Операційні системи: процеси і потоки

М.А. Ходукін

Дана стаття призначена для студентів освітньо-кваліфікаційного рівня «бакалавр» спеціальності 121 «Інженерія програмного забезпечення» денної та заочної форм навчання, що вивчають дисципліну «Операційні системи та безпека даних».

Основним поняттям в будь-якій операційній системі є процес: абстракція, що описує поточну програму.

Процеси - це одна з найстаріших і найбільш важливих абстракцій, властивих операційній системі. Вони підтримують можливість здійснення (псевдо) паралельних операцій навіть при наявності всього одного центрального процесора.

Вони перетворюють один центральний процесор в кілька віртуальних. Без абстракції процесів сучасні обчислення просто не можуть існувати.

Сучасні комп'ютери, як правило, зайняті відразу декількома справами. Всією цією роботою потрібно управляти, і тут нам дуже знадобиться многозадачная система, що підтримує роботу декількох процесів.

У будь-який багатозадачній системі центральний процесор швидко перемикається між процесами, надаючи кожному з них десятки або сотні мілісекунд.

При цьому хоча в кожен конкретний момент часу центральний процесор працює тільки з одним процесом, протягом 1 секунди він може встигнути попрацювати з кількома з них, створюючи ілюзію паралельної роботи. Іноді в цьому випадку говорять про псевдопараллелізме на відміну від справжнього апаратного паралелізму в багатопроцесорних системах (у яких є не менше двох центральних процесорів, що використовують одну і ту ж фізичну пам'ять). Людям досить важко відстежувати кілька дій, що відбуваються паралельно. Тому розробники операційних систем за минулі роки створили концептуальну модель послідовних процесів, що спрощує роботу з паралельними обчисленнями.

Модель процесу

У цій моделі все виконується на комп'ютері програмне забезпечення, іноді включаючи операційну систему, зведено до ряду послідовних процесів, або, для стислості, просто процесів.

Процес - це просто екземпляр виконуваної програми, включаючи поточні значення лічильника команд, регістрів і змінних.

Концептуально у кожного процесу є свій, віртуальний, центральний процесор. Зрозуміло, насправді справжній центральний процесор постійно перемикається між процесами, але, щоб зрозуміти систему, куди простіше думати про набір процесів, запущених в (паралельному режимі, ніж намагатися відслідковувати, як центральний процесор перемикається між

програмами. Це постійне перемикання між процесами, називається Мультипрограмування, або багатозадачним режимом роботи.

На рис. 2.1, а показаний комп'ютер, що працює в багатозадачному режимі і має в пам'яті чотири програми.

На рис. 2.1, б показані чотири процесу, кожен з яких має власний алгоритм управління (тобто власний логічний лічильник команд) і працює незалежно від всіх інших. Зрозуміло, що насправді є тільки один фізичний лічильник команд, тому при запуску кожного процесу його логічний лічильник команд завантажується в реальний лічильник.

Коли робота з процесом буде на деякий час припинена, значення фізичного лічильника команд зберігається в логічному лічильнику команд, що розміщується процесом в пам'яті.

На рис. 2.1, в показано, що за досить тривалий період спостереження просунулися вперед всі процеси, але в кожен окремо взятий момент часу реально працює тільки один процес.

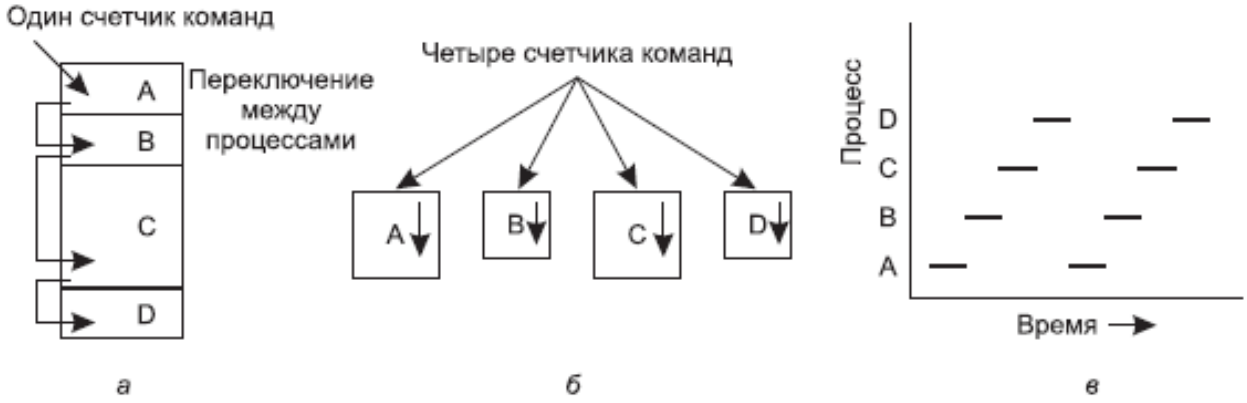

Рис. 2.1. Комп'ютер: а - чотири програми, що працюють в багатозадачному режимі; б - концептуальна модель чотирьох незалежних один від одного послідовних процесів; в - в окремо взятий момент активна тільки одна програма

Оскільки центральний процесор перемикається між процесами, швидкість, з якою процес виконує свої обчислення, що не буде однаковою і, швидше за все, не зможе бути знову показана, якщо той же процес буде запущений ще раз.

Тому процеси не повинні програмуватися з використанням будь-яких жорстко заданих припущень щодо часу їх виконання.

Розглянемо, наприклад, аудіопроцесс, що програє музику для супроводу високоякісного відео, запущеного на іншому пристрої. Оскільки аудіо може бути запущено трохи пізніше відео, аудіопроцесс сигналізує відеосервери про пуск програвання, а потім перед програванням аудіо запускає холостий цикл 10 000 разів.

Якщо цикл послужить надійним таймером, то все пройде як треба, але, якщо ж при виконанні холостого циклу процесор вирішить переключитися на інший процес, аудіопроцесс може поновитися, коли відповідні кадри вже будуть показані, і, на жаль, синхронізація відео і аудіо буде збита.

Коли у процесу є подібні критичні для його роботи вимоги, що стосуються реального масштабу часу, то через певну кількість мілісекунд повинні відбуватися конкретні події, і для того, щоб вони відбулися, повинні бути зроблені спеціальні заходи.

Але, як правило, на більшість процесів не впливають ні встановлений режим багатозадачності центрального процесора, ні відносні швидкості виконання різних процесів.

Різниця між процесом і програмою досить тонка, але дуже суттєва.

Тут нам, напевно, допоможе якась аналогія.

Уявімо собі програміста, який вирішив зайнятися кулінарією і спекти пиріг на день народження дочки. У нього є рецепт пирога, а на кухні є всі інгредієнти: борошно, яйця, цукор, ванільний екстракт і т. Д.

У даній аналогії рецепт - це програма (тобто алгоритм, виражений в якійсь зручній формі записи), програміст - це центральний процесор, а інгредієнти пирога - це вхідні дані.

Процес - це дії, що складаються з читання рецепта нашим кулінаром, вибору інгредієнтів і випічки пирога.

Тепер уявімо, що на кухню вбігає син програміста і кричить, що його вжалила бджола. Програміст записує, на якому місці рецепта він зупинився (зберігається стан поточного процесу), дістає книгу рад по наданню першої допомоги й приступає до виконання викладених в ній інструкцій.

Перед нами процесор, перемкнутий з одного процесу (випічки) на інший процес, що має більш високий ступінь пріоритету (надання медичної допомоги), і у кожного з процесів є своя програма (рецепт проти довідника по наданню першої допомоги).

Після вилучення бджолиного жала програміст повертається до пирога, продовжуючи виконувати дії з того місця, на якому зупинився.

Ключова ідея тут в тому, що процес - це свого роду дії. У нього є програма, вхідні і вихідні дані і стан. Один процесор може спільно використовуватися декількома процесами відповідно до якогось алгоритмом планування, який використовується для визначення того, коли зупинити один процес і обслужити інший.

На відміну від процесу програма може бути збережена на диску і взагалі нічого не робити.

Варто відзначити, що якщо програма запущена двічі, то вважається, що нею зайняті два процеси.

Наприклад, часто можна двічі запустити текстовий процесор або одночасно роздрукувати два файли, якщо одночасно доступні два принтера. Той факт, що два працюючих процесу запущені від однієї і тієї ж програми, до уваги не береться, оскільки це два різних процеси.

Операційна система може дозволити їм використовувати загальний код, тому в пам'яті буде присутній тільки одна копія цього коду, але це чисто технічна деталь, що не міняє концептуальну ситуацію, що стосується двох працюючих процесів.