Кросплатформне програмування

Кросплатформність ПЗ

Кросплатформність (багатоплатфо́рмність, мультиплатформність) — властивість програмного забезпечення працювати більш ніж на одній програмній (в тому числі — операційній системі) або апаратній платформі; технології, що дозволяють досягти такої властивості.

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

Більш широке поняття — інтероперабильність (interoperability). Це характеристика продукту чи системи, інтерфейси якої цілком зрозумілі, яку призначено для роботи з іншими продуктами чи системами, в даний час або в майбутньому, при здійсненні або доступі, без будь-яких обмежень. Синонімом є сумісність.

Історично склалось різноманіття архитектур обчислювльних систем і як наслідок цього — різноманіття операційних платформ. Існують комп'ютерні архітектури, що домінують на ринку пропозицій, наприклад, x86 та її клони, а також операційні платформи — Linux, Windows, UNIX. Але, навіть для достатньо близьких комп'ютерних архітектур та операційних платформ, можіть існувати проблеми із інсталяцією розробленного програмного забезпечення. Тому кросплатформність у проектуванні та розробці ПЗ дозволяє суттєво скоротити витрати на неї або на її адаптацію до певної платформи.

Залежно від засобів реалізації кросплатформність поділяється на:

- кросплатформність на рівні мов програмування;
- кросплатформність на рівні середовища виконання;
- кросплатформність на рівні операційної системи;
- кросплатформність на рівні апаратної платформи.

Кросплатформність ПЗ

Кросплатформне програмне забезпечення (ПЗ) може бути поділено на дві категорії:
1) ПЗ, яке потребує індивідуальну компіляцію для кожної операційної платформи, на якій воно використовуватись;
2) ПЗ, яке може виконуватися на довільній платформі без додаткової підготовки (інтерпретатори або середовища виконання для цього ПЗ є стандартними компонентами або підтримуються на різних платформах).

Кросплатформність на рівні мов програмування досягається шляхом забезпечення незалежності програмного коду від платформи. Кросплатформними є, наприклад, мови програмування C, C++, Fortran, Pascal, для яких кросплатформність досягається на рівні компіляції (існують компілятори для різних операційних платформ). Кросплатформність на рівні редакторів зв'язків (link) досягається реалізацією для різних платформ кросплатформних бібліотек, які підтримують незалежний від платформи інтерфейс, у тому числі — стандартизованих бібліотек. Також існує велика кількість популярних у використанні кросплатформних бібліотек: Qt, GTK+, FLTK, STL, Boost, OpenGL, SDL, OpenAL, OpenCL.

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.

Кросплатформність ПЗ

Java і C# — кросплатформні мови на рівні виконання, тобто їх виконувані файли можна запускати на різних платформах без попередньої перекомпіляції. PHP, ActionScript, Perl, Python, TcliRuby — кросплатформні інтерпретовані мови, їх інтерпретатори існують для багатьох платформ.

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.

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.

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 засновано на архітектурі клієнт-сервер і можуть мати різноманітну складність. Більшість сучасних web-сторінок використовують такі засоби та технології, як:
Ajax, JavaScript, DHTML та HTML5, SVG та інші RIA-компоненти.

Кросплатформність ПЗ

Для розробки ПЗ, яке поєднує у собі кросплатформність та складну функціональність, є значна кількість стратегій:

- Поступове зменшення можливостей (Graceful degradation)
- Множинні кодові репозитарії (Multiple codebases)
- Єдиний кодовий репозитарій (Single codebase)
- Бібліотеки сторонніх розробників (Third-party libraries)
- Чутливий дизайн(Responsive design)

Кросплатформність ПЗ

Поступове зменшення можливостей (Graceful degradation)

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). Наприклад, перемикання сайту певний базовий режим роботи (basic mode) з обмеженими можливостями, що має місце на браузерах з неповною підтримкою сучасних web-технологій.

Множинні кодові репозитарії(Multiple codebases)

Ця 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 десктопних програмних систем.

Кросплатформність ПЗ

Єдиний кодовий репозитарій (Single codebase)

Ця 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, описано 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, яка 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)

Вони 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.

Кросплатформність ПЗ

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 десктопів. Ця стратегія найчастіше застосовується для розробки web-додатків і тому часто називається Responsive web design (RWD).

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.

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

Кросплатформність ПЗ

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, Linux-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 суттєво ускладнюється також відмінностями, які існують навіть 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).