УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе № 7

по предмету «Архитектура компьютерной техники и операционных систем»

Вариант 4

Выполнил:

Гузаев Е.Д.

гр. 351003

Проверил:

Леванцевич В.А.

Минск 2024

Задание:

Для заданного каталога (аргумент 1 командной строки) и всех его подкаталогов вывести в заданный файл (аргумент 2 командной строки) и на консоль имена файлов, их размер и дату создания, удовлетворяющих заданным условиям: 1 – размер файла находится в заданных пределах от N1 до N2 (N1,N2задаются в аргументах командной строки), 2 – дата создания находится в заданных пределах от M1 до M2 (M1,M2задаются в аргументах командной строки).Процедура поиска для каждого подкаталога должна запускаться в отдельном процессе. Каждый процесс выводит на экран свой pid, полный путь, имя и размер просмотренного файла, общее число просмотренных файлов в подкаталоге. Число запущенных процессов в любой момент времени не должно превышать N (вводится пользователем)

Решение:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <dirent.h>

#include <time.h>

#include <fcntl.h>

#include <semaphore.h>

#define SEM\_NAME "/process\_limit"

void process\_directory(const char \*dir\_path, long min\_size, long max\_size, time\_t min\_date, time\_t max\_date, FILE \*output) {

DIR \*dir = opendir(dir\_path);

if (!dir) {

perror("opendir");

return;

}

struct dirent \*entry;

struct stat file\_stat;

char full\_path[PATH\_MAX];

int file\_count = 0;

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

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

continue;

snprintf(full\_path, PATH\_MAX, "%s/%s", dir\_path, entry->d\_name);

if (stat(full\_path, &file\_stat) == -1) {

perror("stat");

continue;

}

if (S\_ISDIR(file\_stat.st\_mode)) {

// Запуск нового процесса для обработки подкаталога

pid\_t pid = fork();

if (pid == 0) { // Дочерний процесс

process\_directory(full\_path, min\_size, max\_size, min\_date, max\_date, output);

exit(0);

} else if (pid > 0) { // Родительский процесс

wait(NULL); // Ожидание завершения дочернего процесса

} else {

perror("fork");

}

} else if (S\_ISREG(file\_stat.st\_mode)) {

// Проверка файла на соответствие условиям

if (file\_stat.st\_size >= min\_size && file\_stat.st\_size <= max\_size &&

file\_stat.st\_mtime >= min\_date && file\_stat.st\_mtime <= max\_date) {

char time\_buf[64];

strftime(time\_buf, sizeof(time\_buf), "%Y-%m-%d %H:%M:%S", localtime(&file\_stat.st\_mtime));

fprintf(output, "PID: %d, Path: %s, Size: %ld bytes, Date: %s\n",

getpid(), full\_path, file\_stat.st\_size, time\_buf);

printf("PID: %d, Path: %s, Size: %ld bytes, Date: %s\n",

getpid(), full\_path, file\_stat.st\_size, time\_buf);

file\_count++;

}

}

}

closedir(dir);

printf("PID: %d, Processed %d files in directory: %s\n", getpid(), file\_count, dir\_path);

}

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

if (argc != 8) {

fprintf(stderr, "Usage: %s <directory> <output\_file> <min\_size> <max\_size> <min\_date> <max\_date> <max\_processes>\n", argv[0]);

return EXIT\_FAILURE;

}

const char \*dir\_path = argv[1];

const char \*output\_file = argv[2];

long min\_size = atol(argv[3]);

long max\_size = atol(argv[4]);

// Преобразование дат в формат time\_t

struct tm tm;

time\_t min\_date, max\_date;

if (strptime(argv[5], "%Y-%m-%d", &tm) == NULL) {

fprintf(stderr, "Invalid min\_date format. Use YYYY-MM-DD.\n");

return EXIT\_FAILURE;

}

min\_date = mktime(&tm);

if (strptime(argv[6], "%Y-%m-%d", &tm) == NULL) {

fprintf(stderr, "Invalid max\_date format. Use YYYY-MM-DD.\n");

return EXIT\_FAILURE;

}

max\_date = mktime(&tm);

int max\_processes = atoi(argv[7]);

// Создание семафора для ограничения числа процессов

sem\_t \*sem = sem\_open(SEM\_NAME, O\_CREAT, 0644, max\_processes);

if (sem == SEM\_FAILED) {

perror("sem\_open");

return EXIT\_FAILURE;

}

FILE \*output = fopen(output\_file, "w");

if (!output) {

perror("fopen");

sem\_close(sem);

sem\_unlink(SEM\_NAME);

return EXIT\_FAILURE;

}

process\_directory(dir\_path, min\_size, max\_size, min\_date, max\_date, output);

fclose(output);

sem\_close(sem);

sem\_unlink(SEM\_NAME);

return EXIT\_SUCCESS;

}