

gpxutils Group

John D. Baker

<https://github.com/bakerjd99/jackshacks/blob/main/gpxutils.ijs>

SHA-256: 92e98deb24402a8de30e508158277c02a0a4210167da4125f02c2b54e21e3a0c

July 19, 2025

Contents

gpxutils Overview	2
gpxutils Interface	2
Installing gpxutils	3
Using gpxutils	3
gpxutils Source Code	4
=: Index	34

gpxutils Overview

gpxutils is a [J script](#) that formats Garmin style waypoint [GPX files](#) from CSV files, Google Maps KML files, and the SmugMug [SQLite](#) mirror.db database. The resulting GPX files can be loaded into the Motion-GPS iPhone app and other GPS devices that import GPX data.

gpxutils is generated from [JOD dictionaries](#) `gps` and `utils`.

NB. open JOD dictionaries and generate gpxutils script

```
load 'general/jod'  
od  '::'gps utils'  
mls 'gpxutils'
```

A generated gpxutils script and sample `small_mirror.db` database are available in the GitHub `jackshacks` repository here:

- <https://github.com/bakerjd99/jackshacks>
- <https://github.com/bakerjd99/jackshacks/tree/main/testdata>

gpxutils Interface

<code>alaska_yukon_wpt</code>	<code>[8]</code>	<i>update alaska yukon waypoints</i>
<code>allrecent</code>	<code>[9]</code>	<i>all recent images from last waypoint generation</i>
<code>csvfrgm</code>	<code>[13]</code>	<i>poi CSV text from google maps lat, lon, desc layout</i>
<code>csvfirtab</code>	<code>[13]</code>	<i>poi CSV text from TAB delimited text file</i>
<code>csvfrwpt</code>	<code>[15]</code>	<i>poi CSV text from waypoint text file</i>
<code>gpskm</code>	<code>[21]</code>	<i>distances in km from Google Maps coordinates</i>
<code>gpxfrmapkml</code>	<code>[23]</code>	<i>gpx from Google maps kml</i>

<code>gpxfirmirror</code>	[24]	<i>extracts geotagged images from mirror_db and generates gpx</i>
<code>gpxfrpoicsv</code>	[25]	<i>converts poi csv files to gpx</i>
<code>gpxfrrecent</code>	[27]	<i>gpx from recent waypoints</i>

Installing gpxutils

If you have a current vesion of J (9.0x+ or later) installed `gpxutils` can be downloaded as a J [addon](#) script by typing the following commands into a JQt or JHS session.

NB. install addon files in ~addons/jacks
`install 'github:bakerjd99/jackshacks'`

NB. installed files
`dir '~addons/jacks'`

NB. load script
`load '~addons/jacks/gpxutils.ijs'`

To get the lastest vesions of `gpxutils` and other addon scripts in `addons/jacks` simply reinstall.

Using gpxutils

To run `gpxutils` inspect interface word comments.

gpxutils Source Code

```
NB.*gpxutils s-- generate gpx waypoint files from various
NB. sources.
NB.
NB. This group formats Garmin style waypoint gpx files from CSV
NB. files, my SmugMug sqlite mirror database, and Google map KML.
NB. The resulting gpx files can be loaded into the Motion-GPS
NB. iPhone app and other GPS devices that import gpx data.
NB.
NB. verbatim: interface words
NB.
NB. alaska_yukon_wpt - update alaska yukon waypoints
NB. allrecent       - all recent images from last waypoint generation
NB. csvfrgm         - poi CSV text from google maps lat, lon, desc layout
NB. csvfrtab        - poi CSV text from TAB delimited text file
NB. csvfrwpt        - poi CSV text from waypoint text file
NB. gpskm           - distances in km from Google Maps coordinates
NB. gpxfrmapkml     - gpx from Google maps kml
NB. gpxfrmirror     - extracts geotagged images from mirror_db and generates gpx
NB. gpxfrpoicsv     - converts poi csv files to gpx
NB. gpxfrrecent     - gpx from recent waypoints
NB.
NB. created: 2019dec11
NB. changes: -----
NB. 19dec18 added (allrecent)
NB. 22jun18 merged (gpxfrmapkml) and dependents
```

NB. 24aug17 added (csvfrtab, gpskm)
NB. 25jul18 (csvfrgm) added
NB. 25jul19 (alaska_yukon_wpt) added

```
require 'data/sqlite regex'  
coclass 'gpxutils'
```

*NB.*dependents*
NB. ()=: GPXFRKMLHEADER GPXHEADER GPXSMUGPLACEMARK GPXTRAILER*
*NB.*enddependents*

```
GPXFRKMLHEADER=: (0 : 0)  
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>  
<gpx version="1.1"  
  creator="J GPX from Google Maps KML script"  
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"  
  xmlns="https://www.topografix.com/GPX/1/1"  
  xsi:schemaLocation="https://www.topografix.com/GPX/1/1/gpx.xsd">  
<metadata>  
<name>{{headername}}</name>  
<desc>{{headerdescription}}</desc>  
<link href="https://analyzethedatanotthedrive1.org/">  
<text>Analyze the Data not the Drive1</text>  
</link>  
</metadata>  
)
```

```
GPXHEADER=: (0 : 0)
<gpx xmlns="https://www.topografix.com/GPX/1/1"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  creator="J Waypoints"
  version="1.1"
  xsi:schemaLocation="https://www.topografix.com/GPX/1/1/gpx.xsd">
<metadata>
<link href="https://www.jsoftware.com">
<text>J (gpxutils) last waypoint = {{date}}</text>
</link>

</metadata>
)

GPXSMUGPLACEMARK=: (0 : 0)
<wpt lat="{{latitude}}" lon="{{longitude}}">
<ele>0</ele>
<name>{{phototitle}}</name>
</wpt>
)

GPXTRAILER=: (0 : 0)
<extensions>
</extensions>
</gpx>
)
NB.*end-header
```

NB. get all images from mirror - select columns

```
AllMirror_sql=: 'select Latitude, Longitude, RealDate, UploadDate, OnlineImageFile from OnlineImage'
```

NB. carriage return character

```
CR=: 13{a.
```

NB. valid gpx name characters

```
GPXNAMECHARS=: ' -()0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

NB. get geotagged images from mirror - rows in desc upload date

```
GpxGeotaggedMirror_sql=: 'select Latitude, Longitude, RealDate, UploadDate, OnlineImageFile from OnlineImage'  
>...>e where Keywords like "%geotagged%"'
```

NB. regular expression matching placeholder variables in html lists

```
HTMLVARBPATTERN=: '{{[a-z]*}}'
```

NB. interface words (IFACEWORDSgpxutils) group

```
IFACEWORDSgpxutils=: <;._1 ' alaska_yukon_wpt allrecent csvfrgm csvfrrtab csvfrwpt gpskm gpxfrmapkml gpxfrmi'  
>...>error gpxfrpoicsv gpxfrrecent'
```

NB. line feed character

```
LF=: 10{a.
```

NB. gpx file written by (gpxutils)

```
MIRRORGPXFILE=: 'c:/pd/coords/gpx/geotagged smugmug images.gpx'
```

NB. home base longitude latitude using Meeus conventions

```
MeeusHomeLonLat=: 0 0
```

NB. root words (ROOTWORDSgpxutils) group

```
ROOTWORDSgpxutils=: <|. _1 ' IFACEWORDSgpxutils ROOTWORDSgpxutils VMDgpxutils alaska_yukon_wpt allrecent csv  
>..>firtab csvfirwpt gpskm gpxfirmapkml gpxfirmirror gpxfirrecent'
```

NB. version, make count, and date

```
VMDgpxutils=: '0.9.2';2;'19 Jul 2025 18:01:57'
```

NB. retains string (y) after last occurrence of (x)

```
afterlaststr=: ] }.~ #@[ + 1&(i.~)@([ E. ])
```

NB. retains string after first occurrence of (x)

```
afterstr=: ] }.~ #@[ + 1&(i.~)@([ E. ])
```

```
alaska_yukon_wpt=: 3 : 0
```

*NB.*alaska_yukon_wpt v-- update alaska yukon waypoints.*

NB.

NB. monad: alaska_yukon_wpt uuIgnore

NB. !()=. jpath smoutput*

```
csv=. csvfirm read txt=. jpath '~TRAVEL/alaska_yukon/yukon_alaska_campgrounds.txt'  
wcnt=. (":#csv),' Alaska/Yukon waypoints'
```



```
(',' fmtcd csv) write camps=. jpath '~TRAVEL/alaska_yukon/yukon_alaska_campgrounds.csv'  
(gpxfirpoicsv camps) write gpx=. jpath '~TRAVEL/alaska_yukon/yukon_alaska_campgrounds.gpx'  
.. wcnt;txt;camps;gpx  
)
```

```
allrecent=: 3 : 0
```

```
NB.*allrecent v-- all recent images from last waypoint generation.
```

```
NB.
```

```
NB. monad: bt =. allrecent clMirrorDb
```

```
NB.
```

```
NB.   trg=. jpath '~addons/jacks/testdata/small_mirror.db'
```

```
NB.   allrecent trg
```

```
NB.
```

```
NB. dyad: bt =. clGpxFile allrecent clMirrorDb
```

```
NB.
```

```
NB.   lastgpx=. 'c:/pd/coords/gpx/geotagged test images.gpx'
```

```
NB.   lastgpx allrecent trg
```

```
MIRRORGPXFILE allrecent y
```

```
:
```

```
waydate=. waystamp gpx=. read x NB. extract last waypoint date
```

```
NB. the last upload date is shifted forward to partly compensate
```

```
NB. for the mixture of UTC and local dates. The times in the database
```

```
NB. come from many time zones and many timestamps are just approximations.
```

```
sql=. AllMirror_sql , ' where UploadDate > date("",waydate,"", ''+16 hours'') order by UploadDate desc '
```

```
sql fst y
)
```

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

```
NB. trims all leading and trailing white space
allwhitetrim=: ] #~ [: -. [: (*./\ . +. */.\) ] e. (9 10 13 32{a.})"_
```

```
NB. arc tangent
arctan=: _3&o.
```

```
NB. signal with optional message
assert=: 0 0"_ $ 13!:8^:((0: e. ])^(12"_))
```

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

```
betweenstrs=: 4 : 0
```

```
NB.*betweenstrs v-- select sublists between nonnested delimiters
```

```
NB. discarding delimiters.
```

```
NB.
```

```
NB. dyad:  blcl =. (clStart;clEnd) betweenstrs cl
```

```
NB.      blnl =. (nlStart;nlEnd) betweenstrs nl
```

```
NB.
```

```
NB. ('start';'end') betweenstrs 'start yada yada end boo hoo start ahh end'
NB.
NB. NB. also applies to numeric delimiters
NB. (1 1;2 2) betweenstrs 1 1 66 666 2 2 7 87 1 1 0 2 2
```

```
's e'=. x
llst=. ((-#s) (|.!.0) s E. y) +. e E. y
mask=. ~:/\ llst
(mask#llst) <|.1 mask#y
)
```

```
NB. boxes open nouns
boxopen=: <^(L. = 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 - 1 NB. change starts
  s=. * d                      NB. reduce < and > to =
  if. s = _1 do.
    b=. 1 #~ # b
    b=. ((1 * # 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
)

```

```

charsub=: 4 : 0

```

```

NB.*charsub v-- single character pair replacements.

```

```

NB.

```

```

NB. dyad: clPairs charsub cu

```

```

NB.

```

```

NB.   '$_$ ' charsub '$123 -456 -789'

```

```

'f t'=. ((#x)$0 1)<@,&a./x

```

```
t {~ f i. y
)

NB. cosine radians
cos=: 2&o.

csvfrgm=: 3 : 0

NB.*csvfrgm v-- poi CSV text from google maps lat, lon, desc layout.
NB.
NB. monad: cl =. csvfrgm clTxt
NB.
NB.    csv=. read jpath '~TRAVEL/alaska_yukon/yukon_alaska_campgrounds.txt'
NB.    (',' fmtcd csvfrgm csv) write camps=. jpath '~TRAVEL/alaska_yukon/yukon_alaska_campgrounds.csv'
NB.    (gpæfrpoicsv camps) write jpath '~TRAVEL/alaska_yukon/yukon_alaska_campgrounds.gpx'

NB. text in latitude, longitude, description
csv=. rebrow ];._2 tlf y -. CR

NB. exactly two commas per line
'comma delimiters invalid' assert 2 = +/"1 ',' = csv

NB. flip to longitude, latitude, description
1 0 2 {"1 allwhitetrims> ',' parsecsv tlf ctl csv
)

csvfirtab=: 3 : 0
```

```
NB.*csvfrtab v-- poi CSV text from TAB delimited text file.
NB.
NB. monad: cl =. csvfrtab clFile
NB.
NB. f=. jpath '~temp/chile_antarctica_2026.txt'
NB. p=. jpath '~temp/chile_antarctica_2026'
NB. t=. csvfrtab f
NB. (toHOST t) write p, '.csv'
NB. g=. gpxfprpoicsv p, '.csv'
NB. (toHOST g) write p, '.gpx'

NB. parse TAB delimited text
ct=. readtd2 y

NB. required columns
'column(s) missing' assert (;:'Location Latitude Longitude') e. 0{ct

Longitude=. ,&','&.> }. ct {"1~ (0{ct) i. <'Longitude'
Latitude=. ,&','&.> }. ct {"1~ (0{ct) i. <'Latitude'
Location=. }. ct {"1~ (0{ct) i. <'Location'

NB. replace any commas in names with blanks
Location=. rebc@(' , '&charsub)&.> Location

NB. form poi CSV
ctl ;"1 Longitude ,. Latitude ,. Location
)
```

```
csvfrwpt=: 3 : 0
```

```
NB.*csvfrwpt v-- poi CSV text from waypoint text file.
```

```
NB.
```

```
NB. monad: cl =. csvfrwpt clFile
```

```
NB.
```

```
NB. f=. jpath '~JACKSHACKS/testdata/gps_oz_nz_2022.txt'
```

```
NB. p=. jpath '~temp'
```

```
NB. t=. csvfrwpt f
```

```
NB. (toHOST t) write p, '.csv'
```

```
NB. g=. gpæfrpoicsv p, '.csv'
```

```
NB. (toHOST g) write p, '.gpx'
```

```
NB. lines from text
```

```
ct=. <;._2 tlf (read y) -. CR
```

```
NB. waypoint names
```

```
wn=. ':'&beforestr&.> ct
```

```
NB. extract longitude and latitude
```

```
lb=. |."1 <;._1"1 ' ,' ,&> -.&' '&.> (':'&afterstr)@(';'&beforestr)&.> ct
```

```
NB. format comma delimited
```

```
em=. 1 0 1 0 1
```

```
lb=. alltrim&.> lb ,. wn
```

```
tlf ctl ;"1 (<' ,' ) (<a:;I. -.em)} em (#^:_1)"1 lb
```

```
)
```

NB. character table to newline delimited list

```
ctl=: }.@(@1&(",1)@(-.@(*./\"1@(&' ' @])))) # ,@((10{a.)&(",1)@]))
```

NB. enclose all character lists in blcl in " quotes

```
dblquote=: '""&,@:(,&'')&.>
```

```
earthdist=: 4 : 0
```

*NB.*earthdist v-- distance in km between n points on the Earth's surface.*

NB.

NB. dyad: (fl | ft) earthdist (fl | ft)

NB.

NB. NB. Paris longitude, latitude

NB. NB. ddfrdms computes decimal degrees from degree, minutes, seconds

NB. l1 =. ddfrdms _2 _20 _14 NB. 2d 20m 14s (East)

NB. theta1 =. ddfrdms 48 50 11 NB. 48d 40m 11s (North)

NB.

NB. NB. Washington

NB. l2 =. ddfrdms 77 3 56 NB. 77d 3m 56s (West)

NB. theta2 =. ddfrdms 38 55 17 NB. 38d 55m 17s (North)

NB.

NB. NB. rounded to 2 decimals matches Meeus

NB. 6181.63 = ". '0.2' 8!:2 (l1,theta1) earthdist l2,theta2

NB.

NB. NB. table arguments

NB. (/: 5 # ,: l1,theta1) earthdist /: 5 # ,: l2,theta2


```
a=. 6378.14      NB. Earth's mean radius (km)
fl=. % 298.257   NB. Earth's flattening (a * 1 - fl) is polar radius

NB. zero distances mask
b=. */ x = y

NB. longitudes and latitudes in decimal degrees
NB. western longitudes +, northern latitudes +
NB. (*)=. l1 l2 theta1 theta2
'l1 theta1'=. x [ 'l2 theta2'=. y

f=.      rfd -: theta1 + theta2
g=.      rfd -: theta1 - theta2
lambda=. rfd -: l1 - l2

sqrsin=. *: @ sin
sqrcos=. *: @ cos

sinlam=. sqrsin lambda [ coslam=. sqrcos lambda
sqrcosg=. sqrcos g [ sqrsing=. sqrsin g
sqrsinf=. sqrsin f [ sqrcosf=. sqrcos f

s=. (coslam * sqrsing) + sinlam * sqrcosf
c=. (coslam * sqrcosg) + sinlam * sqrsinf

omega=. arctan %: s % c
r3=. 3 * (%: s * c) % omega
```

```
d=. +: omega * a
h1=. (<: r3) % +: c
h2=. (>: r3) % +: s

NB. required distance
d=. d * (>: f1*h1*sqrnsinf*sqrncosg) - f1*h2*sqrncosf*sqrnsing

NB. handle any zero distances
if. +./ b do.
  NB. cannot do b*d as d is undefined _ for zero distances
  if. #$ d do. 0 (I. b)} d elseif. b do. 0 elseif. 1 do. d end.
else.
  d
end.
)

eletags=: 4 : 0

NB.*eletags v-- encloses xml text (y) in xml element tag.
NB.
NB. dyad: clTag eletags clXml

tag=. alltrim x
'<',tag,'>',y,'</',tag,'>'
)

NB. format tables as (x) character delimited LF terminated text - see long document
fmtcd=: 4 : 'tlf ctl }."1 ;"1 (x&,)@":&.> y'
```

```
fmtmirrorgpx=: 3 : 0
```

```
NB.*fmtmirrorgpx v-- formats mirror_db sql query results as gpx.
```

```
NB.
```

```
NB. monad: fmtmirrorgpx btSqlDict
```

```
NB. insure any singletons are shaped
```

```
ix=. I. (0 {"1 y) e. ;:'RealDate UploadDate OnlineImageFile'
```

```
y=. (boxopen&.> (<ix;1){y) (<ix;1)} y
```

```
y=. (,&.> 1 {"1 y) (<a;1)} y
```

```
NB. quit if no data
```

```
if. +./ 0 = #&> 1 {"1 y do. '' return. end.
```

```
NB. !(*)=. Latitude Longitude RealDate UploadDate OnlineImageFile
```

```
(0 {"1 y)=. 1 {"1 y
```

```
NB. clean file names
```

```
names=. '['&beforestr@justfile&.> OnlineImageFile
```

```
names=. alltrim&.> names -.&.> names -.&.> <GPXNAMECHARS
```

```
'names cannot be null' assert -. 0 e. #&> names
```

```
NB. format latitude and longitude
```

```
wpt=. (<LF,'<wpt lat=') ,. (dblquote 8!:0 Latitude) ,. (<' lon=') ,. (dblquote 8!:0 Longitude) ,. <'>'
```

```
NB. format dates for gpx
```

```
RealDate=. alltrim@((,&'Z')@('+'&beforestr))&.> RealDate
```

```
UploadDate=. alltrim@((,&'Z')@('+'&beforestr))&.> UploadDate

NB. use real date unless empty else use upload date
ix=. I. 0 = #&> RealDate
RealDate=. (ix{UploadDate) ix} RealDate
wpt=. wpt ,. 'time'&eletags&.> RealDate

NB. waypoint names & descriptions
wpt=. wpt ,. _1 |."1 names ,"0 1 |. tags 'name'

NB. symbols
wpt=. wpt ,. <'sym' eletags 'waypoint'
wpt=. wpt ,. <'</wpt>'

NB. last waypoint upload date
gpxhead=. ('/{date}]/', }: ;0{UploadDate) changestr GPXHEADER

NB. return gpx
gpxhead,(;wpt),LF,'</gpx>'
)

fsd=: 4 : 0

NB.*fsd v-- fetch sqlite dictionary array.
NB.
NB. dyad: bt =. clSql fsd clDb
NB.
NB. trg=. 'c:/smugmirror/documents/xrefdb/mirror.db'
```

```
NB. sql=. 'select ImageKey, OriginalWidth, OriginalHeight, OnlineImageFile, Keywords from OnlineImage'
NB. sql fsd trg
```

```
NB. require 'data/sqlite' !(*)=. sqlclose__db sqldict__db sqlopen_psqlite_
d [ sqlclose__db '' [ d=. sqldict__db x [ db=. sqlopen_psqlite_ y
)
```

```
fst=: 4 : 0
```

```
NB.*fst v-- fetch sqlite reads table.
```

```
NB.
```

```
NB. dyad: bt =. clSql fst clDb
```

```
NB.
```

```
NB. trg=. 'c:/smugmirror/documents/xrefdb/mirror.db'
```

```
NB. sql=. 'select ImageKey, OriginalWidth, OriginalHeight, OnlineImageFile, Keywords from OnlineImage'
```

```
NB. sql fst trg
```

```
NB. require 'data/sqlite' !(*)=. sqlclose__db sqlreads__db sqlopen_psqlite_
d [ sqlclose__db '' [ d=. sqlreads__db x [ db=. sqlopen_psqlite_ y
)
```

```
NB. get pure element text
```

```
geteletext=: ] betweenstrs~ [: tags [: alltrim [
```

```
gpskm=: 3 : 0
```

```
NB.*gpskm v-- distances in km from Google Maps coordinates.
```

```
NB.
NB. monad:  bt =. gpskm clFile
NB.
NB.  dt=. gpskm jpath '~temp/chile_antarctica_2026.txt'
NB.
NB.  NB. sorted by increasing distance
NB.  (] { ~ [: /: {"1) dt
NB.
NB. dyad:  bt =. flMeeusLonLat gpskm clFile
NB.
NB.  NB. distance from Meeus location Longitude +W, Latitude +N
NB.  0 0 gpskm jpath '~temp/chile_antarctica_2026.txt'

NB. home location - Meeus conventions
MeeusHomeLonLat gpskm y
:
NB. read TAB delimited locations
ct=. readtd2 y

NB. required columns
'column(s) missing' assert (;:'Location Latitude Longitude') e. 0{ct

Longitude=. ".&> }. ct {"1~ (0{ct) i. <'Longitude'
Latitude=. ".&> }. ct {"1~ (0{ct) i. <'Latitude'
Location=. }. ct {"1~ (0{ct) i. <'Location'

Location ,. <"0 x earthdist |: (-Longitude) ,. Latitude
```

```
)

gpxfmapkml=: 3 : 0

NB.*gpxfmapkml v-- gpx from Google maps kml.
NB.
NB. monad: clGpx =. gpxfmapkml clKml
NB.
NB. NB. download Google map waypoints as kml
NB. kml=. read jpath '~addons/jacks/testdata/small_mirror.kml'
NB.
NB. NB. convert to gpx and save
NB. gpx=. gpxfmapkml kml
NB. (toHOST gpx) write jpath '~temp/small_kml.gpx'

NB. parse kml form waypoint table
dname=. ;'name' getelete text '<Placemark>' beforestr y
wpt=. ;'Placemark' getelete text y
wpt=. ('name' getelete text wpt) ,. <:_1&> ','&,&.> 'coordinates' getelete text wpt
hdr=. ;:'phototitle longitude latitude'

NB. format gpx header
gpxstamp=. 'Waypoints: ',( "#wpt), ' GPX generated: ',timestamp''
gpxheader=. ('/{headername}}/',dname,'/{headerdescription}}/',gpxstamp) changestr GPXFRKMLHEADER
gpxtrailer=. GPXTRAILER

'idx pkml'=. HTMLVARBPATTERN patpartstr GPXSMUGPLACEMARK
rvarbs=. idx htmlvarbs pkml
```

```
msg=. 'all row variables must exist in data header'
msg assert *./ rvarbs e. hdr
rows=. (#wpt) # ,: pkml
rows=. ((hdr i. <'phototitle'){1 wpt) (<a:;(rvarbs i. <'phototitle'){idx}) rows
rows=. ((hdr i. <'latitude'){1 wpt) (<a:;(rvarbs i. <'latitude'){idx}) rows
rows=. ((hdr i. <'longitude'){1 wpt) (<a:;(rvarbs i. <'longitude'){idx}) rows

gpxheader,(;rows),gpxtrailer
)

gpxfrmirror=: 3 : 0

NB.*gpxfrmirror v-- extracts geotagged images from mirror_db and generates gpx.
NB.
NB. monad: clGpx =. gpxfrmirror clMirrorDb
NB.
NB.   trg=. jpath '~addons/jacks/testdata/small_mirror.db'
NB.   gpx=. gpxfrmirror trg
NB.   (toHOST gpx) write jpath '~temp/geotagged_images.gpx'
NB.
NB. dyad: clGpx =. iaN gpxfrmirror clMirrorDb
NB.
NB.   10 gpxfrmirror trg

0 gpxfrmirror y NB. all waypoints default
:
NB. limit waypoints
```



```
sql=. GpxGeotaggedMirror_sql , ' order by UploadDate desc ' , ;(0<x){'';' limit ',":x
fmtmirrorgpx sql fsd y
)
```

```
gpxfrpoicsv=: 3 : 0
```

*NB.*gpxfrpoicsv v-- converts poi csv files to gpx.*

NB.

NB. This verb converts comma delimited point of interest (POI)

*NB. *.csv files to Garmin compatible gpx files. Example POI files*

NB. can be downloaded from:

NB.

NB. <http://www.poi-factory.com/poifiles>

NB.

NB. monad: clGpx =. gpxfrpoicsv clCsvfile

NB.

NB. csv=. jpath '~addons/jacks/testdata/10_best_us_star_gazing.csv'

NB. gpx=. gpxfrpoicsv csv

NB. (toHOST gpx) write jpath '~temp/star_gazing.gpx'

NB.

NB. dyad: clGpx =. iaRows gpxfrpoicsv clCsvfile

NB.

NB. gpx=. 10 gpxfrpoicsv 'c:\pd\coords\poicsv\ca_park_m.csv'

0 gpxfrpoicsv y NB. format all waypoints default

:

NB. read csv file

csv=. parsecsv tlf read y

```
if. 0<x do. csv=. (x<.#csv) {. csv end.
```

NB. sanity test latitude and longitude

```
lbcheck=. -. _9999 e. , _9999 "&> 0 1 {"1 csv  
'invalid longitude latitude number representations' assert lbcheck
```

NB. clean names

```
names=. 2 {"1 csv  
names=. alltrim&.> names -.&.> names -.&.> <GPXNAMECHARS  
'names cannot be null' assert -. 0 e. #&> names
```

NB. format latitude and longitude

```
csv=. (dblquote 0 1 {"1 csv) (1 0)}"1 csv  
wpt=. (<LF,'<wpt lat=') ,. (0{"1 csv) ,. (<' lon=') ,. (1{"1 csv) ,. <'>'
```

NB. times set to now

```
wpt=. wpt ,. <'time' eletags nstamp=. gpxtimestamp 6!:0''
```

NB. waypoint names & descriptions

```
wpt=. wpt ,. _1 |."1 names ,"0 1 |. tags 'name'  
NB. wpt=. wpt ,. _1 |."1 (alltrim&.> 3 {"1 csv) ,"0 1 |. tags 'desc'
```

NB. symbols

```
wpt=. wpt ,. <'sym' eletags 'waypoint'  
wpt=. wpt ,. <'</wpt>'
```

```
NB. waypoint format date
gpxhead=. ('/{date}}/', }:nstamp) changestr GPXHEADER

NB. return gpx
gpxhead,(;wpt),LF,'</gpx>'
)

gpxfrrecent=: 3 : 0

NB.*gpxfrrecent v-- gpx from recent waypoints.
NB.
NB. monad: clGpx =. gpxfrrecent clMirrorDb
NB.
NB.   trg=. jpath '~addons/jacks/testdata/small_mirror.db'
NB.   gpx=. gpxfrrecent trg
NB.   (toHOST gpx) write jpath '~temp/recent_geotagged.gpx'
NB.
NB. dyad: clGpx =. clGpxFile gpxfrrecent clMirrorDb
NB.
NB.   lastgpx=. jpath '~temp/geotagged_images.gpx'
NB.   lastgpx gpxfrrecent trg

MIRRORGPXFILE gpxfrrecent y
:
waydate=. waystamp gpx=. read x NB. extract last waypoint date

NB. the last upload date is shifted forward to partly compensate
NB. for the mixture of UTC and local dates. The times in the database
```

NB. come from many time zones and many timestamps are just approximations.

```
sql=. GpxGeotaggedMirror_sql , ' and UploadDate > date("",waydate,"", ''+16 hours'') order by UploadDate d
>..>esc '
fmtmirrorgpx sql fsd y
)
```

```
gpxtimestamp=: 3 : 0
```

*NB.*gpxtimestamp v-- format time for Garmin gpx as: yyyy-mm-ddThr:mn:scZ*

NB.

NB. monad: cl =. gpxtimestamp nlTime | ntTime

NB.

NB. gpxtimestamp 6!:0 ''

NB.

NB. gpxtimestamp 10 # ,: 6!:0 '' NB. table

```
r=. }: $y
t=. _6 [\ , 6 {."1 y
d=. '--T::' 4 7 10 13 16 }"1 [ 4 3 3 3 3 3 ": <.t
c=. {: $d
d=. ,d
d=. '0' (I. d=' ')} d
'Z' ,"1~ (r,c) $ d
)
```

NB. extract html placeholder variable names

```
htmlvarbs=: { -.&.> (<{'{'}')"
```

NB. file name from fully qualified file names

```
justfile=: ([ #~ [: -. [: +./\ '.'&=)@([ #~ [: -. [: +./\ . e.&':\')
```

```
parsecsv=: 3 : 0
```

*NB.*parsecsv v-- parses comma delimited files. (x) is the field*

NB. delimiter. Lines are delimited with either CRLF or LF

NB.

NB. monad: btcl =. parsecsv cl

NB. dyad: btcl =. ca parsecsv cl

NB.

NB. ', ' parsecsv read 'c:\comma\delimited\text.csv'

```
', ' parsecsv y
```

```
:
```

```
'separator cannot be the " character' assert -. x -: '''
```

*NB. CRLF delimited *.csv text to char table*

```
y=. x ,. ] ; _2 y -. CR
```

NB. bit mask of unquoted " field delimiters

```
b=. -. }. ~:/\ '"" e.~ ' ' , , y
```

```
b=. ($y) $ b *. , x = y
```

NB. use masks to cut lines

```
b <; _1"1 y
```

```
)
```

```
patpartstr=: 4 : 0
```

```
NB.*patpartstr v-- split list into sublists of pattern and non-pattern.
```

```
NB.
```

```
NB. dyad: (ilIdx ;< blcl) =. clPattern patpartstr clStr
```

```
NB.
```

```
NB. 'hoo' patpartstr 'hooohooohoo'
```

```
NB. 'ab.c' patpartstr 'abhc yada yada abNcabuc boo freaking hoo'
```

```
NB. 'nada' patpartstr 'nothing to match'
```

```
NB.
```

```
NB. NB. result pattern indexes and split list
```

```
NB. 'idx subtrs'=. 'yo[a-z]*' patpartstr 'yo yohomeboy no no yoman'
```

```
NB. idx{subtrs NB. patterns
```

```
NB. require 'regex' !(*)=. rxmatches
```

```
if. #pat=. , "2 x rxmatches y do.
```

```
mask=. (#y)#0
```

```
starts=. 0 {"1 pat
```

```
ends=. starts + <: 1 {"1 pat
```

```
m1=. 1 (0,starts)} mask
```

```
m2=. _1 (!.! 0) 1 ends} mask
```

```
m2=. m1 +. m2
```

```
mask=. 1 starts} mask
```

```
idx=. (m2 {.;.1 mask) # i. +/m2
```

```
idx;< m2 <;.1 y
```

```
else.
```

```
(i.0);<<y
```

```
end.  
)
```

NB. reads a file as a list of bytes

```
read=: 1!:1&[]`<@.(32&>@{3!:0}))
```

NB. read TAB delimited table files - faster than (readtd) - see long document

```
readtd2=: [: <;._2&> (a.{~9) ,&.>~ [: <;._2 [: (] , ((10{a.)"_ = {:) }. (10{a.)"_ ) (13{a.) -._ 1!:1&[]`<@.(  
>..>32&>@{3!:0}))
```

NB. removes multiple blanks (char only)

```
rebc=: ] #~ [: -. ' ' &E.
```

NB. deletes all blank rows from character table

```
rebrw=: ] #~ [: -. [: *./"1 ' ' &=
```

NB. radians from degrees

```
rfd=: *&0.0174532925199432955
```

NB. sine radians

```
sin=: 1&o.
```

NB. xml BEGIN and END tags

```
tags=: '<'&,@,&'>' ; '</'&,@,&'>'
```

```

timestamp=: 3 : 0

NB.*timestamp v-- formats timestamp as dd mmm yyyy hr:mn:sc
NB.
NB. monad: cl =. timestamp zu / nlTime
NB.
NB.   timestamp ''           NB. empty now
NB.   timestamp 2007 9 16    NB. fills missing
NB.   timestamp 1953 7 2 12 33

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
)

NB. appends trailing line feed character if necessary
tlf=: ] , ((10{a.)"_ = {:) }. (10{a.)"_

NB. extract waypoint date from gpx metadata header
waystamp=: [: alltrim '=' afterlaststr '</text>' beforestr ]

```


NB. writes a list of bytes to file

```
write=: 1!:2 ]`<@.(32&>@{3!:0))
```

NB.POST_gpxutils post processor.

```
smoutput IFACE_gpxutils=: (0 : 0)
```

```
NB. (gpxutils) interface word(s): 20250719j180157
```

```
NB. -----
```

```
NB. alaska_yukon_wpt - update alaska yukon waypoints
```

```
NB. allrecent        - all recent images from last waypoint generation
```

```
NB. csvfrgm          - poi CSV text from google maps lat, lon, desc layout
```

```
NB. csvfirtab        - poi CSV text from TAB delimited text file
```

```
NB. csvfrwpt         - poi CSV text from waypoint text file
```

```
NB. gpskm            - distances in km from Google Maps coordinates
```

```
NB. gpxfrmmapkml     - gpx from Google maps kml
```

```
NB. gpxfrmirror      - extracts geotagged images from mirror_db and generates gpx
```

```
NB. gpxfrpoicsv      - converts poi csv files to gpx
```

```
NB. gpxfrrecent      - gpx from recent waypoints
```

```
)
```

```
cocurrent 'base'
```

```
coinsert  'gpxutils'
```

Index

afterlaststr, 8
afterstr, 8
alaska_yukon_wpt, 8
AllMirror_sql, 7
allrecent, 9
alltrim, 10
allwhitetrim, 10
arctan, 10
assert, 10

beforestr, 10
betweenstrs, 10
boxopen, 11

changestr, 11
charsub, 12
cos, 13
CR, 7
csvfrgm, 13
csvfirtab, 13
csvfrwpt, 15
ctl, 16

dblquote, 16
earthdist, 16

eletags, 18

fmtcd, 18
fmtmirrorgpx, 19
fsd, 20
fst, 21

geteletext, 21
gpskm, 21
GPXFRKMLHEADER, 5
gpxfrmakml, 23
gpxfirmirror, 24
gpxfrpoicsv, 25
gpxfirrecent, 27
GpxGeotaggedMirror_sql, 7
GPXHEADER, 6
GPXNAMECHARS, 7
GPXSMUGPLACEMARK, 6
gpxtimestamp, 28
GPXTRAILER, 6

HTMLVARBPATTERN, 7
htmlvarbs, 28

IFACE_gpxutils, 33
IFACEWORDSgpxutils, 7

justfile, 29

LF, 7

MeeusHomeLonLat, 8
MIRRORGPXFILE, 7

parsecsv, 29
patpartstr, 30

read, 31
readtd2, 31
rebc, 31
rebrow, 31
rfd, 31
ROOTWORDSgpxutils, 8

sin, 31

tags, 31
timestamp, 32
tlf, 32

VMDgpxutils, 8

waystmp, 32
write, 33