# 11_File_I-O

September 22, 2019

# 1 SUMMARY OF FILE HANDLING IN Q

## 1.1 1. Running q scripts

- There are 3 ways to run q scripts:
  - from script with a q command: **ł/path/to/file.q**, this is load command:
    - does not work from kdb studio
    - only works with absolute path
  - from script with q's system command: system "l /path/to/file.q"
  - from terminal: ~$$: q /path/to/file.q arg1 arg2 ...
- The 1st and the 2nd is practically the equivalent of load module or import in other languages.

# 2 2. Reading and writing data files

- **Handling binary files**:
  - no instantiation of readers, writers or serializers
  - symbolic handels represents files: `: (backtick + colon + path + filename)` ':path/filename.ext
  - set: save / write function, which saves the data (list, table, etc) in a serialized binary data file (if success, echos symbolic file handle to stdout)
  - get: read function to read in a file (if successful, echos nothing) into q entities (list table etc)
- **Handling txt files**
  - saving strings to text files: "0:" operator
    - syntax: `fileHandle` 0: ("first line";"second line")
  - reading text files to strings: myVar:read0 'simbolic_file_handle
    - read0 will read the lines of a txt file as elements of a string list
- **Handling csv files**
  - three different overloads of the 0: operator
    - writes list of strings to text files (see handling text files)
    - prepares the tables as text
      - q handles the quoting and escaping of special characters
    - parsing formatted csv into q table

[ ]:
```
`:testfiles/tio.csv 0: csv 0: tio / prepares tables as text, creates a comma␣
↪separeted text file, then writes it to a csv file
```

[ ]:
```
csvString:read0 `:testfiles/tio.csv / reads csv file into list of strings
```

[ ]:
```
tio_read:("SF"; enlist ",") 0: `:testfiles/tio.csv / reads csv file directly␣
↪into q table with known column types (symbol and float)
```

- **3 ways to read a binary file into memory as a q entity**
  - łpath/to/file (NOT A FILE HANDLE!)
    - łassigns the content of a file to an entity with the same name as the file
  - get filehandle
  - value ‘filehandle
- Writing and overwriting a file with *dot amend* notation (same as set):
  - serializes q entities to a file
  - .: writes or overwrites a file
  - ,: appends to a file

# 3 11. File I/O

## 3.1 11.1. Bynary data

[3]:
```
path:"/home/iguana/1_Code/4_jupyter_projects/q4m3_tutorial/data/"
path_file:"/home/iguana/1_Code/4_jupyter_projects/q4m3_tutorial/data/fox.us.csv"
```

[1]:
```
 / cast: $
show `$"df df d f" / cast to symbol
hsym `$"df/fg rt/df.ext" / cast to symbolic file handle inserting a colon at␣
↪the beginning of symbol: `$":fdf/df"
hsym `$"/data/file name.csv"
hsym hsym `$"/data/file name.csv" / idempotent: passes output through
```

```
`df df d f
```

[1]:
```
`:df/fg rt/df.ext
```

[1]:
```
`:/data/file name.csv
```

[1]:
```
`:/data/file name.csv
```

[4]:
```
hcount hsym `$path_file / hcount returns the size of file in bytes
```

[4]:
```
158527
```

[7]:
```
hdel hsym `$path,"t" / delete file by hanle name
```

[7]:
```
`:/home/iguana/1_Code/4_jupyter_projects/q4m3_tutorial/data/t
```

### 3.1.1 11.1.3. Serializing and seserializing

- every q entity can be serialized and persisted to storage
- write a q entity to binary file (create or overwrite file): set
  - filehandle set q entity
- read binary file as entity (three ways)
  - get filehandle
  - value filehandle
  - łpath/to/file (NOT A FILE HANDLE!)
    - łassigns the content of a file to an entity with the same name as the file

### 3.1.2 11.1.4. Binary data files

- opening a symbolic handle returns a function, called an *open handle*, that is used to perform operations.
- hopen: returns the *open handle* function
  - hopen filehandle (if file does not exist, creates it)
    - you can use the variable value with the open handle assigned to it to do the same operation as with the variable name
  - can open a websocket
- hclose: closes the open handle function
  - if handle refers to a websocket, it is only closed after all pending data is sent
- Append to file

[8]:
```
file:hsym `$(path,"openHandle")
file set 10 20 30
h:hopen file
h[42]
h 100 200
hclose h / always close handle
```

[8]:
```
`:/home/iguana/1_Code/4_jupyter_projects/q4m3_tutorial/data/openHandle
```

[8]:
```
5i
```

[8]:
```
5i
```

[9]:
```
get file
```

[9]:
```
10 20 30 42 100 200
```

### 3.1.3 Writing and reading binary

- read1 reads in binary data as a list of bytes. this show the internal representation of the serialized q entity
- 1: writes raw binary data to a binary file

[10]:
```
read1 file
```

[10]:
```
0xfe2007000000000003000000000000000a0000000000000014000000000000001e000000000..
```

[ ]:
```
file2:hsym `$path,"answer.bin"
file2  1: 0x06072a
read1 file
```

- *dot amend* (same as set):
  - serializes q entities to a file
  - .: writes or overwrites a file
  - ,: appends to a file

[ ]:
```
.[file; (); :; 1001 1002 1003]
q)get file
```

[ ]:
```
.[file; (); ,; 42]
get file
```

## 3.2 11.2. Save and load on tables

- to save a table into a binary file, use the same syntax as writing or reading any type of q entities (set, get)
- a simpler method is save and load: you do not have to specify the name of the returned table: it implicitely creates the variable / file
  - save filehandle (where the file name has to be the same as the table name)
  - load filehandle (where the resulting table name is the same as the filename)
  - if you specify an extension (**.txt, .csv, .xml or .xls**) it will be converted to that format
    - .txt table saved as tsv (tab separated values)
    - .csv table is saved as comma separated values
- you can save a table as csv:

– filehandle 0: csv 0: table

[ ]:
```
file set ([] c1:`a`b`c; c2:10 20 30; c3:1.1 2.2 3.3)
get file
```

[14]:
```
system "touch fiot"
```

[14]:

[15]:
```
file3:hsym `$path,"fiot"
```

[13]:
```
fiot:([] c1:`a`b`c; c2:10 20 30; c3:1.1 2.2 3.3)
```

[16]:
```
save file3
```

[16]: `:/home/iguana/1_Code/4_jupyter_projects/q4m3_tutorial/data/fiot

[17]:
```
load file3
```

[17]: `fiot

[ ]:
```
 / save as .txt
file4:hsym `$path,"t.txt"
t:([] c1:`a`b`c; c2:10 20 30; c3:1.1 2.2 3.3)
save file4
read0 file4
("SIF"; enlist "\t") 0: file4  / no corresponding load or get. you have to parse
```

[ ]:
```
 / save as .csv
file5:hsym `$path,"t.csv"
t:([] c1:`a`b`c; c2:10 20 30; c3:1.1 2.2 3.3)
save file5
read0 file5
("SIF"; enlist ",") 0: file5 / no corresponding load or get. you have to parse
```

[ ]:
```
 / save as .xml (you have to use outside [libraries](http://code.kx.com/q/
 ↪github) to read xml files)
file6:hsym `$path,"t.xml"
t:([] c1:`a`b`c; c2:10 20 30; c3:1.1 2.2 3.3)
save file6
read0 file6
```

[ ]:
```
file7:hsym `$path,"t.xls"
t:([] c1:`a`b`c; c2:10 20 30; c3:1.1 2.2 3.3)
save file7
```

```
read0 file7
```

## 3.3 11.3. Splayed tables

- serialize large tables by their columns into separate files
- just use a trailing forward slash at the end of the path symbol set table
- Restrictions on tables that can be splayed:
  - All columns must be simple or compound lists (a list of simple lists of uniform type)
  - An arbitrary general list column cannot be splayed.
- Symbol columns must be enumerated.

[ ]:
```
folder1:hsym `$path,"t_folder/"
t:([] c1:10 20 30; c2:1.1 2.2 3.3)
folder1 set t
```

## 3.4 11.4. Text data

### 3.4.1 11.4.3. Reading and writing text files

- read0 reads a .txt file as a list of strings
  - read0 ‘file.txt
- 0: writes a list of strings as a .txt file
  - ‘file.txt 0: tabel_entity
- use hopen and hclose to handle .txt files
- neg[h] (value1;value2) appends string values to the existing file
- saving to .csv file while controlling the filename (vs save hsym `table_name.csv): `-name.csv 0: csv 0: table

### 3.4.2 11.5. Parsing records

- parses text file with equal length rows into a table with columns of specified type and length
- Dyadic forms of 0: and 1: parse individual fields according to data type from text or binary records.
- Field parsing is based on the following field types !!!!

[ ]:
```
data:("JFS D";4 7 10 7 10) 0: `:./data/fixed.txt / creates a nested list
flip `c1`c2`c3`c4!data / create table from the list of lists
```

### 3.4.3 14.5.2. Variable length records

### 3.4.4 14.5.3. Key-value records

[18]:
```
ll1:"S=;" 0: "one=1;two=2;three=3"
"S:/" 0: "one:1/two:2/three:3"
"I=;" 0: "1=one;2=two;3=three"
```

[18]:
```
one  two  three
,"1" ,"2" ,"3"
```

[18]:
```
1     2     3
"one" "two" "three"
```

[19]:
```
flip `k`v!ll1
```

[19]:
```
k     v
----------
one   ,"1"
two   ,"2"
three ,"3"
```