cd_ls_pwd.c

int chdir (char *pathname)

First, we check if user supplied a 2nd argument.
If No 2nd argument, set running->cwd to root.
If user supplied 2nd argument, we load ino number of pathname using getino to check pathname exists. Then load ino into a MINODE (mip) using iget. After using S_ISDIR to check mip->INODE.i_mode is a directory we call iput on the running->cwd to put back the current running directory and call running-> cwd = mip to change the running directory to user specified pathway.

Char * pwd (MINODE *wd)

First, we check if wd is the root or not. If root, Print*return (wd) = /. If not root, call rpwd(wd) to update the global ret PWD variable and return retPWD.

Char * retPWD[NMINODE]; // returns cwd if not root
Char * rpwd (MINODE *wd)

First we loop through the Dir's data block to find wd ino number and Parent ino number. Next we load a MINODE (pip) with Parent's information and call findmyname (pip, my_ino, myname) to loop through parents data block and finds entry w/ same ino number, saving that entry name as a string to my_name. Finally, we recursively call rpwd (pip) on the Parent minode to work our way up to the root while using strcat to save the pathway, thus finally printing out the whole working directory.

int ls (char *pathname)

First, we check whether or not user provided an argument. If argument isn't provided, we call ls_dir(running->cwd) on our current running directory. If argument IS provided, we save the current directory with strcpy (curdir, pwd(running->cwd)). Next we call chdir (pathname) to change to user specified directory, then call ls_dir (running->cwd) on new running directory and finally call chdir (curdir) to change directory back to what it was originally.

int ls_dir (MINODE *mip)

First, we get running->cwd block into a buf by calling get_block (dev, mip->INODE.i_block[0], buf);
Then we step through all entries in the block while calling ls_file (mip, temp) on each entry to print out all data for each entry in the block.

char *t1 = "xwrxwrxwr----"
char *t2 = "----------"
int ls_file (MINODE *mip, char *name)

First, we check the mode of the mip to print out the correct first character // - (reg) d (dir) l (link).
Next, print out the next 8 chars according to which permission bits are available for that mode. We display all information found in linux ls function formatted the same as ls -l call. Finally, we check if the MINODE is a LNK type and if so, print out the linking arrow (->) followed by the link name.