Крос платформене програмування

викладач Канцедал Г.О.

Tg @Heorhii_Kantsedal

Що це таке і з чим його ідять ?

- Кросплатформність (або багатоплатформність, мультиплатформність) — властивість програмного забезпечення працювати більш ніж на одній програмній (в тому числі — операційній системі) або апаратній платформи, та технології, що дозволяють досягти цієї властивості.
- Основна мета кроссплатформного підходу — забезпечити максимальне охоплення аудиторії, економлячи час та ресурси на розробку.

Види кросплатформеності:

- кросплатформність на рівні мов програмування (а також інструментів таких мов: компіляторів та редакторів зв'язків);
- кросплатформність на рівні середовища виконання;
- кросплатформність на рівні операційної системи;
- кросплатформність на рівні апаратної платформи.

Кросплатформність на рівні мов програмування

- Кросплатформність на рівні мов програмування досягається шляхом забезпечення незалежності програмного коду від платформи. Багатоплатформними є більшість сучасних високорівневих мов програмування, для яких реалізовані транслятори, що можуть виконуватись на різних платформах. Наприклад, C, C++ і Pascal — кросплатформні мови на рівні компіляції, тобто для цих мов є компілятори під різні платформи.
- Кросплатформність на рівні редакторів зв'язків досягається реалізацією для різних платформ кросплатформних бібліотек, які реалізують незалежний від платформи інтерфейс, в тому числі — стандартизованих бібліотек. Зокрема, у наборі POSIX (Portable Operating System Interface for uniX) стандартизовано багато бібліотек мови C. Існує також велика кількість нестандартних кросплатформних бібліотек: Qt, GTK+, FLTK, STL, Boost, OpenGL, SDL, OpenAL, OpenCL.

Кросплатформність на рівні середовища виконання

- Кросплатформність на рівні середовищ виконання забезпечується реалізацією в цих середовищах можливостей, необхідних програмам незалежно від платформи. Декларований набір таких можливостей прийнято називати «контрактом» — обов'язком який покладається на середовище, щоб забезпечити виконання програми. Ці обов'язки реалізуються через інтерпретатор, файлові потоки, системні виклики, протоколи, віртуальну машину тощо.
- Java і C# — кросплатформні мови на рівні виконання, тобто їх виконувані файли можна запускати на різних платформах без попередньої перекомпіляції. PHP, ActionScript, Perl, Python, Tcl і Ruby — кросплатформні інтерпретовані мови, їх інтерпретатори існують для багатьох платформ.

Кросплатформність на рівні мов програмування

- Кросплатформність на рівні мов програмування досягається шляхом забезпечення незалежності програмного коду від платформи. Багатоплатформними є більшість сучасних високорівневих мов програмування, для яких реалізовані транслятори, що можуть виконуватись на різних платформах. Наприклад, C, C++ і Pascal — кросплатформні мови на рівні компіляції, тобто для цих мов є компілятори під різні платформи.
- Кросплатформність на рівні редакторів зв'язків досягається реалізацією для різних платформ кросплатформних бібліотек, які реалізують незалежний від платформи інтерфейс, в тому числі — стандартизованих бібліотек. Зокрема, у наборі POSIX (Portable Operating System Interface for uniX) стандартизовано багато бібліотек мови C. Існує також велика кількість нестандартних кросплатформних бібліотек: Qt, GTK+, FLTK, STL, Boost, OpenGL, SDL, OpenAL, OpenCL.

Кросплатформність на апаратному рівні

Кросплатформність на апаратному рівні досягається реалізацією однакових машинних команд та форматом їх представлення, систем переривань, механізмів адресації пам'яті, регістрів тощо. Може досягатись шляхом віртуалізації відповідних ресурсів та механізмів.

Переваги кросс платформного програмування

- **Збільшення аудиторії та доступності:**
  Один додаток для всіх платформ розширює охоплення користувачів, які можуть користуватися додатком як на мобільних, так і на настільних пристроях.
- **Зменшення витрат на розробку та підтримку:**
  Підхід «напиши один раз, використай всюди» знижує витрати на розробку та тестування окремих версій для різних операційних систем.
- **Швидке оновлення та підтримка:**
  Кроссплатформні додатки дозволяють розробникам вносити зміни та оновлення лише один раз, що забезпечує узгодженість і зручність підтримки.
- **Єдиний користувацький досвід (UX):**
  Забезпечує однаковий інтерфейс і функціональність на різних платформах, що підвищує зручність і передбачуваність роботи для користувача.
- **Зменшення часу на тестування**

Стратегії розробки кросплатформного програмного забезпечення

Для розробки ПЗ, яке поєднує у собі кросплатформність та складну функціональність, запропоновано значну кількість стратегій, до складу яких належить:

- Поступове зменшення можливостей (Graceful degradation)

Поступове зменшення можливостей — це властивість ПЗ, яка полягає у тому, що ПЗ надає однакову або схожу функціональність користувачам усіх платформ, зменшуючи можливості до якогось спільного мінімуму, який досягається для систем із найбільшими обмеженнями (наприклад застарілих версій клієнтських браузерів). Прикладом є перемикання сайту Gmail у базовий режим роботи (basic mode) з обмеженими можливостями, що має місце на браузерах з неповною підтримкою сучасних web-технологій.

Стратегії розробки кросплатформного програмного забезпечення

- Множинні кодові репозитарії (Multiple codebases)

Множинні кодові репозитарії — стратегія, заснована на використанні окремих сховищ скомпільованого або вихідного коду для різних (апаратних чи/та програмних) платформ, які мають однакову функціональність. Ця стратегія часто зумовлює наявність великого числа дублікатів спільних фрагментів коду для кожної платформи, до кожного із яких додається платформно-орієнтований код. Ця техніка часто застосовується на практиці при розробці та постачанні настільних програмних систем та комплексів.

- Єдиний кодовий репозитарій (Single codebase)

Єдиний кодовий репозитарій — стратегія, заснована на використанні єдиного сховища скомпільованого або вихідного коду для усіх платформ. При цьому спільний для усіх платформ код не повторюється, а блоки коду, призначені лише для однієї платформи, описані як вибіркові або умовні (conditional) і інтерпретуються, компілюються або виконуються лише у разі потреби. У межах стратегії Single codebase також допускається використання техніки відокремлення функціональності або компонент програмної системи, що дає змогу приховувати від користувача ту функціональність, яку не підтримується у користувача (наприклад, на рівні операційної системи чи браузера), зберігаючи усю іншу функціональність та забезпечуючи при цьому робочий стан програмної системи

Стратегії розробки кросплатформного програмного забезпечення

- Бібліотеки сторонніх розробників (Third-party libraries)

Бібліотеки сторонніх розробників часто надають засоби по забезпеченню кросплатформності у тих випадках, коли її підтримка у оригінальній версії програмного продукту є відсутньою або недостатньою. Часто це робиться шляхом використання загального API, яке приховує від клієнтів деталі реалізації, які враховують особливості кожної платформи.

- Чутливий дизайн (Responsive design)

Чутливий дизайн — підхід до дизайну візуального інтерфейсу програмного продукту, який є оптимальним з точки зору зовнішнього сприйняття. Перегляд та навігація по вікнам програми супроводжується мінімумом змін розміру, панорамувань (panning), скролінгу для широкого кола пристроїв від мобільних пристроїв до настільних комп’ютерів. Ця стратегія найчастіше застосовується для розробки web-додатків і тому часто називається Responsive web design (RWD)

Скрипти та інтерпретовані мови

- ПЗ, розроблене у вигляді набору скриптів може розглядатися як кросплатформне, якщо існують версії інтерпретатора, які працюють на різних платформах. Наприклад, скрипт, написаний на мові Python для Unix-подібної системи буде (у більшості випадків або із незначними змінами) працювати на базі Windows, тому що існують реалізації Python для Windows.
- Те саме стосується багатьох інших відкритих мов програмування скриптового типу. На відміну від бінарних виконуваних файлів, ті самі скрипти можуть використовуватися на машинах із різною архітектурою, на яких встановлені інтерпретатори відповідних скриптів

Підходи до кросплатформного програмування

- Історично першим був підхід, який заснований на створенні різних версій тієї самої програм для різних платформ. Іншими словами, Windows-версія програми має один набір вихідних файлів, Macintosh-версія — інший і т.д. Цей підхід у ідейному плані є найпростішим, але на практиці він вважається більш витратним з огляду на вартість розробки та час розробки. Основна ідея — розробка двох і більше різних програм, які поводять себе однаково.
- Другий підхід — використання спеціального ПЗ, яке приховує або згладжує відмінності, які існують між різними платформами. У межах цього підходу програма «не знає» про платформу, на якій вона працює (platform agnostic). Прикладом ПЗ, розробленого у межах цього підходу є програми, призначені для виконання на віртуальній машині Java (JVM).

Основні мови для кросс платформного програмування

- **JavaScript:**

Популярна мова для веб-розробки, підтримується фреймворками, такими як React Native, які дозволяють створювати мобільні додатки для iOS та Android.

- **Dart:**

Використовується в Flutter — сучасному фреймворку для створення мобільних, веб і настільних додатків з нативною продуктивністю та інтерактивним дизайном.

- **C#:**

Мова програмування для Windows, яка активно використовується у фреймворку Xamarin для розробки мобільних додатків з можливістю поширення на різні платформи.

- **Python:**

Мова з великим обсягом бібліотек, таких як Kivy та PyQt, що забезпечують кроссплатформну підтримку для графічних інтерфейсів додатків.

- **Kotlin:**

Мова, розроблена для Android, з підтримкою кроссплатформного підходу через Kotlin Multiplatform, що дозволяє створювати загальний код для Android, iOS та вебу.

JavaScript для кроссплатформного програмування

- **JavaScript:**
  Мова, яка є базовою для веб-розробки, дозволяє створювати інтерактивні елементи веб-сторінок і широко підтримується для кроссплатформної розробки.
- **React Native:**
  Фреймворк, що працює на JavaScript, призначений для створення кроссплатформних мобільних додатків для iOS та Android. Це один із найпопулярніших інструментів для кроссплатформного програмування, який використовують компанії, як-от Facebook, Instagram і Airbnb.
- **Electron:**
  Інструмент для створення настільних додатків на основі JavaScript, використовується для таких додатків, як Visual Studio Code і Slack. Electron дозволяє створювати додатки для Windows, macOS та Linux.

Переваги JavaScript

- Велика спільнота розробників, доступ до великої кількості бібліотек та плагінів.
- Швидка розробка завдяки знайомим інструментам веб-розробки.
- Можливість легкого створення додатків для кількох платформ з єдиним кодом.

Dart + Flutter

Dart - мова програмування, розроблена Google, яка оптимізована для роботи з Flutter. Відзначається високою продуктивністю та підтримкою багатих UI-компонентів.
Flutter - сучасний фреймворк, що дозволяє розробникам створювати додатки для Android, iOS, вебу та десктопу. Пропонує готові компоненти Material Design і Cupertino, що дає змогу інтегруватися з Android та iOS, створюючи привабливий і нативний дизайн.
Додатки розроблені задопомогою дарт: Google Ads, Alibaba, BMW

Переваги Dart

- Багато віджетів для швидкої розробки UI.
- Висока продуктивність додатків, наближена до нативних, завдяки роботі на графічному рушії.
- Швидкий цикл розробки та можливість «гарячого перезавантаження» для миттєвого перегляду змін у коді.

C# + Xamarin

C# - основна мова для розробки під Windows, використовується також для кроссплатформної розробки через Xamarin.
Xamarin - фреймворк для мобільної розробки від Microsoft, що дозволяє писати код для Android та iOS, використовуючи C#. Містить Xamarin.Forms для створення єдиного коду інтерфейсу, що допомагає зменшити час на розробку та підтримку додатків

Переваги C#

- Інтеграція з екосистемою Microsoft, включно з Azure та Office.
- Доступ до API пристрою, що дозволяє працювати з камерою, GPS та іншими функціями.
- Сумісність з Visual Studio, що полегшує процес розробки

Python для кроссплатформної розробки

Python - легка у вивченні мова, яка підтримує кроссплатформну розробку завдяки таким бібліотекам, як Kivy та PyQt.

Kivy - бібліотека для створення мобільних і настільних додатків з графічним інтерфейсом. Працює на Windows, macOS, Linux, Android та iOS.

PyQt - інструмент для створення настільних додатків з графічним інтерфейсом. Підтримує Windows, macOS і Linux, надаючи можливість створювати потужні додатки з інтерфейсом користувача.

(наукові інтерфейси для експериментів і тд)

Python для кроссплатформної розробки

Python - легка у вивченні мова, яка підтримує кроссплатформну розробку завдяки таким бібліотекам, як Kivy та PyQt.

Kivy - бібліотека для створення мобільних і настільних додатків з графічним інтерфейсом. Працює на Windows, macOS, Linux, Android та iOS.

PyQt - інструмент для створення настільних додатків з графічним інтерфейсом. Підтримує Windows, macOS і Linux, надаючи можливість створювати потужні додатки з інтерфейсом користувача.

(наукові інтерфейси для експериментів і тд)

Популярність мов програмування

Language	Community Size (in millions)	Popular Use	Least Popular In...
JavaScript	17.4	Apps, Web	DS/ML, Embedded
Python	15.7	DS/ML, IoT	Mobile, Web
Java	14.0	Cloud, Mobile	DS/ML, Web
C/C++	11.0	Embedded, IoT	Web, Cloud
C#	10.0	Desktop, Games	DS/ML, Mobile
PHP	7.9	Web, Cloud	DS/ML, Mobile
Kotlin	5.0	Mobile, AR/VR	DS/ML, Desktop
Visual Development Tools	5.0	AR/VR, Desktop	Cloud, Web
Swift	3.5	Mobile, AR/VR	Cloud, Embedded
Go	3.3	Cloud, Apps	Mobile, DS/ML
Objective-C	2.4	AR/VR, IoT	Desktop, Apps
Rust	2.2	AR/VR, IoT	Web, Mobile
Ruby	2.1	IoT, Apps	Web, Embedded
Dart	1.8	Mobile, Apps	Web, Apps
Lua	1.4	Games, IoT	Mobile, Embedded

Логічним висновком було б зупинитись на джава скрипті як найбільш популярним, однак ми зупинимось на вивчані пайтону та котліну. Див переваги котліну в наступних презентаціях

End…