Makiah Heinzmann, Taiya Williams

11442136 11644614

On top of your hard copy: MARK yes or no

1. does your ls work? ls; ls /dir1; ls /dir1/dir3\_\_\_\_\_\_\_\_\_\_\_\_\_YES\_\_\_\_\_\_\_

2. does your cd work? cd /dir1; cd /dir1/dir3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_YES\_\_\_\_\_\_\_

3. does your pwd work? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_YES\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\* cd\_ls\_pwd.c file \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int chdir(char \*pathname) {

printf("chdir %s\n", pathname);

// printf("under construction READ textbook HOW TO chdir!!!!\n");

// READ Chapter 11.7.3 HOW TO chdir

int inode = getino(pathname);

MINODE \* min = iget(dev, inode);

if (S\_ISDIR(min->INODE.i\_mode)) {

iput(running->cwd);

running->cwd = min;

} else {

printf("Failure: [ %s ] Not a directory!\n", pathname);

}

}

int ls\_file(MINODE \*mip, char \*name)

{

// printf("ls\_file: to be done: READ textbook for HOW TO!!!!\n");

// READ Chapter 11.7.3 HOW TO ls

char type, perm[10] = "wrxwrxwrx";

\_\_u16 mode = mip->INODE.i\_mode;

if (S\_ISDIR(mode)) type = 'd'; else type = '-';

for (int i = 0; i < 9; i++) if (!(mode & (1 << i))) perm[i] = '-';

\_\_u16 links = mip->INODE.i\_links\_count;

\_\_u16 owner = mip->INODE.i\_uid;

\_\_u16 group = mip->INODE.i\_gid;

time\_t date = mip->INODE.i\_mtime;

\_\_u32 size = mip->INODE.i\_size;

printf("%c%s% 4d% 4d% 4d %.20s % 8d %s\n",

type, perm, links, owner, group, ctime(&date)+4, size, name);

}

int ls\_dir(MINODE \*mip)

{

// printf("ls\_dir: list CWD's file names; YOU do it for ls -l\n");

char buf[BLKSIZE], temp[256];

DIR \*dp;

char \*cp;

// Assume DIR has only one data block i\_block[0]

get\_block(dev, mip->INODE.i\_block[0], buf);

dp = (DIR \*)buf;

cp = buf;

while (cp < buf + BLKSIZE){

strncpy(temp, dp->name, dp->name\_len);

temp[dp->name\_len] = 0;

// printf("[%d %s] ", dp->inode, temp); // print [inode# name]

ls\_file(iget(dev, dp->inode), temp);

cp += dp->rec\_len;

dp = (DIR \*)cp;

}

printf("\n");

}

int ls(char \*pathname)

{

printf("ls %s\n", pathname);

//printf("ls CWD only! YOU do it for ANY pathname\n");

if (pathname[0] != '\0') {

MINODE \* min = iget(dev, getino(pathname));

if (S\_ISDIR(min->INODE.i\_mode)) {

ls\_dir(min);

iput(min);

} else

printf("Failure: [ %s ] Not a directory!\n", pathname);

} else

ls\_dir(running->cwd);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Algorithm of pwd \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* rpwd( MINODE \*wd){

\* (1). if (wd == root) return;

\* (2). from wd->INODE.i\_block[0], get my\_ino and parent\_ino

\* (3). pip = iget(dev, parent\_ino);

\* (4). from pip->INODE.i\_block[]: get my\_name string by my\_ino as LOCAL

\* (5). rpwd(pip);

\* // recursive call rpwd( pip) with parent minode

\*/

void recursivePWD(MINODE \*curNode) {

if (curNode != root) {

int myINode = 0;

int parentINode = findino(curNode, &myINode);

MINODE \* parent = iget(dev, parentINode);

char curName[255];

findmyname(parent, myINode, curName);

recursivePWD(parent);

iput(parent);

for (int i = 0; curName[i]; i++)

if (curName[i] == '\r') // that took way too long to find...

curName[i] = '\0';

printf("/%s", curName);

}

}

void pwd(MINODE \*wd){

printf("CWD = ");

if (wd == root) printf("/");

recursivePWD(wd);

printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WE WROTE THIS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IN util.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int findmyname(MINODE \*parent, u32 myino, char \*myname) {

char buffer[BLKSIZE], \* current = buffer;

DIR \* dirPtr = (DIR \*) current;

get\_block(parent->dev, parent->INODE.i\_block[0], buffer);

while(myino != dirPtr->inode) {

current += dirPtr->rec\_len;

dirPtr = (DIR \*) current;

}

strcpy(myname, dirPtr->name);

//printf("\n%s\n", myname); //TODO-rm

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

OUTPUT:

checking EXT2 FS ....EXT2 FS OK

bmp=8 imap=9 inode\_start = 10

init()

mount\_root()

root refCount = 1

creating P0 as running process

root refCount = 2

input command : [ls|cd|pwd|quit] ls

cmd=ls pathname=

ls

dw-xw-xwrx 5 0 0 Mar 11 20:54:21 2020 1024 .

dw-xw-xwrx 5 0 0 Mar 11 20:54:21 2020 1024 ..

d------wrx 2 0 0 Mar 11 20:54:20 2020 12288 lost+found

dw-xw-xwrx 3 0 0 Mar 11 20:54:20 2020 1024 dir1

dw-xw-xwrx 2 0 0 Mar 11 20:54:20 2020 1024 dir2

---x--x-rx 1 0 0 Mar 11 20:54:21 2020 0 file1

---x--x-rx 1 0 0 Mar 11 20:54:21 2020 0 file2

input command : [ls|cd|pwd|quit] ls dir1

cmd=ls pathname=dir1

ls dir1

getino: pathname=dir1

tokenize dir1

dir1

===========================================

getino: i=0 name[0]=dir1

search for dir1 in MINODE = [3, 2]

ino rlen nlen name

2 12 1 .

2 12 2 ..

11 20 10 lost+found

12 12 4 dir1

found dir1 : ino = 12

dw-xw-xwrx 3 0 0 Mar 11 20:54:20 2020 1024 .

dw-xw-xwrx 5 0 0 Mar 11 20:54:21 2020 1024 ..

dw-xw-xwrx 2 0 0 Mar 11 20:54:20 2020 1024 dir3

input command : [ls|cd|pwd|quit] ls dir1/dir3

cmd=ls pathname=dir1/dir3

ls dir1/dir3

getino: pathname=dir1/dir3

tokenize dir1/dir3

dir1 dir3

===========================================

getino: i=0 name[0]=dir1

search for dir1 in MINODE = [3, 2]

ino rlen nlen name

2 12 1 .

2 12 2 ..

11 20 10 lost+found

12 12 4 dir1

found dir1 : ino = 12

===========================================

getino: i=1 name[1]=dir3

search for dir3 in MINODE = [3, 12]

ino rlen nlen name

12 12 1 .

2 12 2 ..

14 1000 4 dir3

found dir3 : ino = 14

dw-xw-xwrx 2 0 0 Mar 11 20:54:20 2020 1024 .

dw-xw-xwrx 3 0 0 Mar 11 20:54:20 2020 1024 ..

input command : [ls|cd|pwd|quit] cd dir1

cmd=cd pathname=dir1

chdir dir1

getino: pathname=dir1

tokenize dir1

dir1

===========================================

getino: i=0 name[0]=dir1

search for dir1 in MINODE = [3, 2]

ino rlen nlen name

2 12 1 .

2 12 2 ..

11 20 10 lost+found

12 12 4 dir1

found dir1 : ino = 12

input command : [ls|cd|pwd|quit] ls

cmd=ls pathname=

ls

dw-xw-xwrx 3 0 0 Mar 11 20:54:20 2020 1024 .

dw-xw-xwrx 5 0 0 Mar 11 20:54:21 2020 1024 ..

dw-xw-xwrx 2 0 0 Mar 11 20:54:20 2020 1024 dir3

input command : [ls|cd|pwd|quit] pwd

cmd=pwd pathname=

CWD = /dir1

input command : [ls|cd|pwd|quit] cd ..

cmd=cd pathname=..

chdir ..

getino: pathname=..

tokenize ..

..

===========================================

getino: i=0 name[0]=..

search for .. in MINODE = [3, 12]

ino rlen nlen name

12 12 1 .

2 12 2 ..

found .. : ino = 2

input command : [ls|cd|pwd|quit] pwd

cmd=pwd pathname=

CWD = /

input command : [ls|cd|pwd|quit] ls dir1/dir3

cmd=ls pathname=dir1/dir3

ls dir1/dir3

getino: pathname=dir1/dir3

tokenize dir1/dir3

dir1 dir3

===========================================

getino: i=0 name[0]=dir1

search for dir1 in MINODE = [3, 2]

ino rlen nlen name

2 12 1 .

2 12 2 ..

11 20 10 lost+found

12 12 4 dir1

found dir1 : ino = 12

===========================================

getino: i=1 name[1]=dir3

search for dir3 in MINODE = [3, 12]

ino rlen nlen name

12 12 1 .

2 12 2 ..

14 1000 4 dir3

found dir3 : ino = 14

dw-xw-xwrx 2 0 0 Mar 11 20:54:20 2020 1024 .

dw-xw-xwrx 3 0 0 Mar 11 20:54:20 2020 1024 ..

input command : [ls|cd|pwd|quit] pwd

cmd=pwd pathname=

CWD = /

input command : [ls|cd|pwd|quit] cd dir1/dir3

cmd=cd pathname=dir1/dir3

chdir dir1/dir3

getino: pathname=dir1/dir3

tokenize dir1/dir3

dir1 dir3

===========================================

getino: i=0 name[0]=dir1

search for dir1 in MINODE = [3, 2]

ino rlen nlen name

2 12 1 .

2 12 2 ..

11 20 10 lost+found

12 12 4 dir1

found dir1 : ino = 12

===========================================

getino: i=1 name[1]=dir3

search for dir3 in MINODE = [3, 12]

ino rlen nlen name

12 12 1 .

2 12 2 ..

14 1000 4 dir3

found dir3 : ino = 14

input command : [ls|cd|pwd|quit] pwd

cmd=pwd pathname=

CWD = /dir1/dir3

input command : [ls|cd|pwd|quit] ls

cmd=ls pathname=

ls

dw-xw-xwrx 2 0 0 Mar 11 20:54:20 2020 1024 .

dw-xw-xwrx 3 0 0 Mar 11 20:54:20 2020 1024 ..

input command : [ls|cd|pwd|quit] quit

cmd=quit pathname=