# LCOV - code coverage report

| | | | Found | Hit | Cover |
|---|---|---|---|---|---|
| **Current view:** | directory - fs/ext4 - hash.c (source / functions) | | | | |
| **Test:** | kernel_2_6_31_ext4_round_3.info | **Lines:** | 113 | 48 | 42.5 |
| **Date:** | 2009-10-24 | **Functions:** | 3 | 2 | 66.7 |

```
                :  /*
                :   *  linux/fs/ext4/hash.c
                :   *
                :   * Copyright (C) 2002 by Theodore Ts'o
                :   *
                :   * This file is released under the GPL v2.
                :   *
                :   * This file may be redistributed under the terms of the GNU Public
                :   * License.
                :   */
                :
                :  #include <linux/fs.h>
                :  #include <linux/jbd2.h>
                :  #include <linux/cryptohash.h>
                :  #include "ext4.h"
                :
                :  #define DELTA 0x9E3779B9
                :
                :  static void TEA_transform(__u32 buf[4], __u32 const in[])
                :  {
              0 :          __u32   sum = 0;
              0 :          __u32   b0 = buf[0], b1 = buf[1];
              0 :          __u32   a = in[0], b = in[1], c = in[2], d = in[3];
              0 :          int     n = 16;
                :
                :          do {
              0 :                  sum += DELTA;
              0 :                  b0 += ((b1 << 4)+a) ^ (b1+sum) ^ ((b1 >> 5)+b);
              0 :                  b1 += ((b0 << 4)+c) ^ (b0+sum) ^ ((b0 >> 5)+d);
              0 :          } while (--n);
                :
              0 :          buf[0] += b0;
              0 :          buf[1] += b1;
                :  }
                :
                :
                :  /* The old legacy hash */
                :  static __u32 dx_hack_hash_unsigned(const char *name, int len)
                :  {
              0 :          __u32 hash, hash0 = 0x12a3fe2d, hash1 = 0x37abe8f9;
              0 :          const unsigned char *ucp = (const unsigned char *) name;
                :
              0 :          while (len--) {
              0 :                  hash = hash1 + (hash0 ^ (((int) *ucp++) * 7152373));
                :
              0 :                  if (hash & 0x80000000)
              0 :                          hash -= 0x7fffffff;
              0 :                  hash1 = hash0;
              0 :                  hash0 = hash;
                :          }
              0 :          return hash0 << 1;
                :  }
```

```
                :
                : static __u32 dx_hack_hash_signed(const char *name, int len)
                : {
              0 :         __u32 hash, hash0 = 0x12a3fe2d, hash1 = 0x37abe8f9;
              0 :         const signed char *scp = (const signed char *) name;
                :
              0 :         while (len--) {
              0 :                 hash = hash1 + (hash0 ^ (((int) *scp++) * 7152373));
                :
              0 :                 if (hash & 0x80000000)
              0 :                         hash -= 0x7fffffff;
              0 :                 hash1 = hash0;
              0 :                 hash0 = hash;
                :         }
              0 :         return hash0 << 1;
                : }
                :
                : static void str2hashbuf_signed(const char *msg, int len, __u32 *buf, int num)
       51136495 : {
                :         __u32   pad, val;
                :         int     i;
       51136495 :         const signed char *scp = (const signed char *) msg;
                :
       51136495 :         pad = (__u32)len | ((__u32)len << 8);
       51136495 :         pad |= pad << 16;
                :
       51136495 :         val = pad;
       51136495 :         if (len > num*4)
       12096910 :                 len = num * 4;
      954390680 :         for (i = 0; i < len; i++) {
      903254185 :                 if ((i % 4) == 0)
      248222840 :                         val = pad;
      903254185 :                 val = ((int) scp[i]) + (val << 8);
      903254185 :                 if ((i % 4) == 3) {
      213304994 :                         *buf++ = val;
      213304994 :                         val = pad;
      213304994 :                         num--;
                :                 }
                :         }
       51136495 :         if (--num >= 0)
       39039906 :                 *buf++ = val;
      207885819 :         while (--num >= 0)
      156749324 :                 *buf++ = pad;
       51136495 : }
                :
                : static void str2hashbuf_unsigned(const char *msg, int len, __u32 *buf, int num)
              0 : {
                :         __u32   pad, val;
                :         int     i;
              0 :         const unsigned char *ucp = (const unsigned char *) msg;
                :
              0 :         pad = (__u32)len | ((__u32)len << 8);
              0 :         pad |= pad << 16;
                :
              0 :         val = pad;
              0 :         if (len > num*4)
              0 :                 len = num * 4;
              0 :         for (i = 0; i < len; i++) {
              0 :                 if ((i % 4) == 0)
              0 :                         val = pad;
              0 :                 val = ((int) ucp[i]) + (val << 8);
              0 :                 if ((i % 4) == 3) {
              0 :                         *buf++ = val;
              0 :                         val = pad;
              0 :                         num--;
```

```
       :                 }
       :         }
     0 :         if (--num >= 0)
     0 :                 *buf++ = val;
     0 :         while (--num >= 0)
     0 :                 *buf++ = pad;
     0 : }
       :
       : /*
       :  * Returns the hash of a filename.  If len is 0 and name is NULL, then
       :  * this function can be used to test whether or not a hash version is
       :  * supported.
       :  *
       :  * The seed is an 4 longword (32 bits) "secret" which can be used to
       :  * uniquify a hash.  If the seed is all zero's, then some default seed
       :  * may be used.
       :  *
       :  * A particular hash version specifies whether or not the seed is
       :  * represented, and whether or not the returned hash is 32 bits or 64
       :  * bits.  32 bit hashes will return 0 for the minor hash.
       :  */
       : int ext4fs_dirhash(const char *name, int len, struct dx_hash_info *hinfo)
39038676 : {
       :         __u32   hash;
39038676 :         __u32   minor_hash = 0;
       :         const char      *p;
       :         int             i;
       :         __u32           in[8], buf[4];
       :         void            (*str2hashbuf)(const char *, int, __u32 *, int) =
39038676 :                                 str2hashbuf_signed;
       :
       :         /* Initialize the default seed for the hash checksum functions */
39038676 :         buf[0] = 0x67452301;
39038676 :         buf[1] = 0xefcdab89;
39038676 :         buf[2] = 0x98badcfe;
39038676 :         buf[3] = 0x10325476;
       :
       :         /* Check to see if the seed is all zero's */
39038676 :         if (hinfo->seed) {
39039387 :                 for (i = 0; i < 4; i++) {
39039424 :                         if (hinfo->seed[i])
39039424 :                                 break;
       :                 }
39039387 :                 if (i < 4)
39039232 :                         memcpy(buf, hinfo->seed, sizeof(buf));
       :         }
       :
39038967 :         switch (hinfo->hash_version) {
       :         case DX_HASH_LEGACY_UNSIGNED:
     0 :                 hash = dx_hack_hash_unsigned(name, len);
     0 :                 break;
       :         case DX_HASH_LEGACY:
     0 :                 hash = dx_hack_hash_signed(name, len);
     0 :                 break;
       :         case DX_HASH_HALF_MD4_UNSIGNED:
     0 :                 str2hashbuf = str2hashbuf_unsigned;
       :         case DX_HASH_HALF_MD4:
39039431 :                 p = name;
129215540 :                 while (len > 0) {
51136254 :                         (*str2hashbuf)(p, len, in, 8);
51136542 :                         half_md4_transform(buf, in);
51136678 :                         len -= 32;
51136678 :                         p += 32;
       :                 }
39039855 :                 minor_hash = buf[2];
```

```
   183        39039855 :                 hash = buf[1];
   184        39039855 :                 break;
   185               : 	case DX_HASH_TEA_UNSIGNED:
   186               0 :                 str2hashbuf = str2hashbuf_unsigned;
   187               : 	case DX_HASH_TEA:
   188               0 :                 p = name;
   189               0 :                 while (len > 0) {
   190               0 :                         (*str2hashbuf)(p, len, in, 4);
   191               :                         TEA_transform(buf, in);
   192               0 :                         len -= 16;
   193               0 :                         p += 16;
   194               :                 }
   195               0 :                 hash = buf[0];
   196               0 :                 minor_hash = buf[1];
   197               0 :                 break;
   198               : 	default:
   199               0 :                 hinfo->hash = 0;
   200               0 :                 return -1;
   201               : 	}
   202        39039391 : 	hash = hash & ~1;
   203        39039391 : 	if (hash == (EXT4_HTREE_EOF << 1))
   204               0 :                 hash = (EXT4_HTREE_EOF-1) << 1;
   205        39039391 : 	hinfo->hash = hash;
   206        39039391 : 	hinfo->minor_hash = minor_hash;
   207        39039391 : 	return 0;
   208               : }
```