Elevations
There are 13 elevations for which data is available: FL080, FL100, FL150, FL180, FL210, FL240, FL270, FL300, FL330, FL360, FL390 and FL420.

The elevation represents flight level and so assumes the ISA standard base pressure of 1013.25hPa at mean sea level, or 29.92inHg.

File contents
Each elevation data file contains the mean windspeed and direction for every 2.5 x 2.5 degree grid square over North America, for all 365 days of the year and for times of 00Z, 06Z, 12Z and 18Z within each day. Each data point is a 2-byte unsigned integer. The integer stores two decimal places of precision - the source value is multiplied by 100 then rounded to nearest integer and stored in 16 bits.

File structure

Bytes from index 0 through 10 are the “header” of the file and describe the maximum and minimum latitudes and longitudes contained within, as well as the step size (in degrees) between each grid square. All values in the header, except for step size, represent **inclusive** values. For the North American dataset, minlat/maxlat should be the range [20, 70], minlong/maxlong should be [-175, -45] and total cycles should be [0, 1459].

Byte 0-1: **16-bit signed integer.** Total number of cycles. Acceptable cycles are [0,totalCycles]
Byte 2-3: **16-bit signed integer.** Maximum latitude (furthest north). Divide by 10 to get the original value. This is the **latitude origin**
Bytes 4-5: **16-bit signed integer.** Minimum latitude. Divide by 10 to get the original value
Bytes 6-7: **16-bit signed integer.** Maximum longitude. Divide by 10 to get the original value
Bytes 8-9: **16-bit signed integer.** Minimum longitude (furthest west). Divide by 10 to get the original value. This is the **longitude origin**.
Byte 10: **8-bit signed integer.** Step size - the distance in degrees between two grid points. Divide by 10 to get original value.

Byte 11 - EOF: Multidimensional array stored in row-major format, described below.

Warning: leap years

This data set only contains 365 days, so leap years will break if you use a day of year of 366. To avoid this, on leap years try taking the average of 28th Feb and 01 Mar (59 and 60th days of year) to stand in for Feb 29 on a leap year.

Model cycles
The models that analysed the historical wind data ran for times of 00Z, 06Z, 12Z and 18Z. Each of these runs is called a **cycle**.

Thus each data file contains the mean wind speeds for four cycles per day over 365 days in a year, taken over the 29 year period 1981-2010.

Latitudes and Longitudes

This dataset covers **only** North America. Specifically, it covers the following lat/long boundings:

-175E, 70N	-45E, 70N
-175E, 20N	-45E, 20N

Grid format

The source data treats breaks the globe into 2.5 by 2.5 degree grid squares and provides wind information for each grid square (thus giving 150nm x 150nm of resolution).

For a total of 21 rows (latitude bands 70 - 20 inclusive) and 53 columns (longitudes -174 to -45 inclusive), all with a 2.5 degree step between them.

Within each file this grid is represented as four-dimensional array with the following shape:

```
[cycle_number][latitude_grid][longitude_grid][wind_component]
```

If this array was defined in C, it would look like this:

```
unsigned short windData[1460][21][53][2];
```

The entry **windData[0][0][0][0]** would be: cycle number 0 (Jan 01, 0000Z), latitude grid 0 (+70 degrees), longitude grid 0 (-175 degrees), component 0 (wind speed).

More examples:

windData[1][0][0][1] = wind *direction* at +70, -175 for the 0600Z forecast on 01 Jan.

windData[2][20][52][0] = wind speed at +20, -45 for the 1200Z forecast on 01 Jan.

A latitude and longitude within the above range must be converted to a (row,col) pair before you can index into the array.

Each value in the array represents the long term mean wind speed and direction at the grid point represented by the row and column.

Calculating a lat/long grid from an input lat/long

The following formulas (in Python) provide a solution for this problem:

```
def floorMod(a,n):
  return a - n * math.floor(a/n)

def ll2ij(lat, lon, latOrigin, longOrigin, dLon, dLat ):
  i = ( latOrigin - lat ) / dLat
  j = floorMod( lon - longOrigin, 360 ) / dLon
  return [int(math.round(i)),int(math.round(j))]
```

Where latOrigin, lonOrigin, dLon and dLat can be found in the header of the file. dLon and dLat will be the same (i.e. stepsize of 2.5 degrees).

Calculating the cycle number

Cycle numbers increase by one for every 6 hour increment from midnight UTC (00Z) on 01 Jan. Just like UNIX timestamps represent the number of seconds since 01 Jan 1970, cycle numbers are the number of 6hr steps since 01 Jan. They grow from 0 (representing 0000Z on 01 Jan) to 1459 (representing 1800Z on 31 December).

So if you want the 18Z cycle of the 281st day of the year:

```
cycle_num =  (cycles_per_day * ( day_of_year - 1 ) ) +
(cycle_within_day - 1)

# or for the 18Z (4th) cycle of the 281st day of the year:
cycle_num = (4 * 281 - 1) + (4 - 1) = 1126
```

Where 00Z = 1st cycle, 06Z = 2nd cycle, 12Z = 3rd cycle and 18Z = 4th cycle. Minus 1 to keep a zero-base for the index.

Component number

If you want wind speed, component index is zero (0). **Wind speed is reported in knots**.
If you want wind direction, component index is one (1). **Wind direction is reported in degrees and is the direction the wind is coming from**.

Putting it together
So the mean wind speed for the 12Z cycle of the 100th day of the year, at the coordinate -150E 50N is represented in our multi-dimensional array as:

```
spd = [402][8][10][0]
dirn = [402][8][10][1]
```

Where
```
cycle_num = 402
lat_grid = 8
lon_grid = 10
component = 0
```

Extracting wind data from a grid square

You must know the desired cycle number (zero-based), the latitude and longitude grids (row,col) in order to extract the wind data. You can must use these values to calculate the binary offsets into your elevation file.

The wind data files contain an array of `unsigned short` values in **row-major format**. The following formulas will calculate the offset into the file for the speed and direction components respectively:

```
spd_offset = FILE_HEADER_SIZE + ( ( ( cycle * num_lat + latGrid ) * num_lon + lonGrid ) * num_components + 0 ) * data_size

dirn_offset = FILE_HEADER_SIZE + ( ( ( cycle * num_lat + latGrid ) * num_lon + lonGrid ) * num_components + 1 ) * data_size
```

Where:
```
num_lat = 21
num_lon = 53
num_components = 2
data_size = 2
FILE_HEADER_SIZE = 11 bytes
```

Example:

We want the wind speed and direction for the 1200Z cycle for the 100th day of the year, at the -150 E, 50N (grid reference [8,10]).

```
cycle = ( 4 * ( 100 - 1 ) ) + (3 - 1) = 398
num_lat = 21
num_lon = 53
```

```
num_components = 2
latGrid = 8
lonGrid = 10

speed_file_offset = 11 + ( ( ( 398 * 21 + 8 ) * 53 + 10 ) * 2 + 0 ) * 2 = 1,773,643
dirn_file_offset = 11 + ( ( ( 398 * 21 + 8 ) * 53 + 10 ) * 2 + 1 ) * 2 = 1,773,645
```

Example 2: We want the wind speed and direction for the 0000Z cycle for the 1st day of the year, at 70N, -145 E (grid reference [0,0]):

```
cycle = ( 4 * ( 1 - 1 ) ) + (1 - 1) = 0
num_lat = 21
num_lon = 53
num_components = 2
latGrid = 0
lonGrid = 0

speed_file_offset = 11 + ( ( ( 0 * 21 + 0 ) * 53 + 0 ) * 2 + 0 ) * 2 = 11
dirn_file_offset = 11 + ( ( ( 0 * 21 + 0 ) * 53 + 0 ) * 2 + 1 ) * 2 = 13
```

Converting to original windspeed and direction

Take these values and **divide by 100** to return to the original windspeed and/or direction to two decimal places of precision. In Python:

```
f.seek(spd_offset)
spd = struct.unpack('H',bytearray(f.read(2)))[0]
spd = spd / 100.0

f.seek(dirn_offset)
dirn = struct.unpack('H',bytearray(f.read(2)))[0]
dirn = dirn / 100.0
```

With a 16-bit integer and two decimal places of accuracy the highest wind speed that can be stored is approximately 655kts.

A note on wind direction

The wind direction recorded in this file is the **meteorological direction** - i.e. the direction the wind is blowing **from**. If you need to convert this to the azimuthal direction - the direction the wind is blowing to - then you must add or subtract 180 degrees as appropriate.

Wind direction will be in the range 0 to 359 degrees.

Result

The long term average windspeed and direction at 21,000ft over New York City (40.7127 N, 74.0059 W) on Christmas Day at 12Z:

```
adam@adam-Inspiron-3537:~/Desktop/windsaloft$ python windspd.py elev/north_america/21000
File looks good! Covers lat range [70.0, 20.0] and long range [-175.0, -45.0]. Step size 2.5
degrees. 21.0 rows and 53.0 cols. Cycle range [1,1460].
Enter a date string (MM/DD/YYYY). Leave blank for today: 12/25/2015
OK! Date = 12/25/2015, Day of year = 359
Cycle within day? ( 1 = 00Z, 2 = 06Z, 3 = 12Z, 4 = 18Z): 3
What latitude?: 40.7127
What longitude?: -74.0059
[1434.0, 12, 40] = 63.78kts @ 254.62deg
```

How this data was calculated

This data was calculated using the original source data of daily mean wind speeds from the NCEP/NCAR re-analysis 1 project over 1981-2010. This data set provided isobaric surfaces at 1000, 950, 900, etc, hPa. Each of these pressure surfaces was broken up into the 2.5x2.5 degree grid, with one grid per pressure level per cycle. Each cycle was at 00, 06, 12 and 18Z.

U (east to west) and V (south to north) components of wind were combined to create a five dimensional array of cycle, pressure level, latitude grid, longitude grid, windspeed and direction.

The pressure levels provided by the NCEP/NCAR reanalysis do not line up nicely with the calculated levels of 8000ft to 39,000ft. By working backwards using standard barometric pressure formulas it is possible to determine the estimated atmospheric pressure that corresponds to a desired elevation (say 33,000ft, which corresponds to approximately 262hPa of pressure). From that you can interpolate the source data to calculate windspeed at the desired elevation.

For example, an elevation of 33,000ft = 262hPa of pressure assuming the ISA standard atmosphere (MSL pressure = 1013.25hPa/29.92inHg). Pressure surfaces of 300hPa and 250hPa were then used to interpolate the u and v components of windspeed at 262 hPa using a linear-log pressure formula. The resulting grid was then saved in the source file.

Worldwide data is available but is approximately 10 times larger in file size.