



#include <dirent.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <string>

#include <memory>

#include <vector>

#include <stdlib.h>

#include <malloc.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <limits.h>

#include <errno.h>

#include <pthread.h>

#include <sys/mman.h>

#define PATH "/home/consta/practice1/1/Test1/"

#define BUFFER\_SIZE 256

#include <stdio.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

bool is\_pseudodirectory(struct dirent \*ent)

{

return strcmp(ent->d\_name, (".")) == 0 || strcmp(ent->d\_name, ("..")) == 0;

}

void get\_old\_name(char \*path, struct dirent \*ent, char \*old\_file\_name)

{

sprintf(old\_file\_name, "%s%s", path, ent->d\_name);

}

void get\_new\_path(char \*path, struct dirent \*ent, char \*new\_path)

{

sprintf(new\_path, "%s%s%s", path, ent->d\_name, "/");

}

bool is\_directory(char \*path\_, struct stat \*st)

{

stat(path\_, st);

return S\_ISDIR(st->st\_mode);

}

struct DirInfo

{

int file\_count;

char path[BUFFER\_SIZE];

};

void UpdateMin(DirInfo \*old\_min, DirInfo \*new\_info)

{

if (old\_min->file\_count <= new\_info->file\_count)

return;

old\_min->file\_count = new\_info->file\_count;

strcpy(old\_min->path, new\_info->path);

}

int fd[2];

void count\_files(char \*path)

{

DIR \*dir = opendir(path);

DirInfo dir\_info;

strcpy(dir\_info.path, path);

dir\_info.file\_count = 0;

while (struct dirent \*ent = readdir(dir))

{

struct stat sb;

char file\_name[BUFFER\_SIZE] = "";

get\_old\_name(path, ent, file\_name);

if (stat(file\_name, &sb) == -1)

{

continue;

}

if (is\_pseudodirectory(ent))

{

continue;

}

if (!(sb.st\_mode & S\_IFREG))

{

continue;

}

if (!(sb.st\_mode & S\_IRUSR))

{

continue;

}

// if ((sb.st\_mode & S\_IRWXU))

// {

// continue;

// }

printf("файл на чтение");

dir\_info.file\_count += 1;

}

closedir(dir);

write(fd[1], &dir\_info, sizeof(DirInfo));

}

void \*rec\_dir\_helper(void \*path);

void rec\_dir(char \*path)

{

count\_files(path);

DIR \*dir = opendir(path);

std::vector<std::unique\_ptr<char[]>> paths;

std::vector<pthread\_t> threads;

while (struct dirent \*ent = readdir(dir))

{

struct stat sb;

char dir\_path[BUFFER\_SIZE] = "";

get\_old\_name(path, ent, dir\_path);

if (!(is\_directory(dir\_path, &sb)))

{

continue;

}

if (is\_pseudodirectory(ent))

{

continue;

}

paths.emplace\_back(new char[BUFFER\_SIZE]());

get\_new\_path(path, ent, paths.back().get());

threads.emplace\_back();

pthread\_create(&threads.back(), NULL, rec\_dir\_helper, (void \*)paths.back().get());

}

for (int i = 0; i < threads.size(); i += 1)

{

pthread\_join(threads[i], NULL);

}

closedir(dir);

}

void \*rec\_dir\_helper(void \*path)

{

rec\_dir((char \*)path);

return NULL;

}

void \*main\_rec\_dir(void \*path)

{

rec\_dir((char \*)path);

close(fd[1]);

return NULL;

}

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

{

pipe(fd);

DirInfo answer;

answer.file\_count = INT\_MAX;

strcpy(answer.path, "default dir");

pthread\_t rec\_dir\_thread;

pthread\_create(&rec\_dir\_thread, NULL, main\_rec\_dir, (void \*)PATH);

DirInfo buffer;

while (read(fd[0], &buffer, sizeof(DirInfo)))

{

UpdateMin(&answer, &buffer);

}

printf("%s\n", answer.path);

}