Beeline_tech
31 авг в 19:35

История POSIX: путь к портируемому ПО

8 мин 2.9K

Блог компании билайн, Программирование*, C++*, История IT

Ретроспектива Перевод

Автор оригинала: vorakl

Как мы к этому пришли?

В ранние годы развития компьютеров программисты могли лишь мечтать о портируемости. Все программы писались непосредственно в машинном коде для каждой компьютерной архитектуры, на которой они должны были работать. Языки ассемблера с мнемоническими именами каждой команды CPU и другие удобства сильно упростили жизнь программистов, но программы по-прежнему были привязаны к архитектуре. Тогда ещё не изобрели операционных систем, поэтому программа не только управляла всей компьютерной системой, но и должна была инициализировать всю периферию, а также управлять ею. На самом деле, такие низкоуровневые программы реализовывали драйверы для каждого используемого ими устройства. И каждый раз, когда программу нужно было перенести на

оборудование с другой архитектурой, она в буквальном смысле переписывалась с учётом различий архитектуры набора команд CPU, структуры памяти и так далее.

Именно так произошло с Unix, который изначально был написан Кеном Томпсоном на языке ассемблера более пятидесяти лет назад. Первые версии Unix писались для платформы PDP-7, а для портирования его на PDP-11 нужно было переписывать код. Когда Дэннис Ритчи создал язык программирования C, и вместе с Томпсоном они переписали на нём основную часть кода Unix, внезапно оказалась возможной портируемость ПО. Тому были две главные причины. Во-первых, код, написанный на языке высокого уровня, не зависит от платформы, потому что компиляторы транслируют его в язык ассемблера целевой архитектуры. Это ещё важнее для целевых платформ на основе процессоров RISC, так как они требуют написания гораздо большего количества ассемблерных команд, чем процессоры CISC. Даже при портировании Unix на другую платформу основная сложность заключалась лишь в адаптации зависящих от архитектуры частей кода. С другой стороны, сама операционная система абстрагирует все особенности оборудования от пользовательской программы.

Программистам не нужно реализовывать многозадачность, управление памятью и драйверы для используемых ими устройств, потому что всё это часть ядра ОС и работает в адресном пространстве ядра. Пользовательские программы работают в пользовательском адресном пространстве и получают доступ ко всем предоставляемым ОС функциям при помощи интерфейса системных вызовов. В ОС реального времени, например, в Zephyr OS ситуация немного отличается, но принцип изоляции и защиты памяти для пользовательских программ сохраняется. Это приводит к двум выводам:

- *Пользовательские программы становятся портируемыми, если они написаны на высокоуровневом языке программирования для конкретной ОС*. При соблюдении обоих требований программы компилируются в команды целевого CPU и компонуются с системными функциями, предоставляемыми libc и относящимися к ОС библиотеками, для получения доступа к оборудованию.
- Портируемость обеспечивается на **уровне исходного кода**.

Рождение POSIX

Это могло быть концом истории, но случилось нечто судьбоносное. Из-за юридических ограничений компания AT&T не имела права продавать Unix, а значит, и зарабатывать на новорождённой ОС, которая завоёвывала всё большую популярность. Однако оказалось, что можно распространять Unix любым заинтересованным организациям по цене носителя. Так Unix попал в 1974 году в Беркли и во множество других мест, приведя к созданию различных производных ОС. Одни из самых известных и по-прежнему популярных ОС основаны на ПО, распространявшемся Беркли (BSD), например, FreeBSD и OpenBSD. Несмотря на общих предков и принципы, каждая операционная система пошла по собственному уникальному пути.

Каждая из этих ОС имела уникальный интерфейс (API) и реализацию подсистем ядра, системных вызовов, системных инструментов и так далее. Даже libc, предоставляющая общую функциональность и обёртки поверх системных вызовов, была очень привязана к ОС. Все эти ОС походили на Unix, но в то же время было невозможно взять исходный код программы, написанной для одной ОС, и перекомпилировать его на другой.

Более чем тридцать пять лет назад эти проблемы с портируемостью ПО привели к появлению в 1988 году первого стандарта POSIX. Эту аббревиатуру придумал Ричард Столлман, добавивший в X в конец *Portable Operating System Interface*. Сейчас торговой маркой *POSIX*™ владеет IEEE, а *UNIX*® — это зарегистрированная торговая марка The Open Group. Она должна предоставлять спецификацию интерфейса, общего для различных операционных систем Unix, в том числе языков программирования и инструментов. Важно отметить, что **портируем интерфейс**, а не реализация.

Это стало общей платформой, позволившей компилировать один и тот же исходный код пользовательской программы для любой ОС без модификаций при условии соблюдения обеими сторонами одного стандарта. В определённой степени это справедливо и сегодня, так как большинство современных популярных Unix-подобных систем, например, Linux и *BSD не соответствует полностью и строго стандарту POSIX, а использует его как руководство. Кроме POSIX существует Single UNIX Specification (SUS), в 2001 году консолидированная с несколькими стандартами POSIX. Однако последняя версия SUS (SUSv4 2018) расширяет спецификацией X/Open Curses последний стандарт POSIX (POSIX.1-2017), который, по сути, служит его базовой спецификаций.

Существует множество операционных система наподобие MacOS, полностью совместимых со стандартами POSIX и SUS, проходящих тесты совместимости The Open Group, а потому имеющих право называться операционными системами Unix, а не просто Unix-подобными. Изначально POSIX создавался только для Unix-подобных ОС, но со временем стал настолько популярным, что его спецификацию в виде Operating System Abstraction Layer (OSAL) частично реализовали (некое подмножество интерфейса, применимое к целевой системе) в несвязанных с Unix операционных системах, например, в Windows, FreeRTOS, Zephyr и так далее.

Спецификация POSIX

Самый первый стандарт был ратифицирован IEEE в 1988 году под названием IEEE Std 1003.1-1988, поэтому он называется *POSIX.1-1988*. С тех пор стандарт претерпел множество ревизий, а разные подмножества спецификации ратифицировались под разными названиями. Например, *POSIX.1-1990* (IEEE 1003.1-1990) определяет *системный интерфейс и среду вычислений*, *POSIX.2* (IEEE Std 1003.2-1992) определяет *язык команд (шелл) и инструменты* и так далее. Очень хорошее и краткое описание ревизий стандарта можно найти на странице man Linux standards(7). Можно даже найти ссылки на некоторые старые ревизии, например, на POSIX.2, изучая исходный код Bash. В 2001 году POSIX.1, POSIX.2 и

Single UNIX Specification (SUS) были объединены в общий документ под названием *POSIX.1-2001*. Несмотря на запутывающее название, на самом деле он включает в себя спецификации шелла и инструментов из POSIX.2. **Последней версией стандарта является POSIX.1-2017**, также известный как IEEE Std 1003.1-2017; он почти полностью идентичен POSIX.1-2008.

Документ стандарта, по сути, описывает спецификацию, распространяющуюся на два окружения (среду сборки и среду исполнения); он представлен в виде нескольких томов:

- Базовые определения: определяет общие для всех томов термины и концепции, требования совместимости (символьные константы, опции, группы опций), среду вычислений (локали, регулярные выражения, структуру папок, tty, переменные окружения и так далее) и файлы заголовков языка C, которые должны реализовываться соответствующими стандарту системами.
- Системные интерфейсы: определяет стандарт языка C (ISO C99, ISO/IEC 9899:1999), функции системных сервисов и расширение стандартной библиотеки C (libc) относительно файлов заголовков и функций.
- Шелл и утилиты: определяет интерфейс уровня исходного кода для Shell Command Language (sh) и системных утилит (awk, sed, wc, cat, ...), в том числе поведение, параметры командной строки, статусы выхода и так далее.
- Аргументация: включает в себя вопросы портируемости, субпрофилирования, групп опций и дополнительную аргументацию, не подходящую ко всем остальным томам.

Текущий стандарт POSIX определяет совместимость уровня исходного кода только для двух языков программирования: *языка C (C99)* и *языка команд шелла*. Однако некоторые из программ, определённые в «Утилитах», например, awk, также имеют свой собственный язык. Строго говоря, стандартная библиотека C (libc) не обязана реализовывать никакой дополнительной функциональности (функции и заголовки), не определённой стандартом C (в данном случае ISO C99), но большинство из них это делают. Например, стандарт ISO C99 определяет 24 файла заголовков, включая математические функции (`math.h`), стандартный ввод-вывод (`stdio.h`), дату и время (`time.h`), управление сигналами (`signal.h`), операции со строками (`string.h`) и так далее. Однако последний стандарт POSIX определяет 82 файла заголовков и, будучи полностью совместимым с ISO C99, расширяет его потоками POSIX (`pthreads.h`), семафорами (`semaphore.h`) и многим другим.

Кроме того, современные реализации libc, например, musl libc, тоже сильно привязаны к ОС, представляя библиотечные функции для доступа к сервисам операционных систем (обёртки для системых вызовов). Иногда пересечение со спецификациями POSIX приводит к возникновению трудностей в реализации слоя приложений POSIX в несвязанных с Unix операционных системах, использующих портируемые автономные реализации libc с собственной поддержкой POSIX, например, picolibc вместе с POSIX-библиотекой Zephyr.

Опции и группы опций

Хотя POSIX стандартизирует системный интерфейс (заголовки и функции языка C), шелл и утилиты, необязательно следовать всей спецификации, чтобы быть совместимым с POSIX. Некоторые фичи в «Системных интерфейсах POSIX», «Шелле и утилитах POSIX» и «Системных интерфейсах XSI» опциональны. Файл заголовка unistd.h содержит определения *стандартных символьных констант* для опций, отражающих отдельную фичу, и для групп опций, определяющих множество взаимосвязанных функций или опций. Имена групп опций, в отличие от опций, обычно не начинаются с символа подчёркивания. Совместимые с POSIX (POSIX Conformant) системы должны реализовывать и поддерживать множество обязательных опций с одной или несколькими дополнительными опциями. Символьные константы для обязательных опций должны иметь определённые значения, например, *200809L*, в то время как другие опции могут быть

- *неопределёнными или содержать -1*; это означает, что опция не поддерживается для компиляции
- *0*; это означает, что опция может как поддерживаться, так и не поддерживаться в среде исполнения
- *каким-то другим значением*; это означает, что опция поддерживается всегда

Эти символьные константы используются пользовательскими приложениями для проверки доступности конкретных фич. На уровне исходного кода C константы могут проверяться или во время сборки (в директивах препроцессинга `#if`), или в среде исполнения вызовом одной из следующих функций: *sysconf()*, *pathconf()*, *fpathconf()* или *confstr(3)*. В исходном коде шелла для проверок в среде исполнения должна использоваться утилита getconf. Очень хорошую коллекцию опций POSIX вместе с соответствующими именами для использования в качестве параметров sysconf(3), а также список файлов заголовков и функций, представляющих эти опции, можно найти на странице man Linux posixoptions(7).

Группы опций субпрофилирования предназначены для использования в системах, где реализация полной спецификации POSIX не имеет смысла. Например, встроенные системы реального времени обычно имеют ограничения по ресурсам, поэтому не содержат шеллов, интерфейсов пользователя, а ядра ОС часто проектируются так, чтобы выполняться как единый процесс (с множественными потоками). Такие системы могут реализовывать только подмножества соответствующих функций, определяемых группами опций.

Подведём итог

- Развитие высокоуровневых языков программирования наподобие C наряду с операционными системами, абстрагирующими аппаратные подробности, обеспечили портируемость ПО на уровне исходного кода.

- В 1988 году появился стандарт POSIX, целью которого было создание спецификации портируемого интерфейса для Unix-подобных операционных систем, что позволило компилировать программы под разные платформы.
- Со временем стандарт POSIX эволюционировал; его последней версией стал POSIX.1-2017 (IEEE Std 1003.1-2017).
- Современные Unix-подобные системы наподобие Linux и *BSD соответствуют стандарту POSIX не строго, а используют его в качестве руководства.
- POSIX стандартизирует C API (файлы заголовков и функции), шелл и утилиты.
- От совместимых с POSIX систем ожидается реализация обязательных опций и возможная поддержка дополнительных опциональных фич.
- Приложения могут проверять наличие фич POSIX и во время компиляции, и в среде исполнения при помощи символьных констант и системных функций.
- Для систем с ограниченными ресурсами, например, для встроенных платформ реального времени, POSIX позволяет реализовывать подмножества полной спецификации посредством «субпрофильных» групп опций.

Теги: POSIX, история, архитектура, программирование

Хабы: Блог компании билайн, Программирование, C++, История IT

билайн

Компания

Сайт Сайт

50 Карма 7.1 Рейтинг

@Beeline_tech

Пользователь

Подписаться

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

artyomsoft
22 часа назад

Как я пришёл к пониманию основ создания Live-дистрибутивов Linux, решив починить свой старый SSD

Средний 28 мин 9.1K

Туториал

+51 84 7

slava_rumin
16 часов назад

Вспомнил школьную физику, и теперь продаю горячий воздух на Авито на 12 млн в год

Простой 10 мин 29K

Интервью

+50 35 31

andres_kovalev
19 часов назад

Особенности Effector, которые почему-то никто не обсуждает: опыт ВКонтакте спустя год использования

Средний 31 мин 4.2K

Мнение

+39 37 11

HotReset
21 час назад

Чем убили Фидо

2 мин 6.7K

Дайджест

+36 9 79

mikhailsudakov
вчера в 10:00

YRGB 2024 — конкурс по созданию игр для ZX Spectrum

6 мин 2.7K

+36 27 26

ru_vds
18 часов назад

Как новичку поучаствовать в устранении багов Google Chrome

Средний 11 мин 1.7K

Кейс Перевод

+30 23 0

sobolevn
15 часов назад

Проблемы вызова Python кода из C кода

Сложный 5 мин 1.9K

Ретроспектива

+26 26 0

BiktorSergeev
23 часа назад

RS/6000 SP: суперкомпьютер IBM, обыгравший Каспарова. Что это была за система?

4 мин 3.6K

+22 10 6

DimDimDimDimDim
20 часов назад

Почему нейросети становятся угрозой для природы и что с этим сделать

8 мин 2.9K

+19 15 16

dolovar
15 часов назад

Что не так со статьями о выгорании

Средний 26 мин 3.3K

Мнение

+18 52 27

Показать еще

ВАКАНСИИ КОМПАНИИ «БИЛАЙН»

Архитектор решений (Продукт ЛКР)
beeline · Москва · Можно удаленно

Тестировщик Manual QA
beeline · Москва

Ведущий Devops инженер
от 300 000 ₽ · beeline · Москва · Можно удаленно

Ведущий Golang разработчик (adtech)
от 300 000 ₽ · beeline · Москва · Можно удаленно

Координатор аварийно-восстановительных работ
beeline · Москва · Можно удаленно

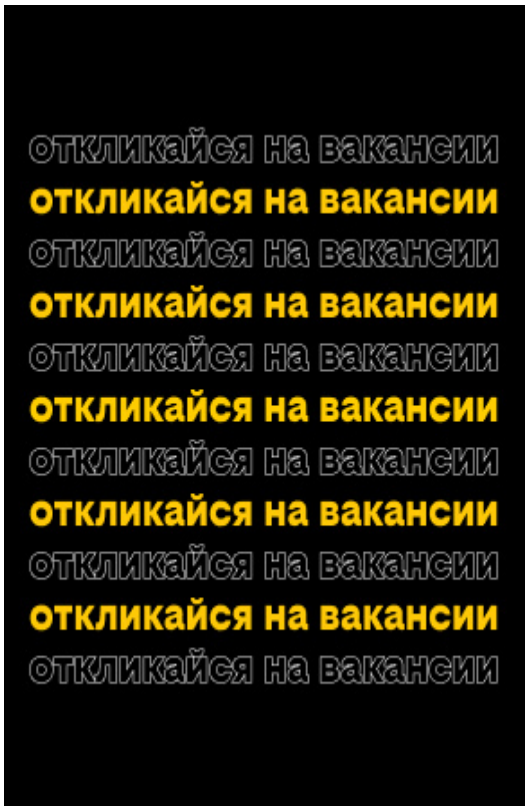
Больше вакансий на Хабр Карьере

ИНФОРМАЦИЯ

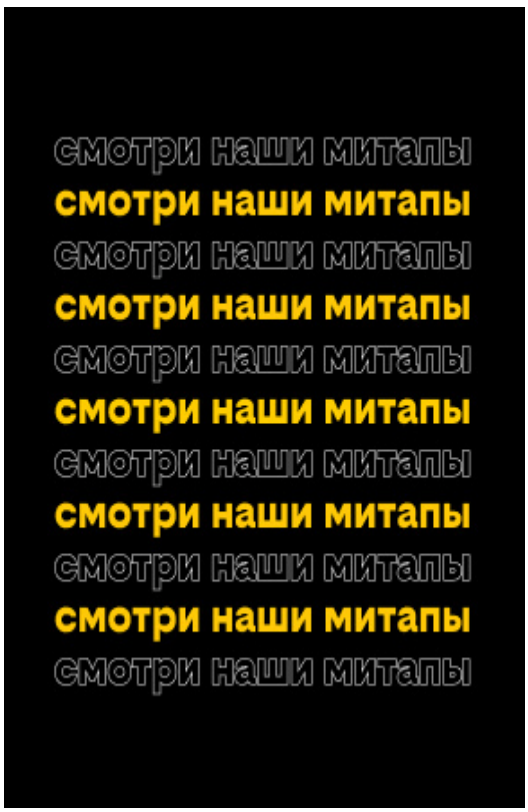
Сайт	beeline.ru
Дата регистрации	27 февраля 2023
Дата основания	2 июня 1992
Численность	свыше 10 000 человек

Местоположение Россия

ВИДЖЕТ



ВИДЖЕТ



БЛОГ НА ХАБРЕ

31 авг в 19:35

История POSIX: путь к портируемому ПО

2.9K 5

23 авг в 22:23

Попытка разогнать сеть для БД со 100 до 200Гб/с или «failure is always an option»

11K 4

16 авг в 13:49

Как искусственный интеллект помогает лечить рак почек. Патология, диагностика, прогноз (часть 2)

42K 1

8 авг в 16:22

Как искусственный интеллект помогает лечить рак почек. Патология, диагностика, прогноз

18K 8

24 июл в 15:05

Строим свой WYSIWYG с помощью LexicalJs

3.2K 10

Настройка языка

Техническая поддержка