Yaffs FAQs

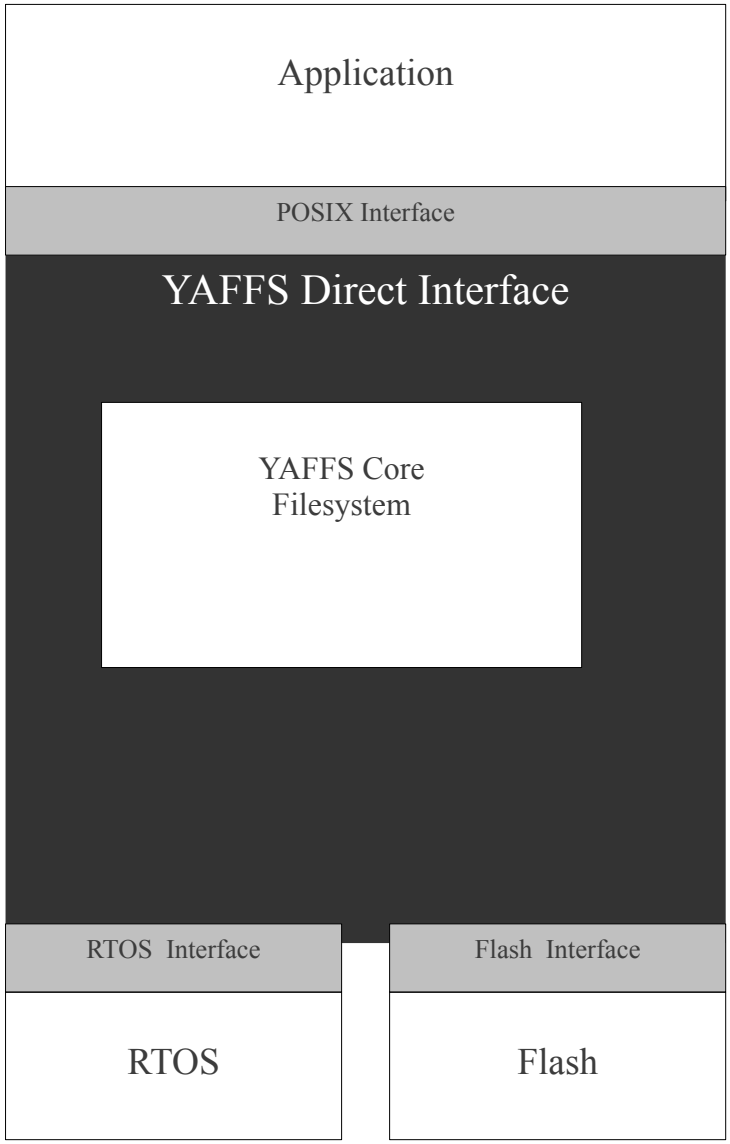
What Operating System (OS) can you use with Yaffs?

Yaffs is agnostic about the OS it is used with; it has been used with many different OSs, including Linux, eCos, Windows CE, ThreadX, pSOS and vXworks. It is POSIX-compliant, so commands in an application calling POSIX functions will usually work.

Windows CE uses the Microsoft file handling model and is not POSIX-like. The difference is handled in the Windows CE wrapper.

Work on closer integration of Yaffs into the Linux kernel has stared in mid 2010. The goal of this effort is to have Yaffs merged into the mainline Linux kernel.

The Yaffs Direct Interface, YDI, is the heart of Yaffs and usually stands between the Application, on one side, and the Flash chips, and any OS, on the other side:-



What CPU can you use with Yaffs?

Yaffs has been used with many 32-bit and 64-bit CPUs including MIPS, 68000, ColdFire, PowerPC, x86 variants and many ARM-cored CPUs. It should work with 16-bit CPUs but has not been tested.

Does Yaffs care Which-Endian the CPU is?

No, Yaffs works with Big-Endian and Little-Endian CPUs

How big can the whole file system be, and How big can individual files be?

Yaffs2, (now the commonly used version because it handles current Flash chips with large page sizes) can handle a File System (FS) of many Gigabytes total storage; some people are using it with 6GB satisfactorily. Individual files are currently limited to 2GB. The basic Yaffs architecture can handle larger single files and some changes would be required to support individual files greater than 2G bytes. There is an upper limit of around 32GB with the present 2k bytes granularity of 'chunks' or page size and the limit of about 16 million chunks per file.

A coarser granularity of larger chunks can probably be obtained, with knock-on effects in other parameters like RAM usage.

How many files can there be?

There is no hard limit. Available flash and RAM will limit this.

What about directories etc?

Yaffs supports many different objects including: directories, regular files, hard links, symbolic links and device nodes.

Yaffs supports long file names, attributes, extended attributes and Unicode.

How much RAM does it need?

This depends particularly on the number of tnodes, which in turn depends on the file size, and is therefore a dynamic variable.

However, Yaffs compares favourably with other similar systems specialised for Flash and includes robust error handling.

How are errors handled?

Error Correction Code (ECC) are used to correct for single-bit errors.

ECC may use built-in code or that provided from other sources (flash controllers, ECC libraries etc).

Yaffs is frequently tested on a "fuzzer" test harness. This randomly corrupts bits of the file system image to check that the file system still mounts and loads correctly.

What about Multi-level cell (MLC) chips?

Because a bit is represented by a smaller amount of charge in the MLC chip compared to a SLC chip, MLC chips are more prone to corruption than SLC.

MLC also has various write restrictions which are obeyed by Yaffs2.

Yaffs2 has some features such as block refreshing to help alleviate the problems associated with MLC.

MLC generally requires multi-bit ECC. Hardware ECC is preferred since it is faster and requires less computation than software multi-bit correction..

How robust is Yaffs ?

Yaffs is designed specifically to take account of manufacturing faults in, and of accumulated wear in, individual cells, and also to handle power failures during use.

Yaffs allocates flash blocks in a way that gives statistical or implicit wear levelling; when a block is freed up it is released into a free pool and the next allocation comes from this free pool. This randomness tends to spread the blocks in use, giving some wear levelling rather than one part of the file always being worked intensively.

Accelerated lifetime testing has been carried out to seek evidence for excessive wear. For example on one professional PDA-style device used by surveyors in the field we determined that extreme lifetime usage would be:

- 20MB per day.
- 200 days per year
- 10 year lifetime
- Total 40Gbytes lifetime write.

We then ran well over 200Gbytes of writing through a specially designed File System Exerciser, a multi-threaded application that writes, reads, verifies, then deletes, repeatedly. There was no sign of a single bit of corruption at the end of the test.

Another test applied randomly timed power failures repeatedly while running test code. Millions of cycles have been executed without any errors being observed.

Yaffs is used (under GPL) in the Android mobile phones sold by Google and others. Extensive use in the field like this helps to ensure its robustness.

How big is its code?

That depends on how much of it you need; it is modular in organisation. The innermost or core part is about 11k lines of code, the the Yaffs Direct Interface giving file handle access and a Posix-like interface is around 2k lines.

Glossary and useful links.

Big-endian and Little-endian	Big-endian is when the most significant data chunk is listed first. Little-endian is when the least significant data chunk is listed first. http://en.wikipedia.org/wiki/Endianness
tnodes	Tnodes are yaffs data structures used to build trees that map from a file to the file data chunks stored in flash. A large file will have multiple data chunks scattered through-out the flash and will need a larger tnode tree. See the HowYaffs Works document for more information.
ECC	Error correction codes, See the Yaffs Error mitigation document.
MLC	Multi-level cell, allows more than one bit state to be stored in a memory cell. MLC Flash often allows four state (ie. 2 bits) to be stored in one cell. http://en.wikipedia.org/wiki/Multi-level_cell