Для цитирования: Мустафин Т.Р., Алехин А.И., Куян А.С., Кравцунов Е.М., Семенихин С.В. Выбор дистрибутива программного обеспечения для индустриальных и бортовых систем, использующего технологию защищенных вычислений архитектуры «Эльбрус» // Вопросы радиоэлектроники. 2017. № 3. С. 44–47.
УДК 004.451.8

Т.Р. Мустафин[1,2], А.И. Алехин[1], А.С. Куян[1,2], Е.М. Кравцунов[1,2], С.В. Семенихин[1,2]

[1] АО «МЦСТ», [2] ПАО «ИНЭУМ им. И.С. Брука»

ВЫБОР ДИСТРИБУТИВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ИНДУСТРИАЛЬНЫХ И БОРТОВЫХ СИСТЕМ, ИСПОЛЬЗУЮЩЕГО ТЕХНОЛОГИЮ ЗАЩИЩЕННЫХ ВЫЧИСЛЕНИЙ АРХИТЕКТУРЫ «ЭЛЬБРУС»

Исследуется возможность реализации полного стека программного обеспечения на основе технологии защищенных вычислений архитектуры «Эльбрус». Подробно рассматриваются компоненты стека, проводится их выбор для дальнейшего переноса в защищенный режим исполнения программ. Описываются части и формы библиотеки uClibc и этапы переноса библиотеки в защищенный режим.

Ключевые слова: *защищенный режим исполнения программ, архитектура Эльбрус, библиотека uClibc, Buildroot.*

Введение

Защищенный режим исполнения программ, реализованный для архитектуры «Эльбрус», интересен возможностью выявлять ошибки в программах на этапе их исполнения. Это особенно важно для промышленных и бортовых систем, в которых программные ошибки могут привести к опасным последствиям. Для таких задач характерны сведенное к допустимому минимуму аппаратное обеспечение и использование конкретного функционала в программном обеспечении. Соответственно, структура дистрибутива, использующего технологию защищенных вычислений, в данном случае ограничивается ядром операционной системы с поддержкой защищенного режима исполнения программ, а также набором работающих в защищенном режиме приложений Linux для встраиваемых систем и небольшой реализацией стандартной библиотеки языка Си.

В этом контексте существенны некоторые особенности компонентов общего программного обеспечения (ОПО) 311–05, в настоящее время поставляемого пользователям с вычислительными комплексами семейства «Эльбрус»:

- Ядро операционной системы «Эльбрус» выполнено с поддержкой защищенного режима [1] и может быть использовано в составе формируемого дистрибутива.
- Входящая в состав ОПО 311–05 реализация *libmcst* стандартной библиотеки языка Си, которая поддерживает исполнение программ в защищенном режиме, не позволяет собрать набор программ Linux для самостоятельной загрузки и работы в защищенном режиме.
- Реализации *Glibc* стандартной библиотеки языка Си и набор приложений Linux на их основе работают не в защищенном режиме.

В связи с этим основными проблемами, которые решались при проведении данного исследования, были выбор инструмента автоматизированного создания набора приложений Linux, а также выбор и технология переноса на архитектуру «Эльбрус» стандартной библиотеки языка Си.

Инструмент сборки приложений Linux

Существует множество инструментов сборки Linux-окружения для встраиваемых систем. В качестве примера можно привести *Embedded Debian* [2], *Yocto Project* [3], *Buildroot* [4]. Сравнение их свойств показывает, что оптимальным может стать выбор *Buildroot* – это простой и эффективный инструмент создания встраиваемых Linux-систем,

использующий технологию кросс-компиляции. С его помощью можно собрать набор инструментов для кросс-компиляции, корневую файловую систему с приложениями Linux, ядро операционной системы Linux и системный загрузчик, а также любую комбинацию перечисленных компонентов. Кроме этого, *Buildroot* обладает простой и гибкой системой конфигурации получаемой корневой файловой системы – в меню настройки можно выбрать пакеты программ для конкретной цели. В процессе сборки *Buildroot* строит граф зависимостей пакетов, компилирует и устанавливает необходимые пакеты в образ целевой файловой системы. Все перечисленные свойства значительно упрощают перенос инструмента на новые архитектуры (в настоящее время *Buildroot* поддерживает 11 различных архитектур [4]). В то же время в полученном с помощью *Buildroot* программном обеспечении нет средств для установки новых пакетов, что в целом допускает его применение во встраиваемых системах.

Buildroot позволяет использовать в качестве стандартной библиотеки языка Си реализацию *Glibc* или *uClibc*. Так как дистрибутив предназначен для применения во встраиваемых системах, где аппаратные ресурсы, в т.ч. память, ограничены, то вследствие малого объема была выбрана библиотека *uClibc*.

Реализация стандартной библиотеки языка Си

uClibc (µ*Clibc*) – это библиотека языка Си для разработки встраиваемых систем Linux. Хотя ее объем намного меньше, чем у *Glibc*, почти все поддерживаемые *Glibc* приложения успешно работают с *uClibs*, а перенос программ обычно включает в себя только повторную сборку исходного кода. Библиотека *uClibc* реализует технологии разделяемых библиотек и исполнения программы в несколько потоков. В настоящее время она поддерживает архитектуры *alpha, ARM, cris, e1, h8300, i386, i960, m68k, microblaze, mips/mipsel, PowerPC, SH, SPARC* и *v850* [5].

Для использования библиотеки *uClibc* в сборке *Buildroot* ее архитектурно-зависимую часть необходимо реализовать в соответствии с архитектурой «Эльбрус». При этом учитывается только та функциональность, которая требуется для эксплуатации во встраиваемых системах. Она включает в себя все стандартные библиотечные функции, системные вызовы, поддержку статического и динамического связываний программ.

Статическое связывание

Статическая библиотека – это специальный архив, содержащий несколько объектных файлов, полученных компиляцией исходного кода библиотеки. В Linux статические библиотеки обычно имеют расширение *.a* (Archive). Статическая библиотека линкуется с программой на этапе ее компоновки. При этом весь объектный код библиотеки, который необходим для работы целевой программы, помещается в исполняемый файл этой программы.

Основное преимущество статического связывания состоит в том, что оно делает программу более автономной – программа может запускаться на любом компьютере, не требуя наличия на нем каких-либо библиотек (они уже «внутри» исполняемого файла). Но у статического связывания есть и существенные недостатки. Во-первых, весь исполняемый код, необходимый для работы программы, должен храниться внутри нее самой, что сильно увеличивает размер исполняемого файла. Во-вторых, при модернизации любой из библиотек необходимо заново перекомпилировать все программы, которые ее используют.

Использование статического связывания требует минимальных средств для создания программ с использованием библиотеки *uClibc* (сборка *uClibc*, поддерживающая только статическое связывание). При этом в *uClibc* необходимо реализовать несколько архитектурно зависимых частей:

- функцию *_start* на ассемблере, определяющую точку входа в программу. Этой функции через стек передаются аргументы командной строки программы (*argc*, *argv*) и указатель на пользовательскую функцию *main*(), которую та вызывает с передачей параметров *argc* и *argv*;
- специальные макросы, которые отвечают за передачу параметров и обращение к системным вызовам. Они архитектурно зависимы, т.к. механизмы обращения к системным вызовам из пользовательской программы на разных платформах отличаются. Параметры для системных вызовов, а также возвращаемые ими значения передаются через архитектурные регистры;
- задание порядка байтов, используемого данной архитектурой (*big endian* или *little endian*);
- задание направления роста стека (в сторону младших или старших адресов);
- задание размера машинного слова (32 или 64 бита);
- задание размера страницы оперативной памяти (обычно 4096 байт).

Динамическое связывание

Динамическая библиотека (или совместно используемая библиотека) – это файл, содержащий объектный код, предназначенный для совместного использования несколькими программами. В Linux динамические библиотеки обычно имеют расширение *.so* (Shared Object). Они подключаются к программе непосредственно в момент ее исполнения. При использовании динамического связывания код совместно используемой библиотеки

не включается в исполняемый файл программы – туда помещается только ссылка на требуемую библиотеку. За подключение совместно используемой библиотеки к исполняемому коду отвечает специальная программа – динамический загрузчик, запускаемый в момент исполнения программы. Ссылка на динамический загрузчик также содержится в исполняемом файле программы. В Linux динамический загрузчик обычно находится по пути */lib/ld.so*.

Динамическая компоновка имеет несколько существенных преимуществ по сравнению со статической. Во-первых, у динамически скомпонованной программы значительно меньший размер, чем у такой же программы, полученной при статической компоновке, т.к. исполняемый файл содержит только ссылку, а не весь код библиотеки. Во-вторых, загруженная библиотека может быть использована сразу несколькими программами, что экономит оперативную память. Существенно также, что в случае модернизации динамической библиотеки не нужно перекомпилировать программы, которые ее используют. Основным недостатком динамического связывания является то, что для исполнения полученной программы требуется наличие в системе всех библиотек, которые она использует.

При загрузке динамически связанной программы в память управление вначале передается динамическому компоновщику, который тоже загружается в адресное пространство этой программы. Он анализирует специальную секцию *.dynamic* исполняемого файла, содержащую информацию о том, какие совместно используемые библиотеки необходимы для работы данной программы, и загружает их в память. Так как динамически связанная программа содержит в себе только ссылки на необходимые для работы библиотеки, а их код загружается в память на этапе выполнения программы, то на момент запуска все адреса в инструкциях вызова функций, импортируемых из библиотек, указывают не на реальное расположение кода функции, а на специальную секцию под названием *.rel.dyn*. Она содержит в себе всю необходимую информацию о том, как нужно модифицировать адреса вызова импортируемых функций, чтобы они стали корректными. Поэтому при старте программы динамический компоновщик анализирует содержимое этой секции и проводит «релокацию» (повторное размещение) кода – связывание на этапе исполнения. После этого все адреса вызова функций станут корректными, и программа будет готова к выполнению.

Процесс «релокации» архитектурно зависим, т.к. инструкции вызова функций и формат адресов на различных аппаратных платформах разные. Поэтому для поддержки динамического связывания программ в библиотеке *uClibc*, портируемой на целевую архитектуру, необходимо реализовать все типы релокаций, используемых данной архитектурой.

Заключение

В статье рассмотрены компоненты дистрибутива программного обеспечения индустриальных и встраиваемых систем, использующие технологию защищенных вычислений архитектуры «Эльбрус»: *Buildroot* – инструмент сборки приложений Linux и *uClibc* – библиотека языка Си для разработки встраиваемых систем Linux. Описаны этапы портирования библиотеки *uClibc* на архитектуру «Эльбрус». В рамках данной работы для архитектуры «Эльбрус» реализована возможность создать корневую файловую систему приложений Linux с помощью *Buildroot*. Направлениями дальнейших исследований являются портирование библиотеки *uClibc* на архитектуру «Эльбрус», реализация поддержки данной библиотекой защищенного исполнения программ и сборка приложений Linux в защищенном режиме с помощью инструмента *Buildroot*.

СПИСОК ЛИТЕРАТУРЫ

1. Волконский В.Ю. Безопасная реализация языков программирования на базе аппаратной и системной поддержки // Вопросы радиоэлектроники. 2008. Т. 4. № 2. С. 98–141.
2. Embedded Debian Overview: Официальный сайт проекта Embedded Debian [Электронный ресурс]. URL: http://www.emdebian.org/about/
3. About Yocto Project: Официальный сайт проекта Yocto [Электронный ресурс]. URL: https://www.yoctoproject.org/about
4. The Buildroot user manual 2016.08: Официальный сайт проекта Buildroot [Электронный ресурс]. URL: https://buildroot.org/downloads/manual/manual.pdf
5. Andersen E. µClibc specifications: Официальный сайт проекта uClibc [Электронный ресурс]. URL: https://uclibc.org/specs.html

ИНФОРМАЦИЯ ОБ АВТОРАХ

Мустафин Тимур Рустемович, аспирант, ПАО «ИНЭУМ им. И.С. Брука»; инженер-программист 1-й категории, АО «МЦСТ», 119334, Москва, ул. Вавилова, д. 24, тел.: 8 (499) 135-33-21, e-mail: timur.r.mustafin@mcst.ru.

Алехин Андрей Игоревич, инженер-программист, АО «МЦСТ», 119334, Москва, ул. Вавилова, д. 24, тел.: 8 (499) 135-33-21, e-mail: andrey.i.alekhin@mcst.ru.

Куян Андрей Сергеевич, аспирант ПАО, «ИНЭУМ им. И.С. Брука»; старший инженер-программист АО «МЦСТ», 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-33-21, e-mail: andrey.s.kuyan@mcst.ru.
Кравцунов Евгений Михайлович, зам. начальника отделения, АО «МЦСТ», ПАО «ИНЭУМ им. И.С. Брука», 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-33-21, e-mail: evgeniy.m.kravtsunov@mcst.ru.
Семенихин Сергей Владимирович, д.т.н., профессор, зам. генерального директора, начальник отделения, АО «МЦСТ», ПАО «ИНЭУМ им. И.С. Брука», 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-20-61, e-mail: svs@mcst.ru.

For citation: Mustafin T.R., Alekhin A.I., Kuyan A.S., Kravtsunov E.M., Semenikhin S.V. Software distribution choice for industry and board systems uses security computing technology of «Elbrus» architecture. Voprosy radioelektroniki, 2017, no. 3, pp. 44–47.

T.R. Mustafin, A.I. Alekhin, A.S. Kuyan, E.M. Kravtsunov, S.V. Semenikhin

SOFTWARE DISTRIBUTION CHOICE FOR INDUSTRY AND BOARD SYSTEMS USES SECURITY COMPUTING TECHNOLOGY OF «ELBRUS» ARCHITECTURE

This research is aimed to feasibility of full software stack on security computing technology of «Elbrus» architecture. We consider on software stack components and make a component chosen for future ports to protected computing mode of «Elbrus» architecture. We also describe uClibc library's parts and library porting stages on protected computing mode.

Keywords: protected computing mode, Elbrus architecture, uClibc library, Buildroot.

REFERENCES

1. Volkonskiy V. Yu. Secure languages implementation on base of hardware and software assist. *Voprosy radioelektroniki,* 2008, vol. 4, no. 2, pp. 98–141 (In Russian).
2. Embedded Debian Overview. Embedded Debian Project website. Available at: http://www.emdebian.org/about/
3. About Yocto Project. Yocto Project website. Available at: http://www.emdebian.org/about/
4. The Buildroot user manual. Buildroot website, 2016, no. 2016.08. Available at: https://buildroot.org/downloads/manual/manual.pdf
5. Andersen E. µClibc specifications. uClibc project website. Available at: https://uclibc.org/specs.html

AUTHORS

Mustafin Timur, graduate student, PJSC «Brook INEUM»; Software Engineer 1st category, JSC «MCST», 24, Vavilova st., Moscow, 119334, Russian Federation, tel.: +7 (499) 135-33-21, e-mail: timur.r.mustafin@mcst.ru.
Alekhin Andrey, software engineer, JSC «MCST», 24, Vavilova st., Moscow, 119334, Russian Federation, tel.: +7 (499)135-33-21, e-mail: andrey.i.alekhin@mcst.ru.
Kuyan Andrey, graduate student, PJSC «Brook INEUM»; senior software engineer, JSC «MCST», 24, Vavilova st., Moscow, 119334, Russian Federation, tel.: +7 (499)135-33-21, e-mail: andrey.s.kuyan@mcst.ru.
Kravtsunov Evgeniy, deputy head of department, JSC «MCST», PJSC «Brook INEUM», 24, Vavilova st., Moscow, 119334, Russian Federation, tel.: +7 (499)135-33-21, e-mail: evgeniy.m.kravtsunov@mcst.ru.
Semenikhin Sergey, Dr., professor, deputy general director, head of department, JSC «MCST», PJSC «Brook INEUM», 24, Vavilova st., Moscow, 119334, Russian Federation, tel.: +7 (499) 135-20-61, e-mail: svs@mcst.ru.