Rmdir.C

int rmdir(char *pathname)

First we get the inumber of the pathname and load its contents into a MINODE (mip). I use S_ISDIR to check if pathname is dir and check mip->refcount must equal 1. The pathname must have less o requal to 2 links count. If Count = 2, check by stepping through all entries that . and .. are only entry. (I use a Canter which increments for each entry). If any checks fail, print an error and put back mip while returning error.
If all checks pass, I deallocate the dir we're removing's minode (mip's) iblocks and also deallocate the inode number before calling iput.
I then use dirname function to get Parent's ino number loaded to a MINODE (pip). I call findmyname to find the entry to remove in the parent minode and call rm_child (pip, temp) to remove that entry from the parent's dir.
Finally, I dec parents link count, touch time, mark as dirty and put inode back on disk.

int rm_child (MINODE *parent, char *name)

First, I get Parent iblock[0] into a buffer and cycle through all entries in the block searching for the name of the entry to delete. I calculate the ideal_length of the entry to remove to help later determine if entry to remove is last entry or in the middle. If entry to remove rec_len is greater than ideal_length then entry to remove is at end.
I keep a Previous dir pointer when stepping through entries, so all I do is add deleted rec_len to Previous entry rec_len and put the block back to the disk. ELSE entry is in the middle of the block. I calculate size of deleted entry and mark the start and end locations of removed entry. Then I step through to the end of the block and add deleted entry length to find entry rec_len. Finally, I memory copy from the start to the end total size - start point and write the block back to the disk