Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет: компьютерных систем и сетей

Кафедра: электронных вычислительных машин

Дисциплина: схемотехника

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

МИКРОПРОЦЕССОРНОЕ УСТРОЙСТВО КОНТРОЛЯ

КОНЦЕНТРАЦИИ МЕТАНА

БГУИР КП 1-40 02 01 109 ПЗ

Студент: гр. 150501 Кардаш С.П.

Руководитель: Селезнев И.Л.

Минск 2023

**СОДЕРЖАНИЕ**

Введение………………………………………………………………………….5

1 Обзор литературы……………………………………………………………...7

   1.1 Анализ аналогов……………………………………………………………7

         1.1.1 Утилита htop…………………………………………………………7

         1.1.2 Диспетчер задач Windows……………………………………………8

   1.2 Постановка задачи…………………………………………………………9

2 Системное проектирование…………………………………………………...11

   2.1 Модули приложения………………………………………………………11

        2.1.1 Модуль чтения данных…………..…………………………………..11

        2.1.2 Модуль преобразования данных…………………………………….11

        2.1.3 Модуль оперативного хранения..……………………………………11

        2.1.4 Модуль пользовательского интерфейса..…………………………...11

        2.1.5 Модуль работы приложения..………………………………………..12

   2.2 Структурная схема………………………………………………………....12

3 Функциональное проектирование…………………………………………….13

   3.1 Структура данных myTime………………………………………………...13

   3.2 Структура данных taskData………….......................................................13

   3.3 Структура данных taskScheduledTime…………………………………13

   3.4 Структура данных commonCpuTime……………………………………….14

   3.5 Структура данных sortType……………………………………………...14

   3.6 Структура данных searchType…………………………………………...15

4 Разработка программных модулей…………………………………………...16

   4.1 Алгоритм по шагам функции fillMemory………………………………16

   4.2 Алгоритм по шагам функции getHostName……..………………………17

   4.3 Схема алгоритма функции getTaskList.………………………………..18

   4.4 Схема алгоритма функции getUptime…………………………………...18

5 Результаты……………………………………………………………………...19

   5.1 Тестирование………………………………………………………………19

   5.2 Программные и системные ограничения………………………………...21

   5.3 Код программы…………………………………………………………….22

Заключение………………………………………………………………………23

Список используемых источников……………………………………………..24

Приложение А (обязательное) Схема структурная……………………………25

Приложение Б (обязательное) Схема алгоритма функции getTaskList……26

Приложение В (обязательное) Схема алгоритма функции getUptime………27

Приложение Г (обязательное) Код программы………………………………..28

Приложение Д (обязательное) Ведомость документов……………………….29

**ВВЕДЕНИЕ**

Природный газ - это смесь газов, основную часть которой составляет метан. Как известно, метан не имеет ни цвета, ни запаха, поэтому при использовании в быту, промышленности в него обычно добавляют одоранты.

Сам по себе метан является наиболее взрывоопасным газом из углеводородов (CxHy).

Природный газ нашёл широкое применение в качестве горючего в жилых частных и многоквартирных домах для отопления, подогрева воды и приготовления пищи; как топливо для автомобилей, котельных, агрегатов, ТЭЦ и других объектов.

Поскольку метан используется практически во всех сферах жизни человека, а также является взрывоопасным газом, очень важно непрерывно отслеживать его предельные концентрации. Так как органы человека не позволяют точно определять концентрации газов, нужно прибегать к использованию электронных устройств.

Устройства, которые определяют концентрации газов в воздухе, называются газоанализаторы. Газоанализаторы могут быть стационарными и переносными, а также различаться по типу подачи пробы диффузионная или принудительная. Однако основной задачей газоанализаторов является сигнализирование или выполнение каких-либо действий при выходе за допустимые границы концентрации газов.

**1 ОБЗОР ЛИТЕРАТУРЫ**

* 1. **Требования к проектируемому устройству**

Основной функцией проектируемого устройства является измерение концентрации метана(пропана) в помещениях. При выходе измеряемых значений за пределы нормы устройство должно каким-либо образом оповестить пользователя об опасности взрыва. Помимо измерения концентрации метана в помещении проектируемый прибор будет измерять температуру и относительную влажность воздуха, а также концентрацию углекислого газа.

Для реализации заданных функций устройство должно включать в свой состав следующие компоненты:

– микроконтроллер;

– датчик метана(пропана);

– датчик температуры;

– датчик влажности;

– датчик углекислого газа;

– жидкокристаллический дисплей;

– спикер;

– светодиоды;

– кнопки.

**1.2 Постановка задачи**

**2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

**2.1 Модули приложения**

**2.1.1 Модуль чтения данных**

Модуль чтения данных выполняет сбор информации из системного каталога «/proc», в котором находится полная информация о всех запущенных процессах, а также состоянии системы в целом.

**2.1.2 Модуль преобразования данных**

Модуль преобразования данных выполняет перевод данных, полученных в модуле чтения данных, в удобный для работы с «Qt» формат, а также помещает их оперативное хранилище.

**2.1.3 Модуль оперативного хранения данных**

Модуль оперативного хранения данных необходим для организации временного хранения информации, полученной в модуле преобразования данных.

**2.1.4 Модуль пользовательского интерфейса**

Модуль пользовательского интерфейса реализует представление информации в удобном для человека формате.

Данный модуль реализован в программе при использовании фреймворка «Qt».

**2.1.5 Модуль работы приложения**

Этот модуль необходим для реализации взаимодействия модуля пользовательского интерфейса с другими модулями программы. Примером может быть нажатие клавиши и ответ в виде изменения интерфейса.

**2.2 Структурная схема**

Структурная схема представлена в **ПРИЛОЖЕНИИ А**

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

**3.1 Структура данных myTime**

Структура данных myTime содержит информацию о времени, представленном в часах, минутах и секундах:

- int hours – часы;

- int minutes – минуты;

- int seconds – секунды.

**3.2 Структура данных taskData**

Структура данных taskData содержит самую важную информацию о процессе:

- char name[MAX\_NAME\_LEN] – имя процесса;

- char state[2] – состояние процесса спящий, зомби и т.д;

- pid\_t pid – пид процесса;

- double memory – занимаемая процессом резидентная память в процентах;

- double loadCpu – отношение выделенного на задачу времени к общему процессорному времени, представленное в процентах.

**3.3 Структура данных taskScheduledTime**

Структура данных taskScheduledTime хранит значения времени, измеряемом в тактах процессора, отведённого на выполнение задачи:

- unsigned long long int utime – время, затраченное на выполнение пользовательского кода процессом;

- unsigned long long int stime – время, затраченное на выполнение системного кода процессом;

- unsigned long long int cutime – время, затраченное на выполнение пользовательского кода процессов-потомков процесса;

- unsigned long long int cstime – время, затраченное на выполнение системного кода процессов-потомков процесса.

**3.4 Структура данных commonCpuTime**

Структура данных commonCpuTime хранит значения общего процессорного времени, измеряемом в тактах процессора:

- unsigned long long int user – время, затраченное на выполнение пользовательского кода процессором;

- unsigned long long int nice – время, затраченное на выполнение пользовательского кода с низким приоритетом процессором;

- unsigned long long int system – время, затраченное на выполнение системного кода процессором;

- unsigned long long int idle – время, когда процессор не занят выполнением каких-либо задач и находится в режиме ожидания.

**3.5 Структура данных sortType**

Структура данных sortType хранит информацию о способе сортировки массива данных о процессе, если значение какого-либо поля единица, то сортировка будет осуществляться по этому параметру:

- int byPid – флаг сортировки по пиду;

- int byname – флаг сортировки по имени;

- int byCpuLoad – флаг сортировки по загрузке процессора;

- int byMemoryLoad – флаг сортировки по проценту используемой памяти.

**3.6 Структура данных searchType**

Структура данных searchType хранит информацию о способе поиска процесса, если значение какого-либо поля единица, то поиск будет осуществляться по этому параметру:

- int byname – поиск процесса по имени;

- int byPid – поиск процесса по пиду.

**4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

* 1. **Алгоритм по шагам функции fillMemory**

1) начало.

2) входные данные:

   - char\* pathToTask – массив строк с путями на каталоги, содержащие информацию о процессах;

   - struct\* taskData task – массив структур с информацией о процессах;

   - int taskNumber – количество активных процессов;

   - char\* pathStatm – путь к файлу с информацией о используемой текущим процессом памяти;

   - long long unsigned int vmRss – объем используемой резидентной памяти;

   - long long unsigned int totalMemory – объем оперативной памяти на устройстве;

   - long long unsigned int freeMemory – объем незанятой оперативной памяти;

   - FILE\* fileStream – поток файла.

   выходные данные:

   - struct\* taskData task – массив структур с информацией о процессах.

3) fileStream = fopen(MEMINFO\_PATH,“r”) – открыть файл с информацией о памяти устройства в режиме чтения.

4) если fileStream == NULL, то перейти к шагу 15, иначе перейти к шагу 5.

5) fscanf(fileStreamMeminfo,"%\*s%llu%\*s%\*s%llu",&totalMemory &freeMemory) - читать из файла информацию о состоянии оперативной памяти устройства.

6) fclose(fileStream) - закрыть файл.

7) начало цикла по i, пока i < taskNumber.

8) strcat(pathStatm,pahtToTask[i]);

        strcat(pathStatm,"/statm") – сформировать путь к файлу с информацией о используемой памяти конкретным процессом.

9) fileStream = fopen(pathStatm,”r”) – открыть файл.

10) если fileStream == NULL, то перейти к следующей итерации, иначе перейти к шагу 11.

11) fscanf(fileStreamStatm,"%\*d%llu",&vmRss) - читать данные о используемой процессом резидентной памяти.

12) fclose(fileStream) - закрыть файл.

13) task[i].memory = (vmRss/(totalMemory-freeMemory))\*100 -рассчитать занимаемую резидентную память конкретным процессом и записать результат вычислений в массив task.

14) конец цикла по i.

15) конец.

**4.2 Алгоритм по шагам функции getHostName**

1) начало.

2) входные данные:

   - FILE\* fileStream – поток файла;

   - int good– хранит количество удачно чтений из файла.

   выходные данные:

   - char\* hostName– хранит имя хоста, считанное из файла.

3) fileStream = fopen(HOSTNAME\_PATH,“r”) – открыть файл с информацией о хосте.

4) если fileStream == NULL, то перейти к шагу 8, иначе перейти к шагу 5.

5) good = fscanf(fileStream,”%s”,hostName) – читать из файла имя хоста.

6) если good != 1, то перейти к шагу 7, иначе перейти к шагу 8.

7) strcpy(hostname,”read error”) – записать сообщение о том, что не удалось считать.

8) конец.

**4.3 Схема алгоритма getTaskList**

Схема алгоритма функции getTaskList представлена в **ПРИЛОЖЕНИИ** **Б**

**4.4 Схема алгоритма getUptime**

Схема алгоритма функции getUptime  представлена в **ПРИЛОЖЕНИИ** **В**

**5 РЕЗУЛЬТАТЫ**

**5.1 Тестирование**

После запуска диспетчера задач под Linux на экране появиться окно приложения. В данном окне будет отображаться производительность системы, представленная в виде загрузки оперативной памяти и общего процента загруженности процессора. Также будут выведены константные данные о устройстве и системе. Изображение окна представлено на рисунке 5.1.

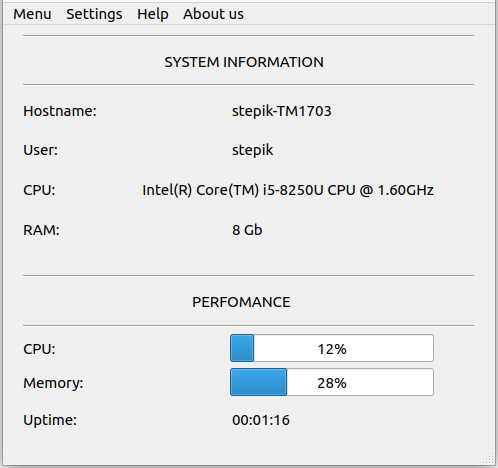


Рисунок 5.1 — Интерфейс приложения в режиме производительность

Для перехода в режим просмотра активных задач требуется перейти во вкладку «menu» и выбрать пункт «tasks». После выполнения этих действий приложение переходит в режим вывода информацию о процессах. Режим «задачи» представлен на рисунке 5.2.

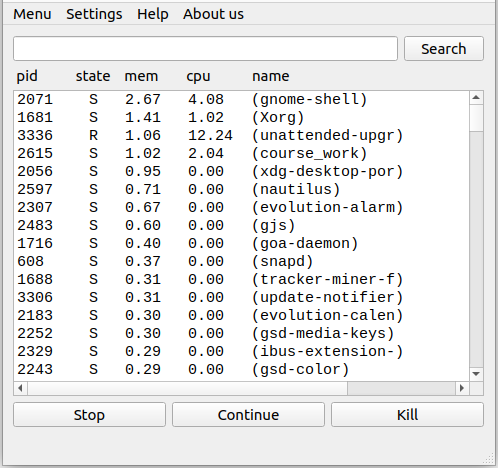


Рисунок 5.2 — Интерфейс приложения в режиме задачи

В верхней части окна расположено поисковая строка. Режимы поиска можно поменять, перейдя во вкладку «settings» и выбрав пункт «search by».

Под поисковой строкой будет расположен список процессов. Данный список имеет несколько колонок: пид, состояние, загрузка памяти, загрузка процессора и имя процесса. Для изменения последовательности вывода списка требуется прейти во вкладку «settings», а затем выбрать пункт «sort by».

В нижней части окна расположено три кнопки. Кнопка «stop» приостанавливает выполнение процесса. Кнопка «continue» продолжает выполнение ранее приостановленного процесса. И кнопка «kill» завершает процесс.

Под кнопками находится панель «статус бар», на которою выводится отчет о выполнение команд. Пример вывода информации на панель представлен на рисунках 5.3, 5.4.

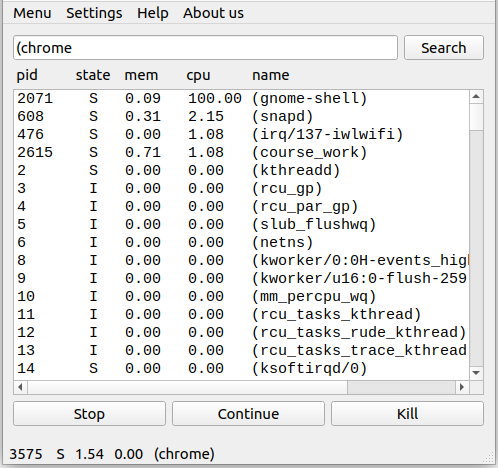


Рисунок 5.3 — Использование панели «статус бар» при поиске

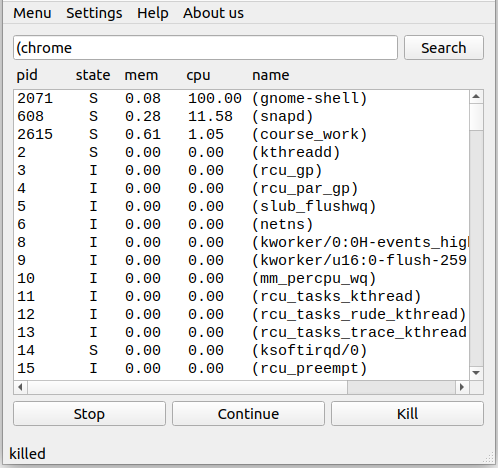


Рисунок 5.4 — Использование панели «статус бар» при завершении процесса

**5.2 Программные и системные ограничения**

Системные ограничения:

- Программа скомпилирована компиляторами gcc и g++. Сборка проекта осуществляется с помощью qmake. Операционная система - Linux.

Программные ограничения:

- не все процессы возможно остановить, продолжить или завершить;

- нет возможности увеличения окна приложения;

- отсутствие русификации приложения;

- при считывании большого пида процесса возможно сползание таблицы.

**5.3 Код программы**

Код программы представлен в **ПРИЛОЖЕНИИ Г**

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта была написано приложение «Диспетчер задач под Linux». Данное приложение выполняет основные функции по управлению процессами и мониторингу системных параметров, а также имеет удобный пользовательский интерфейс.

При написание курсового проекта были освоены основные системные функции такие как opendir(), readdir(), fopen(), fclose() и т. д. Также для реализации удобного пользовательского интерфейса был изучен фреймворк «Qt». Были получены новые знания о операционной системе Linux.

Написанное приложение является аналогом приложений диспетчер задач Windows и утилиты htop.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

[1] Linux. Системное программирование. 2-е изд. - Спб.: Питер, 2014 - 448с.

[2] Вычислительные машины, системы и сети: дипломное проектирование (методическое пособие) [Электронный ресурс]. - Минск, БГУИР 2019.

[3] Лафоре, Р. Объектно-ориентированное программирование в C++, 4-е издание / Р. Лафоре - СПб.: Питер, 2004.

[4] Документация Qt [Электронный ресурс]. - The Qt Company Ltd. 2019.  - Режим доступа: <https://doc.qt.io>.

[5] Документация Си [Электронный ресурс]. – OpenNET - Режим доступа: https://opennet.ru.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Схема структурная**

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Схема алгоритма функции getTaskList**

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Схема алгоритма функции getUptime**

**ПРИЛОЖЕНИЕ Г**

**(обязательное)**

**Код программы**