УДК 004.451

В. М. Локазюк, д-р техн. наук, проф.;
О. С. Савенко, канд. техн. наук, доц.;
С. В. Мостовий, асп.

МОДЕЛЬ ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ КОМП'ЮТЕРНОЇ СИСТЕМИ

Проведено аналіз життєвого циклу процесу комп'ютерної системи, виділено граничний стан, що передує стану взаємоблокування.

Вступ

Вагома частка взаємоблокувань процесів припадає на процеси, в комп'ютерних системах (КС).

Під комп'ютерною системою будемо розуміти сукупність програмно-апаратних засобів, призначених для розв'язання поставленої задачі, а саме персональні комп'ютери (ПК) та відповідне програмне забезпечення (як системне, так і прикладне).

В результаті дослідження відомих методів та алгоритмів уникнення взаємоблокувань процесів в операційних системах (ОС) [1, 2] виявлено, що вони не повною мірою розв'язують поставлену задачу. Не всі алгоритми придатні до реалізації у сучасних операційних системах [3, 4], оскільки мають низку недоліків (блокування роботи ОС, наявність циклів активного очікування, складність програмної реалізації для багатьох процесів, необхідність використання спеціалізованої команди процесора) і є складними для реалізації. Тому більшість сучасних ОС не містять ефективних засобів для вирішення проблеми взаємоблокування процесів [5].

Проте масштабність задач, які розв'язуються за допомогою персонального комп'ютера, зростає і це вимагає розв'язання задачі уникнення взаємоблокування процесів. Тому розробка нових методів та засобів, які б дозволили прогнозувати входження процесів у стан взаємоблокування, є актуальною. Для розробки таких методів та засобів необхідно дослідити життєвий цикл процесів КС та визначити, за яких умов може відбутись взаємоблокування процесів.

Основна частина

Багатозадачність (англ. multitasking) — властивість операційної системи або середовища програмування, забезпечувати можливість паралельної (або псевдопаралельної) обробки декількох процесів [6]. Дійсна багатозадачність операційної системи можлива тільки в розподілених обчислювальних системах.

Процес — це система дій, що реалізує певну функцію в обчислювальній системі й оформлена так, що програма керування обчислювальної системи може перерозподіляти ресурси цієї системи з метою забезпечення багатозадачності [6]. Позначимо множину виконуваних процесів, як $A = \{a_i\}_{i=1}^{y}$, де y — кількість процесів.

Ресурс обчислювальної системи — засіб обчислювальної системи, що може бути виділений процесу обробки даних на певний інтервал часу [6]. Позначимо множину наявних ресурсів ОС як $RE = \left\{re_j\right\}_{j=1}^{x}$, де x — кількість видів ресурсів.

До ресурсів ОС віднесемо наявну пам'ять, процесори, пристрої вводу/виводу, а також дані, необхідні для роботи процесів (файли в пам'яті та на зовнішніх носіях, результати обчислень інших процесів).

Кожен процес від моменту створення до моменту завершення проходить через декілька станів (рис. 1). Проте така діаграма станів не відображає стан взаємоблокування процесів.

Під сигнатурою процесу будемо розуміти сукупність його характеристик, яка однозначно ідентифікує стан процесу в ОС в певний момент часу t:

$$a_i(t) \to \left(a_{i_1}^t, a_{i_2}^t, \ldots, a_{i_z}^t\right), \quad (1)$$

де $a_i(t) \in A$ — поточний процес; $a_{i_1}^t, \ldots, a_{i_z}^t$ — характеристики процесу в поточний момент часу (параметри та ресурси, які використовує процес в цей момент).

До характеристик процесу, що формують сигнатуру, віднесемо: ідентифікатор процесу, ідентифікатор батьківського процесу, ідентифікатор користувача, якому належить процес, пріоритет процесу, квоти процесу (кількість пам'яті і процесорний час доступні процесу), дескриптори відкритих процесом файлів [6].

Оскільки до складу сигнатури процесу входять унікальні характеристики, то в один і той самий момент часу в системі не існує двох абсолютно однакових сигнатур.

Отже, життєвий цикл процесу можна подати у вигляді послідовності станів, через які проходить цей процес. Перехід зі стану в стан відбувається через зміну певних параметрів, якими характеризується процес. Зміна параметрів процесу відбувається з таких причин: дії ОС, дії інших процесів, виконання власного програмного коду. Стан кожного окремого процесу буде впливати на стан КС в цілому.

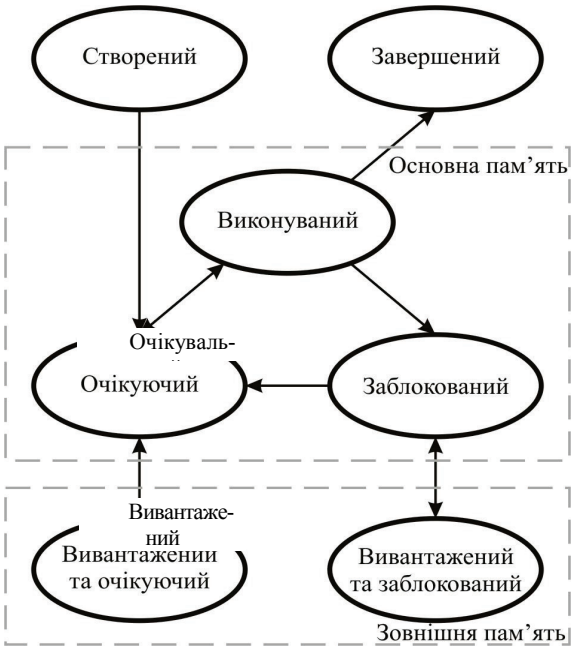

Рис. 1. Діаграма станів процесу

Позначимо стан процесу через w, причину зміни параметрів через r, а перехід із стану в стан через $w_i \xrightarrow{\text{зміна параметрів з причини } r_j} w_{i+1}$. Тоді життєвий цикл процесу матиме вигляд (2)

$$a_i : w_0 \xrightarrow{r_j} w_1 \xrightarrow{r_j} \ldots \xrightarrow{r_j} w_{k-1} \xrightarrow{r_j} w_k, \quad (2)$$

де $w_0 \in W$ — початковий стан процесу (стан «створений»); $w_k \in W$ — кінцевий стан процесу (стан «завершений»); W — множина програмних станів процесу (див. рис. 1); $r_j \in R$ — множина можливих причин зміни параметрів процесу (j = 1, 2, 3…).

Враховуючи визначення сигнатури та (1) і (2), життєвий цикл кожного процесу можна подати у вигляді

$$a_i : \left(a_{i_1}^{t_0}, a_{i_2}^{t_0}, \ldots, a_{i_z}^{t_0}\right) \xrightarrow{r_{j1}} \left(a_{i_1}^{t_1}, a_{i_2}^{t_1}, \ldots, a_{i_z}^{t_1}\right) \xrightarrow{r_{j2}} \ldots \xrightarrow{r_{jk}} \left(a_{i_1}^{t_k}, a_{i_2}^{t_k}, \ldots, a_{i_z}^{t_k}\right). \quad (3)$$

У стан взаємоблокування можуть потрапляти процеси, що взаємодіють між собою у багатозадачних ОС в певний момент часу. До потрапляння у стан взаємоблокування процеси протягом свого життєвого циклу знаходяться в інших станах, а саме стан «створений», стан «очікування», стан «виконуваний», стан «заблокований», стан «завершений» (рис. 2). У стан взаємоблокування процеси потрапляють, як правило, із стану «заблокований» або стану «очікування». Отже, у цей момент часу серед множини процесів можна виділити підмножину процесів, які можуть в наступний момент часу потрапити до стану взаємоблокування. Перед входженням у стан взаємоблокування процес буде знаходитись у певному «граничному» стані [7], після якого ймовірність переходу у стан взаємоблокування буде високою. Взаємоблокування процесів призводить до часткової або повної втрати функційної здатності ОС. Тому вважатимемо стан взаємоблокування процесів неробочим станом ОС, а інші стани процесів — робочим станом ОС.

Віднесемо до робочого стану такі стани процесу: стан «створений», стан «виконуваний», стан «завершений». До «граничного» стану віднесемо такі стани процесу: стан «заблокований», стан «очікування».

Подамо шлях, яким процес потрапляє у стан взаємоблокування, у вигляді схеми, що показана на рис. 3.

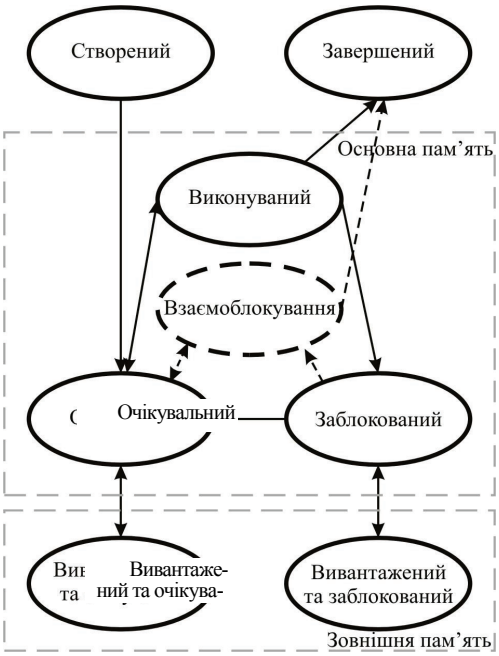

Рис. 2. Діаграма станів процесу, що включає стан взаємоблокування

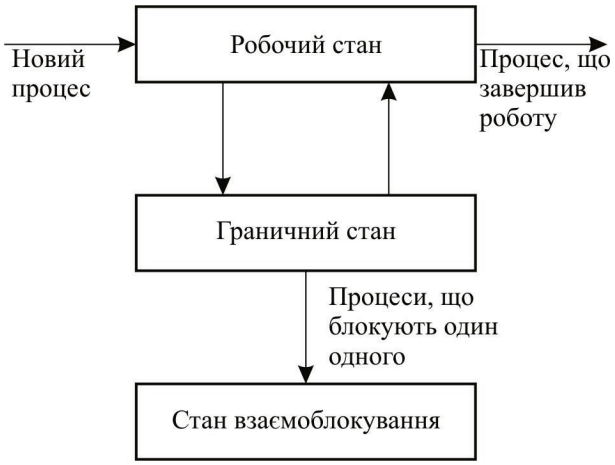

Рис. 3. Схема переходу процесів у стан взаємоблокування

Як видно зі схеми, виникнення взаємоблокування процесів можливе лише для частини процесів, що знаходяться у граничному стані. У разі переходу процесів до граничного стану відбувається зміна їхніх параметрів.

Розглянемо життєвий цикл процесу. В момент створення (стан «створений») процес знаходиться у робочому стані, йому надана частина ресурсів системи. Позначимо цей стан процесу, як $s_{\text{роб}}(t)$. В певний момент часу процесу для подальшого виконання необхідний додатковий ресурс системи, який на цей час є недоступний. Відбувається перехід процесу до стану «заблокований», тобто процес потрапляє у граничний стан. Позначимо цей стан процесу, як $s_{\text{гран.}}(t)$, а перехід до цього стану, як $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$. У разі задоволення потреб процесу у ресурсах він повертається у робочий стан, тобто здійснює перехід $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$. Коли необхідний ресурс отримати неможливо з причини його використання іншим процесом, то процес залишається у граничному стані. Якщо ж другий процес в свою чергу очікує ресурс, що зайнятий першим процесом, то вони обидва потрапляють у стан взаємоблокування. Позначимо цей стан процесу як $s_{\text{взаємоблокування.}}(t)$, а перехід до цього стану як $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$.

Таким чином, враховуючи (3), життєвий цикл процесу буде мати один з таких варіантів:

1. Процес створений, йому надано всі необхідні для завершення роботи ресурси, він виконується і завершується (процес весь час знаходиться в робочому стані $s_{\text{роб}}(t)$), тобто

$$a_i : s_0 \xrightarrow{r_j} s_1 \xrightarrow{r_j} s_k, \tag{4}$$

де $s_0 \in s_{\text{роб}}$; $s_1 \in s_{\text{роб}}$; $s_k \in s_{\text{роб}}$.

2. Процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, через деякий час отримує їх, виконується і завершується (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ в граничний стан $s_{\text{гран}}(t)$, потім $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$ знову в робочий стан $s_{\text{роб}}(t)$), тобто

$$a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_k, \quad (5)$$

де $s_0 \in s_{\text{роб}}$; $s_l \in s_{\text{гран}}$; $s_k \in s_{\text{роб}}$.

3. Процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, які утримує інший процес, який в свою чергу потребує ресурсів першого процесу (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ в граничний стан $s_{\text{гран}}(t)$, із якого відбувається перехід $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$ до стану взаємоблокування).

$$\begin{cases} a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} s_x; \\ a_m : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_n \xrightarrow{r_j} s_x, \end{cases} \quad (6)$$

де $s_0 \in s_{\text{роб}}$; $s_l \in s_{\text{гран}}$; $s_n \in s_{\text{гран}}$; $s_x \in s_{\text{взаємоблокування}}$.

Із розглянутих вище можливих варіантів поведінки процесів критичним для ОС є останній, за якого процеси потрапляють у стан взаємоблокування.

Висновок

Проаналізовано життєвий цикл процесу комп'ютерної системи. В результаті дослідження виявлено «граничний» стан, який передує стану взаємоблокування процесів, що дозволяє в поточний момент часу визначити множину процесів, які можуть потрапити в стан взаємоблокування в наступний момент часу. В подальшому необхідно визначити та дослідити умови та параметри комп'ютерної системи, за яких процеси із виділеної множини потраплять у взаємоблокування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Coffman E. G. System deadlocks / E. G. Coffman, M. J. Elphick, A. Shoshani // Computing Surveys. — June 1971. — Vol. 3, No. 2. — P. 67—78.

2. Isloor S. S. The Deadlock Problem: An Overview / S. S. Isloor, T. A. Marsland // Computer. — 1980. — № 9, vol. 13. — P. 58—78.

3. Kaveh N. Deadlock detection in distribution object systems / Nima Kaveh, Wolfgang Emmerich // Software Engineering Notes. — September 2001. — Vol. 26, No. 5. — P. 44—51.

4. Confirmation of deadlock potentials detected by runtime analysis / [Saddek Bensalem, Jean-Claude Fernandez, Klaus Havelund, Laurent Mounier] // International Symposium on Software Testing and Analysis — 2006. — P. 41—50.

5. Савенко О. С. Дослідження та аналіз блокування процесів в комп'ютерній системі / О. С. Савенко, Ю. П. Кльоц, С. В. Мостовий // Вісник ХНУ. — 2007. — Т. 1, № 3. — С. 248—251.

6. Таненбаум Э. Операционные системы: разработка и реализация. Класика CS / Э. Таненбаум, А. Вудхалл — СПб. : Питер, 2006. — 576 с.

7. Поморова О. В. Теоретичні основи, метод та засоби інтелектуального діагностування комп'ютерних систем : моног. / О. В. Поморова — Хмельницький. : ТОВ «Тріада-М», 2006. — 253 с.

Рекомендована кафедрою обчислювальної техніки

Стаття надійшла до редакції 25.02.11
Рекомендована до друку 4.03.11

Локазюк Віктор Миколайович — завідувач кафедри, ***Савенко Олег Станіславович*** — доцент, ***Мостовий Сергій Володимирович*** — асистент.

Кафедра системного програмування, Хмельницький національний університет, Хмельницький