УДК 019.963.2:004.62, 019.963.2:004.05

А. С. Велижанин, **А. В. Ревнивых**

Тюменский государственный нефтегазовый университет
ул. Володарского, 38, Тюмень, 625000, Россия

Институт вычислительных технологий СО РАН
пр. Акад. Лаврентьева, 6, Новосибирск, 630090, Россия

E-mail: Anatoliy.Velizhanin@gmail.com; alexchr@mail.ru

МОДУЛЬНАЯ АРХИТЕКТУРА КАК ОСОБЕННОСТЬ ПОСТРОЕНИЯ СОВРЕМЕННЫХ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ТЕХНОЛОГИЯ ИНТЕГРАЦИИ МОДУЛЕЙ В ОБЩУЮ СИСТЕМУ

Работа посвящена анализу модульной архитектуры построения современных информационно-вычислительных систем. Рассмотрены особенности технологии интеграции модулей в общую информационно-вычислительную систему, непосредственно влияющие на надежность информационно-вычислительного комплекса в целом.

Ключевые слова: информационная безопасность, надежность, информационные системы, интерфейс, интеграция.

Введение

Информационные системы и технологии становятся все более значимыми для современного человека. Общество, цивилизация, общественный строй интенсивно изменились благодаря появлению сотовых телефонов, на смену которым приходят смартфоны и коммуникаторы, карманных персональных компьютеров, систем автоматизированного проектирования (скажем, SCADA систем объектов нефтегазовой промышленности), систем управления электро-, газо- и водоснабжением. Любые крупные современные транспортные объекты также управляются автоматикой (бортовые компьютеры на современных автомобилях, системы автоматики самолетов и их диспетчерские пункты). Многие из современных систем автоматизации нуждаются во взаимодействии в режиме реального времени, а также требуют безотказного и бесперебойного функционирования. Примерами таких систем могут быть как программно-аппаратные комплексы по управлению процессами на атомных электростанциях, так и уже описанный пример диспетчерского пункта по управлению полетами.

Дополнительной сложностью является необходимость проектирования и создания программных комплексов, призванных работать в гетерогенной среде – на разных программных и аппаратных платформах с различным оборудованием, произведенным различными организациями, а также распределенных программных комплексов, функционирующих в сетях крупных организаций.

Однако помимо сложности современных программных комплексов отметим и аналогичные особенности аппаратных частей. Современная микроконтроллерная и микропроцессорная техника также нуждается в тесной интеграции как между основными компонентами системы, так и с периферийным оборудованием.

Результатом столь высокой распространенности и одновременно чрезвычайной сложности современных информационных систем является не только значительное повышение эффективности производства за счет повсеместной автоматизации, но и сложность, а также

высокая стоимость восстановления работоспособности данных систем. Вследствие указанных и других не менее важных факторов вопросы информационной безопасности на современном этапе рассматриваются как приоритетные в государственных структурах, научных учреждениях и в коммерческих фирмах.

Однако одной из наиболее серьезных проблем при обеспечении безопасности в информационных системах является обеспечение защищенности систем от угроз, связанных с несовершенством самой спроектированной системы. На данном этапе развития информационных технологий одной из причин угроз такого типа является сложность создаваемых систем, которые состоят зачастую из множества различных аппаратных и программных платформ, призванных работать в тесной интеграции друг с другом, используя для обеспечения коммуникаций между элементами системы различные интерфейсы, а также масштабы внедрения автоматизированных систем обработки информации и управления в современных программно-аппаратных вычислительных комплексах.

Технология интерфейсов

Интерфейс (от англ. *interface* – поверхность раздела, перегородка; от лат. *inter* – между, и *face* – поверхность) – совокупность средств, методов и правил взаимодействия (управления, контроля и т. д.) между элементами системы [1].

Этот термин используется практически во всех областях науки и техники. Его значение относится к любому сопряжению взаимодействующих сущностей (как естественно-научных, так аппаратных и человеко-машинных). Под интерфейсом понимают не только устройства, но и правила (протокол) взаимодействия этих устройств.

Примеры интерфейсов:

1) вожжи – главный элемент интерфейса между лошадью и кучером, или интерфейс системы «лошадь – кучер»);

2) руль, педали газа и тормоза, ручка коробки переключения передач – интерфейс (управления) автомобиля, или интерфейс системы «водитель – автомобиль»;

3) электрические вилка и розетка являются интерфейсом энергоснабжения большинства бытовых приборов;

4) элементы электронного аппарата (автомагнитолы, часов и т. д.) – дисплей, набор кнопок и переключателей для настройки, плюс правила управления ими – интерфейс системы «человек – машина»;

5) клавиатура и мышь – элементы интерфейса в системе «пользователь – ЭВМ» (в свою очередь и клавиатура, и мышь имеют собственные интерфейсы сопряжения с компьютером).

В зависимости от контекста понятие применимо как к отдельному элементу (интерфейс элемента), так и к связкам элементов (интерфейс сопряжения элементов).

Отметим, что интерфейсы можно разделить на два вида по критерию наличия обратной связи: с обратной связью и без нее. Кроме того, при сопряжении интерфейсов двух сущностей один из них может быть управляющим, а другой – управляемым. Интерфейсы при сопряжении могут быть и равноправными.

Пользователю элемента незачем знать, как реализован используемый элемент, чтобы управлять им, но используемый элемент должен предоставить интерфейс управления. Например, водителю вовсе не обязательно знать, как устроен двигатель, трансмиссия, тормозная система и рулевое управление, чтобы управлять автомобилем, достаточно пользоваться интерфейсом автомобиля (рулем и педалями). Более того, интерфейс управления избавляет водителя даже от необходимости вообще знать о наличии в составе автомобиля упомянутых выше частей.

Интерфейсы в вычислительной технике являются одним из необходимых условий существования любой информационной системы, поскольку именно они являются основой взаимодействия всех компонентов современных информационных систем, как между собой, так и с окружающими объектами. Если интерфейс какого-либо объекта (персонального компьютера, программы, функции) не изменяется (стабилен, стандартизирован), это даёт возмож-

[1] http://ru.wikipedia.org/wiki/Интерфейс

ность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами (например, научившись работать с одной программой под Windows, пользователь с легкостью освоит и другие, потому что они имеют интерфейс, организованный по одним и тем же принципам).

В любой современной вычислительной системе взаимодействие между ее элементами может осуществляться на нескольких основных уровнях.

1. Человек – устройства ввода / вывода информации

По всей видимости, человека тоже следует считать частью информационной системы, так как именно человек является основным носителем и «устройством» обработки информации, а также именно человек является создателем информационно-вычислительных систем. Принято считать, что человек оперирует информацией посредством сознания. В физиологии органом, отвечающим за переработку и хранение информации, считается мозг. Между тем очевидно, что человеческое тело как биологическая система также участвует в процессах восприятия (через обоняние, осязание, зрение) и обработки информации. Помимо вышеперечисленного, человек так же является существом социальным, что непременно отражается не только во множестве различных общественных организаций, но и в обычном свойстве заводить друзей. В процессе общения же человек, как и любая информационная система, производит передачу информации посредством письменной или устной речи или другими способами. Человеку, например, для устного общения необходимо обмениваться информацией на понятном ему языке, который в данном примере будет являться интерфейсом для обмена информацией.

2. Устройства ввода / вывода информации – аппаратный компонент обработки сигнала.

3. Аппаратный компонент обработки сигнала – программный компонент.

4. Программный компонент – аппаратный компонент обработки сигнала.

5. Аппаратный компонент обработки сигнала – устройства ввода / вывода информации.

Любое техническое устройство является сложным, состоящим из множества различных элементов. Во многих аппаратных элементах (например, микроконтроллерах) имеется встроенное программное обеспечение (так называемая «прошивка»). В связи с этим радикальное деление на программные и аппаратные компоненты является не вполне корректным. Устройство ввода-вывода информации не является элементарным и представляет собой совокупность различных компонентов, поэтому является в то же время и устройством обработки информации, однако для создания более четкой классификации было бы желательно абстрагироваться от деталей исполнения устройств и обратить большее внимание именно на основную задачу этого компонента информационно-вычислительной системы.

При еще более широком взгляде на предложенную классификацию по уровням можно выделить также уровень взаимодействия «программный компонент – программный компонент». Данный уровень сочетает 3-й и 4-й уровни по указанной классификации при большей степени абстрагирования от деталей взаимодействия между программными компонентами и аппаратных элементов обработки сигнала.

Рассматривая указанные уровни, предположим, что они одновременно являются одним из эффективных критериев, по которому было бы целесообразно разделить интерфейсы на типы. Однако необходимо принять во внимание, что каждый элемент вычислительной системы может предоставлять несколько интерфейсов, которые, вполне возможно, будут абсолютно различных типов при классификации по различным критериям. Рассмотрим манипулятор «Мышь». Согласно приведенной выше классификации этот элемент вычислительной системы предоставляет интерфейсы взаимодействия уровня «человек – устройство ввода информации», а также «устройство ввода информации – аппаратный компонент обработки информации».

В данном случае манипулятор «Мышь» также является и устройством обработки информации, так как он содержит, к примеру, компонент, отвечающий за восприятие информации о перемещении, а также нажатий клавиш на нем, с дальнейшей обработкой этой информации с помощью программного обеспечения, встроенного в микросхемах данного устройства. Однако, как уже сообщалось выше, было бы желательно абстрагироваться от деталей исполнения устройств и сосредоточить внимание на основной задаче элемента информационно-вычислительной системы.

В дальнейшем «аппаратный компонент обработки информации» будет взаимодействовать с уровнем «программный компонент», в роли которого будет выступать драйвер и т. д. Кроме того, отдельного внимания заслуживает понятие «пользовательский интерфейс», заключающее суть метода ввода и вывода информации (например, графический – подразумевает под собой визуализацию в виде графических элементов, текстовый – консольное представление и т. п.). Также понятие интерфейсов имеется и в концепции объектно-ориентированного программирования, где основная задача интерфейса – предоставить информацию об имеющихся функциях программного элемента и правилах его использования. Интерфейс – семантическая и синтаксическая конструкция в коде программы, используемая для специфицирования услуг, предоставляемых классом или компонентом. Интерфейс определяет границу взаимодействия между классами или компонентами, специфицируя определенную абстракцию, которую осуществляет реализующая сторона. В отличие от многих других видов интерфейсов интерфейс в объектно-ориентированном программировании является строго формализованным элементом объектно-ориентированного языка и, в качестве семантической конструкции, широко используется кодом программы.

Как уже сообщалось, любой программный или аппаратный модуль использует какого-либо рода интерфейсы. И если рассмотреть крупные программные решения, такие как операционные системы, то все они предоставляют специализированные интерфейсы для взаимодействия с ними других программных продуктов. Примером может быть API (Application Programming Interface) в операционной системе Windows, используемый для создания как прикладного, так и системного программного обеспечения для этой платформы. Подобный API предоставляют и другие операционные системы. Помимо вышеперечисленного программный пакет «Microsoft Office» также предоставляет на основе модели COM (Component Object Model) технологию OLE Automation (Object Linking and Embedding), предназначенную в данном случае для взаимодействия с Microsoft Office. Помимо перечисленного выше, существует и множество другого различного программного обеспечения, предоставляющего интерфейсы для взаимодействия с ним, а также существует множество других технологий, предназначенных для осуществления коммуникаций между различными программными компонентами. Примером технологии межпроцессного взаимодействия может быть система D-Bus, являющаяся частью проекта «freedesktop.org» и способная работать на всех «POSIX-совместимых» операционных системах [2], технология программных каналов, сегментов разделяемой памяти, очередей сообщений и др.

Ввиду сложности программных и аппаратных интерфейсов, объемности множества программных решений, а также большого количества различных компонентов, требующих взаимодействия между собой, в современных информационно-вычислительных системах может возникнуть возможность неправильного использования интерфейсов и вследствие этого возникновения ошибок в программных и аппаратных модулях.

Большая часть аппаратного обеспечения выпускается крупными фирмами и тщательно подгоняется под платформу, в которой планируется его применение. Однако при рассмотрении программной составляющей следует обратить внимание на то, что здесь помимо большого количества крупных производителей имеется и множество небольших фирм, производящих, возможно, не менее качественное программное обеспечение, но при этом рост количества различной программной составляющей значительно выше количества нового различного аппаратного обеспечения. В связи с бурным развитием информационных и компьютерных технологий становится все более злободневной проблема стандартизации интерфейсов и обеспечения надежного и безопасного их использования, а также надлежащего документирования.

Ввиду чрезвычайного разнообразия программного обеспечения (а также большого количества аппаратных платформ, на которых программное обеспечение работает) дополнительного внимания заслуживает уровень «программный компонент – программный компонент».

[2] http://ru. wikipedia.org/wiki/D_Bus

Интерфейсы как основа модульной архитектуры построения современных информационно-вычислительных систем

Как уже упоминалось, взаимодействие элементов любой современной вычислительной системы строится на основе интерфейсов. Интерфейсы предоставляют колоссальный потенциал расширяемости системы и дают возможность удобного распараллеливания процесса создания вычислительной системы, когда каждый отдельный элемент может разрабатывать своя команда разработчиков, а затем интегрировать каждый модуль в общую систему.

Возможность создания модульной архитектуры вычислительной системы обеспечивается интерфейсами каждого отдельного модуля, однако в связи со сложностью современных вычислительных программно-аппаратных комплексов повышается риск неправильного использования интерфейсов из-за нехватки документации. Кроме того, имеется возможность подмены определенного модуля злоумышленником для осуществления каких-либо неправомерных действий. Например, в операционной системе Windows существует проблема, называемая «Ад dll», когда одна из библиотек, хранящаяся в общедоступном месте, заменяется другой с таким же именем файла другим, не предназначенным для нанесения ущерба, программным обеспечением. Кроме того, неработоспособность системы может быть вызвана тем, что доступ к определенному модулю закрыт вследствие ошибки при администрировании или по каким-либо другим причинам. При взаимодействии модулей необходимо соблюдать разрядность не заменяя, к примеру, 32-битные динамические библиотеки 64-битными аналогами.

Неправильное использование интерфейсов уровня «программный компонент – программный компонент» может привести не только к незначительному сбою в прикладном программном обеспечении, но и нанести непоправимый вред всему информационно-вычислительному комплексу в случае возникновения аналогичной ошибки, к примеру, при работе драйвера-фильтра. Ошибки же взаимодействия аппаратного обеспечения вычислительной системы могут привести к физическому разрушению системы. Кроме того, любые ошибки как в программном, так и в аппаратном обеспечении могут нанести непоправимый вред не только в виде разрушения самой информационно-вычислительной системы, но и в виде физического разрушения окружающей среды. Например, ошибки в функционировании системы поддержания режима работы ядерного реактора могут привести к аварии на атомной электростанции. В подобной ситуации масштабы катастрофы могут быть колоссальными. В связи со всеми вышеизложенными фактами безопасность использования интерфейсов становится одной из важнейших тем при создании любых вычислительных систем, а в особенности систем, требующих безотказной и бесперебойной работы, функционирующих в режиме реального времени и управляющих сложными и, возможно, чрезвычайно опасными технологическими процессами.

Для демонстрации эффективности приведенных выводов, рассмотрим пример, в котором в качестве информационной системы выступает ноутбук Acer Aspire 5742G. На данном ноутбуке используется видеокарта NVidia GeForce GT 520M и технология NVidia Optimus. Данная технология предназначена для продления времени автономной работы ноутбуков за счет более эффективного энергопотребления. Эффективность энергопотребления достигается за счет распределения нагрузки на 2 видеокарты, одна из которых более слабая и применяется для прорисовки несложной графики на мониторе, а другая – более мощная и предназначена для воспроизведения более сложной графики. Переключение между видеокартами происходит автоматически без непосредственного участия пользователя. За счет балансирования нагрузки достигается наиболее подходящий режим энергопотребления [3]. Полный набор драйверов для всего оборудования ноутбука предоставляется производителем на интернет-ресурсе «www.acer.com» для скачивания. Также загрузка драйверов для видеокарты фирмы «Nvidia» доступна на официальном сайте производителя (www.nvidia.com). Производитель ноутбука, выпускаемого под торговой маркой «Acer», предлагает комплект драйверов с поддержкой только операционной системы Microsoft Windows 7 различных редакций.

[3] http://www.nvidia.ru/object/optimus_technology_ru.html; http://www.thg.ru/graphic/nvidia_optimus/index.html

В эксперименте использовались операционные системы Microsoft Windows 7 Ultimate 64bit и Microsoft Windows 7 Home Premium 64 bit, а также драйвера от производителя «NVidia» версии 275.33 и драйвера, предлагаемые фирмой «Acer».

Рассмотрим попытку установки официального драйвера NVidia. Загружаем файл дистрибутива с официального сайта производителя. После успешно выполненной загрузки при запуске скачанного файла происходит предварительная распаковка установочных файлов драйвера видеокарты в указанную папку. После запуска распакованной инсталляционной программы производится проверка наличия совместимого оборудования, а в дальнейшем – установка подходящего драйвера. Однако на этапе проверки наличия совместимого оборудования официальной программой инсталляции драйвера версии 275.33 для данного ноутбука происходит ошибка с сообщением: «The graphics driver could not find compatible graphics hardware». Графический драйвер не смог найти совместимое графическое оборудование (рис. 1).

Проведем эксперимент по установке драйвера, предложенного производителем ноутбука. Драйвера также загружены с официального интернет-ресурса (www.acer.com). Версия используемого в эксперименте драйвера – 8.17.12.5903. При попытке установить драйвер возникает аналогичная ошибка (рис. 2). Произведем попытку установки аналогичного драйвера, также предложенного производителем ноутбука другой версии. В данном эксперименте будем использовать драйвер версии 8.17.12.5997. После запуска инсталляционной программы, как и в предыдущем случае, возникает такое же сообщение об ошибке инсталляции, как и в случае с драйвером версии 8.17.12.5903 (см. рис. 2).

Таким образом, имеем факт невозможности установки официальных драйверов как производителя видеокарты «NVidia», так и производителя ноутбука «Acer».

Проведем эксперимент по установке драйвера, используя стандартный диспетчер устройств операционной системы Windows 7. После выбора папки источника драйвера с любым из приведенных выше официальных драйверов установщик также не находит подходящего драйвера для данного устройства.

Рис. 1. Сообщение об ошибке при установке официального драйвера видеокарты от NVidia

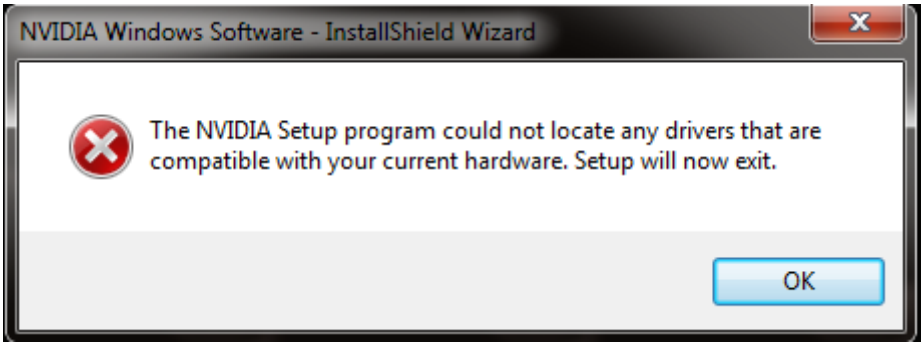

Рис. 2. Сообщение об ошибке при установке официального драйвера видеокарты от производителя ноутбука «Acer»

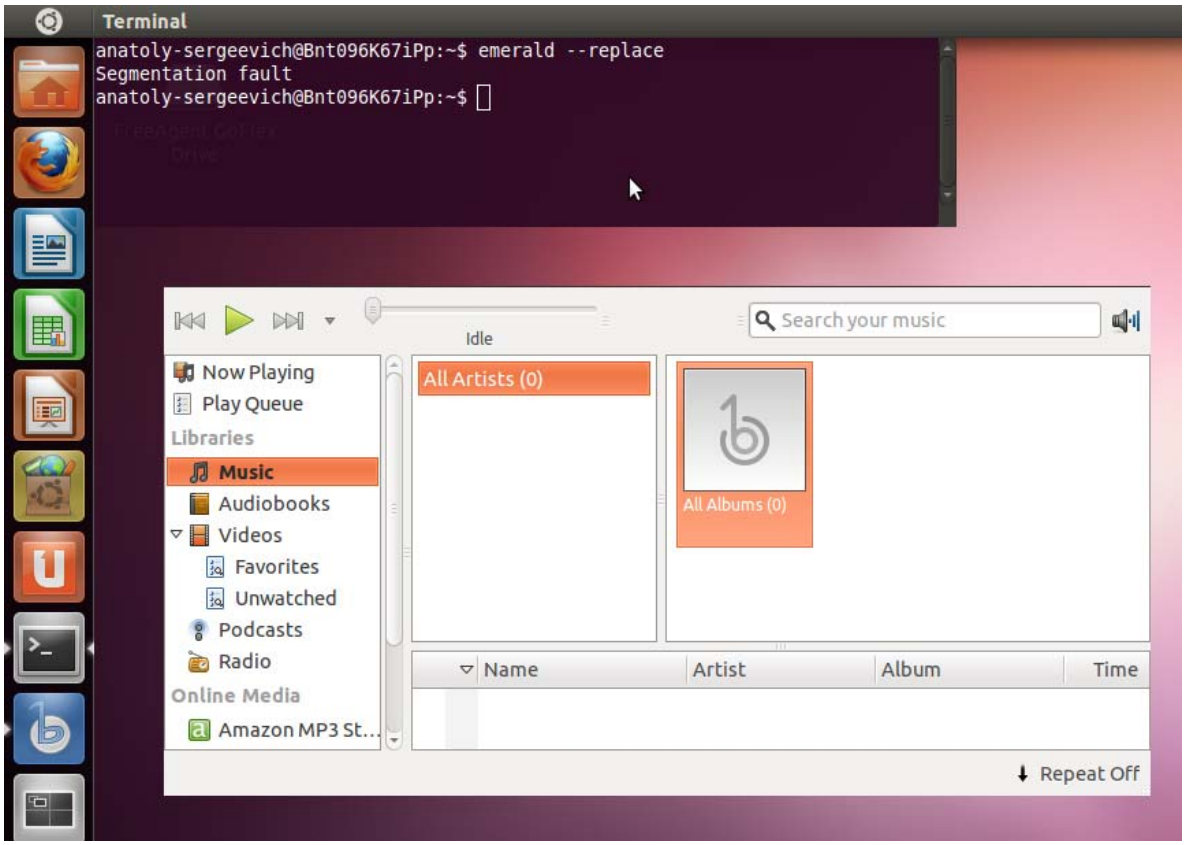

Рис. 3. Иллюстрация сообщения об ошибке при активации Emerald и результата – отсутствия декораций окон

Учитывая тот факт, что производитель видеокарты «NVidia», а также производитель ноутбука «Acer» утверждают, что данные инсталляционные программные модули содержат необходимые драйвера для данной видеокарты, можно предположить, что причиной невозможности установки драйвера могут быть конфигурационные файлы программ инсталляции. Решением проблемы оказалась корректировка конфигурационных файлов инсталляционных пакетов.

Попытка задействовать преимущества технологии NVidia Optimus под операционной системой Linux в дистрибутивах Ubuntu 11.04, Kubuntu 11.04, Mint 11 также вызвала ряд проблем. Наиболее серьезной из них было отсутствие официальной поддержки данной технологии под этой операционной системой. Однако предлагается несколько различных открытых

проектов по реализации данной технологии для операционной системы Linux. Среди рассмотренных проектов одним из наиболее активно развивающихся является проект «Bumblebee». Однако на сегодняшний день данный проект по-прежнему далек от совершенства. Наиболее серьезным недостатком является отсутствие возможности автоматического переключения видеокарт во время работы, и для активации высокопроизводительной видеокарты «NVidia» необходимо вводить команду в терминале, передавая ей в качестве параметра запускаемую программу. Попытки хотя бы запустить графическую оболочку «KDE» с отрисовкой через видеокарту «NVidia» не увенчались успехом. Аналогичные Linux проблемы имеются и в DesktopBSD. Таким образом, становится очевидно, что данная проблема является распространенной среди множества программных платформ.

Другим примером существования проблем интеграции и взаимодействия программного обеспечения в современных вычислительных системах является попытка установки декоратора окон Emerald версии 0.7.2 для оконного менеджера Compiz в операционной системе Linux дистрибутив Ubuntu 11.04 с графической оболочкой Unity. Эксперимент проводился с 64-разрядной редакцией дистрибутива. Запуск данного декоратора окон завершался сообщением об ошибке «Segmentation fault» – ошибка сегментации. Декорации окон не прорисовывались, несмотря на то, что данный программный продукт находится в официальном репозитории данной редакции операционной системы Linux. Эта ошибка возникала как при попытке выполнить команду от имени обычного пользователя, так и суперпользователя (*root*). Эта проблема также обсуждалась на множестве форумов в сети Интернет. Иллюстрация этой проблемы приведена на рис. 3. Видно отсутствие заголовков окна терминала и музыкального проигрывателя.

Дополнительной сложностью при разработке современных крупных информационно-вычислительных комплексов является обеспечение обратной совместимости. Одним из наиболее простых примеров обеспечения обратной совместимости является поддержка чтения и записи старых форматов файлов программным продуктом. В подобном случае для осуществления данной особенности является необходимым, по сути, только сохранить модуль, отвечающий за сохранение и загрузку информации, а также сохранить корректное взаимодействие в новой версии программного обеспечения с указанным модулем. Примером с изменением формата файлов, хранящих данные, является Microsoft Office 2007, поддерживающий сохранение и чтение данных в старом формате Microsoft Office 97 – 2003. Также корпорация «Microsoft» предлагает пакет совместимости для «Microsoft Office» предыдущих версий, однако при конвертировании в старый формат документа зачастую меняется форматирование содержимого, что, скорее всего, связано с отсутствием определенного функционала в предыдущих версиях пакета Microsoft Office. Современный формат хранения документов Microsoft Office носит название Office Open XML [4].

Однако, как уже неоднократно упоминалось выше, современные информационно-вычислительные комплексы состоят из множества различного программного обеспечения, произведенного в большинстве случаев различными производителями. В подобной среде обеспечение обратной совместимости является не столь простым процессом, так как даже в простейшем случае изменения формата файлов хранения данных влечет за собой необходимость проведения изменений в остальном программном обеспечении, которому необходимо взаимодействовать с данными, хранящимися в новых форматах файлов. Примером тому может быть программный продукт «Open Office». В данном программном продукте производитель попытался обеспечить возможность корректной работы с файлами, созданными в Microsoft Office. Переход же Microsoft Office к новому формату хранения данных Office Open XML потребовал также необходимости внесения соответствующих изменений в Open Office. Кроме того, достичь полной совместимости без потерь в форматировании с Microsoft Office до сих пор так и не удалось.

Наименее затратными являются изменения файлов хранения данных, поскольку приводят к необходимости внесения соответствующих изменений только в те программные модули, которые требуют непосредственного взаимодействия с данными в соответствующих файлах, однако если рассмотреть ситуацию с изменением формата исполняемых файлов, то, очевид-

[4] http://ru.wikipedia.org/wiki/Office_Open_XML

но, в данном случае потребуется быстрейшее перекомпилирование наибольшего количества программных решений, используемых в данной информационно-вычислительной системе. Одним из примеров изменения форматов исполняемых файлов является замена формата a.out на формат ELF в операционной системе Linux, FreeBSD и многих других UNIX-подобных операционных системах, которая дала возможность хранить много секций в исполняемом файле [5]. Изменение формата исполняемых файлов в операционных системах также требует от разработчиков программного обеспечения в течение определенного времени поддерживать несколько версий разрабатываемого программного обеспечения для поддержания возможности работать с программными продуктами на компьютерных системах, использующих еще старые версии соответствующих операционных систем с форматом исполняемых файлов предыдущей версии, а также чтобы дать возможность использования этого программного обеспечения на уже обновленных информационно-вычислительных системах, что, возможно, также создает дополнительные сложности команде разработчиков. Аналогичный переход в форматах файлов наблюдался и в операционной системе Microsoft Windows при переходе от 16-битного формата исполняемых файлов MZ, используемого как основной формат исполняемых файлов в системе MS-DOS, NE, используемому в Microsoft Windows 3.x к 32- и 64-битному формату PE, используемому в современных версиях операционной системы Microsoft Windows, а также начиная с Microsoft Windows NT 3.5 и Microsoft Windows 95 [6].

Серьезную угрозу надежности функционирования операционной системы представляет изменение API, предназначенных для взаимодействия с драйверами, выполняющимися в нулевом кольце защиты операционной системы. Подобная несовместимость легко может вызвать невозможность функционирования операционной системы до устранения несовместимого драйвера из системы. Примером тому может быть эксперимент по установке программы Alcohol 120% версии 1.9.5.3105 в операционной системе Microsoft Windows Vista Ultimate 32 Bit. Во время установки происходил крах операционной системы на этапе установки драйвера, после чего операционная система перезагружалась и при запуске автоматически пыталась продолжить установку, которая также заканчивалась крахом системы. В результате нормальная работа с операционной системой была невозможна. Причиной данной проблемы были коренные изменения в модели построения драйверов, появившиеся в операционной системе Microsoft Windows Vista. Используемая ранее драйверная модель называлась WDM (Windows Driver Model), а в операционной системе Microsoft Windows Vista был совершен переход к новой драйверной модели под названием WDF (Windows Driver Foundation), имеющей коренные отличия от WDM [7].

Приведенные примеры несовершенства интеграции компонентов в общую систему являются не столь опасными в случае выхода из строя настольной компьютерной системы домашнего использования, однако возникновение подобного рода проблем на объектах нефтегазовой, транспортной, а в особенности атомной промышленности могут обернуться катастрофой и привести не только к потерям информационно-вычислительной системы в целом, но и к гибели множества людей в случае аварии на атомной электростанции. Одной из причин вышеперечисленных проблем современных компьютерных систем является сложность интеграции различных компонентов вычислительной системы в общую архитектуру. Множество различных аппаратных составляющих (например, видеокарты, материнские платы, жесткие диски и другие компоненты системы), а также различных программных платформ и решений, реализуемых различными производителями без тесного сотрудничества, создают ряд сложностей, препятствующих быстрому и надежному расширению всей вычислительной системы. В связи с этим важно пересмотреть существующую модель коммуникаций и методов интеграции элементов системы в общую архитектуру информационно-вычислительных комплексов, а также разрабатывать наиболее надежные и безопасные механизмы взаимодействия любых – как аппаратных, так и программных – элементов системы. Также является немаловажным усовершенствование принципов реализации обратной совместимости различных элементов информационно-вычислительных систем.

---

[5] http://www.freebsd.org/doc/ru/books/handbook/binary-formats.html

[6] http://ru.wikipedia.org/wiki/EXE

[7] Пенни Орвик, Гай Смит. Windows Driver Foundation: разработка драйверов. bhv. Русская редакция. 2008.

Заключение

В данной работе были рассмотрены основы технологии интерфейсов, приведена одна из возможных классификаций уровней взаимодействия элементов информационно-вычислительных систем. Кроме того, рассмотрены некоторые особенности модульной архитектуры построения современных информационно-вычислительных комплексов. Результатом проведенных исследований можно считать необходимость дальнейшего совершенствования механизмов интеграции модулей в архитектуру современных информационно-вычислительных систем для дальнейшего повышения надежности и безотказности функционирования элементов информационных систем.

Материал поступил в редколлегию 16.01.2012

A. S. Velizhanin, A. V. Revnivykh

THE MODULAR ARCHITECTURE OF A SPECIFIC FEATURE OF MODERN INFORMATION AND COMPUTING SYSTEMS. TECHNOLOGY INTEGRATION MODULES IN AN ENTIRE SYSTEM

The paper describes the modular architecture of the building of modern information and computing systems. Also in the paper have been described some features of the technology integration of modules into a common information-processing system that directly affect the reliability of a whole computers systems.

Keywords: information security, reliability, information systems, interface and integration.