

Tools for putting files into a .d64 image

By Frantic.

When coding on a PC or Mac, you probably use some editor which is capable of executing other programs by pressing keys like CTRL+F1 and so on (after some configuration). Personally I use this to call [makefiles](#), which in turn calls all the crunchers, assemblers and whatnot, but you could also call the programs directly from your editor.

There are a few tools out there which allows you to put the files of your coding project onto a .d64 image, before transferring it to a C64 or executing it in an emulator.

C1541

If you have VICE installed, then you already have this tool on your computer. It allows you to do various operations on .d64 images, such as creating them and writing files to them. You can do several operations in one single call to c1541, as seen in this example:

```
c1541 -format diskname,id d64 my_diskimage.d64 -attach my_diskimage.d64 -write my_program.prg myprog
```

This will create a new .d64 image and write the file "my_program.prg" onto the .d64 image with the name "myprog". The last parameter could be left out, and the file would be named "my_program.prg" on the disk image too. The available options for c1541 looks like this:

```
@ [<command>]
? [<command>]
attach <diskimage> [<unit>]
block <track> <sector> <disp> [<drive>]
copy <source1> [<source2> ... <sourceN>] <destination>
delete <file1> [<file2> ... <fileN>]
dir [<pattern>]
exit
extract [<unit>]
format <diskname,id> [<type> <imagename>] [<unit>]
geosread <source> [<destination>]
geoswrite <source>
help [<command>]
info [<unit>]
list [<pattern>]
name <diskname>[,<id>] <unit>
p00save <enable> [<unit>]
quit
read <source> [<destination>]
rename <oldname> <newname>
show [copying | warranty]
tape <t64name> [<file1> ... <fileN>]
```

```
unit <number>
unlynx <lynxname> [<unit>]
validate [<unit>]
write <source> [<destination>]
zcreate <d64name> <zipname> [<label,id>]
```

cc1541

cc1541 works fine for most purposes, but some of the better coders around says it is buggy, and thus there is also [cc1541](#). cc1541 also supports some more advanced options which gives you better control over how the files are actually laid out onto the image. The available options go like this:

```
*** This is cc1541 version 3.1 built on Sep  1 2019 ***
```

```
Usage: cc1541 -niwfoVTP0lBMmdtuxFsSeErbc45gqvh image.[d64|d71|d81]
```

```
-n diskname  Disk name, default='cc1541'.
-i id        Disk ID, default='00 2a'.
-w localname Write local file to disk, if filename is not set then the
             local name is used. After file written, the filename is unset.
-f filename  Use filename as name when writing next file, use prefix # to
             include arbitrary PETSCII characters (e.g. -f "START#a0,8,1").
-o          Do not overwrite if file with same name exists already.
-V         Do not modify image unless it is in valid CBM DOS format.
-T filetype  Filetype for next file, allowed parameters are PRG, SEQ, USR,
REL
             and DEL. For DEL, the input file is ignored. Default is PRG.
-P         Set write protect flag for next file.
-O         Set open flag for next file.
-l filename  Write loop file (an additional dir entry) to existing file to
             disk, set filename with -f.
-B numblocks Write the given value as file size in blocks to the directory
for
             the next file.
-M numchars  Hash computation maximum filename length, this must
             match loader option FILENAME_MAXLENGTH in Krill's Loader.
             Default is 16.
-m          Ignore filename hash collisions, without this switch a
collision
             results in an error.
-d track    Maintain a shadow directory (copy of the actual directory
without
             a valid BAM).
-t         Use dirtrack to also store files (makes -x useless) (default
no).
-u numblocks When using -t, amount of dir blocks to leave free (default=2).
```

```

-x          Don't split files over dirtrack hole (default split files).
-F          Next file first sector on a new track (default=3).
           Any negative value assumes aligned tracks and uses current
           sector + interleave. After each file, the value falls back to
the
           default. Not applicable for D81.
-S value   Default sector interleave, default=10.
           At track end, reduces this by 1 to accomodate large tail gap.
           If negative, no special treatment of tail gap. Not applicable
for
           D81.
-s value   Next file sector interleave, valid after each file.
           At track end, reduces this by 1 to accomodate large tail gap.
           If negative, no special treatment of tail gap.
           The interleave value falls back to the default value set by -S
D81.
-e          Start next file on an empty track (default start sector is
           current sector plus interleave).
-E          Try to fit file on a single track.
-r track   Restrict next file blocks to the specified track or higher.
-b sector  Set next file beginning sector to the specified value.
           Not applicable for D81.
-c          Save next file cluster-optimized (d71 only).
-4         Use tracks 35-40 with SPEED DOS BAM formatting.
-5         Use tracks 35-40 with DOLPHIN DOS BAM formatting.
-g filename Write additional g64 output file with given name.
-q         Be quiet.
-v         Be verbose.
-h         Print this commandline help.

```

mkd64

Another flexible alternative is [mkd64](#) which takes a modular approach to creating the disk files, so you don't have to follow the default 1541 format at all. In fact, creation of a directory and block allocation map is taken care of by a loadable module named "cbmdos". It also includes all sorts of tricks and tweaks possible with the 1541 format as well as a module providing some pre-made "separator" directory entries using graphical characters. The help pages for mkd64 itself and the cbmdos module look like this:

mkd64 1.4b help

```

mkd64 supports four types of options. Single options trigger some immediate
action, see below. Global options affect the whole disk image generation,
module options are passed just to the last loaded module and file options
control single files written to the image.
Global and module options must come before file options on the command line.

```

Modules can provide their own global and file options, check their help messages (-h MODULE) for reference.

SINGLE options (must be the only option to mkd64):

- h [MODULE] Show this help message or, if given, the help message for the module {MODULE}, and exit.
- V [MODULE] Show version info and exit. If {MODULE} is given, version info for that module is shown instead.
- C OPTFILE Read options from file {OPTFILE} instead of the command line. The file has the same format as the normal command line and the following rules:
 - Strings containing whitespace are escaped using quotes or doublequotes (' or ")
 - The backslash (\) has no special meaning at all
 - Newlines are just normal whitespace and thus ignored
- M Display all available modules and exit.

GLOBAL options:

- m MODULE Activate module {MODULE}. Modules are searched for in the directory of the mkd64 executable for a portable build of mkd64, or in the dedicated module directory (typically /usr/lib/mkd64) for an installable build. Any options following -m are treated as module options to this module, as long as there is no other -m option or a -g option to get back to global scope or a -f option to switch to file scope.
- o D64FILE Write generated disk image to {D64FILE}. This option must be given to actually write something.
- M MAPFILE Write file map of the generated disk image to MAPFILE. The map file format is one line per file on disk:
[startTrack];[startSector];[filename]
- P [MAXPASSES] Allow up to {MAXPASSES} passes, automatically applying options suggested by modules. The default is only one pass if this option is not given or up to 5 passes if it is given without an argument.

MODULE options:

- g Go back to global scope after loading a module. Please see the module documentation or help text (-h MODULE) for options available with specific modules.

FILE options:

- f [FILENAME] Start a new file. {FILENAME} is the name on your PC. It can be omitted for special empty files.
- t TRACK Set fixed start track for current file.
- s SECTOR Set fixed start sector for current file.
- i INTERLEAVE Set sector interleave for current file.
- w Write current file to disk image.

Note that filesystem elements (like the original cbmdos directory and BAM) are implemented by modules. They can provide a sensible default value for sector interleave. A default allocation strategy is built in and determines start track and sector automatically if not given, modules can install their own strategy.

```
mkd64 1.4b help
```

```
* Module `cbmdos':
```

cbmdos implements the default directory and BAM scheme of a 1541 floppy. Interleave is initially set to 10 for every file (cbmdos standard). The following options are recognized:

- d DISKNAME The name of the disk, defaults to an empty name.
- i DISKID The ID of the disk, defaults to two random characters.
this can be up to 5 characters long, in this case it will
overwrite the default `DOS type' string (`2A').
- R DIRBLOCKS reserve {DIRBLOCKS} blocks for the directory. The default
value is 18, which is exactly the whole track #18.
- I INTERLV Set the directory interleave to {INTERLV}. The default value
for directory interleave is 3.
- D DOSVER Set the dos version byte to {DOSVER}, given in hexadecimal.
The default value is (hex) 41. this can be used for soft
write protection, the original floppy will refuse any write
attempts if this value is changed.
- A Allocate all blocks in the BAM.
- 0 Set available blocks to 0 in BAM, but still write flags for
individual sectors.

```
* File options:
```

- n [FILENAME] Activates cbmdos directory entry for the current file. If
{FILENAME} is given, it is used for the cbmdos directory.
- T FILETYPE One of `p', `s', `u', `r' or `d' (for PRG, SEQ, USR, REL or
DEL), defaults to PRG.
- P Make the file write-protected.
- S BLOCKSIZE Force the size written to the directory to be {BLOCKSIZE}.

Example usage from an own project:

```
mkd64 -odisks/demo.d64 \
      -mcbmdos -d'C=64 WORKBENCH' -i'AMIGA' -R1 -Da0 -0 \
      -mseparator \
      -fdemo_bootloader -proundtop -S1 -w \
\
      -fdemo_kickstart -n'DEMO: AMIGADOS' -pfr -t19 -s0 -TU -S0 -i15 -w \
      -fdemo_amigados -pfrmid -TU -S0 -i15 -w \
      -fdemo_music -n'RELEASE 1.09A4' -pfr -TU -S0 -i15 -w \
      -f -n' 2013/12/15 ' -pfr -TD -w \
      -f -n' BY ZIRIAS ' -pfr -TD -w \
```

- f

-proundbot

-TD

-w

For Linux/Windows binaries or Debian packages, check the [release branch](#), folders “win32bin”, “lin32bin”, “deb32bin”, “deb64bin”. *[Section added by Felix Palmen (Zirias)]*

k2xtools

In the [k2xtools](#) collection there are also two tools called “mkd64” (to create d64 images) and “copy2d64” (to copy files onto the d64 images). I don't have them installed on my computer at the moment, so I cannot provide a list of the available options, but at least I remember that these tools worked just fine when I used them a few years ago.

VICE

At least in the Mac version of VICE you can also enable an option which makes sure that .prg files are automatically put onto a temporary .d64 file before they are executed. This may be handy if you are testing a program which only consists of one single .prg file, and no other loading of files from within your .prg.

From: <https://codebase64.org/> - **Codebase 64 wiki**

Permanent link: https://codebase64.org/doku.php?id=base:tools_for_putting_files_into_a_d64_image

Last update: **2019-09-05 11:51**

