УДК 621.3

И.Г. Дровникова[1], Д.А. Кабанов[2]
([1]Воронежский институт МВД России, [2]В/Ч 28683;
e-mail: idrovnikova@mail.ru)

СПОСОБ ФОРМАЛИЗАЦИИ ПРОЦЕССА ФУНКЦИОНИРОВАНИЯ СПЕЦИАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В ОПЕРАЦИОННОЙ СИСТЕМЕ

Предлагается способ формализации процесса функционирования специального программного обеспечения в операционной системе. Способ основан на применении модели теории марковских процессов.

Ключевые слова: специальное программное обеспечение, прикладная программа, операционная система.

I.G. Drovnikova, D.A. Kabanov

WAY OF FORMALIZATION OF PROCESS OF FUNCTIONING OF THE SPECIAL SOFTWARE IN THE OPERATING SYSTEM

The way of formalization of process of functioning of the special software in an operating system is proposed. The way is based on application model of the theory of Markov processes.

Key words: special software, applied program, operating system.

Выполнение ***прикладной программы (ПП)*** в многозадачной ***операционной системе (ОС)*** представляется в виде процесса, имеющего свое адресное пространство, используемое только этим процессом [1]. Процесс, в свою очередь, выполняется посредством более мелких единиц выполнения – потоков. Каждый поток имеет различные атрибуты, указывающие на принадлежность к процессу, текущее состояние потока и т.п. Процесс в многозадачной ОС сводится к выполнению потоков в последовательности, определяемой ОС согласно приоритету каждого потока.

Каждая ПП в составе ***специального программного обеспечения (СПО)*** может содержать в себе код, который при выполнении образует потоки различного типа. Число таких типов потоков строго фиксировано в рамках программы (процесса) и зависит от алгоритма. В процессе выполнения программы могут быть приведены в исполнение как все возможные потоки, так и выборочно различная комбинация тех или иных потоков. Это зависит от степени интерактивности программы, в частности от действий пользователя или от предлагаемого набора исходных данных. Число потоков одного типа может быть переменным, как и время, затраченное процессором на выполнение различных потоков одного типа. В свою очередь, каждый поток содержит определенное число команд процессора (инструкций), которые могут быть сгруппированы в виде отдельных подпрограмм (функций). Эти функции компонуются в отдельные библиотеки, поэтому каждый поток содержит в себе обращения к той или иной библиотечной функции.

Таким образом, каждый поток содержит в себе строго фиксированное в зависимости от алгоритма множество библиотечных функций, при выполнении которых процессор находится в разных режимах работы (реальном или защищенном).

В свою очередь, каждая функция представляет собой последовательность команд, выполняемых процессором в рамках того или иного потока.

В вычислительной системе одновременно могут выполняться несколько ПП, входящих в состав СПО. На рис. 1 приведен пример декомпозиции процесса при выполнении ПП.

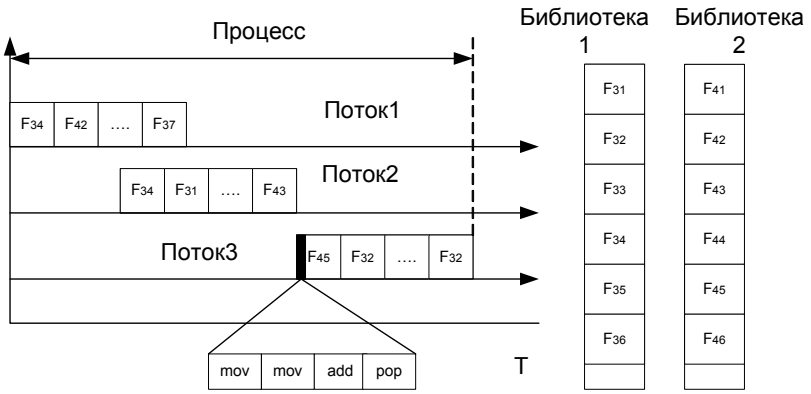

Рис. 1. Декомпозиция процесса выполнения ПП

Принимая во внимание тот факт, что каждый процесс выполняется в своем адресном пространстве, а потоки всех существующих в настоящий момент времени в системе процессов получают квант процессорного времени только в соответствии с собственным приоритетом, независимо от принадлежности к процессу, модель выполнения ПП в многозадачной ОС можно упростить.

Выполнение ПП можно рассматривать вне зависимости от наличия в ОС других процессов. Потоки выполняемого процесса могут иметь равный приоритет и выполняться по очереди, в порядке их появления. Кванты времени, выделяемые для выполнения каждому потоку, могут иметь одинаковую длительность. Самой минимальной единицей исполнения являются команды процессора. В зависимости от процессора, используемого в вычислительной системе, общее число команд может быть разным. Например, для процессоров семейства *Intel* количество команд превышает 300 [2]. Для модели выполнения ПП общее число команд можно ограничить. Наиболее важными для потока команд являются команды, позволяющие менять последовательность выполнения программы. К таким командам относятся команда условного перехода, команда безусловного перехода, а также команды вызова подпрограмм и возврата из них. Проведённый анализ различных библиотечных функций показывает, что каждая из них, как правило, содержит несколько ветвей выполнения. Команды процессора часто группируются в более крупной единице выполнения – подпрограмме (функции).

Следующий уровень агрегирования – поток. В потоке есть команды процессора, которые осуществляют вызов различных подпрограмм. Таким образом, поток состоит из набора функций, вызываемых в контексте этого потока. Так как последовательность вызовов функций в каждом потоке индивидуальна, то она может служить в качестве индивидуальной особенности, которую можно использовать для подтверждения правильного хода выполнения потока. Так как любой поток содержит в себе, как правило, несколько возможных ветвей выполнения, то появление тех или иных функций в нем будет носить вероятностный характер.

Появление в процессе потоков различного типа также может носить вероятностный характер, за исключением случая, когда процесс имеет единственный поток. Следовательно, правильность выполнения процесса может быть подтверждена появлением в рамках этого процесса санкционированных потоков, которые в процессе своего выполнения не имеют отклонений от заданного маршрута.

Таким образом, любая ПП при выполнении имеет свои индивидуальные признаки, позволяющие однозначно её идентифицировать. Например, результаты деассемблирования исполняемого файла, а также прогона программы в отладочном режиме с помощью программы *W32Dasm Version* 8.93 показывают, что текстовый процессор *Word* в процессе выполнения создает четыре потока, импортирует 659 функций из 9 библиотечных модулей и экспортирует в свою очередь 84 функции.

Для формального описания процесса выполнения ПП в ОС введем следующие обозначения [1]:

1. $Z = \{z_1, z_2, \ldots, z_i, \ldots, z_N\}$ – множество процессов, санкционированных для выполнения в ОС, где N – общее число ПП, входящих в состав СПО.

2. $\Pi = \{n_1, n_2, \ldots, n_j, \ldots, n_M\}$ – конечное соответствующее множеству Z множество типов потоков, где M – общее число типов потоков в ОС. В рамках одного типа количество потоков может быть переменным и ограничиваться ресурсами ОС.

3. $C = \{c_1, c_2, \ldots, c_k, \ldots, c_o\}$ – множество функций системных библиотек ОС, предназначенных для использования в рамках одного из потоков множества П, где O – общее число библиотечных функций для данной ОС.

4. $K_П = \{h_1, h_2, \ldots, h_n\}$– алфавит или набор машинных команд $(K_П)$ (команд процессора). Общее число машинных команд n ограничено и зависит от аппаратной реализации ОС (в частности типа процессора).

Формально ПП (z_i) может быть представлена в виде:

$$z_i = \{П_i, C_i, K_П\},$$

где $П_i$ – вектор конечного числа потоков, принадлежащих данному процессу, $П_i \in \Pi$;

C_i – вектор конечного числа функций, принадлежащих данному процессу;

$K_П$ – множество команд процессора.

Как было показано ранее, количество команд, используемых при выполнении ПП, достаточно велико, что определяет целесообразность использования при моделировании процесса выполнения ПП сокращенного множества из 50 основных команд. Такое число команд позволяет моделировать исполняемый код программы с максимальным приближением к оригиналу и будет минимизировать затраты на выполнение вычислений. Данное предположение основано на результатах проведенного анализа, которому были подвергнуты несколько библиотечных функций, выбранных случайным образом из библиотек, составляющих ядро ОС.

Из 50 команд, выбранных для модели, 4 команды имеют особое назначение. Они предназначены для изменения порядка выполнения программы. К их числу относятся:

- команда вызова подпрограммы;
- команда возврата из подпрограммы;
- команда безусловного перехода;
- команда условного перехода.

В реальном наборе команд существует несколько вариантов команд условного перехода в зависимости от постановки условия [2] (например, переход "если больше", переход "если не равно" и т.п.). Для моделирования условного перехода достаточно иметь одну команду, выбор направления перехода в которой осуществляется случайным образом при помощи генератора случайных чисел. Распределение случайных чисел может быть задано неравномерно согласно матрице переходных вероятностей, имитирующей реальную или абстрактную программу, подготовленную к использованию в модели.

Реальная программа (функция) имеет разветвленную структуру, основу которой составляют линейные участки кода. Переход от одного участка к другому носит вероятностный характер, следовательно, существует множество вариантов маршрутов исполнения программы (функции). Длительность выполнения любого варианта зависит не только от числа пройденных линейных участков и длины каждого из них, но и от числа повторений (цикличности) отдельных участков программы (функции).

Следовательно, для моделирования процесса выполнения ПП необходимо обеспечить возможность выполнения программ по нескольким ветвям, выбор которых определяется некоторым распределением вероятностей. Вероятность выполнения каждой отдельной ветви можно получить, используя выражение [3]:

$$\mathrm{P}(V) = \mathrm{P}(\mathrm{A}_1) \cdot \mathrm{PA}_1(\mathrm{A}_2) \cdot \mathrm{PA}_1\mathrm{A}_2(\mathrm{A}_3) \ldots \mathrm{PA}_1\mathrm{A}_2 \ldots \mathrm{A}_{n-1}(\mathrm{A}_n),$$

где А – линейный участок программы;

$P(A)$ – вероятность попадания программы на данный линейный участок;

$PA_i(A_i)$ – условная вероятность того, что событие A_j произойдет после события A_i;

$V = \{A_1, A_2, \ldots, A_n\}$ – ветвь программы (функции).

Как было показано выше, выполнение ПП может быть представлено как выполнение отдельных потоков, что приводит к необходимости моделировать отдельные потоки, имеющие свои индивидуальные отличия. С этой целью необходимо описать структуру потока таким образом, чтобы она моделировала алгоритм, а также определить способ хранения, чтобы обеспечить возможность повторного выполнения.

Таким образом, формальное описание процесса функционирования СПО в ОС требует моделирования хода выполнения ПП, а именно, разработки и применения модели теории марковских процессов (математического аппарата марковской модели и скрытой марковской модели).

Литература

1. ***Модели*** и алгоритмы контроля "защищенности" прикладного программного обеспечения АСУ критического применения: монография / Бочков М.Б., Кабанов Д.А., Ланкин О.В. и др. Воронеж: Воронежский государственный технический университет, 2008. 138 с.
2. ***Бронштейн И.Н., Семендяев К.А.*** Справочник по математике. М.: Наука, 1986.
3. ***Каллан Р.*** Основные концепции нейронных сетей. М.: Изд. дом "Вильямс", 2001.

Статья опубликована 30 мая 2013 г.