Homework Wan Huzaifah bin Wan Azhar

Answer:

Files and Directories

1. Stat

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <stdint.h>

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/sysmacros.h>

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

{

struct stat file\_info;

if (stat(argv[1], &file\_info) == -1) {

perror("stat");

exit(EXIT\_FAILURE);

}

//file name

printf("File: %s\n", argv[1]);

//file inode

printf("File inode: %d\n", file\_info.st\_ino);

//file size

printf("File size: %d\n", file\_info.st\_size);

//Number of block alloc

printf("Block allocated: %d\n", file\_info.st\_blocks);

//Ref link count

printf("Reference link count: %d\n", file\_info.st\_nlink);

return 0;

}

* Link count of a directory changes when the directory in the directory is added or removed
* Link count will not change if files is added or removed.

1. List Files

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <stdint.h>

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/sysmacros.h>

#include <dirent.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdbool.h>

struct stat statfile(char\* filepath) {

struct stat file\_info;

if (stat(filepath, &file\_info) == -1) {

perror("stat");

exit(EXIT\_FAILURE);

}

return file\_info;

}

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

{

char\* temp\_file\_path[256];

char\* working\_path;

struct stat temp\_file\_info;

struct dirent \*currentDirEnt;

DIR \*currentDir;

bool getaddfileinfo = false;

if (argc == 1) {

char\* wd;

getwd(wd);

working\_path = wd;

currentDir = opendir(wd);

}

else if (argc == 2) {

if (strcmp("-l", argv[1]) == 0) {

getaddfileinfo = true;

char\* wd;

getwd(wd);

working\_path = wd;

currentDir = opendir(wd);

}

else {

currentDir = opendir(argv[1]);

working\_path = argv[1];

}

}

else if (argc == 3) {

if (strcmp("-l", argv[1]) == 0) {

getaddfileinfo = true;

currentDir = opendir(argv[2]);

working\_path = argv[2];

}

else if (strcmp("-l", argv[2]) == 0) {

getaddfileinfo = true;

currentDir = opendir(argv[1]);

working\_path = argv[1];

}

}

if (currentDir == NULL) {

printf("Failed to open directory %s\n", argv[1]);

return 1;

}

if (getaddfileinfo == true)

printf("Name\tinode\tsize\tblock\_allocated\treference\_link\taccess\_mode\n");

while ((currentDirEnt = readdir(currentDir)) != NULL) {

if (getaddfileinfo == true) {

if (strcmp(currentDirEnt->d\_name, ".") == 0)

continue;

else if (strcmp(currentDirEnt->d\_name, "..") == 0)

continue;

strcpy(temp\_file\_path, working\_path);

strcat(temp\_file\_path, "/");

strcat(temp\_file\_path, currentDirEnt->d\_name);

temp\_file\_info = statfile(temp\_file\_path);

//file name

printf("%s\t", currentDirEnt->d\_name);

//file inode

printf("%d\t", currentDirEnt->d\_ino);

//file size

printf("%d\t", temp\_file\_info.st\_size);

//Number of block alloc

printf("%d\t", temp\_file\_info.st\_blocks);

//Ref link count

printf("%d\t", temp\_file\_info.st\_nlink);

//Permission

printf("%d\n", temp\_file\_info.st\_mode);

temp\_file\_path[0] = 0;

}

else {

printf("%s\n", currentDirEnt->d\_name);

}

}

return 0;

}

1. Tail

Honestly proud of this one, show how improved I am from my bad C skill.

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <string.h>

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

{

int n = atoi(argv[1]);

n = abs(n);

int fd = open(argv[2], O\_RDONLY);

if (fd < 0) {

perror("open");

exit(EXIT\_FAILURE);

}

char buf[BUFSIZ+1];

lseek(fd, 0, SEEK\_END);

int currOffset;

while (n > 0) {

currOffset = lseek(fd, -2, SEEK\_CUR);

if (currOffset == 0) //If start of text is reached

break;

else { //Rewind until last n line is reached

read(fd, buf, 1);

if (strcmp(buf,"\n") == 0)

n--;

}

}

while (read(fd, buf, 1) > 0) {

printf("%s", buf);

}

close(fd);

}

1. Recursive Search

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <stdint.h>

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/sysmacros.h>

#include <dirent.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdbool.h>

#include <limits.h>

struct stat statfile(char\* filepath) {

struct stat file\_info;

if (stat(filepath, &file\_info) == -1) {

perror("stat");

exit(EXIT\_FAILURE);

}

return file\_info;

}

void print\_folder(char\* absPath, int level) {

char absCurrentPath[256];

DIR \*currentDir = opendir(absPath);

struct dirent \*currentDirEnt;

struct stat currentFileStat;

if (currentDir == NULL) {

printf("Failed to open directory %s\n", absPath);

exit(EXIT\_FAILURE);

}

level++;

while ((currentDirEnt = readdir(currentDir)) != NULL) {

if (strcmp(currentDirEnt->d\_name, ".") == 0)

continue;

else if (strcmp(currentDirEnt->d\_name, "..") == 0)

continue;

strcpy(absCurrentPath, absPath);

strcat(absCurrentPath, "/");

strcat(absCurrentPath, currentDirEnt->d\_name);

currentFileStat = statfile(absCurrentPath);

for (int i = 0; i < level; i++) {

printf("-");

}

printf("%s\n", currentDirEnt->d\_name);

if (S\_ISDIR(currentFileStat.st\_mode)) {

print\_folder(absCurrentPath, level);

}

absCurrentPath[0] = 0;

}

}

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

{

char\* rootDir;

if (argc == 1) {

char cwd[PATH\_MAX];

if (getcwd(cwd, sizeof(cwd)) != NULL)

rootDir = cwd;

else {

perror("getcwd() error");

return 1;

}

}

else if (argc == 2) {

rootDir = argv[1];

}

print\_folder(rootDir, -1);

}

Output:

