9.3 ВІДНОВЛЕННЯ ПІСЛЯ ВЗАЄМНОГО БЛОКУВАННЯ

Припустимо, що наш алгоритм виявлення взаємного блокування успішно відпрацював і виявив таке блокування. Що ж далі? Потрібні якісь методи виходу з нього, що дозволяють системі відновити працездатність. У цьому розділі будуть розглянуті різні способи виходу із взаємного блокування.

Виявивши тупик, можна вивести з нього систему, порушивши одну з умов існування тупика. При цьому, можливо, декілька процесів частково або повністю втратять результати виконаної роботи. Складність відновлення обумовлена рядом чинників. У більшості систем немає досить ефективних засобів, щоб призупинити процес, вивести його із системи і відновити згодом із того місця, де він був зупинений. Додатково введемо робочий цілочисельний вектор *Work* (довжини m) і булевий вектор *Finish* (довжини n). Вектор *Work* відображає пробні виділення ресурсів. Вектор *Finish* представляє інформацію про завершення процесів при цьому стані системи.

9.3.1 Відновлення за допомогою перерозподілу ресурсів

Іноді можна тимчасово відібрати ресурс у його поточного процесу і передати його іншому процесу. У більшості випадків для цього може знадобитися втручання оператора, особливо в операційних системах пакетної обробки, що запускаються на універсальних машинах.

Наприклад, щоб відібрати лазерний принтер у процесу, оператор може скласти усі вже віддруковані листи. Потім процес може бути призупинений (помічений як непрацездатний). Після цього принтер може бути виділений іншому процесу. Коли цей процес завершить свою роботу, стопка віддрукованих листів паперу може бути поміщена назад в приймальний лоток принтера і робота початкового процесу може бути відновлена.

Можливість відібрати ресурс у процесу, дозволити використати його іншому процесу, а потім повернути його без сповіщення процесу багато в чому залежить від природи цього ресурсу. Відновлення цим способом частенько ускладнене або зовсім неможливе. Вибір процесу для призупинення обумовлений тим, який саме процес має той ресурс, що у нього можна легко відібрати.

9.3.2 Відновлення шляхом відкату

У ряді систем реалізовані засоби відкату і перезапуску або рестарту процесу з контрольної точки (збереження стану системи в якийсь момент часу). Якщо проектувальники системи знають, що тупик імовірний, вони можуть періодично організовувати для процесів контрольні точки. Іноді це доводиться робити розробникам прикладних програм. Це означає, що стан процесу записується у файл, що дозволить здійснити його подальший перезапуск. Контрольні точки містять не лише образ пам'яті, але і стан ресурсів, тобто інформацію про те, які ресурси в даний момент виділені процесу. Для більшої ефективності нова контрольна точка повинна записуватися не поверх старої, а в новий файл, щоб під час виконання процесу зібралася ціла послідовність контрольних точок.

При виявленні взаємного блокування нескладно визначити, які ресурси потрібні. Щоб вийти із взаємного блокування, процес, що володіє необхідним ресурсом, відкачується назад до точки, передуючої отриманню цього ресурсу, для чого він запускається з однієї зі своїх контрольних точок. Уся робота, виконана після цієї контрольної точки, втрачається. Наприклад, має бути викинута вся віддрукована після цієї контрольної точки вихідна інформація, оскільки вона буде віддрукована знову. Фактично процес повертається до попереднього моменту, коли він ще не мав того ресурсу, який тепер виділений одному з тих, що беруть участь у взаємному блокуванні процесу. Якщо перезапущений процес намагається знову отримати ресурс, йому доводиться чекати, поки той не стане доступний.

9.3.3 Відновлення шляхом знищення процесів

Найгрубішим, але й найпростішим способом перервати взаємне блокування є знищення одного або декількох процесів. Можна знищити процес, що знаходиться в циклі взаємного блокування. Якщо повезе, то інші процеси зможуть продовжити свою роботу. Якщо це не допоможе, то все можна повторити, поки цикл не буде розірваний.

В якості альтернативи жертвою можна обрати процес, що не знаходиться в циклі, щоб він вивільнив утримувані ним ресурси. При цьому підході знищуваний процес обирається з особливою ретельністю, оскільки він повинен утримувати ресурси, необхідні деяким процесам в циклі. Наприклад, один процес може

утримувати принтер і вимагати плотер, а інший процес, навпаки, утримує плотер і просить принтер. Обидва вони знаходяться в стані взаємного блокування. Третій процес може утримувати інший такий же принтер і інший такий же плотер і успішно працювати. Знищення третього процесу призведе до вивільнення цих ресурсів і зруйнує взаємне блокування перших двох процесів.

По можливості краще убити процес, який може бути безболісно перезапущений із самого початку. Наприклад, компіляція завжди може бути перезапущена, оскільки все, що вона робить – це читає вхідний файл і створює об'єктний файл. Якщо процес компіляції буде знищений на півдорозі, то перший запуск не вплине на другий.

А ось процес, що оновлює базу даних, не завжди можна буде безпечно запустити вдруге. Якщо процес додає одиницю до якого-небудь запису таблиці бази даних, то його первинний запуск, знищення, а потім повторний запуск призведуть до невірного результату, оскільки до поля буде додана двійка.