УДК 004.424 : 378.147

М. А. Дзюба, П. Ю. Катін, м. Київ

МОДЕЛЮВАННЯ БАГАТОПОТОЧНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА БАЗІ WINAPI МЕРЕЖАМИ ПЕТРІ

Paper develops a mathematical model in the form of Petri nets to formalize the work of multi-threaded applications and synchronization mechanisms. It allows to formalize several processes or threads in the operating system and the process of synchronization, simplifies student perceptions of the material and makes it possible to apply methods of mathematical analysis to speed up learning.

Актуальність

Суть моделювання виявляється у створенні умов для вивчення явищ чи процесів, недоступних для безпосереднього спостереження за ними. Зазвичай модель розглядають як “певний аналог відповідного фрагменту дійсності” [1, с.138]. Використання моделей дозволяє реалізувати важливий дидактичний принцип наочності у навчанні, важливий для покращення запам'ятовування словесних відомостей. Наочні посібники використовуються тоді, коли предмети, явища, процеси, що вивчаються на уроках, не можна продемонструвати безпосередньо. Об'єктивна потреба використання наочних засобів у процесі навчання обумовлена їх великим впливом на процес розуміння і запам'ятовування: при дослідній перевірці ефективності запам'ятовування темпу встановлено, що при слуховому сприйманні засвоюється - 15% інформації, при зоровому - 25%, а в комплексі, тобто при зоровому і слуховому одночасно, - 65% [2]. Під час вивчення нового матеріалу, коли в учнів немає чітких і правильних уявлень про об'єкти і явища, це має особливо велике значення.

Дослідження фізіологів показали, що 80% інформації людина одержує через зоровий аналізатор. Пропускна здатність каналів прийому й обробки інформації по лінії "вухо-мозок" дорівнює 50000 біт/с, а по лінії "око-мозок" - 50000000 біт/с.

Ці дані дозволяють зробити висновок про необхідність обов'язкового поєднання учителем словесних і несловесних (зорових, наочних) методів навчання[2].

Постановка задачі

Залежно від форми навчання (лекція, лабораторне чи практичне заняття) моделювання виконує різні завдання і функції. За рахунок моделювання реалізуються такі завдання [3]:

презентація об’єкта вивчення в упорядкованому вигляді;

отримання розгорнутої інформації про предмет чи явище;

пояснення й ілюстрування елементів явища та зв’язків між ними;

забезпечення процесу пізнання.

Але для навчання розробці програмного забезпечення найважливіше, особливо стосовно багатопоточного програмного забезпечення, наочно виділити і показати процес коректного виконання програми,.

Аналіз наукових публікацій і літератури показує [5-10], що процес документування ПЗ має багато різних реалізацій і не завжди формалізований. Найбільш розповсюдженими способами формалізації ПЗ є вихідний код (або псевдокод) з коментарями, блок-схема алгоритму і діаграми UML (англ. Unified Modeling Language). Тим не менш вони не ефективні при візуальному навчанні студентів розробці багатопоточних додатків, що сьогодні особливо актуально. Доступні двоядерні ЦПУ з'явились ще в 2005 році, і тенденція до збільшення кількості ядер у сучасних обчислювальних системах і кількості потоків існує. Паралельні обчислення стали нормою ПП.

Для навчання студентів розробці багатопотокових додатків необхідно поставити у відповідність багатопотокової програми, її вихідному коду наочну формалізовану математичну модель (ММ). Вона повинна дати можливість у графічному вигляді представити множину можливих станів обчислювальної системи і зв'язок між ними. При цьому ММ повинна ураховувати механізми синхронізації, що діють у ОС. Аналіз літератури [5-17] показав, що таку можливість надає математична модель у вигляді мережі Петрі (МП).

Вирішення задачі

Для того щоб ММ (у вигляді мережі Петрі) що буде використовуватися для навчання студентів була більш зрозумілою і реалізовуємою слід поставити певні обмеження. Нехай за умови правильної роботи програма має кінцеву множину станів, а також перехід з одного стану у інший є детермінованим. Зважаючи на це модель можна побудувати на базі класичної, нефарбованої, однорівневої МП.

Введемо формальний опис МП з використанням відомої теорії [4,5, 15-17]. Графічно МП представляється у вигляді двочасткового орієнтованого мультиграфа з двома типами вершин – місцями і переходами. Місце позначається у вигляді кола, перехід – рискою або прямокутником. У цілому, однорівнева МП повністю описується наступними елементами: множиною позицій, переходів, вхідних дуг, вихідних дуг, міток мережі. Отже маємо упорядковану множину, що включає чотири елементи:

$$C = (P, T, I, O), \tag{1}$$

де $P = \{p_1, p_2, ..., p_n\}$ є кінцевою множиною місць $n \geq 0$, а $T = \{t_1, t_2, ..., t_m\}$ являє собою множину переходів $m \geq 0$. Множина місць P і переходів T не перетинаються [4, 15-17].

Вхідна функція є відображенням переходів у комплекти місць виду $I : T \to P_\infty$. Вихідна функція є відображенням з переходів у комплекти місць виду $O : T \to P_\infty$ [16].

Граф МП можна представити у вигляді $G = VA$, де $V = \{u_1, u_2, ..., u_s\}$ – множина вершин, а множина $A = \{a_1, a_2, ..., a_m\}$ є комплектом спрямованих дуг, які визначають взаємозв'язок між вершинами [16].

Маркування МП математично формалізується у вигляді функції, що відображає множину місць у множину натуральних чисел (у векторному вигляді) $\mu = \{\mu_1, \mu_2, ..., \mu_n\}$.

Уведемо функцію $\delta(\mu t_i)$, що називається функцією наступного стану МП, і початкове маркування μ_0 [4, 15-17]. Робота МП здійснюється у вигляді послідовності спрацьовування переходів і зміни маркування. Повний опис роботи і способи аналізу МП докладно описані в [15-17] і в статі не розглядаються.

Практика розробки і налагодження програмного забезпечення дає можливість побудувати ММ у вигляді МП у наданій далі послідовності.

Розглянувши вихідний код і проаналізувавши хід його коректної роботи, отримаємо множину можливих станів, яку формалізуємо у вигляді $V = \{u_1, u_2, ..., u_s\}$. Для цього представимо множину V у вигляді таб.1, що містить наступні рядки: назва місця моделі, позначка місця на моделі, код, що відповідає місцю.

Приклад співставлення множини $V = \{u_1, u_2, ..., u_s\}$ і таблиці реалізований на прикладі спрощеного вихідного коду C++ (WinAPI), що наданий у додатку 1 показаний у таб. 1.

Таблиця 1.

Множина місць

№	Назва місця на моделі	Позначка місця	Код
1	start	M1	Запуск
2	cyclestart	M2	Початок циклу
3	tread1	M3	HANDLE Thread1
4	tread2	M4	HANDLE Thread2
5	tread3	M5	HANDLE Thread3
6	tread1event	M6	SetEvent(param.currThreadDone);
7	tread2event	M7	SetEvent(param.currThreadDone);
8	tread3event	M8	SetEvent(param.currThreadDone);
9	tread1working	M9	Thread1 = CreateThread(…)
10	tread2working	M10	Thread2 = CreateThread(…)
11	tread3working	M11	Thread3 = CreateThread(…)
12	waiting1	M12	WaitForSingleObject(Thread1, INFINITE)

13	waiting2	M13	WaitForSingleObject(Thread2, INFINITE)
14	waiting3	M14	WaitForSingleObject(Thread3, INFINITE)
15	Cycle end	M15	{ exit = true;}
16	end of file1	M16	ThreadsDone[0]
17	end of file2	M17	ThreadsDone[1]
18	end of file3	M18	ThreadsDone[2]
19	end	M19	return 0;

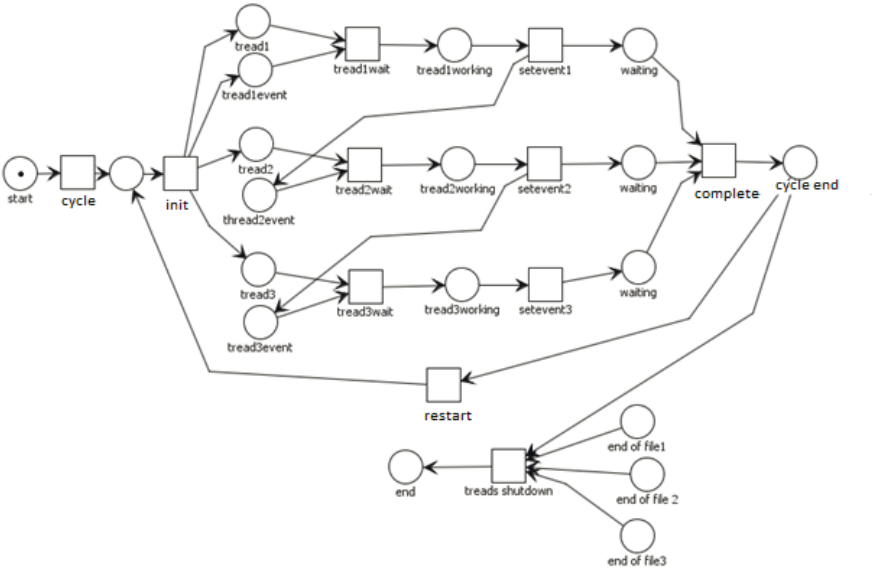

Рис. 1. Побудована мережа Петрі

Кожному рядку таблиці 1 може бути співставлень або true або false, що дає можливість формалізувати стан моделі у вигляді множини міток мережі Петрі $\mu = \{\mu_1, \mu_2, ..., \mu_n\}$. Якщо умова виконується або результат позитивний, змінна приймає значення (true), в цьому випадку в місце, якому відповідає умова або результат, розміщується мітка. В протилежному випадку мітки немає. На початку роботи програми визначається початкова розмітка мережі.

Аналогічно можна представити множину переходів. Кожному переходові відповідає умовна перевірка умов і крок виконання програми ($T = \{t_1, t_2, ..., t_m\}$).

Використовуючи вищеописану методику формалізації, розроблена модель, що показана у вигляді мережі Петрі на рис. 1.

Використання моделі для навчання студентів дає можливість формально в динамці показати узагальнений процес роботи програми.

Початок позначено місцем start. При спрацюванні переходу cycle перевіряється, чи є необхідність перезапуску, після чого перехід init посилає мітки на thread1, thread2, thread3 і thread1working. Мітка на thread1-3 означає активацію відповідного потоку командою CreateThread, але без мітки на threadXevent переходи threadXwait їх не запускають. Так як у нас є мітка на thread1event , то запускається перший потік командою SetEvent(t3DoneEvent). Місце thread1working символізує зчитування файла, після закінчення якого активується перехід setevent1, що посилає дві мітки. Перша потрапляє на thread2working, і запускає другий потік. Друга мітка потрапляє на waiting1, що є очікуванням завершення потоку WaitForSingleObject(Thread1, INFINITE). Аналогічно відбувається для другого і третього потоків. Як тільки всі потоки досягають кінця, то відбувається спрацювання переходу complete, який посилає мітку на cycleend. Далі іде перевірка на кінець файлів наявністю міток на end of file1, end of file2, end of file3. Якщо мітки відсутні, то іде перезапуск посиланням мітки на перехід restart, а звідти на місце cyclestart. Інакше перехід threads shutdown посилає мітку на end, завершуючи програму.

Перевіримо адекватність моделі. програми. Для цього здійснимо покрокову компіляцію програми і її виконання без заходу у функції. Порівняння ММ і ходу покрокового виконання показали, що ММ у вигляді МП адекватна коду вихідної програми. Отже поставлена мета досягнута.

Отримані результати дають можливість стверджувати наступне.

- Модель дозволяє наочно представити роботу програми, для її обговорення під час налагодження або з навчальною метою.

- Мережа Петрі, що використовується у ММ дає можливість використати у подальшому методи математичного аналізу для розробки, налагодження, супроводження або якісного тестування [9-12].

Використання моделі під час навчання і лекцій показало краще сприйняття матеріалу студентами ніж просте використання вихідного коду.

Додаток 1.

```
struct ParamStruct// структура , яка буде передаватися в потік
//…Пропущено код
HANDLE CharToHandle(TCHAR* line);// функція для отримання
успадкованого handle
DWORD WINAPI ReadFromFile(LPVOID params);// функція , яка
запускується в потоці
HANDLE globalFile;        // прийнятий від першого потоку handle файлу
int globalcounter = 0;                // лічильник для підрахунку кількості зчитувань
bool ThreadsDone[3] = { false, false, false };
int main(int argc, TCHAR* argv[])
```

```
{
//…Пропущено код
	HANDLE Thread1, Thread2, Thread3//handle'и потоків
	HANDLE t1DoneEvent = CreateEvent(NULL, FALSE, FALSE, NULL),
// подія першого потоку
//…Пропущено код
	ParamStruct thread1Params, thread2Params, thread3Params; // аргументи
для потоків
	ZeroMemory(&thread1Params, sizeof(thread1Params)); //очистка
аргументів
//…Пропущено код
	_tcscpy(thread1Params.filePath, _T("..\\..\\F21.txt"));		// копіюємо
шлях до файлу для 1 потоку
	thread1Params.threadID = 0; //номер першого потоку 0
//…Пропущено код
	thread1Params.prevThreadDone = t3DoneEvent;// ісля закінчення
третего має запуститися перший
	thread1Params.currThreadDone = t1DoneEvent;// в'язуємо поточний з
його подією
/*…Пропущено код - Аналогічно для наступних*/
	globalFile = CharToHandle(argv[0]);		// отримуємо handle файлу
в який потрібно записувати
	Thread1 = CreateThread(NULL, 0, ReadFromFile, &thread1Params, 0,
NULL); // створюємо перший потік
//…Пропущено код
	SetEvent(t3DoneEvent);		// вказуємо що подія відбулася для
запуску першого потоку
	WaitForSingleObject(Thread1, INFINITE); // очікуємо завершення
першого потоку
//…Пропущено код
}
DWORD WINAPI ReadFromFile(LPVOID params)
{
//…Пропущено код
/*Підготовка до зчитування*/
	if (FileToRead != INVALID_HANDLE_VALUE)
	{
			do{
// очікуємо сигналу від події прив'язаного до попереднього потоку
		WaitForSingleObject(param.prevThreadDone, INFINITE);
				if (!ThreadsDone[param.threadID])
				{
				// зчитуємо з файлу в буффер 2 байти
```

```
                        ReadFile(FileToRead, &buffer, 2, &bytesRead, NULL);
                                        if (bytesRead > 0)
                                        {
                                                globalcounter++;
                                        }
                                        if (bytesRead < 2)
                                        {
                                        ThreadsDone[param.threadID] = true;
                                        }
                                // записати у файл з буффера
                        WriteFile(globalFile, buffer, bytesRead, &bytesWritten, NULL);
                                        }
                          // сигналізуємо що поточний потік звільняє подія
                                        SetEvent(param.currThreadDone);
                        if (ThreadsDone[0] && ThreadsDone[1] && ThreadsDone[2])
                                        {
                                                exit = true;
                                        }
                                } while (exit == false);
                        CloseHandle(FileToRead);                        // закриваємо файл
                }
                else
                {
                        printf("File wasn't opened\n");
                }
                return 0;
}
HANDLE CharToHandle(TCHAR* line)    // функція для перетворення handle з рядка в handle
//…Пропущено код
```


1. *Батароев К.Б.* Аналогии и модели в познании. Новосибирск: Наука,1981. – 320 с.
2. *Пальчевький С. С.* Педагогіка Друге вид. - К. : Каравела, 2008. - 496 с.
3. *Голуб Н.* Моделювання як метод навчання у вищій школі // ВІСНИК ЛЬВІВ. УН-ТУ Серія філол. - Вип.50. – Львів:2010 - С.66-72.
4. *Міщенко В.О.* Побудова мереж Петрі для документування програмного забезпечення на базі функцій API WINDOWS / В.О. Міщенко, П.Ю. Катін // Енергетика і автоматика – Вип.4 – К.:2011.
5. *Солтер Н.А.* C++ для профессионалов; [пер. с англ.] / Солтер Николас, Клепер Скотт Дж.; пер. с англ. –М.: ООО «И.Д. Вильямс», 2006. 912 с.
6. *Харт, Джонсон, М.* Системное программирование в среде Win32, изд.: Пер. с англ. –М. Издательский дом Вильямс. 2001. –464 с.
7. *Золотова Е.В.* Применение сетей Петри при разработке графического языка программирования: материалы Дальневосточной математической школы-семинара имени академика / Е.В. Золотова, В.Е. Мельников, Д.И. Харитонов. –Владивосток,

ИПМ ДВО РАН 1998, с. 59.
8. *Анисимов Н.А*. Композиционные методы разработки протоколов на основе сетей Петри : дис.доктора техн. наук : 05.13.11 / Анисимов Николай Александрович. – Владивосток, 1994. –337 с.
9. *Дубинин В.Н.* Языки логического программирования в проектировании вычислительных систем и сетей / В.Н. Дубинин, С.А. Зинкин. –Пенза: Издательство Пенз. гос. техн. ун-та. 1997. –100 с.
10. *Федотов И.Е.* Некоторые приемы параллельного программирования / Федотов И.Е. – М.: Московский государственный институт радиотехники, электроники и автоматики (технический университет), 2008. –188 с.
11. *Kok Mun Ng* Visual Microcontroller Programming Using Extended S-System Petri Nets/Kok Mun Ng, Zainal Alam Haron// WSEAS TRANSACTIONS on COMPUTERS – June 2010 Issue 6, Volume 9, ISSN: 1109-2750
12. *M.R. Frankowiak* Microcontroller-Based Process Monitoring Using Petri-Nets/M.R. Frankowiak, R.I. Grosvenor, P.W. Prickett// Hindawi Publishing Corporation EURASIP Journal on Embedded Systems Volume 2009, Article ID 282708,12pages
13. *Paulo Maciel* A Petri Net Model for Hardware/Software Codesign / Paulo Maciel, Edna Barros, Wolfgang Rosenstiel// Design Automation for Embedded Systems, 4, 243–310 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.
14. *Wilfried Brauer* Carl Adam Petri and"PetriNets""/ Wilfried Brauer , WolfgangReisig// Informatik-Spektrum, Vol. 29,Nr.5, pp 369-374, Springer-Verlag, 2006.
15. *Murata T.* Petri Nets: Properties, Analysis, and Applications" / T. Murata // Proceedings of the IEEE, Vol. 77. –№ 4. P.p. 541-580. 1989.
16. *Питерсон Дж.* Теория сетей Петри и моделирование систем / Питерсон Д. –М.: Мир, 1984. –264 с.
17. *Котов В.Е.* Сети Петри / В.Е. Котов. –М.: Наука, 1984. –160 с.

Поступила 3.02.2014р.

УДК 681.5 (045)

Є. А. Реуцький, Л. М. Щербак, м. Київ

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ МЕТРОЛОГІЧНИХ ХАРАКТЕРИСТИК ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНОЇ СИСТЕМИ

Abstract. In this article discusses the issues of information providing monitoring metrological characteristics of the measuring system, to define its purpose and tasks. An example of the developed method for measuring shows the parameters of the system energy.

Вступ

Інформаційно-вимірювальні системи (ІВС) є потужним вимірювальним інструментарієм розробки сучасних інформаційних технологій, відіграють

