNPUTDATE,CNCREATION,CNSIZE,CNEXPLAIN

NB. name.n or name.name separator character
NDOT=: '.'

NB. trim right (trailing) path delimiters !(*)=. PATHDEL
rpdtrim=: ] #~ [: -. [: *./\. PATHDEL"_ = ]

NB. split backup name pattern cl
splitbname=:  (NDOT&beforestr ; NDOT&afterstr)

NB.*enddependents
NB.*end-header

NB. file component: J dictionary creator version string
CNJVERSION=: 12

NB. initial documentation list: latex ; html ; text
DOCINIT=: <;._1 '   '

ERR050=: 'no dictionaries open'

ERR051=: 'not a put dictionary ->'

ERR052=: 'unable to initialize ->'
```

```
ERR053=: 'word(s) do not exist ->'

ERR054=: 'unable to load directory'

ERR055=: 'directory-data inconsistency'

ERR056=: 'jfile replace failure'

ERR057=: 'directory update failure'

ERR058=: 'jfile append failure'

ERR059=: 'full rooted path required'

ERR060=: 'unable to create directory ->'

ERR061=: 'invalid dictionary name;path[;documentation]'

ERR062=: 'invalid characters in name'

ERR063=: 'invalid characters in path'

ERR064=: 'target drive is required'

ERR065=: 'not enough space on drive/volume ->'
```

```
ERR066=: 'dictionary name in use'

ERR067=: 'unable to create subdirectories'

ERR068=: 'unable to setup dictionary file(s)'

ERR069=: 'error updating master'

ERR070=: 'request exceeds open limit'

ERR071=: 'already open ->'

ERR072=: 'another task opened read/write ->'

ERR073=: 'missing dictionary file(s) ->'

ERR074=: 'cannot read dictionary parameters ->'

ERR075=: 'unable to open directory ->'

ERR076=: 'master-dictionary inconsistency - try (dpset) ->'

ERR077=: 'unable to update master'

ERR079=: 'unable to load references'
```

```
ERR080=: 'not open ->'

ERR081=: 'path mismatch'

ERR082=: 'unable to set reference path'

ERR083=: 'not on path ->'

ERR084=: 'unable to read data'

ERR085=: 'words(s) not defined ->'

ERR086=: 'not in put dictionary ->'

ERR087=: 'nothing in put dictionary'

ERR088=: 'jfile read failure'

ERR089=: 'text(s) to long'

ERR090=: 'file offset invalid'

ERR091=: 'definition failure'

ERR092=: 'duplicate dictionary id number'
```

```
ERR093=: 'directory damaged'

ERR094=: 'exceeds locale symbol table size - no words defined'

ERR095=: 'dictionary file attributes do not allow read/write ->'

ERR096=: 'linux/unix dictionary paths must be / rooted ->'

ERR097=: 'invalid dictionary document must be character list'

ERR098=: 'master/dictionary file path mismatch - name/DIDNUM ->'

ERR099=: 'invalid name/creation/lastput table'

ERR100=: 'name/creation/lastput length mismatch'

ERR101=: 'invalid date(s) name/creation/lastput table'

ERR102=: 'timestamp table shape invalid'

ERR103=: 'no backup(s) to restore or search'

ERR104=: 'no registered dictionaries'

ERR105=: 'unreadable or missing backup timestamp'
```

```
ERR106=: 'invalid backup number(s)'

ERR107=: 'not in backup(s) -> '

ERR108=: 'cannot register binary incompatible read/write dictionary ->'

NB. directory and reserved components in *.ijf files
OFFSET=: 39

OK050=: 'dictionary created ->'

OK051=: ' word(s) put in ->'

OK052=: 'opened ('

OK054=: 'closed ->'

OK055=: ' explanation(s) put in ->'

OK056=: ' references put in ->'

OK057=: '(s) put in ->'

OK058=: 'dictionary registered ->'
OK059=: 'put in ->'

OK060=: ' word(s) defined'

OK061=: '(s) deleted from ->'

OK062=: 'dictionary document updated ->'

OK063=: '(DOCUMENTDICT = 0) - dictionary document not updated ->'

OK064=: ') timestamps updated - ('

OK065=: ') not in put ->'

NB. path report title
PATHTIT=: 'Path*'

NB. visible read status text
READSTATS=: <;._1 ' ro rw'

NB. retains string (y) after last occurrence of (x)
afterlaststr=: ] }.~ #@[ + 1&(i:~)@([ E. ])

NB. contains string in lists of list of names
allnlctn=: [ /:~@:nlctn&.> [: < ]
```

```
NB. match prefixes in lists of lists of names - (pathnl) related
allnlpfx=: [ /:~@:nlpfx&.> [: < ]

NB. match suffixes in lists of lists of names
allnlsfx=: [ /:~@:nlsfx&.> [: < ]


apptable=: 4 : 0

NB.*apptable v-- appends (name,text) and (name,class,text) tables to file.
NB.
NB. dyad:  bl apptable bt

'ttype ixn cnn fp DL'=. x  NB. directory object !(*)=. DL
sizes=. #&> {:"1 y         NB. sizes
pf=. PUTFACTOR__DL

NB. words and macros have class or type
if. wmt=. ttype e. WORD,MACRO do. class =. ; 1 {"1 y end.

texts=. y
cnall=. i.0
y=. {."1 y   NB. no longer required

while. #texts do.
  cnt=. pf <. #texts
  tn=.  cnt {. texts
  un0=. <"1 tn
  un1=. <"1 ({."1 tn) ,"0 1 DOCINIT

  if. badappend cn=. (, un0 ,. un1) jappend fp do.
    jderr ERR058 return.  NB. errmsg: append failure
  else.
    cnall=. cnall , fod cn
  end.

  texts=. cnt }. texts
end.

NB. append directory and inverted lists
msg=. ERR057  NB. errmsg: directory update failure
if. (tc=. #y) ~: #cnall do. jderr msg return. end.

stamp=. tc#nowfd now ''
un0=. stamp;stamp;sizes;<tc#a:
un1=. CNPUTDATE,CNCREATION,CNSIZE,CNEXPLAIN

if. wmt do.
  dropnc__DL ttype  NB. force class reload
  un0=. class;un0
  un1=. CNCLASS,un1
end.

if. badrc msg=. un0 invappend fp;un1 do. msg
```

```
else.
  NB. update directory
  y=.    (".ixn) , y
  cnall=. (".cnn) , cnall
  if. badrc (ttype,fp) savedir__DL y;cnall do. jderr msg else. ok tc end.
end.
)

appwords=: 4 : 0

NB.*appwords v-- appends new words in blocks of (PUTFACTOR).

'loc DL'=. x  NB. directory object !(*)=. DL
wp=. WP__DL [ pf=. PUTFACTOR__DL
names=. y
lnames=. y ,&.> locsfx loc
size=. class=. cnall=. i.0

while. #names do.
  cnt=. pf <. #names
  wn=.  cnt {. names [ lwn=. cnt {. lnames
  val=. wrep&.> lwn   NB. word values
  bsz=. #&> val       NB. NIMP word byte sizes (size testing)
  bnc=. nc lwn
  un0=. <"1 wn ,. (<"0 bnc) ,. val
  un1=. <"1 wn ,"0 1 DOCINIT

  NB. append words
  if. badappend cn=. (, un0 ,. un1) jappend wp do.
    jderr ERR058 return.  NB. errmsg: append failure
  else.
    cnall=. cnall , fod cn
    size=.  size , bsz
    class=. class , bnc
  end.

  names =. cnt }. names [ lnames =. cnt }. lnames
end.

NB. append directory and inverted lists
msg=. ERR057  NB. errmsg: directory update failure
if. (#y) ~: #cnall do. jderr msg return. end.
wc=. #y  NB. number of words

stamp=. wc#nowfd now ''
un0=. class;stamp;stamp;size;<wc#a:
un1=. CNCLASS,CNPUTDATE,CNCREATION,CNSIZE,CNEXPLAIN NB. NIMP word append
if. badrc msg=. un0 invappend wp;un1 do. msg
else.
  NB. update word directory
  y=.    WORDIX__DL , y
  cnall=. WORDCN__DL , cnall
  if. badrc (WORD,wp) savedir__DL y;cnall do. jderr msg else. ok wc end.
end.
)
```

```
backupdates=: 4 : 0

NB.*backupdates v-- scans put dictionary backup files and returns
NB. backup dates.
NB.
NB. This  verb  attempts  to   read  component  index  1  of  put
NB. dictionary  (jwords)  backup files.  The resulting data takes
NB. these possible forms.
NB.
NB. verbatim:
NB.
NB.    1. bnum,timestamp - pack count and timestamp
NB.    2. bnum,0         - pack count and 0
NB.    3. _1`            - jread error - probably an older unreadable binary
NB.    4. _2             - trapped jread error - serious problemos
NB.
NB. dyad:  bt =. blObj backupdates ilBnums
NB.
NB.    NB. DL is put dictionary object
NB.    NB. bnums is a list of put dictionary backup numbers
NB.
NB.    DL backupdates bnums

NB. HARDCODE: component 1
uv=. >jread"1 (<1) ,.~  (<BAK__x) ,&.> (":&.> <"0 y) ,&.> 0{JDFILES
bstamps=. }."1 uv [ bn=. 0 {"1 uv
```

```
NB. format timestamps
bstamps=. (<"0 bn) ,. <"1 tstamp"1 bstamps

NB. errmsg: unreadable or missing backup timestamp
bstamps=. (<ERR105) (<(I. 0>bn);1)} bstamps
)

NB. bad jfile components - first names do not match list
badcn=: [: -. [ -: [: {.&> ]


bchecknames=: 4 : 0

NB.*bchecknames v-- checks backup name patterns.
NB.
NB. dyad:  ilObjBn bchecknames blclBnames
NB.
NB.   NB. valid ordered put dictionary backup numbers
NB.   bn=. rv_ajod_ checkback__ST__JODobj _1{0{DPATH__ST__JODobj
NB.
NB.   NB. first item of (x) is a dictionary object code
NB.   (WORD,bn) bchecknames__ST__JODobj <;._1' booo hhh re.12 bx.14 er.99'
NB.
NB.   NB. names are not required for the special DICTIONARY case
NB.   (DICTIONARY,bn) bchecknames__ST__JODobj <;._1' .71 .73 .65'

NB. errmsg: invalid name pattern(s)
if. +./ badcl&> y do. jderr ERR010 return. end.
```

```
NB. split backup name patterns
nbk=. (splitbname&> y) -.&.> ' '

NB. if backup number is absent use most recent
nbk=. (<":1{x) (<(I. 0 = #&> 1 {"1 nbk);1)} nbk

NB. names must be valid
if. DICTIONARY = 0{x do. bnm=. 0 {"1 nbk
elseif. badrc bnm=. checknames 0 {"1 nbk do. bnm return.
elseif.do. bnm=. }.bnm
end.

NB. backup numbers must be valid
if. 0 e. (1 {"1 nbk) *./@e.&> <DIGITS do. jderr ERR106 return. end.
bn=. , _1&".&> 1 {"1 nbk

NB. errmsg: invalid backup number(s)
if. 0 e. bn e. x do. jderr ERR106 return. end.

NB. return unique checked names and backup numbers
ok <~.bnm ,. <"0 bn
)

bgetdicdoc=: 3 : 0

NB.*bgetdicdoc v-- get backup versions of the dictionary document.
NB.
NB. monad:  bgetdicdoc btNameBn

NB. there is only one document per dictionary unique
NB. dictionary backup numbers insure no redundant file reads
bn=. ~.1 {"1 y

NB. put dictionary object  !(*)=. doj
doj=. {:{.DPATH

NB. dictionary document results combine dictionary name
NB. with backup numbers to differentiate versions
NB. NOTE: the resulting label may not be a valid J name
NB. unless the JOD dictionary name is a valid J name.
ro=. ((<DNAME__doj) ,&.> '_' ,&.> ":&.> bn) ,. a:

NB. backup path and file suffix
'pth fsx'=. bpathsfx WORD

ubn=. ;bn
for_bob. ubn do.

  fn=. pth,(":bob),fsx NB. backup file

  NB. read document component
  if. badjr dat=. jread fn;CNDICDOC do. jderr ERR088 return. end.

  NB. update results
  ro=. dat (<(I. bob=ubn);1)} ro

end.

NB. insure any empty documents have literal datatype
ok <btextlit ro
)

bgetexplain=: 4 : 0

NB.*bgetexplain v-- get short explanations from backups.
NB.
NB. dyad: il bgetexplain btNameBn

NB. object names
nnm=. 0 {"1 y [ obj=. 0{x

NB. results are boxed name literal value tables
ro=. nnm ,. <,''

'pth fsx'=. bpathsfx obj

NB. fetch backup objects by backup number - optimizes file reads
cpm=. CNLIST,CNEXPLAIN
ubn=. ~.bn=. ; 1 {"1 y
for_bob. ubn do.

  fn=. pth,(":bob),fsx NB. backup file
  NB. read backup explanations errmsg: read failure
  if. badjr 'ixn sex'=. jread fn;cpm do. jderr ERR088 return. end.

  NB. explanations must exist in backup(s) errmsg: not in backups ->
  sn=. nnm {~ rx=. I. bob=bn
  if. 0 e. uv=. sn e. ixn do. (jderr ERR107),(sn #~ -.uv) ,&.> <NDOT,":bob return. end.

  NB. update results
  ro=. (sex {~ ixn i. sn) (<rx;1)} ro

  NB. distinquish object names with backup number suffix
  ro=. (((<rx;0){ro) ,&.> <'_',":bob) (<rx;0)} ro

end.

NB. insure any empty explanations have literal datatype
ok <btextlit ro
)

bgetgstext=: 4 : 0

NB.*bgetgstext v-- get backup versions of group/suite headers.
NB.
NB. dyad: il bgetobjects btNameBn

if. badrc uv=. (x,0) bgetobjects y do. uv else. ok <0 1 {"1 rv uv end.
)
```

```
bgetobjects=: 4 : 0

NB.*bgetobjects v-- get objects from backups.
NB.
NB. dyad: il bgetobjects btNameBn

NB. object code, offset and names
nnm=. 0 {"1 y [ 'obj offset'=. x

NB. HARDCODE: 2 indicates fetching group/suite list(s)
offset=. (bgslist=. offset=2){offset,0

NB. results are boxed name value tables
NB. words & macro have three columns
ro=. nnm ,"0 1 (1 + (offset=0) * obj e. WORD,MACRO)$a:

NB. HARDCODE: result columns
cols=. 0 _1
if. (0=offset) *. -.bgslist do. cols=. i. {:$ro end.

NB. backup path and file suffix
'pth fsx'=. bpathsfx obj

NB. fetch backup objects by backup number - optimizes file reads
cpm=. CNLIST,CNCOMPS
ubn=. ~.bn=. ; 1 {"1 y
for_bob. ubn do.
```

```
  fn=. pth,(":bob),fsx NB. backup file

  NB. read backup directory index  errmsg: read failure
  if. badjr 'ixn ixc'=. jread fn;cpm do. jderr ERR088 return. end.

  NB. objects must exist in backup(s) errmsg: not in backups ->
  sn=. nnm {~ rx=. I. bob=bn
  if. 0 e. uv=. sn e. ixn do. (jderr ERR107),(sn #~ -.uv) ,&.> <NDOT,":bob return. end.

  NB. read object components
  if. badjr dat=. jread fn;offset+(ixn i. sn){ixc do. jderr ERR088 return. end.

  NB. update results
  ro=. (cols {"1 >dat) rx} ro

  NB. distinquish object names with backup number suffix
  ro=. (((<rx;0){ro) ,&.> <'_',":bob) (<rx;0)} ro

end.

NB. for nonwords insure any empty texts have literal datatype
if. obj~:WORD do. ro=. btextlit ro end.

ok <ro  NB. return object table
)

bnlsearch=: 4 : 0
```

```
NB.*bnlsearch  v-- searches put dictionary backup name  lists for
NB. simple character list patterns.
NB.
NB. dyad: ilObjOptNc bnlsearch clPattern

NB. at most one '.' character errmsg: invalid name pattern
if. 1 < +/ y e. NDOT do. jderr ERR010 return. end.

NB. maintains argument compatibility with (dnl)
bv=. DEFAULT ~: 2{x
if. bv *. (0{x) e. TEST,GROUP,SUITE do. jderr ERR001 return. end.

NB. put dictionary directory object
DL=. {:0{DPATH

NB. extant backup numbers errmsg: no backup(s) to restore or search
if. badrc uv=. checkback DL do. uv return. else. bn=. rv uv end.

NB. search name pattern and requested backup
'pat rbk'=. splitbname y

NB. use most recent backup if none specified
if.     isempty rbk           do. rbk=. {.bn
elseif. 0 e. rbk e. DIGITS    do. jderr ERR010 return.
elseif. -.(rbk=. ".rbk) e. bn do. jderr ERR103 return.
end.
```

```
NB. nonempty patterns must be valid J names without embedded locales
if. #uv=. pat -. ' ' do.
  if. badrc uv=. checknames pat do. uv return. end.
end.

bdot=. (,NDOT)-:alltrim y

if. bdot *. INPUT={.x do.

  NB. show pack/backup dates
  ok <DL backupdates bn

elseif. bdot do.

  NB. return backup suffixes
  dot=. (0<#bn){'';NDOT
  ok dot ,&.> 'r<0>0.d' 8!:0 bn

elseif. bfile=. ({.x) dbakf__DL rbk
        NB. errmsg: jfile read failure

  badjr uv=. jread bfile;(1{CNDIR),CNCLASS do. (jderr ERR088,' ->'),<bfile

elseif.
        ol=. uv{ol [ uv=. /: ol [ 'ol oc'=. uv
        NB. reduce object list for words and macros if class specified
```

```
    if. bv *. (0{x) e. WORD,MACRO do. ol=. (oc = 2{x)#ol [ oc=. uv{oc end.

  isempty pat do. ok ol  NB. return sorted last backup name list

elseif. 0=#ol do. ok ol  NB. nothing left to match
elseif. do.              NB. match prefix, infix suffix
  select. 1{x
    case. 1 do. ok ol nlpfx pat
    case. 2 do. ok ol nlctn pat
    case. 3 do. ok ol nlsfx pat
    case. do. jderr ERR010
  end.
end.
)


bnums=: 3 : 0

NB.*bnums  v--  returns unique backup ordered list  of dictionary
NB. backup numbers.
NB.
NB. monad:  il =. bnums clPath
NB.
NB.   bnums BAK NB. (BAK) directory object noun

NB. requires first character of all (JDFILES) to be the same
\:~ ~. , ". ({.;JDFILES)&beforestr&> {."1 (1!:0) <y,'*',IJF
)
```

```
bpathsfx=: 3 : 0

NB.*bpathsfx v-- backup file path and file name suffix.
NB.
NB. monad:  (clPath ; clSfx) =. bpathsfx iaObj
NB.
NB.   NB. calls in object context
NB.   bpathsfx__ST__JODobj WORD_ajod_
NB.   bpathsfx__ST__JODobj MACRO_ajod_

doj=. {:{.DPATH          NB. put dictionary object
fsx=. (;y{JDFILES),IJF  NB. backup file name suffix

NB. backup file path !(*)=. doj
pth=. ". ({.;dncn__doj y),'P__doj'
pth=. (>:pth i: PATHDEL) {. pth

NB. return path and suffix
(pth , (;{:JDSDIRS) , PATHDEL);fsx
)

btextlit=: 3 : 0

NB.*btextlit v-- force any empty backup text to literal datatype.
NB.
NB. To insure that (ed) can always edit  (bget) backup name value
NB. tables force any empty texts to a literal dataype. If this is
NB. not done the result may  fail the name, value argument  tests
NB. of (ed).
NB.
NB. monad:  bt =. btextlit bt

(<'') (<(I. 0 = #&> _1 {"1 y);_1)} y
)

checkback=: 3 : 0

NB.*checkback v-- return list of put dictionary backup numbers.
NB.
NB. monad:  ilbn checkback baObj
NB.
NB.   checkback {:0{DPATH

NB. extant backup numbers errmsg: no backup(s) to restore or search
if. 0=#bn=. bnums BAK__y do. jderr ERR103 else. ok bn end.
)

checkntstamp=: 3 : 0

NB.*checkntstamp v--  checks name, creation  and  last  put  date
NB. arrays.
NB.
NB. The boxed timestamp array fetched  by the _14 option of (get)
NB. is one of the most complex and idiosyncratic JOD results. The
NB. layout was  motivated by  the  need  to  serialize  timestamp
NB. information so that dump scripts might  preserve the creation
NB. and last put date of objects.
NB.
NB. monad:  checkntstamp btNts
NB.
NB.   'rc nts'=. 0 _14 get }. dnl ''
NB.   checkntstamp__ST__JODobj nts

msg=. ERR099 NB. errmsg: invalid name/creation/lastput table
if. badbu y do. jderr msg
elseif. -.2 1 -: $y      do. jderr msg
elseif. badfl uv=. ;1{y  do. jderr msg
elseif. (2 ~: #$uv) +. 2 ~: #uv  do. jderr msg
NB. errmsg: name creation/lastput length mismatch
elseif. ~:/ {:@$&> y  do. jderr ERR100
NB. creation must precede or equal last put
elseif. 0 e. <:/ uv   do. jderr msg
elseif. badrc tn=. checknames ;0{y  do. jderr msg
NB. timestamp names must be unique
elseif. badunique tn=. }.tn  do. jderr msg
NB. dates are in fractional day yyyymmdd.fd format
NB. check that floored numbers are actual Gregorian dates
NB. errmsg: invalid date(s) name/creation/lastput table
elseif. 0 e. valdate datefrnum ,uv  do. jderr ERR101
elseif.do. ok < (<tn) (<0;0)} y  NB. insures deblanked names
end.
)
```

```
checkopen=: 3 : 0

NB.*checkopen v-- are any dictionaries open?
NB.
NB. monad:  checkopen uuIgnore

if. #DPATH do. OK else. jderr ERR050 end. NB. errmsg: no dictionaries open
)

checkpath=: 3 : 0

NB.*checkpath  v-- returns  ok  if the current  path matches  the
NB. current   dictionary's  reference   path.  Path  matching  is
NB. critical to the integrity of groups and suites.
NB.
NB. monad:  checkpath bacl
NB.
NB.   checkpath <'6'   NB. directory object reference

DL=. y  NB. directory object !(*)=. DL
rpath=. ,RPATH__DL
dpath=. ,> 1 {"1 DPATH

if. #rpath do.
  if. rpath -: dpath do. OK else. jderr ERR081 end. NB. errmsg: path mismatch
else.

  NB. dictionary path empty save current path and return ok
  if. badreps (<dpath) jreplace UF__DL;CNRPATH do.
    jderr ERR082 NB. errmsg: unable to set reference path
  else.
    RPATH__DL=:  dpath
    OK
  end.

end.
)

checkput=: 3 : 0

NB.*checkput  v--  is  the  first  path  dictionary a  read/write
NB. dictionary?
NB.
NB. monad:  checkput uuIgnore

if. #DPATH do.
  DL=. 3{0{DPATH  NB. directory object !(*)=. DL

  NB. return directory object reference or errmsg: not a put dictionary
  if. RW__DL do. ok DL else. (jderr ERR051),<DNAME__DL end.
else.
  jderr ERR050
end.
)

closedict=: 4 : 0
```

```
NB.*closedict v-- closes dictionaries. Dictionary names have been
NB. validated prior to calling this verb. Destroys  all directory
NB. objects. The state  of directories on file are maintained  by
NB. other verbs. So no directory updating is required here.
NB.
NB. monad:  closedict blclDictionary
NB.
NB.   closedict 'd0';'d1'   NB. close di

NB. close request seems valid - mark master
if. badrc msg=. markmast 1 do. msg return. end.

NB. destroy open directory objects
uv=. ({."1 DPATH) e. y
if. +./uv do.
  coerase"0 uv#{:"1 DPATH
  DPATH=:  DPATH #~ -. uv
else.
  (jderr ERR080),<y NB. errmsg: not open
end.

NB. update master open status and release
x=. > x
uv=. (0{x) i. y
x=. < (<0) (<3;uv)} x
if. badreps x jreplace JMASTER;CNMFTAB do.
```

```
  jdmasterr ERR077  NB. errmsg: unable to update master
elseif. badrc msg=. markmast~0  do. msg
elseif. do. (ok OK054),y
end.
)

createst=: 3 : 0

NB.*createst v-- storage object creation verb. (y) is the object
NB. locale reference returned by (conew).
NB.
NB. monad: createst uuIgnore
NB.
NB.   createst__ST ST;MK;UT;<SO

NB. object references !(*)=. JOD ST MK UT SO
'JOD ST MK UT SO'=:  y

NB. word and macro type/name class codes !(*)=. HASTYPE
HASTYPE=:  (i. 4),MACROTYPE

NB. brand storage object with unique id !(*)=. JODOBID
saveobid JODOBID=:  didnum 0

NB. inverted data/code component cross reference !(*)=. INCNXR
uv=. CNCLASS,CNCREATION,CNPUTDATE,CNSIZE
1 [ INCNXR=:  (INCLASS,INCREATE,INPUT,INSIZE) ,: uv
)
```

```
defwords=: 4 : 0

NB.*defwords v-- fetches and defines words.
NB.
NB. dyad:  bacl defwords blcl
NB.
NB.   (<'base') defwords ;:'please define my words'

if. badrc y=. checknames y do. y return. end.
wrds=. y=. }.y

NB. if all words are not on path get nothing
if. badrc wnl=. pathnl WORD do. wnl return. end.
wnl=. }. wnl

NB. errmsg: exceeds symbol table limit for locale
if. SYMBOLLIM <: #wnl do. jderr ERR094 return. end.

NB. remove any empty dictionaries from path
b=.   0&<@:#&> wnl
wnl=. b#wnl [ dpath=. b#DPATH

if. *./b=. y e. ; wnl do.

  loc=. locsfx x

  NB. run down the path fetching the first word occurrences
  for_dp. wnl do.
    ix=. (dp=. >dp) i. y

    NB. if any words in current dictionary load them
    if. +./wf=. ix<#dp do.
      if. badrc msg=. (wf#ix) loadwords loc,{:dp_index{dpath do.
        msg return.
      end.

      NB. remove fetched words from list quit if no more words
      if. 0=#y=. (-.wf)#y do. break. end.
    end.
  end.

  NB. test name class of fetched words
  if. 1&e. b=. 0&> nc wrds=. wrds,&.>loc do.
    (jderr ERR085),b#wrds NB. errmsg: words(s) not defined
  else.
    ok (":#b),OK060
  end.

else.
  (jderr ERR083),(-.b)#y NB. errmsg: not on path
end.
)

delstuff=: 4 : 0
```

```
NB.*delstuff v-- deletes words, tests, groups, suites and macros
NB.
NB. dyad:  (iaObject ; il ; bacl) delstuff blcl
NB.
NB.   cn =. CNPUTDATE,CNCREATION,CNEXPLAIN
NB.   (GROUP;cn;<DL) delstuff ;:'we groups are toast'

'obj cn DL'=. x  NB. directory object !(*)=. DL

if. badrc y=. checknames y do. y
elseif. loaddir__DL obj do.
  jderr ERR054 NB. errmsg: unable to load directory
elseif. #ix =.".(>dnix__DL obj),'__DL' do.

  oc=. +/b=. ix e. y=. ~.}.y

  if. oc ~: #y do.
    (jderr ERR086),(-.y e. ix)#y NB. errmsg: not in put dictionary
    return.
  end.

  list=. (b=. -.b)#ix
  comp=. b#".(in=. >dncn__DL obj),'__DL'
  fp=. dfopen__DL in=. {.in

  NB. remove old inverted data from object
  dropinv__DL 0
  NB. delete from inverted lists and main directory
  if. badrc msg=. b invdelete fp;cn do.
    msg [ dfclose__DL in return.
  elseif. badrc msg=. (obj,fp) savedir__DL list;comp do.
    msg [ dfclose__DL in return.
  end.

  NB. remove any put dictionary word references
  if. WORD=obj do.
    if. badrc msg=. DL delwordrefs y do. msg [ dfclose__DL in return. end.
  end.

  dfclose__DL in
  msg=. ' ',>dnnm__DL obj
  (ok (": oc),msg,OK061),<DNAME__DL

elseif.do.
  jderr ERR087 NB. errmsg: nothing in put dictionary
end.
)

delwordrefs=: 4 : 0

NB.*delwordrefs v-- deletes word references. Word reference
NB. deletion is required when deleting words to insure that words
NB. do not leave "reference shadows." A reference shadow occurs
NB. when a word with references is deleted and moved to a
NB. dictionary further down  on the path. The reference reporting
NB. mechanism picks  up  the shadow and never fetches the  actual
NB. reference list. Words are the  only  JOD  objects with stored
NB. references.
NB.
NB. dyad:  ba delwordrefs blclWords

DL=. x NB. directory object !(*)=. DL

NB. errmsg: unable to load references
if. loadref__DL WORD do. jderr ERR079
elseif.do.

  NB. find any references to deleted words
  uv=. WORDREFIX__DL e. y
  if. +./uv do.

    dfopen__DL 'U'
    fp=. UP__DL

    NB. remove any references from put dictionary
    uv1=. (uv=. -.uv)#WORDREFIX__DL
    uv2=. uv#WORDREFCN__DL

    NB. update reference directory and close
    if. badrc msg=. (WORD,fp) saveref__DL uv1;uv2 do. msg [ dfclose__DL 'U' return. end.
    dfclose__DL 'U'
  end.

  OK
end.
)

didstats=: 4 : 0

NB.*didstats v-- dictionary statistics. Returns a table of object
NB. counts and reference paths for each dictionary in path order.
NB.
NB. dyad:  uuIgnore didstats uuIgnore

NB. are any dictionaries open?
if. badrc uv=. checkopen 0 do. uv return. end.

NB. gerund of directory object (loadstamps) calls
ger=. (<'loadstamps') ,&.> locsfx ol=. {:"1 DPATH
if. +./(ger `:0) 0 do.
  jderr ERR054  NB. errmsg: unable to load directory
else.
  dn=. DIRTS__oj [ oj=. {. {:"1 DPATH  NB. (*)=. oj
  hd=. '' ; '--' ; HEADNMS__oj

  NB. collect values of directory object nouns
  uv=.('RW';'RPATH';dn) fullmonty&><ol
  rpaths=. 1{uv [ rs=. (;{.uv){READSTATS
  dt=. ({."1 DPATH) ,. rs ,. {.&> |: 2 }. uv
  dt=. hd , dt

  NB. read master to get as complete a list of names and numbers
  NB. as possible. Some (DIDNUM)'s may still be missing - missing
  NB. dictionaries reported as dictionary numbers - hey life is cruel!
  if. badjr uv=. jread JMASTER;CNMFTAB do.
    jderr ERR006 return. NB. errmsg: cannot read master
  end.

  rb=. <"0 (~. ;rpaths) -. ;1{uv =. >uv
  hd=. (0{uv) , ":&.> rb  NB. all dictionary names
  dn=. (1{uv) , rb        NB. dictionary numbers

  NB. display formatted paths with each dictionary using current names
  rpaths=. ;&.>PATHSHOWDEL,L:0((<;dn)i.&.>rpaths){&.><hd
  ok <dt ,. PATHTIT ; rpaths
end.
)

dnlsearch=: 4 : 0

NB.*dnlsearch  v--  searches  dictionary name  lists  for  simple
NB. character list patterns.
NB.
NB. dyad: ilObjOptNc dnlsearch (clPattern ; clDir)
NB.
NB.   3 2 7 dnlsearch 'boo'   NB. suite names containing 'boo'
NB.   3 _2 0 dnlsearch 'boo'  NB. nouns with names containing 'boo'

mop=. ERR001
if. -.(second x) e. PATOPS do. jderr mop return. end.

NB. following code is essentially (pathnl) - maintained
NB. inline because (pob) and (oj) used elsewhere
pob=. {:"1 DPATH [ dt=. |{.x
if. badrc msg=. dt loadalldirs pob do. msg return. end.
nl=. (>dnix__oj dt) fullmonty pob [ oj=. {.pob  NB. (*)=. oj

if. DEFAULT~:{:x do.
   NB. object noun !(*)=. HASTYPE
   if. (({.x) e. WORD,MACRO) *. ({:x) e. HASTYPE do.
     ger=. (<'loadnc') ,&.> locsfx pob
     if. +./(ger `:0) dt do.
       jderr ERR054 return. NB. errmsg: unable to load directory
     end.
     dc=. ;&.> (>dnnc__oj dt) fullmonty pob

     NB. remove items of (nl) that do not have type ({:x)
     nl=. (dc =&.> <{:x) #&.> nl

   else.
     jderr mop return.
   end.
end.
x=. second x
if. isempty y do.
  if. 0>x do. ok (/:~)&.> nl return. else. ok sortdnub nl end.
elseif. do.
  y=. ,y
  NB. insure nulls behave
  sublists=. 0>x
  shape=. (sublists#0),0
  nl=. (<shape$'') (I. 0=#&> nl)} nl
  NB. remove any empties
  if. 0=#nl=. nl -. a: do. ok'' return. end.
  select. |x
    case. 1 do. if. sublists do. ok nl allnlpfx y else. ok nl nubnlpfx y end.
    case. 2 do. if. sublists do. ok nl allnlctn y else. ok nl nubnlctn y end.
    case. 3 do. if. sublists do. ok nl allnlsfx y else. ok nl nubnlsfx y end.
    case. do. jderr mop
  end.
end.
)

NB. select only duplicate names in table based on first column
dupnames=: ] #~ (0 { "1 ]) e. (0 { "1 ]) #~ [: -. [: ~: 0 { "1 ]

freedisk=: 3 : 0

NB.*freedisk v-- returns free disk/volume space in bytes.
NB.
NB. monad:  freedisk clDisk | clLinuxVolume
NB.
NB.   freedisk 'c:\'       NB. :\ required for windows
NB.   freedisk '/sd1/dev'  NB. linux file system root - null sums all devices

NB. NOTE: assume enough space for IOS, Android and unknown?

NB. Default behaviour has been changed to not size volumes
NB. when FREESPACE is 0. Volume sizing can peform poorly
NB. on large network volumes and fail completely on cloud drives.
NB. Empty JOD dictionaries are small (<60k) - assuming sufficient
NB. space is safe in all but extreme circumstances.
if.     0=FREESPACE      do. 1
elseif. IFWIN            do. freediskwin y
elseif. UNAME-:'Linux'   do. freedisklinux y
elseif. IFIOS            do. >:FREESPACE
elseif. UNAME-:'Darwin'  do. freediskmac y
elseif. UNAME-:'Android' do. >:FREESPACE
elseif.do. >:FREESPACE
end.
)

freedisklinux=: 3 : 0

NB.*freedisklinux v-- bytes free on not 'none' linux volumes.
NB.
NB. NOTE: NIMP: I don't know how to determine which
NB. linux volume the dictionary will be on so I return
NB. the minimum of all not 'none' mounted volumes.
NB.
NB. monad:  fl =. freedisklinux uuIgnore
NB.
NB.   freedisklinux 0   NB. bytes (possibly floating) free on mounted filesystems

NB. linux shell command fetches free 1k blocks - expected format is:
NB. Filesystem           1K-blocks      Used Available Use% Mounted on
NB. /dev/sda1            149301564  11113004 130604408   8% /
NB. none                    764396       648    763748   1% /dev
NB. none                    771004      1364    769640   1% /dev/shm
NB. none                    771004        96    770908   1% /var/run
NB. none                    771004         0    771004   0% /var/lock
txt=. host 'df -l'

NB. cut into lines and drop header
txt=. }. <;._2 txt

NB. remove all 'none' filesystems HARDCODE: length of 'none'
NB. NIMP: ignoring empty result - hey there
NB. has to be at least one mounted filesystem!
txt=. txt #~ -. 'none'&-:&> 4 {.&.> txt

NB. min bytes free using 1000 byte blocks - this will
NB. underestimate free space and leave a little safety cushion
<./ 1000 * 3 {"1 ] _1&".&> txt
```

```
)

freediskmac=: 3 : 0

NB.*freediskmac v-- free disk bytes on mac dictionary volume.
NB.
NB. monad:  iaBytes =. freediskmac clMacVolume

NB. NIMP: assume enough space for now
>:FREESPACE
)

freediskwin=: 3 : 0

NB.*freediskwin v-- returns free disk/volume space in btyes for win systems
NB.
NB. monad:  freediskwin clDisk | clLinuxVolume
NB.
NB.   freediskwin 'c:\'       NB. :\ required for windows

s=. 'kernel32 GetDiskFreeSpaceA i *c *i *i *i *i' cd y;(,0);(,0);(,0);(,0)
*/ ; 2 3 4 { s
)

NB. returns lists of directory object noun values - see long documentation
fullmonty=: [: ".&.> ([: < [) ,&.> [: locsfx ]
```

```
getdicdoc=: 3 : 0

NB.*getdicdoc v-- fetches put dictionary documentation.
NB.
NB. monad:  cl =. getdicdoc uuIgnore

NB. assumes a put dictionary is open.
DL=. {:{.DPATH  NB. directory object !(*)=. DL
if. badjr dat=. jread WP__DL;CNDICDOC do. jderr ERR088 NB. errmsg: read failure
else.
  ok ,>dat
end.
)

getdocument=: 4 : 0

NB.*getdocument v-- get object documentation
NB.
NB. dyad: iaObject getdocument blcl

if. badrc uv=. (x,1) getobjects y do. uv else. ok <0 3 {"1 rv uv end.
)

getexplain=: 4 : 0

NB.*getexplain v-- gets short explanations.
NB.
NB. Note: Similar  to (invfetch)  and  (getobjects) but different
NB. enough to justify new verb.
NB.
NB. dyad:  iaObject getexplain blcl
NB.
NB.   WORD getexplain ;:'you have some explaining to do'

if. badrc y=. checknames y do. y return. end.
obs=. y=. }.y
if. badrc tnl=. pathnl x do. tnl return. end.

NB. remove any empty dictionaries from path
tnl=. }. tnl
b=. 0&<@:#&> tnl
tnl=. b#tnl [ dpath=. b#DPATH

NB. if all objects are not on path get nothing
if. *./b=. y e. ; tnl do.

  DL=.  {:{:DPATH                    NB. any object
  fp=.  ({.>dncn__DL {.x),'P__DL'  NB. file pointer
  res=. (#obs)$a:                    NB. result list

  NB. run down path
  for_dp. tnl do.
    ix=. (dp=. >dp) i. y
    NB. get data in current dictionary
    if. +./bm=. ix<#dp do.
      DL =. {:dp_index{dpath  NB. directory object !(*)=. DL
      if. badjr dat=. jread (".fp);CNEXPLAIN do.
        jderr ERR088 return. NB. errmsg: read failure
      end.
      dat=. (bm#ix){>dat

      NB. merge data into final result order matters here
      res=. dat (obs i. bm#y)} res

      NB. remove fetched objects from list quit if no more
      if. 0=#y=. (-.bm)#y do. break. end.
    end.
  end.

  NB. return objects in requested order
  ok <obs ,. res

else.
  (jderr ERR083),y #~ -. b  NB. errmsg: not on path
end.
)

getgstext=: 4 : 0

NB.*getgstext v-- get group and suite text.
NB.
NB. dyad:  iaObject getgstext blcl

if. badrc uv=. (x,0) getobjects y do. uv else. ok <0 1 {"1 rv uv end.
)


getntstamp=: 4 : 0

NB.*getntstamp v-- get name, creation and last put timestamps.
NB.
NB. dyad:  iaDcode getntstamp blcl
NB.
NB.    1 getntstamp__ST__JODobj }. 1 revo  ''

if. badrc uv=. (x,INCREATE,INPUT) invfetch y do. uv else. ok <(<y) ,: 1{uv end.
)


getobjects=: 4 : 0

NB.*getobjects  v-- fetches object names and values. A successful
NB. result  is  a  boxed  table. Column  0 holds  names remaining
NB. columns hold types and values. If there  is no  type or  name
NB. class only two columns are returned.
NB.
NB. dyad:  il getobjects blcl
NB.
NB.   NB. 2 columns (name,value)
NB.   (TEST,0) getobjects ;:'some test names ehh'
NB.
NB.   NB. 3 columns (name,class,value)
NB.   (WORD,0) getobjects ;:'words are us'

if. badrc y=.checknames y do. y return. end.
ord=. y=. }.y

'obj offset'=. x
if. badrc onl=. pathnl obj do. onl return. end.

NB. remove any empty dictionaries from path
onl=. }. onl
b=. 0&<@:#&> onl
onl=. b#onl [ dpath =. b#DPATH
val=. 0 0$''

NB. if all objects are not on path get nothing
if. *./b=. y e. ; onl do.

  doj=. {: {.dpath              NB. any directory object
  cnn=. (uv=. >dncn__doj obj),'__DL'  NB. object component noun name
  fp=. ({.uv),'P__DL'           NB. file pointer noun name

  NB. run down the path fetching first occurrences
  for_dp. onl do.
    ix=. (dp=. >dp) i. y
    NB. NIMP GETFACTOR not used yet
    NB. get any objects in current dictionary
    if. +./wf=. ix<#dp do.
      DL=. {:dp_index{dpath  NB. directory object !(*)=. DL
      if. badjr dat=. jread (".fp);(wf#ix){offset+".cnn do.
        jderr ERR088 return. NB. errmsg: read failure
      end.
      val=. val , >dat

      NB. remove fetched objects from list quit if no more objects
      if. 0=#y=. (-.wf)#y do. break. end.
    end.
  end.

  NB. insure objects are returned in requested order
  val=. (({."1 val) i. ord) { val
  ok <val

else.
  (jderr ERR083),(-.b)#y NB. errmsg: not on path
end.
)

getrefs=: 4 : 0

NB.*getrefs  v-- fetches reference lists. A  successful result is
NB. an OK  boxed table of  boxed  character lists. Column 0 holds
NB. names  and column  1  holds  boxed reference lists. Currently
NB. only words have stored  references  but  this verb  has  been
NB. coded to  allow  for additional  reference types  as the need
NB. arises.
NB.
NB. dyad:  iaObject getrefs blcl
NB.
NB.   WORD getrefs ;:'get our references please'

if. badrc y=.checknames y do. y return. end.
y=. }.y

NB. if all objects are not on path get nothing
if. badrc onl=. pathnl x do. onl return. end.
if. 0 e. b=. y e. ; }.onl do.
  (jderr ERR083),(-.b)#y return.  NB. errmsg: not on path
end.

NB. reference table
rft =. i. 0 0

NB. objects with stored references
if. badrc onl=. pathref x do. onl return. end.

NB. remove dictionaries with no references from path
onl=. }. onl
b=. 0&<@:#&> onl
onl=. b#onl [ dpath =. b#DPATH
```

```
NB. if any stored references get them
if. #dpath do.

  NB. reference component noun name in directory object
  DL=. {:{. dpath
  cnn=. >0 dnrn__DL x

  NB. run down the path fetching the first occurrences
  for_dp. onl do.
    rix=. (dp=. >dp) i. y

    NB. NIMP GETFACTOR not used yet
    NB. if any references in current dictionary get them
    if. +./rf=. rix<#dp do.
      DL=. {:dp_index{dpath  NB. directory object !(*)=. DL
      if. badjr dat=. jread UF__DL;(rf#rix){".cnn,'__DL' do.
        jderr ERR088 return.  NB. errmsg: read failure
      end.
      rft=. rft , >dat

      NB. remove names with fetched references from list quit if no more
      if. 0=#y=. (-.rf)#y do. break. end.
    end.
  end.
end.
```

```
NB. any remaining objects currently have no stored references
if. #y do. ok <rft , (y ,"0 1 <x),.<'';'' else. ok <rft end.
)

gslistnl=: 4 : 0

NB.*gslistnl v-- returns  a group or  suite name list.  Prior  to
NB. calling this verb a  dictionary  must be  open  and  the (x)
NB. object code argument validated. The name list returned is the
NB. first one found on the current path.
NB.
NB. dyad:  iaObject gslistnl clName
NB.
NB.  GROUP gslistnl 'groupname'

if. badrc path=. pathnl x do. path return. end.
uv=. (path=. }.path) fopix y

if. uv=#path do. (jderr ERR083),<y return. end. NB. errmsg: not on path

uv=. {:uv{DPATH  NB. directory object reference (*)=. uv

cn=. (".(ln=. >dnix__uv x),'__uv') i. <y
cn=. cn { ".(>dncn__uv x),'__uv'    NB. file component of list

if. badjr cn=. jread (".({.ln),'P__uv');cn do.
  jderr ERR084 NB. errmsg: unable to read data
else.
```

```
  ok >{:>cn  NB. stored list is unique and sorted
end.
)

inputdict=: 4 : 0

NB.*inputdict v-- tests for objects in put dictionary
NB.
NB. dyad:  (iaObject ;< ba) inputdict blcl
NB.
NB.  (WORD;<DL) inputdict ;:'are we in put dictionary'

'obj DL'=. x  NB. directory object !(*)=. DL

NB. errmsg: unable to load directory
if. loaddir__DL obj do. jderr ERR054
elseif. ix=. ".(>dnix__DL obj),'__DL'
  */b=. y e. ix do. OK
elseif.do.
  (jderr ERR086),(-.b)#y NB. errmsg: not in put dictionary
end.
)

invappend=: 4 : 0

NB.*invappend v-- appends items to inverted data lists.  The (x)
NB. argument is a boxed list append  list. (y)  is  a boxed list
NB. containing a file pointer and inverted component numbers.
NB.
NB. dyad:  blul invappend blul
NB.
NB.   apps invappend WF__DL ; CNCLASS,CNPUTDATE,CNSIZE

msg=. ERR057  NB. errmsg: directory update failure

NB. file pointer & component list
'fp cmpl'=. y
if. (#x)~:#cmpl do. jderr msg return. end.
rc=. i.0

NB. get the total number of expected elements from 0 component
if. badjr dat=. jread fp;CNMARK do. jderr msg return. end.
oldlen=. >{.>dat

NB. loop for maximum safety and space savings
for_cn. cmpl do.

  if. badjr dat=. jread fp;cn do. jderr msg return. end.
  dat=. >dat

  NB. all inverted list lengths must match expected
  if. oldlen ~: #dat do. jderr msg return. end.

  dat=. dat , >cn_index{x
  rc=. rc, (<dat) jreplace fp ; cn
end.

NB. test replacements for errors
if. badreps rc do. jderr msg else. OK end.
)

invdelete=: 4 : 0

NB.*invdelete v--  deletes items  from  inverted data lists.  The
NB. (x)  argument  is a mask list. (y) consists of a boxed list
NB. containing a file pointer and inverted component numbers.
NB.
NB. dyad:  pl invdelete blul
NB.
NB.   mask invdelete WF__DL ; CNCLASS,CNPUTDATE,CNCREATION,CNSIZE

NB. file pointer & component list
'fp cmpl'=. y
msg=. ERR057  NB. errmsg: directory update failure
rc=. i.0 [ len=. #x

NB. get the total number of expected elements from 0 component
if. badjr dat=. jread fp;CNMARK do. jderr msg return. end.
oldlen=. >{.>dat

NB. loop for maximum safety and space savings
for_cn. cmpl do.
```

```
  if. badjr dat=. jread fp;cn do. jderr msg return. end.
  dat=. >dat

  NB. all inverted list lengths must match expected
  if. oldlen ~: #dat do. jderr msg return. end.

  rc=. rc, (<x#dat) jreplace fp;cn
end.

NB. test replacements for errors
if. badreps rc do. jderr msg else. OK end.
)

invfetch=: 4 : 0

NB.*invfetch v-- reads inverted  numerical  data  lists  from
NB. dictionary  files.  Assumes  the  (x)   argument   has  been
NB. validated prior to calling.
NB.
NB. dyad:  ilDcodes invfetch blcl
NB.
NB.   NB. first code is JOD object code
NB.   0 12 13 14 15 invfetch__ST__JODobj }. dnl''
NB.   2 13 14  invfetch__ST__JODobj }. 2 dnl''
NB.   (SUITE_ajod_,INCREATE_ajod_,INPUT_ajod_) invfetch__ST__JODobj }. SUITE_ajod_ dnl''

if. badrc y=. checknames y do. y return. end.
```

```
obs=. y=. }.y
if. badrc tnl=. pathnl {.x do. tnl return. end.

NB. remove any empty dictionaries from path
tnl=. }. tnl
b=. 0&<@:#&> tnl
tnl=. b#tnl [ dpath=. b#DPATH

NB. if all objects are not on path get nothing
if. *./b=. y e. ; tnl do.

  NB. map external codes to inverted data components
  cninv=. ((0{INCNXR) i. }.x) { 1{INCNXR   NB. object noun !(*)=. INCNXR
  DL=.    {:{:DPATH                     NB. any object
  fp=.    ({.>dncn__DL {.x),'P__DL'     NB. file pointer
  res=.   ((#cninv),#obs)$0             NB. result table

  NB. run down path
  for_dp. tnl do.
    ix=. (dp=. >dp) i. y

    NB. get data in current dictionary
    if. +./bm=. ix<#dp do.
      DL =. {:dp_index{dpath  NB. directory object !(*)=. DL
      if. badjr dat=. jread (".fp);cninv do.
        jderr ERR088 return. NB. errmsg: read failure
      end.
      dat=. (bm#ix) {"1 > dat

      NB. merge data into final result order matters here
      res=. dat (<a:;obs i. bm#y)} res

      NB. remove fetched objects from list quit if no more
      if. 0=#y=. (-.bm)#y do. break. end.
    end.
  end.

  NB. returns a list when only one item otherwise table
  ok < ]`,@.(1&=@:#) res

else.
  (jderr ERR083),y #~ -. b  NB. errmsg: not on path
end.
)

invreplace=: 4 : 0

NB.*invreplace v-- replaces  items  from inverted data lists. The
NB. (x) argument is a boxed list of positions and  replacements.
NB. (y) is a  boxed list containing a file pointer and  inverted
NB. component numbers.
NB.
NB. dyad:  blul invreplace blul
NB.
NB.   (pos;reps) invreplace WF__DL ; CNCLASS,CNPUTDATE,CNSIZE
```

```
msg=. ERR057  NB. errmsg: directory update failure

NB. file pointer & component list
'fp cmpl'=.  y
'pos repl'=. x
if. (#repl)~:#cmpl do. jderr msg return. end.
rc=. i.0

NB. replacements do not change the length of inverted lists
NB. get the total number of elements from 0 component
if. badjr dat=. jread fp;CNMARK do. jderr msg return. end.
len=. >{.>dat

NB. loop for maximum safety and space savings
for_cn. cmpl do.

  if. badjr dat=. jread fp;cn do. jderr msg return. end.
  dat=. >dat

  NB. all inverted list lengths must match
  if. len ~: #dat do. jderr msg return. end.

  dat=. (>cn_index{repl) pos} dat
  rc=. rc, (<dat) jreplace fp ; cn
end.
```

```
NB. test replacements for errors
if. badreps rc do. jderr msg else. OK end.
)


NB. 1 if dictionary is a library
islib=: '*'"_ = [: {. [: > {.

iswriteable=: 3 : 0

NB.*iswriteable  v--  tests  a blcl of full  path file  names for
NB. writeablity.
NB.
NB. This verb takes  a list of full path file names and tests the
NB. read/write status of  the files.  The  result is boolean list
NB. with 1 denoting "writeable" and 0 denoting "not-writeable."
NB.
NB. monad:  pl =. iswriteable blclPathFile

if. IFWIN do. iswriteablewin y else. iswriteablelinux y end.
)

iswriteablelinux=: 3 : 0

NB.*iswriteablelinux  v-- tests a blcl of full path  linux  files
NB. for writeablity.
NB.
NB. monad:  pl =. iswriteablelinux blclPathFile
NB. NIMP: check linux file read/write/access status
NB. NIMP: returns all 1's for now
(#,y)#1
)

NB. tests permissions/attributes of a blcl of full path file names for writeablity
iswriteablewin=: 'w-'"_ -:"1 [: ] 1 3"_ { "1 [: ;"1 [: ] _2: {."1 [: > [: ,&(1!:0)&.> ]

jdatcreate=: 4 : 0

NB.*jdatcreate v-- creates an empty dictionary data file. (y) is
NB. a path and (x) is a file name
NB.
NB. dyad:  clFile jdatcreate clPath
NB.
NB.   'jtests' jdatcreate 'c:\temp\jdict2a\'
NB.   'jgroups' jdatcreate 'c:\blanks are cool\jdict 2a\'

fn=. (alltrim y) , x -. ' '
msg=. ERR052 NB. errmsg: unable to initialize

if.     -.jcreate fn do. (jderr msg),<fn
elseif. c=. < 0 ; t=. now ''          NB. length and directory stamp
        c=. c , <''                   NB. c1 RESERVED
        badappend c=. (c , (OFFSET-#c) # a:) jappend fn do. (jderr msg),<fn
elseif. do.
```

```
 ok {: c  NB. return last component
end.
)

jwordscreate=: 4 : 0

NB.*jwordscreate v-- creates an empty word file. (y) argument is
NB. a  fully  qualified  file  name.  (x)  is  a  boxed list  of
NB. dictionary  creation  parameters.  The  target  directory  is
NB. assumed to exist. Result is a return code and message.
NB.
NB. dyad:  blParms jwordscreate clFile
NB.
NB.  (doc;parms) jwordscreate 'c:\temp\jdict2a\jwords' NB. no extension

msg=. ERR052 NB. errmsg: unable to initialize

if.     -.jcreate y  do. (jderr msg),<y
elseif. c=. < 0 ; t=. now ''        NB. c0 length and directory stamp
        c=. c , <''                 NB. c1 pack count and last backup or restore timestamp.
        c=. c , 0{x                 NB. c2 this dictionary's documentation
        c=. c , <}. x               NB. c3 dictionary parameters
        a=. badappend c=. (c , (OFFSET-#c) # a:) jappend y
        NB. store J version string that created this dictionary
        b=. badjr (<9!:14'') jreplace y;CNJVERSION
        a +. b do. (jderr msg),<y
elseif. do.
 ok {: c  NB. return last component
end.
)

loadalldirs=: 4 : 0

NB.*loadalldirs v-- loads all (x) directories for each open (y)
NB. dictionary.
NB.
NB. dyad:  iaObject loadalldirs blcl
NB.
NB.   WORD loadalldirs {:"1 DPATH

x=. |x
for_oj. y do.
  if. loaddir__oj x do.
    jderr ERR054 return. NB. errmsg: unable to load directory
  end.
end.
OK
)

loadallrefs=: 4 : 0

NB.*loadallrefs v-- loads all references for (y) dictionary.
NB.
NB. dyad:  iaObject loadallrefs blcl
NB.
NB.   WORD loadallrefs {:"1 DPATH

for_oj. y do.
  if. loadref__oj x do.
    jderr ERR079 return. NB. errmsg: unable to load references
  end.
end.
OK
)

loadwords=: 4 : 0

NB.*loadwords v-- loads dictionary words into target locales.

DL=. {: y  NB. obfuscate (/:)=: directory object !(*)=. DL

NB. NIMP GETFACTOR not used yet
NB. read words and determine name class
if. badjr wu=. jread WF__DL;x{WORDCN__DL do.
  jderr ERR088  NB. errmsg: read failure
else.
  bu=.   0 ~: ; 1&{&> wu
  loc=. >{. y  NB. target locale

  NB. define words that are not nouns
  NB. NIMP may be able to speed things up by switching
  NB. to target locale in top of script and then switching
  NB. back to current - eliminates need to hard wire target locale
  NB. to each word.

  try.
    if. #vu=. bu#wu do.
      0!:0  ; (({.&> vu) ,&.> <loc,'=:') ,&.> ({:&> vu) ,&.> <LF
    end.

    NB. define nouns - override mixed assignments (<:)=:
    if. #nu=. (-.bu)#wu do.
      vu=. ({.&> nu) ,&.> <loc
      (vu)=:  (3!:2)&.> {:&> nu
    end.
  catch. jderr ERR091 return. end.
  OK
end.
)

mainddir=: 3 : 0

NB.*mainddir v--  creates  the main  dictionary  directory from a
NB. path.
NB.
NB. monad:  mainddir clPath
NB.
NB.   mainddir 'c:\go\ahead\create\my\directory'

NB.#ASSERT 0 < #y.
y=.  (-PATHDEL={: y) }. y , PATHDEL
```

```
drv=. alltrim (,&':'`]@.(0&=@:#)) justdrv y

NB. standard path format
sp=. alltrim justpath y
y=. drv,sp,PATHDEL

NB. path must begin with (PATHDEL) to force user to
NB. think carefully about where dictionary is placed
if. PATHDEL~:{. sp,' ' do.
  jderr ERR059 NB. errmsg: full rooted path required
  return.
end.

NB. subpath list with any drive attached
sp=. ;&.> <"1 ,/\ <;.1 sp
sp=. (<drv) ,&.> sp

NB. attempt to create last directory on path
if. 1=makedir {: sp do. ok y

NB. upon failure run down paths attempting to create all
NB. intermediate directories - many operations will
NB. typically fail because some intermediates will exist
elseif. makedir"0 }: sp
        1=makedir {: sp do. ok y
elseif. do.
  (jderr ERR060),<y NB. errmsg: unable to create directory
```

```
end.
)

mnlsearch=: 4 : 0

NB.*mnlsearch v-- master name list search.
NB.
NB. dyad:  ilOpt mnlsearch clNamePattern

NB. ERR006 cannot read master
if. badjr d=. >jread (JMASTER,IJF);CNMFTAB do. jderr ERR006 return. end.

NB. ERR104 no registered dictionaries
if. 0 e. $d do. jderr ERR104 return. end.
if. fex f=. (tslash2&.> 2{d) ,&.> <(;(0{x){JDFILES),IJF do.
  r=. 0 2$<'' [ y=. ,y
  g=. (<: |1{x){nlpfx`nlctn`nlsfx

  NB. read class if not default and WORD or MACRO
  b=. ((0{x) e. WORD,MACRO) *. DEFAULT ~: 2{x

  for_i. i.#f do.
    o=. i{f [ n=. i{0{d

    NB. ERR088  jfile read failure
    if. badjr p=. >jread o;CNLIST do. jderr ERR088 return. end.
    if. b do.
      if. badjr s=. >jread o;CNCLASS do. jderr ERR088 return. end.
      p=. p #~ s = 2{x
    end.

    if. 0=#p do. continue. end.
    r=. r , (p (g `: 6) y) ,. n
  end.
  r=. /:~ r
  if. 0 > 1{x do. ok <dupnames r else. ok <r end.
else.
  b=. (1:@(1!:4) ::0:) f
  (jderr ERR073) , f #~ -. b
end.
)

newdparms=: 3 : 0

NB.*newdparms  v--  sets  the dictionary  parameters  for  a  new
NB. dictionary.
NB.
NB. monad:  newdparms bluu
NB.
NB.   newdparms sd;dp;dname;dn;path

NB. subdirectories, parameters, name, unique number and path
'sd dp name dn path'=. y

NB. name, number, creation, last dump, [paths], J version, J system
uv=. name ; dn ; (now '') ; (6#0) ; (<path) ,&.> sd ,&.> PATHDEL
```

```
uv=. uv , (9!:14'');9!:12 ''

NB. dictionary number path context - empty until references created
uv=. uv , <i.0

NB. reduce user parameter table to names and values
uv , < |: 0 2 {"1 dp
)

newregdict=: 4 : 0

NB.*newregdict v-- creates a new dictionary  or registers an extant
NB. dictionary.
NB.
NB. dyad:  iaOptions newregdict (clDictionary ; clPathroot)
NB.
NB.   NB. register extant dictinary
NB.   0 newregdict 'dictionary name';'c:\where\it\lives' NB. drive required
NB.
NB.   NB. create new dictionary
NB.   1 newregdict 'new name';'c:\new\location'

mf=.  JMASTER  NB. master file
msg=. ERR061   NB. errmsg: invalid dictionary name;path[;documentation]

if. (badbu y) +. 1~:#$ y  do. jderr msg
elseif. (3<#y) +. 2>#y    do. jderr msg
elseif. +./badcl&> y      do. jderr msg
elseif.do.

  NB. names and paths cannot be empty - sorry
  'name path doc'=. 3{.y,<''
  name=. alltrim name [ path=. hostsep alltrim path
  if. 0&e. (#name),#path do. jderr msg return. end.

  NB. restrict dictionary name and path characters
  if. 0&e. name e. ' ',ALPHA do.
    jderr ERR062 return. NB. errmsg: invalid characters in name
  elseif. 0&e. path e. PATHCHRS,ALPHA do.
    jderr ERR063 return. NB. errmsg: invalid characters in path
  end.

  if. IFWIN do.
    NB. check for UNC paths
    if. (2#PATHDEL) -: 2{.path do.
      NB. insure UNC paths are terminated
      path=. path,(PATHDEL={:path)}.PATHDEL

      NB. NIMP: NOTE: (freedisk)'ing windows network drives (more
      NB. than once) is time consuming and typically unnecesary!
      NB. These volumes are often huge and JOD empty dictionaries
      NB. are tiny - hence we ASSUME sufficient space. The following
      NB. commented code tests UNC volumes.

      disk=.'' NB. empty disk suppresses space testing
      NB. test if the maximum size of subpaths exceeds threshold
      NB. depends on (freedisk) returning zero for invalid paths
      NB. omit root \\ and last nonexistant path
      NB. if. 0=#uv=. _1 }. 2 }.;&.> <"1 ,/\ <;.2 path do.
      NB.   (jderr ERR065),<path return. NB. errmsg: not enough space on drive
      NB. end.
      NB. if. (>./freedisk&> uv) < FREESPACE do. (jderr ERR065),<path return. end.
    else.
      NB. check for windows drive letter (required) and
      NB. determine if there is enough space on the target drive
      NB. errmsg: target drive is required
      if. isempty tdrv=. justdrv path do. jderr ERR064  return. end.

      NB. windows drive letters
      disk=. tdrv,':',PATHDEL
    end.
  else.
    NB. require rooted linux paths
    if. PATHDEL ~: {.path do. (jderr ERR096),<path return. end.

    NB. NIMP: how does one determine the volume name for a given
    NB. fully qualified linux file that resides on said volume?
    disk=. path
  end.

  if. (x=1) *. 0<#disk do. NB. HARDCODE (x) option
```

```
  bytes=. freedisk disk
  NB. errmsg: not enough space
  if. bytes < FREESPACE do. (jderr ERR065),<disk return. end.
end.

NB. attempt to read master
if. badjr uv=. jread mf;CNMFTAB,CNMFPARMS,CNMFDLOG do.
  jderr ERR006 return. NB. errmsg: cannot read master
end.

NB. mark master - this verb updates if successful
NB. all error calls should use (jdmasterr) until
NB. the master is cleared at the end of this verb
if. badrc msg=. markmast 1 do. msg return. end.

NB. master table, dictionary parameters, number log
'mdt dpt dlg'=. uv
NB. errmsg: dictionary name in use
if. (<name) e. 0{mdt do. jdmasterr ERR066 return. end.

if. x=1 do.
  NB. attempt to create main root directory
  if. badrc path=. mainddir path do. path [ markmast~0 return. end.

  NB. attempt to create standard subdirectories
  path=. {: path
  if. 0&e. uv=. makedir"0 path ,&.> JDSDIRS do.
```

```
    jdmasterr ERR067 return. NB. errmsg: unable to create subdirectories
  end.
  path=. > path

  dn=. didnum 0   NB. unique dictionary number
  uv=. newdparms JDSDIRS;dpt;name;dn;path

  NB. create empty dictionary files
  uv=. <(doc;uv) jwordscreate path,>0{JDFILES
  uv=. uv , (}.JDFILES) jdatcreate&.> <path
  if. 0&e. ;{.&> uv do.
    jdmasterr ERR068 return. NB. errmsg: unable to setup dictionary file(s)
  end.
  newmdt=. mdt,.name;dn;path;0
  okm=.OK050
else.
  path=. (-PATHDEL={:path) }. path,PATHDEL

  NB. test existence of dictionary files
  fn=. JDFILES ,&.> <IJF
  if. 1 e. uv=. -. fex"1 dcfiles=. <@:;"1 (<path) ,"0 / fn do.
    (jdmasterr ERR073),<name return. NB. errmsg: missing dictionary file(s)
  end.

  NB. NIMP should run under a trap here to protect
  NB. against files that appear to be dictionary but are not
  NB. NOTE: this component will hold a J version string for J's
  NB. after 9.04. For dictionaries created with older J versions it
  NB. is either empty or holds a version string. This redundant
  NB. storage of the creator version is to get around binary
  NB. incompatibilities of extended precision integers.
  if. badjr dicver=. jread (file=. path,>{.JDFILES);CNJVERSION do.
    NB. errmsg: jfile read failure
    jdmasterr ERR088 return.
  elseif. dicver=. (>dicver) jvn 0
          bck0=. (dicver < JEPOCHVER)  *. JVERSION < JEPOCHVER  NB. old dict, old j OK
          bck1=. (dicver < JEPOCHVER)  *. JEPOCHVER <: JVERSION NB. old dict, new j OK
          bck2=. (JEPOCHVER <: dicver) *. JEPOCHVER <: JVERSION NB. new dict, new j OK
          bck0 +. bck1 +. bck2 do. OK
  elseif. (JEPOCHVER <: dicver) *. JVERSION < JEPOCHVER  do.  NB. new dict, old J BAD
    NB. errmsg: cannot register binary incompatible dictionary
    (jdmasterr ERR108),name;dicver;JVERSION
    return.
  end.

  NB. NOTE: this read fails when J's prior to 9.04 attempt to
  NB. read the parameters created by a j 9.04 system. The stored
  NB. extended integers are not compatibile for older J's
  NB. read dictionary parameter table and documentation
  if. badjr dat=. jread file;CNPARMS,CNDICDOC do.
    jdmasterr ERR088 return. NB. errmsg: jfile read failure
  end.
```

```
NB. dictionary parameters and unique id
'dpt olddoc'=. dat
dn=. 1 {:: dpt

NB. didnum's must be unique
NB. errmsg: duplicate dictionary id number
if. dn e. ; 1{mdt do. jdmasterr ERR092 return. end.

NB. if not a libary adjust dictionary paths, name and documentation
if. -.islib dpt do.

  NB. cannot register read/write dictionaries that are not binary
  NB. binary compatible with current version of J 9.04+ HARDCODE:
  if. bck0 +. bck2 do. OK
  elseif. bck1 do.
    NB. errmsg: cannot register binary incompatible dictionary
    (jdmasterr ERR108),name;dicver;JVERSION return.
  end.

  NB. test dictionary file attributes - we must be able to read/write
  if. 0 e. iswriteable dcfiles do.
    NB. errmsg: dictionary file attributes do not allow read/write
    jdmasterr ERR095 return.
  end.

  dpt=. ((<path) ,&.> JDSDIRS ,&.> PATHDEL) PARMDIRS} dpt
  dpt=. (<name) 0} dpt
```

```
      doc=. (*#doc){olddoc;doc
      if. badreps (dpt;doc) jreplace file;CNPARMS,CNDICDOC do.
        jdmasterr ERR056  NB. errmsg: jfile replace failure
      end.
    end.

    newmdt=. mdt,.name;dn;path;0
    okm=. OK058
  end.

  NB. update master dictionary table+backup, didnum log, open status
  uv=. (newmdt;mdt;dlg,dn) jreplace mf;CNMFTAB,CNMFTABBCK,CNMFDLOG
  if. 0&> <./uv do. jdmasterr ERR069 return. end. NB. errmsg: error updating master

  NB. free master file for other tasks
  if. badrc msg=. markmast~0 do. msg return. end.

  ok okm;name;jpathsep path
end.
)

NB. names containing substring: (;:'cats bats') nlctn 'at'
nlctn=: ([: I. [: +./"1 ([: ,: ]) E. [: > [) {  [

NB. match prefixes (optimize for large lists): (;:'he bo boat') nlpfx 'bo'
nlpfx=: [ #~ ([: < [: , ]) -:&> ([: # [: , ]) {.&.> [
NB. match name suffixes: (;:'yada yada yo') nlsfx 'da'
nlsfx=: [ #~ ([: < [: , ]) -:&> ([: - [: # [: , ]) {.&.> [

NB. containing pattern in raised and nubbed
nubnlctn=: ([: sortdnub [) nlctn ]

NB. match prefixes in raised and nubbed
nubnlpfx=: ([: sortdnub [) nlpfx ]

NB. match suffixes in raised and nubbed
nubnlsfx=: ([: sortdnub [) nlsfx ]

opendict=: 4 : 0

NB.*opendict v-- opens dictionaries. Dictionary names and  master
NB. table  have been  validated prior  to calling this verb.  The
NB. dictionary  system  does  not  leave  files  open  as   this
NB. significantly decreases crash  resistence. Instead the master
NB. dictionay table is marked with 1 when dictionaries are opened
NB. read/write.  Only one task can open a  dictionary read/write.
NB. Many tasks can open the same dictionary read/only.
NB.
NB. dyad:  blclDictionary opendict (iaOption ; btMdt)
NB.
NB.   ('d0';'d1') opendict 1;jread JMASTER;CNMFTAB NB. open di r/w

NB. quit if open limit exceeded - limits the number of directory objects
NB. errmsg: request exceeds open limit
if. DPLIMIT<(#x)+#DPATH do. jderr ERR070 return. end.

NB. if any dictionary is already on the path quit with error
uv=. x e. {."1 DPATH
if. 1 e. uv do. (jderr ERR071),uv#x return. end. NB. errmsg: already open

NB. open status and master dictionary table
'os mdt'=. y

NB. get locations of dictionaries to open
pd=. (0{mdt) i. x
ld=. (<2;pd){mdt

NB. if any dictionary is already open read/write quit with error
NB. note: because other tasks may have a dictionary open read/write
NB. it does not appear on the path of this task - HARDCODE: rs code
rs=. 0 < ; (<3;pd){mdt
NB. errmsg: another task opened read/write
if. 1 e. rs do. (jderr ERR072),(1=rs)#x return. end.

NB. standard files with extension
fn=. JDFILES ,&.> <IJF

NB. test existence of alleged dictionary files
if. 1 e. uv=. -. fex"1 dcfiles=. <@:;"1  ld ,"0 / fn do.
  (jderr ERR073),uv#x return. NB. errmsg: missing dictionary file(s)
```

```
end.

NB. open request seems valid - mark master
if. badrc msg=. markmast 1 do. msg return. end.

dpath=. DPATH
libstatus=. i.0

for_dp. ld do.  NB. depends on (#x)=(#pd)=#ld

   NB. get dictionary parameters
   if. badjr pdp=. jread (;dp,{.fn);CNPARMS do.
     NB. errmsg: cannot read dictionary parameters
     (jdmasterr ERR074),dp_index{x return.
   end.

   NB. master table didnum must match current dictionary didnum
   if. ((<1;dp_index{pd){mdt) -: 1{>pdp do.

     NB. is the master path a prefix of stored dictionary paths?
     NB. assumes: all subdir path prefixes are the same - this
     NB. is true for dictionaries created by (newd)
     nppfx=. -.0{(;dp) E. ;(0{PARMDIRS){>pdp

     if. nppfx *. islib >pdp do.
       NB. remap paths for libraries if necessary - allows LAN file sharing
       NB. of libraries for many users/tasks with different access paths
```

```
    NB. WARNING: if these directories are on locked down LAN volumes
    NB. JOD commands like: make'' may return cannot write errors
    pdp=. >pdp
    npth=. PATHDEL ,&.>~ dp ,&.> PATHDEL&afterlaststr&.> rpdtrim&.> PARMDIRS{pdp
    pdp=. <npth PARMDIRS}pdp
  else.
    NB. master/stored dictionary paths must match for read/write
    if. nppfx do.
      if. #dpath=. ({:"1 dpath) -. {:"1 DPATH do. coerase"0 dpath end.
      NB. errmsg: master/dictionary file path mismatch - have owner set READONLY name/DIDNUM ->
      (jdmasterr ERR098),0 1{>pdp return.
    end.

    NB. for read/write dictionaries (not-libraries) insure
    NB. the dictionary file permissions/attributes allow writing
    if. 0 e. iswriteable dp_index{dcfiles do.
      if. #dpath=. ({:"1 dpath) -. {:"1 DPATH do. coerase"0 dpath end.
      NB. errmsg: dictionary file attributes do not allow read/write ->
      (jdmasterr ERR095),dp_index{x return.
    end.
  end.

  NB. create new directory object
  DL=. conew 'ajoddob'
  name=. dp_index{x
  if. createdl__DL nppfx;name;dp;os;pdp do.
    NB. append to path copy
```

```
      dpath=. dpath , (a: ,~ name , 1{>pdp),DL
      NB. are we a read only library?
      libstatus=. libstatus,LIBSTATUS__DL
    else.
      if. #dpath=. ({:"1 dpath) -. {:"1 DPATH do. coerase"0 dpath end.
      (jdmasterr ERR075),dp_index{x NB. errmsg: unable to open directory
      return.
    end.

  else.

    NB. destroy any directory objects opened before inconsistency
    if. #dpath=. ({:"1 dpath) -. {:"1 DPATH do. coerase"0 dpath end.
    (jdmasterr ERR076),dp_index{x   NB. errmsg: master-dictionary inconsistency
    return.

  end.
end.

NB. update master read/write status and release
NB. read/write dictionaries are marked with unique
NB. id and read/only dictionaries are marked with 0
DPATH=:  dpath
NB. do not mark any library (read/only) dictionaries open
pd=. (-.libstatus)#pd
mdt=. (<JODOBID * 1=os) (<3;pd)} mdt     NB. object noun !(*)=. JODOBID
if. badreps (<mdt) jreplace JMASTER;CNMFTAB do.
```

```
  jdmasterr ERR077 NB. errmsg: unable to update master
elseif. badrc msg=. markmast~0  do. msg  NB. HARDCODE: r/w codes
elseif. os e. 1 2 do.
  uv=. (1=os){rs=. '/' ,&.> READSTATS    NB. read/only and read/write strings
  (ok OK052,(}.;libstatus{(uv,0{rs)),') ->'),x
elseif.do. jderr ERR001
end.
)

pathnl=: 3 : 0

NB.*pathnl v-- returns a complete path order list of objects (y).
NB.
NB. monad:  pathnl iaObject
NB.
NB.   pathnl WORD  NB. all words on current path

pob=. {:"1 DPATH
if. badrc uv=. y loadalldirs pob do. uv return. end.
ok (>dnix__uv y) fullmonty pob [ uv=. {.pob
)

pathref=: 3 : 0

NB.*pathref  v-- returns a complete path  order list  of  objects
NB. with reference lists.  Currently only words have stored
NB. references but more may be added as the need arises.
NB.
NB. monad:  pathref iaObject
NB.
NB.   pathref WORD  NB. all words on current path with stored references

pob=. {:"1 DPATH
if. badrc uv=. y loadallrefs pob do. uv return. end.
ok (>dnrn__uv y) fullmonty pob [ uv=. {.pob
)

putdicdoc=: 3 : 0

NB.*putdicdoc v-- writes put dictionary documentation.
NB.
NB. monad:  putdicdoc clDoc

NB. assumes a put dictionary is open
if. badcl y do. jderr ERR097 NB. errmsg: invalid dictionary document must be character list
else.
  DL=. {:{.DPATH  NB. directory object !(*)=. DL

  NB. Whether the put dictionary document is stored depends on the
  NB. value of the "new" dictionary parameter DOCUMENTDICT.
  dictdoc=. 1
  if. 0=nc<'DOCUMENTDICT'   do. dictdoc=. 1=DOCUMENTDICT
  elseif.
    NB. if setting exists in put dictionary directory use it
    0=nc<'DOCUMENTDICT__DL' do. dictdoc=. 1=DOCUMENTDICT__DL
  end.

  NB. remind user DOCUMENTDICT is off
  if. -.dictdoc do. ok OK063;DNAME__DL return. end.

  if. badreps (<y) jreplace WP__DL;CNDICDOC do. jderr ERR056 NB. errmsg: replace failure
  else.
    ok OK062;DNAME__DL
  end.

end.
)

putexplain=: 4 : 0

NB.*putexplain v-- stores short object explanation text.
NB.
NB. dyad:  (iaObject ;< ba) putexplain bt|blcl

NB. validate explain texts
if. badrc y=. checknttab y do. y return. else. y=. rv y end.
if. +/.MAXEXPLAIN < #&> {:"1 y do. jderr ERR089 return. end.  NB. errmsg: text(s) to long

'obj DL'=. x  NB. directory object !(*)=. DL

if. badrc uv=. x inputdict {."1 y do. uv
else.
```

```
  ix=. (>dnix__DL obj),'__DL'      NB. directory object noun name
  fp=. ".({.>dncn__DL obj),'P__DL' NB. file pointer

  pos=. (".ix) i. {."1 y           NB. inverted list replacement positions

  NB. objects exist in put dictionary update explain text
  if. badrc uv=. (pos;<<{:"1 y) invreplace fp;CNEXPLAIN do. uv return. end.

  uv=. ' ',>dnnm__DL obj
  ok ((":#pos),uv,OK055) ; DNAME__DL
end.
)

putgs=: 4 : 0

NB.*putgs  v--  stores  dictionary groups  and  suites.  Prior to
NB. calling this verb  names, path and put dictionary status have
NB. been validated.
NB.
NB. dyad:  (bacl ; ia ; ia) putgs blcl
NB.
NB.   ((<'6');WORD;GROUP) putgs ;: 'group and members'

'DL code gtype'=. x  NB. directory object !(*)=. DL

if. badrc msg=. pathnl code do. msg return. end.
y=. /:~ ~. }. y [ gn=. {. y
if. */b=. y e. ; }. msg do.
```

```
NB. change/create group -- insure group directory is ready
if. loaddir__DL gtype do.
  jderr ERR054  NB. errmsg: unable to load directory
elseif. do.

  NB. depends on first char of index list matching (cP_DL) nouns
  fc=. {. ix=. (>dnix__DL gtype),'__DL'
  cn=. (>dncn__DL gtype),'__DL'

  NB. groups/suites are either new or replacements
  uv=. (".ix) i. gn
  dfopen__DL fc
  gp=. ".fc,'P__DL'

  if. uv=#".ix do.

    NB. group is new - create

    NB. EDGE CONDITION?? if another group with the same
    NB. name exists on the path copy the group/suite text
    NB. of that group to this new group.  Use of this system has shown
    NB. that this is desirable behaviour because of the common
    NB. practice of "regrouping" in the put dictionary new versions
    NB. of the same group that are deeper on the path.
    if. +./uv=. (<gn) e.&> }. pathnl gtype do.
      if. badrc uv2=. gtype getgstext gn do. uv2 return. else. uv=. (1;0 1){::uv2 end.
```

```
else.
  uv=. '' NB. default script is empty
end.

gdat=. <gn , uv ; < y       NB. (cn)   name, script, contents
gdat=. gdat , <gn , 3$<''   NB. (cn+1) name, latex, html, text, et cetera

NB. append group
if. badappend apcn =. gdat jappend gp do.
  jderr ERR058 [ dfclose__DL fc return.  NB. errmsg: append failure
end.

stamp=. nowfd now ''
uv=.  stamp;stamp;<a:
uv2=. CNPUTDATE,CNCREATION,CNEXPLAIN NB. NIMP group append
if. badrc msg=.uv invappend gp;uv2 do. msg
else.
  NB. update directory
  uv=.  (".ix) , gn
  uv2=. (".cn) , {. apcn
  if. badrc (gtype,gp) savedir__DL uv;uv2 do.
    jderr msg [ dfclose__DL fc return.
  else.

    NB. stamp good directory change
    (<(#".ix);now '') jreplace gp;CNMARK
  end.
```

```
    end.

  else.

    NB. group exists - update
    apcn=. uv { ".cn
    if. badjr uv2=. jread gp;apcn do.
      jderr ERR088 [ dfclose__DL fc return.  NB. errmsg: read failure
    elseif. gn -: 0 {>uv2 do.

      NB. update group list - group script is not changed
      if. badreps (<(}:>uv2),<y) jreplace gp;apcn do.
        jderr ERR056 [ dfclose__DL fc return.  NB. errmsg: replace failure
      end.

      uv2=. uv;nowfd now ''
      if. badrc msg=.uv2 invreplace gp;CNPUTDATE do. msg return. end.

    elseif.do.
      jderr ERR055 return. NB. errmsg: directory-data inconsistency
    end.
  end.

  dfclose__DL fc
  uv=. ,>dnnm__DL gtype
  ok(uv,' <',(>gn),'> ',OK059);DNAME__DL
end.
```

```
else.
  (jderr ERR083),y #~ -. b  NB. errmsg: not on path
end.
)

putntstamp=: 4 : 0

NB.*putntstamp v-- store name, creation and last put timestamps.
NB.
NB. dyad: (iaObject ;< ba) putntstamp btNts
NB.
NB.   'rc nts'=: 0 _14 get }. revo ''
NB.   DL=: {:{:DPATH__ST__JODobj
NB.   (WORD;<DL) putntstamp__ST__JODobj nts

NB. validate name/creation/lastput array
if. badrc uv=. checkntstamp y do. uv return. else. uv=. rv uv end.

NB. directory object !(*)=. DL
'obj DL'=. x

NB. timestamp names must exist on current path:  errmsg: not on path ->
tn=. ;0{uv [ pn=. ; }. pathnl obj
if. 0 e. bm=. tn e. pn do. (jderr ERR083),(-.bm)#tn return. end.

NB. get current timestamps and object index
if. badrc cts=. gettstamps__DL obj do. cts return. else. cts=. rv cts end.
```

```
oix=. ".(>dnix__DL obj),'__DL'

pos=. oix i. tn           NB. timestamp name positions in index
pix=. pos -. #oix         NB. put dictionary name positions
npn=. (I. pos = #oix){tn  NB. names that are not in put dictionary
ppn=. pix{oix             NB. names that are in put dictionary

NB. update put dictionary timestamps - insure shape is unchanged
scts=. $cts
cts=. ((tn i. ppn) {"1 ;1{uv) pix}"1 cts
if. -.scts -: $cts do. jderr ERR102 return. end.

NB. attempt to save changes
if. badrc uv=. obj puttstamps__DL cts do. uv
else.
  ok ('(',(":#ppn),OK064,(":#npn),OK065);(<ppn),<npn
end.
)

puttable=: 4 : 0

NB.*puttable v-- stores (name,text) and (name,type,value) tables.
NB. Used  to  store tests,  macros, and word tables. Result  is a
NB. return code and message. Note: the directory object reference
NB. (DL) has been set before calling this verb.
NB.
NB. dyad:  (iaObj ; bacl) puttexts btNameScript|btNameTypeValue
NB.
NB.   (TEST;<DL) puttable ('name1';'name2') ,. 'script...';'script...'

'code DL' =. x  NB. directory object !(*)=. DL

if. loaddir__DL code do.
  jderr ERR054  NB. errmsg: unable to load directory
else.
  y=. >{:y

  NB. depends on first char of index list matching (cP_DL) nouns
  fc=. {. ixn =. (>dnix__DL code),'__DL'
  cnn=. (>dncn__DL code),'__DL'

  NB. either new or replacements
  uv=. (".ixn) i. {."1 y
  b=.  uv = #".ixn
  pc=. 0

  NB. replace (will not change key directory lists)
  dfopen__DL fc
  fp=. ".fc,'P__DL'

  if. 0 e. b do.
    if. badrc msg=. (code;ixn;cnn;fp;<DL) rplctable (<(-.b)#y),<(-.b)#uv do.
      msg [ dfclose__DL fc return.
    end.
    pc=. pc + rv msg
  end.

  NB. append (always appends to key directory lists)
  if. 1 e. b do.
    if. badrc msg=. (code;ixn;cnn;fp;<DL) apptable b#y do.
      msg [ dfclose__DL fc return.
    end.
    pc=. pc + rv msg

    NB. stamp good directory change
    (<(#".ixn);now '') jreplace fp;CNMARK
  end.
  dfclose__DL fc

  uv=. ' ',,>dnnm__DL code
  ok ((":pc),uv,OK057) ; DNAME__DL
end.
)

puttexts=: 4 : 0

NB.*puttexts v-- stores object documentation and group/suite
NB. texts.
NB.
NB. dyad:  (iaObject ; iaOffset ;< ba) puttexts bt|blcl

NB. validate texts
if. badrc y=. checknttab y do. y return. else. y=. rv y end.
```

```
'obj offset DL'=. x  NB. directory object !(*)=. DL

if. -.offset e. 0 1 do. jderr ERR090  NB. errmsg: file offset invalid
elseif. badrc uv=. (obj;<DL) inputdict {."1 y do. uv
elseif.do.

  ix=. (>dnix__DL obj),'__DL'   NB. directory object index noun
  cn=. (>dncn__DL obj),'__DL'   NB. directory object component name
  fp=. ".({.cn),'P__DL'         NB. file pointer

  NB. text components
  rcn=. (".ix) i. uv=. {."1 y
  rcn=. offset + rcn{".cn

  NB. read components and test contents
  dat=. jread fp;rcn
  if. uv badcn dat do.
    jderr ERR055 return.  NB. errmsg: directory-data inconsistency
  end.

  dat=. >dat  NB. HARDCODE: group/suite index 1, document index 3
  dat=. ({:"1 y) (<a:;offset{1 3)} dat

  if. badreps (<"1 dat) jreplace fp;rcn do.
    jderr ERR056 return.  NB. errmsg: replace failure
  end.
  uv=. ' ',(>dnnm__DL obj),' '
  ok ((":#rcn),uv,(>offset{'text';'document'),OK057) ; DNAME__DL
end.
)

putwords=: 4 : 0

NB.*putwords v-- stores  words  in the  words  file.  Result is a
NB. return code and message.
NB.
NB. dyad:  (cl ; baObj) putwords blclWords
NB.
NB.   ('locale';<<'2') putwords 'words';'are';'us'

if.     badrc uv=. checknames y  do. uv
elseif. y=. ~.}.uv      NB. unique deblanked names
        'loc DL'=. x    NB. source locale and directory object !(*)=. DL
        b=. wex uv=. y ,&.> locsfx loc  NB. do words exist
        0 e. b  do. (jderr ERR053) , (-.b)#uv  NB. errmsg: word(s) do not exist
NB. insure word directory is ready
elseif. loaddir__DL  WORD do.
  jderr ERR054 NB. errmsg: unable to load directory
elseif. do.

  NB. words are either new or replacements
  uv=. WORDIX__DL i. y
  b=.  uv = #WORDIX__DL
  pc=. 0

  dfopen__DL 'W'
  NB. replace words (will not change key directory lists)
  if. 0 e. b do.
    dropnc__DL WORD  NB. replacements can change word class
    if. badrc msg=. x rplcwords (<(-.b)#y),<(-.b)#uv do.
      msg [ dfclose__DL 'W' return.
    end.
    pc =. pc + rv msg
  end.

  NB. append new words (always appends to key directory lists)
  if. 1 e. b do.
    dropnc__DL WORD  NB. new words - force reload of name class if necessary
    if. badrc msg=. x appwords b#y do. msg [ dfclose__DL 'W' return. end.
    pc=. pc + rv msg

    NB. stamp good directory change
    (<(#WORDIX__DL);now '') jreplace WP__DL;CNMARK
  end.
  dfclose__DL 'W'

  ok ((":pc),OK051) ; DNAME__DL
end.
)

putwordxrs=: 4 : 0
```

```
NB.*putwordxrs v-- stores global word references
NB.
NB. dyad:  (cl ;< ba) putwordxrs blcl

'name DL'=. x NB. directory object !(*)=. DL

NB. check path prior to storing or changing references
if. badrc msg=. checkpath DL do. msg
elseif. loadref__DL WORD do. jderr ERR079  NB. errmsg: unable to load references
elseif.do.

  NB. word references are either new or replacements
  pos=. WORDREFIX__DL i. <name
  b=. pos = #WORDREFIX__DL

  NB. categorize references and test for path existence
  if. #y=.catrefs y do.
    if. badrc uv3=. pathnl WORD do. uv3 return. end.
    if. 0 e. uv2=. (uv=. 0{::y) e. ;}.uv3 do.
      (jderr ERR083),uv #~ -.uv2 return.  NB. errmsg: not on path
    end.
  end.

  dfopen__DL 'U'
  fp=. UP__DL
```

```
uv3=. 0
if. b do.
  NB. append new references
  NB. append only non-null lists
  if. #y do.
    NB. append reference list
    y=. <name;WORD;<y
    if. badappend cn=. y jappend fp do.
      jderr ERR058 [ dfclose__DL 'U' return.  NB. errmsg: append failure
    end.
    uv=.  WORDREFIX__DL , <name
    uv2=. WORDREFCN__DL , cn
    uv3=. 1
  end.
else.
  NB. replace references (removing nulls if necessary)
  NB. NIMP directory-data consistency check
  if. #y do.
    NB. non-null replacement list
    y=. <name;WORD;<y
    if. badreps y jreplace fp;pos{WORDREFCN__DL do.
      jderr ERR056 [ dfclose__DL 'U' return.  NB. errmsg: replace failure
    end.
  else.
    NB. replacement has no references remove from directory
    uv3=. 0 pos} (#WORDREFIX__DL)#1
    uv=.  uv3#WORDREFIX__DL
```

```
      uv2=. uv3#WORDREFCN__DL
      uv3=. 1
    end.
  end.

  NB. update reference directory and close
  if. uv3 do.
    if. badrc msg=. (WORD,fp) saveref__DL uv;uv2 do. msg [ dfclose__DL 'U' return. end.
  end.
  dfclose__DL 'U'

  ok ('<',name,'>',OK056) ; DNAME__DL
end.
)

rplctable=: 4 : 0

NB.*rplctable v-- replaces (name,text) and (name,type,value) tables to file.
NB.
NB. dyad:  bl rplctable (btTable ,< ilPositions)

'ttype ixn cnn fp DL'=. x  NB. directory object !(*)=. DL
y=. 0 [ 'tab pos'=. y

sizes=. #&> {:"1 tab
tc=. #tab [ cn=. pos{".cnn
pf=. PUTFACTOR__DL
```

```
NB. for words and macros record class/type
if. wmt=. ttype e. WORD,MACRO do.
  class=. ; 1 {"1 tab
end.

while. #tab do.
  cnt=. pf <. #tab
  tn=.  cnt{.tab [ rcn=. cnt{.cn

  NB. read components and test contents
  dat=. jread fp;rcn
  if. ({."1 tn) badcn dat do.
    jderr ERR055 return.  NB. errmsg: directory-data inconsistency
  end.

  NB. replace
  if. badreps (<"1 tn) jreplace fp;rcn do.
    jderr ERR056 return.  NB. errmsg: replace failure
  end.

  tab=. cnt}.tab [ cn=. cnt}.cn
end.

NB. set up replacements
if. wmt do.
  dropnc__DL ttype NB. replacements can change class/type
  invcmps=. CNCLASS,CNPUTDATE,CNSIZE
  reps=. pos;<class;(tc#nowfd now '');sizes
else.
  invcmps=. CNPUTDATE,CNSIZE
  reps=. pos;<(tc#nowfd now '');sizes
end.

if. badrc msg=. reps invreplace fp;invcmps do. msg else. ok tc end.
)

rplcwords=: 4 : 0

NB.*rplcwords v-- replaces extant words.

'loc DL'=. x  NB. directory object !(*)=. DL
'names pos'=. y
cn=. pos{WORDCN__DL
wc=. #pos

wp=. WP__DL [ pf=. PUTFACTOR__DL
lnames=. names ,&.> locsfx loc
size=. class=. i.0

while. #names do.
  cnt=. pf <. #names
  wn=. cnt{.names [ lwn=. cnt{.lnames [ rcn=. cnt{.cn

  NB. read components and test contents - this slows things
  NB. down but significantly improves database hygiene
  dat=. jread WP__DL;rcn
  if. wn badcn dat do.
    jderr ERR055  NB. errmsg: directory-data inconsistency
  end.

  val=. wrep&.> lwn   NB. word values
  bsz=. #&> val       NB. NIMP word byte sizes (size test)
  bnc=. nc lwn        NB. word name class
  val=. , <"1 wn ,. (<"0 bnc) ,. val

  NB. replace words
  if. badreps val jreplace wp;rcn do.
    jderr ERR056  NB. errmsg: replace failure
  else.
    size=. size , bsz
    class=. class , bnc
  end.

  names=. cnt}.names [ lnames=. cnt}.lnames [ cn=. cnt}.cn
end.

msg=. ERR057  NB. errmsg: directory update failure
if. wc ~: #size do. jderr msg return. end.

reps=. pos;<class;(wc#nowfd now '');size
if. badrc msg=.reps invreplace wp;CNCLASS,CNPUTDATE,CNSIZE do. msg else. ok wc end.
)
```

```
NB. raise, nub and sort blblcl name lists
sortdnub=: [: /:~ [: ~. ;
```

jodmake Source Code

```
NB. *jodmake c-- script making & code manipulation: extension of (jod).
NB.
NB. This subclass defines utilities for making scripts from
NB. groups and suites.  It also contains code for analyzing
NB. name references in J words.
NB.
NB. Interface nouns & verbs:
NB.   getallts      gets all timestamps
NB.   makedump      dumps objects on path to put dump directory
NB.   makegs        make group and suite scripts
NB.   namecats      classifies names in J code
NB.   putallts      puts all timestamps - see (getallts)
NB.   wttext        word and test text
NB.   wrdglobals    extracts global names from J code
NB.
NB. Notes:
NB.   Error messages (jodmake range 150-199)

coclass 'ajodmake'
coinsert  'ajod'

NB.*dependents x-- objects with definition dependencies

NB. tags JOD dump script rebuild commands
DUMPTAG=: ' NB.{*JOD*}'
```

```
NB. large text wrap temporary noun name and line width
WRAPTMPWID=: 'zz';67

NB. expression that clears scratch object
SOCLEAR=: '".soclear',DUMPTAG

NB. expression that rebuilds groups and suites from scatch object data
SOGRP=: ' grp&> ". ". ''',(>{.WRAPTMPWID),'_'',SOLOCALE,''_'' [ cocurrent ''base''',DUMPTAG

NB. expression that stores words in the scratch object in JOD
SOPUT=: 'soput ".''nl_'',SOLOCALE,''_ i.4'' [ cocurrent ''base''',DUMPTAG

NB. expression that stores (name,text) tables in scratch object
SOPUTTEXT=: ' put ". ".''',(>{.WRAPTMPWID),'_'',SOLOCALE,''_'' [ cocurrent ''base''',DUMPTAG

NB. expression that switches to numbered scratch locale
SOSWITCH=: 'cocurrent SO__JODobj',DUMPTAG

NB.*enddependents


NB.*end-header

NB. direct definition escape tokens - order matters
DDEFESCS=: ;:'{{}})'

DUMPMSG0=: 'NB. JOD dictionary dump: '
DUMPMSG1=: 'Names & DidNums on current path'

NB. should appear as quoted text when displayed
DUMPMSG2=: '''NB. end-of-JOD-dump-file regenerate cross references with:  0 globs&> }. revo '''''''' '''

NB. version prefix text for JOD dumps
DUMPMSG3=: 'NB. Generated with JOD version'

NB. J version that created this dumpfile
DUMPMSG4=: 'NB. J version: '

ERR0150=: 'confused declarations ->'

ERR0151=: 'word syntax'

ERR0152=: 'no definition ->'

ERR0153=: 'file write failure'

ERR0154=: 'invalid group/suite name'

ERR0155=: 'unable to append to dumpfile ->'

ERR0156=: 'unable to create dumpfile ->'
```

```
ERR0157=: 'directory-component name class inconsistency -- dump aborted ->'

ERR0158=: 'invalid fully qualified dump file name'

ERR0159=: 'mixed assignments ->'

ERR0160=: 'invalid object timestamp table'

NB. multiplicative factor for small text dumps
EXPLAINFAC=: 10

NB. first table of valid single line explicit headers
EXPPFX0=: 4 5$'1 : ''2 : ''3 : ''4 : '''

NB. second table of valid single line explicit headers
EXPPFX1=: 3 8$'3 : ('':''3 : (,'':4 : (,'':'

NB. report labels for reference case of (globs)
GLOBCATS=: <;._1 ' Global Local (*)=: (*)=. for.'

NB. string marking end of class header
HEADEND=: 'NB.*end-header'

NB. explict J argument names
JARGS=: <;._1 ' x y u v m n $:'
```

```
NB. mixed assignment override tag
MIXEDOVER=: '(<:)=:'

OK0150=: 'file saved ->'

OK0151=: 'object(s) on path dumped ->'

NB. portable box drawing characters
PORTCHARS=: ,:'+++++++++|-'

NB. name of monadic identity verb that displays and passes argument
SOPASS=: 'showpass '

btclfrcl=: 3 : 0

NB.*btclfrcl v-- inverse of clfrbtcl.
NB.
NB. monad:  btcl =. btclfrcl cl

NB. length of prefix
len=. ".(pos=. y i. ' '){. y
y=. (>:pos) }. y

NB. prefix and shape of bt
shp=. 2 {. pfx=. ". len {. y
pfx=. 2 }. pfx
tab=. len }. y

if. #tab do.
  zm=. 0<pfx
  bm=. 0 #~ #tab
  ri=. }:0,+/\zm#pfx
  bm=. 1 ri } bm
  shp $ zm #^:_1 bm <;.1 tab
else.
  shp$<''  NB. all nulls
end.
)

clearso=: 3 : 0

NB.*clearso v-- empty scratch object.
NB.
NB. monad:  clearso uuIgnore

if. #s=. nl__SO i.4 do. (4!:55) s ,&.> locsfx SO end.  NB. !(*)=. SO
)

clfrbtcl=: 3 : 0

NB.*clfrbtcl v-- btcl to encoded cl.
NB.
NB. This verb  converts a boxed table of character lists to a  cl
NB. representation that can be used  to recreate the boxed table.
NB. It  is used instead of (5!:5) for btcl as (5!:5)  generates a
NB. large  a.   index  representation  for  character  data   when
NB. selected "control" characters are present.
NB.
NB. monad:  cl =. clfrbtcl btcl

NB. shape and lengths of all char lists
sp=. $ y
lc=. , #&> y

NB. first number is length of prefix
pfx=. ":sp,lc
(":$pfx),' ',pfx, ;y
)


createmk=: 3 : 0

NB.*createmk v-- initializes maker objects
NB.
NB. monad:  createmk bl0brefs
NB.
NB.   createmk__MK JOD;ST;MK;UT;<SO

NB. object references !(*)=. JOD ST MK UT SO
'JOD ST MK UT SO'=:  y
)
```

```
ddefescmask=: 3 : 0

NB.*ddefescmask v-- direct definition escape token mask.
NB.
NB. Returns  a  bit mask  of direct  definition )? tokens.  These
NB. tokens  would  be  seen  as globals  if passed  to  JOD  name
NB. analysis verbs.
NB.
NB. monad:  pl =. ddefescmask blclJTokens
NB.
NB.   toks=. 3 pick parsecode__MK__JODobj jcr__JODobj 'ddef00_base_'
NB.   toks #~ -.ddefescmask toks  NB. escape tokens

p=. >:I. (0{DDEFESCS)=y      NB. first token after ddef starts
b=. (2{DDEFESCS) e.~ p{y     NB. ddef )? escapes
0 ((b # p),b # >:p)} (#y)#1  NB. escape token mask
)

dec85=: 3 : 0

NB.*dec85  v-- decodes ASCII85  (name,text)  and (name,code,text)
NB. tables.
NB.
NB. monad:  cl55=. dec85 cl

tab=. fromascii85 y
tab=. btclfrcl tab
```

```
NB. there are two types of tables in JOD (name,text) and (name,code,text)
assert. ({:$tab) e. 2 3

if. 3={:$tab do.
  NB. codes must be integers in JOD name,code,text tables
  codes=. <a:;1
  val=. ".&.> codes { tab
  tab=. val codes } tab
end.

NB. put commands are expecting (5!:5) strings
5!:5 <'tab'
)

dumpdictdoc=: 3 : 0

NB.*dumpdictdoc  v-- appends  dictionary  documentation  text  to
NB. dumpfile.
NB.
NB. monad:  dumpdictdoc clPathFile
NB.
NB.   dumpdictdoc 'c:\go\ahead\dump\my\dictionary.ijs'

NB. cannot fetch document or document is empty
if. badrc uv=. DICTIONARY get '' do. (jderr ERR0155),<y return. end.
if. 0=#uv=. >1{uv do. OK return. end.

tag=. DUMPTAG,LF
```

```
putso=. (2#LF),SOSWITCH,LF

NB. expression to store dictionary documentation text in scratch locale
soputdoc=. SOPASS,(":DICTIONARY),' put >1{,".".''zz_'',SOLOCALE,''_'' [ cocurrent ''base'' ',tag

NB. format document text for dump
dicdoc=. putso,WRAPTMPWID fmtdumptext ,:'';uv
dicdoc=. dicdoc,LF,soputdoc,SOCLEAR,2#LF

NB. append dictionary documentation error msg: unable to append to dumpfile
if. _1 -: (toHOST dicdoc) fap <y do. (jderr ERR0155),<y else. OK end.
)

dumpdoc=: 4 : 0

NB.*dumpdoc v-- dumps object documentation text.
NB.
NB. dyad:  (iaBlksize ; iaObject ; clPathfile) dumpdoc blclNames
NB.
NB.   (0;50;'c:\dump\on\me.ijs') dumpdoc ;:'word name list'

NB. block size, object, output file
'blk obj out'=. x

NB. append short and long object documentation - short documents are small
NB. hence we process in blocks (EXPLAINFAC) times larger than the dump block
if. badrc uv=. ((EXPLAINFAC*blk);(obj,EXPLAIN);out) dumptext y do. uv
elseif. badrc uv=. (blk;(obj,DOCUMENT);out) dumptext y do. uv
elseif.do. OK
end.
)

dumpgs=: 4 : 0

NB.*dumpgs v-- dump groups and suites on path.
NB.
NB. dyad:  iaBlksizeObject dumpgs clPathFile
NB.
NB.   (50,GROUP) dumpgs 'c:\dump\your\groups.ijs'

'dmp obj'=. x
putso=. LF,SOSWITCH,LF
cmd=. SOPASS,(":obj),SOGRP,LF,SOCLEAR,LF
out=. <y

if. badrc uv=. obj dnl '' do. uv
elseif. a: e. uv          do. OK  NB. no groups or suites
elseif.do.

  uv=. (-dmp) <\ uv=.}.uv
  for_blk. uv do.

    NB. get blblcl of all objects in groups/suites
    gnames=. obj grp&.> >blk

    NB. check all return codes error msg: unable to dump group/suite list(s)
    if. 0 e. {.&> gnames do. jderr ERR0157 return. end.

    NB. remove return codes, attach group/suite names and format as text
    gnames=. (<"0 >blk) ,&.> }.&.> gnames
    gnames=. 5!:5 <'gnames'

    NB. append if any text
    if. #gnames=. WRAPTMPWID wraplinear gnames do.
      gnames=. toHOST putso,gnames,LF,cmd
      if. _1 -: gnames fap out do. (jderr ERR0155),out return. end.
    end.

  end.

  NB. dump group/suite header scripts
  if. badrc msg=. (dmp;obj;y) dumptext ;uv do. msg return. end.

  NB. dump group/suite documentation
  if. badrc msg=. (dmp;obj;y) dumpdoc ;uv do. msg return. end.
end.
OK
)

dumpheader=: 3 : 0

NB.*dumpheader v-- creates the dumpfile and writes  header
NB. information.
NB.
NB. monad:  dumpheader clPathFile
NB.
NB.   dumpheader 'c:\go\ahead\dump\my\dictionary.ijs'

NB. error msg: unable to create dumpfile
if. _1 -: '' (write :: _1:) y do. (jderr ERR0156),<y return. end.

NB. make box characters portable
9!:7 , PORTCHARS [ curchars=. , 9!:6 ''

NB. format header text
head=. DUMPMSG0 , tstamp ''
head=. head,LF,DUMPMSG3 , ;(<'; ') ,&.> ":&.>JODVMD
head=. head,LF,DUMPMSG4 , ": , 9!:14 ''
head=. head,LF,ctl 'NB. ',"1 ' ' , DUMPMSG1 , ": 0 1 {"1 DPATH__ST
head=. head,LF,LF

NB. reset box characters
9!:7 curchars

NB. set up J environment to process script - assumes that
NB. JOD is loaded and that a target put dictionary is open
tag=. DUMPTAG,LF

NB. retain white space
head=. head,'9!:41 [ 1',tag
```

```
head=. head,'cocurrent ''base''',tag
head=. head,'sonl_z_=: ''sonl__MK__JODobj i.4''',tag
head=. head, (SOPASS-.' '),'_z_=:] [ 1!:2&2',tag
head=. head,'SOLOCALE_z_=:":>SO__JODobj',tag
head=. head,'soput_z_=:SOLOCALE&put',tag
head=. head,'soclear_z_=: ''0 0 $ clearso__MK__JODobj 0''',tag

NB. append header error msg: unable to append to dumpfile
if. _1 -: (toHOST head) fap <y do. (jderr ERR0155),<y else. OK end.
)

dumpntstamps=: 4 : 0

NB.*dumpntstamps v-- appends object timestamps text to dumpfile.
NB.
NB. dyad:  paRag dumpntstamps clPathFile
NB.
NB.   1 dumpntstamps'c:\go\ahead\dump\my\dictionary.ijs'

if. x do.

  NB. fetch all object timestamps
  if. badrc ots=. getallts 0 do. ots return. else. ots=. rv ots end.

  NB. if no objects exist dump nothing
  if. 0 = >./ , #&> (0 1){ots do. OK return. end.

  tag=. DUMPTAG,LF
  putso=. LF,SOSWITCH,LF

  NB. make sure older versions of JOD can execute dumps with timestamps without errors.
  putup=. 'cocurrent ''base'' ',tag
  putup=. putup, 'puttstamps_ijod_=: (((1;''upgrade JOD'')"_)`putallts__MK__JODobj)@.(3 = (4!:0)<''putallt
>..>s__MK__JODobj'')',tag

  NB. expression to store timestamps from text in scratch object
  soputts=. putup,SOPASS,'puttstamps ".".''zz_'',SOLOCALE,''_'' [ cocurrent ''base'' ',tag

  NB. text in scratch object
  tstext=. putso,(WRAPTMPWID,(getascii85 0);<1) wraplinear 5!:5 <'ots'
  tstext=. tstext,LF,soputts,SOCLEAR,2#LF

  NB. write to test file
  NB. (toHOST tstext) write jpath '~temp/dumpnts.ijs'

  NB. append timestamps msg: unable to append to dumpfile
  if. _1 -: (toHOST tstext) fap <y do. (jderr ERR0155),<y else. OK end.
else.
  OK
end.
)


dumptext=: 4 : 0

NB.*dumptext v-- appends text tables to dump file.
NB.
NB. dyad:  (iaBlksize ; ilObjCode ; clPathFile) dumptext blclNames
NB.
NB.  (50;1 8;'c:\temp\dump.ijs') dumptext ;:'test case names'

NB. block size, object & option code, output file
'bsize noc out'=. x
out=.<out
bnames=.(-bsize) <\ y
putso=. LF,SOSWITCH,LF

NB. reload command for object
cmd=. SOPASS,(":noc),SOPUTTEXT,LF,SOCLEAR,LF

NB. dump text in blocks
for_blk. bnames do.
  if. badrc uv=. noc get >blk do. uv return. else. uv=. rv uv end.

  NB. append only when we have text
  if. #uv=. WRAPTMPWID fmtdumptext uv do.
    uv=. toHOST putso,uv,LF,cmd
    NB. error msg: unable to append to dumpfile
    if. _1 -: uv fap out do. (jderr ERR0155),out return. end.
  end.

end.
OK
```

```
)

dumptm=: 4 : 0

NB.*dumptm v-- dumps test cases and macros on path.
NB.
NB. dyad:  ilBlksizeObject dumptm clPathFile
NB.
NB.   50 1 dumptm 'c:\dump\on\me.ijs'

'blk obj'=. x

if. badrc uv0=. obj dnl '' do. uv0  return. end.
if. a: e. uv0 do. OK return. end. NB. no test cases or macros

if. #uv0=. }.uv0 do.
  if. badrc uv1=. (blk;obj;y) dumptext uv0 do. uv1 return. end.
  if. badrc uv1=. (blk;obj;y) dumpdoc uv0  do. uv1 return. end.
end.
OK
)

dumptrailer=: 3 : 0

NB.*dumptrailer v-- appends terminal text to dumpfile.
NB.
NB. monad:  dumptrailer clPathFile
NB.
NB.   dumptrailer 'c:\go\ahead\dump\my\dictionary.ijs'

tag=. DUMPTAG,LF
tail=. LF,'cocurrent ''base''',tag
tail=. tail,'0 0$(4!:55);:''sonl_z_ SOLOCALE_z_ soput_z_ soclear_z_''',tag
tail=. tail,SOPASS,DUMPMSG2,tag

NB. append trailer error msg: unable to append to dumpfile
if. _1 -: (toHOST tail) fap <y do. (jderr ERR0155),<y else. OK end.
)

dumpwords=: 4 : 0

NB.*dumpwords v-- dumps path words to an ASCII script file. Nouns
NB. are dumped first in alphabetic order and then remaining words
NB. are dumped in alphabetic order.
NB.
NB. dyad:  iaBlockSize dumpwords clPathFile
NB.
NB.   50 dumpwords 'c:\j405\addons\jod\joddev\dump\joddev.ijs'

NB. dump all nouns
if. badrc nouns=. did 0 do. nouns return.
else.
  if. 2=#nouns do. NB. HARDCODE 2
    NB. only one dictionary on the path - common case
    if. badrc nouns=. (WORD,1,WORD) dnl '' do. nouns return. else. nouns=. }. nouns end.
  else.
    NB. more than one dictionary on path - requires deeper look to determine
    NB. whether a path order fetched word is a noun or something else
    if. badrc nouns=. 0 _1 0 dnl '' do. nouns return. end.
    if. badrc other=. 0 _1 dnl ''   do. other return. end.

    other=. }.other
    nouns=. }.nouns
    other=. other -.&.> nouns

    NB. sorted list of nouns that will be retrieved in path order
    nouns=. /:~ ~. ; nouns -.&.> ~.@:;&.> <"1 ,\ other

  end.
end.

putclr=. LF,LF,SOPASS,SOPUT,LF,SOCLEAR
putso=. LF,SOSWITCH,LF
noc=. WORD,0
out=. <y
space=. 2           NB. generates one blank line between objects
wnc=. WORD,INCLASS  NB. word name class code

if. (0<#nouns) *. -. a: e. nouns do.
  if. badrc wnc=. (WORD,INCLASS) invfetch__ST nouns do. wnc return.
  else. wnc=.(-x) <\ rv wnc
  end.
  names=. (-x) <\ nouns
  for_blk. names do.

    NB. get block of nouns
    if. badrc uv=. noc getobjects__ST >blk do. uv return. else. uv=. rv uv end.

    NB. check component-directory name class for consistency - classes must
    NB. be consistent to insure that the dump script can properly reload
    if. 1 e. mask=.-.(>blk_index{wnc) = ; 1 {"1 uv do.
      NB. error msg: directory-component name class inconsistency -- dump aborted
      (jderr ERR0157),mask#0{"1 uv return.
    end.

    NB. convert to linear representations
    NB. NIMP not wrapping large binaries
    if. badrc uv=. 0 nounlrep uv do. uv return. else. uv=.rv uv end.
    uv=. space jscript jscriptdefs uv

    NB. insert JOD commands to reload
    uv=. toHOST putso,uv,putclr

    NB. append to file
    if. _1 -: uv fap out do. (jderr ERR0155),out return. end.
  end.
end.

NB. append all remaining words that are stored as text
```

```
if. badrc names=. dnl '' do. names return. else. vnc=. (names=. }.names -. a:) -. nouns end.
nouns=.0

if. #vnc do.
  if. badrc wnc=. (WORD,INCLASS) invfetch__ST vnc do. wnc return.
  else. wnc=.(-x) <\ rv wnc
  end.
  vnc=. (-x) <\ vnc
  for_blk. vnc do.
    if. badrc uv=. noc getobjects__ST >blk do. uv return. else. uv=. rv uv end.
    if. 1 e. mask=.-.(>blk_index{wnc) = ; 1 {"1 uv do.
      (jderr ERR0157),mask#0{"1 uv return.
    end.
    uv=. space jscript jscriptdefs uv
    uv=. toHOST putso,uv,putclr
    if. _1 -: uv fap out do. (jderr ERR0155),out return. end.
  end.
end.

NB. dump word documentation
if. -. a: e. names do. (x;WORD;out) dumpdoc names else. OK end.
)

extscopes=: 3 : 0

NB.*extscopes v--  handles  exceptions  to  normal  J assignment
NB. scoping rules. The exceptions are:
NB.
NB. monad:  extscopes blclParsed
NB.
NB.   'quoted locals'=.
NB.   '`acr locals'=.
NB.   'quoted globals'=:
NB.   '`acr globlas'=:
NB.
NB.   for_loopvar. x do.
NB.       $ loopvar        NB. implict for. local references
NB.       loopvar_index
NB.   end.

NB. get any quoted assignments from syntactically correct code
qlocs=. (}.@:}:) &.> u #~ '''' = {.&> u=. y #~ 1|.y = <'=.'
qgbls=. (}.@:}:) &.> u #~ '''' = {.&> u=. y #~ 1|.y = <'=:'
if. #qlocs do. qlocs=. jnfrblcl <;._1 ; ' ' ,&.> qlocs -.&.> '`' end.
if. #qgbls do. qgbls=. jnfrblcl <;._1 ; ' ' ,&.> qgbls -.&.> '`' end.

NB. get any implicit for. locals
flocs=. ''
if. +./ u=. ((4&{.&.> y) e. <'for_') *. '.' = {:&> y do.
  u=. (4&}.@:}:)&.> u # y
  u=. u , u ,&.> <'_index'  NB. possible implicits
  flocs=. , y #~ y e. u
end.

(<qgbls),(<qlocs,flocs),<flocs
```

```
)

NB. direct file append with error trap
fap=: 1!:3 ::(_1:)

fmtdumptext=: 4 : 0

NB.*fmtdumptext v-- formats (name,text) tables for dumping.
NB. Result is a J script character list or null.
NB.
NB. dyad:  (clName ; iaWidth) fmtdumptext btNameText
NB.
NB.   ('z';67) fmtdumptext 1 pick 0 8 get }. dnl ''

NB. remove null entries
if. #text=. y #~ 0 < #&> {:"1 y do.

  ascii85=. getascii85 0

  NB. The (5!:5) representation will produce
  NB. a large a. index representation when any
  NB. unprintable characters are present.  To get
  NB. a compact representation for ASCII85 5!:5 must
  NB. be replaced in this context
  if. ascii85 do. text=. clfrbtcl ":&.> text else. text=. 5!:5 <'text' end.

  (x,<ascii85) wraplinear text
```

```
else.
  ''
end.
)

fromascii85=: 3 : 0

NB.*fromascii85 v-- decode ASCII85 representation.
NB.
NB. Inverse of (toascii85).
NB.
NB. monad:  cl =. fromascii85 clA85

r=. ,y
r=. a.i.r
r=. (r > 32) # r
r=. (2 * (a.i.'<~') -: 2 {. r) }. r
r=. (-2 * (a.i.'~>') -: _2 {. r) }. r
m=. r = a.i.'z'
r=. r - 33
r=. 0 (I.m) } r
r=. (1+4*m) # r
b=. 5 | #r
r=. r,84 #~ b{ 0 4 3 2 1
r=. a.{~ ,(4#256) #: 85 #. _5 [\ r
r }.~ - b { 0 0 3 2 1
)
```

```
getallts=: 3 : 0

NB.*getallts v-- gets all timestamps.
NB.
NB. Returns a boxed table of  all object timestamps. The creation
NB. and lastput dates are fractional  day yyyymmdd.fd floats. The
NB. (5!:5)  representation  of  floats  includes  all significant
NB. decimals which can bloat up linear representations. This verb
NB. applies a simple run  length encoding compression scheme that
NB. can significantly reduce the  number of (5!:5) bytes when the
NB. same timestamp value occurs frequently.
NB.
NB. monad:  btCts =. getallts uuIgnore
NB.
NB.   getallts__MK__JODobj 0

NB. last row of (cts) indicates compression scheme (0=none, 1=rle)
cts=. ((#OBJECTNC)#<0) (2)} (3,#OBJECTNC)$a:
inc=. -INPUT

for_obj. OBJECTNC do.

  NB. fetch timestamps - ignore empty object lists
  if. badrc nts=. (obj,inc) get }. obj dnl '' do. continue. end.
  nts=. rv nts

  NB. object names and uncompressed timestamps
  cts=. (<;0{nts) (<0;obj_index)} cts
  cts=. (<;1{nts) (<1;obj_index)} cts

  ets=. rlefrnl , sts=. ;1{nts
  NB. insure rle timestamps decode properly
  if. (,sts) -: nlfrrle ets do.
    NB. if run encoded timestamps are smaller use them
    if. (*/$ets) <: */$sts do.
      cts=. (<ets) (<1;obj_index)} cts
      cts=. (<1) (<2;obj_index)} cts
    end.
  end.

end.

ok <cts
)

getascii85=: 3 : 0

NB.*getascii85 v-- returns ASCII85 setting (1=On, 0=Off).
NB.
NB. monad:  getascii85 uuIgnore

ascii85=. 0 NB. do not use ascii85 default

NB. if setting exists in class use it
if. 0=nc<'ASCII85'   do. ascii85=. 1-:ASCII85
```

```
elseif.
  NB. if ASCII85 setting exists in put dictionary directory use it
  do=. {: {.DPATH__ST
  0=nc<'ASCII85__do' do. ascii85=. 1-:ASCII85__do
end.

ascii85
)

NB. 0's every other 1 in even groups of 1's
halfbits=: ] (*.) 1 0"_ $~ #

NB. clips head and tail delimited lists - see long documentation
htclip=: [ (] }.~ [: >: ] i. [) ] }.~ [: - [: >: [ i.~ [: |. ]

jnb=: 3 : 0
NB.*jnb v-- blanks out J code leaving only comments
y jnb~ masknb y
:
(x * >: i. $ x){' ',,y
)

NB. definition table to script text
jscript=: [: ; (([: <"0 [) #&.> (10{a.)"_) ,&.> ]

NB. name, class, definition table to assigned name table
jscriptdefs=: (([: {."1 ]) ,&.> (<'=:')"_) ,&.> [: {:"1 ]
```

```
makedump=: 3 : 0

NB.*makedump  v-- dumps the current path as a J  script file. The
NB. dump script can be run back  into  JOD  to rebuild  a  single
NB. dictionary that contains all objects on the current path. The
NB. dump script is a simple ASCII file  that is intended for long
NB. term storage of J  words in a form that is immune to  changes
NB. in binary storage formats.
NB.
NB. monad:  makedump uuIgnore

NB. do we have a dictionary open?
if. badrc uv=. checkopen__ST 0 do. uv return. end.

NB. create dump file in put dump directory !(*)=. DL
DL=.{:{.DPATH__ST

NB. dumpfactor is set from the put dictionary
df=. DUMPFACTOR__DL

NB. default dump file name is the put dictionary name
if. isempty y   do. dumpfile=. DMP__DL,DNAME__DL,IJS
elseif. badcl y do. jderr ERR0158 return. NB. error msg: invalid dump file
elseif.do. dumpfile=. y
end.

NB. HARDCODE: are we retaining object age?
if. 0=nc<'RETAINAGE__DL' do. rag=. 1 -: RETAINAGE__DL  else. rag=. 0 end.

NB. standardize path character
dumpfile=. jpathsep dumpfile

if.     badrc uv=. dumpheader dumpfile        do. uv
elseif. badrc uv=. df dumpwords dumpfile      do. uv
elseif. badrc uv=. (df,TEST) dumptm dumpfile  do. uv
elseif. badrc uv=. (df,MACRO) dumptm dumpfile do. uv
elseif. badrc uv=. (df,GROUP) dumpgs dumpfile do. uv
elseif. badrc uv=. (df,SUITE) dumpgs dumpfile do. uv
elseif. badrc uv=. dumpdictdoc dumpfile       do. uv
elseif. badrc uv=. rag dumpntstamps dumpfile  do. uv
elseif. badrc uv=. dumptrailer dumpfile       do. uv
elseif.do.
  (ok OK0151),<dumpfile
end.
)

makegs=: 4 : 0

NB.*makegs  v--  make  group   and  suite  scripts.  Objects  are
NB. assembled by name class and within class alphabetically.
NB.
NB. dyad:  iaObject makegs clName
NB.
NB.   2 makegs 'group'
```

```
'obj wf'=. x
DL=.{:{.DPATH__ST

NB. for postive option codes generate files only if the object
NB. is in the put dictionary for negative codes generate files
NB. regardless of where on the path it occurs. Generated files
NB. are ALWAYS written to the put dictionary script directory
wf=. |wf [ po=. 0<wf

NB. errmsg: invalid group/suite name
if. (isempty +. badcl) y do. jderr ERR0154 return. end.
if. badrc head=. obj getgstext__ST y do. head return. end.

NB. generate files for dictionary objects
if. (1=wf) *. po do.
  if. badrc uv=. (obj;<DL) inputdict__ST <y=. y-.' ' do. uv return. end.
end.

NB. get group or suite list and generate text
if. badrc uv=. obj gslistnl__ST y do. uv return. end.
if. isempty >1{uv do. uv=.''
else.
  if. DODEPENDENTS do.
    NB. process any dependent sections in headers and adjust lists
    if. badrc deps=. obj gdeps y do. deps return. else. deps=.}. deps end.
  else.
    deps=.''
  end.
  NB. dependents may empty group/suite list
  if. #uv=. (}.uv)-.deps do.
    if. badrc uv=. ((obj-2),0) getobjects__ST uv do. uv return. end.
    if. badrc uv=. ((obj-2),0) wttext rv uv do. uv return. end.
    uv=.rv uv
  else. uv=.''
  end.
end.

NB. trim any header and append to word or test text
if. #head=. alltrim@:lfcrtrim (1;0 1) {:: head do. uv=. head,LF,HEADEND,LF,LF,uv end.

NB. write file or return character list result
if. 1=wf do. (obj;y) writeijs uv else. ok uv end.
)


masknb=: 3 : 0

NB.*masknb v-- bit mask of unquoted comment starts.
NB.
NB. monad:  masknb ct
NB. dyad:  cl masknb ct

'NB.' masknb y
:
c =. ($y)$x E. ,y
```

```
+./\"1 c > ~:/\"1 y e. ''''
)

namecats=: 4 : 0

NB.*namecats v-- extracts and classifies names in J code.
NB.
NB. dyad:  pa namecats ctJcode
NB.
NB. name classifications
NB.   global           global reference or assignment
NB.   local            local reference of assignment
NB.   declared global  names marked with global comment tag (*)=:
NB.   declared local   names marked with local command tag (*)=.
NB.   override mixed   allow mixed assignments (<:)=:
NB.   for. local       implicit for. locals
NB.
NB.  0 namecats jcr 'wordname' NB. only globals
NB.  1 namecats jcr 'wordname' NB. full classification

if. badrc parsed=. parsecode y do.
  parsed  NB. parse error
else.
  'dgbls dlocs parsed'=. }. parsed

  NB. handle quoted assignments and implicit for. locals
  'mgbls mlocs flocs'=. extscopes parsed
```

```
NB. declarations override other scopes
mgbls=. mgbls -. dlocs [ mlocs=. mlocs -. dgbls
gbls=. dgbls,mgbls [ locs=. dlocs,mlocs

NB. pick out assignments
parsed=. parsed -. ;:')'
uv0=. parsed #~ 1|.parsed = <'=.'
uv1=. parsed #~ 1|.parsed = <'=:'

NB. forbid names from being both local and global
uv1=. uv0 -. uv0 -. uv1

NB. errmsg: mixed scopes
if. 0 < #uv1 do.
  NB. check for mixed assignment override
  if. -.MIXEDOVER +./@E. ,y do. (jderr ERR0159),uv1 return. end.
end.

uv1=. parsed -. uv0
gbls=. gbls , (jnfrblcl uv1) -. locs,JARGS

if. x do.
  NB. complete name classification
  locs=. locs,jnfrblcl uv0
  uv1=. (<gbls),(<locs),(<dgbls),(<dlocs),<flocs
  ok <GLOBCATS ,. (/:~)@:~. &.> uv1
else.
```

```
    NB. return only unique sorted globals
    ok /:~ ~. gbls
  end.
end.
)

NB. numeric list from run length encoding table - see (rlefrnl) long document
nlfrrle=: #~/@:|:

nounlrep=: 4 : 0

NB.*nounlrep v-- converts nouns stored as binary  to linear  text
NB. representations. Uses  a  scratch locale to temporarily define
NB. nouns.
NB.
NB. dyad:  iaNoex nounlrep bt

NB. override mixed assignments (<:)=:
if. #y do.
  clearso 0
  names=. (errnames=. {."1 y) ,&.> locsfx SO  NB. !(*)=. SO
  try.
    (names)=:  (3!:2)&.> {:"1 y
    names=. (5!:5@<)&.> names
  catch. (jderr ERR016),errnames return. end. NB. retain scratch on failure
  if. x do. names=. names ,&.> LF end.
  y=. names (<a:;2)} y
  clearso 0
end.
ok <y
)

opaqnames=: 4 : 0

NB.*opaqnames v-- extract  opaque names  from J  code.  An opaque
NB. name is a declared reference.
NB.
NB. dyad:

b=.  +./"1 x        NB. text mask
x=. b # x [ y=. b # y
y=. x jnb y         NB. search only comment text
if. +./ '(*)=' E. , y do.

  NB. replace any single quotes ' with blanks
  NB. quotes will confuse (masknb) below
  y=. ($y)$ ' ' ( I. (,y) = '''')} ,y

  NB. this is a rare instance of where HARDCODE is
  NB. beneficial.  The tags used to mark declared
  NB. globals and locals in J code are sprinkled
  NB. throughout many programs.  If the tags where
  NB. ever changed in this verb it would not properly
  NB. process changed tags. By hardcoding the tags
  NB. they are difficult to change which is what I want!
  locals=. (,y) #~ , '(*)=.' masknb y
  locals=. ~. <;._1 ' ',locals #~ -. '   ' E. locals
  locals=. <jnfrblcl locals
  globals=. (,y) #~ , '(*)=:' masknb y
  globals=. ~. <;._1 ' ',globals #~ -. '   ' E. globals
  globals=. <jnfrblcl globals
  locals,globals
else.
  '';''
end.
)

parsecode=: 3 : 0

NB.*parsecode v-- parses J word code.  Normal result  is a  three
NB. item boxed list of boxed lists containing declared  names and
NB. parsed tokens. Will return  an  error if  given syntactically
NB. invalid J code.
NB.
NB. monad:  parsecode cl|ctJcr
NB.
NB.   parsecode jcr 'wordname'

if. 0 e. $parsed=. tabit y do. ok'' return.
NB. possible quoted single line explicit
elseif. 1=#parsed do. parsed=. uqtsingle parsed
end.
```

```
NB. end with a blank and compute comment mask
parsed=. parsed ,"1 ' '
mask=. masknb parsed
locs=. gbls=. ''

NB. if any declared names extract them
if. 1 e. '(*)=' E. , parsed do.
  'locs gbls'=. mask opaqnames parsed
  olap=. locs -. locs -. gbls  NB. intersection
  NB. errmsg: confused declarations
  if. 0<# olap do. (jderr ERR0150),olap return. end.
end.

NB. blank comments, clear mask and remove blank rows
mask=. 0 [ parsed=. parsed jnb~ -. mask
parsed=. parsed #~ parsed +./ . ~: ' '
parsed=. (;: :: 0:)&.> <"1 parsed  NB. parse code
if. parsed e.~ <0 do.
  jderr ERR0151  NB. errmsg: word syntax
else.
  if. (0{DDEFESCS) e. parsed=. ;parsed do.
    NB. remove any direct definition escape tokens
    parsed=. parsed #~ ddefescmask parsed
  end.
  parsed=. ok(<gbls),(<locs),<parsed
end.
)
```

```
putallts=: 3 : 0

NB.*putallts v-- puts all timestamps - see (getallts).
NB.
NB. monad:  putallts btCts
NB.
NB.   cts=. getallts__MK__JODobj 0
NB.   putallts__MK__JODobj cts

NB. insure dictionaries are open
if. badrc msg=. checkopen__ST 0 do. msg return. end.

NB. HARDCODE: errmsg: invalid object timestamp table
if. -.(3,#OBJECTNC) -: $y do. jderr ERR0160 return. end.

NB. put dictionary name and object names
do=. {:{.DPATH__ST
onames=. DIRNMS__do [ dname=. DNAME__do

NB. HARDCODE: shapes
inc=. -INPUT [ ecb=. ;2{y [ nots=. 0 = #&> 0{y [ msg=. i. 0 4

for_obj. OBJECTNC do.

  NB. empty object timestamps
  if. obj_index{nots do. continue. end.
  NB. object name timestamps
  nts=. (<0 1; ,obj_index){y
  uv=. 2 , #&> 0{nts

  NB. decode any run encodings
  if. obj_index{ecb do. nts=. (<uv $ nlfrrle ;1{nts) (1)} nts end.

  NB. store timestamps - note errors but proceed
  msg=. msg , (2 {. (obj,inc) put nts) , (obj_index{onames) , <dname

end.

msg
)

NB. run list encoding from numeric list - see long document
rlefrnl=: (1 ,~ 2&(~:/\)) ({. , #);.2 ]

sexpin=: 3 : 0
NB.*sexpin v-- single line explicit definition test.
if.      EXPPFX0 e.~ 5 {. hd=. alltrim 20 {. ,y  do. 1
elseif.  EXPPFX1 e.~ 8 {. hd  do. 1  NB. monad null
elseif. do. 0
end.
)

sonl=: 3 : 0
NB.*sonl v-- scratch object namelist.
NB.
NB. monad:  sonl il

nl__SO y
)


NB. promotes only atoms and lists to tables
tabit=: ]`,:@.(1&>:@(#@$))^:2


toascii85=: 3 : 0

NB.*toascii85 v-- to ascii85 representation.
NB.
NB. From convert/misc/ascii85 addon.
NB.
NB. Converts  a  list  of  bytes to  an  ASCII85  representation:
NB. essentially all the  "visible" ASCII characters.  Useful  for
NB. encoding arbitrary byte  lists as a  portable stream. Returns
NB. lines of length no more than 75 + LF
NB.
NB. The encoding does not begin with <~, though sometimes this is
NB. allowed.  However  PDF  files  do  not  accept  this  prefix.
NB. Decoding does support the prefix.
NB.
NB. monad:  clA85 =. toascii85 cl
```

```
r=. ,y
len=. #r
assert. 4 <: len  NB. fails on short cl
r=. 256 #. _4[\ a.i.r
m=. 0 (_1) } r = 0
n=. 5 * I.m
r=. a. {~ 33 + ,(5#85) #: r
r=. 'z' n } r
m=. 1 n } 5 # -. m
r=. m # r
r=. (- (4|len) { 0 3 2 1) }. r
r=. }: ,(_75 [\ r),.LF
('~>',LF) ,~ (r i: ' ') {. r
)

uqtsingle=: 3 : 0
NB.*uqtsingle v-- unquotes single line explicit definitions
if. sexpin y do.
  m99=. '''' htclip alltrim ,y
  m99=. tabit m99 #~ -. halfbits '''' = m99
  ]`('''''&,"1)@.(':'''&-:@(2&{.@,)) m99  NB. correct dyad
else.
  y
end.
)

wraplinear=: 4 : 0
```

```
NB.*wraplinear v-- wraps the linear representation of  large  J
NB. objects into a series of script lines.
NB.
NB. The linear form  of large  J  objects  can produce  very long
NB. lines  in  script files. Many editors cannot deal  with  very
NB. long lines. This verb  produces  an equivalent representation
NB. that can always be edited.
NB.
NB. dyad:  (clTempName ; iaWidth) wraplinear clLinear
NB.        (clTempName ; iaWidth ; paAscii85) wraplinear clLinear
NB.
NB.   ('z';67) wraplinear 5!:5 <'bighonkingarray'
NB.   ('z';67;1) wraplinear btcl
NB.   ('z';67;1;1) wraplinear cl

NB. temporary noun name, line length, ascii85 representation
'temp width ascii85 tablst'=. 4 {. x,0;<0

if. ascii85 do.
  NB. use ASCII85 encoding.  This representation is
  NB. about three times more compact than the default
  NB. representation but requires roughly three times
  NB. the CPU with current algorithms to encode/decode
  decoder=. (;tablst{'dec85';'fromascii85'),'__MK__JODobj 0 :'
  temp,'=:',decoder,' 0',LF,')' ,~ toascii85 y
else.
  head=. temp,'=:'''''                NB. null header
  tail=. temp,'=:',(":#y),'{.',temp   NB. trim to correct length
  line=. temp,'=:',temp,','           NB. next line

  NB. wrap text and insure each line is properly quoted
  body=. ctl line ,"1 quote"1 (-width) ]\ y
  head,LF,body,LF,tail
end.
)

wrdglobals=: 4 : 0

NB.*wrdglobals v--  extracts names from J  words. Assumes name is
NB. valid.
NB.
NB. dyad:  pa wrdglobals clName
NB.
NB.   0 wrdglobals 'wordname'  NB. only globals
NB.   1 wrdglobals 'wordname'  NB. full name classification

code=. jcr :: 0: y
NB. errmsg: no definition
if. code -: 0 do. (jderr ERR0152),<y else. x namecats code end.
)

writeijs=: 4 : 0

NB.*writeijs v-- writes file to put dictionary directory
NB.
NB. dyad:  (iaObject ; clFile) writeijs clText

'obj file'=. x
DL=.{:{.DPATH__ST
NB. get put dictionary script directory
path=.jpathsep dfp__DL obj
m=. (toHOST y) (write :: _1:) path=.path,file,IJS
NB. errmsg: file write failure with target path and file appended
if. m -: _1 do. (jderr ERR0153),<path else. (ok OK0150),<path end.
)

wttext=: 4 : 0

NB.*wttext v-- returns annotated word or test text.
NB.
NB. This  verb converts dictionary  words and  tests to formatted
NB. script   text.  (y)   is   a  boxed   (name,class,value)   or
NB. (name,value) table.  The result is either  a single cl script
NB. or a btcl of object scripts.
NB.
NB. dyad:  (paRc ; blcl) =. iaObjExFtab wttext bt
NB.        (paRc ; btcl) =. iaObjExFtab wttext bt

NB. object code, explanation bit, formatted table bit
NB. default table bit is off - this verb is frequently
NB. called with a two item (x) argument
'obj noex nftab'=. 3{.x,0
```

```
if. WORD=obj do.
  y=. (/: ; 1 {"1 y){y      NB. sort words by name class
  nr=. ((; 1 {"1 y)>0) i. 1
  NB. convert noun values to linear representations
  if. badrc m2=. noex nounlrep nr{.y do. m2 return. end.
  y=. (rv m2) , nr}.y
end.

if. nftab do. nms=. 0 {"1 y end.  NB. retain sorted names

if. noex do.
  NB. no explanations and no LF's depends on caller
  m=. (#y)#0
elseif. +./m=. -.LF e.&> {:"1 y do.
  NB. prefix any short explanations for single line definitions
  m2=. m#{."1 y
  if. badrc et=. obj getexplain__ST m2 do. et return. end.
  m2=. 0<#&> et=. {:"1 rv et
  et=. (<"0 m2) #&.> (<'NB. ') ,&.> et ,&.> LF
  y=. (et ,&.> m#{."1 y) (<(I. m);0)} y
  NB. number of LF's between corresponding objects
  m=. (>:2*-.m) + m (#^:_1) m2
  m=. m + 2*firstone 1=m
elseif.do.
  NB. 3 LF's between all multi-line defs HARDCODE
  m=. (#y)#3
```

```
end.

NB. construct J object scripts
if. WORD=obj do. y=.jscriptdefs y else. y=. {:"1 y end.

NB. return formated (name,script) table or cl script
if. nftab do. ok <nms ,. y else. ok ({.m)}.m jscript y end.
)
```

jodutil Source Code

```
NB. *jodutil c-- a collection of JOD utility words: extension of (jod).
NB.
NB. This subclass defines a set of handy utilites that use the core
NB. facilities of JOD to perform tasks of general use to J programmers.
NB.
NB. Interface nouns & verbs:
NB.   compj    extreme compression of dictionary words
NB.   de       drop error code from JOD results
NB.   disp     display dictionary objects
NB.   doc      format comments in words and documents
NB.   ed       edit objects from JOD
NB.   et       edit text
NB.   gt       get text out of edit windows
NB.   revo     list recently revised objects
NB.   rm       run macros
NB.   rtt      run tautology tests
NB.   jodhelp  browse PDF help
NB.
NB. Notes:
NB.   error & ok messages (jodutil range 00250-00399)

coclass 'ajodutil'
coinsert 'ajod'

NB.*dependents d-- dependent words
NB. documentation mark for assumes
ASSUMESMARK=: 'assumes:'

NB. documentation mark for author
AUTHORMARK=: 'author:'

NB. documentation mark for created
CREATEDMARK=: 'created:'

NB. documentation mark for dyad hungarian and examples
DYADMARK=: 'dyad:'

NB. documentation mark for monad hungarian and examples
MONADMARK=: 'monad:'

NB. documentation mark for verbatim
VERBATIMMARK=: 'verbatim:'

NB. documentation mark for root words
ROOTWORDSMARK=: 'rootwords:'

NB. documentation marks - depends on other marks
DOCUMENTMARKS=:  ASSUMESMARK;AUTHORMARK;CREATEDMARK;DYADMARK;MONADMARK;VERBATIMMARK;ROOTWORDSMARK

NB. command line quotes OS dependent: jod !(*)=. dblquote
qt=:  ]`dblquote@.IFWIN
```

```
NB.*enddependents

NB.*end-header

NB. remove only white space tag
CWSONLY=: '(-.)=:'

NB. text editor to use when running JOD in jconsole on Windows systems
EDCONSOLE=: '"c:\Program Files\Microsoft VS Code\code.exe"'

NB. default edit file name
EDTEMP=: '99'

ERR0250=: ' is a noun no internal document'

ERR0251=: 'not loaded - load'

ERR0252=: 'not J script(s) ->'

ERR0253=: 'invalid locale name'

ERR0254=: 'unable to get TEMP/*.ijs text'

ERR0255=: 'unable to open TEMP/*.ijs for editing'
```

```
ERR0256=: 'J error in script ->'

ERR0260=: 'PDF reader not found'

ERR0261=: 'macro is not a J script - not formatted'

ERR0262=: 'not supported on current J system'

NB. jodutil interface words
IzJODutinterface=: <;._1 ' compj de disp doc ed et gt jodhelp revo rm rtt'

NB. valid characters in J names
NAMEALPHA=: 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_'

NB. obfuscate local identifiers tag
OBFUSCATE=: '(/:)=:'

NB. name obfuscation limit - higher values less likely to clash
OBFUSCCNT=: 100000

NB. obfuscation local identifier prefix
OBFUSCPFX=: 'o_'

OK0250=: ' documented in ->'

OK0251=: 'edit locale cleared'
```

```
OK0252=: 'edit locale ->'

OK0255=: 'starting PDF reader'

OK0256=: 'jod.pdf not installed - use pacman to install the addon general/joddocument'

NB. PDF document indicator
PDF=: 'PDF'

NB. PDF reader - must accept command line argument
PDFREADER=: 'C:\Program Files\Adobe\Reader 8.0\Reader\acrord32.exe'

NB. on Mac's use the OS open command for PDFs
PDFREADERMAC=: 'open'

NB. character used to mark scriptdoc headers - currently a '*'
SCRIPTDOCCHAR=: '*'

blkaft=: 3 : 0

NB.*blkaft v-- appends necessary blanks to J tokens.
NB.
NB. This verb appends some  necessary blanks to  J tokens so that
NB. raising  a token list and reparsing  produces  the same token
NB. list. A few unecessary blanks may be inserted.
NB.
NB. monad:  blkaft blcl
NB.
NB.   NB. line of J code
NB.   line=. 'c=. +./\"1 c > ~:/\"1 y. e. '''''''''
NB.   tokens=.  ;: line
NB.
NB.   NB. compare
NB.   (;: ; tokens) -: tokens
NB.   (;: ; blkaft tokens) -: tokens

NB. assume no blanks are required
r=. 0 #~ # y
t=. y

while.do.
 u=. ;: ;\ t
 v=. ~.&.>  ( <"1 |: u) -. L: 1  a:
 r=. r +. y e. ; {.&.> (1 < #&> v)#v

 if. y -: {: u do.
   NB. last tokenized row matches original
   break.
 else.
   NB. insert required blanks and reparse
   t=. ((r#t),&.>' ') (I. r)} t
 end.
end.

NB. insert required blanks and raise tokens
; ((r#y),&.>' ') (I. r)} y
)

changetok=: 4 : 0

NB.*changetok  v--  replaces  J name tokens within  a string. See
NB. long documentation.
NB.
NB. dyad:  clChanged =. clTokens changetok clStr
NB.
NB.   '/boo/hoo' changetok 'boo boo boohoo boohoo'

if. #pairs=. 2 {."(1) _2 [\ <;._1 x do.
  pairs=. pairs #~ _2 ~: (4!:0) {."1 pairs  NB. eliminate non-token pairs
end.
cnt=._1 [ lim=. # pairs
while. lim > cnt=. >:cnt do.         NB. process each change pair
  't c'=. cnt { pairs                NB. /target/change (*)=. t c
  if. +./b=. t E. y do.              NB. next if no targets
    w=. I. b                         NB. target starts
    'l o'=. #&> cnt { pairs          NB. lengths (*)=. l o
    q=. (w { ' ' , y) e. NAMEALPHA        NB. head chars
    r=. ((w + l) { y , ' ') e. NAMEALPHA  NB. tail chars
    w=. w #~ -. q +. r               NB. eliminate embedded tokens
    if. 0 = #w do. continue. end.    NB. next if no targets
    b=. 1 w} 0 #~ # b                NB. reset target mask
    p=. w + 0,+/\(<:# w)$ d =. o - l   NB. change starts
    s=. * d                          NB. reduce < and > to =
    if. s = _1 do.
      b=. 1 #~ # b
      b=. ((l * # w)$ 1 0 #~ o,l-o) (,w +/ i. l)} b
      y=. b # y
      if. o = 0 do. continue. end.   NB. next for deletions
    elseif. s = 1 do.
      y=. y #~ >: d w} b             NB. first target char replicated
    end.
    y=. (c $~ o *# w) (,p +/i. o)} y  NB. insert replacements
  end.
end. y                               NB. altered string
)

compclut=: 3 : 0

NB.*compclut  v-- removes  comments and  reduces  multiple  blank
NB. lines to singles.
NB.
NB. This  verb removes  all  comments  from  J code  and  reduces
NB. multiple blank  lines  to  one.  All  leading  whitespace  is
NB. preserved.  This representation  is surprisingly  useful when
NB. debugging and reading  code as it removes  annoying "literary
NB. artifacts" while preserving the structure of code.
NB.
NB. monad:  cl =. compclut ctJcr
NB.
NB.  complut jcr 'compclut'

t=. 0 decomm y
LF ,~ ctl t #~ (-.b) +. firstone b=. *./"1 ' '=t
)

compj=: 3 : 0

NB.*compj v-- compresses nonnouns  by removing  white  space and
NB. shortening local identifiers.
NB.
NB. (compj)  does not shorten  global identifiers, object or  locale
NB. names  and implicit local (for.) names. The names  changed  by
NB. (compj) are labeled local by (11 globs).
NB.
NB. WARNING: code compression requires that all ambigous names in
NB. J  code  are properly declared  using  (globs) scope tags. If
NB. such names are not properly identified compression will break
NB. your code.
NB.
NB. monad:  cl =. compj blclNames
NB.
NB.   compj ;:'the byte diet'
NB.
NB. dyad:  cl =. iaOption compj blclNames
NB.
NB.   1 compj ;:'remove comments preserving leading whitespace'
```

```
0 compj y
:
if. badil x do. jderr ERR001 return. end.

NB. get word definitions
if. badrc dat=. (WORD,NVTABLE) get y do. dat return. else. dat=. rv dat end.

NB. mask of non-nouns
b=. 0 < ; 1 {"1 dat

NB. set compression
cv=. compressj`compclut @. (1 -: x)

NB. compress non-nouns - remove any embedded tabs
dat=. (cv@:ctit&.> (b#{:"1 dat) -.&.> TAB) (<(I. b);2)} dat

NB. generate packed script
(WORD,1) wttext__MK dat
)

compressj=: 3 : 0

NB.*compressj v--  removes  all  white  space  from  J words  and
NB. shortens local names. This process reduces the readability of
NB. code and should only be applied to production code.
NB.
NB. monad:  cl =. compressj ct
NB.
NB.   compressj jcr 'verbname'
NB.
NB.   NB. call in object context
NB.   compressj__UT__JODobj jcr_ajod_ 'compressj_base_'

NB. check for presence of white space only removal tag
w=. 1 e. CWSONLY E. ,y

NB. always remove white space
u=. dewhitejcr y
if. w do. u return. end.

NB. do not compress identifiers in code that cannot be
NB. reliably classified by the namecats verb.

NB. BUG: j 8.05 win64 can lose y shapes - sy$,y recovers y's shape

if. badrc m=. 1 namecats__MK y do. u return. end.
d=. ~. ;(<2 3 4;1){m=. rv m

NB. check for presence of obfuscation tag
if. o=. 1 e. OBFUSCATE E. ,y do.
  NB. local names less J arguments
  l=. ;(<1;1){m
  l=. l -. JARGS__MK
else.
```

```
  NB. local names less any single char names
  l=. ;(<1;1){m
  s=. l #~ 1 = #&> l
  l=. l -. s
end.

NB. remove object references
l=. l -. exobrefs l,;(<0;1){m

NB. local names less any declared and for. names
if. 0=#m=. l -. d do. u return. end.

NB. remove any names with embedded locale references
if. 0=#m=. m #~ -. islocref&> m do. u return. end.

if. o do.
  NB. form obsfucated name replacements - drop trailing _ in (NAMEALPHA)
  bnr=. (<:#NAMEALPHA)&#.@((}:NAMEALPHA)&i.)^:_1
  r=. ' ' -.~ ,'/' ,"1  (>m) ,"1 '/' ,"1 OBFUSCPFX ,"1 bnr (#m)?OBFUSCCNT
else.
  NB. form replacements from any remaining chars !(*)=. SHORTNAMES
  NB. J arguments m n x y u v are not on SHORTNAMES
  if. 0=#r=. SHORTNAMES -. ,&.> s do. u return. end.
  if. (#r) < #m do.
    NB. we have more replacements than available SHORTNAMES
    NB. form base (#r) numbers using SHORTNAMES digits
    bnr=. (#r)&#.@((;r)&i.)^:_1
    r=. r,<"1(#r) }. bnr i. #m
  end.
  r=. ; '/' ,&.> m ,. (#m) {. r
end.

NB. replace tokens
r changetok u
)

createut=: 3 : 0

NB.*createut v-- initializes utility objects.
NB.
NB. monad:  createut bl0brefs
NB.
NB.   createut__UT JOD;ST;MK;UT;<SO

NB. object references !(*)=. JOD ST MK UT SO
'JOD ST MK UT SO'=:  y

NB. set shortnames !(*)=. SHORTNAMES
SHORTNAMES=:  ,&.> <"0 [ 52 {. NAMEALPHA
SHORTNAMES=:  SHORTNAMES -. ;:'m n x y u v'

NB. add to overall jod interface
IZJODALL__JOD=:  IZJODALL__JOD,IzJODutinterface

NB. define direct (ijod) locale interface for utilities
".&.> UT defzface IzJODutinterface
)

NB. convert LF delimited character lists to character tables
ctit=: [: ];._2 ] , (10{a.)"_

NB. display JOD result without return code
de=: list@:}.

dewhitejcr=: 3 : 0

NB.*dewhitejcr  v--  removes  all redundant  blanks from  J code.
NB. Result is a character list in linear representation format.
NB.
NB. monad:  cl =. dewhitejcr ct
NB.
NB.   dewhitejcr jcr 'anyword'

tt=. ;:&.> <"1 (ljust@:decomm) y  NB. list of tokenized lines
; (blkaft&.> tt) ,&.> LF          NB. insert blanks, LF's and raise
)

dewhitejscript=: 3 : 0

NB.*dewhitejscript  v--  removes  all  redundant  blanks  from  J
NB. scripts.
NB.
NB. monad:  dewhitejscript cl
NB.
NB.   dewhitejscript read 'c:\any\j\script.ijs'

NB. replace any tabs with single blanks
y=. ' ' (I. y=TAB)} y

NB. remove blank lines and all comments from script
y=. ;:&.> <"1 decomm ];. _1 LF,y-.CR

NB. remove redundant blanks in code
; (blkaft&.> y) ,&.> <CRLF
)


disp=: 3 : 0

NB.*disp  v--  display  dictionary  objects  as text.  This  verb
NB. returns  a character  list instead of  a the usual (rc;values
NB. ...) boxed list.
NB.
NB. monad:  disp cl/blcl
NB. dyad:  iaObject disp cl/blcl

0 disp y
:
if. badrc uv=. x obtext y do. uv else. >{:uv end.
)
```

```
doc=: 3 : 0

NB.*doc v-- formats document text  using  the conventions  of the
NB. (docct) verb.
NB.
NB. monad:  doc clName
NB.
NB.   doc 'word'    NB. format leading block of explicit defn comments
NB.
NB. dyad:  iaObject doc clName
NB.
NB.   1 doc 'test'       NB. format test document text
NB.   0 9 doc 'longdoc'  NB. format long word documentation text

docword y
:
x doctext y
)

docct2=: 4 : 0

NB.*docct2 v-- formats leading comments.
NB.
NB. This verb formats the  leading comments in a character table.
NB. There  are  three  basic  types  of  tables:  (1)  character
NB. representations  of  explicit  words with  leading  contigous
NB. comment blocks, (2) general J  scripts with leading contigous
NB. comment  blocks,  (3)  long  JOD documentation  text  without
NB. leading comments (no 'NB.'s).  Long documentation follows the
NB. same formatting conventions without the leading 'NB.'s
NB.
NB. Result is a character table.
NB.
NB. dyad:  ctFormatted =. (iaWidth;iaStarPos;iaBlockIdx;clPfx) docct2 ctText
NB.
NB.   (41;0;1;'NB.') docct2__UT__JODobj ];._1 LF,disp 'docct2'             NB.(1)
NB.   (40;0;0;'NB.') docct2__UT__JODobj ];._1 LF,(4 disp 'scriptstub')-.CR NB.(2)
NB.   (57;0;0;'') docct2__UT__JODobj ];._1 LF,(4 disp 'docstub')-.CR       NB.(3)

NB. width, star-row, block-index, prefix
'wid star blidx pfx'=. x
plen=. #pfx

NB. get any first block of comments
if. plen do.
  if. -. +./b1=. +./"1 (,: pfx) E. y do. y return. end.
  txt=. ((firstone b1) +. firstone -.b1) <;.1 y
else.
  NB. the prefix is null - the first comment block is all text
  txt=. <y
end.

if. blidx >: #txt do. y return. else. blk=. > blidx { txt end.
if. +./ (star,plen) >: $blk do. y return. end.
```

```
NB. apply formatting only to scriptdoc'ed text
if. SCRIPTDOCCHAR=(<star;plen){blk do.

  NB. clear scriptdoc mark remove any prefixes
  blk=. ' ' (<star;plen)} blk
  if. plen do. blk=. (#pfx) }."1 ljust blk end.

  NB. format comments remark for scriptdoc
  'head tail'=. (wid;DOCUMENTMARKS) docfmt2 blk
  head=. SCRIPTDOCCHAR (<star;plen)} (pfx,' ') ,"1 head
  tail=. pfx ,"1 tail

  NB. return formatted text
  ; (<head,tail) (blidx)} txt
else.
  y
end.
)

docfmt2=: 4 : 0

NB.*docfmt2 v-- formats comment region.
NB.
NB. dyad:  (iaWid ; blclMarks) docfmt2 ct
NB.
NB.  (67;MONADMARK;DYADMARK) docfmt2 5#,:'to comment or not to comment'

NB. text width and n marks
```

```
width=. >{. x
marks=. }. x

NB. leave all text following any marks alone
b=. +./"1 +./ (,:&.> marks) E.&> <y
b=. 1 (0)} firstone +./\ b
'head tail' =. 2 {. (b <;.1 y),<i.0 0

NB. format paragraphs of head
head=. ljust head
head=. width textform2&.> (1(0)}*./"1' '=head)<;.1 head

NB. remove null paragraphs, remerge and mark for scriptdoc
head=. (0 < #&> head)#head
head=. (-0=#tail) }. ;head,&.> ' '

NB. return formatted text and unformatted tail
head;tail
)

doctext=: 4 : 0

NB.*doctext v-- formats long document, object and header text.
NB.
NB. dyad:  iaObject doctext clName
NB.
NB.   0 doctext 'word'
NB.   1 doctext 'test'
```

```
if. badcl y do. jderr ERR001  NB. errmsg: invalid options
elseif. badrc uv=. checkput__ST y   do. uv
elseif. badrc uv=. checknames__ST y do. uv
elseif. ((1=#x) *. ({.x) e. TEST,MACRO) +. x e. (GROUP,SUITE) ,. 1  do.
        if. badrc uv=. x obtext y do. uv return. else. uv=. >{:uv end.
        NB. format leading comments of test, macro and group/suite header scripts
        DL=.{:{.DPATH__ST
        uv=. ctl (DOCUMENTWIDTH__DL;0;0;'NB.') docct2 ];._1 LF,uv-.CR
        if. x-:MACRO do.
          NB. format only J script macros
          if. badrc uv2=. (MACRO,INCLASS) get y do. uv2 return. end.
          NB. errmsg: macro is not a J script - not formatted
          if. JSCRIPT=>{:uv2 do. x put y;JSCRIPT;uv else. jderr ERR0261 end.
        else.
          x put y;uv
        end.
elseif. -. (<x) e. {OBJECTNC;DOCUMENT do. jderr ERR001
elseif. y=. }. uv
        DL=.{:{.DPATH__ST
        badrc uv=. ((x=.{.x);<DL) inputdict__ST y do. uv
elseif. badrc uv=. x getdocument__ST y do. uv
elseif.do.
  NB. document text using same formatting
  NB. conventions applied to words.
  uv=. (1;0 1){:: uv
  uv=. ];. _2 (uv -. CR),LF
  uv=. ctl ; (DOCUMENTWIDTH__DL;DOCUMENTMARKS) docfmt2 uv
  (x,DOCUMENT) put y,<uv
end.
)

docword=: 3 : 0

NB.*docword v-- formats the leading  comment block  in dictionary
NB. verbs, adverbs and conjunctions.  Nouns do not have  internal
NB. documentation. Attempts  to  document  a noun  results in  an
NB. error.
NB.
NB. Note: nouns do have  external  documentation in  the form  of
NB. short explanations and supplemental document text. See  (put)
NB. and (get).
NB.
NB. monad:  docword clName

if. badcl y do. jderr ERR001
elseif. badrc uv=. checkput__ST y do. uv
elseif. badrc uv=. checknames__ST y do. uv
elseif. y=. }. uv
        DL=.{:{.DPATH__ST
        badrc uv=. (WORD;<DL) inputdict__ST y do. uv
elseif. badrc uv=. (WORD,0) getobjects__ST y do. uv
elseif. 0 = (1;0 1){:: uv do.
  NB. errmsg: is a noun no internal document
  jderr '<',(' ' -.~ ,>y),'>' , ERR0250
elseif. cr=. (1;0 2){:: uv
        cr=. (-LF = {:cr) }. cr,LF
        NB. cr=. ctl DOCUMENTWIDTH__DL docct ];._2 cr  NB. OLDCODE
        cr=. ctl (DOCUMENTWIDTH__DL;0;1;'NB.') docct2 ];._2 cr
        uv=. , 1 {:: uv
        uv=. (<cr) 2} uv
        badrc msg=. (WORD,NVTABLE) put uv do. msg
elseif.do.
  (ok '<',(>{.uv),'>',OK0250),{:msg
end.
)

ed=: 3 : 0

NB.*ed v-- edit dictionary objects.
NB.
NB. (ed) typically fetches, formats and places object(s) in an edit window.
NB.
NB. monad:  ed cl/blcl/bt
NB.
NB.   ed 'wordname'
NB.
NB.   ed ;:'many words mashed into one edit script'
NB.
NB.   NB. edit contents of (name,value) and (name,class,value) tables
NB.   ed ; }. 0 10 get }. dnl 're'
NB.   ed ; }. 4 get }. 4 dnl 'build'
NB.
NB.   NB. place many backup versions in edit window
NB.   ed ; }. bget <;._1 ' word.12 word.11 word.09 word.02'
NB.   ed ; }. 4 bget <;._1 'macro.9 macro.7 macro.2'
NB.
NB. dyad:  iaObject|ilObjOpt ed cl|blcl
NB.
NB.   1 ed 'testname'           NB. edit test
NB.   0 9 ed 'worddocument'     NB. document text associated with word
NB.   2 ed 'group'              NB. generate entire group script and edit
NB.   2 1 ed 'grouptext'        NB. edit only group text

0 ed y
:
if. 2=#$ y do.
  if. badrc uv=. checknttab3 y do. uv return.
  elseif. 3 = {:$ uv=. rv uv   do.
    if. 3 >: <./ jc=. ;1{"1 uv do.
      NB. convert binary nouns to linear representations
      jc=. I. 0=jc
      if. badrc nv=. 0 nounlrep__MK jc{uv do. nv return. end.
      uv=. (rv nv) jc} uv
      NB. format words for editing
      text=. _2 }. ; (0 {"1 uv) ,. (<'=:') ,. (2 {"1 uv) ,. <2#LF
    else.
      NB. format non words for editing
      text=. _2 }. ; ({:"1 uv) ,&.>  <2#LF
    end.
  elseif.do.
    NB. format non words for editing
    text=. _2 }. ; ({:"1 uv) ,&.>  <2#LF
  end.
  NB. set default object name - if there is more than one
  NB. object reset (x) to prevent affixing document command
  oname=. ;0{0{uv [ x=.  1 < #uv
elseif. badrc uv=. x obtext y do. uv return.
elseif.do.
  'oname text'=. }.uv
end.

NB. append user defined document command
NB. the pattern {~N~} is a name placeholder, e.g.
NB.   DOCUMENTCOMMAND_ijod_ =: 'showpass pr ''{~N~}'''
NB. append only when editing single words
if. (x -: 0) *. wex <'DOCUMENTCOMMAND_ijod_' do.
  text=. text,LF,LF,('/{~N~}/',oname) changestr DOCUMENTCOMMAND_ijod_
end.

oname et text
)

et=: 3 : 0

NB.*et v-- edit text
NB.
NB. monad:  et clText
NB. dyad:  clFile et clText

EDTEMP et y  NB. default edit file
:
NB. write to J temp directory - created by J install
try.

  (toHOST y) write file=. jpath '~temp/' , x , IJS

  NB. open in various editors !(*)=. IFJ6 IFWIN IFJHS IFQT IFIOS IFGTK open

  NB. J6 no longer supported
  NB. if. */ wex ;:'IFJ6 IFWIN'  do.
  NB.  if. IFJ6 * IFWIN do. smopen_jijs_ file return. end. NB. J 6.0x win systems
  NB. end.

  if. IFQT do. open file  NB. jqt ide

  elseif. IFJHS do.
    NB. show edit command in JHS to remind users to adjust
    NB. browser pop ups and keep a handy recall line
    0 0$(1!:2&2) 'edit_jhs_ ',(quote file), '  NB. allow browser pop ups'
    edit_jhs_ file

  NB. running in jconsole on Windows systems
  NB. WARNING: there is no indication of fork failures
  NB. testing the existence of (EDCONSOLE) and the alleged
  NB. (file) for every edit operation would slow down normal use
  elseif. IFWIN *. IFJHS +: IFQT do. fork_jtask_ EDCONSOLE,' ',file

  NB. remaining editors are marginal, deprecated or rarely used with JOD

  NB. iPhone/iPad
  elseif. IFIOS do. je_z_ file

  NB. GTK systems are deprecated and no longer supported
  NB. elseif. wex <'IFGTK' do.
  NB.  if. IFGTK do. open_jgtk_ file  else. jderr ERR0255 end. NB. GTK

  elseif.do. jderr ERR0262  NB. errmsg: not supported on current J system
  end.

catch. jderr ERR0255  NB. errmsg: unable to open TEMP/*.ijs for editing
end.
)

NB. extract object references from blcl of names
exobrefs=: a:"_ -.~ [: ~. [: ; [: <;._1&.> ([: +./\&.> (<'__')"_ E.&.> ]) #&.> ]


gt=: 3 : 0

NB.*gt v-- get J script text from J temp directory.
NB.
NB. monad:  gt cl|zl
NB.
NB.    gt ''  NB. read text in 99 file
NB.    gt 'whatever'

if. isempty y do. y=.'99' end.
NB. use J temporary edit directory
NB. (jpath) is a J system utility loaded by standard profile
try.   read jpath  '~temp\'  ,y , IJS
catch. jderr ERR0254
end.
)

NB. formats (jodhelp) command line and spawns browser or pdfreader
jodfork=: [: fork_jtask_ [: ; 1 0 2 {  ' ' ; qt

jodhelp=: 3 : 0

NB.*jodhelp v-- display PDF JOD help.
NB.
NB. monad:  jodhelp uuIgnore
NB.
NB.    jodhelp ''  NB. display JOD help - start PDF browsing

jodpdf=. jpath '~addons/general/joddocument/pdfdoc/jod.pdf'
if. fex<jodpdf do.
  NB. jod.pdf is installed and local
  pdfrdr=. pdfreader 0
  if. UNAME-:'Darwin' do.
    NB. require 'task' !(*)=. shell
    ok OK0255 [ shell pdfrdr,' ',qt jodpdf NB. msg starting PDF reader
  elseif. fex<pdfrdr do.
    NB. spawn PDF browse task - requires configured PDF reader on host
    ok OK0255 [ jodfork pdfrdr;jodpdf
  elseif.do.
    (jderr ERR0260),<pdfrdr NB. errmsg: PDF reader not found
  end.
else.
  NB. jod.pdf is not installed advise user to download joddocument addon
  ok OK0256 NB. msg: jod.pdf not installed - use pacman to install the addon general/joddocument
end.
)

NB. left justify table
ljust=: ' '&$: :(] |."_1~ i."1&0@(] e. [))

obtext=: 4 : 0

NB.*obtext v-- assembles and returns object text
NB.
NB. dyad:  bt =. iaObject obtext blcl

if. badrc text=. checkopen__ST y do. text return. end.
select. x
case. WORD do.
```

```
  if. badrc y=. checknames__ST y do. y return. else. y=. }.y end.
  if. badrc text=. (WORD,NVTABLE) get y do. text return. end.
  if. badrc text=. WORD wttext__MK rv text do. text return. else. text=. rv text end.
  file=. >{.y
case. DICTIONARY do.
  if. badrc text=. DICTIONARY get '' do. text return. else. text=. rv text end.
  file=. (' ' -.~ ;0{0{DPATH__ST__JODobj),'_DTXT' NB. HARDCODE document text suffix
case. SUITE;GROUP do.
  if. badrc text=. (x,2) make y do. text return. else. text=. rv text end.
  file=. y -. ' '
case. TEST;MACRO do.
  if. badrc y=. checknames__ST y do. y return. else. y=. }.y end.
  if. badrc text=. x get y do. text return. end.
  if. badrc text=. x wttext__MK rv text do. text return. else. text=. rv text end.
  file=. >{.y
case.do.
  if. (<x) e. {(SUITE,GROUP);HEADER  do.
    NB. group and suite headers are frequently edited
    if. badcl y do. jderr ERR0154__MK return. end.
    if. badrc uv=. ({.x) get y do. uv return. else. 'file text'=. , rv uv end.
  elseif. (<x) e. ,{OBJECTNC;DOCUMENT,EXPLAIN do.
    NB. get object documentation text
    if. badrc uv=. x get y do. uv return.
    else.
      NB. merge all document texts
      file=. >{.{. uv=. rv uv
      text=. ; ({:"1 uv) ,&.> <2#LF  NB. HARDCODE 2
```

```
     end.
  elseif.do.
    jderr ERR001 return. NB. errmsg: invalid option(s)
  end.
end.
ok file;text
)

pdfreader=: 3 : 0

NB.*pdfreader v-- returns a pdf reader from available options.
NB.
NB. monad:  clPDFExe =. pdfreader uuIgnore

NB. prefer J's pdf readers otherwise take JOD reader !(*)=. PDFREADER
if. wex<'PDFREADER__UT__JODobj' do. pdfrdr=. PDFREADER__UT__JODobj else. pdfrdr=.'' end.

NB. on Mac's use the open command for PDF's
if.     UNAME-:'Darwin'    do. pdfrdr=. PDFREADERMAC
elseif. wex<'PDFReader_j_' do. if. 0<#PDFReader_j_ do. pdfrdr=. PDFReader_j_ end. NB. J 7.0x
elseif. wex<'PDFREADER_j_' do. if. 0<#PDFREADER_j_ do. pdfrdr=. PDFREADER_j_ end. NB. J 6.0x
end.

pdfrdr
)

reb=: 3 : 0
```

```
NB.*reb v-- removes redundant blanks - leading, trailing multiple
NB.
NB. monad:  reb cl
NB. dyad:  ua reb ul

' ' reb y
:
y=. x , y
b=. x = y
}.(b*: 1|.b)#y
)

revo=: 3 : 0

NB.*revo v-- recently revised objects. Lists recently changed put
NB. dictionary objects in order of latest to oldest.
NB.
NB. monad:  revo zl | cl
NB.
NB.   revo ''     NB. all put dictionary words in revision order
NB.   revo 'pat'  NB. recently changed words beginning with 'pat'
NB.
NB. dyad:  iaObject revo zl | cl
NB.
NB.   1 revo ''   NB. all revised tests
NB.   2 revo 'g'  NB. recently changed groups beginning with 'g'

WORD revo y  NB. word default
```

```
:
if. badil x do. jderr ERR001
elseif. badrc uv=. ((x=.{.x),_1) dnl y do. uv  NB. HARDCODE _1
elseif. isempty new=.rv uv do. ok new  NB. no matches
elseif.do.
  age=. rv (x,INPUT) get new    NB. last put timestamps
  ok (\: age) { new
end.
)

rm=: 3 : 0

NB.*rm v-- runs J macro scripts
NB.
NB. monad:  rm cl|blcl
NB. dyad:  pa rm cl|blcl

NB. (/:)=: obfuscate names
0 rm y
:
if. badrc uv=. MACRO get y do. uv return. end.
uv=. rv uv

if. *./um=. JSCRIPT = ; 1 {"1 uv do.

  scr=. ;({:"1 uv) ,&.> LF
  curr=. 18!:5 ''
  try.
    NB. j profile !(*)=. cocurrent
    NB. run from base, (display default, suppress x.-:1) stop on errors
    cocurrent 'base'
    if. x-:1 do. 0!:100 scr else. 0!:101 scr end.
    cocurrent curr
  catchd.
    cocurrent curr  NB. restore locale
    (jderr ERR0256),<13!:12 '' return.
  end.

else.
  NB. errmsg: not J script(s)
  (jderr ERR0252),(-.um)#{."1 uv
end.
)

rtt=: 3 : 0

NB.*rtt v-- runs J test scripts
NB.
NB. monad:  rtt cl|blcl
NB.
NB.   rtt 'runmytautology'
NB.   rtt ;: 'run these tautology tests in order'
NB.
NB. dyad:
NB.
NB.   0 rtt 'tautology'
NB.   1 rtt 'silenttautology'
NB.   2 rtt 'plaintest'
NB.   3 rtt 'suite'   NB. make and run tautology test suite
NB.   4 rtt 'suite'   NB. make suite and run silently

0 rtt y
:

NB. HARDCODE: option codes (/:)=: obfuscate names
if. (3-:x) +. 4-:x do.
  if. badrc uv=. (SUITE,_2) make y do. uv return. end.
  scr=.rv uv
  x=. x-3 NB. run option
else.
  if. badrc uv=. TEST get y do. uv return. end.
  uv=. rv uv
  scr=. ;({:"1 uv) ,&.> LF
end.

curr=. 18!:5 ''

NB. j profile !(*)=. cocurrent
NB. run from base, (display default, suppress x-:1), stop on errors
cocurrent 'base'
try.
  if. 0-:x     do. 0!:2 scr
  NB. Note: silent execution that fails suppresses all output
  elseif. 1-:x do. (] [ 1!:2&2) 0!:3 scr
  elseif. 2-:x do. 0!:001 scr
  elseif.do.
    cocurrent curr
    jderr ERR001 return.
  end.
catchd.
  cocurrent curr
  (jderr ERR0256),<13!:12 '' return.
end.

NB. back to original locale
cocurrent curr
)

textform2=: 63&$: :(4 : 0)

NB.*textform2 v-- wraps and justifies character table (y).
NB.
NB. This  verb forms an  (n*len)  character matrix. The blanks in
NB. each  row of the output matrix are padded so that the line is
NB. right  and left justified.  The number  of rows in the output
NB. table depends upon how many are needed to hold the input data
NB. in the justified format.
NB.
NB. Note: This verb is  a verbatim translation of  an APL utility
NB. and has not been optimized for J.
NB.
NB. monad:  cmWrap =. textform2 c[0..2]Text
NB.
NB.   textform2 1000$' How can I justify this ehh. '
NB.
NB. dyad:  cmWrap =. iaWidth textform2 c[0..2]Text
NB.
NB.   50 textform2 10#,:' four score and seven years ago our '

i=. 0
v=. reb , y ,"1 ' '
j=. #v
b=. j$0
while. j > a=. i + x do.
  k=.  i + i. >:a - i
  if. #c=. (' ' = k{v)#k do.
    i=. >: {: c
    g=. ({:k) - <:i
    c=. (1 >. <:#c) {. c
    f=. #c
    d=. f $ <. g%f
    d=. (>:{.d) (i. f|g)} d
    b=. d ((f?f){c)} b
  else.
    b=. 1 a} b
    i=. a
  end.
end.
v=. (>:b) # v
e=. >: x
r=. >.(#v) % e
(r,x) {. (r,e)$(e*r){.v
)
```

jodtools Source Code

```
NB.*jodtools c-- derived tools class: extension of (jodutil).
NB.
NB. Interface words:
NB.  addgrp   add words/tests to group/suite
NB.  allnames combines names from (refnames) and (obnames)
NB.  allrefs  all names referenced by objects on name list
NB.  delgrp   remove words/tests from groups/suites
NB.  getrx    get required to execute
NB.  hlpnl    displays short descriptions of objects on (y)
NB.  jodage   days since last change and creation of JOD objects
NB.  jodhelp  display PDF JOD help
NB.  lg       make and load JOD group
NB.  mls      make load script
NB.  noexp    returns a list of objects with no explanations
NB.  notgrp   words or tests from (y) that are not in groups or suites
NB.  nt       gets name and text from edit windows
NB.  nw       edit a new explicit word using JOD conventions
NB.  obnames  unique sorted object and locale names from (uses) result
NB.  pr       put and cross reference a word - very handy as an editor DOCUMENTCOMMAND
NB.  refnames unique sorted reference names from (uses) result
NB.  revonex  returns a list of put dictionary objects with no explanations
NB.  swex     set short word explaination from (doc) header
NB.  usedby   returns a list of words from (y) that DIRECTLY call words on (x)
NB.
NB. Notes:
```

```
NB.   Error messages (jodtools range 00400-001000)

(9!:41) 0  NB. discard whitespace

coclass 'ajodtools'
coinsert  'ajodutil'


NB.*end-header


NB. jodage header text
AGEHEADER=: <;._1 '|Name|Date First put|Days from First put|Date Last put|Days from Last put'

NB. carriage return character
CR=: 13{a.

NB. (nw) edit text template - stored in this form to preserve embedded comments
DOCUMENTMARK=: 123 126 78 126 125 61 58 32 123 126 67 126 125 32 58 32 48 10 10 78 66 46 42 123 126 78 126
>..>125 32 123 126 84 126 125 45 45 32 119 111 114 100 116 101 120 116 10 78 66 46 10 78 66 46 32 109 111 110
>..>97 100 58 32 32 123 126 78 126 125 32 63 63 10 78 66 46 32 100 121 97 100 58 32 32 63 63 32 123 126 78 126
>..> 125 32 63 63 10 10 39 78 73 77 80 32 123 126 78 126 125 39 10 41{a.


ERR00400=: 'load script is not unique - edit startup.ijs ->'

ERR00401=: 'tag error in startup.ijs file ->'
```

```
ERR00402=: 'cannot write/create startup.ijs file ->'

ERR00403=: 'invalid make load script option (0 or 1)'

ERR00404=: 'J script error in group ->'

ERR00405=: 'words refer to objects/locales ->'

ERR00406=: 'invalid delimiter'

ERR00407=: 'ROOTFOLDER must be a character list configured (jpath) expression like: ~user/jodroot'

ERR00408=: 'unable to write load script ->'

NB. locgrp Group Suite display text
GROUPSUITES=: <;._1 ' Groups Suites'

NB. JODTOOLS interface - loaded into (ijod) - see (setjodinterface)
IzJODtools=: <;._1 ' addgrp allnames allrefs delgrp fsen getrx hlpnl jodage lg locgrp mls noexp notgrp nt n
>..>w obnames pr refnames revonex swex usedby'


NB. comment tag marking end of scripts
JODLOADEND=: 'NB.</JOD_Load_Scripts>'
```

```
NB. comment tag marking start of scripts
JODLOADSTART=: 'NB.<JOD_Load_Scripts>'

NB. JODTOOLS version, make and date
JODTOOLSVMD=: '1.0.23 - dev';29;'26 Jan 2023 15:15:16'

NB. line feed character
LF=: 10{a.

OK00400=: 'load script saved ->'

OK00401=: 'file saved ->'

OK00402=: ' added to ->'

OK00403=: ' deleted from ->'

OK00404=: ' group loaded'

OK00405=: ' group loaded with postprocessor'

OK00406=: ') words loaded into -> '

NB. postprocessor prefix
POSTAMBLEPFX=: 'POST_'
```

```
NB. name of test used as a template
TESTSTUB=: 'teststub'

WARNING00400=: 'NB. WARNING: JOD managed section do not edit!'

NB. words tests display text
WORDTESTS=: <;._1 ' words tests'

addgrp=: 4 : 0

NB.*addgrp v-- add words/tests to group/suite.
NB.
NB. monad:  clGroup addgrp blclNames
NB.         (clGroupSuite;iaObject) addgrp blclNames
NB.
NB.   'jodhlp' addgrp  ;:'addgrp delgrp'
NB.   ('testsuite';3) addgrp ;:'test and moretests'

'group code'=. 2{.(boxopen x),<2
uv0=. code grp group
if. 0=>{.uv0 do. uv0
elseif. 1=>{.uv0=.code grp (group;}.uv0),y=.boxopen y do.
  gtyp=.,>(code=2 3)#WORDTESTS
  ok ((":#y),' ',gtyp,OK00402);group NB. okmsg: added to
elseif.do. uv0
end.
)
```

```
addloadscript=: 4 : 0

NB.*addloadscript  v--  inserts  (mls)  generated   scripts  into
NB. startup.ijs.
NB.
NB. Changed: 08jun12 this verb was modifying the scripts.ijs file
NB. in the  J system tree. This file is now frequently updated by
NB. JAL so startup.ijs is now modified.
NB.
NB. Changed:  11feb02 j  7.01  introduced Public_j_  in place  of
NB. PUBLIC_j_  modified   to   use  new  noun.   Path  separation
NB. characters also standardized.
NB.
NB. dyad:  baPublic addloadscript (clGroup ; clPathGroup)

NB. standardize path separation character
y =. jpathsep&.> y

if. 1=x do.

  NB. get startup.ijs
  NB. J path utility !(*)=. jpath
  tags=. JODLOADSTART;JODLOADEND
  if. fex<cfg=. jpath '~config/startup.ijs' do.
    scripts=. read cfg
    'p c'=. tags betweenidx scripts
  else.
    NB. no startup.ijs
    p=. scripts=. ''
  end.

  if. 1=#p do.
    if. badrc ld=. (;p{c) addloadscript1 y do. ld return. else. ld=.>1{ld end.
    NB. insure 'buildpublic' text starts with an LF
    mlscfg=. toHOST ;(<(LF }.~ LF-:{.ld),ld) p} c
  elseif. 0=#p do.
    NB. no JOD load scripts append current
    ld=. (0{tags),(<LF,'buildpublic_j_ 0 : 0',LF),(0{y),(<'  '),(1{y),(<LF,')',LF),1{tags
    mlscfg=. toHOST scripts , (2#LF), WARNING00400 , LF , ;ld
  elseif.do.
    NB. errmsg: tag error in startup.ijs file
    (jderr ERR00401),<cfg return.
  end.

  NB. create/update startup.ijs
  if. _1 -: mlscfg (write :: _1:) cfg do.
    NB. errmsg: cannot write/create startup.ijs file
    (jderr ERR00402),<cfg return.
  end.

  NB. directly update public script noun if present
  y=.  y ,&.> '';IJS
  if.     wex <'Public_j_' do. Public_j_=:  Public_j_ updatepublic y  NB. J 7.0x
  elseif. wex <'PUBLIC_j_' do. PUBLIC_j_=:  PUBLIC_j_ updatepublic y  NB. J 6.0x
```

```
  end.

  ok OK00400;1{y  NB. okmsg: load script saved
elseif. 0=x do.
  ok OK00401;(1{y) ,&.> <IJS NB. okmsg: file saved
elseif.do.
  NB. errmsg: invalid make load script option (0 or 1)
  jderr ERR00403
end.
)

addloadscript1=: 4 : 0

NB.*addloadscript1 v-- appends or replaces a script in the load script section of startup.ijs
NB.
NB. dyad:  clJODLoadScripts addloadscript1 (clGroup ; clPath)

NB. insure we have text
if. 0=#x do. ok x return. end.

NB. cut into lines
ldl=. <;._1 ((LF={.x)}.LF),x -. CR

NB. search for group name - can occur at most once
NB. searches only group names ignoring path file text
msk=. (' '&beforestr &.> ldl) e. 0{y
if. 1 >: +/msk do.
```

```
  NB. load script name and path
  scr=. <;(<'  ') (1)} 1 0 1 #^:_1 y

  NB. add extension if missing
  if. -.IJS -: ;(-#IJS) {.&.> scr do. scr=. scr ,&.> <IJS end.

  NB. if name exists replace it else add it at end
  if. +./msk do.
    ldl=. scr (I. msk)} ldl
  else.
    NB. find ) and insert before
    msk=. 1 ,~ -. (ldl -.&.> ' ') e. <,')'
    ldl=. scr (I. -.msk)} msk #^:_1 ldl
  end.

  NB. return modified
  ok }. ; LF ,&.> ldl
else.
  NB. errmsg: load script is not unique
  (jderr ERR00400),0{y
end.
)

NB. all names from uses:  allnames 31 uses 'name'
allnames=: ~.@('__'&beforestr&.>@obnames , refnames)

NB. all nonlocale name references: allrefs ;:'return my references'
allrefs=: [: /:~ [: ~. ] , [: refnames 31&uses
```

```
betweenidx=: 4 : 0

NB.*betweenidx v-- indexed sublists between nonnested delimiters.
NB.
NB. Cuts  up  lists  containing  balanced  nonnested  start/end
NB. delimiters into boxed lists of indexed sublists.
NB.
NB. Note:  this  verb does a simple count for  delimiter balance.
NB. This  is  a  necessary  but  not  sufficient  condition  for
NB. delimiter balance.
NB.
NB. dyad:  (ilIdx ;< blcl) =. (clStart;clEnd) betweenidx cl
NB.        (ilIds ;< blnl) =. (nlStart;nlEnd) betweenidx nl
NB.
NB.   ('start';'end') betweenidx 'start yada yada end boo hoo start ahh end'
NB.
NB.   '{}' betweenidx 'go{ahead}{}cut{me}{up}{}'
NB.
NB.   NB. also applies to numeric delimiters
NB.   (1 1;2 2) betweenidx 1 1 66 666 2 2 7 87 1 1 0 2 2

if. #y do.
  's e'=. x                    NB. start/end delimiters
  assert. -. s -: e            NB. they must differ
  em=. e E. y                  NB. end mask
  sm=. (-#s) |.!.0 s E. y      NB. start mask
  mc=. +/sm                    NB. middle count
  assert. mc=+/em                   NB. delimiter balance
  c=. (1 (0)} sm +. em) <;.1 y      NB. cut list

  NB. insert any missing middles to insure all indexed
  NB. sublists correspond to a location in the cut list
  ex=. 1 #~ >: +: mc
  ex=. (-. sm {.;.1 em) (>: +: i. mc)} ex
  c=. ex #^:_1 c

  ((# i.@#) (#c)$0 1);<c            NB. prefix indexes
else.
  (i.0);<y                          NB. empty arg result
end.
)

createjodtools=: 3 : 0

NB.*createjodtools v-- initializes new jod tools object
NB.
NB. monad:  createjodtools blclObjects
NB.
NB.   JODtools_ijod_=: conew 'ajodtools'         NB. new tools object
NB.   createjodtools__JODtools JODtools,JODobj   NB. pass self and tools

NB. use JOD object reference to locate extant subobjects
NB. Note: currently these object references are not used
NB. but are defined so that native JOD words can be accessed
NB. by words in JODtools instances in future additions to this class
NB. !(*)=. ST MK UT SO
self=.0{y [ jod=.1{y
ST=: ST__jod
MK=: MK__jod
UT=: UT__jod
SO=: SO__jod

NB. append object reference to list of JOD extensions
NB. adding to this list allows (destroyjod) to destroy
NB. all JOD extension objects with JOD core objects
JODEXT__jod=:  JODEXT__jod,self

NB. add tool words to overall JOD (ijod) locale interface
NB. (*)=. IZJODALL JODEXT
IZJODALL__jod=:  IZJODALL__jod,IzJODtools,<'JODtools'

NB. define direct (ijod) locale interface for tools - if the (ijod)
NB. trap word (jodsf) exists define an error trapping interface
(i.0 0)"_ ".&.> self defzface IzJODtools
)

dayage=: 3 : 0

NB.*dayage v-- age in days.
NB.
NB. monad:  dayage ilYYYYMMDD
NB.
NB.   dayage 1953 7 2
NB.
NB. dyad:  pa dayage iaYYYYMMDD | iuYYYYMMDD
NB.
NB.   1 dayage 4 4$20000101 19500202 19000303
NB.   0 dayage 1986 8 14

0 dayage y
:
if. x do. n=. today~ 0 else. n=. today 0 end.
(x todayno n) - x todayno y
)

delgrp=: 4 : 0

NB.*delgrp v-- remove words/tests from groups/suites.
NB.
NB. monad:  clGroup delgrp blclNames
NB.         (clGroupSuite;iaObject) delgrp blclNames
NB.
NB.   'jodhlp' delgrp  ;:'addgrp delgrp'
NB.   ('testsuite';3) delgrp ;:'test and moretests'

'group code'=. 2{.(boxopen x),<2
uv0=. code grp group
if. 0=>{.uv0 do. uv0
elseif. 1=>{.uv0=.code grp group;}.uv0-.y=.boxopen y do.
  gtype=.,>(code=2 3)#WORDTESTS
  ok ((":#y),' ',gtype,OK00403);group NB. okmsg: deleted from
elseif.do. uv0
end.
)

firstcomment=: 3 : 0

NB.*firstcomment v-- extracts the first comment sentence from J words.
NB.
NB. monad:  firstcomment clLinear
NB.
NB.   firstcomment 5!:5 <'firstcomment'
NB.   firstcomment disp 'jodword'
NB.
NB.   NB. first comments from many JOD non-nouns
NB.   n=. (}. grp 'JOD') -. 0 1 0 dnl''
NB.   t=. 1 pick 0 8 get n
NB.   n=. ({."1 t) #~  0= #&> {:"1 t
NB.   d=. 1 pick 0 10 get n
NB.   c=. n ,. firstcomment&.> 2{"1 d

NB. char table of just comment text
comtext=. 3 }."1 ljust onlycomments ];._2 (y-.CR),LF

NB. drop text below any monad and dyad marks
mask=. +./"1 ((,:MONADMARK) E. comtext) +. (,:DYADMARK) E. comtext
comtext=. ,' ' ,. comtext #~  -. +./\ mask

NB. take the first comment to end with a '.'
NB. excluding any J argument strings, eg. x. y.
NB. NIMP may not hold in j 6.01
comtext=. reb comtext {.~ firstperiod comtext
if. #comtext do.

  NB. trim scriptdoc style headers if any
  if. '*'={.,comtext do.
    alltrim '--' afterstr comtext
  end.

end.
)

firstperiod=: 3 : 0

NB.*firstperiod v-- returns the index of first sentence period.
NB.
NB. monad:  firstperiod cl

NB. first period in at most 500 chars
y=. (500<.#y){.y

NB. inflected names have been long deprecated in J
NB. there is no need to mask them in later code
NB. args=. ;&.> , { (<<"0' ([{'),<;:'m. n. x. y. u. v. *.'
NB. y=.' ' (I. _2  (|. !. 0) +./ args E.&> <y)} y

NB. first period
y i. '.'
)

NB. first document sentence
fsen=: ] ; [: firstcomment disp

getrx=: 3 : 0

NB.*getrx v-- get required to execute. (getrx) gets all the words
NB. required to execute words on (y).
NB.
NB. Warning:  if  the words listed on  (y)  refer to  object  or
NB. locale references  this verb  returns  an  error because such
NB. words generally cannot be run out of context.
NB.
NB. monad:  getrx clName | blclNames
NB.
NB.   NB. loads words into base locale
NB.   getrx 'stuffineed'
NB.   getrx ;:'stuff we words need to run'
NB.
NB. dyad:  clLocale getrx clName | blclNames
NB.
NB.   'targetlocale' getrx ;:'load the stuff we need into locale'

'base' getrx y
:
```

```
if. badrc uv0=. 31 uses y do. uv0
NB. errmsg: words refer to objects/locales
elseif. #uv1=. obnames uv0 do. (jderr ERR00405),uv1
elseif. uv0=.~.({."1 >{:uv0),refnames uv0
        badrc uv1=. x get uv0 do. uv1
elseif.do.
  ok '(',(":#uv0),OK00406,x
end.
)

hlpnl=: 3 : 0

NB.*hlpnl v-- displays short descriptions of objects on (y)
NB.
NB. monad:  hlpnl clName | blclNames
NB.
NB.   hlpnl refnames uses 'explainmycalls'
NB.
NB. dyad:  iaObject hlpnl clName|blclNames
NB.
NB.   2 hlpnl }.grp''

0 hlpnl y
:
if. empdnl y do. ok ''
elseif. 0=>{.uv0=. (x,EXPLAIN) get y do. uv0
elseif.do.
  uv0=.>{:uv0
  (>{."1 uv0) ; >{:"1 uv0
end.
)

jodage=: 3 : 0

NB.*jodage v-- days since last change and creation of JOD
NB. objects.
NB.
NB. monad:  jodage cl | blcl
NB.
NB.   jodage 'jodage'
NB.   jodage }. dnl 're'
NB.
NB. dyad:  iaCode jodage cl | blcl
NB.
NB.   2 jodage }. grp''

0 jodage y
:
if. badil x do. jderr ERR001
elseif. y=. ,boxopen y
        badrc changed=. (({.x),14) get y do. changed
elseif. badrc created=. (({.x),13) get y do. created
elseif.do.
  g=. /:daychanged=. <.,.1 dayage <.changed=. rv changed
  daycreated=. ,.<.1 dayage <.created=. rv created
  NB. header=. ;:'name changed created datechanged datecreated'
  header=. AGEHEADER
  NB. header ,: (<g) {&.> (>y);daychanged;daycreated;(,.changed);<,.created
  ok<header ,: (<g) {&.> (>y);(,.created);daycreated;(,.changed);<daychanged
end.
)

lg=: 3 : 0

NB.*lg v-- make and load JOD group.
NB.
NB. (lg) assembles and loads  JOD group  scripts. The monad loads
NB. without  the  postprocessor  and  the  dyad  loads  with  the
NB. postprocessor.
NB.
NB. monad:  lg clGroup
NB.
NB.   lg 'groupname'  NB. no postprocessor
NB.
NB. dyad:  uu lg clGroup
NB.
NB.   2 lg 'group'    NB. no postprocessor
NB.   lg~  'group'    NB. postprocessor

NB. (/:)=: obfuscate names
2 lg y
:
if. x-:2 do.
  NB. 2 _2 make assembles entire group script
  NB. with preamble regardless of where the
  NB. group appears on the JOD path
  msg=. OK00404 NB. okmsg: group loaded
  t=. 2 _2 make y
else.
  msg=. OK00405 NB. okmsg: group loaded with postprocessor
  t=. 2 mls y
end.
'r s'=. 2{.t
NB. j profile !(*)=. cocurrent
if. r do.
  curr=. 18!:5 ''   NB. current locale
  cocurrent 'base'  NB. run script from base
  try. 0!:0 s
  catchd.
    cocurrent curr  NB. restore locale
    NB. errmsg: J script error in group
    (jderr ERR00404),y;13!:12 ''
    return.
  end.
  cocurrent curr    NB. restore locale
  ok (y),msg
else.
  t
end.
)

locgrp=: 3 : 0
```

```
NB.*locgrp v-- list groups and suites with name.
NB.
NB. monad:  locgrp clName
NB.
NB.   locgrp 'dd'

NB. get group and suite names
gs=. 2 3 dnl&.> <''
if. *./ m=. ; {.&> gs do.
  gs=. }.&.> gs
  gnl=. 2 3 }.@:grp &.> &.> gs
  m=. gnl (+./@:e.)&>&.> <<<,y
  ok <GROUPSUITES ,. m#&.> gs
else.
  >{. (-.m) # &.> gs
end.
)

mls=: 3 : 0

NB.*mls v-- make load script.
NB.
NB. Generates a J (load) script from a JOD group and  an optional
NB. POST_ process macro script.
NB.
NB. monad:  mls clGroupName
NB.
NB.   NB. generate script and add to public scripts
NB.   mls 'JODaddon'
NB.
NB.   scripts 'e'       NB. JODaddon is now on scripts
NB.   load 'JODaddon'   NB. load's like any J load script
NB.
NB. dyad:  baPublic mls clGroupName
NB.
NB.   NB. make script but do not add to public scripts
NB.   0 mls 'JODaddon'
NB.
NB.   NB. make script and return text
NB.   2 mls 'JODaddon'

1 mls y
:

NB. HARDCODE: option qualifier codes
NB. 2 _2 make assembles entire group script
NB. with preamble regardless of where the
NB. group appears on the JOD path
v=. 2 _2 make gn=. y -. ' '
'r s'=. 2{.v
if. r do.
  NB. group make succeeded - append any POST_ script
  postpfx=. POSTAMBLEPFX
  if. badrc sp=. 4 dnl postpfx do. sp return. end.
  if. (<ps=. postpfx , gn) e. }.sp do.
    v=. 4 get ps
    'r p'=. 2{.v
    if. r do. s=. s , (2#LF) , (<0;2) {:: p else. v return. end.
  end.
  if. 2-:x do. ok s
  else.
    pdo=. {:0{DPATH__ST__JODobj   NB. put dictionary directory object
    rf=. gf=. SCR__pdo            NB. default directory

    NB. redirect script output if ROOTFOLDER exists and is configured - standard profile !(*)=. jpath
    if. wex <'ROOTFOLDER__pdo' do.
      NB. errmsg: ROOTFOLDER must be a character list configured (jpath) expression like: ~user/jodroot
      if. badcl ROOTFOLDER__pdo do. jderr ERR00407 return. end.
      if. 0 < #rf=. alltrim ROOTFOLDER__pdo do.
        if. '~' ~: {. rf do. jderr ERR00407 return. end.
        NB. do not expand relative path strings - relative paths must be configured
        if. rf -: gt=. jpath rf do. jderr ERR00407 return. else. gf=. tslash2 gt end.
        rf=. tslash2 rf
      else.
        rf=. gf
      end.
    end.

    lsn=. gf,gn,IJS__JODobj  NB. errmsg: unable to write load script
    if. _1 -: (toHOST s) (write :: _1:) lsn do. (jderr ERR00408),<lsn return. end.
    NB. update scripts.ijs
```

```
    x addloadscript gn;rf,gn

  end.
else.
  v
end.
)

noexp=: 3 : 0

NB.*noexp v-- returns a list of objects with no explanations.
NB.
NB. monad:  noexp zl|clPattern
NB.
NB.   noexp ''  NB. words without short explanations
NB.
NB. dyad:  iaCode noexp zl | clPattern
NB.
NB.   2 noexp 'jod'       NB. groups without explanations
NB.   (i.5) noexp"0 1 ''  NB. all objects without explanations

0 noexp y
:
if. badrc uv=.x dnl y do. uv
elseif. a: e. uv       do. ok ''
elseif. badrc uv=. (({.x),EXPLAIN) get }.uv do. uv
elseif. 0=#uv=. rv uv  do. ok''
elseif.do.
```

```
 ok (0 = #&> {:"1 uv) # {."1 uv
end.
)

notgrp=: 3 : 0

NB.*notgrp v-- words or tests from (y) that are not in groups or
NB. suites. Useful for finding loose ends and dead code.
NB.
NB. monad:  notgrp blcl
NB.
NB.   notgrp }. revo ''     NB. recent ungrouped words
NB.
NB. dyad:  iaObject notgrp blcl
NB.
NB.   2 notgrp }. dnl ''    NB. ungrouped words
NB.   3 notgrp }. 1 dnl ''  NB. tests that are not in suites

GROUP notgrp y
:
if. badrc y=. checknames y do. y return. end.
y=. }. y
select. x
  case. GROUP do. ok y -. ; grp&.> }. GROUP dnl ''
  case. SUITE do. ok y -. ; SUITE grp&.> }. SUITE dnl ''
  case.do. jderr ERR001
end.
)
```

```
nt=: 3 : 0

NB.*nt v-- edit a new test script using JOD conventions.
NB.
NB. This  verb  looks   for  (TESTSTUB)  on  the  path   of  open
NB. dictionaries allowing easy user defined test script formats.
NB.
NB. monad:  nt clName
NB.
NB.   nt 'scriptname'
NB.
NB. dyad:  clSreps nt clName
NB.
NB.   NB. the dyad allows more general string
NB.   NB. replacements to be applied to stubs
NB.
NB.   '#{{boo}}#<<newboo>>#{{hoo}}#??newhoo??' nt 'newscript'

'' nt y
:
if. badcl y do. jderr ERR002 return. end. NB. errmsg: invalid name(s)
if. badcl x do. jderr ERR001 return. end. NB. errmsg: invalid option(s)
name=. y -. ' ' [ dl=. {. x,'/'

NB. HARDCODE: invalid delimiters
if. dl e. '{}~ADST' do. jderr ERR00406 return. end. NB. errmsg: invalid delimiter
```

```
NB. get teststub document from open dictionaries
'r s'=.2{.t=. 1 get TESTSTUB
if. r do.
  'datess timess'=.yyyymondd 0
  shortdate=. 2 }. datess
  test=. dl,'{~T~}',dl,name,dl,'{~D~}',dl,datess,dl,'{~SD~}',dl,shortdate
  NB. insert any visible cl !(*)=. CLASSAUTHOR
  NB. NOTE: nouns in locale (ijod) are visible here
  if. wex <'CLASSAUTHOR' do.
    NB. (CLASSAUTHOR) is a cl without (dl)
    if. (-.badcl CLASSAUTHOR) *. -.dl e. CLASSAUTHOR do. test=. test,dl,'{~A~}',dl,CLASSAUTHOR end.
  end.
  name et (test,x) changestr >1{,s
else.
  t
end.
)

nw=: 3 : 0

NB.*nw v-- edit a new explicit word using JOD conventions.
NB.
NB. monad:  nw clWord
NB.
NB.   nw 'verb'
NB.
NB. dyad:  iaClass nw clWord
NB.
NB.   1 nw 'adverb'

3 nw y
:
name=. y -. ' '
if. -.x e. i. 5 do. x=.3 end.
class=. x{'nacvv'

NB. user defined post proc !(*)=. DOCUMENTCOMMAND
if. 0= (4!:0) <'DOCUMENTCOMMAND' do.
  word=.DOCUMENTMARK,LF,LF,DOCUMENTCOMMAND
else.
  word=.DOCUMENTMARK
end.

reps=. '/{~N~}/',(y-.' '),'/{~C~}/',(":x),'/{~T~}/',class
word=. reps changestr word
name et word
)

NB. object/locale names from uses:  allnames 31 uses 'name'
obnames=: [: /:~ [: ~. [: ; 2: { "1 [: > {:

onlycomments=: 3 : 0

NB.*onlycomments v-- removes all J code leaving comments.
NB.
NB. monad:  ct =. onlycomments ctJcode
NB.
NB.   onlycomments jcr 'onlycomments'  NB. self comments

NB. mask of unquoted comment starts
c =. ($y)$'NB.' E. ,y
c =. -. +./\"1 c > ~:/\"1 y e. ''''
y =. ,y

NB. blank out code
y =. ' ' ((,c)# i. # y)} y
y =. y $~ $c
y #~ y +./ . ~: ' '  NB. remove blank rows
)

NB. put and cross reference word
pr=: 0&globs ,:~ put

NB. referenced nonlocale names from uses:  allnames 31 uses 'name'
refnames=: [: /:~ [: ~. [: ; 1: { "1 [: > {:

revonex=: 3 : 0

NB.*revonex v-- returns a  list of put dictionary objects with no
NB. explanations.
NB.
NB. This verb is  similiar to (noexp) except it only searches put
NB. dictionary objects and (noexp) searches the entire path.
NB.
NB. monad:  revonex zl | clPattern
NB.
NB.   revonex ''  NB. put dictionary words without short explanations
NB.
NB. dyad:  iaCode revonex zl|clPattern
NB.
NB.   2 revonex 'jod'       NB. put dictionary groups without explanations
NB.   (i.5) revonex"0 1 ''  NB. all put dictionary objects without explanations

/:~ 0 revonex y
:
if. badrc uv=./:~ x revo y do. uv
elseif. a: e. uv  do. ok ''
elseif. badrc uv=. (({.x),EXPLAIN) get }.uv do. uv
elseif. 0=#uv=. rv uv  do. ok ''
elseif.do.
 ok (0 = #&> {:"1 uv) # {."1 uv
end.
)

NB. extract single line explanation from word header comment and save
swex=: 0 8&put@:fsen

today=: 3 : 0
```

```
NB.*today v-- returns todays date.
NB.
NB. monad:  ilYYYYMMDD =. today uu
NB.
NB.   today 0    NB. ignores argument
NB.
NB. dyad:  iaYYYYMMDD =. uu today uu
NB.
NB.   0 today 0

3&{.@(6!:0) ''
:
0 100 100 #. <. 3&{.@(6!:0) ''
)

todayno=: 3 : 0

NB.*todayno v-- convert dates to day numbers, converse  (todate).
NB.
NB. WARNING: valid only for  Gregorian dates after  and including
NB. 1800 1 1.
NB.
NB. monad:  todayno ilYYYYMMDD
NB.
NB.   dates=. 19530702 19520820 20000101 20000229
NB.   todayno 0 100 100 #: dates
NB.
NB. dyad:  pa todayno itYYYYMMDD
NB.
NB.   1 todayno dates

0 todayno y
:
a=. y
if. x do. a=. 0 100 100 #: a end.
a=. ((*/r=. }: $a) , {:$a) $,a
'y m d'=. <"_1 |: a
y=. 0 100 #: y - m <: 2
n=. +/ |: <. 36524.25 365.25 *"1 y
n=. n + <. 0.41 + 0 30.6 #. (12 | m-3),"0 d
0 >. r $ n - 657378
)

updatepublic=: 4 : 0

NB.*updatepublic v-- updates public scripts table.
NB.
NB. dyad:  btcl =. btclPublic updatepublic blNamePath
NB.
NB.   Public_j_ updatepublic 'name';'c:/where/the/script/things/are.ijs'

p=. (0 {"1 x) i. 0{y
if. p<#x do.
  NB. update entry
  x=. y p} x
else.
```

```
  NB. new entry - sort public scripts
  x=. x , y
  x=. (/:0 {"1 x){x
end.
)

usedby=: 4 : 0

NB.*usedby v-- returns a list of words from (y) that DIRECTLY
NB. call words on (x). The result of this verb depends on JOD
NB. dictionary references being up-to-date.
NB.
NB. dyad:  cl/blcl usedby blcl
NB.
NB.   'wordname' usedby }. dnl ''
NB.   ('word';'names') usedby }. revo ''
NB.
NB.   'putgs__ST' usedby }. dnl ''

NB. (uses) is expensive for large word lists.
if. badrc uv=.uses y do. uv
else.
  uv=. >{: uv
  wnames=. boxopen x

  NB. BUGFIX: 21sep10 - was not returning names like: EMCS_END_CHECK_sql
  NB. search object and locale references if _ occurs in any name
  NB. col=. >: +./ '_'&e.&> wnames
  NB. ok /:~ ({."1 uv) #~  ; (col {"1 uv) +./@e.&.> < wnames

  ok /:~ ({."1 uv) #~ +./"1 ;"1 (1 2 {"1 uv) +./@e.&.> <wnames
end.
)


yyyymondd=: 3 : 0

NB.*yyyymondd v-- today in (yyyymondd;hrmnss) format.
NB.
NB. Yet another date format verb. We can never have enough!
NB.
NB. monad:  (clDate ; clTime) =. yyyymondd uuIgnore

fmt=.'r<0>2.0'
months=. _3 [\ '   janfebmaraprmayjunjulaugsepoctnovdec'
'yy mn dd'=. 3{.now=. 6!:0''
date=. (":yy),(mn{months),,fmt (8!:2) dd
time=. }.;':' ,&.> fmt (8!:0) _3 {. now
date;time
)


NB.*jodtools s-- jodtools postprocessor.

NB. retain whitespace
(9!:41) 1
```

```
NB. insure base
cocurrent 'base'

NB. create/initialize a JOD tools object
JODtools_ijod_=:  conew 'ajodtools'                NB. new tools object
(1!:2&2) createjodtools__JODtools JODtools,JODobj  NB. pass self and JODs
```

Index