Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«Диспетчер задач для Linux»

БГУИР КP 1-40 02 01 14\* ПЗ

Студент: гр. 550502 Кессо П.И.

Руководитель: ассистент кафедры ЭВМ Лавникевич Д.А.

Минск, 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5

1 ПОСТАНОВКА ЗАДАЧИ 6

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА

СПЕЦИФИКАЦИЙ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ………… 7

2.1 Обзор cуществующих аналогов 7

2.2 Разработка спецификации функциональных требований 11

3 ОБОСНОВАНИЕ ТЕХНИЧЕСКИХ ПРИЁМОВ

ПРОГРАММИРОВАНИЯ…………………………………………………… 12

3.1 Выбор языка 12

3.2 Детали реализации 13

4 ОПИСАНИЕ ОСНОВНЫХ АЛГОРИТМОВ 19

4.1 update 19

4.2 updateProcess 19

5 КОД ПРОГРАММЫ 20

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ И РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 23

ЗАКЛЮЧЕНИЕ 25

СПИСОК ЛИТЕРАТУРЫ 26

**ВВЕДЕНИЕ**

Диспетчер задач – компьютерная программа для вывода на экран списка запущенных процессов и потребляемых ими ресурсов. В качестве дополнительных функций, диспетчер задач может предложить возможность завершить один из процессов или наблюдать загруженность процессора, оперативной памяти, и других ресурсов.

Программы имеют свойство зависать – от этого никуда не денешься, даже в Linux. Поэтому один из вопросов, через который обязательно проходит начинающий пользователь Linux: где диспетчер задач? Как завершить процесс?

Для того что бы завершить процесс в Linux необходимо знать PID процесса.

PID (англ. process identifier) – уникальный номер (идентификатор) процесса в многозадачной операционной системе.

В большинстве случаем именно для того чтобы узнать PID процесса, или непосредственно для завершения процесса и используются диспетчеры задач.

Для принудительного завершения работы приложения в Linux, если такую функцию не предусматривает сам диспетчер задач, используется команда kill. Когда вы выполняете команду kill, то фактически вы посылаете системе сигнал, чтобы заставить её завершить некорректно ведущее себя приложение. Для завершения процесса можно использовать один из следующих сигналов: SIGTERM (запрашивает остановку работы процесса) или SIGKILL (заставляет процесс прекратить работы немедленно).

**1 ПОСТАНОВКА ЗАДАЧИ**

Пользователь программы должен иметь возможность просматривать список процессов, запущенных на компьютере, просматривать информацию о каждом из процессов, просматривать информацию о ресурсах компьютера. Необходимо реализовать возможность завершать процессы.

Программа должна иметь удобный пользовательский интерфейс, позволяющий в полной мере использовать весь функционал программы. Пользователь должен иметь возможность поиска процессов из списка, предоставленного программой.

# 

# МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И

# РАЗРАБОТКА СПЕЦИФИКАЦИЙ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

**2. 1 Обзор существующих аналогов**

На сегодняшний день пользователи Linux имеют большой выбор диспетчеров задач. Существующие диспетчеры задач можно разделить на два типа: имеющие графический интерфейс и не имеющие графический интерфейс.

Самой и распространенной и удобной утилитой не имеющей графического интерфейса является top (см. рисунок 2.1). Top предоставляет список работающих в системе процессов и информации о них, информацию об использовании процессора, оперативной памяти.

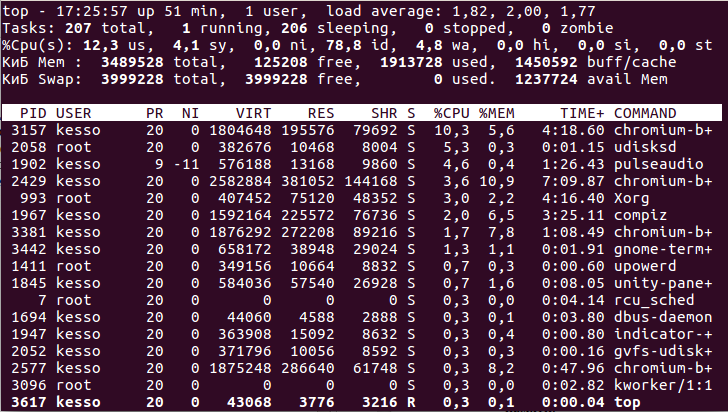


Рисунок 2.1 – программа top

Простота top, большое количество возможностей, привели к широкому распространению программы, в частности программа часто используется системными администраторами. Явным недостатком, для большинства начинающих пользователей Linux, является отсутствие графического интерфейса в данной программе, а также то, что top показывает не все процессы в системе. Можно подвести итог: top полезный, но не слишком удобный.

По имени top суффикс “-top” получило большое количество подобных программ. В качестве примера таких программ можно привести htop (см. рисунок 2.2) или atop (см. рисунок 2.3).

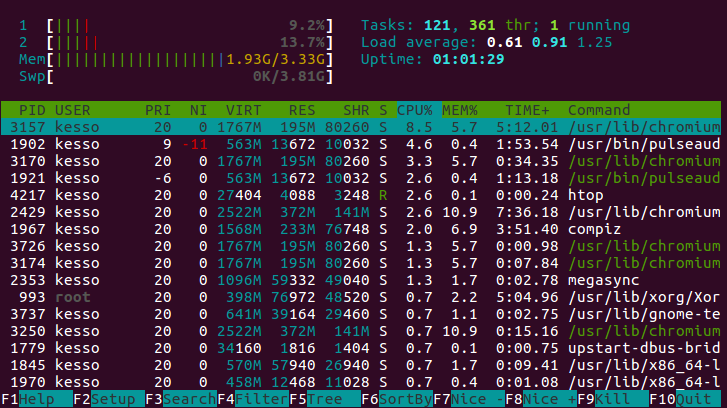


Рисунок 2.2 – программа htop

Htop — продвинутый монитор процессов, написанный для Linux. Он был задуман заменить стандартную программу top. Htop показывает динамический список системных процессов, список обычно выравнивается по использованию ЦПУ. В отличие от top, htop показывает все процессы в системе. Также показывает время непрерывной работы, использование процессоров и памяти. Htop часто применяется в тех случаях, когда информации даваемой утилитой top недостаточно, например, при поиске утечек памяти в процессах. Htop написан на языке Си и использует для отображения библиотеку Ncurses. Хотя работает программа в консоли, интерфейс все-таки вполне можно назвать если не графическим, то псевдографическим точно.

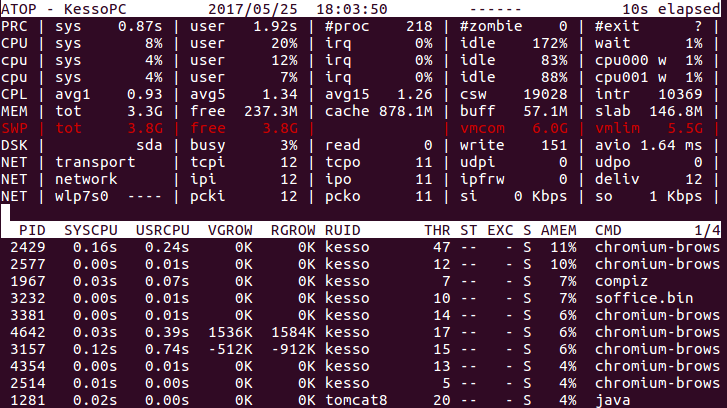
Atop имеет два режима работы — сбор статистики и наблюдение за системой в реальном времени. В режиме сбора статистики atop запускается как демон и раз в N времени (обычно 10 мин) скидывает состояние в двоичный журнал. Потом по этому журналу atop'ом же (ключ -r и имя лог-файла) можно бегать вперёд-назад кнопками T и t, наблюдая показания atop'а с усреднением за 10 минут в любой интересный момент времени. 

Рисунок 2.3 – программа atop

В отличие от top отлично знает про существование блочных устройств и сетевых интерфейса, способен показывать их загрузку в процентах.   
 Atop - незаменимое средство для поиска источников лагов на сервере, так как сохраняет не только статистику загрузки системы, но и показатели каждого процесса — то есть «долистав» до нужного момента времени можно увидеть, кто этот счастливый момент с LA> 30 создал. И что именно было причиной — IO программ, своп (нехватка памяти), процессор или что-то ещё. Помимо большего количества информации ещё способен двумя цветами подсказывать, какие параметры выходят за разумные пределы.

Наличие графического решения, по-умолчанию, для просмотра и завершения процессов в Linux зависит от Вашего дистрибутива.

В качестве примера диспетчера задач с графическим интерфейсом можно привести Gnome System Monitor. Gnome System Monitor – программа для мониторинга работы системы. Является штатным системным мониторам в рабочей среде Gnome. Интерфейс программы довольно простой. Вся информация распределена по трем вкладкам.

В первой вкладке «Процессы» (см. рисунок 2.4) можно просматривать и управлять, работающими процессами. Каждый процесс можно приостановить, остановить, изменить приоритет и выполнить некоторые другие действия.

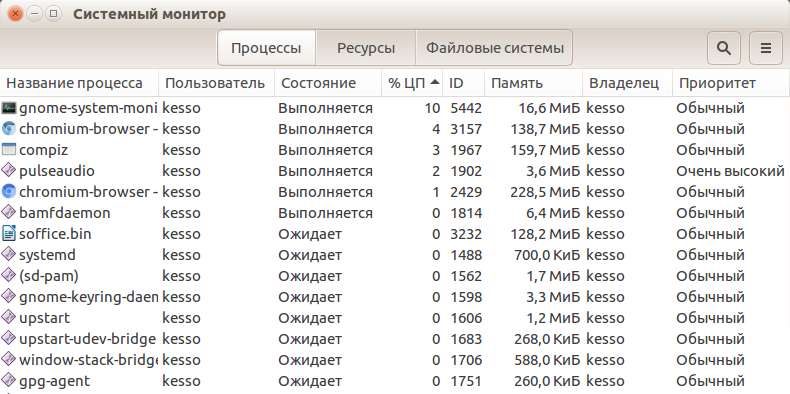


Рисунок 2.4 – программа Gnome System Monitor

В последней вкладке можно просматривать информацию о файловых системах (см. рисунок 2.5).

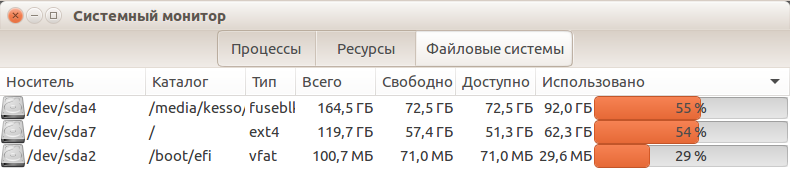


Рисунок 2.5 – программа Gnome System Monitor

Наиболее интересно выглядит вкладка «Ресурсы» (см. рисунок 2.6). В ней утилита выводит в виде графиков информацию в реальном времени о ресурсах — использование процессора (CPU), использование оперативной памяти (RAM) и файла подкачки (SWAP), а также использование сети.

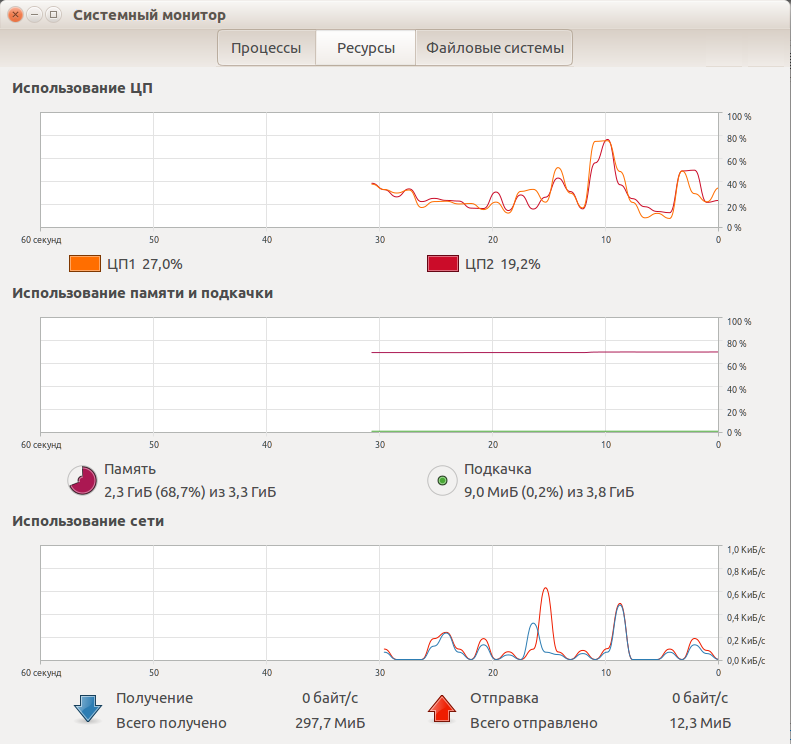


Рисунок 2.6 – программа Gnome System Monitor

**2. 2 Разработка спецификации функциональных требований**

После ознакомления существующих аналогов, был выделен ряд основных требований, которые должны быть выполнены при реализации курсовой работы.

Пользователь программы должен иметь возможность просматривать список процессов, запущенных на компьютере, просматривать информацию о каждом из процессов, просматривать информацию о ресурсах компьютера. Необходимо реализовать возможность завершать процессы.

Программа должна иметь удобный пользовательский интерфейс, позволяющий в полной мере использовать весь функционал программы. Пользователь должен иметь возможность поиска процессов из списка, предоставленного программой

1. **ОБОСНОВАНИЕ ТЕХНИЧЕСКИХ ПРИЁМОВ**

**ПРОГРАММИРОВАНИЯ**

**3.1 Выбор языка**

Си — это универсальный язык программирования с компактным способом записи выражений, современными механизмами управления структурами данных и богатым набором операторов. Си не является ни языком "очень высокого уровня", ни "большим" языком, не рассчитан он и на какую-то конкретную область применения. Однако благодаря широким возможностям и универсальности для решения многих задач он удобнее и эффективнее, чем предположительно более мощные языки. Первоначально Си был создан Деннисом Ритчи как инструмент написания операционной системы UNIX для машины PDP-11 и реализован в рамках этой операционной системы. И операционная система, и Си- компилятор, и, по существу, все прикладные программы системы UNIX написаны на Си. Си не привязан к конкретной аппаратуре или системе, однако на нем легко писать программы, которые без каких- либо изменений переносятся на другие машины, где осуществляется его поддержка.

В Си имеются основные управляющие конструкции, используемые в хорошо структурированных программах: составная инструкция ({...}), ветвление по условию (if-else), выбор одной альтернативы из многих (switch), циклы с проверкой наверху (while, for) и с проверкой внизу (do), а также средство прерывания цикла (break). В качестве результата функции могут возвращать значения базовых типов, структур, объединений и указателей. Любая функция допускает рекурсивное обращение к себе. Как правило, локальные переменные функции — "автоматические", т. е. они создаются заново при каждом обращении к ней. Определения функций нельзя вкладывать друг в друга, но объявления переменных разрешается строить в блочно- структурной манере. Функции программы на Си могут храниться в отдельных исходных файлах и компилироваться независимо. Переменные по отношению к функции могут быть внутренними и внешними. Последние могут быть доступными в пределах одного исходного файла или всей программы.

Не смотря на все свои достоинства Си не способен предоставить полное решение для поставленной задачи, а именно реализовать удобный пользовательский графический интерфейс. Для реализации этих задач, отлично подойтёт C++, а именно кроссплатформенная библиотека разработки GUI на С++ - Qt.

Со времени своего появления в 1996 году библиотека Qt легла в основу тысяч успешных проектов во всём мире. Кроме того, Qt является фундаментом популярной рабочей среды KDE, входящей в состав многих дистрибутивов Linux.

Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC)— предварительной системы обработки исходного кода. MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным.

* 1. **Детали реализации**

Для реализации поставленных задач, необходимо получение актуальной информации о всех запущенных процессах в системе, подробной информации о каждом из процессов. Также необходима получение информации о использовании ресурсов компьютера.

Всю необходимую информацию можно получить из псевдо-файловой системы /proc.

Файловая система /proc является особой для GNU/Linux. Это виртуальная файловая система и она не занимает места на вашем диске. Изначально файловая система /proc разрабатывалась как средство предоставления информации о выполняющихся в системе процессах. Это очень удобный способ получить информацию о системе, тем более, что большинство файлов в этом каталоге читаемы для человека. Большинство программ реально получают информацию из файлов в /proc, форматируют их своим способом, а затем отображают. Так делают все программы, которые отображают информацию о процессах. /proc это также хороший источник информации о ваших аппаратных средствах, и таким же образом некоторые программы являются только интерфейсами к информации, содержащейся в /proc.

Существует также специальный подкаталог /proc/sys. Он позволяет изменять некоторые параметры ядра в реальном режиме времени или отображать их.

Для начала необходимо получить информацию о процессах, запущенных в системе.

Если вы просмотрите содержимое каталога /proc, вы увидите много каталогов, названиями которых являются номера (см. рисунок 3.1).

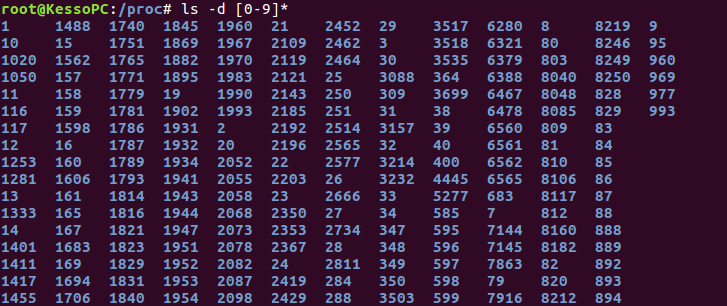


Рисунок 3.1 – содержимое директории /proc

Название каждого каталога обозначает PID процесса, а содержимое сам каталог содержит информацию о каждом запущенном в системе процессе. Когда процесс заканчивается, его каталог исчезает из системы. Если открыть любой из этих каталогов в нем будут содержаться следующие файлы (см. рисунок 3.2).

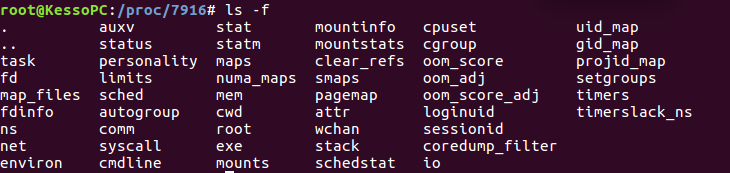


Рисунок 3.2 – содержимое директории /proc/[PID]

Рассмотрим основные файлы из этой директории:

cmdline — содержит команду с помощью которой был запущен процесс, а также переданные параметры.

cwd — символическая ссылка на текущую рабочую директорию.

exe — ссылка на исполняемый файл.

root — ссылка на папку суперпользователя.

maps, statm, mem — информация о памяти процесса.

status, stat — состояние процессам.

Для реализации большинства поставленных задач, связанных с процессами, нам будет достаточно двух последних файлов.

Подробнее рассмотрим каждый из файлов:

/proc/[PID]/stat — файл, содержащий информацию о состоянии процесса. Содержимое файла представлено (см. рисунок 3.3).

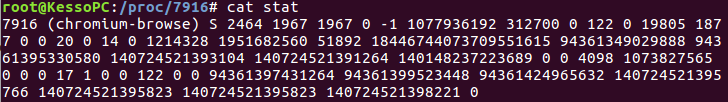


Рисунок 3.3 – содержимое файла stat

Подробную расшифровку файла можно получить при помощи команды man. Мы же рассмотрим только интересующие нас содержимое файла.

(1) pid — pid процесса

(2) comm — имя исполняемого файла в скобках

(3) state — один символ, обозначающий состояние процесса

(4) ppid — pid родительского процессам

(14) utime — время CPU, которое этот процесс затратил в user mode

(15) stime — время CPU, которое этот процесс затратил в kernel mode

(18) priority — число, соответствующее приоритету процессам

(23) vsize — размер виртуальной памяти в байтах

/proc/[PID]/status — предоставляет большую часть информации из /proc[PID]/stat но в более удобной для чтения человеком. Главное отличие файла status от файла stat, это наличие информации о UID (идентификатора пользователя) в файле status. Эта информация позволяет узнать имя пользователя, запустившего процесс. Именно из-за этой информации, в программе будет использованы оба файла.

Для получения информации о системе необходимо рассмотреть файлы находящиеся в корне папки proc, в них больше всего информации о системе.

Вся информация необходимая приложению находится в трех файлах: /proc/meminfo, /proc/cpuinfo, /proc/stat.

/proc/meminfo — в этом файле отображается статистика использования памяти в системе (см. рисунок 3.4). Каждая строка файла содержит имя параметра, значение параметра и единицу измерения.

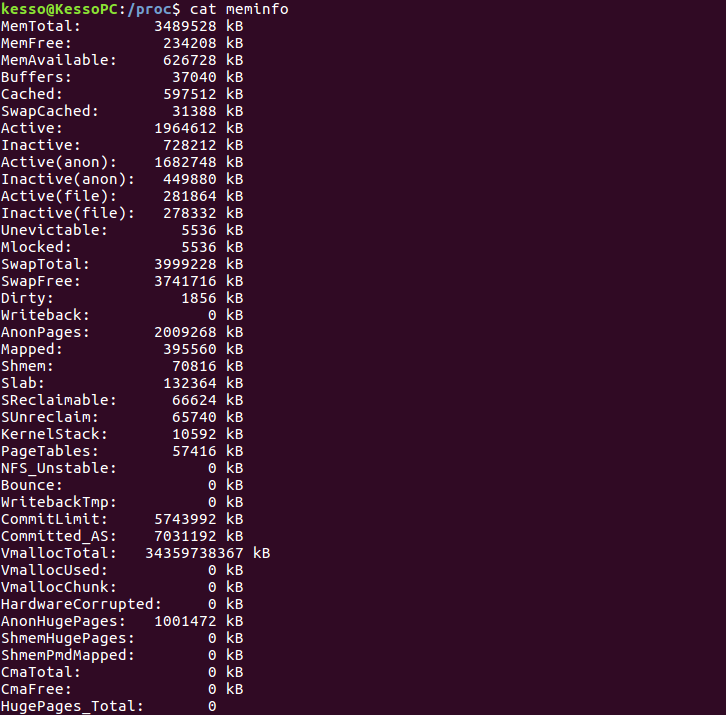


Рисунок 3.4 – содержимое файла meminfo

Нас интересуют две первые строки файла:

MemTotal — доступный объём оперативной памяти (физическое ОЗУ без учёта нескольких зарезервированных битов и двоичного кода ядра)

MemFree — объём оперативной памяти доступный для немедленного выделения процессам.

/proc/cpuinfo — содержит информацию, зависящую от процессора и системной архитектуры.

/proc/stat — В этом файле содержится статистическая информация о системе. Такая как частота процессора, количество тактов, количество системных прерываний, время загрузки (см. рисунок 3.5).

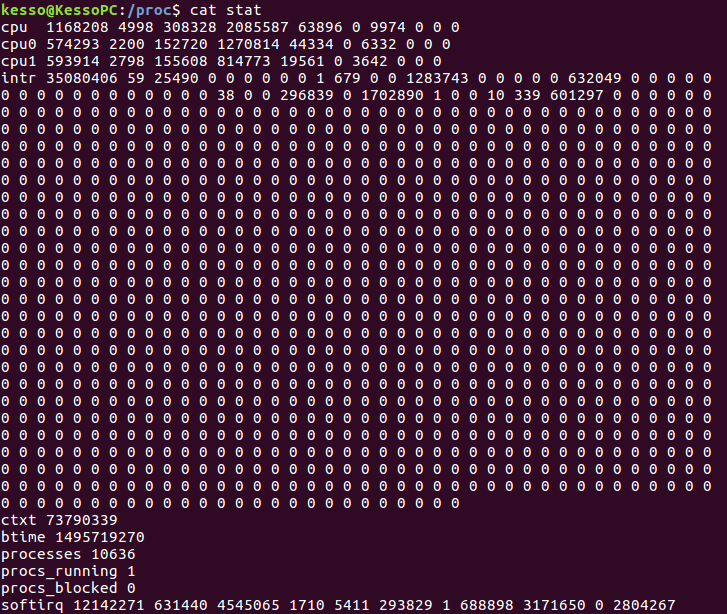


Рисунок 3.5 – содержимое файла stat

Нас интересуют первые строки файла, из них можно узнать количество времени, которое CPU тратил на выполнение различных задач. Значение колонок по очереди:

user — обычные процессы, которые выполняются в user mode.

Nice — процессы с nice в user mode.

system — процессы в kernel mode.

idle — время в простое.

iowait — ожидание операций i/o.

irq — обраотка прерываний.

softirq — обработка прерываний от периферийных устройств.

steal — время, потраченное другими операционными системати при использовании виртуализации.

guest — обработка виртуальных процессов.

Сложив все эти данные, мы получи общее время работы процессора, включая время бездействия.

Файл содержит как информация для процессора в целом, так и для каждого ядра по отдельности.

**4 ОПИСАНИЕ ОСНОВНЫХ АЛГОРИТМОВ**

**4.1 update**

Алгоритм update является алгоритмом обновления информации о процессе. Основная проблема, возникшая при разработке данного алгоритма, то что для подсчёта загрузки процессом CPU в % используется формула: cpu = (procTime1 - procTime0) / (coreTime1 – coreTime0) \* 100, где время замера procTime0 и procTime1, а также coreTime0 и coreTime1, должно различаться на промежуток времени от 1 до 5 секунд. Для решения этой проблемы при подсчёте использованного процессом CPU за procTime0 и cpuTime0 берутся значения, полученные при предыдущем вызове функции, а за procTime1 и cpuTime0 - значения, полученные за время текущего выполнения функции.  
Схема работы алгоритма update представлена на рисунке 4.1.

**4.2 updateProcess**

Алгоритм updateProcess является алгоритмом обновления списка процессов и обновления информации об отдельных процессах. Данный алгоритм позволяет использовать информацию о процессах, которая не могла измениться, полученную ранее. Эта информация включает в себя название процесса, его PID, родительский PID, владельца процесса. Схема работы алгоритма updateProcess представлена на рисунке 4.2.

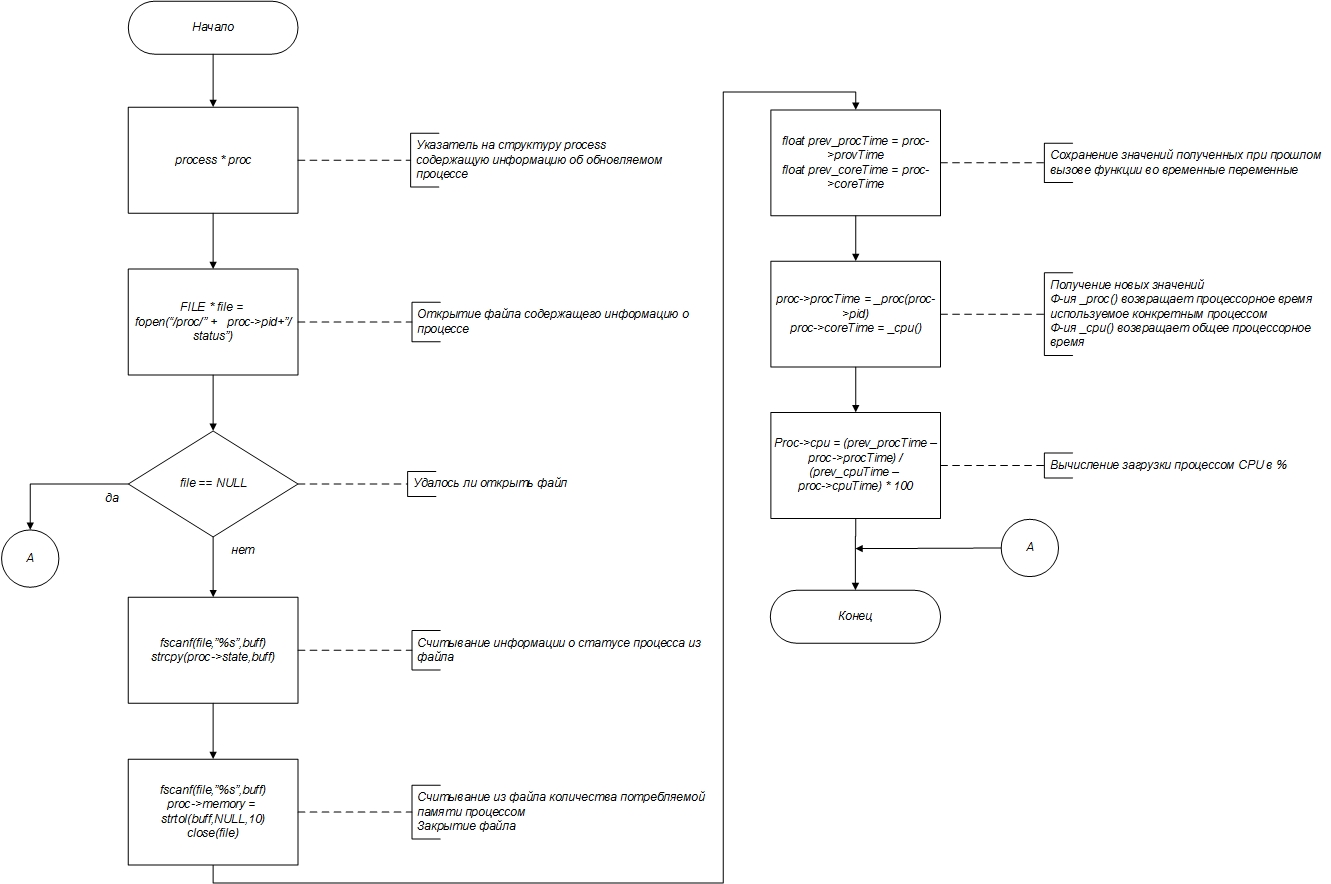


Рисунок 4.1 – схема алгоритма update

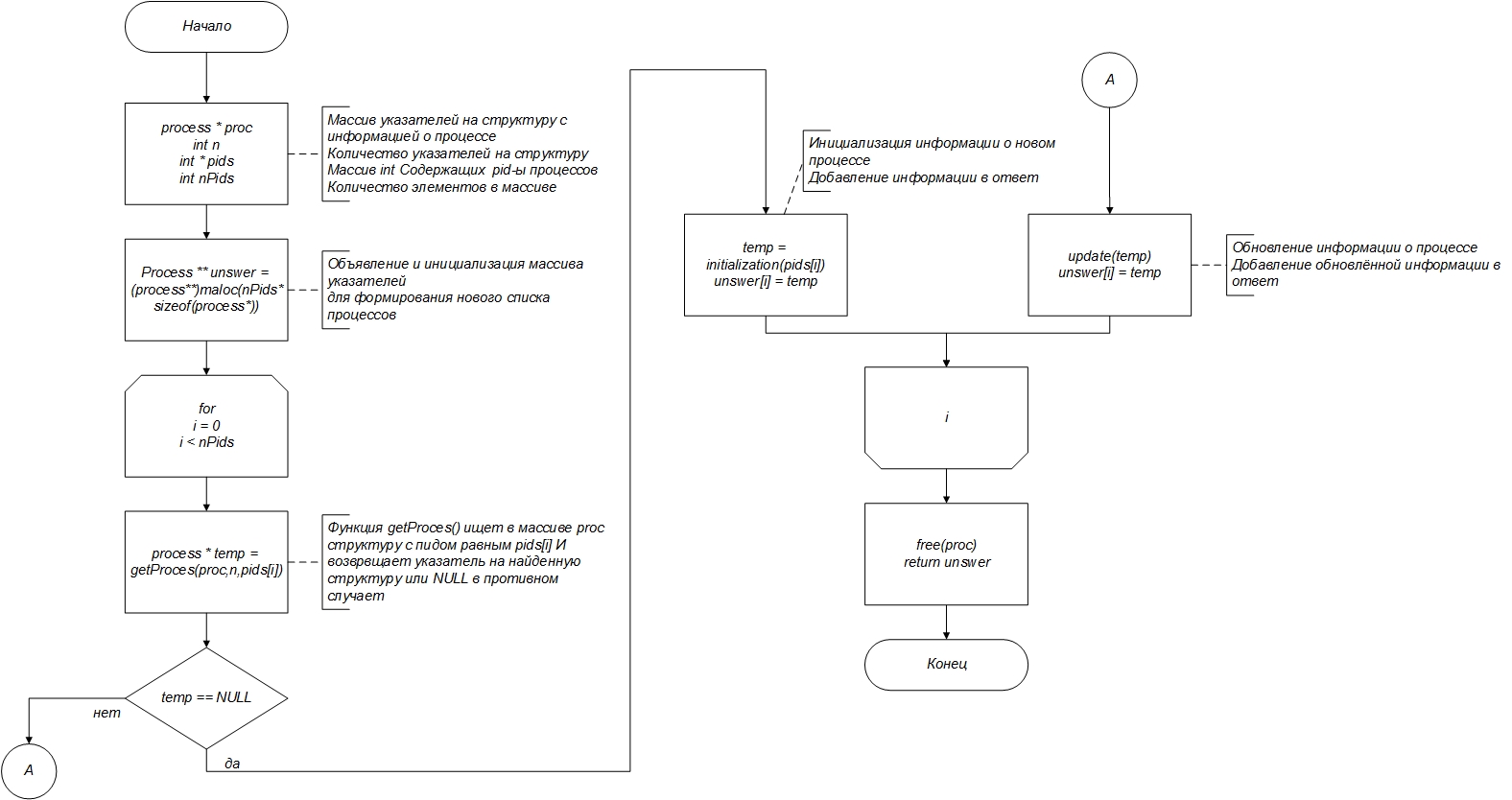


Рисунок 4.2 – схема алгоритма updateProcess

**5 КОД ПРОГРАММЫ**

https://github.com/KessoPavel/course\_project

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ И РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

Для использования программы требуется произвести её установку.

Для работы с программой не требуется никаких специальных навыков. Сразу после открытия программы пользователь имеет возможность воспользоваться всеми функциями программы (см. рисунок 6.1).

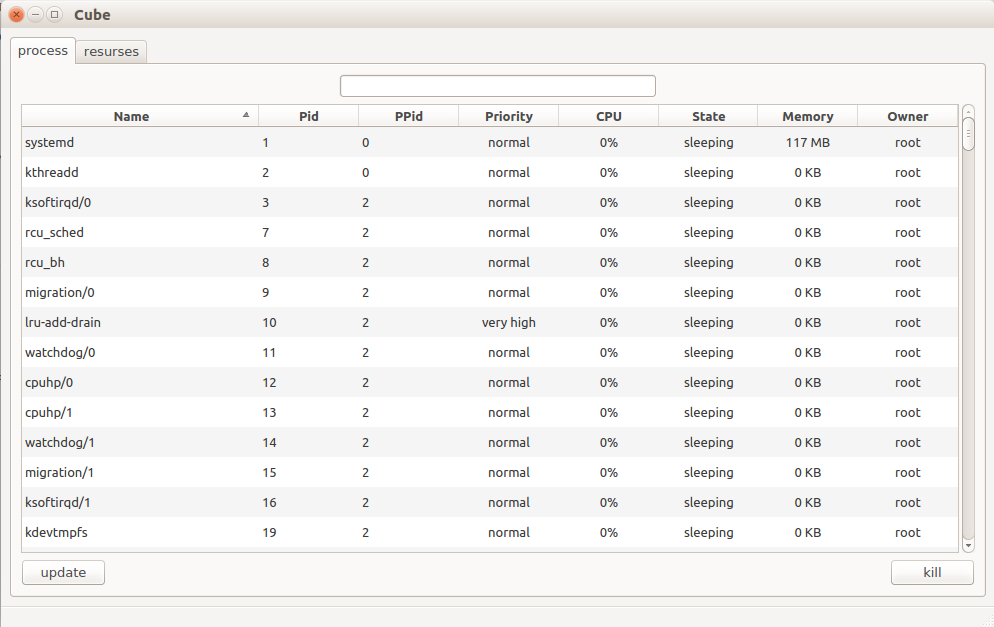


Рисунок 6.1 – программа Сube

Окно программы содержит две вкладки. В первой содержится таблица с информацией о процессах, запущенных в системе. Пользователь имеет возможность активно взаимодействовать с таблицей. Для поиска процесса по имени, надо начать вводить имя процесса в строку ввода, которая располагается выше таблицы, поиск проводится без учёта регистра.

Для того что бы завершить какой-либо процесс, необходимо навести курсор на строку в таблице с нужным процессом и нажать левую клавишу мыши. После выбора процесса необходимо нажать кнопку “kill” в правом нижнем углу.

Вторая вкладка содержит информацию о используемых ресурсах компьютера (см. рисунок 6.2). В этой вкладке содержится информация о загрузке процессора (в целом и для каждого ядра в отдельности), а также информация об оперативной памяти компьютера.

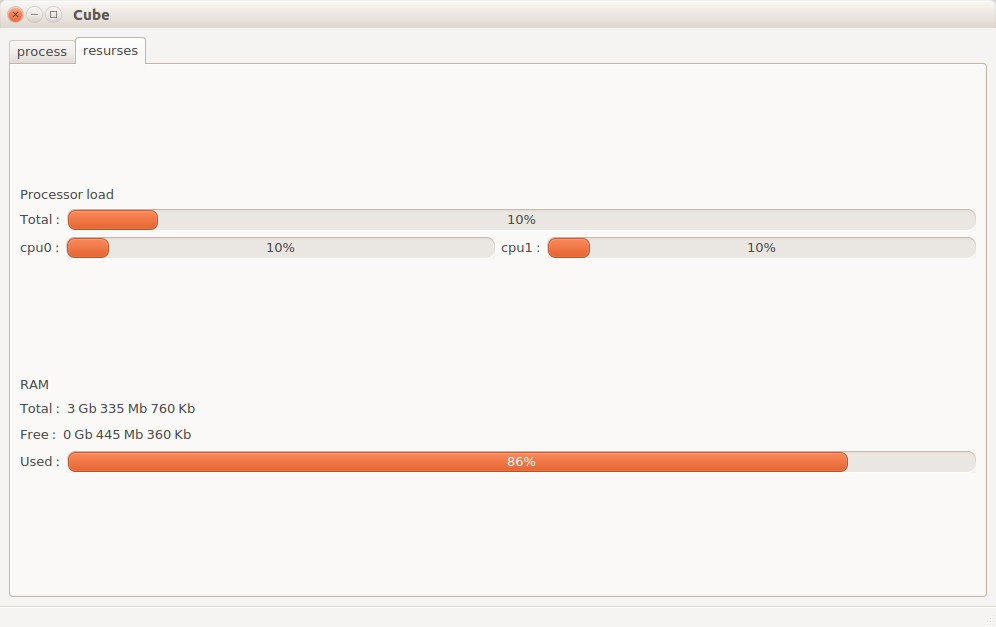


Рисунок 6.2 – программа Cube

**ЗАКЛЮЧЕНИЕ**

Данный курсовой проект является результатом процесса, включающего анализ поставленной задачи, исследование существующих аналогов, работу с литературой, проектирование структур данных, разработку алгоритмов, тестирование созданного приложения, составление руководства пользователя.

Приложение предоставляет пользователю все требуемые согласно поставленной задаче возможности. В результате выполнения курсового проекта была разработана программа, позволяющая просматривать, искать и завершать процессы.

К достоинствам программы можно отнести простой и понятный интерфейс, что в свою очередь обеспечивает удобство эксплуатации для обычных пользователей.

**СПИСОК ЛИТЕРАТУРЫ**

[1] Лав, Р. Linux. Системное программирование: [изучаем ядро Linux и библиотеки C] / Роберт Лав; [перевод с английского О. Сивченко]. - Санкт-Петербург [и др.] : Питер : Питер Пресс, 2016. - 445 с.

[2] Таненбаум Э.С. Современные операционные системы / Э. Таненбаум, Х. Бос; [перевели с английского А. Леонтьева, М. Малышева, Н. Вильчинский]. - Санкт-Петербург [и др.] : Питер : Питер Пресс, 2017. - 1119 с. - (Классика computer science).

[3] Мэтью, Н. Основы программирования в Linux : [перевод с английского] / Нейл Мэтью, Ричард Стоунс. - Санкт-Петербург : БХВ-Петербург, 2009. - XIV, 882 с.