QLTOOLS v2.4

Note: This document also (mainly) describes the functionality of qltools >2.4. If in doubt, please refer to the fine source code.

What is it

This document describes the features and limitations of the program qltools v2.4.

qltools v2.4 is a program for systems such as Linux, Windows NT, OS/2 and MSDOS (and Win95) to read and write floppy disks in the 'QL5A' or 'QL5B' format used by the QDOS operating system and its clones and near-siblings such as SMS2, SMSQ, SMSQ/E and most other things starting SMS.

qltools can be compiled for and should run on other Unix systems. It will also handle disk images under VAX/VMS, but this is not included in the distribution.

qltools v2.4 is based on the qltools program of Giuseppe Zanetti, with modifications by Valenti Omar and Richard Zidlicky.

v2.4 has been almost entirely rewritten by Jonathan R Hudson.

Features

- Fully supports level 2 directory operations
- Supports DD (720K) and HD (1440K) disks
- Supports disk images on all platforms.
- Read, write, delete files
- Create, delete sub-directories
- Initialise disks in SMS/QDOS format
- Dump disk sectors in ASCII or binary
- Display disk map

Changes from earlier versions

The support for level 2 sub-directories has resulted in almost a completely re-written program. Some options that were previously available (but didn't seem to work) have been removed. For example, I never got the the deleted file recovery to work, this functionality is now available via binary dump of sectors; I can verify that this does work !

Files are now only referred to by name, the level 2 scheme for enumerating files in sub-directories having made the previous file numbering scheme impractical.

Disk Images

Users of previous versions of 'qltools' under Unix may be familiar with the concept of a disk image created using the dd (1) program and well documented in Giuseppe's original manual. This functionality is now available to DOS (for DOS read MSDOS/Win95/OS2/NT) users.

A disk image is a verbatim copy of an SMS/QDOS disk stored in a hard disk file. All operations that can be performed on an actual floppy can also be performed on the disk image. The disk image can be transferred back to floppy disk, thus making multiple changes to a disk much faster. Due to restrictions in my current DOS compiler and the Win95 floppy disk driver, Win95 long file names can only be used with disk images and not floppy disks. Long file names are supported for both disk images and floppies under Unix, NT and OS/2.

Disk images are created under DOS using the supplied 'qdc' (Quintessential Disk Cloner) program as follows:

QDC

qdc -d dtest.dsk	Creates a blank hard disk file 'dtest.dsk', ready for initialising as a SMS/QDOS DD disk image (720Kb).
qdc -h HTest.dsk	Creates a blank hard disk file 'HTest.dsk', ready for initialising as a SMS/QDOS HD disk image (1440Kb).
qdc a: atest.dsk	Creates a hard disk image file from the SMS/QDOS format floppy disk in drive 'A'.
qdc btest.dsk b:	Copies the hard disk image file to the floppy disk in drive 'B'. The disk must be formatted (under either DOS, Unix or SMS/QDOS). Any data on the disk will be overwritten.

Notes:

a. disk images created with 'qdc' -d and -h options must be initialised (qltools -f) before they can be used

e.g. qltools dtest.dsk -fd "DD Image"
qltools HTest.dsk -fh "HD Image"

b. 'qdc' does not check that disk image file sizes and floppy disks are of compatible sizes; you do that !

Using Long Files Names

'qltools' v2.4 supports long files to both disk images and floppy disks when run under Linux, (Unix), Windows NT and OS/2 operating systems.

The MSDOS version of qltools only supports MSDOS 8.3 names, which is limiting, particularly for sub-directories. If you are running Windows 95, you can use 'qltools' and long files TO DISK IMAGES ONLY by running the NT version of the program.

Note that the NT version of 'qltools' will not access floppy disks when run under Win95 due to the emasculated drivers/Win32 API in Win95.

Using 'qltools'

qltools is invoked as:

qltools dev_name option [files]

where dev_name is either a device name or the name of a disk image file.

Device Names

The device names supported are:

MSDOS, OS/2:

a:
b:

NT

\\.\a:
\\.\b:

Linux

/dev/fd0H720
/dev/fd0H1440

or as a shorthand

fd=/dev/fd0H720
hd=/dev/fd0H1440
export fd
export hd
qltools $hd ..etc..

Other Unix

as documented for your system

Options

The following options are supported.

-d	list directory
-s	list short directory
-i	list info
-m	list disk map
-c	list conversion table of sectors
-w <name>	write <name> file to disk, query overwrite
-W <name>	write <name> file to disk, no query
-x <name> <size>	make <name> executable with dataspace=size
-r <name>	remove file <name>
-n <name>	copy file <name> to stdout
-fxx <name>	format as xx=hd|dd disk with label <name>
-l	list names as written
-uN	dumps cluster N (1536 bytes) to stdout in ASCII format.
-UN	dumps cluster N (1536 bytes) to stdout as raw (binary) data.
-M <dirname>	creates level 2 directory <dirname>

The following descriptions assume a Linux implementation, substitute a: (or \\.\a:) for DOS/OS2 or NT disks or a disk image name. The Linux shell prompt in the examples is "bash#".

Option -d

Lists a QDOS directory, in the following format: (Note comment markers in [square brackets]).

```
bash# qltools $fd -d

qltools 24                                          [1]
1086/1440 sectors.                                  [2]

copyright                              1389 18/09/95 17:06:54 v0      [3]
dterm                            (dir) 64                             [4]
dterm_x_dat                            4839 18/09/95 17:06:54 v0
dtest                            (dir) 384
dtest_dsub                       (dir) 128
dtest_dsub_a_bas                        177 18/09/95 17:06:56 v0
dtest_dsub_b_bas                        177 18/09/95 17:06:56 v0
```

```
dtest_x_bas                                177 18/09/95 17:06:56 v0
dtest_x_bat                                177 18/09/95 17:06:57 v0
dtest_x_dat                               4839 18/09/95 17:06:57 v0
dtest_x_yyy                                177 18/09/95 17:06:57 v0
dtest_y_bas                                177 18/09/95 17:06:58 v0
echoOn_bas                                  80 18/09/95 17:06:58 v0
echooff_bas                                 83 18/09/95 17:06:58 v0
exit_bas                                    47 18/09/95 17:06:58 v0
fax                                (dir)   128
fax_bas                                    501 18/09/95 17:06:58 v0
fax_basic                          (dir)   128
fax_basic_this_is_an_extremely_long1       759 18/09/95 17:06:58 v0
fax_basic_what_a_long_one                  759 18/09/95 17:06:59 v0
makefile                                   298 18/09/95 17:06:59 v0
qltools_c                                34641 18/09/95 17:06:59 v0
testsub                            (dir)   64
testsub_subfile                    (dir)   64
testsub_subfile_test                      4839 18/09/95 17:07:04 v0
unzip                              E      91400 18/09/95 17:07:05 v0     (40268)
```

Notes:

[1] The volume label for the disk, set the SMS/QDOS format or qltools -f option.

[2] The used and good sectors

[3] A file listing. This consists of

name
type (blank if zero)
- E - executable
- R - relocatable
- (dir) - level 2 directory
- (NNN) - unrecognised type NNN

file size in bytes
date of last update
time of last update
version number (I count from 0).

(data space) - for type 'E'

[4] Level 2 directories are marked (dir) and show the size.

Note that the above selection was compiled to test various options of qltools and InfoZIP and is unrepresentative of anything else !

Option -s

Lists a directory in short format, note that directories are listed to, this is because qltools can will copy this to native format in such a way that they can be re-created on the QDOS device on re-copy. This is described in the '-n' and '-w/-W' options.

```
bash# qltools $fd –s

copyright
dterm                                      [1]
dterm_x_dat
dtest
```

```
dtest_dsub
dtest_dsub_a_bas
dtest_dsub_b_bas
dtest_x_bas
dtest_x_bat
dtest_x_dat
dtest_x_yyy
dtest_y_bas
echoOn_bas
echooff_bas
exit_bas
fax
fax_bas
fax_basic
fax_basic_this_is_an_extremely_long1
fax_basic_what_a_long_one
makefile
qltools_c
testsub
testsub_subfile                              [2]
testsub_subfile_test
unzip
```

Notes:

[1] This is the name of a directory
[2] So is this !

Option -i

This lists information about the disk.

```
bash# qltools $fd -i

Disk ID          : QL5A
Disk Label       : qltools 24
sectors per track: 9
sectors per cyl. : 18
number of cylind.: 80
allocation block : 3
sector offset/cyl: 5
random           : 6ed6
Updates          : 27
free sectors     : 1086
good sectors     : 1440
total sectors    : 1440
directory is     : 1 sectors and 256 bytes

logical-to-physical sector mapping table:

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

physical-to-logical sector mapping table:

0 6 c 1 7 d 2 8 e 3 9 f 4 a 10 5 b 11
```

'qltools' write a random number when it initialises a disk (-f), and updates the update field when a file is updated or deleted.

Option -m

This lists the disk map. To save space, I'll only illustrate a small portion of it.

```
bash# qltools $fd -m | more

block   file    pos

0       3968    0       (f80, 000) map
1       0       0       (000, 000) directory
2       1       0       (001, 000) copyright
3       2       0       (002, 000) dterm
4       2052    0       (804, 000)
5       2052    1       (804, 001)
6       2052    2       (804, 002)
7       2052    3       (804, 003)
8       3       0       (003, 000) dtest
9       2057    0       (809, 000)
10      2058    0       (80a, 000)
11      2059    0       (80b, 000)
12      2060    0       (80c, 000)
13      2061    0       (80d, 000)
14      2062    0       (80e, 000)
15      2062    1       (80e, 001)
16      2062    2       (80e, 002)
17      2062    3       (80e, 003)
18      2066    0       (812, 000)
19      2067    0       (813, 000)
20      4       0       (004, 000) echoOn_bas
21      5       0       (005, 000) echooff_bas
22      6       0       (006, 000) exit_bas
23      7       0       (007, 000) fax
24      2072    0       (818, 000)
25      2073    0       (819, 000)
26      2074    0       (81a, 000)
27      2075    0       (81b, 000)
28      8       0       (008, 000) makefile
29      9       0       (009, 000) qltools_c
30      9       1       (009, 001) qltools_c
31      9       2       (009, 002) qltools_c
32      9       3       (009, 003) qltools_c
33      9       4       (009, 004) qltools_c
34      9       5       (009, 005) qltools_c
--More--
```

The file numbers OR'd with 0x800 are files in level two sub-directories. You can follow the directory threads by dumping (-u) the directory sectors, for example, -u8 to dump the cluster containing the 'dtest' sub-directory. Note that the file numbers recorded in the sub-directory records are one greater that the values in the map.

For example, the entry in dtest for dtest_x_bas has the file number as 0x80d in the directory record; in the map this corresponds to 0x80c, started in cluster 12. This is shown in the -u example.

Option -c

The -c option lists the conversion table converting QDOS logical sector (clusters of 3x512 byte disk blocks) to physical address. Note that the physical sector numbers (as per the SMS/QDOS manual) start at 0, when doing direct sector addressing, sectors start at 1. The unix_dev value can be used (*512) as input to lseek(2).

```
bash# qltools $fd -c | more
```

CONVERSION TABLE

logic	track	side	sector	unix_dev
0	0	0	0	0
1	0	0	3	3
2	0	0	6	6
3	0	1	0	9
4	0	1	3	12
5	0	1	6	15
6	0	0	1	1
7	0	0	4	4
8	0	0	7	7
9	0	1	1	10
10	0	1	4	13
11	0	1	7	16
12	0	0	2	2
13	0	0	5	5
14	0	0	8	8
15	0	1	2	11

Option -x

Sets the data space for executable (type 1) files. Note that 'qltools' automatically records dataspace sizes for executable files created by the Lux68 and XTc68 cross compilers and programs copied from SMS/QDOS disks.

For example, if we had not recorded the dataspace size for 'unzip', then it could be set by:

```
qltools $fd -x unzip 40268
or
qltools a: -x unzip 40268          DOS/OS2
qltools \\.\a: unzip 40268         NT
```

Option -r

This option deletes files. Sub-directory files can only be deleted when the directory is empty.

```
qltools $fd -r unzip               # delete unzip program

qltools $fd -r dtest               # fails !
```

Giving the message:

Directory must be empty to delete (8)

The number in brackets (8) is the number of files in the directory (and its sub-directories)

Option -n

The -n option writes files to stdout (standard output). This may be redirected using the '>' notation. For DOS and friends the file is 'binary' (i.e. no end of line translation). If the file is a subdirectory, then a 'special' file is written that is 16 bytes long and contains the text

```
qltools:type255
```

followed by a LF character. For example, the following 'Perl' script (qlcp.pl) copies all the files off our example floppy into the current (linux) directory.

```
#!/usr/bin/perl
# Perl script to copy all files from a QL DD disk under Linux
# _ in QDOS file names are changed to .
# usage : qlcp device

$dev = $ARGV[0];

open (QL, "qltools $dev -s |");
while (<QL>)
{
      chomp;
      $unx = $_;
      $unx =~ s/_/\./g;
      system "qltools $dev $_ >$unx";
}
close (QL);
```

Translating the SMS/QDOS '_' to '.'.

Listing the linux directory gives:

```
bash# ls -l
total 163
-rw-r--r--   1 root     root         1389 Sep 19 22:41 copyright
-rw-r--r--   1 root     root           16 Sep 19 22:41 dterm
-rw-r--r--   1 root     root         4839 Sep 19 22:41 dterm.x.dat
-rw-r--r--   1 root     root           16 Sep 19 22:41 dtest
-rw-r--r--   1 root     root           16 Sep 19 22:41 dtest.dsub
-rw-r--r--   1 root     root          177 Sep 19 22:41 dtest.dsub.a.bas
-rw-r--r--   1 root     root          177 Sep 19 22:41 dtest.dsub.b.bas
-rw-r--r--   1 root     root          177 Sep 19 22:41 dtest.x.bas
-rw-r--r--   1 root     root          177 Sep 19 22:41 dtest.x.bat
-rw-r--r--   1 root     root         4839 Sep 19 22:41 dtest.x.dat
-rw-r--r--   1 root     root          177 Sep 19 22:41 dtest.x.yyy
-rw-r--r--   1 root     root          177 Sep 19 22:41 dtest.y.bas
-rw-r--r--   1 root     root           80 Sep 19 22:41 echoOn.bas
-rw-r--r--   1 root     root           83 Sep 19 22:41 echooff.bas
-rw-r--r--   1 root     root           47 Sep 19 22:41 exit.bas
-rw-r--r--   1 root     root           16 Sep 19 22:41 fax
-rw-r--r--   1 root     root          501 Sep 19 22:41 fax.bas
-rw-r--r--   1 root     root           16 Sep 19 22:41 fax.basic
-rw-r--r--   1 root     root          759 Sep 19 22:41
                                          fax.basic.this.is.an.extremely.long1
-rw-r--r--   1 root     root          759 Sep 19 22:41 fax.basic.what.a.long.one
-rw-r--r--   1 root     root          298 Sep 19 22:41 makefile
-rw-r--r--   1 root     root        34641 Sep 19 22:41 qltools.c
-rw-r--r--   1 root     root           16 Sep 19 22:41 testsub
-rw-r--r--   1 root     root           16 Sep 19 22:41 testsub.subfile
-rw-r--r--   1 root     root         4839 Sep 19 22:41 testsub.subfile.test
-rw-r--r--   1 root     root        91408 Sep 19 22:41 unzip
```

Note the 16 byte directory markers.

If this directory tree is then copied back to an SMS/QDOS disk or disk image.

```
bash# dd of=../demo.dsk if=/dev/zero count=1440 bs=512        # create blank
bash# qltools ../demo.dsk -fd Qlone                           # initialise
bash# qltools ../demo.dsk -w *                                # populate
```

The first bit of the directory looks like:

```
bash# qltools ../demo.dsk -d

Qlone
1086/1440 sectors.

copyright                                1389 19/09/95 22:53:16 v0
dterm                                (dir) 64
dterm_x_dat                              4839 19/09/95 22:53:16 v0
```

Note that the 'dterm' level 2 directory has been automagically created.

Note also that if no other options are given, then the -n option can be omitted.

```
qltools a: test_bas >test.bas      ; for DOSsers.


qltools $fd qltools_c              # == cat qltools_c
```

Option -f

This option initialises a disk or disk image. A floppy disk must have been low-level formatted first using the SMS/QDOS 'format' or the DOS/NT/OS2 'format' command, or the equivalent for Unix (Linux 'fdformat' command). As qltools always records the number of good sectors as the maximum the media supports, I recommend that you always verify foreign formats and only use known good disks.

```
bash# fdformat $fd

Double-sided, 80 tracks, 9 sec/track. Total capacity 720 kB.
Formatting ... done
Verifying ... done
bash# qltools $fd -fd 'SMS/QDOS !'
bash# qltools $fd -i
Disk ID          : QL5A
Disk Label       : SMS/QDOS !
sectors per track: 9
sectors per cyl. : 18
number of cylind.: 80
allocation block : 3
sector offset/cyl: 5
random           : 15da
Updates          : 1
free sectors     : 1434
good sectors     : 1440
total sectors    : 1440
directory is     : 0 sectors and 64 bytes

logical-to-physical sector mapping table:

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

physical-to-logical sector mapping table:

0 6 c 1 7 d 2 8 e 3 9 f 4 a 10 5 b 11
```

The '-f' must be followed IMMEDIATELY by 'd' for DD disks or 'h' for HD disks, and the media name.

Option -u, -U

The -u command dumps a logical 'sector' (a cluster of three physical sectors) to stdout. Given the example for '-m' (map info), we saw that the 'dtest' directory was in logical sector 8. Dumping logical sector 8 gives.

```
bash# qltools $fd -u8 | more

000 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
010 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
020 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
030 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
040 :  00 00 00 c0 00 ff 00 00 00 00 00 00 00 00 00 0a  ................
050 :  64 74 65 73 74 5f 64 73 75 62 00 00 00 00 00 00  dtest_dsub......
060 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
070 :  00 00 00 00 41 4b 19 b0 00 00 08 0a 41 4b 19 af  ....AK......AK..
080 :  00 00 00 f1 00 00 00 00 00 00 00 00 00 00 00 0b  ................
090 :  64 74 65 73 74 5f 78 5f 62 61 73 00 00 00 00 00  dtest_x_bas.....
0a0 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0b0 :  00 00 00 00 41 4b 19 b0 00 00 08 0d 41 4b 19 b0  ....AK......AK..
0c0 :  00 00 00 f1 00 00 00 00 00 00 00 00 00 00 00 0b  ................
0d0 :  64 74 65 73 74 5f 78 5f 62 61 74 00 00 00 00 00  dtest_x_bat.....
0e0 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0f0 :  00 00 00 00 41 4b 19 b1 00 00 08 0e 41 4b 19 b1  ....AK......AK..
(and more)
```

The first column is offset (hex), then the hex data followed by ASCII representation of printable characters.

The file dtest_x_bas has a file number recorded [from *(short *)0x0ba] as 080d (hex). Subtract 0x801 as this is a sub-directory record, and the logical cluster number is 0xc (i.e. 12), and the file number in the map will be 0x80c (see -m example again).

We could now dump out this file (from logical sector 12) as

```
bash# qltools $fd -u12 | more

000 :  00 00 00 f1 00 00 00 00 00 00 00 00 00 00 00 0b  ................
010 :  64 74 65 73 74 5f 78 5f 62 61 73 00 00 00 00 00  dtest_x_bas.....
020 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
030 :  00 00 00 00 41 4b 19 b0 00 00 08 0d 41 4b 19 b0  ....AK......AK..
040 :  31 30 20 43 4c 49 45 4e 54 20 27 71 74 70 69 27  10 CLIENT 'qtpi'
050 :  0a 32 30 20 52 45 51 55 45 53 54 20 27 71 74 70  .20 REQUEST 'qtp
060 :  69 27 2c 27 77 61 6b 65 27 0a 33 30 20 52 45 4d  i','wake'.30 REM
070 :  61 72 6b 20 52 45 51 55 45 53 54 20 27 71 74 70  ark REQUEST 'qtp
080 :  69 27 2c 27 64 69 61 6c 20 6a 72 68 27 2c 61 24  i','dial jrh',a$
090 :  0a 33 33 20 52 45 51 55 45 53 54 20 27 71 74 70  .33 REQUEST 'qtp
0a0 :  69 27 2c 27 73 70 61 77 6e 20 66 6c 70 31 5f 6a  i','spawn flp1_j
0b0 :  64 27 2c 61 24 0a 33 35 20 42 45 45 50 20 32 30  d',a$.35 BEEP 20
0c0 :  30 30 2c 32 30 20 3a 20 50 52 49 4e 54 20 27 73  00,20 : PRINT 's
0d0 :  70 61 77 6e 20 3d 20 27 3b 61 24 0a 34 30 20 46  pawn = ';a$.40 F
0e0 :  52 45 45 43 4c 49 45 4e 54 20 27 71 74 70 69 27  REECLIENT 'qtpi'
0f0 :  0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

If this file had been accidentally deleted, we could have 'rescued' it using the -U command to dump the logical sectors to stdout. Had the file been more than one logical sector, this would have to have been done multiple times, getting the file sector numbers from the map (80c,000), (80c,001) etc.

```
bash# qltools $fd -U12

ñ
AK°10 CLIENT 'qtpi'
20 REQUEST 'qtpi','wake'
30 REMark REQUEST 'qtpi','dial jrh',a$
```

```
33 REQUEST 'qtpi','spawn flp1_jd',a$
35 BEEP 2000,20 : PRINT 'spawn = ';a$
40 FREECLIENT 'qtpi'
```

Note that there is some rubbish at the beginning of the dumped file as 'qltools' dumps complete logical sectors of 1536 bytes.

As a further example, the 'dterm' directory is is logical sector 3, dumping this reveals the file 'dterm_x_dat' starts in logical sector 4; from the map (-m) it can be seen that this file occupies logical sectors 4,5,6,7 (file number 0x804).

Options -w, -W and -l

These options control the writing of files to the SMS/QDOS disk or disk image. 'qltools' converts any '.'s in the native file names to '_' when the files are written to the SMS/QDOS disk.

The '-w' option queries for file overwrites.

```
bash# qltools demo.dsk -w qltools.c

file qltools_c exists, overwrite [Y/N/A/Q] : N
```

The '-W' option copies file without prompting for overwrite.
The '-l' option lists (to stderr) file names as they are copied.

Wildcards are supported on all platforms for write commands.

```
bash# qltools $fd -l -W *
```

or (just in case we have an OS/2 aficionado)

```
[C:\] qltools a: -l -W *.*
```

qltools strips any Unix or DOS-ish directory names from the file before it is written to the SMS/QDOS disk.

i.e.

~/test/demofile.txt -----> demofile_txt

Option -M

The -M command makes level 2 sub-directories. This works like the level 2 MAKE_DIR. Any files that 'match' will be moved into the sub-directory.

Given a file test_file (or even test.file) (on the foreign disk), then

```
bash# qltools $fd -w test_file      # write file
bash# qltools $fd -d                # check it
test_file                                 293 19/09/95 23:35:20 v0
bash# qltools $fd -M test           # create dir
bash# qltools $fd -d                # check it
test                                (dir) 64
test_file                                 293 19/09/95 23:35:20 v0
```

Just for fun, we could have done.

```
bash# qltools $fd -M test           # create dir
bash# qltools $fd -d                # check it
test                                (dir) 0
bash# qltools $fd -w test_file      # write file
```

```
bash# qltools $fd -d                 # check it
test                                 (dir)  64
test_file                                  293 19/09/95 23:35:20 v0
```

i.e. Exactly the same.

As many qltools commands can follow on the command line, we could also have done:

```
bash# qltools $fd -w test_file -M test
```
or
```
bash# qltools $fd -M test -w test_file
```

The effect of the two command is essentially the same.

And if the 'special' file 'test' contained the magic 16 characters "qltools:type255\n", then

```
bash# qltools $fd -w test -w test_file
```
and
```
bash# qltools $fd -w test_file -w test
```

Would also have the same effect.

and to delete it again !!!
```
bash# qltools demo.dsk -r test_file -r test
```
BUT NOT
```
bash# qltools demo.dsk -r test -r test_file      # Why not ??
```
Directory must be empty to delete (1) # Of course !

Distribution

qltools is copyright (c) Giuseppe Zanetti and others. Please see Giuseppe's file 'copyright'.

The level 2 and system specific bits are (c) Jonathan Hudson

The software is freely distributable.

Compiling

qltools 2.4 compiles without errors or warnings using gcc 2.6.3 on Linux and Watcom C/C++ 10.0 on DOS/NT/OS2. It also compiles on DOS using Borland C 3.1.

Note that the 'wildargv' code linked into the distributed DOS/NT/OS2 binaries is (c) Watcom and cannot be distributed. Similar code is available for other compilers.

Recent versions (2.12 etc) are compiled with

Linux	egcs (gcc 2.91 or later)
DOS/Win9x	gcc 2.72 ('go32')
NT	gcc 2.95

Keeper of the code

The qltools code is currently maintained by:

// Jonathan R Hudson, PO Box 2272, Ruwi 112, Sultanate of Oman.
// Tel/Fax : +968 699407 (24 hours)
// BBS : +968 699407 (18:00-03:00 GMT)
Jonathan Hudson (jrhudson@bigfoot.com)

List Of Updates Since v2.4

qltools 2.7:

1. Internal (code) tidy up
2. Compiled with highest optimisation levels
3. displays file size on -l -w (or -l -W)

qltools 2.6 :

1. Adds more sensible error messages when a sector is unreadable
2. Correctly handles 'XTcc' data blocks on type 1 QDOS files.

v2.5 of qltools:

1. Fixed corruption of files written to DOS_LIKE systems (MSDOS,OS2,NT) caused by the output files being opened in text mode.
2. Writes zero bytes to the first 18K of the disk in an attempt to prevent MSDOS from finding a secondary FAT.

The manual is unchanged from "docs/qltools24.readme".

The source code for qltools for all supported platforms (Unix, NT/Win95, OS2, MSDOS) is available in the linux archive. (qltools27.tgz)

The most recent DOS/OS2/Win95/WinNT code is cross compiled with gcc. This allows long file names on W95 floppies. If you're running it under DOS, you need a DPMI server. The enclosed CWSCPMI.EXE will suffice; it needs to be on your PATH. Windows and OS2 provide DPMI services.

Jonathan Hudson
jrhudson@bigfoot.com

v2.6 03/12/95
v2.7 14/07/96