МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО"

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №1**

з дисципліни “Системне програмування”

тема “Управління файловою системою та методи

мультиплексованого вводу-виводу”

Виконав студент 3 курсу

групи КП-92

Сусленко Матвій Олегович

**Тема**: Єдиний державний реєстр осіб, які вчинили корупційні правопорушення

Київ 2021

Завдання

**Частина 1: Функції послідовного вводу-виводу.**

Скласти програму, яка:

1. Отримує аргументами командної строки два імені файлів

2. Перший файл відкриває для читання, другий до перезапису, або створює, якщо його

не існує (для створення використати атрибути 0644)

3. Послідовно читає дані з першого файлу буферами фіксованого об’єму (наприклад

512 байт) та записує у другий.

4. Перед записом із вмістом буферу проводить таке перетворення – усі рядкові літери

латинського алфавіту перетворює на відповідні прописні. Для цього розрахувати

відповідний зсув за таблицею ASCII (див. man ascii).

5. В процесі перезапису проводить підрахунок перезаписаних байт, та наприкінці

виводить у вихідний потік сумарний обсяг переписаних даних.

|  |
| --- |
| #include <stdio.h>  #include <fcntl.h>  #include <string.h>  #include <errno.h>  #include <stdlib.h>  #include <unistd.h>  #define BUFFER\_SIZE 512  #define REQUIRED\_ARG\_COUNT 3  #define WRITE\_FILE\_CREATE\_RULES 0644  #define ASCII\_UPPECASE\_MIN 65  #define ASCII\_UPPECASE\_MAX 90  #define ASCII\_DIFF 32  void throwError(char\* message) {  fprintf(stderr, "%s: %s\n", message, strerror(errno));  exit(1);  }  char\* initializeBuffer(size\_t size) {  char\* buffer = malloc(size);  memset(buffer, 0, size);  return buffer;  }  void toLower(char\* string, size\_t length) {  for (size\_t i = 0; i < length; i++) {  char letter = string[i];  if (letter >= ASCII\_UPPECASE\_MIN && letter <= ASCII\_UPPECASE\_MAX) {  string[i] = letter + ASCII\_DIFF;  }  }  }  int main(int argc, char\* argv[]) {  if (argc != REQUIRED\_ARG\_COUNT) {  throwError("Invalid count of input arguments");  }  char \* readFileName = argv[1];  char \* writeFileName = argv[2];  // read file  int readFileDescriptor = open(readFileName, O\_RDONLY);  if (readFileDescriptor == -1) {  throwError("Read file can't be opened!");  }  // write to file  int writeFileDescriptor = open(writeFileName, O\_WRONLY | O\_CREAT | O\_TRUNC, WRITE\_FILE\_CREATE\_RULES);  if (writeFileDescriptor == -1) {  throwError("Write file can't be opened!");  }  char \*buffer = initializeBuffer(BUFFER\_SIZE);  int readBytesCount = 0;  int allBytesCount = 0;  while((readBytesCount = read(readFileDescriptor, buffer, BUFFER\_SIZE)) > 0) {  toLower(buffer, readBytesCount);  write(writeFileDescriptor, buffer, readBytesCount);  allBytesCount += readBytesCount;  }  // read from file error  if (readBytesCount == -1) {  throwError("Read file error");  }  printf("Rewritten bytes: %i\n", allBytesCount);  // end stage  free(buffer);  close(readFileDescriptor);  close(writeFileDescriptor);  return 0;  } |

**Частина 2: Функції мультиплексованого вводу-виводу**.

Скласти програму, яка:

1. Отримує аргументом командної строки довільний строковий ідентифікатор.

2. Налаштовує системний виклик select для очікування читання у вхідному потоці (

файловий дескріптор STDIN\_FILENO ) з таймаутом 5 секунд.

3. При отриманні можливості читання, прочитати з потоку буфер довжиною не більше

1024 байта та вивести його у вихідний потік з поміткою у вигляді ідентифікатора п.1.

4. При спливанні таймауту, вивести у потік помилок повідомлення про це з поміткою у

вигляді ідентифікатора п.1 та знову налаштувати системний виклик select (п.2).

5. Протестувати роботу програми отримуючи через вхідний потік результати вводу з

клавіатури.

|  |
| --- |
| #include <stdio.h>  #include <errno.h>  #include <sys/select.h>  #include <unistd.h>  #include <string.h>  #include <stdlib.h>  #define REQUIRED\_ARG\_COUNT 2  #define TIME\_INTERVAL\_SECONDS 5  #define BUFFER\_SIZE 1024  void throwError(char\* message) {  fprintf(stderr, "%s: %s\n", message, strerror(errno));  exit(1);  }  char\* initializeBuffer(size\_t size) {  char\* buffer = malloc(size);  memset(buffer, 0, size);  return buffer;  }  int main(int argc, char\* argv[]) {  if (argc != REQUIRED\_ARG\_COUNT) {  throwError("Invalid count of input arguments");  }  fd\_set readFileDescriptors;  struct timeval timeValue;  // initialize stage  FD\_ZERO(&readFileDescriptors);  FD\_SET(0, &readFileDescriptors);  timeValue.tv\_sec = TIME\_INTERVAL\_SECONDS;  timeValue.tv\_usec = 0;  // main stage  char\* key = argv[1];  char\* buffer = initializeBuffer(BUFFER\_SIZE);  int selectStatus = 0;  while((selectStatus = select(STDIN\_FILENO + 1, &readFileDescriptors, NULL, NULL, &timeValue)) != -1) {  if (selectStatus) {  int readBytes = read(STDIN\_FILENO, buffer, BUFFER\_SIZE);  if (readBytes == -1) {  throwError("Read error");  }  fprintf(stdout, "[%s]: %s", key, buffer);  } else {  fprintf(stderr, "[%s] No input at last 5 seconds\n", key);  }  FD\_ZERO(&readFileDescriptors);  FD\_SET(STDIN\_FILENO, &readFileDescriptors);  timeValue.tv\_sec = 5;  timeValue.tv\_usec = 0;  memset(buffer, 0, BUFFER\_SIZE);  }  if (selectStatus == -1) {  throwError("Select crashed");  }  // end stage  free(buffer);  return 0;  } |