QLTools

Introduction

`Qltools` is a utility that allows you to access files stored on a QL Floppy Disc, or an image of a floppy, whether you are running on Linux (or other Unices) or Windows.

`Qltools` was originally written by **Giuseppe Zanetti** and the following list of people have had a hand in it, somewhere or other:

- Valenti Omar;
- Richard Zidlicky;
- Jonathan Hudson;
- Graeme Gregory;
- Norman Dunbar.

Downloading Binaries

For now, binaries are only available from my own Github account at https://github.com/NormanDunbar/qltools/releases where you will always be able to find the latest version.

You may also like to check https://github.com/SinclairQL/qltools/releases too, as that is the official site for `qltools`.

Compiling From Source

In theory, you should change to the `qltools` directory and `make xxxx` where `xxxx` is the environment you wish to create a working binary for, however, it appears that this only works on a Linux environment.

It is assumed that whether you are working on a Windows or Linux desktop, that you have the gcc compiler installed. On Linux, this is pretty much the default. On Windows, however, it will unlikely be there. I find that the Code::Blocks IDE is a very good IDE for working with Unix utilities on a Windows environment for many reasons, but specifically:

- It works!
- It includes the `gcc` and `g++` compilers;
- It includes a debugger too - `gdb`;
- All the Unix `#include` header files, `unistd.h` for example,are included with the IDE;
- It's free!

If you have, for example, Visual C or Embarcadero C/Borland C, then `unistd.h` will not be found and you will not be able to compile the utility until you find a replacement.

The compilation process is:

On Windows:

```
cd qltools\Win7
mingw32-make
```

In case you are wondering, `mingw32-make` is the supplied `make` utility that comes with `Code::Blocks`.

On Linux/Unix:

```
cd qltools/Unix
make
```

It should be noted that the `Hxcfe` directory has not been updated for any recent changes in `qltools`, nor, according to Graeme Gregory, has it been amended due to changes in the API used. It is assumed to be a dead system at the moment.

Within the `Win7` directory, you will find a `Code::Blocks` project file for use on Windows.

Inbuilt Help

If you run `qltools` without any options, it will display the following, potentially helpful, screen of information:

```
qltools

Error: too few parameters

Usage: qltools device -[options] [filename]

Options:

    -d         List directory          -s         List short directory
    -i         List disk info          -m         List disk map
    -c         List conversion table   -l         List files on write

    -w <files> Write files (query)     -W <files> (Over)write files
    -r <name>  Remove file <name>      -n <flle>  Copy <file> to stdout

    -uN        ASCII dump cluster N    -UN        Binary dump
    -M <name>  Make level 2 directory <name>
    -x <name> <size> Make <name> executable with dataspace <size>
    -t         Do not translate '.' to '_' in filenames

    -fxx <name> Format as xx=hd|dd|ed disk with label <name>

QLTOOLS for Linux/win32 (version 2.15.5, Jan 31 2019)

'Device' is either a file with the image of a QL format disk
or a floppy drive with a SMS/QDOS disk inserted in it (e.g. diskimage)

by Giuseppe Zanetti,Valenti Omar,Richard Zidlicky, Jonathan Hudson
& Norman Dunbar
```

I can never remember the options, so this is very useful, in my opinion!

Usage of Qltools

Once you have a binary, the general form of the command line is as follows:

```
qltools device -[options] [filename]
```

Not all options require a filename.

In the following text, the terms **block**, **cluster** and, perhaps, **allocation unit** will be seen to be used interchangeably. This is because the options given by the `qltools` utility make reference to **clusters** and label output with the title **blocks** when, in actual fact, these all refer to the same thing - an **allocation unit** on the disc.

Allocation units are the smallest chunk of floppy that can be assigned to a single file. If an empty file is created on a floppy, it may *appear* to be zero bytes in size, but it actually takes up one **allocation unit** of space on the floppy.

The actual number of bytes in an **allocation unit** is determined by the floppy type, DS/DD, DS/HD or DS/ED where the sizes are 1,536, 1,536 and 2,048 bytes respectively.

Why so many names for the same thing!

The Device

The device can be an image file or it can be a hardware device holding an actual floppy disc. `Qltools` can work happily with floppy discs and images (henceforth referred to as `floppies`) in the following formats:

- DS/DD - 720 Kb Double sided floppies;
- DS/HD - 1,440 Kb Double sided floppies;
- DS/ED - 3.2 Mb Double sided floppies.

The Options

The following options are permitted when using `qltools`.

Formatting

The following formatting options are permitted:

- `-fdd` to format a DS/DD floppy;
- `-fhd` to format a DS/HD floppy;
- `-fed` to format a DS/ED floppy.

An optional floppy name is permitted, as per the following example:

```
qltools ed.img -fed TestImage
```

There's no confirmation that it has been successful, in the usual Unix manner of things. However `%ERORLEVEL%` or `$?` should give you a clue as zero means all worked ok.

File Name Conversion

When writing files to the floppies, QL filenames have an underscore as the directory and file extension separator - `Win1_Source_qltools_c`, for example, while Other Operating Systems do not.

By default, `qltools` will convert to and from the QL and Host naming convention but if you do not wish this to happen, then using the `-t` option suppresses name translation.

For example:

```
qltools ed.img -w linux.c
qltools ed.img -t -w linux.c

qltools ed.img -s

linux_c
linux.c
```

The `-t` option must go prior to the `-w` and the file list.

List Directory

The `-d` option lists a floppy directory including:

- The floppy name
- Free/total sectors
- File details

For example:

```
qltools ed.img -d

TestImage
1591/1600 sectors.

linux_c       1918 31/01/2019 15:05:50 v0
linux.c       1918 31/01/2019 15:06:44 v0
qltools_exe E 4416 31/01/2019 15:13:00 v0  1234
```

The last entry above has an 'E' to indicate executable, and the '1234' at the end is the data space.

The first column of numbers is the file size in bytes, and is followed by the date and time it was created and the version number.

List Short Directory

The `-s` option is similar to the above, but only lists the file names found on the floppy.

For example:

```
qltools ed.img -s

linux_c
linux.c
qltools_exe
```

List Disc Information

The `-i` option lists information about the floppy.

For example, this is an ED format disc image:

```
qltools ed.img -i


Disk ID               : QL5B
Disk Label            : TestImage
Number of sides       : 2
Sectors per track     : 10
Sectors per cylinder  : 20
Number of cylinders   : 80
Sectors per block     : 1
Sector offset/cylinder: 2
Random                : 0645
Updates               : 4
Free sectors          : 1570
Good sectors          : 1600
Total sectors         : 1600
Directory is          : 0 sectors and 256 bytes

Logical-to-physical sector mapping table:

0 2 4 6 8 80 82 84 86 88 1 3 5 7 9 81 83 85 87 89
```

List Disc Map

The `-m` option lists the entries in the floppy's map. There is one line of output for every entry in the map which means up to 1600 lines for an ED formatted floppy. You might wish to paginate the output with `more` or `less` rather than watching the text scroll up the screen at a high rate of knots!

For example:

```
qltools ed.img -m | less

block   file   pos

0       3968   0      (f80, 000) map
1       3968   1      (f80, 001) map
2       3968   2      (f80, 002) map
3       0      0      (000, 000) directory
4       1      0      (001, 000) linux_c
5       2      0      (002, 000) linux.c
...
9       4063   4095   (fdf, fff) unused
10      4063   4095   (fdf, fff) unused
...
1599    4063   4095   (fdf, fff) unused
```

The first column shows the block number. A block being a specific number of sectors, which the List Disc Information option above shows.

The second column shows the file number in decimal, followed by the section number of the file. In the example above, the map is made up of three sections taking up the first 3 blocks (aka allocation units aka clusters) of the floppy.

The two columns in parenthesis are just a repeat of the file and section number, this time in hexadecimal.

Finally, we see the file name, or one of the following:

- `map` for the floppy map;
- `unused` for free blocks;
- `bad` for blocks that are, ahem, bad;
- `directory` for blocks in the root directory of the device;
- `not existent` for blocks that don't exist on the floppy.

Write Files

There are two ways to write a file, or files, into the floppy from the Operating Systems. The options are:

- `-w` write the file and prompt to overwrite if it already exists;
- `-W` write the file and overwrite it, if it exists, without prompting.

For example:

```
qltools ed.img -w linux.c
file linux_c exists, overwrite [Y/N/A/Q] :
```

And, to overwrite without a care in the world:

```
qltools ed.img -W linux.c
```

You may, if you wish, supply a list of files, separated by spaces, so that multiple files can be written into the floppy in one execution.

Delete Files

The `-r` option allows you to remove files from a floppy *one at a time*. A list of files cannot be supplied.

For example:

```
qltools ed.img -t -r linux_c
qltools ed.img -t -r linux.c
qltools ed.img -t -r qltools_exe
```

I've used the `-t` option to ensure that the files removed are the ones I typed. I find it easier to use that option and not to have to try and remember that translation will be done. For example, if I were to type:

```
qltools ed.img -r linux.c
```

The file *actually* removed would be the one with the translated name, `linux_c`, so if I want to remove file `linux.c` I *must* use the `-t` option to prevent `qltools` translating my file names!

Extract Files

Files can be extracted from a floppy and are always written to the standard output or screen. However, you can redirect this to a named file.

The following example should make it clear:

```
qltools ed.img -n qltools_exe > qltools.exe
```

In this case, the binary file `qltools_exe` is extracted from the floppy and written to the file `qltools.exe` on the operating system.

Dump Clusters

The clusters of a file can be dumped out in ASCII (well, in hexadecimal and ASCII) or exactly as they appear on the floppy. For the former you would use the `-u` option and the latter uses the `-U`. In either case, the cluster number must appear after the `u` or `U` *without any spaces*.

The cluster number is found from listing the floppy map, as explained in List Disc Map above.

To dump out a cluster in ASCII:

```
qltools ed.img -u4

000 :  00 00 07 ff 00 00 00 00 00 00 00 00 00 00 00 07  ................
010 :  6c 69 6e 75 78 5f 63 00 00 00 00 00 00 00 00 00  linux_c.........
020 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
030 :  00 00 00 00 6d 40 8f 48 00 01 00 02 6d 40 8f 48  ....m@.H....m@.H
040 :  23 69 6e 63 6c 75 64 65 20 3c 73 74 64 69 6f 2e  #include <stdio.
050 :  68 3e 0d 0a 23 69 6e 63 6c 75 64 65 20 3c 73 74  h>..#include <st
060 :  64 6c 69 62 2e 68 3e 0d 0a 23 69 6e 63 6c 75 64  dlib.h>..#includ
...
7c0 :  4e 4f 4e 3b 0d 0d 0a 09 6f 6c 64 2e 63 5f 6c 66  NON;....old.c_lf
7d0 :  6c 61 67 20 7c 3d 20 45 43 48 4f 3b 0d 0d 0a 0d  lag |= ECHO;....
7e0 :  0d 0a 09 69 66 28 74 63 73 65 74 61 74 74 72 28  ...if(tcsetattr(
7f0 :  30 2c 20 54 43 53 41 44 52 41 49 4e 2c 20 26 6f  0, TCSADRAIN, &o
```

As this is an ED floppy, there are 2048 bytes in a sector and one sector in a cluster.

DD and HD floppies have 512 bytes in a sector and both have 3 sectors in a block, a cluster dump of those will display 1536 bytes.

You will note, hopefully, that the first 64 bytes of the first cluster in a file are a copy of the directory entry for the file and are not actually part of the file itself. For the *very first* cluster of a file, the actual file contents start at offset 0x40 - or '040' in the extract above.

To dump out a cluster *exactly* as it is on the floppy, you would proceed as follows:

```
qltools ed.img -U4
```

Which would write the contents of the cluster to the screen, binary codes and all, thus (possibly) corrupting your terminal sessions, or hanging it up if an `XOFF` code is sent (CTRL-S). YOu can, of course, dump the cluster to a file:

```
qltools ed.img -U4 > cluster_4.clu
```

Creating Level 2 Directories

Once you have written your files onto a floppy, you can organise the files into directories by creating a level 2 directory.

Unfortunately, it doesn't appear possible to write files from directories on the operating system into already existing directories on the floppy - or, at least, I have not yet found a way.

Equally, if a number of files already exists on the floppy, all with the same leading part of the name,

separated by underscores, then creating a directory will move the files into that directory.

You should note that a directory of the floppy will list both the directory and the files within. This is acceptable as there is (currently) no way to list the contents of a sub-directory.

Make files Executable

If a file on a floppy is currently not executable, or if you wish to change the dataspace for an existing executable, then the `-x` option is required.

This option will always make a file executable, and set the dataspace to the value given, which *should* be even, but will be rounded up to the next even number if not.

For example:

```
qltools ed.img -x qltools_exe 1234
qltools ed.img -d | grep "exe"

qltools_exe E 4416 31/01/2019 15:13:00 v0  1234
```

If the file is already executable, only the dataspace will be changed:

```
qltools ed.img -x qltools_exe 2467
qltools ed.img -d | grep "exe"

qltools_exe E 4416 31/01/2019 15:13:00 v0  2468
```

You can see, in the above example, that the odd dataspace value has been rounded up to an even number.

Index