

The IceFileSystem 2.x Disk Layout

By Leif Salomonsson (c) 2011

Last updated: June 2 2011.

Contents:

- Introduction.
- Conventions.
- Meta Headers.
- Meta Objects.
- Extents.
- Admin Space.
- Constants.
- Functions.

Introduction

IceFileSystem layout is extent based, 64bit, with checksummed meta data and a meta level journal.

Main space allocator is based on TLSF algorithm, adapted for on-disk storage.

All metadata except extent headers are located in special extents called pools and use a local bitmap to keep track of free meta space. Metadata is very compact.

Design goals and priorities

64bit disk filesystem supporting a high number of useful features, by a design prioritised like following:

1-Reliability 2-Scalability 3-Efficiency 4-Speed

Features

- 64bit file/partition/extent sizes (actually very close to 2^{63} bytes). No self imposed fragmentation of large files unless there is not enough contiguous space available.
- All metadata on disk is checksummed. This means any errors on disk will be detected a lot quicker.
- Meta level journalling.
- Hardlinks (directory and file), softlinks, file comments.
- Supports block sizes from 512 bytes to 32 KiB.
- Filesystem does not get slower for larger partitions (scales very well), or when heavily fragmented.
- No limit in # of files/dirs in partition/dirs.
- Recycle directory.
- Automatically truncated log files and file change log.
- Wastes only 128k (and preallocates another 128k for meta space) for filesystem administration data, regardless of partition size.

Conventions

All meta data stores information in big endian format.

All on-disk pointers express a byte offset relative to the start of the volume.

All fields in all structures described are of the unsigned integer kind, with sizes ranging from 1 to 8 bytes, or array of unsigned integer kind.

Field types:

- p64 – 64bit pointer
- p32 – 32bit pointer
- i64 – 64bit integer
- i32 – 32bit integer
- i16 – 16bit integer
- i8 - 8bit integer

Meta Headers

All meta objects starts with a meta header structure, the "meta" object:

```
-----  
selfadr    p64  
checksum   i32  
metasize   i16  
metarsrvd  i8  
metatag    i8  
-----  
SIZEOF = 16
```

selfadr:
 pointer to our selves.

checksum:
 total sum of all 32bit words (with "checksum" field itself
 cleared) that make up the whole object (size of object in
 "metasize" field).

metasize:
 total size of meta object.

reserved:
 zero.

metatag:
 integer describing the type of meta object.

Additionally, all meta that is stored in meta pools have one extra "bmapadr" field. The "pmeta" object:

```
-----  
selfadr    p64  
checksum   i32  
metasize   i16  
metarsrvd  i8  
metatag    i8  
bmapadr    p64  
-----  
SIZEOF = 24
```

bmapadr:
 pointer to the "bitmap" object.

Meta Objects

The "filesystem" object:

```
-----  
selfadr      p64  
checksum     i32  
metasize     i16  
metarsrvd    i8  
metatag      i8  
creation     i64  
totalsize    i64  
rootentry    p32  
rootnode     p32  
roothashtab p32  
recyclednode p32  
logfilenode  p32  
fsversion    i32  
fsrevision   i32  
blocksize    i32  
reserved2    [2] i32  
hashtype     i32  
extinfo      p32  
journal      p32  
startofextents p32  
endofextents p64  
powertabs    [64] p32  
orig_selfadr p64  
reserved     [19] i64  
-----
```

metasize:
512

metatag:
MTAG_FILESYS

creation:
time of creation expressed in microseconds since start
of time [1]

totalsize:
total size of volume expressed in bytes.

rootentry:
pointer to root "entry" object.

rootnode:
pointer to root "node" object.

roothashtab:
pointer to root "hashtab" object.

recyclednode:
pointer to recycled "node" object. may be zero for pre v2.3.

logfilenode:
pointer to logfile "node" object. may be zero for pre v2.3.

fsversion:
 version of filesystem handler used to format the volume.
 Always 2, for icefs 2.x.

fsrevision:
 revision for filesystem handler used to format the volume.

blocksize:
 Always power of 2. minimum 512 bytes, current maximum 32k.

reserved2:
 reserved, zero filled.

hashtype:
 HASHTYPE_XXX.

extinfo:
 pointer to "extinfo" object.

journal:
 pointer to "journal" object.

startofextents:
 Pointer to the first extent in volume.
 This extent is always a preallocated meta pool created
 when formatting the volume.

endofextents:
 pointer to end of last extent in volume.

powertabs:
 array of 64 pointers to "power" objects.

orig_selfadr:
 pointer to our selves (the "fileys" object).
 Useful in fileys backup to remember the original address.

reserved:
 array of reserved and zeroed 64bit words.

[1] start of time is Jan-1-1978 00:00

The "journal" object:

```

-----
selfadr      p64
checksum     i32
metasize     i16
metarsrvd    i8
metatag      i8
nummods      i32
rsrvd        i32
<data>
-----

```

metasize:
 128..32768 (preferably blocksize aligned)

metatag:

MTAG_JOURNAL

nummods:
 number of meta objects in journal.

rsrvd:
 zero.

<data>:
 meta objects of various sizes.

The "extinfo" object:

```
-----  
selfadr      p64  
checksum     i32  
metasize     i16  
metarsrvd    i8  
metatag      i8  
reuse_first  p64  
reuse_last   p64  
recycledfiles i32  
recycledbytes i64  
reserved     [21]i32  
-----
```

metasize:
 128

metatag:
 MTAG_EXTINFO

reuse_first:
 first metapool extent in list of non full meta pools.

reuse_last:
 last metapool extent in list of non full meta pools.

recycledfiles:
 number of files currently in recycled directory.

recycledbytes:
 number of bytes currently in recycled directory.

reserved:
 zero.

The "power" object:

```
-----  
selfadr      p64  
checksum     i32  
metasize     i16  
metarsrvd    i8  
metatag      i8  
freespace    i64  
pad12        [12]i64  
power        i32  
pad          i32  
table        [64]p64
```

```

-----
metasize:
    640

metatag:
    MTAG_POWER

freespace:
    total space in free extents linked by this power table.

pad12:
    zero.

power:
    power number of this power. first power is 1, last is 64.
    filesystem.powertabs[0] points to power 1. only power 9..63 are
    actually used, but all powers should still be initialised.

pad:
    zero.

table:
    Free extents are chained together in a doubly
    linked list and the table contains pointers to the
    first extent in each list, or zero if empty.

```

The "entry" object:

```

-----
selfadr    p64
checksum   i32
metasize   i16
metarsrvd  i8
metatag    i8
bmapadr    p64
link       p64
parent     p64
nameext    p64
nexthlink  p64
type       i8
flags      i8
namelen    i16
dirnext    p64
dirprev    p64
hashnext   p64
hashprev   p64
name       [36]i8
-----

metasize:
    128

metatag:
    MTAG_ENTRY

link:
    pointer to either "softname" if type = ETYPE_SOFTENTRY, or "node"
    if type = ETYPE_HARDENTRY.

parent:

```

pointer to parent entry. zero if root.

nameext:
 pointer to "nameext" object if filename exceeds 36 characters,
 else zero.

nextlink:
 next hard link. pointer used to link hard links for an
 entry together. original entry is not part of this linkage.

type:
 ETYPE_XXX. either ETYPE_HARDENTRY or ETYPE_SOFTENTRY.

flags:
 EFLAG_XXX. only one flag for now: EFLAG_HIDDEN.

namelen:
 total length of filename in characters.

dirnext:
 pointer to next entry in directory, or zero if last.

dirprev:
 pointer to previous entry in directory, or zero if first.

hashnext:
 pointer to next entry in hash table linkage.

hashprev:
 pointer to previous entry in hash table linkage.

name:
 array containing the (first 36 characters of) filename.

The "nameext" object:

```

-----
selfadr    p64
checksum   i32
metasize   i16
metarsrvd  i8
metatag    i8
bmapadr    p64
str        [1000]i8
-----

```

metasize:
 128..1024 (METABLOCKSIZE aligned)

metatag:
 MTAG_NAMEEXT

str:
 array containing up to 1000 characters extending filename
 above the 36 byte limit.

The "softname" object:

```
-----  
selfadr    p64  
checksum   i32  
metasize   i16  
metarsrvd  i8  
metatag    i8  
bmapadr    p64  
softtime   p64  
str        [992]i8  
-----
```

metasize:
128..1024 (METABLOCKSIZE aligned)

metatag:
MTAG_SOFTNAME

softtime:
microsecond timestamp describing time of creation.

str:
0-terminated target string for softlink.

The "node" object:

```
-----  
selfadr    p64  
checksum   i32  
metasize   i16  
metarsrvd  i8  
metatag    i8  
bmapadr    p64  
type       i8  
rsrvd     i8  
flags      i16  
protbits   i32  
reserved   i32  
ownerinfo  i32  
origentry  p64  
addentries p64  
modified   i64  
data_first p64  
data_last  p64  
dir_first  p64 @ data_first  
dir_last   p64 @ data_last  
extleft    i64  
numentries i64 @ extleft  
filesize   i64  
hash       p64 @ filesize  
comment    p64  
rsrvd3     [3]i64  
-----
```

metasize:
128

metatag:
MTAG_NODE

type:

Either NTYPE_FILE or NTYPE_DIR.

rsrvd:

zero.

flags:

Defined flags are NFLAG_LOGFILE and NFLAG_RECYCLEDIR.

protbits:

Protection bits. Like amiga protection bits but
without the inversion of the lower nibble.

reserved:

zero.

ownerinfo:

Like amiga ownerinfo.

origentry:

pointer to the "original" entry.

addentries:

pointer to the first hard link entry, or zero.

modified:

modification timestamp expressed in micro seconds since
start of time.

data_first:

pointer to first filedata extent, or zero.

data_last:

pointer to last filedata extent, or zero.

dir_first:

pointer to first entry in directory, or zero.

dir_last:

pointer to last entry in directory, or zero.

extleft:

number of bytes left in last extent.

numentries:

number of entries in directory. not really used by
anything, but should be updated anyway.

filesize:

the current filesize, if NTYPE_FILE.

hash:

pointer to "hashtab" object, if NTYPE_DIR.

comment:

pointer to "comment" object, or zero.

rsrvd3:

zero.

The "hashtab" object:

```
-----  
selfadr    p64  
checksum   i32  
metasize   i16  
metarsrvd  i8  
metatag    i8  
bmapadr    p64  
parent     p64  
table      [124]p64  
-----
```

metasize:
1024

parent:
back pointer to the "node" object.

table:
array of pointers to "entry" objects.

The "bitmap" object:

```
-----  
selfadr    p64  
checksum   i32  
metasize   i16  
metarsrvd  i8  
metatag    i8  
bmapadr    p64  
reserved   [12]i64  
pooladr    p64  
longs      [32]i32  
-----
```

metasize:
256

metatag:
MTAG_BITMAP

reserved:
zero.

pooladr:
always selfadr - SIZEOF extent.

longs:
128 bytes of bitmap. Each bit represents a 128 bytes block.
A set bit means the block is allocated. The first bit in bitmap maps to the selfadr of the bitmap itself (that is, directly after extent header). This means first two bits are always set to allocate the bitmap itself. The last bit of the bitmap is always set as it would otherwise represent free space outside of the metapool. Because of the bitmap beginning 64 bytes into metapool, metadata starting address in metapools is only 64-byte aligned instead of 128byte aligned.

Extents

All extents starts with the "extent" object header, which in turn starts with the "meta" object header.

The "extent" object:

```
-----  
selfadr    p64  
checksum   i32  
metasize   i16  
metarsrvd  i8  
metatag    i8  
extsize    i64  
ext_prev   p64  
data_next  p64  
data_prev  p64  
dataofs    i32  
data_extnum i32  
data_node  p64  
/* unions */  
range_next p64 @ data_next  
range_prev p64 @ data_prev  
reuse_next p64 @ data_next  
reuse_prev p64 @ data_prev  
bitmap     p64 @ dataofs  
-----
```

metasize:
64

selfadr:
always blocksize aligned.

metasize:
always 64

metatag:
Type of extent (ETAG_XXX).
There are three types of extents:
ETAG_FREE: free space.
ETAG_METAPPOOL: 128KiB meta pool.
ETAG_FILEDATA: file data extent.

extsize:
size of this extent in bytes. always blocksize aligned.

ext_prev:
pointer to previous extent by placement on disk.

#if metatag = ETAG_FILEDATA

data_next:
pointer to next filedata extent.

data_prev:
pointer to previous filedata extent.

dataofs:

Start of filedata relative to start of extent described in bytes.
Either SIZEOF extent (64) for files that fits in (blocksize - SIZEOF extent),
or blocksize for larger files.

```
data_extnum:
    filedata extent number, starting at 0.

data_node:
    pointer back to the "node" object.

#endif

#if metatag = ETAG_FREE

range_next:
    pointer to next free extent in this range.

range_prev:
    pointer to previous free extent in this range.

#endif

#if metatag = ETAG_METAPPOOL

reuse_next:
    pointer to next non full metapool extent.

reuse_prev:
    pointer to previous non full metapool extent.

bitmap:
    pointer to "bitmap" object. ALWAYS selfadr + 64.

#endif
```

Admin Space

The size of adminspace is 128KiB. It is located at beginning of partition and starts with the "icestart" structure:

```
-----  
ice0      i32  
fsys      i32  
filesys   p32  
-----
```

ice0:
 the magic word ID_ICEFS_DISK

fsys:
 the magic word "FSYS"

filesys:
 pointer to the "filesys" object.

Meta objects in admin space:

```
1  Filesys  
1  Extinfo  
64 Power's  
1  Journal
```

Backed up objects.

A volume ends with a copy of the root hashtable object.
As of revision 3, also the filesys object is backed up, directly in front of the root hashtable backup. Both hashtable and filesys backup have their "selfadr" field changed to backup location. The "filesys" object keeps a backup of the original "selfadr" in its "orig_selfadr" field.

Constants

ID_ICEFS_DISK = 0x49434502

MTAG_NONE = 0

ETAG_FREE = 1
ETAG_METAPPOOL = 2
ETAG_FILEDATA = 3
ETAG_RESERVED = 4

MTAG_FILESYS = 5
MTAG_ENTRY = 6
MTAG_NODE = 7
MTAG_NAMEEXT = 8
MTAG_COMMENT = 9
MTAG_SOFTNAME = 10
MTAG_HASHTAB = 11
MTAG_NOTUSED = 12
MTAG_POWER = 13
MTAG_JOURNAL = 14
MTAG_BITMAP = 15
MTAG_EXTINFO = 16
MTAG_RESERVED = 17

METAPPOOLSIZE = 1024 * 128
ADMINSIZE = 1024 * 128

METABLOCKSIZE = 128
METABLOCKMASK = 127
METABLOCKSHIFT = 7

MAXBITMAPINDEX = 31
BITMAPLONGS = 32
BITMAPRESERVED = 12

ETYPE_HARDENTRY = 0
ETYPE_SOFTENTRY = 1
EFLAG_HIDDEN = 1
MAXENTRYNAMEBYTES = 36

NTYPE_FILE = 0
NTYPE_DIR = 1
NFLAG_LOGFILE = 1
NFLAG_RECYCLEDIR = 2

HASHTABENTRIES = 124

MAXNAMEEXTSTRBYTES = 1000
MAXSOFTNAMESTRBYTES = 992
MAXCOMMENTSTRBYTES = 1000

MINPOWER = 6
MAXPOWER = 63

RANGESPERPOWER = 64
RANGESPERPOWERSHIFT = 6

FILESYSRESERVED = 19

HASHTYPE_NEW = 2
HASHTYPE_NEW_CASE = 3

EXTINFORESERVED = 21

MAXJOURNALSIZE = 32768

MAXMETASIZE = 1024

Functions

Functions are written in E language, but should not be too hard to translate into C or something else.

The directory entry name hashing algorithm:

Case sensitive:

```
PROC hashDiskNameNewCase124(name:PTR TO CHAR)
  DEF i, v=0:LONG, c

  -> get a small seed from number of characters
  WHILE c := name[v]
    EXIT c = "/"
    v++
  ENDWHILE

  -> compute hash
  WHILE c := name[]++
    EXIT c = "/"
    v := v * 13 + c
  ENDWHILE

  -> and adjust into range
  i := v AND 127
  IF i >= 124 THEN i := i - 124 -> wrap

ENDPROC i
```

Case insensitive:

```
PROC hashDiskNameNew124(name:PTR TO CHAR)
  DEF i, v=0:LONG, c

  -> get a small seed from number of characters
  WHILE c := name[v]
    EXIT c = "/"
    v++
  ENDWHILE

  -> compute hash
  WHILE c := name[]++
    EXIT c = "/"
    IF c >= "a"
      IF c <= "z"
        c -= 32
      ENDIF
    ENDIF
    IF c >= 224
      IF c <= 254
        IF c <> 247
          c -= 32
        ENDIF
      ENDIF
    ENDIF
    v := v * 13 + c
  ENDWHILE

ENDPROC i
```

```

-> and adjust into range
i := v AND 127
IF i >= 124 THEN i := i - 124 -> wrap

ENDPROC i

```

The TLSF power/index lookup function:

```

PROC get_extsize_powernum(extsize:WIDE) (LONG, LONG)
  DEF powerval:WIDE, p=64, index

  REPEAT
    p--
    powerval := @ 1 SHL p
  UNTIL @ powerval AND extsize

  index := @ extsize - powerval SHL RANGESPERPOWERSHIFT SHR p

ENDPROC p, index

```

The meta set/check sum functions:

```

PROC setMetaSum(meta:PTR TO meta)
  DEF v=NIL:LONG, ptr:PTR TO LONG, len
  ptr := meta
  meta.checksum := NIL
  len := meta.metasize SHR 2 + 1
  WHILE len-- DO v := v + ptr[]++
  meta.checksum := v
ENDPROC

PROC checkMetaSum(meta:PTR TO meta)
  DEF v=NIL:LONG, ptr:PTR TO LONG, len
  ptr := meta
  len := meta.metasize SHR 2 + 1
  WHILE len-- DO v := v + ptr[]++
ENDPROC (v - meta.checksum) = meta.checksum

```