1. Основні поняття кросплатформного програмування

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*.

Кросплатформність дозволяє суттєво скоротити витрати на розробку нового або адаптацію існуючого програмного забезпечення. Залежно від засобів реалізації кросплатформність поділяється на

- 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.

Кросплатформне програмне забезпечення (ПЗ) може бути поділено на дві категорії:

1) ПЗ, яке потребує індивідуальну компіляцію для кожної платформи, під яку воно розроблено;
2) ПЗ, яке може виконуватися на довільній платформі без спеціальної підготовки, тобто, ПЗ написане на інтерпретуючій мові (яка може підтримувати попереднє створення переносимого байт-коду), інтерпретатори або середовища виконання якої є стандартними компонентами або підтримуються більшістю сучасних платформ.

Наприклад, Microsoft Windows на машина з архітектурою x86, Linux на машинах із тією самою архітектурою та macOS на x86-based Apple Macintosh системах. Кросплатформне ПЗ може працювати як на усіх існуючих системах, так і на лише деяких із них (щонайменше на двох).

Кросплатформне програмування — практика написання програм, які можуть працювати більше ніж на одній платформі.

2. Поняття платформи

У комп'ютерних науках термін «платформа» є багатозначним і використовується у різноманітних контекстах:

1) тип процесора або іншого апаратного забезпечення, на якому працює операційна система та прикладні програми.
2) тип операційної системи, встановленої на комп'ютері;
3) комбінація типу апаратного забезпечення та операційної системи.

Апаратні платформи

Говорячи про апаратні платформи, мають на увазі архітектуру процесора (набір інструкцій). Наприклад, архітектура x86 та її різновиди IA-32 (третє покоління архітектури x86, у якому відбувся перехід на 32-розрядні обчислення) та x86-64 (64-бітне розширення набору команд для архітектури x86, вперше розроблене компанією AMD або Intel 64 — відповідна версія архітектури, розроблена корпорацією Intel).

Машини з архітектурою x86 можуть працювати під ОС Microsoft Windows, Linux/Unix, OpenBSD, NetBSD, macOS або FreeBSD.

32-бітові ARM (Advanced RISC Machine) архітектури (і їх більш сучасні 64-бітові версії) використовуються у смартфонах та планшетах, які працюють під керівництвом Android, iOS та інших мобільних операційних систем.

Програмні платформи

Під програмними платформами розуміють операційні системи, програмні середовища (або комбінацію програмного середовища та ОС) та незалежні від ОС віртуальні машини для виконання байт-коду.

Найбільш популярні програмні платформи:

- BlackBerry.
- Android для смартфонів та планшетів (x86, ARM).
- iOS (ARM).
- Microsoft Windows (x86, ARM).

- .NET Framework (також відоме під назвою CLI) і його крос платформний варіант Mono.
- Java.
- Web-браузери (JavaScript web-додатки).
- Linux (x86, PowerPC, ARM та інші архітектури).
- macOS (x86, PowerPC).
- Solaris (SPARC, x86).
- PlayStation 4 (x86).

3. Рівні кросплатформності

Кросплатформність 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 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.

Емуляція

Якщо програма не призначена для виконання (запуску) на певній платформі, але для цієї платформи існує *емулятор платформи*, базовий для цієї програми, то програма може бути виконана в середовищі емулятора.

Зазвичай виконання програми в середовищі емулятора призводить до *зниження продуктивності* в порівнянні з аналогічними програмами, для яких платформа є базовою, тому що значна частина ресурсів системи витрачається на виконання функцій емулятора.

Кросплатформність програм

Для того, щоб ПЗ можна було вважати крос платформним, воно повинно функціонувати на різних апаратних архітектурах та під різними ОС. Розробка такого ПЗ є складним завданням, оскільки різноманітні ОС мають різні API (Application programming interface — прикладний програмний інтерфейс). Для прикладу, Linux та Windows мають різні API. Крім того, з того, що операційна система може працювати на різних архітектурах, не випливає, що ПЗ, розроблене для цієї ОС, буде працювати під усіма цими

архітектурами. Так само, програма, написана на довільній популярній мові, наприклад C чи C++, буде працювати на довільній ОС, під яку існують реалізації цієї мови.

Велика кількість прикладних програм є кросплатформними. Особливо ця якість виражається у програм, що були спершу розроблені для UNIX-подібних операційних систем. Важливою умовою їх переносимості на інші платформи є сумісність платформ з рекомендаціями POSIX, а також існування відповідних компіляторів (наприклад, GCC) або інтерпретаторів для платформи, на яку здійснюється перенесення.

Приклади:

- Apache
- BinkD
- CVS
- Emacs
- GIMP
- GoldEd
- Inkscape
- Lotus Notes
- Mozilla Firefox, Mozilla Thunderbird, SeaMonkey
- MySQL

Кросплатформне ПЗ для Web

Web-сторінки та пов’язані із ними web-додатки (web applications) зазвичай розглядаються як кросплатформні, тому що, в ідеалі, вони мають бути доступні з будь-якого браузера, який працює під управлінням довільної операційної системи. Такі додатки часто засновані на архітектурі клієнт-сервер і можуть мати різноманітну складність, яка часто може бути на заваді їх кросплатформності.

Більшість статичних web-сторінок задовольняють умовам крос-платформності. Однак більшість нових сучасних сайтів використовує велику кількість засобів та технологій, до складу яких входять, наприклад,

Ajax, JavaScript, Dynamic HTML, SVG та інші сучасні компоненти (це так звані rich Internet applications). Ці можливості у повному обсязі є доступними тільки для останніх версій популярних браузерів.

4. 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.

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.

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).

Інтелектуальні стратегії тестування

Розробка кросплатформного ПЗ значно ускладнює процес його тестування, який потрібно проводити окремо для кожної цільової платформи. Додатковою складністю є те, що іноді заборонено одночасно встановлювати різні версії одного і того самого програмного продукту на тому самому комп'ютері (це має місце, наприклад, для різних версій того самого браузера). Для спрощення часто використовують віртуальні версії програмних платформ та емулятори.

Також існують smart-інтсрументи тестування, наприклад Page Object Model, які дають змогу запускати один тестовий скрипт для різних версій програмних продуктів.

5. 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. Це пов’язано із тим, що скрипти у загальному випадку зберігаються у текстових файлах. Незважаючи на проблеми, які пов’язані із різним специфічними системними символами, які використовуються для розмітки текстових файлів (символ нового, кінція файлу, тощо), шляхом нескладного форматування можна досягти того, щоб скрипти, написані для однієї системи, працювали на іншій.

Прикладом популярних кросплатформних мов скриптового (інтерпретованого) типу є:

- Perl — скриптова мова для програмування CGI для WWW, адміністрування малих систем і т.п.
- PHP — скриптова мова для програмування, яка є однією з найбільш популярних мов для розробки web-додатків.
- Python — скриптова мова, у якій зроблено акцент на швидкому та легкому написанні коду програм.
- Ruby — скриптова об’єктно-орієнтована мова, версія якої призначена для розробки web-додатків (Ruby on Rails).

- Tcl — мова динамічного програмування з широким діапазоном застосування.

Спільний недолік скриптових мов — порівняно невисока швидкість виконання програмного коду порівняно із бінарним виконуваним кодом, написаним на мовах, які підтримують процес компіляції. Перевага скриптових мов — значно вища переносимість коду.

6. Підходи до кросплатформного програмування

Існують різні підходи до вирішення проблеми написання кросплатформних програм. 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. Часто реалізація цієї ідеї супроводжується значною кількістю проблем, пов’язаних із тестуванням та супроводом ПЗ, оскільки різні набори вихідних файлів мають різних розробників та різні дефекти. Крім того процес розробки сильно ускладнюється також відмінностями, які існують між різними ОС тієї самої фірми.

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).

Деякі програмні продукти використовують комбінацію обох підходів. Прикладом є web-браузер Firefox, який використовує платформно-орієнтовані набори вихідних кодів для реалізації GUI та реалізацію більше ніж однієї скріптової мови для забезпечення переносимості. Firefox

реалізовує XUL, CSS та JavaScript для розширення можливостей браузера на додаток для класичних плагінів Netscape-браузерів.

7. Підтримка кросплатформності на базі платформи .NET Framework

.NET Framework — програмна платформа, першу версію якої випустила компанія Microsoft в 2002 році. *Основою платформи є загальномовне середовище виконання* Common Language Runtime (CLR), яке підходить для розробки програмних засобів на різних мовах програмування.

Іншою складовою компонентою платформи .NET є *загальна система типів* (Common Type System) або, скорочено, система CTS. В специфікації CTS наведено повний опис усіх можливих типів даних і програмних конструкцій, які підтримує виконуюче середовище CLR, того, вони можуть взаємодіяти і зображуватися у форматі метаданих .NET.

Будь-яка із визначених в CTS функціональних можливостей може не підтримуватися в окремій .NET-сумісній мові. Тому існує ще одна *загальномовна специфікація* (Common Language Specification (CLS), у якій описано тільки ту підмножину загальних типів і програмних конструкцій, які мають підтримувати усі без винятку .NET-сумісні мови.

Архітектура .NET Framework описана и опублікована в специфікації *Common Language Infrastructure* (CLI), затвердженій ISO и ECMA. В CLI описані типи даних .NET, формат метаданих про структуру програми, система виконання байт-коду, тощо.

Об’єктні класи .NET, доступні для усіх мов програмування, що підтримуються, містяться у бібліотеці Framework Class Library (FCL). До складу FCL входять класи:

- Windows Forms — засоби для створення настільних програм;
- ADO.NET — засоби для взаємодії та використання серверів баз даних;

- ASP.NET — засоби для створення клієнт-серверних Web-застосувань;
- Language Integrated Query (LINQ) — вбудована мова інтегрованих запитів;
- Windows Presentation Foundation — технологія створення та обробки ділової графіки,
- Windows Communication Foundation — технологія створення розподілених систем.

Ядро FCL називається **Base Class Library** (BCL).

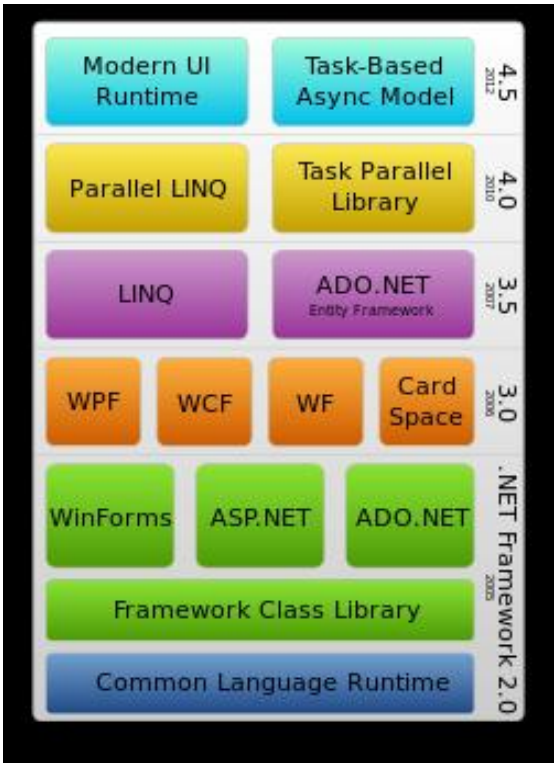

Рис. 1. Стек технологій .NET Framework

Створення та виконання ПЗ у середовищі .NET

1. Спочатку пишеться програма з використанням .NET-сумісної мови програмування.
2. Програма компілюється у MSIL-код, який зберігається у збірці (рис. 2).

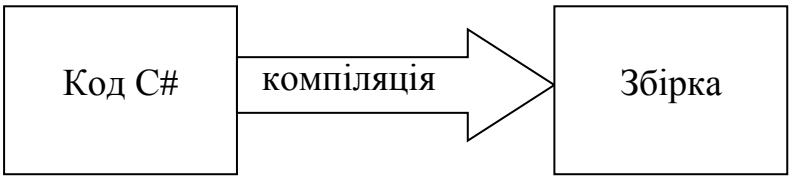

Рис. 2. Компіляція коду у збірку

3. При першому запуску цього коду (в результаті запуску виконуваного файлу або при виклику із іншого коду) він у першу чергу компілюється у рідний код з використанням JIT-компілятора (just in time compiler) (рис. 3).

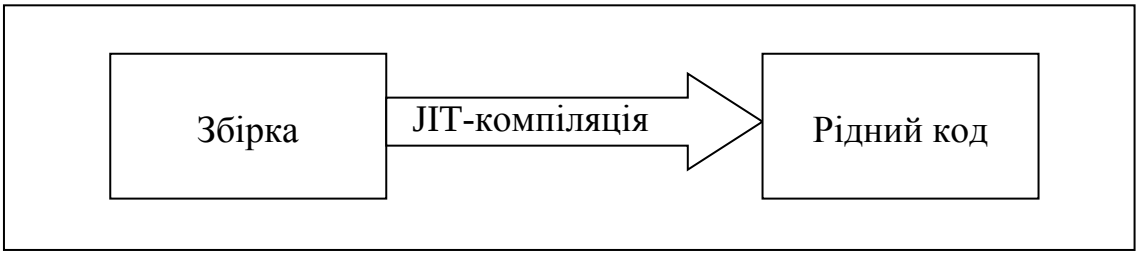

Рис. 3. Компіляція збірки у рідний код

4. Після цього рідний код виконується у контексті CLR (Common Language Runtime) разом з усіма іншими програмами чи процесами, як це наведено на рис. 4 або транслюється за допомогою утиліти NGen.exe у виконуваний код для цільового процесора.

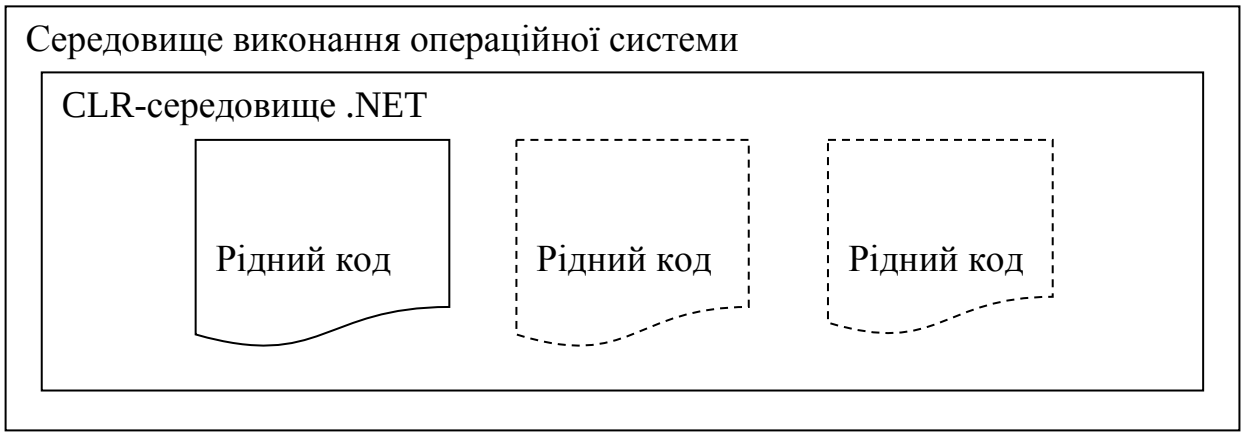

Рис. 4. Виконання рідного коду в CLR.

Шляхом до кросплатформності ПЗ, розробленого для .NET є платформа Mono, яка є відкритою реалізацією .NET. За рахунок використання Mono .NET-додатки можуть працювати під Mac OS X, Solaris, AIX та численними версіями Unix/Linux. Недолік Mono — відставання у часі від .NET Framework (стосовно реалізації підтримки нових можливостей).