Subject: - fix-some-coding-style-errors-in-autofs.patch removed from -mm tree
Posted by akpm on Sat, 12 May 2007 05:34:44 GMT
View Forum Message <> Reply to Message

The patch titled
    Fix some coding-style errors in autofs
has been removed from the -mm tree. Its filename was
    fix-some-coding-style-errors-in-autofs.patch

This patch was dropped because it was merged into mainline or a subsystem tree

----
Subject: Fix some coding-style errors in autofs
From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Fix coding style errors (extra spaces, long lines) in autofs and autofs4 files
being modified for container/pidspace issues.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: <containers@lists.osdl.org>
Cc: Eric W. Biederman <ebiederm@xmission.com>
Cc: Ian Kent <raven@themaw.net>
Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

```
 fs/autofs/inode.c  |   29 ++++++++--------
 fs/autofs/root.c   |   77 +++++++++++++++++++++---------------------
 fs/autofs4/inode.c |   16 +++-----
 fs/autofs4/root.c  |   18 +++++-----
 4 files changed, 70 insertions(+), 70 deletions(-)

diff -puN fs/autofs/inode.c~fix-some-coding-style-errors-in-autofs fs/autofs/inode.c
--- a/fs/autofs/inode.c~fix-some-coding-style-errors-in-autofs
+++ a/fs/autofs/inode.c
@@ -34,12 +34,12 @@ void autofs_kill_sb(struct super_block *
  if (!sbi)
   goto out_kill_sb;

- if ( !sbi->catatonic )
+ if (!sbi->catatonic)
   autofs_catatonic_mode(sbi); /* Free wait queues, close pipe */

  autofs_hash_nuke(sbi);
- for ( n = 0 ; n < AUTOFS_MAX_SYMLINKS ; n++ ) {
-  if ( test_bit(n, sbi->symlink_bitmap) )
```

```
+ for (n = 0; n < AUTOFS_MAX_SYMLINKS; n++) {
+  if (test_bit(n, sbi->symlink_bitmap))
    kfree(sbi->symlink[n].data);
  }

@@ -69,7 +69,8 @@ static match_table_t autofs_tokens = {
  {Opt_err, NULL}
 };

-static int parse_options(char *options, int *pipefd, uid_t *uid, gid_t *gid, pid_t *pgrp, int *minproto,
int *maxproto)
+static int parse_options(char *options, int *pipefd, uid_t *uid, gid_t *gid,
+  pid_t *pgrp, int *minproto, int *maxproto)
 {
  char *p;
  substring_t args[MAX_OPT_ARGS];
@@ -140,7 +141,7 @@ int autofs_fill_super(struct super_block
  int minproto, maxproto;

  sbi = kzalloc(sizeof(*sbi), GFP_KERNEL);
- if ( !sbi )
+ if (!sbi)
   goto fail_unlock;
  DPRINTK(("autofs: starting up, sbi = %p\n",sbi));

@@ -169,14 +170,16 @@ int autofs_fill_super(struct super_block
   goto fail_iput;

  /* Can this call block?  - WTF cares? s is locked. */
- if (
parse_options(data,&pipefd,&root_inode->i_uid,&root_inode->i_gid,&sbi->oz_pgrp,&minproto,&m
axproto) ) {
+ if (parse_options(data, &pipefd, &root_inode->i_uid,
+    &root_inode->i_gid, &sbi->oz_pgrp, &minproto,
+    &maxproto)) {
   printk("autofs: called with bogus options\n");
   goto fail_dput;
  }

  /* Couldn't this be tested earlier? */
- if ( minproto > AUTOFS_PROTO_VERSION ||
-      maxproto < AUTOFS_PROTO_VERSION ) {
+ if (minproto > AUTOFS_PROTO_VERSION ||
+      maxproto < AUTOFS_PROTO_VERSION) {
   printk("autofs: kernel does not match daemon version\n");
   goto fail_dput;
  }
@@ -184,11 +187,11 @@ int autofs_fill_super(struct super_block
```

```
  DPRINTK(("autofs: pipe fd = %d, pgrp = %u\n", pipefd, sbi->oz_pgrp));
  pipe = fget(pipefd);

- if ( !pipe ) {
+ if (!pipe) {
    printk("autofs: could not open pipe file descriptor\n");
    goto fail_dput;
  }
- if ( !pipe->f_op || !pipe->f_op->write )
+ if (!pipe->f_op || !pipe->f_op->write)
    goto fail_fput;
  sbi->pipe = pipe;
  sbi->catatonic = 0;
@@ -230,7 +233,7 @@ static void autofs_read_inode(struct ino
  inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
  inode->i_blocks = 0;

- if ( ino == AUTOFS_ROOT_INO ) {
+ if (ino == AUTOFS_ROOT_INO) {
    inode->i_mode = S_IFDIR | S_IRUGO | S_IXUGO | S_IWUSR;
    inode->i_op = &autofs_root_inode_operations;
    inode->i_fop = &autofs_root_operations;
@@ -241,12 +244,12 @@ static void autofs_read_inode(struct ino
  inode->i_uid = inode->i_sb->s_root->d_inode->i_uid;
  inode->i_gid = inode->i_sb->s_root->d_inode->i_gid;

- if ( ino >= AUTOFS_FIRST_SYMLINK && ino < AUTOFS_FIRST_DIR_INO ) {
+ if (ino >= AUTOFS_FIRST_SYMLINK && ino < AUTOFS_FIRST_DIR_INO) {
    /* Symlink inode - should be in symlink list */
    struct autofs_symlink *sl;

    n = ino - AUTOFS_FIRST_SYMLINK;
-   if ( n >= AUTOFS_MAX_SYMLINKS || !test_bit(n,sbi->symlink_bitmap)) {
+   if (n >= AUTOFS_MAX_SYMLINKS || !test_bit(n,sbi->symlink_bitmap)) {
      printk("autofs: Looking for bad symlink inode %u\n", (unsigned int) ino);
      return;
    }
diff -puN fs/autofs/root.c~fix-some-coding-style-errors-in-autofs fs/autofs/root.c
--- a/fs/autofs/root.c~fix-some-coding-style-errors-in-autofs
+++ a/fs/autofs/root.c
@@ -67,8 +67,8 @@ static int autofs_root_readdir(struct fi
    filp->f_pos = ++nr;
    /* fall through */
  default:
-   while ( onr = nr, ent = autofs_hash_enum(dirhash,&nr,ent) ) {
-     if ( !ent->dentry || d_mountpoint(ent->dentry) ) {
+   while (onr = nr, ent = autofs_hash_enum(dirhash,&nr,ent)) {
+     if (!ent->dentry || d_mountpoint(ent->dentry)) {
```

```
     if (filldir(dirent,ent->name,ent->len,onr,ent->ino,DT_UNKNOWN) < 0)
      goto out;
     filp->f_pos = nr;
@@ -88,10 +88,10 @@ static int try_to_fill_dentry(struct den
  struct autofs_dir_ent *ent;
  int status = 0;

- if ( !(ent = autofs_hash_lookup(&sbi->dirhash, &dentry->d_name)) ) {
+ if (!(ent = autofs_hash_lookup(&sbi->dirhash, &dentry->d_name))) {
   do {
-   if ( status && dentry->d_inode ) {
-    if ( status != -ENOENT )
+   if (status && dentry->d_inode) {
+    if (status != -ENOENT)
      printk("autofs warning: lookup failure on positive dentry, status = %d, name = %s\n", status,
dentry->d_name.name);
     return 0; /* Try to get the kernel to invalidate this dentry */
    }
@@ -106,7 +106,7 @@ static int try_to_fill_dentry(struct den
     return 1;
    }
    status = autofs_wait(sbi, &dentry->d_name);
-  } while (!(ent = autofs_hash_lookup(&sbi->dirhash, &dentry->d_name)) );
+  } while (!(ent = autofs_hash_lookup(&sbi->dirhash, &dentry->d_name)));
  }

  /* Abuse this field as a pointer to the directory entry, used to
@@ -124,13 +124,13 @@ static int try_to_fill_dentry(struct den

  /* If this is a directory that isn't a mount point, bitch at the
     daemon and fix it in user space */
- if ( S_ISDIR(dentry->d_inode->i_mode) && !d_mountpoint(dentry) ) {
+ if (S_ISDIR(dentry->d_inode->i_mode) && !d_mountpoint(dentry)) {
   return !autofs_wait(sbi, &dentry->d_name);
  }

  /* We don't update the usages for the autofs daemon itself, this
     is necessary for recursive autofs mounts */
- if ( !autofs_oz_mode(sbi) ) {
+ if (!autofs_oz_mode(sbi)) {
   autofs_update_usage(&sbi->dirhash,ent);
  }

@@ -157,7 +157,7 @@ static int autofs_revalidate(struct dent
  sbi = autofs_sbi(dir->i_sb);

  /* Pending dentry */
- if ( dentry->d_flags & DCACHE_AUTOFS_PENDING ) {
```

```
+ if (dentry->d_flags & DCACHE_AUTOFS_PENDING) {
   if (autofs_oz_mode(sbi))
    res = 1;
   else
@@ -173,7 +173,7 @@ static int autofs_revalidate(struct dent
  }

  /* Check for a non-mountpoint directory */
- if ( S_ISDIR(dentry->d_inode->i_mode) && !d_mountpoint(dentry) ) {
+ if (S_ISDIR(dentry->d_inode->i_mode) && !d_mountpoint(dentry)) {
   if (autofs_oz_mode(sbi))
    res = 1;
   else
@@ -183,9 +183,9 @@ static int autofs_revalidate(struct dent
  }

  /* Update the usage list */
- if ( !autofs_oz_mode(sbi) ) {
+ if (!autofs_oz_mode(sbi)) {
   ent = (struct autofs_dir_ent *) dentry->d_time;
-  if ( ent )
+  if (ent)
    autofs_update_usage(&sbi->dirhash,ent);
  }
  unlock_kernel();
@@ -258,7 +258,7 @@ static struct dentry *autofs_root_lookup
   * doesn't do the right thing for all system calls, but it should
   * be OK for the operations we permit from an autofs.
   */
- if ( dentry->d_inode && d_unhashed(dentry) )
+ if (dentry->d_inode && d_unhashed(dentry))
   return ERR_PTR(-ENOENT);

  return NULL;
@@ -277,18 +277,18 @@ static int autofs_root_symlink(struct in
  autofs_say(dentry->d_name.name,dentry->d_name.len);

  lock_kernel();
- if ( !autofs_oz_mode(sbi) ) {
+ if (!autofs_oz_mode(sbi)) {
   unlock_kernel();
   return -EACCES;
  }

- if ( autofs_hash_lookup(dh, &dentry->d_name) ) {
+ if (autofs_hash_lookup(dh, &dentry->d_name)) {
   unlock_kernel();
   return -EEXIST;
```

```
  }

  n = find_first_zero_bit(sbi->symlink_bitmap,AUTOFS_MAX_SYMLINKS);
- if ( n >= AUTOFS_MAX_SYMLINKS ) {
+ if (n >= AUTOFS_MAX_SYMLINKS) {
    unlock_kernel();
    return -ENOSPC;
  }
@@ -297,14 +297,14 @@ static int autofs_root_symlink(struct in
  sl = &sbi->symlink[n];
  sl->len = strlen(symname);
  sl->data = kmalloc(slsize = sl->len+1, GFP_KERNEL);
- if ( !sl->data ) {
+ if (!sl->data) {
    clear_bit(n,sbi->symlink_bitmap);
    unlock_kernel();
    return -ENOSPC;
  }

  ent = kmalloc(sizeof(struct autofs_dir_ent), GFP_KERNEL);
- if ( !ent ) {
+ if (!ent) {
    kfree(sl->data);
    clear_bit(n,sbi->symlink_bitmap);
    unlock_kernel();
@@ -312,7 +312,7 @@ static int autofs_root_symlink(struct in
  }

  ent->name = kmalloc(dentry->d_name.len+1, GFP_KERNEL);
- if ( !ent->name ) {
+ if (!ent->name) {
    kfree(sl->data);
    kfree(ent);
    clear_bit(n,sbi->symlink_bitmap);
@@ -354,23 +354,23 @@ static int autofs_root_unlink(struct ino

  /* This allows root to remove symlinks */
  lock_kernel();
- if ( !autofs_oz_mode(sbi) && !capable(CAP_SYS_ADMIN) ) {
+ if (!autofs_oz_mode(sbi) && !capable(CAP_SYS_ADMIN)) {
    unlock_kernel();
    return -EACCES;
  }

  ent = autofs_hash_lookup(dh, &dentry->d_name);
- if ( !ent ) {
+ if (!ent) {
    unlock_kernel();
```

```
   return -ENOENT;
  }

  n = ent->ino - AUTOFS_FIRST_SYMLINK;
- if ( n >= AUTOFS_MAX_SYMLINKS ) {
+ if (n >= AUTOFS_MAX_SYMLINKS) {
   unlock_kernel();
   return -EISDIR; /* It's a directory, dummy */
  }
- if ( !test_bit(n,sbi->symlink_bitmap) ) {
+ if (!test_bit(n,sbi->symlink_bitmap)) {
   unlock_kernel();
   return -EINVAL; /* Nonexistent symlink?  Shouldn't happen */
  }
@@ -392,23 +392,23 @@ static int autofs_root_rmdir(struct inod
  struct autofs_dir_ent *ent;

  lock_kernel();
- if ( !autofs_oz_mode(sbi) ) {
+ if (!autofs_oz_mode(sbi)) {
   unlock_kernel();
   return -EACCES;
  }

  ent = autofs_hash_lookup(dh, &dentry->d_name);
- if ( !ent ) {
+ if (!ent) {
   unlock_kernel();
   return -ENOENT;
  }

- if ( (unsigned int)ent->ino < AUTOFS_FIRST_DIR_INO ) {
+ if ((unsigned int)ent->ino < AUTOFS_FIRST_DIR_INO) {
   unlock_kernel();
   return -ENOTDIR; /* Not a directory */
  }

- if ( ent->dentry != dentry ) {
+ if (ent->dentry != dentry) {
   printk("autofs_rmdir: odentry != dentry for entry %s\n", dentry->d_name.name);
  }

@@ -429,18 +429,18 @@ static int autofs_root_mkdir(struct inod
  ino_t ino;

  lock_kernel();
- if ( !autofs_oz_mode(sbi) ) {
+ if (!autofs_oz_mode(sbi)) {
```

```
   unlock_kernel();
   return -EACCES;
  }

  ent = autofs_hash_lookup(dh, &dentry->d_name);
- if ( ent ) {
+ if (ent) {
   unlock_kernel();
   return -EEXIST;
  }

- if ( sbi->next_dir_ino < AUTOFS_FIRST_DIR_INO ) {
+ if (sbi->next_dir_ino < AUTOFS_FIRST_DIR_INO) {
   printk("autofs: Out of inode numbers -- what the heck did you do??\n");
   unlock_kernel();
   return -ENOSPC;
@@ -448,13 +448,13 @@ static int autofs_root_mkdir(struct inod
  ino = sbi->next_dir_ino++;

  ent = kmalloc(sizeof(struct autofs_dir_ent), GFP_KERNEL);
- if ( !ent ) {
+ if (!ent) {
   unlock_kernel();
   return -ENOSPC;
  }

  ent->name = kmalloc(dentry->d_name.len+1, GFP_KERNEL);
- if ( !ent->name ) {
+ if (!ent->name) {
   kfree(ent);
   unlock_kernel();
   return -ENOSPC;
@@ -483,7 +483,7 @@ static inline int autofs_get_set_timeout
      put_user(sbi->exp_timeout / HZ, p))
   return -EFAULT;

- if ( ntimeout > ULONG_MAX/HZ )
+ if (ntimeout > ULONG_MAX/HZ)
   sbi->exp_timeout = 0;
  else
   sbi->exp_timeout = ntimeout * HZ;
@@ -511,15 +511,14 @@ static inline int autofs_expire_run(stru
  pkt.hdr.proto_version = AUTOFS_PROTO_VERSION;
  pkt.hdr.type = autofs_ptype_expire;

- if ( !sbi->exp_timeout ||
-      !(ent = autofs_expire(sb,sbi,mnt)) )
+ if (!sbi->exp_timeout || !(ent = autofs_expire(sb,sbi,mnt)))
```

```
   return -EAGAIN;

  pkt.len = ent->len;
  memcpy(pkt.name, ent->name, pkt.len);
  pkt.name[pkt.len] = '\0';

- if ( copy_to_user(pkt_p, &pkt, sizeof(struct autofs_packet_expire)) )
+ if (copy_to_user(pkt_p, &pkt, sizeof(struct autofs_packet_expire)))
   return -EFAULT;

  return 0;
@@ -537,11 +536,11 @@ static int autofs_root_ioctl(struct inod

  DPRINTK(("autofs_ioctl: cmd = 0x%08x, arg = 0x%08lx, sbi = %p, pgrp =
%u\n",cmd,arg,sbi,process_group(current)));

- if ( _IOC_TYPE(cmd) != _IOC_TYPE(AUTOFS_IOC_FIRST) ||
-      _IOC_NR(cmd) - _IOC_NR(AUTOFS_IOC_FIRST) >= AUTOFS_IOC_COUNT )
+ if (_IOC_TYPE(cmd) != _IOC_TYPE(AUTOFS_IOC_FIRST) ||
+      _IOC_NR(cmd) - _IOC_NR(AUTOFS_IOC_FIRST) >= AUTOFS_IOC_COUNT)
   return -ENOTTY;

- if ( !autofs_oz_mode(sbi) && !capable(CAP_SYS_ADMIN) )
+ if (!autofs_oz_mode(sbi) && !capable(CAP_SYS_ADMIN))
   return -EPERM;

  switch(cmd) {
diff -puN fs/autofs4/inode.c~fix-some-coding-style-errors-in-autofs fs/autofs4/inode.c
--- a/fs/autofs4/inode.c~fix-some-coding-style-errors-in-autofs
+++ a/fs/autofs4/inode.c
@@ -218,8 +218,7 @@ static match_table_t tokens = {
 };

 static int parse_options(char *options, int *pipefd, uid_t *uid, gid_t *gid,
-    pid_t *pgrp, unsigned int *type,
-    int *minproto, int *maxproto)
+  pid_t *pgrp, unsigned int *type, int *minproto, int *maxproto)
 {
  char *p;
  substring_t args[MAX_OPT_ARGS];
@@ -314,7 +313,7 @@ int autofs4_fill_super(struct super_bloc
  struct autofs_info *ino;

  sbi = kmalloc(sizeof(*sbi), GFP_KERNEL);
- if ( !sbi )
+ if (!sbi)
   goto fail_unlock;
  DPRINTK("starting up, sbi = %p",sbi);
```

```
@@ -363,10 +362,9 @@ int autofs4_fill_super(struct super_bloc
  root->d_fsdata = ino;

  /* Can this call block? */
- if (parse_options(data, &pipefd,
-     &root_inode->i_uid, &root_inode->i_gid,
-     &sbi->oz_pgrp, &sbi->type,
-     &sbi->min_proto, &sbi->max_proto)) {
+ if (parse_options(data, &pipefd, &root_inode->i_uid, &root_inode->i_gid,
+     &sbi->oz_pgrp, &sbi->type, &sbi->min_proto,
+     &sbi->max_proto)) {
   printk("autofs: called with bogus options\n");
   goto fail_dput;
  }
@@ -396,11 +394,11 @@ int autofs4_fill_super(struct super_bloc
  DPRINTK("pipe fd = %d, pgrp = %u", pipefd, sbi->oz_pgrp);
  pipe = fget(pipefd);

- if ( !pipe ) {
+ if (!pipe) {
   printk("autofs: could not open pipe file descriptor\n");
   goto fail_dput;
  }
- if ( !pipe->f_op || !pipe->f_op->write )
+ if (!pipe->f_op || !pipe->f_op->write)
   goto fail_fput;
  sbi->pipe = pipe;
  sbi->pipefd = pipefd;
diff -puN fs/autofs4/root.c~fix-some-coding-style-errors-in-autofs fs/autofs4/root.c
--- a/fs/autofs4/root.c~fix-some-coding-style-errors-in-autofs
+++ a/fs/autofs4/root.c
@@ -759,7 +759,7 @@ static int autofs4_dir_unlink(struct ino
  struct autofs_info *p_ino;

  /* This allows root to remove symlinks */
- if ( !autofs4_oz_mode(sbi) && !capable(CAP_SYS_ADMIN) )
+ if (!autofs4_oz_mode(sbi) && !capable(CAP_SYS_ADMIN))
   return -EACCES;

  if (atomic_dec_and_test(&ino->count)) {
@@ -833,7 +833,7 @@ static int autofs4_dir_mkdir(struct inod
  struct autofs_info *p_ino;
  struct inode *inode;

- if ( !autofs4_oz_mode(sbi) )
+ if (!autofs4_oz_mode(sbi))
   return -EACCES;
```

```
  DPRINTK("dentry %p, creating %.*s",
@@ -871,11 +871,11 @@ static inline int autofs4_get_set_timeou
  int rv;
  unsigned long ntimeout;

- if ( (rv = get_user(ntimeout, p)) ||
-      (rv = put_user(sbi->exp_timeout/HZ, p)) )
+ if ((rv = get_user(ntimeout, p)) ||
+      (rv = put_user(sbi->exp_timeout/HZ, p)))
   return rv;

- if ( ntimeout > ULONG_MAX/HZ )
+ if (ntimeout > ULONG_MAX/HZ)
   sbi->exp_timeout = 0;
  else
   sbi->exp_timeout = ntimeout * HZ;
@@ -906,7 +906,7 @@ static inline int autofs4_ask_reghost(st
  DPRINTK("returning %d", sbi->needs_reghost);

  status = put_user(sbi->needs_reghost, p);
- if ( status )
+ if (status)
   return status;

  sbi->needs_reghost = 0;
@@ -975,11 +975,11 @@ static int autofs4_root_ioctl(struct ino
  DPRINTK("cmd = 0x%08x, arg = 0x%08lx, sbi = %p, pgrp = %u",
   cmd,arg,sbi,process_group(current));

- if ( _IOC_TYPE(cmd) != _IOC_TYPE(AUTOFS_IOC_FIRST) ||
-      _IOC_NR(cmd) - _IOC_NR(AUTOFS_IOC_FIRST) >= AUTOFS_IOC_COUNT )
+ if (_IOC_TYPE(cmd) != _IOC_TYPE(AUTOFS_IOC_FIRST) ||
+      _IOC_NR(cmd) - _IOC_NR(AUTOFS_IOC_FIRST) >= AUTOFS_IOC_COUNT)
   return -ENOTTY;

- if ( !autofs4_oz_mode(sbi) && !capable(CAP_SYS_ADMIN) )
+ if (!autofs4_oz_mode(sbi) && !capable(CAP_SYS_ADMIN))
   return -EPERM;

  switch(cmd) {
_
```

Patches currently in -mm which might be from sukadev@us.ibm.com are

origin.patch

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers