УДК 004.451

О.С. САВЕНКО, С.В. МОСТОВИЙ

Хмельницький національний університет, Україна

ЖИТТЄВИЙ ЦИКЛ ПРОЦЕСІВ КОМП'ЮТЕРНОЇ СИСТЕМИ

В роботі надано інформацію щодо актуальності проблеми взаємоблокування процесів в сучасних операційних системах. Також проведено аналіз життєвого циклу процесу, виділені поняття «процес» та «ресурс». Було розглянуто основні стани процесів, і як наслідок здійсненого аналізу було виділено граничний стан, що передує стану взаємоблокування. Розуміння та наявність чітких характеристик цього граничного стану дає можливість в поточний момент часу визначити множину процесів, які можуть потрапити в стан взаємоблокування в наступний момент часу.

Ключові слова: *процес, ресурс, взаємоблокування, стан процесу, сигнатура процесу, життєвий цикл процесу.*

Вступ

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

Під комп'ютерною системою будемо розуміти сукупність програмно-апартних засобів, призначених для вирішення 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 в повному обсязі вирішують поставлену задачу. Не всі алгоритми придатні до реалізації у сучасних операційних системах [3,4], оскільки мають ряд 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 вирішення задачі уникнення взаємоблокування процесів. Тому задача розробки 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 ивзначити при яких умовах 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 керуюча програма 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 $RE = \{re_j\}_{j=1}^{x}$, де x – кількість видів ресурсів.

Рис. 1. Діаграма станів процесу

До ресурсів ОС віднесемо наявну пам'ять, процесори, пристрої вводу/виводу, а також дані, необ-

хідні 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 ряд 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:

$$a_i(t) \to (a^t_{i_1}, a^t_{i_2}, ..., a^t_{i_z}), \qquad (1)$$

де $a_i(t) \in A$ – поточний процес,

$a^t_{i_1}, \ldots, a^t_{i_z}$ – характеристики процесу в поточний момент часу (параметри та ресурси, які використовує процес в даний момент).

До характеристик процесу, що формують сигнатуру, віднесемо наступні: 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 із стану в стан відбувається через зміну певних параметрів, якими характеризується процес. Зміна параметрів процесу відбувається з ряду причин: дії ОС, дії інших процесів, виконання власного програмного коду. Стан кожного окремого процесу буде впливати на стан КС в цілому.

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 буде мати вигляд (2):

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

де $w_0 \in W$ – початковий стан процесу (стан «створений»),

$w_k \in W$ – кінцевий стан процесу (стан «завершений»),

W – множина програмних станів процесу (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 як

$$a_i : (a^{t_0}_{i_1}, a^{t_0}_{i_2}, ..., a^{t_0}_{i_z}) \xrightarrow{r_j} (a^{t_1}_{i_1}, a^{t_1}_{i_2}, ..., a^{t_1}_{i_z}) \xrightarrow{r_j} (a^{t_{k-1}}_{i_1}, a^{t_{k-1}}_{i_2}, ..., a^{t_{k-1}}_{i_z}) \xrightarrow{r_j} (a^{t_k}_{i_1}, a^{t_k}_{i_2}, ..., a^{t_k}_{i_z}). \qquad (3)$$

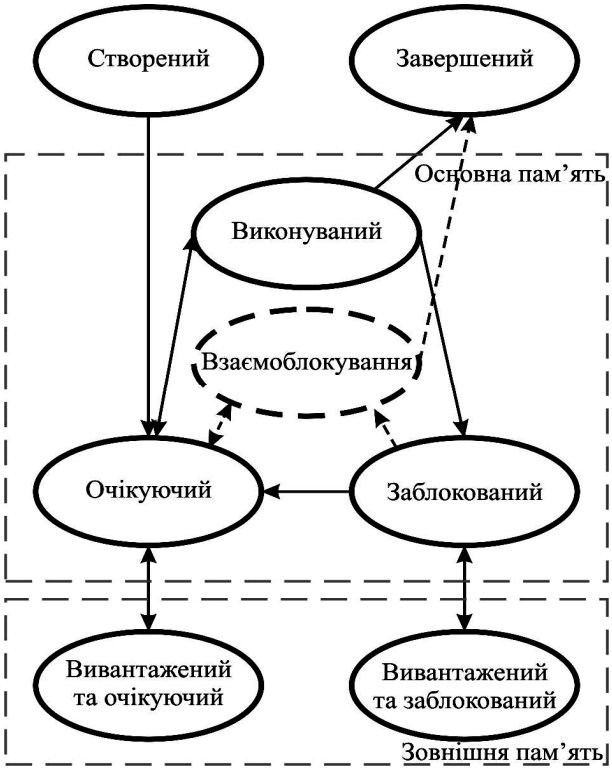

Рис. 2. Діаграма станів процесу, що включає стан взаємоблокування

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 «очікуючий», стан «виконуваний», стан «заблокований», стан «завершений» (рис. 2). У стан взаємоблокування процеси потрапляють, як правило, із стану «заблокований» або стану «очікуючий». Отже, у даний 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 «очікую-

чий». Представимо шлях, яким процес потрапляє у стан взаємоблокування, у вигляді наступної схеми (рис. 3).

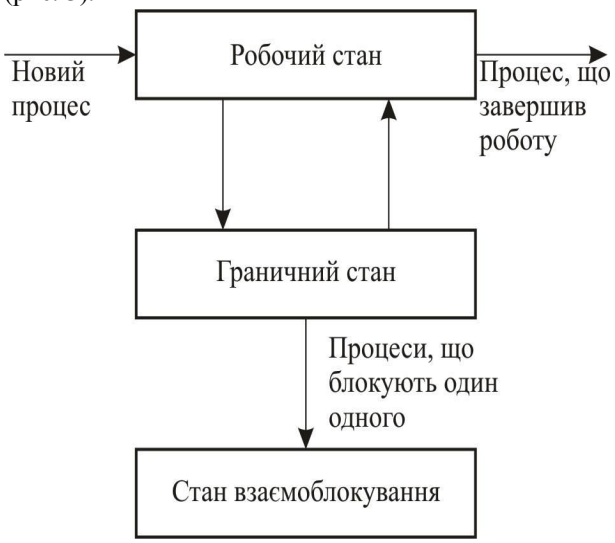

Рис. 3. Схема переходу процесів у стан взаємоблокування

Як видно із схеми, виникнення взаємоблокування процесів можливе лише для частини процесів, що знаходяться у граничному стані. При переході 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 $s_{роб}(t)$. В певний момент часу процесу для подальшого виконання необхідний додатковий ресурс системи, який на даний час є недоступний. Відбувається перехід процесу до стану «заблокований», тобто процес потрапляє у граничний стан. Позначимо цей стан процесу, як $s_{гран.}(t)$, а перехід до даного стану, як $s_{роб}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{гран}(t)$.

При задоволенні потреб процесу у ресурсах він повертається у робочий стан, тобто здійснює перехід $s_{гран}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{роб}(t)$.

Коли необхідний ресурс отримати неможливо по причині 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 $s_{взаємоблокування.}(t)$, а перехід до даного стану, як $s_{гран}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{взаємоблокування}(t)$.

Таким чином, враховуючи (3), життєвий цикл процесу буде мати один з наступних виглядів:

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

$$a_i : s_0 \xrightarrow{r_j} s_l \xrightarrow{r_j} s_k, \qquad (4)$$

де $s_0 \in s_{роб}$, $s_l \in s_{роб}$, $s_k \in s_{роб}$.

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 $s_{роб}(t)$, потім переходить $s_{роб}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{гран}(t)$ в граничний стан $s_{гран}(t)$, потім $s_{гран}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{роб}(t)$ знову в робочий стан $s_{роб}(t)$), тобто:

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

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

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 $s_{роб}(t)$, потім переходить $s_{роб}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{гран}(t)$ в граничний стан $s_{гран}(t)$, із якого відбувається перехід

$$s_{гран}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{взаємоблокування}(t)$$

до стану взаємоблокування.

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

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

Із розглянутих вище можливих варіантів поведінки процесів критичним для ОС є останній, при якому процеси потрапляють у стан взаємоблокування

Висновок

В роботі проведений аналіз життєвого циклу процесу. В результаті дослідження виявлений «граничний» стан, який передує стану взаємоблокування процесів. Це 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 КС, при яких процеси із виділеної множини потраплять у взаємоблокування.

Література

1. Coffman E.G. System deadlocks / E.G. Coffman, M.J. Elphick, A. Shoshani // Computing Surveys. – June 1971. – Vol.3, № 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 / N. Kaveh, W. Emmerich // Software Engineering Notes. – September 2001. – Vol. 26, № 5. – P. 44-51.

4. Bensalem S. Confirmation of deadlock potentials detected by runtime analysis / S. Bensalem, J.-C. Fernandez, K. Havelund, L. 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. – № 3, т.1. – 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: моногр. / О.В. Поморова. – Хм.: ТОВ «Тріада-М», 2006. – 253 с.

Надійшла до редакції 7.02.2010

Рецензент: д-р техн. наук, проф., проф. кафедри системного програмування О.В. Поморова, Хмельницький національний університет, Хмельницький, Україна.

ЖИЗНЕННЫЙ ЦИКЛ ПРОЦЕССОВ КОМПЬЮТЕРНОЙ СИСТЕМЫ

О.С. Савенко, С.В. Мостовой

В работе представлена информация об актуальности проблемы взаимоблокировки процессов в современных операционных системах. Также проведен анализ жизненного цикла процесса, выделены понятия «процесс» и «ресурс». Были рассмотрены основные состояния процессов, и как следствие осуществленного анализа было выделено граничное состояние, предшествующее состоянию взаимоблокировки. Понимание и наличие четких характеристик этого граничного состояния позволяет в текущий момент времени определить множество процессов, которые могут попасть в состояние взаимоблокировки в следующий момент времени.

Ключевые слова: процесс, ресурс, взаимоблокировки, состояние процесса, сигнатура процесса, жизненный цикл процесса.

LIFE CYCLE OF PROCESSES OF COMPUTER SYSTEM

O.S. Savenko, S.V. Mostovoy

The paper provides information on the relevance of the problem of deadlock processes in modern operating systems. Also, an analysis of the process life cycle was done, the concept of "process" and "resource" were highlighted. The main statuses of the process were considered, and the boundary state prior to the deadlock state was defined as a result of analysis. Understanding and clear characteristics of this boundary state allows determine in the current time the set of processes that can be caught in a deadlock state in the next time.

Keywords: processes, resources, deadlocks, status of the process, process signature, process life cycle.

Савенко Олег Станіславович – канд. техн. наук, доц., декан факультету комп'ютерних систем та програмування, Хмельницький національний університет, Хмельницький, Україна, e-mail: kism@beta.tup.km.ua.

Мостовий Сергій Володимирович – асистент кафедри системного програмування, Хмельницький національний університет, Хмельницький, Україна, e-mail: ks99@datasvit.km.ua.