

# DSFTP: FTP server for the Nintendo DS

© 2006 Björn Giesler <bjoern@giesler.de>

This is a full-fledged FTP server and software component for the NDS, with the additional ability to boot .nds files directly from the FTP prompt (currently only tested on SuperCard CF and DS, but should work on most).

DSFTP is a stand-alone program that you can use to run a FTP server on your DS. It is also a **software component** that makes it easy for you to **add an FTP server to your own program** with a couple lines of code. With this, you can speed up the build process radically, because you can upload the new version of your program **to your own running program**, and reboot it immediately. This can even be automated via the command line. No swapping of flash cards ever again! Please refer to the Development section of this document or the reference documentation for a quick introduction.

## Changes

### Version 2.6:

- Fixed hangs on some hardware combinations
- Fixed bug which caused directory contents not to be reported correctly
- Introduced new config option **transferblocksize**

### Version 2.5:

- Upgraded to devkitpro 21, so linking to libDSFTP should work now
- Uses DSWifi 0.34, so people with newer DS Lites should have no problems
- Uses libfat instead of gba\_nds\_fat, which should be more stable
- Uses DSGUI for graphics and file-system interface

### Version 2.4:

The FAT driver has been upgraded to chishm's DLDI file system driver interface. The precompiled version comes with Supercard CF driver installed. **No other interfaces will be supported any more.** If you have a different flash card, please patch the DSFTP binary yourself.

### Version 2.3:

- UNZP added (submitted by Bertrand Augereau)
- DS->PC file transfer speeded up a lot (patch to DSWifi submitted by masscat)
- Linked with Devkit R18 again (fixes some showstopper bugs)
- More bugfixes

### Version 2.2:

- Glaring bug fixed that caused GBAMP not to work anymore
- Booting verified on SuperCard and GBAMP now
- CDUP command supported
- More bugfixes

### Version 2.1:

- Unified bootloading should now support most cards
- Auto-generated user & pass if none specified
- More bugfixes

## Using the Stand-alone Server

This section describes how to setup, configure and use the stand-alone DSFTP server.

## Configuration

DSFTP reads its settings from a configuration file, which can exist at `/data/settings/ftp.conf`. No, there is no alternative path and no way to specify one.

**Note:** This file is not strictly necessary; if it isn't found, DSFTP will create a default user with a random password (which will be shown on the screen). So if you think you're not up to the task of creating a configuration file, don't worry.

Here is an example of a configuration file:

```
motd /ftp/motd.txt
logfile /data/logs/ftp.log
loglevel 4
timeout 60
transferblocksize 4000
portrangestart 9000
portrangeend 9999

screensaver 30
wakeonlog false

user bjoern
pass whatever
```

```

        root /
        home /
        write true
        boot true
    end user

    user anonymous
        root /ftp/anonymous
        write false
    end user

```

Whitespace in the configuration file is ignored (the indents in the example file are purely cosmetic), as are empty lines and lines starting with "#". The following settings are valid:

**motd** FILE

Defines a file that contains the message of the day, which will be printed on logon.

**logfile** FILE

Defines the name of a file to store log messages in.

**loglevel** LEVEL

Only messages with a loglevel below LEVEL will be printed / logged to file. LEVEL ranges from 0 (critical) to 5 (informational).

**masquerade** NAME

Specifies that the server masquerades as the host NAME. This is very useful if the server is behind a NAT gateway, and its actual IP address is not visible to the world. In this case, for clients using PASV instead of EPSV for the passive mode, the reply to PASV will contain the IP address for NAME, not the actual server's IP.

**Note:** Do not use this if you're not behind a NAT gateway and the server is not actually reachable under the given name. It won't work.

**timeout** SECONDS

Specifies that the server will close the data connection after the given period of inactivity. Default is 60 seconds.

**listen** PORT

Tells the server to listen to PORT. If this is not given, the standard FTP port 21 will be used.

**portrange**start NUM / **portrange**end NUM

Specifies what port range to use for passive connections. If these aren't given, the portrange goes from 1024 to INT\_MAX.

**screensaver** SECONDS

Specifies the timeout for the screen saver. Can also be the string "off", which disables the screen saver altogether. Default is 60 seconds.

**wakeonlog** BOOL

If this is "true", the screensaver will wake up on every log entry. Default is true.

**transferblocksize** NUM

Transfer data in NUM-bytes sized chunks. Small values make for slower transfers, large values may cause hangs on some hardware. Minimum 512, maximum 8192, default is 4096.

**user** NAME

Starts a new user block, which must be finished by "end user". Only users named by a user block will be accepted for login.

Exception to this rule: **If no users at all are defined**, the stand-alone server defines a user named "default", with an automatically generated password which is displayed on the log output.

Settings valid in the user block are:

**pass** PASSWORD

Set the password for the current user, stored in cleartext. If the user's name is "anonymous", this will be ignored, and the user will be asked to specify his email address for a password.

**root** DIR

Restrict the user to the directory hierarchy below DIR.

**home** DIR

Upon logon, set the current directory to DIR (relative to the user's root dir).

**write** BOOL

Grant the user write permission (including permission to rename and delete!) for everything below his root dir.

**boot** BOOL

Grant the user permission to boot files and to power down the DS.

## Protocol Additions

DSFTP defines the additional command "BOOT", with the filename to be booted as an argument. Since the "BOOT" command is not in the official protocol, the FTP client doesn't know it. For text-based ftp, the "quote" prefix allows you to send it anyway, like this:

```
quote BOOT /mysdfile.nds
```

Possible reply codes to the BOOT command are 530 (No permission to boot) and 250 (Bootng file). Booting currently only works on Supercard CF. No checks are performed; it is legal to boot a ".txt" file, which will most likely crash your DS. On a successful boot, the data connection to the client is closed. Since booting doesn't return on success, a success message is always printed. If the file to boot could not be found, nothing is reported to the FTP client, but the connection remains intact.

DSFTP defines the additional command "POWR", which can be used to power the DS off. Possible reply codes are 530 (No permission to power off) and 250 (Powering off).

DSFTP defines the additional command "UNZP" to uncompress files compressed with GNU zip. Usage is:

```
quote UNZP /zipfile.gz
```

This will create a file named /zipfile (without the .gz ending). The original, zipped file will not be removed. Possible reply codes are 530 (Unable to open file or error unpacking) and 226 (unpacking succeeded).

## Development

This section describes how you can use libDSFTP to add an FTP server to your own applications. For all of these steps, you can use the included source for DSFTP as an example.

### Preliminaries

Adding an FTP server is very easy. First, you should make sure that the following components are added to your build process. That means their include files must be accessible, and their libraries/object files must be linked with your program.

- gba\_nds\_fat (either chishm's FAT lib or REIN's)
- dswifi
- libDSFTP

Then, you initialize the Wifi and FAT libraries as usual. Please refer to their respective examples for info on how to do it, or to the DSFTP source.

### Creating the server

Next, you create an instance of BFTPServer somewhere in your program, like so:

```
BFTPServer server;
```

If you want to, you can also create an instance of BFTPConfigurator to configure the server. This is not mandatory, but it allows you to configure the FTP server via config file. This would look as follows.

```
BFTPConfigurator configurator(&server);
configurator.configureFromFile("/data/settings/ftp.conf");
```

...or wherever you want the config file to be.

If you **do not** use a BFTPConfigurator, you **must** add users to the FTP server manually. Otherwise, you won't be able to log in. BFTPServer does **not** define any fallback/backdoor users. The simplest way to add users is like this:

```
server.addUser("myusername", "mypassword");
```

Please refer to BFTPServer.h for other arguments to this command.

### Letting it do its work

That's it, your FTP server is ready! Now, whenever your program is idle (e.g. from the main loop), you should let the server do its work, like this:

```
server.handle();
```

### Wrapping it up

And that's all you have to do. All the code inside of a nice little mainloop function would look like this:

```
void mainloop(void)
{
    BFTPServer server;
    BFTPConfigurator configurator(&server);
    configurator.configureFromFile("/data/settings/ftp.conf");
```

```
while(true)
{
    server.handle();
    swiWaitForVBlank();
}
}
```

Please refer to the source code of DSFTP and the header files of libDSFTP for other possible settings and features.

### **Known Bugs and Limitations**

The following bugs and limitations are known at the time of writing:

- Booting only verified to work on Supercard CF and Supercard SD.
- Performance tested using the following clients. Performance with other clients unknown, but should work.
  - command-line FTP on Mac/DOS
  - RBrowser Lite on Mac
  - MacOS X virtual FTP file system
  - Firefox
  - FileZilla
  - FireFTP (Firefox plugin)
  - WS\_FTP

If any bugs pop up, please send me the log output with a full description to [bjoern@giesler.de](mailto:bjoern@giesler.de). Thank you and have fun!