**Problem Set 2 – Filesystem Scavenger Hunt**

1. **Source Code**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <errno.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <dirent.h>

#include <pwd.h>

#include <grp.h>

int traverse (const char\* path, const char\* root, const struct stat fstat, int readableByOther);

int compareFiles(const char \*file1,const char \*file2);

int main(int argc, char \*\*argv){

if(argc != 3){

printf("Insufficient parameters. Required program parameters: [target file] [start path]\n");

exit(1);

}

char\* targetFile = argv[1];

char\* rootPath = argv[2];

struct stat fstat;

char targetFileDir[PATH\_MAX];

sprintf(targetFileDir, "%s/%s", rootPath, targetFile);

if(stat(targetFileDir, &fstat) < 0){

fprintf(stderr, "DEBUG: Failed to open [%s] stat: %s\n", targetFile, strerror(errno));

exit(1);

}

printf("DEBUG: Target [%s] is %li bytes long, dev %li, ino %lu\n",targetFileDir, fstat.st\_size, fstat.st\_dev,fstat.st\_ino);

printf("DEBUG: Hard Link #: %lu, Block size: %li, # of 512B blocks allocated: %li\n\n", fstat.st\_nlink, fstat.st\_blksize, fstat.st\_blocks);

(fstat.st\_mode & (S\_IROTH | S\_IXOTH)) ? traverse(rootPath, targetFileDir, fstat, 1): traverse(rootPath, targetFileDir, fstat, 0);

return 0;

}

int traverse (const char\* path, const char \*root, const struct stat target\_fstat, int readableByOther){

DIR\* dir;

char\* child;

char newPath[PATH\_MAX];

struct dirent \*ls;

struct stat fstat;

ssize\_t r;

char \*linkname;

if(!(dir = opendir(path))){

//report if directory failed to open

fprintf(stderr,"DEBUG: Failed to open directory [%s]: %s\n", path, strerror(errno));

} else {

while((ls = readdir(dir))){

child = ls->d\_name;

//ignore . and .. file for efficiency

if(strcmp(child, ".")!=0 && strcmp(child, "..")!=0){

sprintf(newPath, "%s/%s", path, child);

if(strcmp(newPath, root)!=0){

switch(ls->d\_type){

case DT\_REG:

//if traverse fails to open stats

if(stat(newPath, &fstat) < 0) {

fprintf(stderr, "DEBUG: Failed to open [%s] stat for file: %s\n", newPath, strerror(errno));

continue;

}

if((ls->d\_ino == target\_fstat.st\_ino) &&(fstat.st\_dev == target\_fstat.st\_dev)){

printf("[%s]\tHARD LINK TO TARGET\t", newPath);

((fstat.st\_mode & S\_IROTH) && readableByOther) ? printf("OK READ BY OTHERS\n"):printf("NOT READABLE BY OTHERS\n");

} else {

//inspect file if they have same size

if(fstat.st\_size == target\_fstat.st\_size){

if(compareFiles(root, newPath) == 0) {

printf("[%s]\tDuplicate of Target (nlink = %lu)\t" , newPath, fstat.st\_nlink);

((fstat.st\_mode & S\_IROTH) && readableByOther)? printf("OK READ BY OTHERS\n"):printf("NOT READABLE BY OTHERS\n");

}

}

}

break;

case DT\_LNK:

if(stat(newPath, &fstat) < 0) {

fprintf(stderr, "DEBUG: [%s] links to something not a file: %s, skipping\n", newPath, strerror(errno));

continue;

}

linkname = malloc(fstat.st\_size + 1);

r = readlink(newPath, linkname, fstat.st\_size+1);

if(r < 0) {

fprintf(stderr, "Error opening symlink [%s]: %s\n", newPath, strerror(errno));

}

linkname[r] = '\0';

if((fstat.st\_ino == target\_fstat.st\_ino)&&(fstat.st\_dev == target\_fstat.st\_dev)) printf("[%s]\tSYMLINK RESOLVES TO TARGET\n", newPath);

else{

if(fstat.st\_size == target\_fstat.st\_size){

sprintf(newPath, "%s/%s", path, linkname);

if(compareFiles(newPath, root) == 0)//symlink resolves to duplicate

printf("[%s]\tSYMLINK [%s] RESOLVES TO DUPLICATE\n", newPath, linkname);

}

}

free(linkname);

break;

case DT\_DIR:

if(stat(newPath, &fstat) < 0) {

fprintf(stderr, "DEBUG: Failed to open [%s] stat for directory: %s\n", newPath, strerror(errno));

} else {

((fstat.st\_mode & S\_IXOTH) && readableByOther) ? traverse(newPath, root, target\_fstat, 1):traverse(newPath, root, target\_fstat, 0);

}

break;

default:

;

}

}

}

}

if(closedir(dir) == -1) fprintf(stderr,"DEBUG: Failed to close directory %s: %s\n", path, strerror(errno));

}

return 0;

};

//compare bitwise

//returns 0 on same file, 1 on different files

int compareFiles(const char \*file1, const char \*file2){

int fd1, fd2, one, two, result;

char buf1[BUFSIZ];

char buf2[BUFSIZ];

memset(buf1, 0, sizeof(buf1));

memset(buf2, 0, sizeof(buf2));

if((fd1 = open(file1, O\_RDONLY)) < 0) {

fprintf(stderr, "DEBUG: Error opening file %s: %s\n", file1, strerror(errno));

return -1;

}

if((fd2 = open(file2, O\_RDONLY)) < 0) {

fprintf(stderr, "DEBUG: Error opening file %s: %s\n", file2, strerror(errno));

return -1;

}

while(one = read(fd1, buf1, sizeof(buf1)) != 0){

two = read(fd2, buf2, sizeof(buf2));

if(one < 0) {

fprintf(stderr, "DEBUG: Error reading file %s: %s\n", file1, strerror(errno));

return -1;

}

if(two < 0) {

fprintf(stderr, "DEBUG: Error reading file %s: %s\n", file2, strerror(errno));

return -1;

}

if(result = memcmp(buf1, buf2, BUFSIZ) != 0){

return 1;

}

}

if(close(fd1)<0) {

fprintf(stderr, "DEBUG: Error closing file %s: %s\n", file1, strerror(errno));

return -1;

}

if(close(fd2)<0) {

fprintf(stderr, "DEBUG: Error closing file %s: %s\n", file2, strerror(errno));

return -1;

}

return 0;

}

1. **Problem Set 2 Write Up**

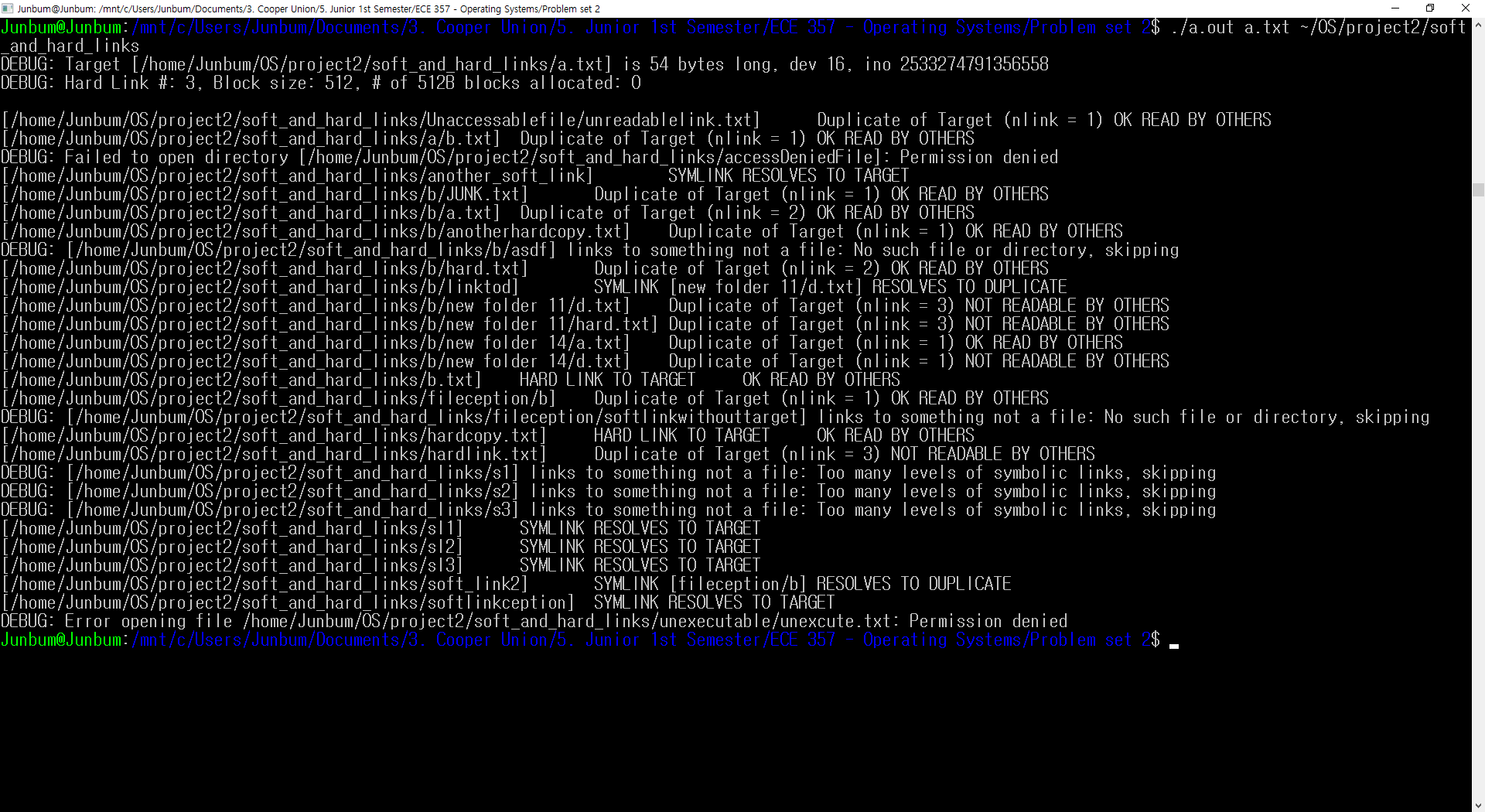
**Q. Could a file be read by an "other" user who happens to know the pathname of the matching file?**

A. Obviously, the user needs to have read permission to the file to read that file. More importantly, whether the user has traversal permission over the directories involved needs to be considered.

The way that the directory permission works is a bit different from the way that a regular file works with permissions. Write permission allows user to create directories inside a directory. Read permission allows one to list directories within a directory. Execute permission allows one to enter file and access files within them.

Going back to the original question, the answer is NO. Even if an “other” user has read permission to the pathname, the user will not be able to read file unless he or she has executable permission to all parent directory.

1. **Screenshot of a Sample Run**



1. **Extra Credit – Filesystem Corruption and Recovery**
2. Creating Ext2 file system without journaling. – Ran under OS Knoppix

**root@Microknoppix:/media/sde1# umount /dev/sde1**

umount: /dev/sde1: not mounted

**root@Microknoppix:/media/sde1# mkfs.ext2 /dev/sde1**

mke2fs 1.43.3 (04-Sep-2016)

/dev/sde1 contains a ext2 file system

created on Fri Oct 6 12:31:17 2017

Proceed anyway? (y,n) y

Creating filesystem with 3824380 4k blocks and 956592 inodes

Filesystem UUID: 59813bd9-ec17-4041-8793-bdb8316de056

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done

Writing inode tables: done

Writing superblocks and filesystem accounting information: done

**root@Microknoppix:/media/sde1# mount /dev/sde1**

**root@Microknoppix:/media/sde1# umount /dev/sde1**

**root@Microknoppix:/media/sde1# mount /dev/sde1 -v**

mount: /dev/sde1 mounted on /media/sde1.

**root@Microknoppix:/media/sdc1/log# debugfs -R features /dev/sde1**

debugfs 1.43.3 (04-Sep-2016)

Filesystem features: ext\_attr resize\_inode dir\_index filetype sparse\_super large\_file

1. Source Code Creating, Renaming, and Deleting many files.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <errno.h>

#include <fcntl.h>

#include <sys/types.h>

#include <dirent.h>

int makefile(const char \*filename, const char \*s, int c, int depth);

int removefile(const char \*filename, int c, int depth);

int main(){

char \*filename = "/media/sde1/files/randomfile";

char pathname[PATH\_MAX];

char \*s = "some random string: ";

char buf[BUFSIZ];

int fd, fdwr, fdcl, fdrm, ctr;

ctr = 1;

while(ctr++){

makefile(filename, s, ctr, 0);

removefile(filename, ctr, 0);

}

}

int makefile(const char \*filename, const char \*s, int c, int depth){

char pathname[PATH\_MAX];

char buf[BUFSIZ];

int fd, fdwr, fdcl, ctr;

if(depth++ < 6){

sprintf(pathname, "%s%d", filename, c);

sprintf(buf, "%s %d at depth: \n", s, c, depth);

fd = open(pathname, O\_WRONLY|O\_CREAT|O\_DSYNC|O\_TRUNC, 0666);

if(fd < 0) fprintf(stderr, "Error opening file %s, %s\n", pathname, strerror(errno));

fdwr = write(fd, buf, sizeof(buf));

if(fdwr < 0) fprintf(stderr, "Error writing file %s, %s\n", pathname, strerror(errno));

fdcl = close(fd);

if(fdcl < 0) fprintf(stderr, "Error closing file %s, %s\n", pathname, strerror(errno));

for(int i = 0; i < 5; i++) makefile(filename, s, 10\*c+i, depth);

}else{

return 0;

}

}

int removefile(const char \*filename, int c, int depth){

int fd;

char pathname[PATH\_MAX];

char oldpath[PATH\_MAX];

char newpath[PATH\_MAX];

if(depth++ < 3){

sprintf(pathname, "%s%d", filename, c);

fd = unlink(pathname);

if(fd < 0) fprintf(stderr,"Error deleting file %s, %s\n", pathname, strerror(errno));

for(int i = 0; i < 5; i++) removefile(filename, 10\*c+i, depth);

} else if (depth++ < 6){

sprintf(oldpath, "%s%d", filename, c);

sprintf(newpath, "%s%d", filename, c \*10000000 + c^2);

fd = rename(oldpath,newpath);

if(fd < 0) fprintf(stderr,"Error renaming file %s, %s\n", pathname, strerror(errno));

for(int i = 0; i < 5; i++) removefile(filename, 10\*c + 1, depth);

}else{

return 0;

}

}

1. FSCK for ext2 filesystem – after forcefully removing USB while the program was running

**root@Microknoppix:/media/sdc1/log# fsck.ext2 /dev/sde1**

e2fsck 1.43.3 (04-Sep-2016)

/dev/sde1 was not cleanly unmounted, check forced.

Pass 1: Checking inodes, blocks, and sizes

HTREE directory inode 833953 has an invalid root node.

Clear HTree index<y>? yes

Inode 833953, i\_size is 376832, should be 712704. Fix<y>? yes

Inode 833953, i\_blocks is 744, should be 1400. Fix<y>? yes

Running additional passes to resolve blocks claimed by more than one inode...

Pass 1B: Rescanning for multiply-claimed blocks

Multiply-claimed block(s) in inode 833953: 3371169--3371182

Multiply-claimed block(s) in inode 840117: 3371169--3371170

Multiply-claimed block(s) in inode 840118: 3371171--3371172

Multiply-claimed block(s) in inode 840119: 3371173--3371174

Multiply-claimed block(s) in inode 840120: 3371175--3371176

Multiply-claimed block(s) in inode 840121: 3371177--3371178

Multiply-claimed block(s) in inode 840122: 3371179--3371180

Multiply-claimed block(s) in inode 840123: 3371181--3371182

Pass 1C: Scanning directories for inodes with multiply-claimed blocks

Pass 1D: Reconciling multiply-claimed blocks

(There are 8 inodes containing multiply-claimed blocks.)

File /files (inode #833953, mod time Fri Oct 6 14:33:46 2017)

has 14 multiply-claimed block(s), shared with 7 file(s):

/files/randomfile324314 (inode #840123, mod time Fri Oct 6 14:33:39 2017)

/files/randomfile324313 (inode #840122, mod time Fri Oct 6 14:33:39 2017)

/files/randomfile324312 (inode #840121, mod time Fri Oct 6 14:33:39 2017)

/files/randomfile324311 (inode #840120, mod time Fri Oct 6 14:33:39 2017)

/files/randomfile324310 (inode #840119, mod time Fri Oct 6 14:33:39 2017)

/files/randomfile32431 (inode #840118, mod time Fri Oct 6 14:33:39 2017)

/files/randomfile324304 (inode #840117, mod time Fri Oct 6 14:33:39 2017)

Clone multiply-claimed blocks<y>? yes

File /files/randomfile324304 (inode #840117, mod time Fri Oct 6 14:33:39 2017)

has 2 multiply-claimed block(s), shared with 1 file(s):

/files (inode #833953, mod time Fri Oct 6 14:33:46 2017)

Multiply-claimed blocks already reassigned or cloned.

....

/dev/sde1: \*\*\*\*\* FILE SYSTEM WAS MODIFIED \*\*\*\*\*

/dev/sde1: 12397/956592 files (0.0% non-contiguous), 94240/3824380 blocks

1. Creating ext4 filesystem with journaling

**root@Microknoppix:/media# mkfs.ext4 /dev/sdd1**

mke2fs 1.43.3 (04-Sep-2016)

/dev/sdd1 contains a ext4 file system

last mounted on /media/sde1 on Fri Oct 6 15:57:33 2017

Proceed anyway? (y,n) y

Creating filesystem with 3824380 4k blocks and 956592 inodes

Filesystem UUID: 2b9f3305-de87-4c4c-b04b-16387ab6fdca

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done

Writing inode tables: done

Creating journal (16384 blocks): done

Writing superblocks and filesystem accounting information: done

1. FSCK for ext4 filesystem – after forcefully removing USB while the program was running

**root@Microknoppix:/media# fsck.ext4 /dev/sde1**

e2fsck 1.43.3 (04-Sep-2016)

/dev/sde1: recovering journal

Setting free inodes count to 956409 (was 956581)

Setting free blocks count to 3737354 (was 3737699)

/dev/sde1: clean, 183/956592 files, 87026/3824380 blocks

1. Excerpt from dmesg after the fsck was completed – seems like journaling feature in fsck.ext4 did not work correctly, but not sure what the exact problem was.

[ 251.452587] EXT4-fs warning (device sdd1): htree\_dirblock\_to\_tree:958: inode #2: lblock 0: comm ls: error -5 reading directory block

[ 253.612783] EXT4-fs warning (device sdd1): htree\_dirblock\_to\_tree:958: inode #2: lblock 0: comm ls: error -5 reading directory block

[ 256.596364] EXT4-fs warning (device sdd1): htree\_dirblock\_to\_tree:958: inode #2: lblock 0: comm bash: error -5 reading directory block

[ 256.596492] EXT4-fs warning (device sdd1): htree\_dirblock\_to\_tree:958: inode #2: lblock 0: comm bash: error -5 reading directory block

[ 257.459864] EXT4-fs error (device sdd1): ext4\_put\_super:837: Couldn't clean up the journal

[ 287.266915] EXT4-fs (sde1): mounted filesystem with ordered data mode. Opts: (null)

[ 454.122080] usb 1-1.4: USB disconnect, device number 7

[ 454.124423] sd 10:0:0:0: [sde] tag#0 FAILED Result: hostbyte=DID\_NO\_CONNECT driverbyte=DRIVER\_OK

[ 454.124426] sd 10:0:0:0: [sde] tag#0 CDB: Write(10) 2a 00 00 44 0e 92 00 00 28 00

[ 454.124427] blk\_update\_request: I/O error, dev sde, sector 4460178

[ 454.124447] Aborting journal on device sde1-8.

[ 454.124460] JBD2: Error -5 detected when updating journal superblock for sde1-8.

[ 454.144455] EXT4-fs error (device sde1): ext4\_journal\_check\_start:56: Detected aborted journal

[ 454.144460] EXT4-fs (sde1): Remounting filesystem read-only

[ 454.144472] EXT4-fs (sde1): previous I/O error to superblock detected

[ 454.184418] EXT4-fs error (device sde1): ext4\_put\_super:837: Couldn't clean up the journal

[ 455.194380] usb 1-1.4: new high-speed USB device number 8 using ehci-pci

[ 455.281874] usb 1-1.4: New USB device found, idVendor=0781, idProduct=5575

[ 455.281877] usb 1-1.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3

[ 455.281878] usb 1-1.4: Product: Cruzer Glide

[ 455.281880] usb 1-1.4: Manufacturer: SanDisk

[ 455.281881] usb 1-1.4: SerialNumber: 20044731011DE9322940

[ 455.282148] usb-storage 1-1.4:1.0: USB Mass Storage device detected

[ 455.282270] scsi host9: usb-storage 1-1.4:1.0

[ 456.285558] scsi 9:0:0:0: Direct-Access SanDisk Cruzer Glide 1.26 PQ: 0 ANSI: 6

[ 456.285726] sd 9:0:0:0: Attached scsi generic sg4 type 0

[ 456.286864] sd 9:0:0:0: [sdd] 15633408 512-byte logical blocks: (8.00 GB/7.45 GiB)

[ 456.288238] sd 9:0:0:0: [sdd] Write Protect is off

[ 456.288241] sd 9:0:0:0: [sdd] Mode Sense: 43 00 00 00

[ 456.289235] sd 9:0:0:0: [sdd] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA

[ 456.307759] sdd: sdd1

[ 456.311269] sd 9:0:0:0: [sdd] Attached SCSI removable disk

[ 481.505337] EXT4-fs (sdd1): mounted filesystem with ordered data mode. Opts: (null)

[ 506.377375] aufs may\_rename\_srcdir:453:URL Classifier[4136]: renaming dir who has child(ren) on multiple branches, is not supported

[ 584.026263] usb 1-1.4: USB disconnect, device number 8

[ 584.032600] Buffer I/O error on dev sdd1, logical block 0, lost sync page write

[ 584.032603] EXT4-fs (sdd1): re-mounted. Opts: (null)

[ 584.983255] usb 1-1.4: new high-speed USB device number 9 using ehci-pci

[ 585.071305] usb 1-1.4: New USB device found, idVendor=0781, idProduct=5575

[ 585.071308] usb 1-1.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3

[ 585.071310] usb 1-1.4: Product: Cruzer Glide

[ 585.071311] usb 1-1.4: Manufacturer: SanDisk

[ 585.071312] usb 1-1.4: SerialNumber: 20044731011DE9322940

[ 585.071634] usb-storage 1-1.4:1.0: USB Mass Storage device detected

[ 585.071755] scsi host10: usb-storage 1-1.4:1.0

[ 586.074508] scsi 10:0:0:0: Direct-Access SanDisk Cruzer Glide 1.26 PQ: 0 ANSI: 6

[ 586.074626] sd 10:0:0:0: Attached scsi generic sg4 type 0

[ 586.075851] sd 10:0:0:0: [sde] 15633408 512-byte logical blocks: (8.00 GB/7.45 GiB)

[ 586.077240] sd 10:0:0:0: [sde] Write Protect is off

[ 586.077245] sd 10:0:0:0: [sde] Mode Sense: 43 00 00 00

[ 586.078225] sd 10:0:0:0: [sde] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA

[ 586.096936] sde: sde1

[ 586.100623] sd 10:0:0:0: [sde] Attached SCSI removable disk

[ 588.697941] EXT4-fs warning (device sdd1): htree\_dirblock\_to\_tree:958: inode #2: lblock 0: comm ls: error -5 reading directory block

[ 602.268620] EXT4-fs (sde1): recovery complete

[ 602.268623] EXT4-fs (sde1): mounted filesystem with ordered data mode. Opts: (null)