CS 314 – Operating Systems Lab
Lab–10 Report

Student 1: Sahaja Nandyala **Roll-No :** 200010032

1 Immediate files

In this lab, we are required to Implement immediate files in the Minix File System, for files of size up to 32 bytes.

we are adding following line in cost.h which will be used further for implementing immediate files.

```
#define I_IMMEDIATE     0110000 /* immmediate file */
```

In /minix/lib/libc/gen/fslib.c Returning regular file mode if is not one of the above file modes as immediate file will be not recognised and will abort

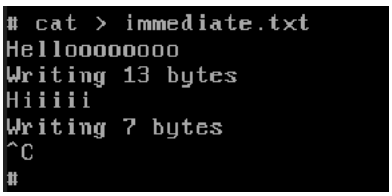
```
//-----------------------------------------------------------------
uint8_t fs_mode_to_type(mode_t mode)
{
    if(S_ISREG(mode)) return DT_REG;
    else if(S_ISDIR(mode)) return DT_DIR;
    else if(S_ISLNK(mode)) return DT_LNK;
    else if(S_ISCHR(mode)) return DT_CHR;
    else if(S_ISBLK(mode)) return DT_BLK;
    else if(S_ISFIFO(mode)) return DT_FIFO;
    else if(S_ISSOCK(mode)) return DT_SOCK;
    // returning regular file mode if is not one of the above
    return DT_REG; // added this
    // assert(0 && "unknown type"); // commented this

    /* assert()s are removed on NDEBUG builds. */
    abort();
}
//-----------------------------------------------------------------
```

2 File creation

we start the file by creating as an immediate file. so, in vfs/open.c we change the mode from regular to immediate in common_open function while creating a file by doing following changes.

```
// commented the regular file type
// omode = I_REGULAR | (omode & ALLPERMS & fp->fp_umask);
//-----------------------------------------------------------------
// file type is set to immediate
omode = I_IMMEDIATE | (omode & ALLPERMS & fp->fp_umask);
//-----------------------------------------------------------------
```

Srceen shot of file creation and writing in minix after buliding



3 File read & File write

For file read of an immediate file, we can respond with the inode structure contents. If not, we can follow the default behavior of looking up zones.

For file write it is similar to read but we must take care to ensure that if we want to write to the inode structure, then the new file size is still within 32B. When a regular file shrinks less than 32 bytes, there is no need to come back to immediate mode. If file size becomes more than 32B then file type is changed to regular

For reading and writing the following changes are made.

In mfs/write.c file in write_map function we return no block as immediate files doesn't require a block

```
//-----------------------------------------------------------------
if((rip->i_mode & I_TYPE) == I_IMMEDIATE) // added for immediate type
{
    return(NO_BLOCK);
}
//-----------------------------------------------------------------
```

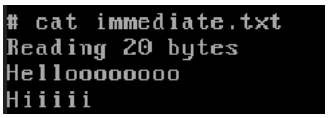
In mfs/read.c if the file is in immediate mode and is to be written, if the file size is ¿ 32 bytes, mode is changed to regular and the contents of the inode are copied to a new block which is marked dirty. This makes the immediate to regular transition. If the file is in immediate mode and wants to read or write, its size being ¡= 32 bytes, it fetches contents from and writes to inode itself.

```
//-----------------------------------------------------------------
if(mode_word==I_IMMEDIATE) // for immediate files
{
 if(rw_flag==WRITING) // if flag is set to WRITING
```

```
 {
// checking if file size becomes > 32 after the content
// added if yes then the filetype is chaged to regular
   if(position+nrbytes>32)
   {
         char* temp;
         char inode_data[40];
         register struct buf* bp;
         // coping the data in inode into inode_data
         for(int i=0; i<f_size; i++)
         {
           if(i%4 == 0)
           {
               temp = (char*)rip->i_zone + i;
           }
           inode_data[i] = temp[i%4];
     }
     wipe_inode(rip); // clearing data from immediate type inode
        // setting mode type to regular
     rip->i_mode = (I_REGULAR | (rip->i_mode & ALL_MODES));
     mode_word = rip->i_mode & I_TYPE;
     if ((bp = new_block(rip, (off_t) ex64lo(0))) == NULL) // creating a new block
       return(err_code);

      //writing data in inode_data to new block
     for(int i=0; i<f_size; i++)
     {
       ((char*)bp->data)[i] = inode_data[i];
     }
     MARKDIRTY(bp); // setting dirty bit
     put_block(bp, PARTIAL_DATA_BLOCK);
   }
   else // if < 32 the file is in immediate
   {
        immediate=1;
   }
 }
 else // READ Immediate
 {
   if(position>=f_size) // if seek position is > file size then set not immediate
   { immediate=0;  }
   else
   {  immediate=1; }
 }
```

```
}
if(immediate ==1) // if file is immediate type
{
 if(rw_flag == READING) // for reading
 {
   printf("Read immediate %lld bytes\n",f_size);
   r = sys_safecopyto(VFS_PROC_NR, gid, (vir_bytes)cum_io,
                           (vir_bytes) rip->i_zone,(size_t) f_size);

    for(i=0; i<f_size; ++i)
    {
       if(i%4 == 0)
         temp_bytes = (char*)rip->i_zone + i;
       buffer[i] = temp_bytes[i%4];
       printf("%c", buffer[i]);
    }

   if(r==OK)
   {
     nrbytes = 0;
     cum_io += f_size;
     position += f_size;
   }
 }
 else // for writing
 {
   printf("Write immediate %d bytes\n",nrbytes);
   vir_bytes zone;
   zone = (vir_bytes) rip->i_zone;
   r = sys_safecopyfrom(VFS_PROC_NR, gid, (vir_bytes)cum_io,
                                zone+position, (size_t) nrbytes);
   IN_MARKDIRTY(rip);
   if(r==OK)
   {
     cum_io += nrbytes;
     position += (off_t)nrbytes;
     nrbytes = 0;
   }
 }
}
// ---------------------------------------------------------------------
```

Screen shot for reading from an immediate file



4 File deletion

In vfs/link.c file the truncate_vnode function has been made the following change to add the immediate type file. Deletion will happen as regular files.

```
//-----------------------------------------------------------------------
// including immediate mode files
if (!(S_ISREG(vp->v_mode) || (vp->v_mode & S_IFMT) == 0110000)
            && !S_ISFIFO(vp->v_mode)) return(EINVAL);
//----------------------------------------------------------------------
```

Screen shot for deleting from an immediate file