www.yaffs.net

Yaffs Tuning

Charles Manning

2012-07-22

Yaffs has many options for tuning for speed or memory use. This document details them for Yaffs Direct and Linux, covering compile time flags and settings, initialisation parameters, tracing, and state inspection.

Table of Contents

1 Yaffs Direct Configuration....................2
1.1 Initialisation Parameters....................2
1.2 Compile-time settings....................4
2 Tracing mechanism....................5
3 Linux....................6
3.1 Linux compile-time flags....................6
3.2 Linux: /proc/yaffs trace adjustment....................6
3.3 Linux: /sys/ interface....................6
3.4 Linux: /proc/yaffs state dumping....................6
4 Conclusions....................7

1 Yaffs Direct Configuration

Yaffs behaviour is dictated by compile time and run-time flags.

1.1 Initialisation Parameters

Each yaffs device is configured by a yaffs_DeviceParameterStruct structure which is part of the yaffs_DeviceStruct structure. This contains the following fields:

name	An optional name for the device
totalBytesPerChunk	The number of bytes in the data area of the flash page. If inbandTags is not set then this whole area will be used to store data and the tags will be stored in the spare area. If inbandTags is set then part of this area will be used to store tags and the rest will be used to store data.
spareBytesPerChunk	The number of available bytes in the spare area of the flash page. This excludes space used by ECC, bad block markers etc.
startBlock	The first block in the partition to be used. 0 for first block.
endBlock	The last block in the partition to be used.
nReservedBlocks	Number of good blocks blocks reserved for garbage collection etc. Needs to be at least 2, but 5 would be a more typical number.
inbandTags	Yaffs2 only:Flag indicating whether tags should be stored in the data area. If inbandTags is set then the short Op cache must be enabled (ie nShortOpCaches must be non-

	zero.
useNANDECC	Yaffs1 only: Flag indicating whether the driver performs ECC. If this is zero then Yaffs will perform ECC.
noTagsECC	Yaffs2 only: Flag indicating whether the tags have ECC attached to them.
isYaffs2	Flag indicating if this is using the Yaffs2 mechanism. If this is not set then Yaffs1 mode of operation is provided.
nShortOpCaches	Number of chunks to store in the short operation cache. Zero disables caching. Typical value is 10 to 20 or so. Caching is required for inband tags.
emptyLostAndFound	Flag to delete all files in lost and found on mount.
skipCheckpointRead	Yaffs2 only: Flag to skip reading checkpoint on mount. If set then a re-scan is forced.
skipCheckpointWrite	Yaffs2 only: Flag to skip writing checkpoint on sync or unmount.
refreshPeriod	Yaffs2 only: How often Yaffs should do a block refresh. Values less than 10 disable block refreshing. Typical values would be1000.
initialiseNAND	Pointer to function to initialise flash driver.
deinitialiseNAND	Pointer to function to de-initialise flash driver
eraseBlockInNAND	Pointer to function to erase a flash block
writeChunkToNAND	Yaffs1 only: Pointer to function to write a chunk
readChunkFromNAND	Yaffs1 only: Pointer to function to read a chunk.
writeChunkWithTagsToNAND	Yaffs2 only: Pointer to function to write a chunk plus tags
readChunkWithTagsFromNAND	Yaffs2 only: Pointer to function to read a chunk plus tags
markNANDBlockBad	Yaffs2 only: Function to mark a block bad
queryNANDBlock	Yaffs2 only: Function to query a block state
gcControl	Callback function that returns the garbage collector

	control flags. This is optional.
removeObjectCallback	Callback function called when an object is removed. This is set by the wrapper.
markSuperBlockDirty	Callback function that is called when a clean file system is first modified. This is set by the wrapper.
disableSoftDelete	Yaffs1 only. Debug only. Disables soft deletion if non-zero.
useHeaderFileSize	Debug only. Leave as zero
disableLazyLoading	Debug only. Leave as zero.
wideTnodeDisabled	Debug only. Leave as zero.
deferDirectoryUpdate	Used to defer directory updates.

1.2 Compile-time settings

These setting are used to apply compile-time configurations.

CONFIG_YAFFS_ALWAYS_CHECK_CHUNK_ERASED

Normally Yaffs checks that chunks are erased only on the first chunk write to each block. This flag allows you to force all chunks to be checked. This is slower, but is worth using when doing early debugging.

CONFIG_YAFFS_CASE_INSENSITIVE

When selected case insensitive name comparisons are used. Note that this will not magically change Yaffs to be case insensitive in a case sensitive OS. This needs to be set according to the OS policy. Typically this is only used in Windows CE.

CONFIG_YAFFS_DIRECT

Used when compiling Yaffs Direct.

CONFIG_YAFFS_DISABLE_LAZY_LOAD

Lazy loading is used to defer loading up object details until the objects are accessed. This makes mount time faster. This flag can disable lazy loading. Debug only.

CONFIG_YAFFS_ECC_WRONG_ORDER

This flag should only be used with old Linux MTD code which had a bug in the ECC byte ordering.

CONFIG_YAFFS_NO_YAFFS1

Used to say that only Yaffs2 operation should be compiled in. Not currently supported.

CONFIG_YAFFS_SHORT_NAMES_IN_RAM

Set to configure Yaffs to store short object names in RAM. This is faster, but uses up more memory.

CONFIG_YAFFS_UNICODE

Set to configure Yaffs to use Unicode file names (eg. With WinCE).

CONFIG_YAFFS_USE_OWN_SORT

By default, Yaffs uses qsort for sorting blocks at mount scanning time. This is faster. Using this flag forces Yaffs to use a slow internal sort instead. Debug only.

CONFIG_YAFFS_UTIL

Used when compiling Yaffs for utilities.

CONFIG_YAFFS_WINCE

Set when compiling Yaffs for WinCE.

CONFIG_YAFFS_YAFFS2

Set when using Yaffs2 mode. At present this must always be set.

CONFIG_YAFFSFS_PROVIDE_VALUES

Used when compiling Yaffs Direct only.

2 Tracing mechanism

Yaffs has a lot of built in tracing to help debug and integrate. The tracing is controlled by yaffs_traceMask bitmask which allows various sets of tracing to be disabled or enabled at runtime.

The trace bits are specified in the yaffs_trace.h file.

The tracing mechanism has also been 'hijacked' to provide a control mask for extra verification code when the YAFFS_TRACE_VERIFYxxx trace bits are selected.

The tracing mechanism is readily integrated into the system via printf-like statements. Tracing is all done through macros which allow the strings to be converted into unicode or similar if need be.

3 Linux

3.1 Linux compile-time flags

The Linux compile time flags are documented in the Kconfig file and will not be duplicated here.

3.2 Linux: /proc/yaffs trace adjustment

Under Linux, the tracing mechanism is hooked up to both the /proc/yaffs and the sys interface. The /proc/yaffs interface is very user friendly.

```
#cat +os > /proc/yaffs  # Enables YAFFS_TRACE_OS
#cat -all+gc > /proc/yaffs # Disables everything then enable
YAFFS_TRACE_GC
#cat 0xf000 > /proc/yaffs # Set trace mask to specified bitmask.
```

3.3 Linux: /sys/ interface

The Linux /sys/ interface is used by many Linux OS components to access the state of control variables. Some Yaffs variables are available via /sys/module/yaffs2/parameters/ *. Values can be read and modified using standard shell commands or from application programs.

```
# ls /sys/module/yaffs2/parameters/ yaffs_auto_checkpoint
yaffs_traceMask  yaffs_wr_attempts
# cat /sys/module/yaffs2/parameters/yaffs_auto_checkpoint 1
# echo "2" > /sys/module/yaffs2/parameters/yaffs_auto_checkpoint
# cat /sys/module/yaffs2/parameters/yaffs_auto_checkpoint 2
```

Yaffs currently supports the following variables:

yaffs_auto_checkpoint: Controls when checkpoints should be written.

yaffs_traceMask: An alternative, and less user friendly,interface to modify tracing.

yaffs_wr_attempts: The number of attempts that Yaffs will make to write a chunk.

yaffs_gc_control: Controls the garbage collector.

3.4 Linux: /proc/yaffs state dumping

Some of the state of Yaffs mounts can be determined by reading the /proc/yaffs entry.

```
# cat /proc/yaffs
charles@charles-laptop:/opt/y/cvs/yaffs2$ cat /proc/yaffs
YAFFS built:Mar 16 2010 11:52:25
$Id: yaffs_fs.c,v 1.101 2010-03-15 22:27:15 charles Exp $
$Id: yaffs_guts.c,v 1.119 2010-03-12 02:48:34 charles Exp $

Device 0 "NAND simulator partition 0"
```

```
startBlock......... 0
endBlock........... 511
totalBytesPerChunk. 2048
useNANDECC......... 1
noTagsECC.......... 0
isYaffs2........... 1
inbandTags......... 0
emptyLostAndFound.. 1
disableLazyLoad.... 0
refreshPeriod...... 10000
nShortOpCaches..... 10
nReservedBlocks.... 5

nDataBytesPerChunk. 2048
chunkGroupBits..... 0
chunkGroupSize..... 1
nErasedBlocks...... 152
blocksInCheckpoint. 0

nTnodesCreated..... 2300
nFreeTnodes........ 1852
nObjectsCreated.... 2300
nFreeObjects....... 1842
nFreeChunks........ 27843

nPageWrites........ 65993
nPageReads......... 19631
nBlockErasures..... 1133
nGCCopies.......... 37
garbageCollections. 161
passiveGCs......... 137
nRetriedWrites..... 0
nRetireBlocks...... 0
eccFixed........... 0
eccUnfixed......... 0
tagsEccFixed....... 0
tagsEccUnfixed..... 0
cacheHits.......... 44110
nDeletedFiles...... 0
nUnlinkedFiles..... 42917
refreshCount....... 1
nBackgroudDeletions 0
```

4 Conclusions

Yaffs includes many options for changing speed or memory use, with specific flags for the Yaffs Direct Interface as well as Linux. Under Linux many of the control variables can be altered using standard Linux systems.