CS 314 – Operating Systems Lab

180010008

Lab-6

Outline: - This report contains the modifications in minix code for the working of mfs file system which is mounted at /home only.

The following changes(prints) in respective files and their corresponding outputs are given below: -

1) File Delete: -

➢ Location: - Changes are made in file "/usr/src/minix/servers/vfs/link.c"

➢ Function: - do_unlink()

➢ Code Changes: -

Here we need to check if deleted file is in /home, only then printf is called.

```
//Changing Here for Delete Case
lookup_init(&stickycheck, resolve.l_path, PATH_RET_SYMLINK, &vmp2, &vp);
stickycheck.l_vmnt_lock = VMNT_READ;
stickycheck.l_vnode_lock = VNODE_READ;
vp = advance(dirp, &stickycheck, fp);

if (strcmp(vmp->m_mount_path, "/home") == 0){
  printf("file deleted: %llu\n", vp->v_inode_nr);
}
if (vp != NULL){
  unlock_vnode(vp);
  put_vnode(vp);
}
```

➢ Output: -

Here we can see that current location is in '/home'.

And file deleted printed successfully.

```
# cd /home
# ls
Minix: PID 357 created
Quantum Alloted: 200, Quantum Used: 200
PID 107 swapped in
minix        newfile.out
Minix: PID 357 exited
# rm newfile.out
Minix: PID 358 created
Quantum Alloted: 200, Quantum Used: 200
PID 108 swapped in
file deleted: 5
Minix: PID 358 exited
# _
```

2) File Creation: -

➢ Location: - Changes are made in file "/usr/src/minix/servers/vfs/open.c"

➢ Function: - common_open()

➢ Code Changes: -

Here we need to check if newly created file is in /home, only then printf is called.

```
//Change here for Create
struct vmnt *vmpPath;
vmpPath = find_vmnt(vp->v_fs_e);
if (strcmp(vmpPath->m_mount_path, "/home") == 0)
{
    printf("file created: %llu\n", vp->v_inode_nr);
}
```

➢ Output: -

File Created printed successfully.

```
# touch newfile.out
Minix: PID 359 created
Quantum Alloted: 200, Quantum Used: 200
PID 109 swapped in
file created: 5
Minix: PID 359 exited
# _
```

3) File Reading: -

➢ Location: - Changes are made in file "/usr/src/minix/servers/vfs/read.c"

➢ Function: - read_write ()

➢ Code Changes: -

Here the rw_flag differentiates between reading and writing. Its value should be equal to be READING while reading a particular file

```
//Change here for Read and Write
struct vmnt *vmp;
vmp = find_vmnt(vp->v_fs_e);
if (rw_flag == WRITING && strcmp(vmp->m_mount_path, "/home") == 0)
{
  printf("file written: %llu; nbytes = %zu; offset = %llu\n", vp->v_inode_nr, size, f->filp_pos);
}
if (rw_flag == READING && strcmp(vmp->m_mount_path, "/home") == 0)
{
  printf("file read: %llu; nbytes = %zu; offset = %llu\n", vp->v_inode_nr, size, f->filp_pos);
}
```

➢ Output: -

File read printed successfully.

```
# cat newfile.out
Minix: PID 362 created
Quantum Alloted: 200, Quantum Used: 200
PID 112 swapped in
file read: 5; nbytes = 4096; offset = 28
New line added for testing.
file read: 5; nbytes = 4096; offset = 28
Minix: PID 362 exited
#
```

4) File Writing: -

➢ Location: - Changes are made in file "/usr/src/minix/servers/vfs/read.c"

➢ Function: - read_write ()

➢ Code Changes: -

Here the rw_flag differentiates between reading and writing. Its value should be equal to be WRITING while writing a particular file

```
//Change here for Read and Write
struct vmnt *vmp;
vmp = find_vmnt(vp->v_fs_e);
if (rw_flag == WRITING && strcmp(vmp->m_mount_path, "/home") == 0)
{
  printf("file written: %llu; nbytes = %zu; offset = %llu\n", vp->v_inode_nr, size, f->filp_pos);
}
if (rw_flag == READING && strcmp(vmp->m_mount_path, "/home") == 0)
{
  printf("file read: %llu; nbytes = %zu; offset = %llu\n", vp->v_inode_nr, size, f->filp_pos);
}
```

➢ Output: -

File Written printed successfully.

```
# echo "Hello World" > newfile.out
file written: 5; nbytes = 12; offset = 12
# _
```