Joseph Bentivegna

Professor Hakner

Project 2

10/8/17

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <errno.h>

ino\_t targINo;

off\_t targSize;

dev\_t targDev;

char \*target;

int dirPerm;

int parsefs(char\* direct);

int checker(char\* file);

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

char \*start;

struct stat st;

dirPerm = 1;

if (argc != 3) {

printf("Usage: ./hunt [filename] [starting path]\n");

return -1;

} else {

target = argv[1];

start = argv[2];

}

if (stat(target, &st) < 0) {

fprintf(stderr, "Cannot stat target file %s: %s\n", target, strerror(errno));

return -1;

} else {

targINo = st.st\_ino;

targSize = st.st\_size;

targDev = st.st\_dev;

}

parsefs(start);

return 0;

}

int parsefs(char \*direc) {

DIR \*dir;

struct dirent \*entry;

struct stat st;

ino\_t tempINo, sysINo;

nlink\_t tempLink;

off\_t tempSize, sysSize;

dev\_t tempDev, sysDev;

mode\_t tempMode;

int r;

char \*permStr;

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

fprintf(stderr, "Warning: Cannot open directory %s for reading: %s\n", direc, strerror(errno));

return -1;

}

while ((entry = readdir(dir)) != NULL) {

char path[1024];

char link[1024];

snprintf(path, sizeof(path), "%s/%s", direc, entry->d\_name);

if (stat(path, &st) < 0) {

fprintf(stderr, "Cannot stat file %s: %s\n", path, strerror(errno));

return -1;

}

if (entry->d\_type == DT\_DIR) {

if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0) {

continue;

}

if ((st.st\_mode & S\_IXOTH) && (dirPerm == 1)) {

dirPerm = 1;

} else {

dirPerm = 0;

}

parsefs(path);

} else if (entry->d\_type == DT\_REG) {

tempINo = st.st\_ino;

tempLink = st.st\_nlink;

tempSize = st.st\_size;

tempDev = st.st\_dev;

if (tempMode & S\_IROTH) {

permStr = "Read by Other: Y";

} else {

permStr = "Read by Other: N";

}

if (tempINo == targINo && tempDev == targDev) {

printf("%s Hard Link to Target %s\n", path, permStr);

} else if (tempSize == targSize) {

if (checker(path) == 1) {

printf("%s Duplicate of Target (nlink = %hu) %s\n",path, tempLink, permStr);

}

}

} else if (entry->d\_type == DT\_LNK) {

if((r = readlink(path, link, sizeof(link))) < 0) {

fprintf(stderr, "Cannot read symlink %s: %s\n", path, strerror(errno));

return -1;

}

stat(link, &st);

sysINo = st.st\_ino;

sysSize = st.st\_size;

sysDev = st.st\_dev;

if (sysINo == targINo && sysDev == targDev) {

printf("%s Symlink Resolves to Target\n", path);

} else if (sysSize = targSize) {

if (checker(link) == 1) {

printf("%s Symlink (%s) Resolves to Duplicate\n", path, link);

}

}

} else {

fprintf(stderr, "Warning: %s has unknown type, skipping...", path);

return -1;

}

}

if (closedir(dir) < 0) {

fprintf(stderr, "Can't close directory %s: %s\n", dir, strerror(errno));

return -1;

}

return 0;

}

int checker(char\* file) {

int fd1, fd2, n, m, ret, cl1, cl2;

char buff1[BUFSIZ];

char buff2[BUFSIZ];

memset(buff1, 0, BUFSIZ);

memset(buff2, 0, BUFSIZ);

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

fprintf(stderr, "Cannot open file %s for reading: %s\n", file, strerror(errno));

return 0;

}

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

fprintf(stderr, "Cannot open file %s for reading: %s\n", target, strerror(errno));

return 0;

}

while (((n = read(fd1, buff1, sizeof(buff1))) != 0) && (m = read(fd2, buff2, sizeof(buff2)) != 0)) {

if ((ret = memcmp(buff1, buff2, BUFSIZ)) != 0) {

return 0;

}

}

if (cl1 = close(fd1) < 0) {

fprintf(stderr, "Cannot close file %s: %s\n", file, strerror(errno));

}

if (cl2 = close(fd2) < 0) {

fprintf(stderr, "Cannot close file %s: %s\n", target, strerror(errno));

}

return 1;

}