**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ДЗЕРЖИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)**

**Кафедра АВТОМАТИЗАЦИЯ И ИНФОРМАЦИОННЫЕ СИСТЕМЫ**

**РЕФЕРАТ**

**По дисциплине «Операционные системы»**

**«KolibriOS»**

РУКОВОДИТЕЛЬ

Все вам скажи

СТУДЕНТ

Ещё один анонимус

10-ИСТ

Дзержинск

2012 г.

Оглавление

[Введение 3](#_Toc325583425)

[Общие сведения о KolibriOS 4](#_Toc325583426)

[Краткое описание и особенности 4](#_Toc325583427)

[История КолибриОС 5](#_Toc325583428)

[Перспективы использования и развития КолибриОС 7](#_Toc325583429)

[Структура KolibriOS 8](#_Toc325583430)

[Управление процессами. 8](#_Toc325583431)

[Управление вводом-выводом: аппаратура с точки зрения ядра 9](#_Toc325583432)

[Управление вводом-выводом: аппаратура с точки зрения приложений 11](#_Toc325583433)

[Файловая система 13](#_Toc325583434)

[Управление памятью 14](#_Toc325583435)

[Интерфейс прикладного программирования и интерфейс пользователя 17](#_Toc325583436)

[Заключение 18](#_Toc325583437)

Введение

Операционная система, сокр. ОС (англ. operating system, OS) —комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы, прикладными программами и пользователем, а с другой стороны — предназначены для управления аппаратурой, вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надёжных вычислений. Это определение применимо к большинству современных операционных систем. Одной из них является KolibriOS, о ней и будет рассказано в данном реферате.

Общие сведения о KolibriOS

Краткое описание и особенности

KolibriOS (КолибриОС) — операционная система, полностью написанная на языке flat assembler (свободно распространяемом многопроходном ассемблере, написанном на самом себе), распространяемая на условиях GPL (универсальной общедоступной лицензии — пользователь получает свободу исследования и редактирования исходного кода программы, но автор сохраняет за собой авторство). КолибриОС создана на основе MenuetOS, использует собственные стандарты и не является POSIX или UNIX совместимой. Система рассчитана на использование ассемблера для написания приложений, но есть и программы, написанные на языках высокого уровня. Особенности:

* Множество вариантов загрузки, в том числе с NTFS-раздела. Также возможна загрузка прямо из Microsoft Windows (при этом Windows, конечно, завершается), не требующая установки системы.
* GUI на основе VESA. Возможность увеличения частоты монитора за счет уменьшения разрешения.
* Среда разработки: редактор, макро-ассемблер (fasm) для сборки ядра и приложений.
* Поддержка TCP/IP-стека и некоторых сетевых карт.
* Основной дистрибутив помещается на одной 3,5" дискете.
* Для запуска достаточно 8 мегабайт оперативной памяти.
* Вытесняющая многозадачность, возможность создания потоков, параллельное исполнение системных вызовов.
* Поддержка файловых систем FAT12, FAT16, FAT32, NTFS (только чтение), ISO 9660, тестируется поддержка Ext2, Ext3.
* Поддержка звукового кодека AC'97 для чипсетов Intel, nForce, nForce2, nForce3, nForce4, SIS7012, FM801, VT8233, VT8233C, VT8235, VT8237, VT8237R, VT8237R Plus и EMU10K1X.
* Поддержка USB (OHCI, UHCI).
* Поддержка воспроизведения Audio CD и DVD-дисков.
* Высокая отказоустойчивость, пониженное потребление электроэнергии системой.
* Частично портированы такие программы как DosBox, эмулятор NES, Spectrum, игры Quake, Doom, Косилка и другие.

История КолибриОС

Начало

Во второй половине 1990-х годов финский студент Вилле Турьянмаа писал графическую оболочку для DOS. Ему хотелось, чтобы его программы на Паскале выполнялись в многозадачном защищенном режиме и имели оконный графический интерфейс. Оболочка обрастала возможностями и со временем превратилась в самодостаточную операционную систему (DOS уже не требовался), которая запускалась с дискеты. Называлась эта операционная система MrOS, а объем ее исходных текстов на Паскале составлял 25 тысяч строк. В дистрибутив входили три десятка программ типа часов, проигрывателя AudioCD, игр, синтезатора пианино, регулятора громкости... Однако вскоре Вилле понял, что его 16-битная ОС устарела и требуется что-то менять. С другой стороны, ему не нравилось медленное выполнение программ, написанных на интерпретируемых языках, наличие прослоек между программой и железом. Он решил написать самое быстрое ядро в мире, и единственным подходящим языком для этого оказался ассемблер.

Через несколько месяцев был выпущен релиз новой ос под названием MenuetOS 0.01. Сообщество разработчиков MenuetOS было очень маленьким до 5 сентября 2001 года, когда было опубликовано интервью на сайте OSNews. Операционная система с графическим интерфейсом, написанная на ассемблере и влезающая на дискету, – подобного мир не видел с начала 1990-х годов. Люди загружали эту ОС, разносили весть о ней по форумам, создавали посвященные ей сайты. Отметим, что российские сайты были сделаны буквально через неделю после интервью. События 11 сентября 2001 года отвлекли внимание от MenuetOS, но развитие продолжалось.

 MenuetOS была еще сырым проектом, и в ней многого недоставало. К Вилле стали присоединяться разработчики. В начале 2002 года был портирован ассемблер FASM, и стало возможным компилировать ядро и программы прямо в MenuetOS.

MenuetOS RE

Осенью 2004 года Иван Поддубный выпустил 3 бета-версии дистрибутива RE#8 (Russian Edition), которые включили в себя новую подсистему GUI и плеер wav-файлов для материнских плат со встроенным звуком. Весной 2005 года Иван Поддубный выложил ядро, размер которого был уменьшен на 20% по сравнению с официальным при большей функциональности. Это ядро легло в основу дистрибутивов Kolibri 3 и Kolibri 3 SE, представивших улучшенный интерфейс пользователя (минимизацию окон и модифицированная панель задач), расширение набора системных функций и поддержку формата rtf. Выпущенный вскоре дистрибутив Kolibri 4 принес автоматическое определение мыши и дисков/разделов ЖД, а также новый загрузочный экран, позволяющий выбирать частоту вертикальной развертки монитора.

 В июне 2005 года вышла Menuet64 0.01, и Вилле дал понять, что развитие этой ОС должно быть основной целью не только для него, но и для всего сообщества, что вызвало раскол среди разработчиков. Исходные тексты Menuet64 были закрыты, поэтому ведущих программистов сообщества Menuet32 она не привлекла. Была продолжена работа над дистрибутивом Menuet32 - Kolibri.

KolibriOS

Выход очередной версии Kolibri в октябре 2005 года стал знаковым: поддержка FAT12 и менеджмент памяти, улучшенный драйвер RAM-диска и новая семантика обращения к жестким дискам. Различие между ядром Kolibri и MenuetOS достигло 50% – признак того, что родилась новая операционная система! Эта версия получила название KolibriOS 0.50 в знак сохранения преемственности от дистрибутива Kolibri.

От версии к версии KolibriOS развивалась и дополнялась, основательно менялась работа с жесткими дисками, модель работы с памятью, добавлялась поддержка актуальных файловых систем. Не раз было переработано ядро и обновлен графический интерфейс, добавлена поддержка многих форматов файлов, также была переписана вся подсистема работы с сетью. Добавилась поддержка USB, появился новый драйвер для относительно современных видеокарт ATI, стал экспериментально поддерживаться HD audio звук. Была портирована среда разработки приложений. На данный момент идет активная работа с сетевым функционалом системы, в том числе разрабатывается полноценный интернет-браузер. Система продолжает постепенно совершенствоваться и дополняться новыми программами, заметно расширяющими её функциональность.

Перспективы использования и развития КолибриОС

Несомненными преимуществами KolibriOS перед многими другими операционными системами на дискете являются: высокое быстродействие, возможности обращения к внешним дисковым накопителям и сохранения результатов работы, настраиваемый и приятный интерфейс, открытость исходных текстов. Ее можно рекомендовать программистам, желающим попробовать свои силы в создании ОС или изучении ассемблера. В случае, если Windows перестанет загружаться, KolibriOS поможет скопировать важные данные с раздела NTFS (или FAT) на другой раздел, притом что скорость копирования в ней на 6-8% выше, чем в Windows. KolibriOS видится удачной ос для применения во встраиваемых системах, системах сбора информации и управления производством, тонких клиентах. К тому же Колибри может быть неплохой ОС для устаревших компьютеров.

К сожалению, дальнейшее развитие KolibriOS до уровня популярных на сегодня операционных систем семейств Windows и Unix представляется практически невозможным без решения ряда принципиальных вопросов. Пожалуй, главный из них зародился вместе с появлением прародителя – MenuetOS. Проблема заключается в том, что изначально операционная система разрабатывалась одним единственным человеком, естественно без строгой стандартизации и что более важно почти без всякого проектирования операционной системы. Операционная система - очень сложный технический проект, писать который без проектирования в целом невозможно. Ядро операционной системы до сих пор время от времени дорабатывается и изменяется, и вследствие этого, одни его части становятся все менее согласованы с другими и ос в какой то момент может стать нестабильной. Да и просто приложения, написанные для ранних версий KolibriOS, могут оказаться неработоспособными в более современных версиях, а переписывать их при отсутствии гарантий совместимости с будущими версиями ос не слишком заманчиво. Несовместимость с другими популярными ос не позволяет использовать существующие приложения и предоставляемые ими возможности и имеющиеся в них наработки. Также немаловажными проблемами являются любительский статус проекта, и как следствие, отсутствие какой либо материальной выгоды и финансирования и малый коллектив разработчиков. Сплоченность этого коллектива однако внушает некоторый оптимизм относительно будущего KolibriOS.

Структура KolibriOS

Управление процессами.

1. Понятие процесса как такового в Колибри очень зачаточное: процесс - объединение потоков с одним и тем же адресным пространством. У всех таких объединяемых потоков одно и то же имя и один и тот же размер используемой памяти.

 Потоки, впрочем, существуют и обладают следующими характеристиками

* идентификатор (TID);
* состояние потока: активен, заморожен, завершается, ждёт переключения задач на него;
* окно: каждый поток имеет ровно одно окно, которое может быть невидимым, но обязательно существует;
* использование процессора: число тактов за последнюю секунду, которое процессор потратил на выполнение именно этого потока;
* имя процесса (имя исполняемого файла);
* маска событий, о которых система извещает поток;
* системный стек;
* список объектов ядра, ассоциированных с этим потоком;
* карта разрешённых портов ввода/вывода;

2. Процесс не идентифицируется никак; информация о потоках внутри ОС собрана в статический массив на 255 входов (нумеруемых от 0 до 255, причём 0-й слот не может использоваться, так что всего в системе может быть не более 255 потоков)

 Некоторые системные функции принимают номер слота, некоторые – идентификатор потока.

3. Взаимодействия процессов практически нет - есть только специальная системная функция для передачи данных от процесса-источника процессу-приёмнику, причём приёмник должен заранее подготовить буфер и ожидать этих данных, и некоторые возможности по отладке приложений.

 4. Соответственно синхронизации тоже практически нет - один процесс может только проверить, завершился ли другой.

5. Планировщик циклически выделяет процессорное время всем активным потокам. Без всяких дополнительных ухищрений.

Управление вводом-выводом: аппаратура с точки зрения ядра

Устройства бывают разные. Бывают стандартные устройства, которые понимает система. Система самостоятельно работает с таймером, мышью, клавиатурой, видеокартой, аудио, системным динамиком, сетевыми картами, CD/DVD, жёсткими дисками, не давая приложениям доступа к этим устройствам напрямую.

Для получения данных мыши есть специальная системная функция и специальное событие - при любом передвижении мыши система извещает все потоки, что с мышью что-то произошло. Приложение может управлять формой курсора мыши для своего окна, настройками движения мыши, может переместить курсор в нужную позицию, может симулировать нужное состояние клавиш мыши.

Внутренне в системе происходит следующее. Драйвер мыши при поступлении очередного события вызывает функцию SetMouseData с нужными аргументами, сообщая ядру, что именно произошло с мышью. Ядро преобразует данные о движении мыши в перемещения курсора в соответствии с настройками, обновляет свои переменные и устанавливает флаг активности мыши, а когда главный цикл системы получит управление, он проверит этот флаг и известит все приложения, что что-то произошло с мышью. Работа идёт по схеме (мышь) <-> (драйвер) <-> (ядро) <-> (приложения).

Работа с клавиатурой. У приложения есть два режима получения данных о нажатых клавиш: ASCII и сканкоды. Действие нажатой клавиши зависит от следующих вещей:

 является ли эта клавиша модификатором (Alt/Shift/Ctrl/\*Lock) или нет;

 в каком режиме находится активное окно (ASCII/сканкоды);

 было ли установлено соответствующее сочетание клавиш как горячая комбинация для захвата каким-то другим приложением.

Обработчик клавиатуры обновляет состояние клавиатуры при нажатии (Alt/Shift/Ctrl/\*Lock), проверяет, не нажато ли Ctrl+Alt+Del, сканирует список установленных горячих комбинаций. ASCII - таблицы для каждого языка свои, переключение языка заключается в установке правильной таблицы. Клавишам F1-F12 тоже соответствуют определённые коды. Поддержка клавиатуры выполнена в ядре. Схема обработки:

(клавиатура) <-> (ядро) <-> (приложения).

Работа с видеокартой – подразумевается в GUI. Вывод на экран осуществляет ядро, для видеокарт от ATI есть специальный драйвер, вспомогательный для ядра, поддерживающий аппаратный курсор и вроде в некоторых случаях аппаратное ускорение (через API для приложений). Поддерживаются стандартные видеорежимы EGA/CGA и VGA и видеорежимы, возвращаемые VESA BIOS. Установка видеорежима осуществляется средствами BIOS при загрузке ещё в реальном режиме процессора, а схема работы в типичном случае выглядит так:

(видеокарта) <-> (ядро) <-> (приложения),

 \ /

 (драйвер)

Иногда приложения могут обращаться напрямую к драйверу видеокарты.

Поддержка аудио есть для AC97-кодеков на определённой аппаратуре, здесь ядро уже не принимает прямого участия, а приложение общается напрямую с соответствующим драйвером, который предоставляет соответствующие API.

Системным таймером управляет исключительно ядро. При загрузке система программирует таймер на срабатывание 100 раз в секунду. Обработчик прерывания от таймера делает следующее:

 увеличивает текущее время, вызывает процедуру обработки звуков для системного динамика

 каждую 100-ю итерацию (каждую секунду) обнуляет счётчик "тактов в предыдущую секунду" (поле в структуре потока) у всех потоков;

 служит планировщиком, переключаясь на следующую задачу, а при переключении увеличивается счётчик тактов у текущего потока (от которого управление уходит) и заполняются системные структуры - стек, карта разрешения ввода/вывода, отладочные регистры.

Работа с сетевыми картами зашита в ядро и осуществляется через API.

Чтение данных с CD/DVD, равно как и работа с жёсткими дисками и дискетами, относится скорее к области файловой системы, хотя и реализовано в ядре.

Управление вводом-выводом: аппаратура с точки зрения приложений

Есть устройства, о которых система ничего не знает. Тут появляются два варианта работы с ними.

Напрямую из приложений.

 Ядро предоставляет API, позволяющее приложениям самостоятельно обрабатывать аппаратуру напрямую, для чего нужно:

1. иметь возможность доступа к соответствующим портам ввода/вывода;

 и/или

2. иметь возможность доступа к пространству PCI;

 и/или

3. иметь возможность доступа к нужной области физической памяти;

 и/или

4. обрабатывать IRQ от устройства.

1. Приложение может попытаться зарезервировать диапазон портов, нужный для общения с устройством. Ядро хранит список уже зарезервированных портов и не даст зарезервировать диапазон, перекрывающийся с чем-то уже занятым. Если же всё в порядке, то ядро в карте разрешения ввода/вывода процессора для запрашивающего потока разрешает обращения к запрошенным портам, а также добавляет диапазон в общий список зарезервированных. После этого приложение получает возможность обращаться напрямую к нужным портам. При загрузке ядро резервирует для себя некоторые системные порты. Поток может освободить ранее выделенные им же порты.

2. Здесь ядро берёт на себя все операции, а приложения обращаются к аппаратуре через API.

3. Доступ из приложения к любым конкретным адресам в физической памяти безусловно запрещен. К ним прикладные программы должны обращаться через API-функции и драйверы устройств.

Однако в случае с нестандартными устройствами Колибри предоставляет пользовательскому приложению возможность отображения скрытых от него физических адресов MMIO на видимое ему "своё" адресное пространство. Разработчик нового драйвера или инженер-электронщик, тестирующий новое устройство, должен заранее указать его PCI-адрес в системной константе mmio\_pci\_addr. По умолчанию эта константа в ядре не определена и пользовательский доступ к MMIO закрыт.

4. Разные прерывания обрабатываются различными способами. Некоторые IRQ ядро обрабатывает самостоятельно (IRQ0 - таймера, и IRQ1 - клавиатура, IRQ6 - дискета, IRQ14 и IRQ15 - жёсткие диски). Одну часть остальных IRQ может перехватить приложение, IRQ другую может перехватить драйвер (о драйверах - ниже). Перехват IRQ приложением и определение действий при его поступлении осуществляется также с помощью функций API.

Из соответствующего драйвера.

Драйвер выполняется с наивысшими правами и, как следствие, получает доступ ко всему, чему можно. В частности, с чтением/записью в порты никаких проблем не возникает. Кроме того, ядро предоставляет некоторые сервисы драйверам. В их числе есть сервис (AttachIntHandler и GetIntHandler) для перехвата определенных IRQ, при приходе такого IRQ ядро просто вызывает функцию драйвера, адрес которой драйвер указывает в AttachIntHandler, сервис (Pci{Read/Write}) для чтения/записи PCI-регистров, сервис (GetPgAddr) для получения физического адреса по известному виртуальному.

Файловая система

1. Собственной файловой системы у Колибри нет, стандартно используется FAT, есть чтение с NTFS, делается поддержка файловой системы ext2 (на данный момент реализовано чтение каталогов).

 2. Управляющих блоков для файлов нет. При всех операциях с файлами приложение задаёт полное имя файла (а ядро, соответственно, это полное имя каждый раз разбирает).

 3. Существует API файловой системы.

 4. Ограничений доступа нет (всё, что в принципе можно сделать, может сделать любое приложение). Соответственно и авторизации нет.

 5. Запись есть только на FAT. Освобождение дискового пространства заключается просто в пометке соответствующих кластеров в таблице FAT как свободных. Выделение нового дискового пространства делается покластерно, для нахождения очередного свободного кластера сканируется таблица FAT на предмет поиска кластеров, помеченных в таблице FAT как свободные; для FAT12 (дискет и рамдиска) сканируется таблица от начала до конца, для FAT16 и FAT32 (жёстких дисков) есть внутренняя переменная - номер кластера, с которого начинается поиск, и каждый следующий поиск начинается с кластера, следующего за предыдущим найденным свободным и здесь поиск, дойдя до конца, продолжается с начала тома.

 7. Основная система - FAT, с NTFS и EXT2 есть только чтение, соответственно журналирования нет.

Управление памятью

Используемый тип управления: Страничная организация памяти, плоская модель. Нижние 2 Гб виртуальной памяти (диапазон адресов 0-0x7FFFFFFF включительно) отводятся приложению (свои для каждого процесса), верхние 2 Гб - для системы (и разделяются между всеми процессами).

Принцип отображения адресного пространства: Стандартным образом, через таблицы страниц. Файла подкачки нет. Для преобразования адресов выполнены следующие утверждения:

преобразование нетождественно;

преобразование нижних 2 Гб зависит от текущего процесса, преобразование верхних 2 Гб не меняется при переключении задач;

начало системных адресов размечается на начало физической памяти

первые 4 Мб (куда входит само ядро и часть системных таблиц) размечаются одной "длинной" страницей;

размер остальных страниц - по 4Кб;

Для 4Кб-страниц действует преобразование линейных адресов в физические:

обращаться к таблицам страниц можно только имея наивысшие права, по линейным адресам. Однако существует важный диапазон линейных адресов, для которых таблицы страниц расположены в системной области в виде "плоского" статического массива. Длина этого диапазона равна размеру доступной физической памяти, но не может превышать 2Гб.

Управление физической памятью.

Система хранит массив битов, который для каждой физической страницы описывает, выделена она или свободна, а также некоторые вспомогательные переменные. Физические страницы выделяются по принципу first-fit, возвращается первый подходящий вариант (первая свободная страница либо первый свободный блок нужной длины).

Управление адресным пространством ядра.

 В ядре есть специальные процедуры, которые соответственно выделяют и освобождают непрерывный диапазон в адресном пространстве ядра.

Управление памятью ядра.

Есть функция получения физического адреса по указанному линейному, функция установки для указанной физической страницы указанного линейного адреса, способная также очистить линейный адрес страницы. Существует аналогичная функция для непрерывного блока адресов и обратная ей функция. Есть общая функция выделения памяти ядра, которая одновременно выделяет место в адресном пространстве ядра, физическую память, устанавливает соответствие между ними и возвращает линейный адрес блока, а также обратная ей функция освобождения памяти ядра. Наличествуют функции, предназначенные для выделения маленьких блоков памяти, когда выделение памяти целыми страницами излишне.

Выделение памяти для приложений

Берется размер памяти, указанный в заголовке бинарного файла программы, и сам загруженный в память ядра файл. Проверяется, достаточно ли свободной физической памяти для приложения; если недостаточно, возвращается ошибка. Выделяется новая страница, размечается по линейному адресу. Устанавливается текущая таблица страниц, создаётся нужное количество страниц. В адресное пространство нового процесса помещается загруженный бинарный файл. Теперь текущая таблица страниц соответствует адресному пространству нового процесса.

Освобождение памяти от приложений

Выясняется, является ли текущий поток последним в своём процессе (ищутся потоки с тем же адресным пространством), в противном случае освобождение памяти не выполняется. Каждая выделенная страница помечается как зарезервированная системой, вызывается вспомогательная процедура, освобождающая все выделенные страницы, затем освобождается текущая таблица страниц.

Работа с памятью приложений.

Раньше адресное пространство приложения было обязано быть непрерывным диапазоном, который для приложения начинался с нулевого адреса; с тех пор в структуре, возвращаемой функцией API для потока, есть поля "адрес процесса в памяти" и "размер используемой памяти" (точнее, лимит = размер-1). В то время единственной возможностью по динамическому перераспределению памяти было изменение размера адресного пространства, и с того времени идёт функция API, которая при уменьшении используемой памяти проходит по "лишнему" пространству и освобождает выделенные страницы, а потом проходит по "добавляемому" пространству и выделяет запрошенные страницы. Кроме того, в конце она вызывает вспомогательную функцию, проходящую по списку потоков и для всех потоков текущего процесса обновляющую поле с размером памяти в структуре потока.

Однако при таком подходе далеко не всегда можно освободить память, которая стала ненужной. Поэтому была написана структура данных (куча) для приложений, и функции, работающие с отдельными страницами и блоками страниц.

Использование кучи несовместимо с перераспределением памяти первым способом, так что для активации режима кучи нужно вызвать соответствующую функцию API, которая инициализирует кучу. Организация данных: есть некоторые поля в структуре потока, хранящие базу кучи и размер адресного пространства, отводимого под кучу; в куче бывают выделенные и свободные блоки (все блоки занимают целое число страниц), все блоки организованы в односвязный список. Выделение блока - алгоритм first-fit, в цикле по блокам находим первый свободный блок подходящего размера; если он оказался в точности запрошенного размера, то он просто переводится в статус занятого, иначе от него отделяется хвост, остающийся свободным блоком. В любом случае страницы нового блока помечаются значением 2 (отложенное выделение физической памяти). Функция освобождения блока освобождает все выделенные страницы из блока, помечает блок как свободный, после чего проходит по списку блоков, объединяя соседние свободные блоки в один свободный блок. Функция перераспределения блока при уменьшении размера блока освобождает лишние страницы и либо создаёт новый свободный блок, либо расширяет следующий свободный блок. При увеличении размера блока функция смотрит, есть ли сразу после запрошенного блока свободный блок нужного размера (можно ли увеличить запрошенный блок на месте), если нет, то ищет свободный блок полного размера (тоже first-fit), затем переопределяет все физические страницы из старого блока в новый, помечает старый блок как свободный и запускает объединение свободных блоков. Добавленные страницы в любом случае помечаются значением 2. В конце вызывается функция, проходящая по списку потоков и для всех потоков текущего процесса обновляющая поле с размером памяти в структуре потока.

Интерфейс прикладного программирования и интерфейс пользователя

Вследствие любительского статуса KolibriOS, очень малой её распространенности и того что практически каждый её постоянный пользователь является разработчиком данной операционной системы либо приложений к ней, и интерфейс прикладного программирования в данном случае отчасти является интерфейсом пользователя. API представлено приблизительно 70тью функциями, достаточно примитивными, что подразумевается их количеством. При использовании таких функций программа работает весьма быстро, но процесс написания программ заметно усложняется, ведь подчас разработчику приходится самому создавать из них сложные и универсальные конструкции, работать с которыми удобнее. Разумеется, концепция быстрой разработки приложений в принципе не может быть реализована. Программисту приходится обращать больше внимания на алгоритм программы, а не на данные с которыми она работает. Как и в других операционных системах, использование api-функций сопряжено с проблемой совместимости старых программ и новых версий операционной системы, так как интерфейс прикладного программирования может меняться от версии к версии путем удаления старых, неэффективных функций и добавления новых. Соответственно, если функция была удалена, приложение работать не сможет или не будет обладать полным закладывавшимся при его разработке функционалом. Пожалуй, наиболее сложные и мощные api-функции используются при разработке графического интерфейса пользователя, хотя это тоже очень трудоемкий процесс, по сравнению, например, с разработкой интерфейса windows-приложений. Тип графического интерфейса пользователя простой: типовые экранные формы и стандартные элементы интерфейса, обеспечиваемые самой подсистемой GUI, так как сложность реализации собственного графического интерфейса в подавляющем большинстве случаев неоправданна. Тем не менее, существующий на данный момент графический интерфейс пользователя достаточно функционален, и прибегать к интерфейсу в виде командной строки, несмотря на её наличие, нет нужды. Одним из требований к хорошему графическому интерфейсу программной системы является концепция «делай то, что я имею в виду» или DWIM (англ. Do What I Mean). DWIM требует, чтобы система работала предсказуемо, чтобы пользователь заранее интуитивно понимал, какое действие выполнит программа после получения его команды. Здесь нельзя не отметить, что в отличие от многих других ОС Колибри концепции соответствует.

Заключение

Операционная система КолибриОС является крайне необычным продуктом. Она не удовлетворяет требованиям к современной ОС, поскольку не является расширяемой, переносимой, совместимой и достаточно безопасной. Она лишь частично удовлетворяет эксплуатационным требованиям, хотя и является надежной и отказоустойчивой. Данные проблемы крайне сложны для преодоления, однако решаемы. Текущее же предназначение ОС делает описанные проблемы несущественными, а её преимущества начинают значительно перевешивать недостатки. И тем не менее все же очень хочется увидеть операционную систему от русскоязычных разработчиков выступающую на мировом рынке наравне с Windows и Unix системами.