Operating Systems Lab 10

05/04/2023

200030041 - Pavan Kumar V Patil
200010022 - Karthik J Ponarkar

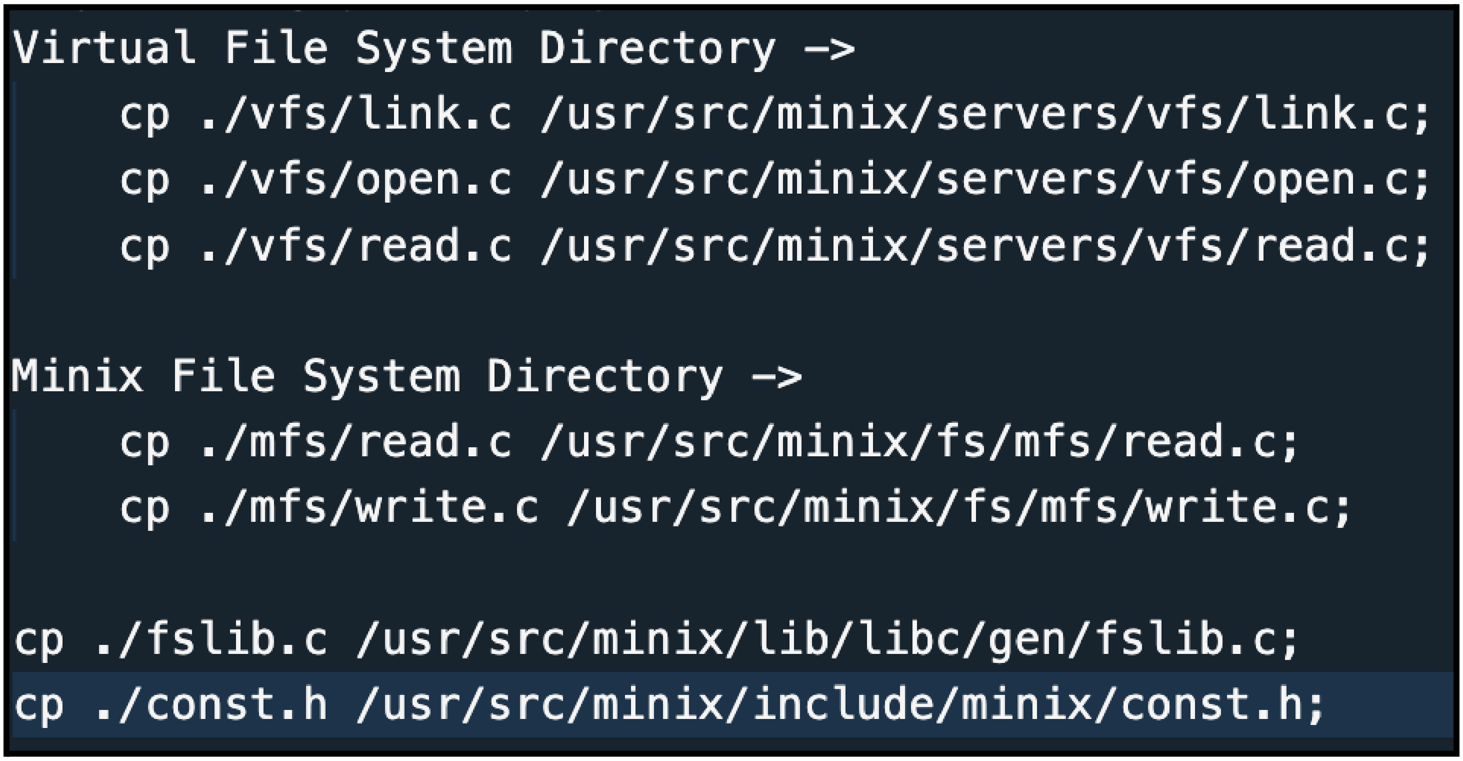
Goal:

In this lab, our task is to implement immediate files in the Minix File System, for files of sizes up to 32 bytes. Implement only for the file system mounted at /home.

Some of the broad functionalities that need to be implemented are:

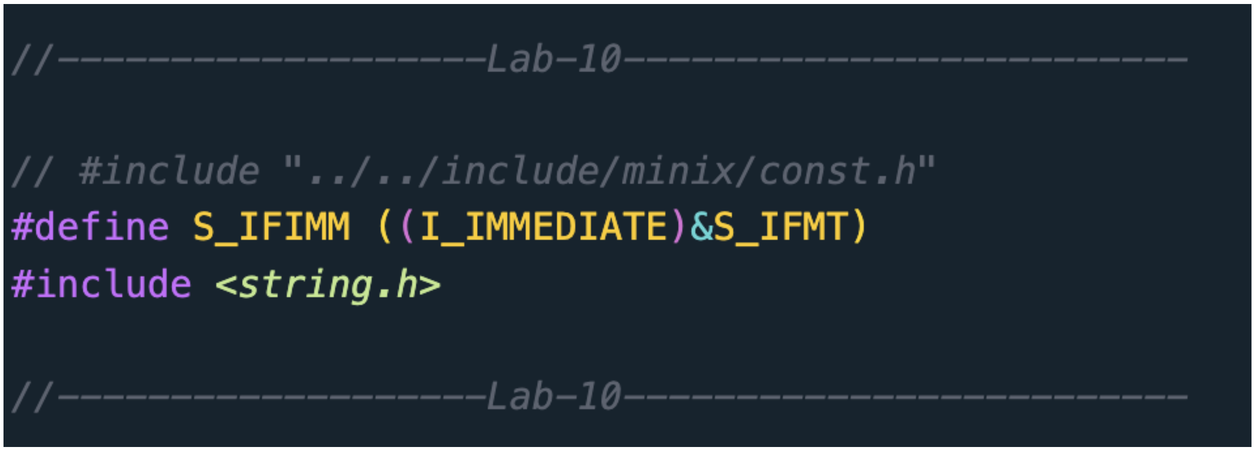
- **File creation**: Start by creating the file as an immediate one. When a file grows beyond 32B, it will be made a regular file.
- **File read**: If it is an immediate file, respond with the inode structure contents. If not, follow the default behavior of looking up zones.
- **File write**: Similar to reading, one must ensure that if we want to write to the inode structure, then the new file size is still within 32 bytes. When a regular file shrinks to less than 32 bytes, there is no need to come back to immediate mode.
- **File delete**: Deleting immediate files does not require any handling of zones.

Files modified for implementation of the above features:

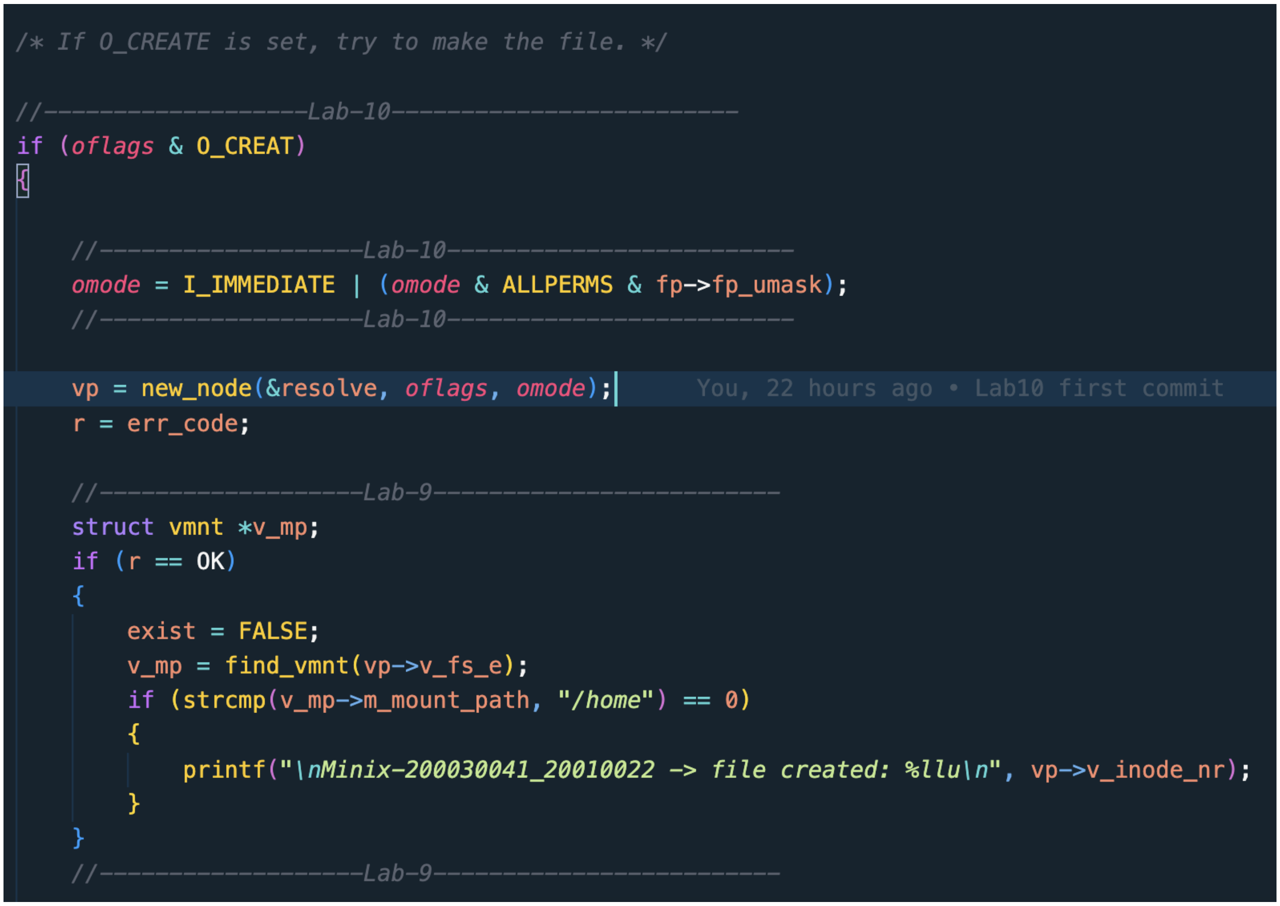


```
Virtual File System Directory ->
    cp ./vfs/link.c /usr/src/minix/servers/vfs/link.c;
    cp ./vfs/open.c /usr/src/minix/servers/vfs/open.c;
    cp ./vfs/read.c /usr/src/minix/servers/vfs/read.c;

Minix File System Directory ->
    cp ./mfs/read.c /usr/src/minix/fs/mfs/read.c;
    cp ./mfs/write.c /usr/src/minix/fs/mfs/write.c;

cp ./fslib.c /usr/src/minix/lib/libc/gen/fslib.c;
cp ./const.h /usr/src/minix/include/minix/const.h;
```

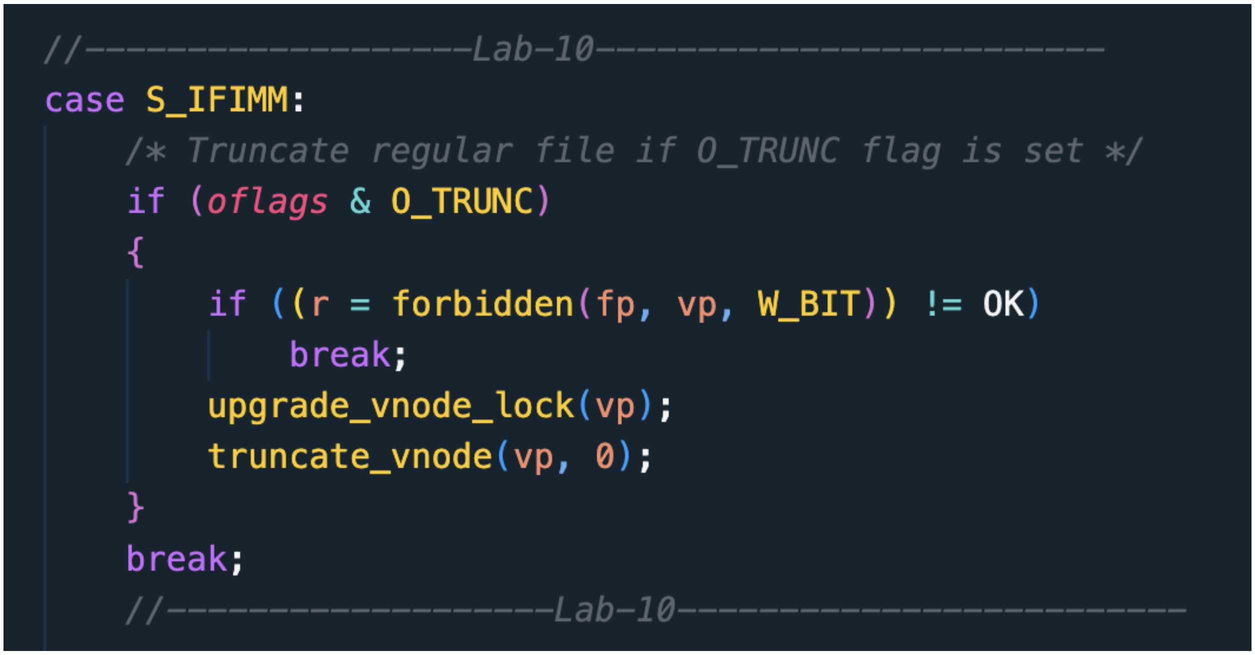
vfs/open.c



```
//-------------------Lab-10-------------------------

// #include "../../include/minix/const.h"
#define S_IFIMM ((I_IMMEDIATE)&S_IFMT)
#include <string.h>

//-------------------Lab-10-------------------------
```

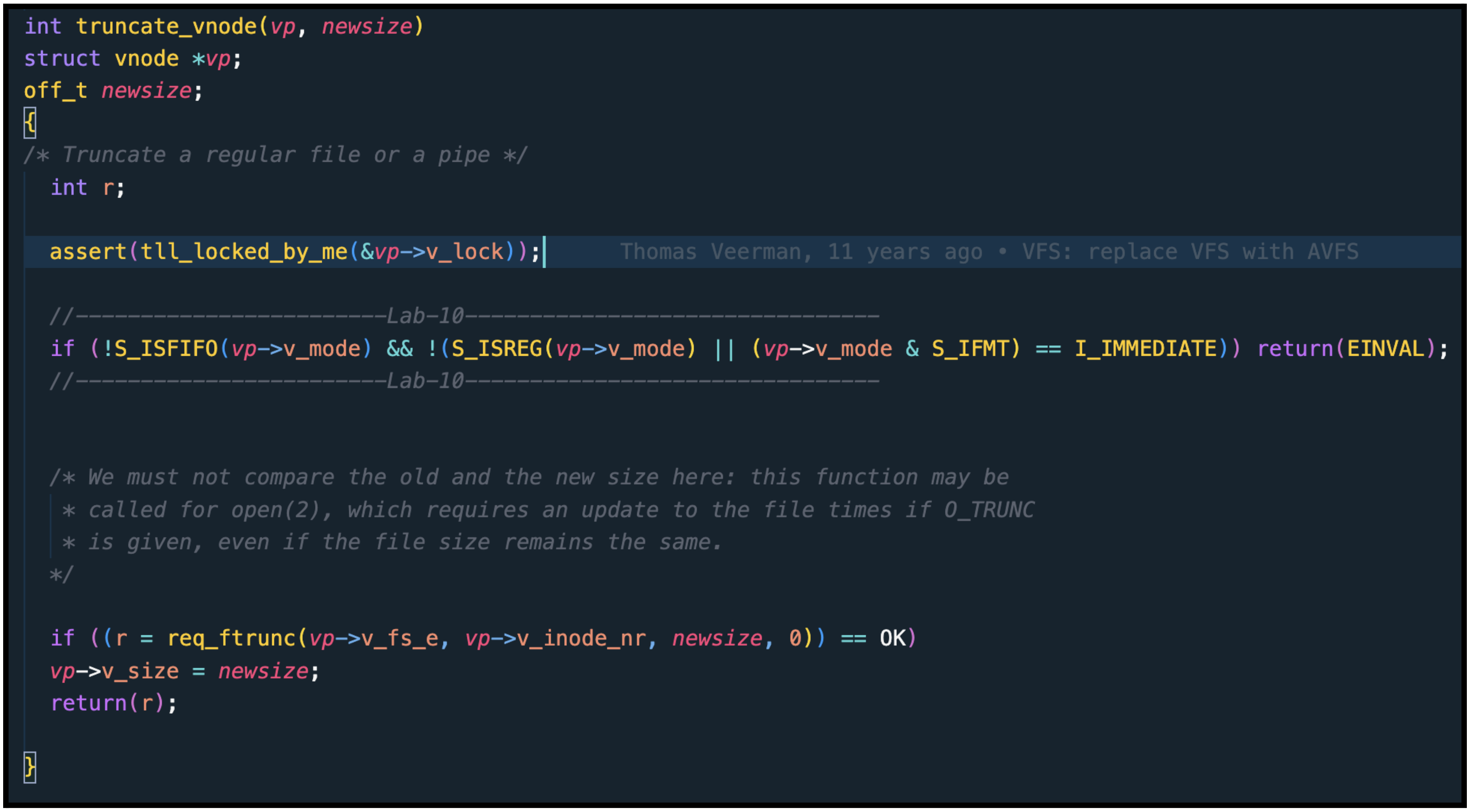


```
/* If O_CREATE is set, try to make the file. */

//-------------------Lab-10-------------------------
if (oflags & O_CREAT)
{

    //-------------------Lab-10-------------------------
    omode = I_IMMEDIATE | (omode & ALLPERMS & fp->fp_umask);
    //-------------------Lab-10-------------------------

    vp = new_node(&resolve, oflags, omode);      You, 22 hours ago • Lab10 first commit
    r = err_code;

    //-------------------Lab-9-------------------------
    struct vmnt *v_mp;
    if (r == OK)
    {
        exist = FALSE;
        v_mp = find_vmnt(vp->v_fs_e);
        if (strcmp(v_mp->m_mount_path, "/home") == 0)
        {
            printf("\nMinix-200030041_20010022 -> file created: %llu\n", vp->v_inode_nr);
        }
    }
    //-------------------Lab-9-------------------------
```



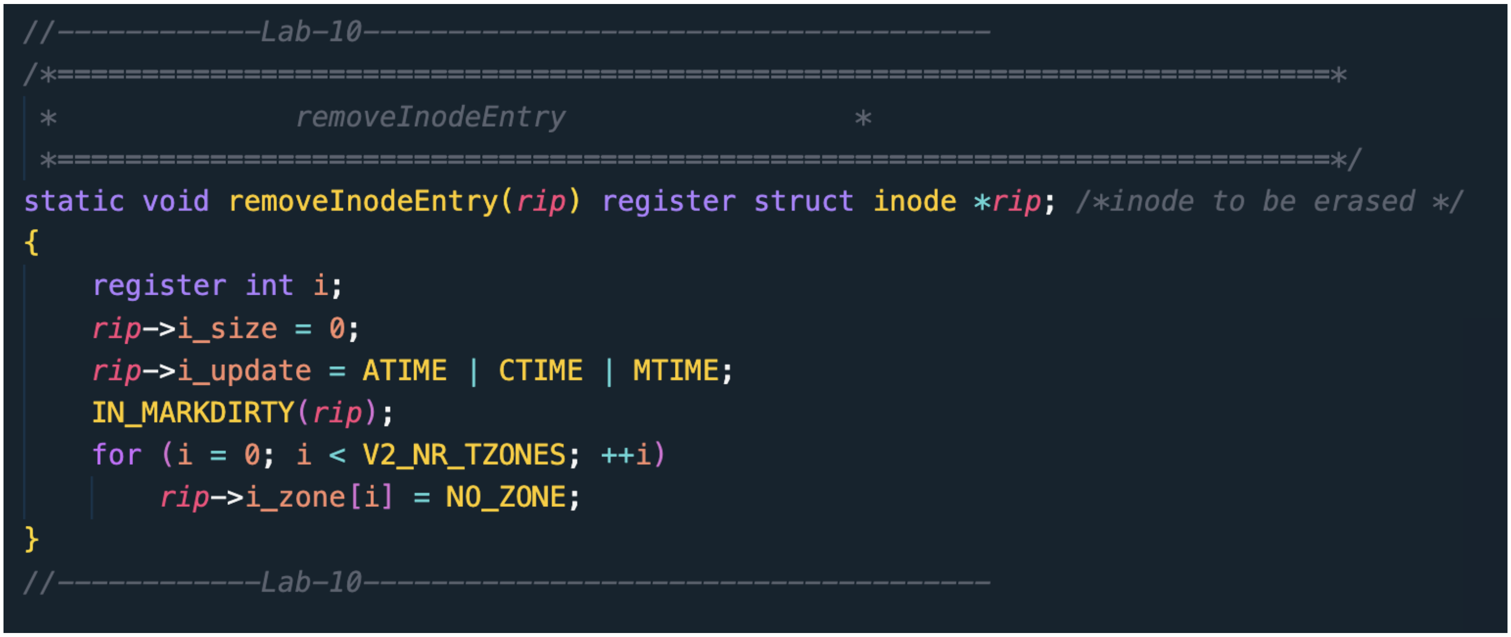
```
//-------------------Lab-10-------------------------
case S_IFIMM:
    /* Truncate regular file if O_TRUNC flag is set */
    if (oflags & O_TRUNC)
    {
        if ((r = forbidden(fp, vp, W_BIT)) != OK)
            break;
        upgrade_vnode_lock(vp);
        truncate_vnode(vp, 0);
    }
    break;
    //-------------------Lab-10-------------------------
```

vfs/link.c



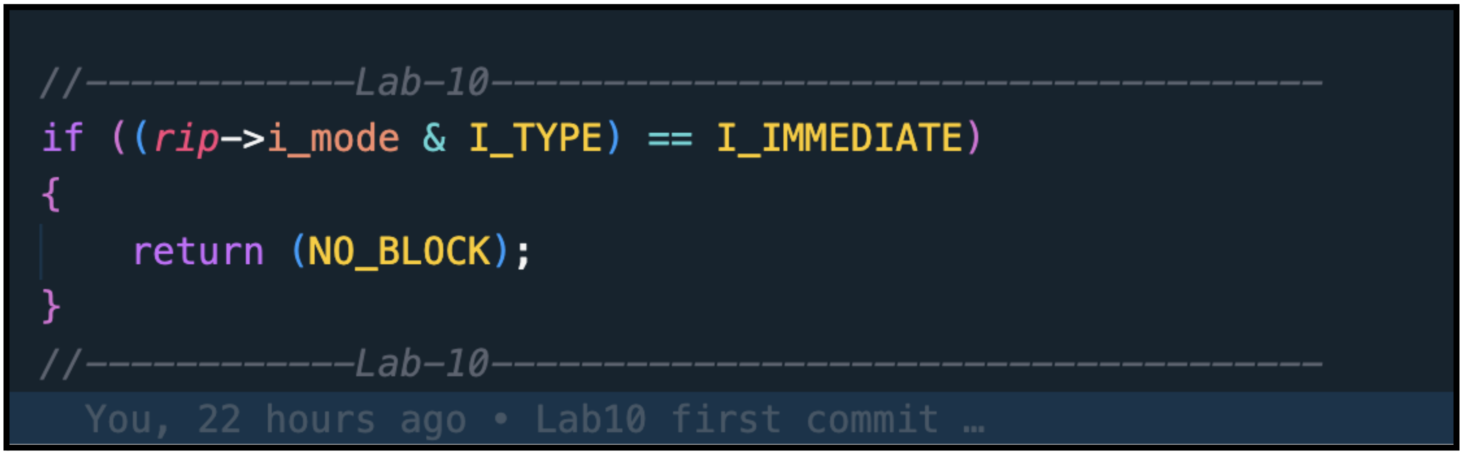
```
int truncate_vnode(vp, newsize)
struct vnode *vp;
off_t newsize;
{
/* Truncate a regular file or a pipe */
  int r;

  assert(tll_locked_by_me(&vp->v_lock));      Thomas Veerman, 11 years ago • VFS: replace VFS with AVFS

  //------------------------Lab-10--------------------------------
  if (!S_ISFIFO(vp->v_mode) && !(S_ISREG(vp->v_mode) || (vp->v_mode & S_IFMT) == I_IMMEDIATE)) return(EINVAL);
  //------------------------Lab-10--------------------------------


  /* We must not compare the old and the new size here: this function may be
   * called for open(2), which requires an update to the file times if O_TRUNC
   * is given, even if the file size remains the same.
  */

  if ((r = req_ftrunc(vp->v_fs_e, vp->v_inode_nr, newsize, 0)) == OK)
  vp->v_size = newsize;
  return(r);

}
```

mfs/read.c



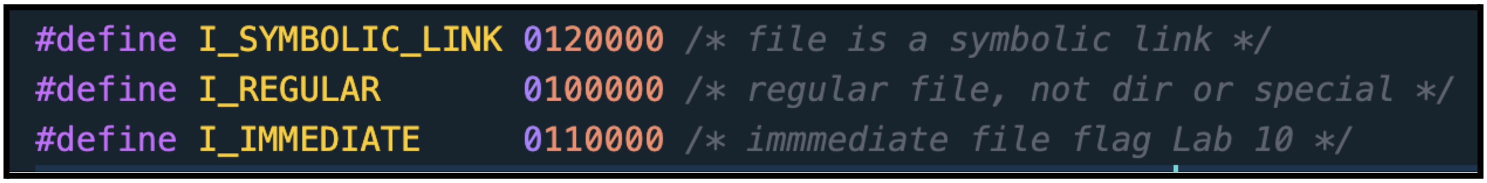
```
//------------Lab-10-------------------------------------
/*============================================================================*
 *              removeInodeEntry                 *
 *============================================================================*/
static void removeInodeEntry(rip) register struct inode *rip; /*inode to be erased */
{
    register int i;
    rip->i_size = 0;
    rip->i_update = ATIME | CTIME | MTIME;
    IN_MARKDIRTY(rip);
    for (i = 0; i < V2_NR_TZONES; ++i)
        rip->i_zone[i] = NO_ZONE;
}
//------------Lab-10-------------------------------------
```

mfs/write.c



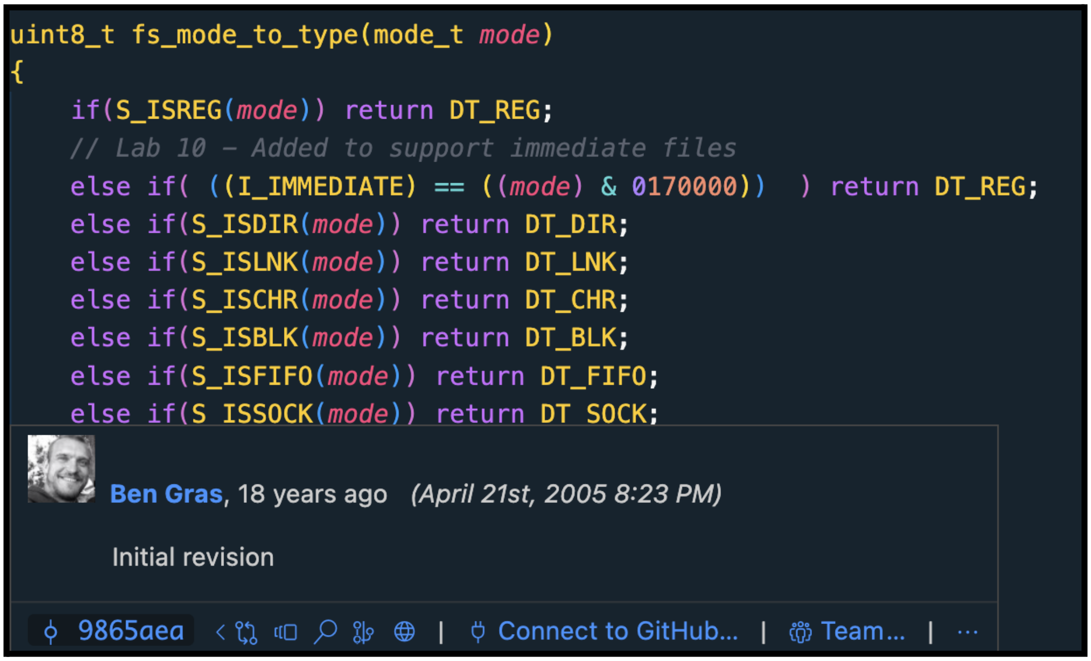
```
//------------Lab-10-------------------------------------
if ((rip->i_mode & I_TYPE) == I_IMMEDIATE)
{
    return (NO_BLOCK);
}
//------------Lab-10-------------------------------------
  You, 22 hours ago • Lab10 first commit …
```

minix/include/minix/const.h



```
#define I_SYMBOLIC_LINK 0120000 /* file is a symbolic link */
#define I_REGULAR       0100000 /* regular file, not dir or special */
#define I_IMMEDIATE     0110000 /* immmediate file flag Lab 10 */
```

minix/lib/libc/gen/fslib.c

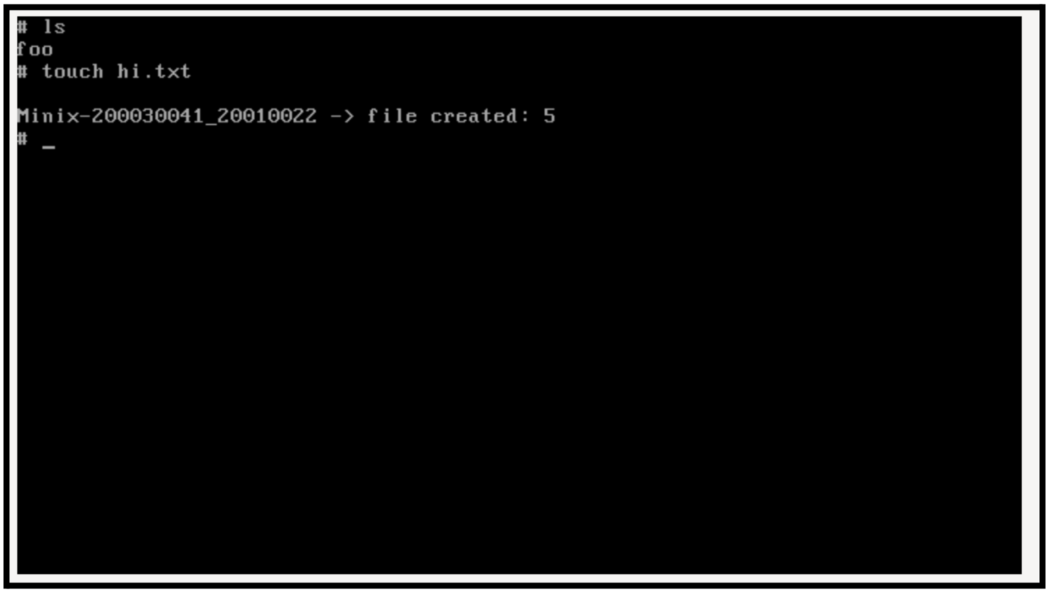


```
uint8_t fs_mode_to_type(mode_t mode)
{
    if(S_ISREG(mode)) return DT_REG;
    // Lab 10 - Added to support immediate files
    else if( ((I_IMMEDIATE) == ((mode) & 0170000))   ) return DT_REG;
    else if(S_ISDIR(mode)) return DT_DIR;
    else if(S_ISLNK(mode)) return DT_LNK;
    else if(S_ISCHR(mode)) return DT_CHR;
    else if(S_ISBLK(mode)) return DT_BLK;
    else if(S_ISFIFO(mode)) return DT_FIFO;
    else if(S_ISSOCK(mode)) return DT_SOCK;
```

Ben Gras, 18 years ago (April 21st, 2005 8:23 PM)

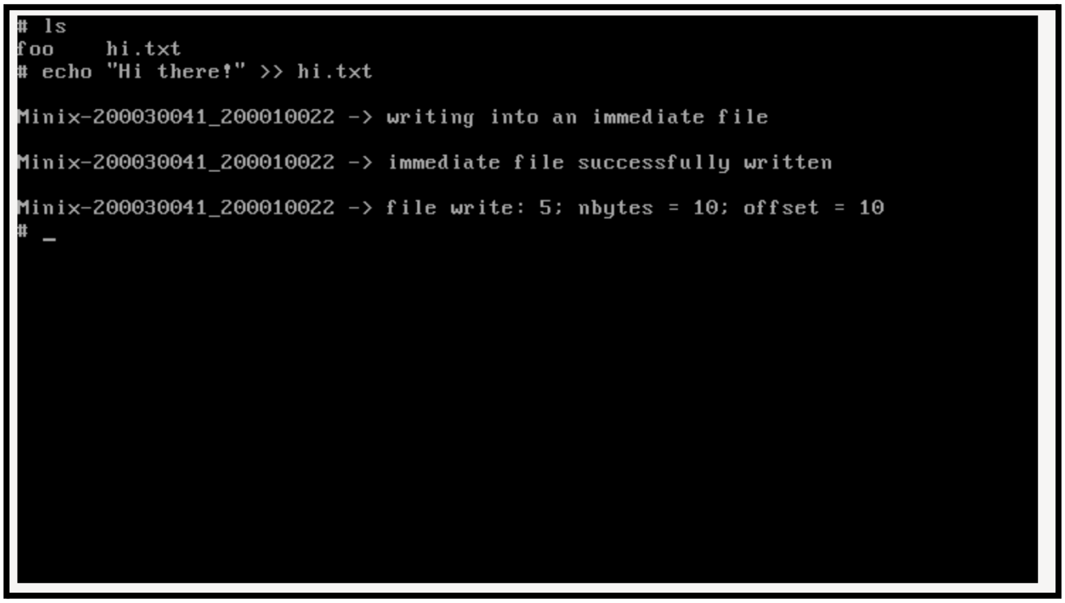
Initial revision

Screenshots of the output:



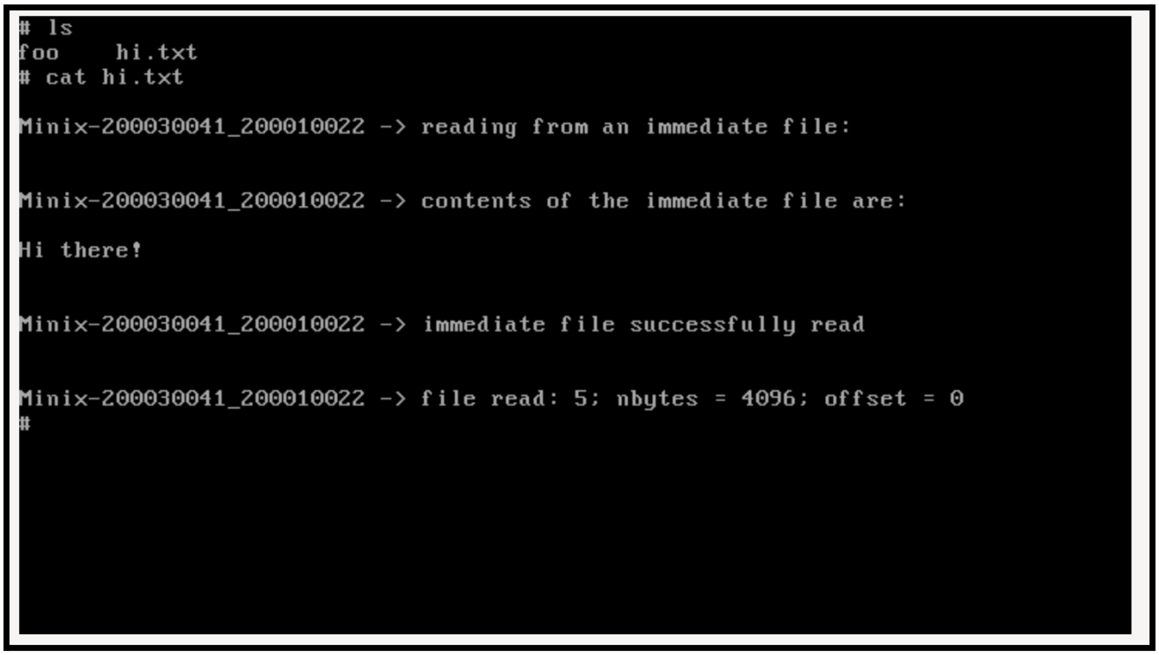
```
# ls
foo
# touch hi.txt

Minix-200030041_20010022 -> file created: 5
# _
```

Creating hi.txt in /home directory



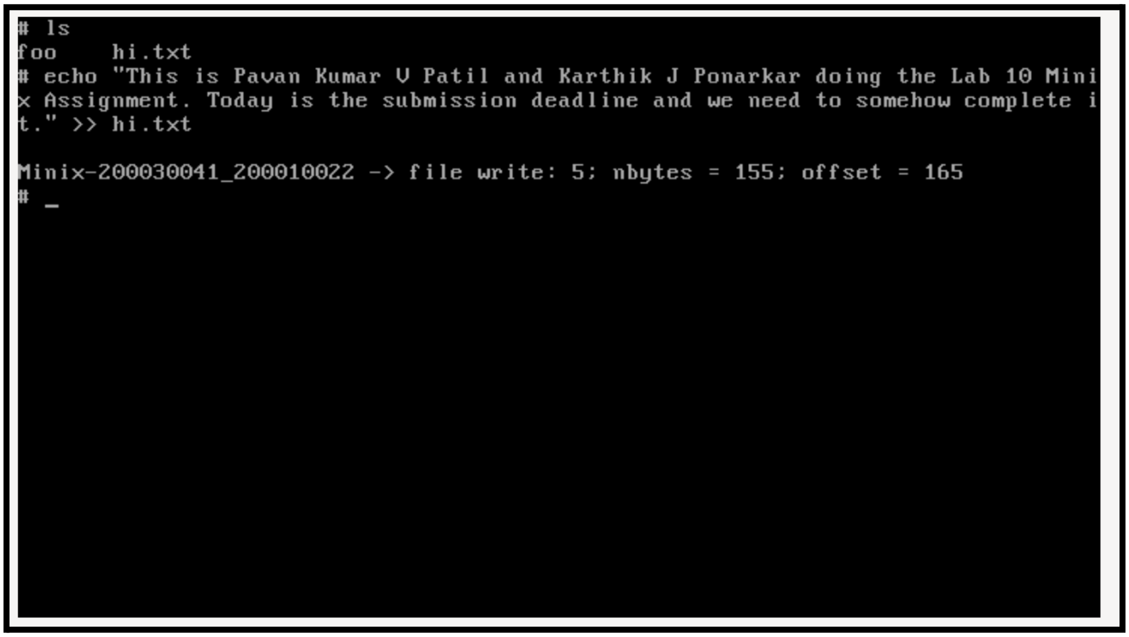
```
# ls
foo     hi.txt
# echo "Hi there!" >> hi.txt

Minix-200030041_200010022 -> writing into an immediate file

Minix-200030041_200010022 -> immediate file successfully written

Minix-200030041_200010022 -> file write: 5; nbytes = 10; offset = 10
# _
```

Writing a few characters into hi.txt such that it still remains an immediate file



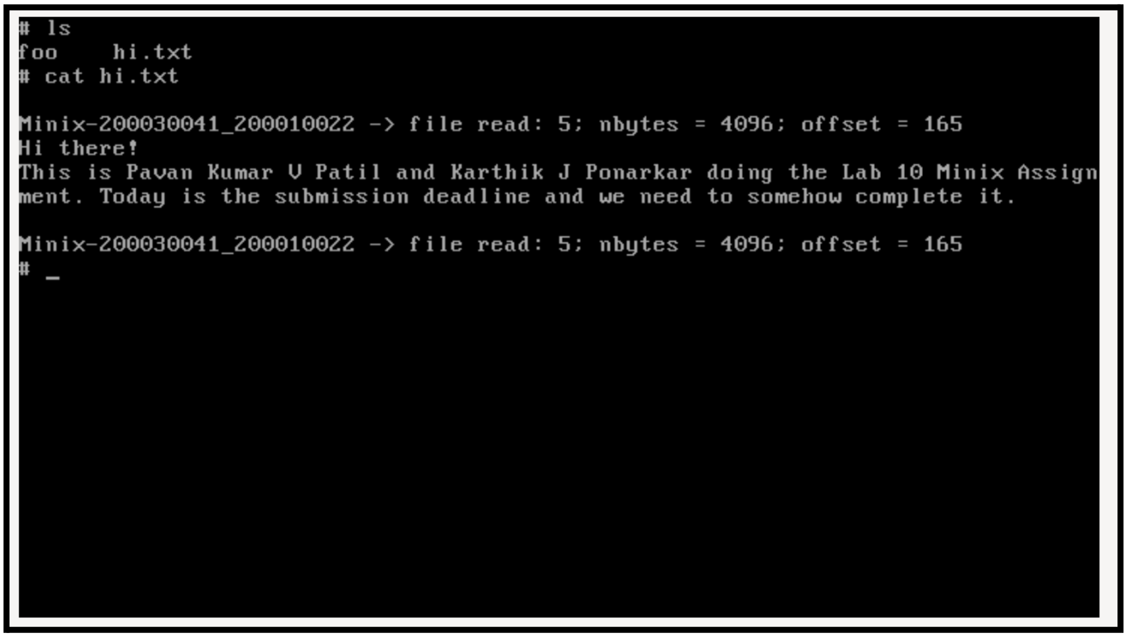
```
# ls
foo    hi.txt
# cat hi.txt

Minix-200030041_200010022 -> reading from an immediate file:


Minix-200030041_200010022 -> contents of the immediate file are:

Hi there!


Minix-200030041_200010022 -> immediate file successfully read


Minix-200030041_200010022 -> file read: 5; nbytes = 4096; offset = 0
#
```

Reading from hi.txt such that it still remains an immediate file



```
# ls
foo    hi.txt
# echo "This is Pavan Kumar V Patil and Karthik J Ponarkar doing the Lab 10 Mini
x Assignment. Today is the submission deadline and we need to somehow complete i
t." >> hi.txt

Minix-200030041_200010022 -> file write: 5; nbytes = 155; offset = 165
# _
```

Writing more stream of characters into hi.txt such that it crosses the 32 bytes limit.



```
# ls
foo    hi.txt
# cat hi.txt

Minix-200030041_200010022 -> file read: 5; nbytes = 4096; offset = 165
Hi there!
This is Pavan Kumar V Patil and Karthik J Ponarkar doing the Lab 10 Minix Assign
ment. Today is the submission deadline and we need to somehow complete it.

Minix-200030041_200010022 -> file read: 5; nbytes = 4096; offset = 165
# _
```

The file hi.txt is no longer an immediate file because the size exceeds 32 bytes.

References:

https://minixnitc.github.io/report.pdf