

# [PATCH]a tar filesystem for 2.6.10-rc1-mm3

**Source:** <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-11/1961.html>

---

**From:** andyliu ([liudeyan\\_at\\_gmail.com](mailto:liudeyan_at_gmail.com))

**Date:** 11/07/04

Date: Sun, 7 Nov 2004 13:47:38 +0800  
To: linux-kernel@vger.kernel.org

hi

let's think about the way we access the file which contained in a tar file  
may we can untar the whole thing and we find the file we want to access  
or we can use the t option with tar to list all the files in the tar  
and then untar  
the only one file we want to access.

but with the help of the tarfs,we can mount a tar file to some dir and access  
it easily and quickly.it's like the tarfs in mc.

just mount -t tarfs tarfile.tar /dir/to/mnt -o loop  
then access the files easily.

it was written by Kazuto Miyoshi ([kaz@earth.email.ne.jp](mailto:kaz@earth.email.ne.jp)) Hirokazu  
Takahashi ([h-takaha@mub.biglobe.ne.jp](mailto:h-takaha@mub.biglobe.ne.jp)) for linux 2.4.0

and i make it work for linux 2.6.0. now a patch for linux 2.6.10-rc1-mm3

thanks

```
diff -urN linux-2.6.10-rc1/fs/Kconfig linux-2.6.10-rc1-tarfs/fs/Kconfig
--- linux-2.6.10-rc1/fs/Kconfig 2004-11-06 14:49:07.000000000 +0800
+++ linux-2.6.10-rc1-tarfs/fs/Kconfig 2004-11-06 12:29:23.000000000 +0800
@@ -1007,6 +1007,12 @@
```

To compile this as a module, choose M here: the module will be called  
ramfs.

```
+
+config TAR_FS
+tristate "Tar fs support"
+help
+tarfs can help you mount a .tar file. then you can access
the files in the
+tar easily.
```

endmenu

## Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
diff -urN linux-2.6.10-rc1/fs/Makefile linux-2.6.10-rc1-tarfs/fs/Makefile
--- linux-2.6.10-rc1/fs/Makefile 2004-11-06 14:49:07.000000000 +0800
+++ linux-2.6.10-rc1-tarfs/fs/Makefile 2004-11-06 12:23:54.000000000 +0800
@@ -97,3 +97,4 @@
obj-$(CONFIG_HOSTFS) += hostfs/
obj-$(CONFIG_HPPFS) += hppfs/
obj-$(CONFIG_CACHEFS) += cachefs/
+obj-$(CONFIG_TAR_FS) += tarfs/
diff -urN linux-2.6.10-rc1/fs/tarfs/dir.c linux-2.6.10-rc1-tarfs/fs/tarfs/dir.c
--- linux-2.6.10-rc1/fs/tarfs/dir.c 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/dir.c 2004-11-06
12:17:19.000000000 +0800
@@ -0,0 +1,99 @@
+/*
+ * tarfs/dir.c
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of the GNU General Public License as published by the
+ * Free Software Foundation; either version 2 of the License, or (at your
+ * option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
+ * General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software Foundation,
+ * Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * Version: $Id: dir.c,v 1.9 2001/02/25 21:39:03 kaz Exp $
+ *
+ * Copyright (C) 2001
+ * Kazuto Miyoshi (kaz@earth.email.ne.jp)
+ * Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)
+ *
+ * 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.*
+ */
+
+#include <linux/fs.h>
+
+#include "tarfs.h"
+
+static int tarfs_readdir(struct file * filp,
+ void * dirent, filldir_t filldir)
+{
+ struct inode *inode = filp->f_dentry->d_inode;
+ int err;
+ struct tarent *dir_tarent, *ent;
+ int dtype=0;
```

```

+ int count, stored;
+
+ dir_tarent = TARENT(inode);
+
+ message("tarfs: tarfs_readdir (dir_tarent %p, f_pos %ld)\n",
+ dir_tarent, (long)filp->f_pos);
+
+ stored=0;
+
+ /* entry 0: dot */
+ if (filp->f_pos==0){
+ err = filldir(dirent, ".", 1,
+ filp->f_pos, inode->i_ino, DT_DIR);
+ filp->f_pos++;
+ stored=1;
+ goto out;
+ }
+
+ /* entry 1: dot dot */
+ if (filp->f_pos==1){
+ err = filldir(dirent, "..", 2,
+ filp->f_pos, dir_tarent->parent->ino, DT_DIR);
+ filp->f_pos++;
+ stored=1;
+ goto out;
+ }
+
+ /* . and .. is already read, we start count from 2 */
+ ent=dir_tarent->children;
+ for(count=2; count<filp->f_pos && ent!=0; count++){
+ ent=ent->neighbors;
+ }
+
+ if (ent==0)
+ goto out;
+
+ for(; ent!=0; ent=ent->neighbors){
+ if (S_ISDIR(ent->mode))
+ dtype=DT_DIR;
+ else if (S_ISREG(ent->mode))
+ dtype=DT_REG;
+
+ message("tarfs: tarfs_readdir : adding %s\n", ent->name);
+ err = filldir(dirent, ent->name, strlen(ent->name),
+ filp->f_pos, ent->ino, dtype);
+ if (err)
+ goto out;
+
+ stored++;
+ filp->f_pos++;
+ }

```

## Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
+
+ out:
+ message("tarfs: tarfs_readdir() done\n");
+ return stored;
+}
+
+struct file_operations tarfs_dir_operations = {
+ read: generic_read_dir, /* just put error */
+ readdir: tarfs_readdir,
+};
diff -urN linux-2.6.10-rc1/fs/tarfs/file.c
linux-2.6.10-rc1-tarfs/fs/tarfs/file.c
--- linux-2.6.10-rc1/fs/tarfs/file.c 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/file.c 2004-11-06
12:15:25.000000000 +0800
@@ -0,0 +1,75 @@
+/*
+ * tarfs/file.c
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of the GNU General Public License as published by the
+ * Free Software Foundation; either version 2 of the License, or (at your
+ * option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
+ * General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software Foundation,
+ * Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * Version: $Id: file.c,v 1.8 2001/02/25 21:39:03 kaz Exp $
+ *
+ * Copyright (C) 2001
+ * Kazuto Miyoshi (kaz@earth.email.ne.jp)
+ * Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)
+ *
+ * 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.*
+ */
+
+#include <linux/fs.h>
+#include <linux/sched.h>
+#include "tarfs.h"
+
+static loff_t tarfs_file_lseek(
+ struct file *file,
+ loff_t offset,
+ int origin)
```

Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```

+{
+ static int event;
+ struct inode *inode = file->f_dentry->d_inode;
+
+ switch (origin) {
+ case 2:
+ offset += inode->i_size;
+ break;
+ case 1:
+ offset += file->f_pos;
+ }
+ if (offset<0)
+ return -EINVAL;
+ if (((unsigned long long) offset >> 32) != 0) {
+ if (offset > TARFS_MAX_SIZE)
+ return -EINVAL;
+ }
+ if (offset != file->f_pos) {
+ file->f_pos = offset;
+ //file->f_reada = 0;
+ file->f_version = ++event;
+ }
+ return offset;
+}
+
+static int tarfs_open_file (struct inode * inode, struct file * filp)
+{
+ if (!(filp->f_flags & O_LARGEFILE) &&
+ inode->i_size > TARFS_MAX_SIZE)
+ return -EFBIG;
+ return 0;
+}
+
+struct file_operations tarfs_file_operations = {
+ llseek: tarfs_file_lseek,
+ read: generic_file_read,
+ mmap: generic_file_mmap,
+ open: tarfs_open_file,
+};
+
diff -urN linux-2.6.10-rc1/fs/tarfs/inode.c
linux-2.6.10-rc1-tarfs/fs/tarfs/inode.c
--- linux-2.6.10-rc1/fs/tarfs/inode.c 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/inode.c 2004-11-06
12:16:10.000000000 +0800
@@ -0,0 +1,123 @@
+/*
+ * tarfs/inode.c
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of the GNU General Public License as published by the

```

## Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
+ * Free Software Foundation; either version 2 of the License, or (at your
+ * option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
+ * General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software Foundation,
+ * Inc., 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * Version: $Id: inode.c,v 1.13 2001/02/25 21:39:03 kaz Exp $
+ *
+ * Copyright (C) 2001
+ * Kazuto Miyoshi (kaz@earth.email.ne.jp)
+ * Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)
+ *
+ * 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.*
+ */
+
+#include <linux/fs.h>
+#include <linux/buffer_head.h>
+
+#include "tarfs.h"
+
+extern struct address_space_operations tarfs_aops;
+extern struct inode_operations tarfs_symlink_inode_operations;
+void tarfs_read_inode (struct inode * inode)
+{
+ struct tarent *ent;
+
+ printk("tarfs: tarfs_read_inode(ino=%ld)\n", inode->i_ino);
+
+ ent=lookup_tarent(inode->i_sb, inode->i_ino);
+ if (!ent)
+ goto err_out;
+
+ inode->i_mode = ent->mode;
+ message("tarfs: tarfs_read_inode() -> mode %d\n", inode->i_mode);
+ inode->i_uid = ent->uid;
+ inode->i_gid = ent->gid;
+ inode->i_nlink = ent->nlink;
+ inode->i_size = ent->size;
+ inode->i_atime.tv_sec = ent->atime;
+ inode->i_ctime.tv_sec = ent->ctime;
+ inode->i_mtime.tv_sec = ent->mtime;
+ TARENT(inode) = ent;
+
+ if (S_ISREG(inode->i_mode)) {
```

```

+ inode->i_fop = &tarfs_file_operations;
+ inode->i_mapping->a_ops = &tarfs_aops;
+ } else if (S_ISDIR(inode->i_mode)) {
+ inode->i_op = &tarfs_dir_inode_operations;
+ inode->i_fop = &tarfs_dir_operations;
+ } else if (S_ISLNK(inode->i_mode)) {
+ inode->i_op = &tarfs_symlink_inode_operations;
+ } else {
+ /* XXX: dev, fifo, contype etc. */
+ error("tarfs: unsupported file type (i_mode=%d)\n", inode->i_mode);
+ }
+
+ return;
+
+ err_out:
+ make_bad_inode(inode);
+ return;
+ }
+
+ /*
+ * Getblock for tarfs
+ */
+static int tarfs_get_block(struct inode *inode, sector_t iblock,
struct buffer_head *bh_result, int create)
+{
+ if (create){
+ printk("Can not create file on tarfs\n");
+ return -EIO;
+ }
+
+ //bh_result->b_bdev = inode->i_bdev;
+ //bh_result->b_blocknr = TARENT(inode)->pos + iblock;
+ //bh_result->b_state |= (1UL << BH_Mapped);
+ map_bh(bh_result, inode->i_sb, TARENT(inode)->pos + iblock);
+
+ return 0;
+ }
+
+ /*
+ * Read and fill a page.
+ */
+int tarfs_readpage(struct file *file, struct page *page)
+{
+ printk("tarfs_readpage\n");
+ return block_read_full_page(page, tarfs_get_block);
+ }
+
+ struct address_space_operations tarfs_aops = {
+ readpage: tarfs_readpage,
+ };
+

```

## Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
+/* symlinks */
+static int tarfs_readlink(struct dentry *dentry, char *buffer, int buflen)
+{
+ char *s = (char *)TARENT(dentry->d_inode)->linkname;
+ return vfs_readlink(dentry, buffer, buflen, s);
+}
+
+static int tarfs_follow_link(struct dentry *dentry, struct nameidata *nd)
+{
+ char *s = (char *)TARENT(dentry->d_inode)->linkname;
+ return vfs_follow_link(nd, s);
+}
+
+struct inode_operations tarfs_symlink_inode_operations = {
+ readlink: tarfs_readlink,
+ follow_link: tarfs_follow_link,
+};
diff -urN linux-2.6.10-rc1/fs/tarfs/Makefile
linux-2.6.10-rc1-tarfs/fs/tarfs/Makefile
--- linux-2.6.10-rc1/fs/tarfs/Makefile 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/Makefile 2004-11-06
12:02:26.000000000 +0800
@@ -0,0 +1,8 @@
+#
+# Makefile for the linux tarfs-filesystem routines.
+#
+#
+
+obj-$(CONFIG_TAR_FS) += tarfs.o
+
+tarfs-y := dir.o file.o inode.o namei.o super.o tarent.o
diff -urN linux-2.6.10-rc1/fs/tarfs/namei.c
linux-2.6.10-rc1-tarfs/fs/tarfs/namei.c
--- linux-2.6.10-rc1/fs/tarfs/namei.c 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/namei.c 2004-11-06
12:15:24.000000000 +0800
@@ -0,0 +1,52 @@
+/*
+ * tarfs/namei.c
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of the GNU General Public License as published by the
+ * Free Software Foundation; either version 2 of the License, or (at your
+ * option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
+ * General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
```

## Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
+ * along with this program; if not, write to the Free Software Foundation,  
+ * Inc., 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA.  
+ *  
+ * Version: $Id: namei.c,v 1.8 2001/02/25 21:39:03 kaz Exp $  
+ *  
+ * Copyright (C) 2001  
+ * Kazuto Miyoshi (kaz@earth.email.ne.jp)  
+ * Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)  
+ *  
+ * 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.*  
+ *  
+ */  
+  
+#include <linux/fs.h>  
+#include "tarfs.h"  
+  
+static struct dentry *tarfs_lookup(struct inode * dir, struct dentry *dentry)  
+{  
+ struct tarent *ent;  
+  
+ message("tarfs: tarfs_lookup()\n");  
+  
+ for(ent=TARENT(dir)->children; ent!=0; ent=ent->neighbors){  
+ if (!strcmp(ent->name, dentry->d_name.name)){  
+ d_add(dentry, iget(dir->i_sb, ent->ino));  
+ goto out;  
+ }  
+ }  
+ d_add(dentry, NULL);  
+  
+ out:  
+ return NULL;  
+}  
+  
+struct inode_operations tarfs_dir_inode_operations = {  
+ lookup: tarfs_lookup,  
+};  
+  
diff -urN linux-2.6.10-rc1/fs/tarfs/README  
linux-2.6.10-rc1-tarfs/fs/tarfs/README  
--- linux-2.6.10-rc1/fs/tarfs/README 1970-01-01 07:00:00.000000000 +0700  
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/README 2004-11-06  
12:04:37.000000000 +0800  
@@ -0,0 +1,10 @@  
+tarfs was written by  
+Kazuto Miyoshi (kaz@earth.email.ne.jp)  
+Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)  
+  
+in 2001 for linux 2.4.0  
+  
+and I port it to run on Linux 2.6.* in 2004
```

Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
+
+deyan liu liudeyan@gmail.com
+
diff -urN linux-2.6.10-rc1/fs/tarfs/super.c
linux-2.6.10-rc1-tarfs/fs/tarfs/super.c
--- linux-2.6.10-rc1/fs/tarfs/super.c 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/super.c 2004-11-06
14:14:32.000000000 +0800
@@ -0,0 +1,215 @@
+/*
+ * tarfs/super.c
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of the GNU General Public License as published by the
+ * Free Software Foundation; either version 2 of the License, or (at your
+ * option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
+ * General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software Foundation,
+ * Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * Version: $Id: super.c,v 1.18 2001/02/25 21:39:03 kaz Exp $
+ *
+ * Copyright (C) 2001
+ * Kazuto Miyoshi (kaz@earth.email.ne.jp)
+ * Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)
+ *
+ * 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.*
+ */
+#include <linux/fs.h>
+#include <linux/module.h>
+#include <linux/init.h>
+#include <linux/statfs.h>
+#include <linux/buffer_head.h>
+
+#include "tarfs.h"
+
+void tarfs_put_super (struct super_block * sb)
+{
+
+ printk("tarfs : tarfs_put_super\n");
+
+
+ /* Drop all tarent hash table */
+ delete_all_tarents_and_hash(sb);
```

```

+
+ /* Drop fs dependent sb data */
+ if (TARSB(sb)){
+ kfree(TARSB(sb));
+ }
+ }
+
+int tarfs_statfs (struct super_block * sb, struct kstatfs * buf)
+{
+ message("tarfs : tarfs_statfs\n");
+
+
+ buf->f_type = TARFS_SUPER_MAGIC;
+ buf->f_bsize = sb->s_blocksize;
+ buf->f_blocks = TARSB(sb)->s_files;
+ buf->f_bfree = 0;
+ buf->f_bavail = 0;
+ buf->f_files = TARSB(sb)->s_files;
+ buf->f_ffree = 0;
+ buf->f_namelen = TARFS_NAME_LEN;
+ return 0;
+ }
+
+int tarfs_remount (struct super_block * sb, int * flags, char * data)
+{
+ message("tarfs : tarfs_remount\n");
+
+
+ if (!( *flags & MS_RDONLY)){
+ /* Remount as RDWR */
+ return -EINVAL;
+ }
+
+
+ return 0;
+ }
+
+static int check_magic(struct super_block *sb)
+{
+ struct buffer_head *bh;
+ struct posix_header *ph;
+ extern int allow_v7_format;
+ int ok=0;
+
+
+ if (allow_v7_format){
+ /* Dangerous when corrupted/non-tar file is specified. */
+ return 1;
+ }
+
+
+ bh = sb_bread(sb, 0);
+ if (!bh)
+ return 0;
+
+
+ ph = (struct posix_header*)((char *)bh->b_data);

```

Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```

+ message("magic %s\n", ph->magic);
+ ok = !strncmp(ph->magic, "ustar", 5);
+ brelse(bh);
+
+ return ok;
+}
+
+
+extern struct super_operations tarfs_sops;
+
+static int tarfs_fill_super(struct super_block *sb, void *data, int silent)
+{
+
+ sb_set_blocksize(sb, TAR_BLOCKSIZE);
+
+ printk("tarfs: tarfs_fill_super()\n");
+ message("tarfs: mount flags: %ld\n", sb->s_flags);
+ //sb->s_blocksize_bits = 9; /* Log2(TAR_BLOCKSIZE) */
+ //sb->s_blocksize = TAR_BLOCKSIZE;
+
+ if (!check_magic(sb)){
+ printk("tarfs: Not a tar file, or old v7 format\n");
+ goto err_out;
+ }
+
+ /* Alloc tarfs dependent data */
+ TARSB(sb) = kmalloc(sizeof(struct tarfs_sb_info), GFP_KERNEL);
+ if (!TARSB(sb)){
+ printk("tarfs: Can not allocate fs dependent sb data\n");
+ goto err_out;
+ }
+
+ TARSB(sb)->parsed=0;
+ TARSB(sb)->ihash=(struct tarent**)
+ kmalloc(sizeof(struct tarent *[TARENT_HASHSIZE]), GFP_KERNEL);
+ if (!TARSB(sb)->ihash)
+ goto err_out;
+ memset(TARSB(sb)->ihash, 0, sizeof(struct tarent *[TARENT_HASHSIZE]));
+
+ TARSB(sb)->s_files=0;
+ TARSB(sb)->s_blocks=0;
+ TARSB(sb)->s_maxino=1;
+
+ sb->s_op = &tarfs_sops;
+ sb->s_root = d_alloc_root(iget(sb, TARFS_ROOT_INO));
+ if (!sb->s_root) {
+ goto err_out;
+ }
+
+ /* Read-only fs */
+ sb->s_flags = MS_RDONLY;

```

```

+ sb->s_dirt = 0;
+ return 0;
+
+ err_out:
+ if (TARSB(sb) && TARSB(sb)->ihash){
+ kfree(TARSB(sb)->ihash);
+ }
+ if (TARSB(sb)){
+ kfree(TARSB(sb));
+ }
+ printk("tarfs:fillsuper error\n");
+ return 1;
+}
+
+static struct super_block * tarfs_get_super (struct file_system_type
* fs_type,
+ int flags, const char *dev_name, void * data)
+{
+ return get_sb_bdev(fs_type, flags, dev_name, data, tarfs_fill_super);
+}
+
+extern void tarfs_read_inode (struct inode * inode);
+extern void tarfs_write_inode (struct inode * inode, int wait);
+extern void tarfs_put_inode (struct inode * inode);
+extern void tarfs_delete_inode (struct inode * inode);
+extern void tarfs_truncate (struct inode * inode);
+
+static struct super_operations tarfs_sops = {
+ read_inode: tarfs_read_inode,
+ put_super: tarfs_put_super,
+ statfs: tarfs_statfs,
+ remount_fs: tarfs_remount,
+};
+
+//static DECLARE_FSTYPE_DEV(tar_fs_type, "tarfs", tarfs_read_super);
+
+static struct file_system_type tar_fs_type = {
+ owner: THIS_MODULE,
+ name: "tarfs",
+ get_sb: tarfs_get_super,
+ kill_sb: kill_block_super,
+ fs_flags: FS_REQUIRES_DEV,
+};
+
+/* Implemented as module */
+
+MODULE_DESCRIPTION("Linux 2.4 Design & Implementation sample module");
+MODULE_AUTHOR("Linux Japan magazine");
+
+int tarfs_debug=0;
+int allow_v7_format=0;

```

Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
+MODULE_PARM(tarfs_debug, "i");
+MODULE_PARM_DESC(debug, "Tarfs debug output flag");
+MODULE_PARM(allow_v7_format, "i");
+MODULE_PARM_DESC(allow_v7_format, "Allow v7 format (no magic check)");
+
+static int __init init_tar_fs(void)
+{
+ printk(KERN_INFO "tarfs: init_tar_fs (debug=%d)\n", tarfs_debug);
+ return register_filesystem(&tar_fs_type);
+}
+
+static void __exit exit_tar_fs(void)
+{
+ printk(KERN_INFO "tarfs: exit_tar_fs\n");
+ unregister_filesystem(&tar_fs_type);
+}
+
+//EXPORT_NO_SYMBOLS;
+
+module_init(init_tar_fs)
+module_exit(exit_tar_fs)
diff -urN linux-2.6.10-rc1/fs/tarfs/tarent.c
linux-2.6.10-rc1-tarfs/fs/tarfs/tarent.c
--- linux-2.6.10-rc1/fs/tarfs/tarent.c 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/tarent.c 2004-11-06
12:16:32.000000000 +0800
@@ -0,0 +1,608 @@
+/*
+ * tarfs/tarent.c
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of the GNU General Public License as published by the
+ * Free Software Foundation; either version 2 of the License, or (at your
+ * option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
+ * General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software Foundation,
+ * Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * Version: $Id: tarent.c,v 1.18 2001/02/25 21:39:03 kaz Exp $
+ *
+ * Copyright (C) 2001
+ * Kazuto Miyoshi (kaz@earth.email.ne.jp)
+ * Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)
+ *
+ * 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.*
```

```

+ *
+ */
+
+#include <linux/fs.h>
+#include <linux/ctype.h>
+#include <asm/current.h>
+#include <linux/buffer_head.h>
+
+#include "tarfs.h"
+
+/*
+ * Octal strtoul for tarfs.
+ * tar size, mtime, etc. sometimes NOT null terminated.
+ */
+static unsigned long getval8(char *p, size_t siz)
+{
+ char buf[512];
+ unsigned long val;
+ int i;
+
+ if (siz>512 || siz <=0){
+ error("tarfs: zero or too long siz for getval8\n");
+ }
+
+ for(i=0; i<siz; i++){
+ if (!isspace(p[i])) /* '\0' is control character */
+ break;
+ }
+
+ memcpy(buf, p+i, siz-i);
+ buf[siz-i]='\0';
+
+ val=simple_strtoul(buf, NULL, 8);
+ message("getval8 [%s] %ld\n", buf, val);
+
+ return val;
+}
+
+/*
+ * Feed a new ino
+ */
+static unsigned long get_new_ino(struct super_block *sb)
+{
+ message("tarfs: feeding ino %ld\n", TARSB(sb)->s_maxino);
+ return ++(TARSB(sb)->s_maxino);
+}
+
+/*
+ * Add tarent to hash table
+ */
+static void hash_tarent(struct super_block *sb, struct tarent *t,

```

```

unsigned long ino)
+{
+ struct tarent **tentp = TARSB(sb)->ihash+ino%TARENT_HASHSIZE;
+
+ /*
+ * Adding prior to the exiting entries.
+ * Duplicated entries are hidden by the new one.
+ */
+ t->next_hash = *tentp;
+ *tentp = t;
+ }
+
+ /*
+ * Lookup tarent by ino
+ */
+struct tarent *lookup_tarent(struct super_block *sb, unsigned long ino)
+{
+ struct tarent *ent;
+
+ if (!TARSB(sb)->parsed){
+ message("tarfs: !!! sb->parsed == 0\n");
+ if (parse_tar(sb))
+ goto err_out;
+ TARSB(sb)->parsed=1;
+ }
+
+ if (ino < 2){
+ error("tarfs: lookup_tarent() no such inode %ld\n", ino);
+ return 0;
+ }
+
+ for(ent=TARSB(sb)->ihash[ino%TARENT_HASHSIZE];
+ ent!=0;
+ ent=NEXT_HASH(ent)){
+ if (ent->ino==ino){
+ return (ent->hardlinked ? ent->hardlinked : ent);
+ }
+ }
+
+ err_out:
+ return 0;
+ }
+
+static struct tarent *add_tarent(struct super_block *, unsigned long,
+ struct posix_header *, char *);
+
+ /*
+ * Fill contents of 'te' according to the posix header 'ph'
+ */
+static void set_tarent(struct super_block *sb,
+ struct posix_header *ph, struct tarent *te, int blk)

```

```

+{
+ unsigned long mode;
+ time_t mtime;
+ struct tarent *link;
+
+ message("set_tarent (blk=%d)\n", blk);
+
+ mode=getval8(ph->mode, sizeof(ph->mode));
+ switch(ph->typeflag){
+ case TARFS_DIRTYTYPE:
+ /* Directory */
+ te->mode = S_IFDIR|(mode & TARFS_MODEMASK);
+ /*
+ * Do not overwrite size/nlink, because get_new_tarent()
+ * and following add_to_parent() correctly count up
+ * these value.
+ */
+ break;
+
+ case TARFS_LNKTYPE:
+ /* Hardlinks */
+ te->linkname = kmalloc(strlen(ph->linkname)+1, GFP_KERNEL);
+ if (!te->linkname){
+ error("tarfs: Can not assign linkname\n");
+ te->linkname = 0;
+ }
+ strcpy(te->linkname, ph->linkname);
+ message("hardlink %s->%s\n", te->name, te->linkname);
+
+ /* Lookup (preassign) original entry */
+ link=add_tarent(sb, 0, 0, te->linkname);
+ te->hardlinked = link;
+ break;
+
+ case TARFS_SYMTYPE:
+ /* Symlinks */
+ te->mode = S_IFLNK|(mode & TARFS_MODEMASK);
+ te->nlink = 1;
+ te->linkname = kmalloc(strlen(ph->linkname)+1, GFP_KERNEL);
+ if (!te->linkname){
+ error("tarfs: Can not assign linkname\n");
+ te->linkname = 0;
+ }
+ strcpy(te->linkname, ph->linkname);
+ message("symlink %s->%s\n", te->name, te->linkname);
+ break;
+
+ case TARFS_REGTYPE:
+ case TARFS_AREGTYPE:
+ /* Regular files */
+

```

```

+ te->mode = S_IFREG|(mode & TARFS_MODEMASK);
+ /*
+ * Overwrite size and nlink, because at the first time,
+ * tarent is created as directory
+ */
+ te->size = getval8(ph->size, sizeof(ph->size));
+ te->nlink = 1;
+ break;
+
+ default:
+ /* Unsupported types, dev, fifo */
+ error("Unsupported typeflag %d (%c) blk %d, ignored\n",
+ ph->typeflag, ph->typeflag, blk);
+ te->mode = S_IFREG|(mode & TARFS_MODEMASK);
+ te->size = getval8(ph->size, sizeof(ph->size));
+ te->nlink = 1;
+ break;
+ }
+
+ mtime=getval8(ph->mtime, sizeof(ph->mtime));
+ te->mtime = mtime;
+ te->atime = mtime;
+ te->ctime = mtime;
+
+ te->uid = getval8(ph->uid, sizeof(ph->uid));
+ te->gid = getval8(ph->gid, sizeof(ph->gid));
+ te->pos = blk;
+ }
+
+ /*
+ * Alloacte a new tarent (default value)
+ */
+static struct tarent *get_new_tarent(char *n, int len, unsigned long ino)
+{
+ struct tarent *te;
+
+ te = (struct tarent *)kmalloc(sizeof(struct tarent), GFP_KERNEL);
+ if (!te)
+ goto err_out;
+ te->name=kmalloc(len+1, GFP_KERNEL);
+ if (!te->name)
+ goto err_out;
+
+ strcpy(te->name, n);
+
+ te->linkname=0;
+
+ te->ino=ino;
+ te->children=te->neighbors=0;
+ te->parent=te;
+
+

```

```

+ /*
+ * Default value for the directories which IS NOT contained
+ * in the tar file.
+ * For the files contained in the tarfile,
+ * set_tarent() overwrites these values.
+ */
+ te->mtime = 0;
+ te->atime = 0;
+ te->ctime = 0;
+
+ //te->uid = current->fsuid;
+ //te->gid = current->fsgid;
+ te->pos = 0;
+
+ te->nlink = te->size = 2;
+ te->mode=S_IFDIR|0777;
+
+ te->hardlinked = 0;
+
+ return te;
+
+ err_out:
+ if (te && te->name)
+ kfree(te->name);
+ if (te)
+ kfree(te);
+
+ return 0;
+}
+
+static void delete_tarent(struct tarent *te)
+{
+ if (te){
+ if (te->name){
+ kfree(te->name);
+ }
+ kfree(te);
+ }
+}
+
+/*
+ * Link c to p as a child tarent
+ */
+static void add_to_parent(struct tarent *p, struct tarent *c)
+{
+ message("tarfs: add_to_parent %p %p\n", p, c);
+
+ if (!(p && S_ISDIR(p->mode)))
+ BUG();
+
+ if (c==0)

```

```

+ BUG();
+
+ c->parent = p;
+ c->neighbors = p->children;
+ p->children = c;
+ p->size+=1;
+ p->nlink+=1;
+}
+
+static struct tarent *lookup_child_and_chop(struct tarent *p, char *n,
+ char **next, int *len)
+{
+ char *c;
+ struct tarent *pp;
+ int i;
+
+ if (p==0)
+ BUG();
+
+ message("tarfs: lookup_child_and_chop %p %p\n", p, n);
+ if (n){
+ message("tarfs: name = %s\n", n);
+ }
+
+ /* chop */
+ for(c=n, i=0; ; i++, c++){
+ if (*c=='/'){
+ *c=0;
+ *next=c+1;
+ break;
+ }
+ if (*c=='\0'){
+ *next=0;
+ break;
+ }
+ }
+
+ message("tarfs: len %d\n", i);
+ *len=i;
+
+ for(pp=p->children; pp!=0; pp=pp->neighbors){
+ message("pp %p\n", pp);
+ if (pp==0){
+ message("pp==0 BUG!\n");
+ break;
+ }else if (pp->name==0){
+ message("pp->name==0. BUG!\n");
+ break;
+ }else{
+ message("comparing %s to %s\n", pp->name, n);
+ }
+ }

```

```

+ if (!strcmp(pp->name, n)){
+ message("tarfs: lookup_child_and_chop found child\n");
+ return pp;
+ }
+ }
+ message("tarfs: lookup_child_and_chop found NO child\n");
+ return 0;
+}
+
+static struct tarent *add_until_slash(struct super_block *sb,
+ struct tarent *p, char *n, char **next)
+{
+ struct tarent *te;
+ int len;
+ unsigned long ino;
+
+ message("add_until_slash (%s)\n", n);
+ te = lookup_child_and_chop(p, n, next, &len);
+ if (te){
+ return te;
+ }
+
+ ino=get_new_ino(sb);
+
+ te = get_new_tarent(n, len, ino);
+ if (!te)
+ goto err_out;
+
+ hash_tarent(sb, te, ino);
+ add_to_parent(p, te);
+
+ return te;
+
+ err_out:
+ if (te)
+ delete_tarent(te);
+ return 0;
+}
+
+static char *skip_slash(char *n)
+{
+ for(; *n=='/'; n++)
+ ;
+
+ return n;
+}
+
+/*
+ * Count tar header length from 'blk'
+ */
+static unsigned long count_header_blocks(struct super_block *sb,

```

```

+ unsigned long blk, int *fatal)
+ {
+ struct buffer_head *bh;
+ int i;
+ int blocks=0;
+ int typeflag, isextended;
+
+ bh = sb_bread(sb, blk);
+ if (!bh){
+ error("tarfs: failed to read headers\n");
+ goto err_out;
+ }
+
+ blocks++;
+ typeflag = ((struct posix_header *) (bh->b_data))->typeflag;
+ brelse(bh);
+
+ switch(typeflag){
+ case TARFS_REGTYPE:
+ case TARFS_AREGTYPE:
+ case TARFS_LNKTYPE:
+ case TARFS_SYMTYPE:
+ case TARFS_CHRTYPE:
+ case TARFS_BLKTYPE:
+ case TARFS_DIRTYPE:
+ case TARFS_FIFOTYPE:
+ case TARFS_CONTTYPE:
+ /* Posix header only */
+ break;
+ case TARFS_GNU_DUMPDIR:
+ case TARFS_GNU_LONGLINK:
+ case TARFS_GNU_LONGNAME:
+ case TARFS_GNU_MULTIVOL:
+ case TARFS_GNU_NAMES:
+ case TARFS_GNU_SPARSE:
+ case TARFS_GNU_VOLHDR:
+ /* Gnu extra header exists */
+ printk("Encountering guu extra header\n");
+ bh = sb_bread(sb, blk+1);
+ if (!bh){
+ error("tarfs: failed to read extra headers\n");
+ goto err_out;
+ }
+ blocks++;
+ isextended = ((struct extra_header *) (bh->b_data))->isextended;
+ brelse(bh);
+ if (isextended)
+ goto out;
+
+ /* Gnu sparse header exists */
+ printk("Encountering guu sparce header\n");

```

```

+ for(i=0;;i++, blocks++){
+ bh = sb_bread(sb, blk+2+i);
+ if (!bh){
+ error("tarfs: failed to read extra headers\n");
+ goto err_out;
+ }
+ isextended = ((struct sparse_header *) (bh->b_data))->isextended;
+ brelse(bh);
+ if (isextended)
+ break;
+ }
+ break;
+ default:
+ error("tarfs: invalid or unsupported typeflag. Headers can not be
skipped correctly.\n");
+ goto err_out;
+ }
+
+ out:
+ message("non fatal %d\n", blocks);
+ *fatal=0;
+ return blocks;
+
+ err_out:
+ message("fatal\n");
+ *fatal=1;
+ return 0;
+ }
+
+ /*
+ * Add tarent
+ * ph != 0, name==0 : normal use.
+ * ph == 0, name!=0 : just pre-assign a entry (for hardlink)
+ */
+static struct tarent *add_tarent(struct super_block *sb, unsigned long blk,
+ struct posix_header *ph, char *name)
+{
+ char *n=0;
+ struct tarent *te=0;
+ struct tarent *parent;
+
+ /* lookup_tarent can not be used here */
+ parent = TARSB(sb)->root_tarent;
+
+ n = (name ? name : ph->name);
+
+ message("Adding tarentry [%s]\n", ph->name);
+
+ for(;;){
+ if (n==0){
+ message("tarfs: no more components(1).\n");

```

```

+ goto out;
+ }
+
+ n = skip_slash(n);
+ if (*n=='\0'){
+ message("tarfs: no more components(2).\n");
+ goto out;
+ }
+ if ( *n=='.' && (*(n+1)=='/' || *(n+1)=='\0' )){
+ message("tarfs: skipping dot\n");
+ n++;
+ continue;
+ }
+ if ( *n=='.' && *(n+1)=='.' && (*(n+2)=='/' || *(n+2)=='\0' )){
+ n+=2;
+ if (parent==TARSB(sb)->root_tarent){
+ message("tarfs: Oops going higher than root, ignored.\n");
+
+ /* Drop this entry */
+ goto err_out;
+ }
+ parent=parent->parent;
+
+ continue;
+ }
+
+ te=add_until_slash(sb, parent, n, &n);
+ parent = te;
+ }
+
+ out:
+ if (ph)
+ set_tarent(sb, ph, parent, blk);
+
+ err_out:
+ return parent;
+ }
+
+ /*
+ * Delect all tarents and hash.
+ */
+ void delete_all_tarents_and_hash(struct super_block *sb)
+ {
+ struct tarent *te, **tp;
+ int i;
+
+ if ( (tp=TARSB(sb)->ihash)==0 )
+ return;
+
+ for(i=0; i<TARENT_HASHSIZE; i++){
+ for(te=*(tp+i); te!=0; te=NEXT_HASH(te))

```

```

+ delete_tarent(te);
+ }
+ kfree(TARSB(sb)->ihash);
+}
+
+/*
+ * Parse tar archive and gather necessary information
+ */
+static int parse_tar(struct super_block *sb)
+{
+ struct buffer_head *bh;
+ struct tarent **tent=0, *root;
+ int tarblk, fatal;
+ struct posix_header *ph;
+ //dev_t dev;
+ unsigned long size, headers;
+
+ //dev = sb->s_dev;
+ //printk("tarfs: parse start (%d,%d)\n", MAJOR(dev), MINOR(dev));
+
+ /* create dummy root */
+ root=get_new_tarent("root", 4, 2);
+ if (!root)
+ goto err_out;
+ hash_tarent(sb, root, get_new_ino(sb) /* 2 */);
+ root->mode=S_IFDIR|0777;
+ TARSB(sb)->root_tarent=root;
+
+ tarblk=0;
+ for(;;){
+ bh = sb_bread(sb, tarblk);
+ if (!bh){
+ /* XXX: disk error or end-of-device */
+ break;
+ }
+
+ ph = (struct posix_header*)((char *)bh->b_data);
+ if (ph->name[0] == '\0'){
+ brelse(bh);
+ break;
+ }
+
+ /* Skip headers */
+ headers=count_header_blocks(sb, tarblk, &fatal);
+ if (fatal){
+ error("Encountering unsupported typeflag blk %d, abort parsing\n",
+ tarblk);
+ break;
+ }
+
+ tarblk+=headers;

```

Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```

+ add_tarent(sb, tarblk, ph, 0);
+
+ /* Skip contents of the file */
+ size=getval8(ph->size, sizeof(ph->size));
+ tarblk += (size+TAR_BLOCKSIZE-1)/TAR_BLOCKSIZE;
+
+ brelse(bh);
+ }
+
+ printk("tarfs: parse done (%ld inodes created)\n",
+ TARSB(sb)->s_maxino-2);
+
+ /* XXX: Poor statistics info */
+ TARSB(sb)->s_files = TARSB(sb)->s_maxino-2;
+ TARSB(sb)->s_blocks = tarblk;
+
+ return 0;
+
+ err_out:
+ if (tent)
+ *tent = 0;
+ if (root)
+ delete_tarent(root);
+ return -1;
+}
diff -urN linux-2.6.10-rc1/fs/tarfs/tarfs.h
linux-2.6.10-rc1-tarfs/fs/tarfs/tarfs.h
--- linux-2.6.10-rc1/fs/tarfs/tarfs.h 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/tarfs.h 2004-11-06
12:16:56.000000000 +0800
@@ -0,0 +1,118 @@
+/*
+ * tarfs/tarfs.h
+ *
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of the GNU General Public License as published by the
+ * Free Software Foundation; either version 2 of the License, or (at your
+ * option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
+ * General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software Foundation,
+ * Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
+ *
+ * Version: $Id: tarfs.h,v 1.13 2001/02/25 21:39:03 kaz Exp $
+ *
+ * Copyright (C) 2001

```

```

+ * Kazuto Miyoshi (kaz@earth.email.ne.jp)
+ * Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)
+ *
+ * 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.*
+ *
+ */
+
+#include "tarinf.h"
+
+/* Debug support */
+extern int tarfs_debug;
+#define message if (tarfs_debug) printk
+#define error printk
+
+/* 'tafs' in little endian */
+#define TARFS_SUPER_MAGIC 0x73666174
+
+/* Size, root ino, etc. */
+#define TAR_BLOCKSIZE 512
+#define TARFS_NAME_LEN 255
+#define TARFS_MAX_SIZE ((unsigned long)0x7fffffff)
+#define TARFS_ROOT_INO 2
+
+#define TARENT(inode) ((struct tarent*)((inode)->u.generic_ip))
+
+/* Each file */
+struct tarent
+{
+ /* File attributes */
+ umode_t mode;
+ uid_t uid;
+ gid_t gid;
+ loff_t size;
+ nlink_t nlink;
+ time_t atime;
+ time_t ctime;
+ time_t mtime;
+
+ /* Offset from start of the archive */
+ unsigned long pos;
+
+ /* File name */
+ char *name;
+
+ /* Symlink */
+ char *linkname;
+
+ /* Inodo nr assigned to the file */
+ int ino;
+
+ /* Parent directory, link to children, neighbors */

```

```

+ struct tarent *parent;
+ struct tarent *children;
+ struct tarent *neighbors;
+
+ /* Link to hash */
+ struct tarent *next_hash;
+
+ /* Link to hardlinked original */
+ struct tarent *hardlinked;
+ };
+
+ #define TARENT_HASHSIZE 1024
+ #define NEXT_HASH(ent) ((ent)->next_hash)
+
+ /* Additional information for tarfs superblock */
+ struct tarfs_sb_info {
+ /* 1 if already have archive information */
+ int parsed;
+
+ /* Root tarent */
+ struct tarent *root_tarent;
+
+ /* Hash table */
+ struct tarent **ihash;
+
+ /* Information for statfs */
+ unsigned long s_files;
+ unsigned long s_blocks;
+
+ /* Max inode assigned */
+ unsigned long s_maxino;
+ };
+
+ #define TARSB(sb) ((struct tarfs_sb_info*)((sb)->s_fs_info))
+
+ /* Prototypes */
+ struct tarent *lookup_tarent(struct super_block *sb, unsigned long ino);
+ extern struct inode_operations tarfs_dir_inode_operations;
+ extern struct inode_operations tarfs_file_inode_operations;
+ extern struct file_operations tarfs_file_operations;
+ extern struct file_operations tarfs_dir_operations;
+ int parse_tar(struct super_block *sb);
+ void delete_all_tarents_and_hash(struct super_block *sb);
+ ssize_t tarfs_read(struct inode *inode,
+ char * buf, size_t count, loff_t *ppos);
+ int tarfs_writepage(struct page *page);
+ int tarfs_readpage(struct file *file, struct page *page);
diff -urN linux-2.6.10-rc1/fs/tarfs/tarinf.h
linux-2.6.10-rc1-tarfs/fs/tarfs/tarinf.h
--- linux-2.6.10-rc1/fs/tarfs/tarinf.h 1970-01-01 07:00:00.000000000 +0700
+++ linux-2.6.10-rc1-tarfs/fs/tarfs/tarinf.h 2004-11-06

```

12:17:02.000000000 +0800

@@ -0,0 +1,115 @@

+/\*

+ \* tarfs/tarinf.h

+ \*

+ \* This program is free software; you can redistribute it and/or modify it  
+ \* under the terms of the GNU General Public License as published by the  
+ \* Free Software Foundation; either version 2 of the License, or (at your  
+ \* option) any later version.

+ \*

+ \* This program is distributed in the hope that it will be useful, but  
+ \* WITHOUT ANY WARRANTY; without even the implied warranty of  
+ \* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
+ \* General Public License for more details.

+ \*

+ \* You should have received a copy of the GNU General Public License  
+ \* along with this program; if not, write to the Free Software Foundation,  
+ \* Inc., 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA.

+ \*

+ \* Version: \$Id: tarinf.h,v 1.3 2001/02/25 21:39:03 kaz Exp \$

+ \*

+ \* Original file tar-1.13 src/tar.h

+ \* Modified by

+ \* Kazuto Miyoshi (kaz@earth.email.ne.jp)

+ \* Hirokazu Takahashi (h-takaha@mub.biglobe.ne.jp)

+ \*

+ \* 2004 deyan liu (liudeyan@gmail.com) make it work in Linux 2.6.\*

+ \*

+ \*/

+

+/\* Information in tar archive \*/

+

+#define TARFS\_REGTYPE '0'

+#define TARFS\_AREGTYPE '\0'

+#define TARFS\_LNKTYPE '1'

+#define TARFS\_SYMTYPE '2'

+#define TARFS\_CHRTYPE '3'

+#define TARFS\_BLKTYPE '4'

+#define TARFS\_DIRTYPE '5'

+#define TARFS\_FIFOTYPE '6'

+#define TARFS\_CONTTYPE '7'

+

+#define TARFS\_GNU\_DUMPDIR 'D'

+#define TARFS\_GNU\_LONGLINK 'K'

+#define TARFS\_GNU\_LONGNAME 'L'

+#define TARFS\_GNU\_MULTIVOL 'M'

+#define TARFS\_GNU\_NAMES 'N'

+#define TARFS\_GNU\_SPARSE 'S'

+#define TARFS\_GNU\_VOLHDR 'V'

+

+/\* Mode field \*/

```

+#define TARFS_SUID 04000
+#define TARFS_SGID 02000
+#define TARFS_SVTX 01000
+#define TARFS_MODEMASK 00777
+
+struct posix_header
+{
+ char name[100];
+ char mode[8];
+ char uid[8];
+ char gid[8];
+ char size[12];
+ char mtime[12];
+ char chksum[8];
+ char typeflag;
+ char linkname[100];
+ char magic[6];
+ char version[2];
+ char uname[32];
+ char gname[32];
+ char devmajor[8];
+ char devminor[8];
+ char prefix[155];
+};
+
+#define SPARSEES_IN_EXTRA_HEADER 16
+#define SPARSEES_IN_OLDGNU_HEADER 4
+#define SPARSEES_IN_SPARSE_HEADER 21
+
+struct sparse
+{
+ char offset[12];
+ char numbytes[12];
+};
+
+/* GNU extra header */
+struct extra_header
+{
+ char atime[12];
+ char ctime[12];
+ char offset[12];
+ char realsize[12];
+ char longnames[4];
+ char unused_pad1[68];
+ struct sparse sp[SPARSEES_IN_EXTRA_HEADER];
+ char isextended;
+};
+
+struct sparse_header
+{
+ struct sparse sp[SPARSEES_IN_SPARSE_HEADER];

```

Linux-Kernel: [PATCH]a tar filesystem for 2.6.10-rc1-mm3

```
+ char isextended;
+};
+
+struct oldgnu_header
+{
+ char unused_pad1[345];
+ char atime[12];
+ char ctime[12];
+ char offset[12];
+ char longnames[4];
+ char unused_pad2;
+ struct sparse sp[SPARSESES_IN_OLDGNU_HEADER];
+ char isextended;
+ char realsize[12];
+};
```

--

Yours andyliu

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>