УДК 004.4

ОБЩАЯ ПЛАТФОРМА ИСПОЛНЕНИЯ ПРИЛОЖЕНИЙ

И. А. Левашев

Алтайский государственный технический университет им. И.И. Ползунова,
г. Барнаул

Статья посвящена проблеме совместимости программ и библиотек на уровне машинных кодов, а также выработке подхода к решению этих проблем.

Ключевые слова: совместимость, переносимость, объектная модель.

Введение

В области информационных технологий так же, как и в других областях, наряду с опознанными потребностями существуют неосознаваемые гипотетические возможности, но их неосознанность приводит к тому, что реальные проблемы могут длительное время оставаться нерешёнными.

Современная деятельность во многих отраслях зависит от ПК и ПО для них, при этом существующие стандарты и спецификации на платформы исполнения далеки от того, чтобы покрывать спектр потребностей разработчика. Это приводит к тому, что ПО разрабатывается не по де юре стандартам (которых нет), а по де факто, к формированию неестественных гегемоний и привязки к поставщику.

Часто возникает ситуация, когда разработчик предлагает пользоваться Wine для исполнения своей программы на OS Linux. Фактически разработчик использует применяемую в OS Windows комбинацию WinAPI и PE/COFF как де факто стандарты разработки переносимых программ, несмотря на то, что они не создавались для этого. Другие стандарты и спецификации, такие, как POSIX, OpenStep и SOM, каждый взятый по отдельности, не достаточны для прикладной разработки, а стандарт, который бы опирался на них и не содержал пробелов, не разработан.

Такие средства разработки переносимых программ, как Java и сценарные языки программирования, оказались неприемлемы в достаточно широком диапазоне прикладного ПО. Разработчики вновь и вновь обращаются к языкам, транслируемым в машинные коды, поэтому необходимо провести стандартизацию именно на этом уровне.

Анализ проблем

Рассматриваемые проблемы совместимости программ и библиотек можно разделить на несколько групп:

– совместимость на уровне машинных кодов между разными версиями;

– совместимость интерфейсов библиотек между разными языками программирования;

– совместимость между программами и библиотеками машинными кодами для разных архитектур процессора;

– совместимость между программами и библиотеками, транслированными в машинные коды для одной архитектуры процессора, но разных операционных систем.

Пример проблемы из первой группы – так называемая проблема «хрупкого базового класса», но есть и другие. Представление об этих проблемах можно получить из [2]. Другой взгляд на эту проблему содержится в [1] – со стороны разработчиков, пользующихся инструментарием, никак не помогающим обеспечивать совместимость. Проблемы эти могут быть незнакомы малым коллективам разработчиков, но неизбежно проявляются в больших проектах. Эти проблемы нужно обязательно решить любым разработчикам операционных систем, в составе которых есть менеджер пакетов или магазин приложений, через которые распространяются транслированные в машинные коды программы и библиотеки.

Проблемы совместимости из второй группы в общем заключаются в отсутствии широко применимых межъязыковых стандартов, создавая интерфейсы библиотек на которых, можно было бы в автоматическом режиме получать программные интерфейсы для других языков программирования. Будучи

нерешёнными, проблемы из этой группы нередко вынуждают разработчиков выбирать для разработки язык программирования, руководствуясь не полезными особенностями языка программирования и компиляторов на нём, а тем фактом, на каких языках программирования разработчик сможет быстрее приступить к работе. Если существуют сразу две полезных библиотеки, но на разных языках программирования, то разработчикам требуется потратить время на то, чтобы интерфейс одной из библиотек сделать доступной из другого языка программирования.

Примеры решений проблем третьей группы можно подразделить на три подхода:

– эмуляция компьютера целиком: Mac OS Classic Environment на Mac OS X PowerPC и QEMU;

– эмуляция другой архитектуры в пределах одного процесса OS: Rosetta на Mac OS X Intel, qemu-user в Linux;

– избирательная эмуляция процессора только для компонент, транслированных в машинные коды для не родной процессору архитектуры: Intel libhoudini[5] на Intel Android.

При этом от первого к третьему снижается универсальность решения, но повышается общая производительность.

Проблемы четвёртой группы решаются, например:

– Wine — Windows–подобное окружение на POSIX–совместимых OS;

– HX DOS Extender — Windows–подобное окружение на OS DOS;

– Open32 — средство миграции и запуска Windows приложений на OS/2;

– CrossKylix — содержит средство запуска Linux приложения (Kylix) на OS Windows;

– Linuxulator — средство исполнения Linux приложений на FreeBSD.

Обзор стандартов и спецификаций

POSIX (англ. portable operating system interface for Unix — переносимый интерфейс операционных систем Unix) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов. Международная организация по стандартизации (ISO) совместно с Международной электротехнической комиссией (IEC) приняли данный стандарт (POSIX) под названием ISO/IEC 9945. POSIX стандартизирует не тот уровень абстракции, который требуется разработчику прикладных программ. Эту разницу можно понять на примере проблем с реализацией fork() в OS, не реализующей этот стандарт[4]. Кроме того, POSIX не стандартизирует технические подробности на уровне машинных кодов и объектных файлов, и объектные файлы для практически всех не имеющих друг от друга преемственности OS, реализующих POSIX, взаимно несовместимы.

OpenStep — объектно–ориентированный интерфейс программирования приложений (API) для объектно–ориентированных операционных систем, которые используют любую современную операционную систему в качестве ядра[4]. В отличие от POSIX, стандартизирует именно тот уровень абстракции, который требуется для разработки прикладного ПО. В отличие от таких библиотек, как Gtk+ и Qt, являющихся единственными реализациями своего интерфейса, OpenStep только на OS Windows был в том или ином виде реализован пятикратно: YellowBox, GNUStep, Cocotron, Apple Application Support, Project Islandwood. Это делает OpenStep привлекательным как основу для предлагаемого к разработке стандарта. В спецификации имеются те же пробелы, что и в POSIX: нет стандартов на уровне машинных кодов и формата объектных файлов. Кроме того, интерфейс описан на языке программирования Objective-C, на особенности компиляции которого нет общего стандарта. Как минимум 3 реализации взаимно несовместимы только из–за отличий в особенностях компилятора.

IBM System Object Model (SOM) — система динамических объектно–ориентированных программных библиотек. Не имеет формальной спецификации, но де факто стандарт мог бы стать частью предлагаемого к разработке стандарта. SOM мог бы заполнить пробелы, присутствующие в других спецификациях, а именно стандартизировать интерфейс на уровне машинных кодов. Кроме того, SOM позволяет сочетать разные языки программирования.

Предлагаемое устройство общей платформы исполнения приложений

Формат объектных файлов — PE/COFF, поддерживаемые наборы инструкций — x86(ia32) и x64(amd64). Стандарт может указывать поддерживаемое подмножество инструкций.

API платформы вынужденно включает в себя минимальное подмножество WinAPI, но основное API основано на современной версии OpenStep, такой, которая применяется в OS Mac OS X и реализуется GNUStep, при этом интерфейс Objective-C должен быть заменён на SOM или его модификацию.

Будучи полностью реализованной, такая платформа, например, могла бы позволить разработчикам, не имеющим ПК Эльбрус для тестирования своих программ, разработать их на OS Linux или Windows, а эти программы, будучи запущенными на ПК Эльбрус, использовали средства мультимедиа и другие критичные для производительности участки кода из библиотек, реализованных на родной для ПК Эльбрус архитектуре, а остальной код работал бы с небольшой потерей производительности в режиме эмуляции x86. Тем более такая платформа могла бы обеспечить безболезненный переход на OS Linux или другие OS на архитектурах x86/x64.

Заключение

Несмотря на общее стремление к стандартизации, ПО для ПК несколько последних десятков лет разрабатывается без стандартов, которые в полной мере помогали бы разработчикам разрабатывать совместимое и переносимое ПО. Чтобы ликвидировать привязки к поставщикам, необходим общий стандарт на платформу исполнения. Предлагается разработать такой стандарт и даны предложения как по реализации, так и по стандартизации такой платформы.

СПИСОК ЛИТЕРАТУРЫ

1. Binary Compatibility Issues With C++ [Электронный ресурс] // KDE TechBase. Berlin, 2014. URL: https://techbase.kde.org/Policies/Binary_Compatibility_Issues_With_C++ (дата обращения: 24.07.2015)

2. Forman I.R., Conner M.H., Danforth S.H., Raper L.K. Release-to-Release Binary Compatibility and the Correctness of Separate Compilation // OOPSLA ’95 Conference Proceedings. New York: ACM, 1995. P. 426–438. doi:10.1145/217838.217880

3. Highlights of Cygwin Functionality [Электронный ресурс]. 2015. URL: https://cygwin.com/cygwin-ug-net/highlights.html#ov-hi-process (дата обращения: 26.10.2015)

4. OpenStep Specification [Электронный ресурс]. 1994. URL: http://www.gnustep.org/resources/OpenStepSpec/OpenStepSpec.html (дата обращения: 26.10.2015)

5. Running ARM apps on Android x86 [Электронный ресурс]. 2014. URL: http://pilcrowpipe.blogspot.ru/2014/03/running-arm-apps-on-android-x86.html (дата обращения: 26.07.2015)

Левашев Иван Александрович – тел.: (902) 141 33 18, e-mail: octagram@bluebottle.com.