File and Path Handling

MATLAB works very well across operating systems and users. However, the naming structure for files and folders often differs between computers and users. This document aims to reduce problems associated with files, and paths to files.

For example, on a Mac, the full name of a file might be

`/Users/davidatkinson/matlab/startup.m`

whereas on a Windows PC, it might be

`C:\Users\David\matlab\startup.m`

We avoid putting the '/' or '\' separators explicitly in code, and instead use the function `fullfile`. For example;

```
fullfilename = fullfile('Users','davidatkinson','matlab','startup.m') ;
```

To get the path, filename and extension, use `fileparts`. For example,

```
[ pathnm, filenm, ext] = fileparts(fullfilename) ;
```

To reconstruct a complete filename, note that after using `fileparts`, the `filenm` part does not include the extension and it needs to be added on as below.

```
newfilenm = fullfile( pathnm, [filenm ext])
```

```
newfilenm =
'Users/davidatkinson/matlab/startup.m'
```

Paths

Files and folders are arranged in a folder tree. The term 'path' usually refers to all the folders above the current. An absolute path starts at the top level, whereas a relative path might be from a given point in the folder tree. For example, the absolute path to the startup.m file above is '/Users/davidatkinson/matlab/'. This should be unambiguous from anywhere on the computer, but is often user dependent because it contains the username. If you were working from the folder '/Users/davidatkinson', then the relative path to the file would be 'matlab/'.

To improve clarity, it is common to have a project root folder and then in code express paths in relative terms.

The function `exist` can be useful for checking that naming is correct. For example

```
if ~exist( newfilenm, "file")
    disp( newfilenm + " not found - check path")
end
```

The term "path" also means the set of folder paths that MATLAB will use when searching for a file. If a file is "on the path" it is in a one of the folders listed in the path. Usually people add commonly used folders to their path so that they don't have to specifiy absolute or relative paths when they want to use that file. When sharing code with others, it is hard to rely on any specific path being available to a different user.

Where am I?

The current working folder is returned by pwd:

```
currentFolder = pwd
```

```
currentFolder =
'/Users/davidatkinson/matlab/Projects/education/source'
```

When running LiveScripts, the file and path of the running LiveScript can be found:

```
matlab.desktop.editor.getActiveFilename
```

```
ans =
'/Users/davidatkinson/matlab/Projects/education/source/odds_plot.mlx'
```

Note that you can run a LiveScript when in a folder that is different from the one containing the LiveScript.

If you arrange files and folders for example as below, you can use the above function and a relative path to avoid needing to know the names of the higher level folders.

```
Projects
|-- education
|-- source
|-- path_handling.mlx
|-- data
|-- measurements.txt
```

```
rpathData = 'data' ; % relative path from the 'education' folder
fullnameThisScript = matlab.desktop.editor.getActiveFilename
```

```
fullnameThisScript =
'/Users/davidatkinson/matlab/Projects/education/source/odds_plot.mlx'
```

```
pathThisScript  = fileparts(fullnameThisScript) ; % will give the path to
the script
pathAboveScript = fileparts(pathThisScript) ;     % goes 'up' one level
(here to education
pathData = fullfile( pathAboveScript, rpathData) ;
```

However, if using buildtool to export a Live Script, the `getActiveFilename` refers to the buildfile. See the samplingandse LiveScript for more details.