УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №7

по предмету «Архитектура компьютерной техники и операционных систем»

Вариант 2

Выполнил:

Бражалович А.И.

Проверил:

[Аврамец](https://iis.bsuir.by/employees/d-avramets) Д.В.

Группа 351004

Минск 2024

**Задание**

1. Написать программу синхронизации двух каталогов, например, ***Dir1*** и ***Dir2***. Пользователь задаёт имена ***Dir1*** и ***Dir2***в качестве первого и второго аргумента командной строки. В результате работы программы файлы, имеющиеся в ***Dir1***, но отсутствующие в ***Dir2***, должны скопироваться в ***Dir2*** вместе с правами доступа. Процедуры копирования должны запускаться в отдельном процессе для каждого копируемого файла с использованием функций ***read ()*** и ***write ()***. Каждый процесс выводит на экран свой ***pid,*** полный путь, имя копируемого файла и число скопированных байт. Число запущенных процессов в любой момент времени не должно превышать ***N*** (вводится пользователем).

**Код программы 7.1**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <string.h>

#define MAX\_PATH 1024

#define BUFFER\_SIZE 4096

void copy\_(char\* src\_path, char\* dst\_path) {

int src\_fd, dst\_fd;

ssize\_t n\_read, n\_written, total\_written = 0;

char buffer[BUFFER\_SIZE];

struct stat st;

src\_fd = open(src\_path, O\_RDONLY);

if (src\_fd < 0) {

perror("open src");

} else {

if (fstat(src\_fd, &st) < 0) {

perror("fstat");

return;

} else {

umask(0);

dst\_fd = open(dst\_path, O\_WRONLY | O\_CREAT | O\_TRUNC, st.st\_mode);

if (dst\_fd < 0) {

perror("open dst");

close(src\_fd);

return;

} else {

while ((n\_read = read(src\_fd, buffer, BUFFER\_SIZE)) > 0) {

n\_written = write(dst\_fd, buffer, n\_read);

if (n\_written != n\_read) {

perror("write");

close(src\_fd);

close(dst\_fd);

return;

}

total\_written += n\_written;

}

close(src\_fd);

close(dst\_fd);

printf("PID: %d, File: %s, Bytes: %zd\n", getpid(), dst\_path, total\_written);

}

}

}

}

void copy\_\_Files(char\* src\_path, char\* dst\_path, int max\_proc) {

DIR \*direct;

int active\_processes = 0;

struct dirent \*entry;

struct stat st;

char s\_path[MAX\_PATH];

char d\_path[MAX\_PATH];

direct = opendir(src\_path);

if (!direct) {

printf("Ошибка открытия первой директории!\n");

} else {

while (entry = readdir(direct)) {

snprintf(s\_path, MAX\_PATH, "%s/%s", src\_path, entry->d\_name);

if (stat(s\_path, &st) == 0) {

if (S\_ISREG(st.st\_mode)) {

snprintf(d\_path, MAX\_PATH, "%s/%s", dst\_path, entry->d\_name);

if (stat(d\_path, &st) != 0) {

if (active\_processes >= max\_proc) {

wait(NULL);

active\_processes--;

}

pid\_t pid = fork();

if (pid == 0) {

//

copy\_(s\_path, d\_path);

//

} else if (pid > 0) {

active\_processes++;

} else {

printf("processor error");

}

}

}

}

}

}

while (active\_processes > 0) {

wait(NULL);

active\_processes--;

}

}

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

if (argc != 4)

printf("Введите путь <src dir> <dst dir> <max\_proc>\n");

else {

char \*src\_path = argv[1];

char \*dst\_path = argv[2];

int max\_proc = atoi(argv[3]);

copy\_\_Files(src\_path, dst\_path, max\_proc);

}

return 0;

}

**Код программы 7.2**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <string.h>

#include <pthread.h>

#define MAX\_PATH 1024

#define BUFFER\_SIZE 4096

typedef struct {

char src\_path[MAX\_PATH];

char dst\_path[MAX\_PATH];

} copy\_args;

void copy\_(void \*new\_args) {

copy\_args \*paths = (copy\_args \*)new\_args;

int src\_fd, dst\_fd;

ssize\_t n\_read, n\_written, total\_written = 0;

char buffer[BUFFER\_SIZE];

struct stat st;

src\_fd = open(paths->src\_path, O\_RDONLY);

if (src\_fd < 0) {

perror("open src");

pthread\_exit(NULL);

} else {

if (fstat(src\_fd, &st) < 0) {

perror("fstat");

pthread\_exit(NULL);

return;

} else {

umask(0);

dst\_fd = open(paths->dst\_path, O\_WRONLY | O\_CREAT | O\_TRUNC, st.st\_mode);

if (dst\_fd < 0) {

perror("open dst");

close(src\_fd);

pthread\_exit(NULL);

return;

} else {

while ((n\_read = read(src\_fd, buffer, BUFFER\_SIZE)) > 0) {

n\_written = write(dst\_fd, buffer, n\_read);

if (n\_written != n\_read) {

perror("write");

close(src\_fd);

close(dst\_fd);

pthread\_exit(NULL);

return;

}

total\_written += n\_written;

}

close(src\_fd);

close(dst\_fd);

printf("File: %s, Bytes: %zd\n",paths->dst\_path, total\_written);

pthread\_exit(NULL);

}

}

}

}

void copy\_\_Files(char\* src\_path, char\* dst\_path, int max\_threads) {

DIR \*direct;

int active\_threads = 0;

struct dirent \*entry;

struct stat st;

pthread\_t threads[max\_threads];

char s\_path[MAX\_PATH];

char d\_path[MAX\_PATH];

direct = opendir(src\_path);

if (!direct) {

printf("Ошибка открытия первой директории!\n");

} else {

while (entry = readdir(direct)) {

snprintf(s\_path, MAX\_PATH, "%s/%s", src\_path, entry->d\_name);

if (stat(s\_path, &st) == 0) {

if (S\_ISREG(st.st\_mode)) {

snprintf(d\_path, MAX\_PATH, "%s/%s", dst\_path, entry->d\_name);

if (stat(d\_path, &st) != 0) {

if (active\_threads >= max\_threads) {

pthread\_join(threads[active\_threads--], NULL);

}

copy\_args \*new\_arg = malloc(sizeof(copy\_args));

strcpy(new\_arg->src\_path, s\_path);

strcpy(new\_arg->dst\_path, d\_path);

if (pthread\_create(&threads[active\_threads++], NULL, copy\_, new\_arg)) {

printf("processor error");

free(new\_arg);

}

}

}

}

}

}

for (int i = 0; i < active\_threads; i++) {

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

}

}

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

if (argc != 4)

printf("Введите путь <src dir> <dst dir> <max\_proc>\n");

else {

char \*src\_path = argv[1];

char \*dst\_path = argv[2];

int max\_threads = atoi(argv[3]);

copy\_\_Files(src\_path, dst\_path, max\_threads);

}

return 0;

}

**Блок-схема алгоритма**

