УДК 629.783.051:681.3

К. А. Кондратьев, Н. Н. Шумаков

УПРАВЛЕНИЕ ДИНАМИЧЕСКОЙ ПАМЯТЬЮ В БОРТОВОМ ПРОГРАММНОМ ОБЕСПЕЧЕНИИ СПУТНИКОВ СВЯЗИ И НАВИГАЦИИ

Рассматриваются основные принципы управления динамической памятью в бортовом программном обеспечении спутников связи и навигации. Предлагаются изменения, повышающие эффективность и надежность управления и распределения динамической памяти.

Ключевые слова: управление динамической памятью.

Одной из основных функций, выполняемой любой операционной системой (ОС), является управление памятью, а именно: контроль и слежение за свободной частью памяти, выделение запрашиваемой процессами или задачами требуемой им памяти, а также освобождение выделенной ранее памяти в процессе работы или после ее завершения.

В бортовом программном обеспечении (БПО) динамической памятью является зарезервированная область памяти в оперативном запоминающем устройстве (ОЗУ), используемая для выделения памяти задачам в процессе их выполнения. Эту область памяти также называют «кучей» (pool или heap). Память выделяется для сохранения стеков задач или динамического создания системных объектов, в том числе дескрипторов фрагментов памяти.

При реализации алгоритмов выделения памяти неизбежно возникают проблемы оптимизации процесса поиска свободной памяти, фрагментации памяти, защиты структур управления динамической памятью от искажения.

Рассмотрим некоторые аспекты реализации существующих систем, алгоритмов и стратегий поиска свободных фрагментов, а также пути разрешения проблем фрагментации памяти и оптимизации основных операций работы с динамической памятью.

В зависимости от решаемой задачи используются различные алгоритмы поиска свободного фрагмента памяти. В случае если программы требуют фрагментов памяти фиксированной величины, алгоритмы поиска свободного фрагмента максимально упрощены [1]. Но если запрашиваются блоки различных величин и при этом они освобождаются в свободном порядке, то возникает проблема фрагментации свободного пространства.

Каждому выделяемому фрагменту памяти ставится в соответствие дескриптор, который описывает размер блока и представляет другую дополнительную информацию. Дескриптор может быть как отдельным объектом в памяти, так и частью самого выделяемого фрагмента, при этом он может быть в виде заголовка фрагмента или окружать фрагмент с его начала и окончания (например, при использовании алгоритма парных меток) [2].

Иногда свободные фрагменты памяти объединяются в двунаправленный список, что является полезным при объединении двух стоящих рядом свободных фрагментов.

Поиск подходящего фрагмента из списка может вестись по двум принципам:

– нахождения первого подходящего фрагмента, если размер свободного фрагмента меньше требуемого или равен ему;

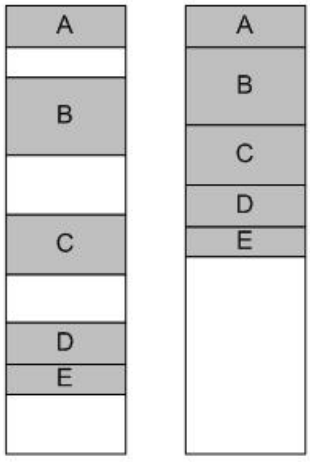
– нахождения наиболее подходящего фрагмента, если размер фрагмента максимально близок к требуемому.

Принцип нахождения первого подходящего фрагмента порождает специфическую проблему. Если каждый раз просматривать список с одного и того же места, то крупные блоки в начале списка будут делиться чаще. Соответственно мелкие блоки будут иметь тенденцию скапливаться в начале списка, что увеличивает среднее время поиска свободного фрагмента в дальнейшем. Эту проблему может разрешить поочередный поиск по двум направлениям либо с применением кольцевого списка фрагментов, при котором поиск нового фрагмента начинается со следующего после последнего фрагмента, найденного при предыдущем поиске.

Принцип нахождения наиболее подходящего фрагмента требует пересмотра всей очереди, кроме того, он повышает уровень фрагментации памяти, так как каждый раз при выделении памяти из максимально подходящего фрагмента остается максимально малый свободный «хвост». Таким образом, этот принцип порождает большое количество мелких свободных фрагментов, мало пригодных при дальнейшем процессе выделения памяти.

В ситуациях когда требуется выделять фрагменты памяти нескольких фиксированных размеров, принцип поиска наиболее подходящего фрагмента более эффективен. В остальных ситуациях рекомендуется использовать принцип поиска первого подходящего фрагмента.

Как мы уже отмечали, одной из проблем, возникающих при организации динамической памяти, является проблема фрагментации памяти.



а *б*

Рис. 1. Дефрагментация (уплотнение) памяти:
а – память до уплотнения; *б* – память после уплотнения

Разрешить эту проблему можно с помощью процесса дефрагментации (рис. 1), в рамках которого применяются две операции:

– слияние – объединение двух рядом стоящих свободных фрагмента;

– уплотнение – организация перемещения занятых фрагментов памяти в последовательную цепочку [3].

Уплотнение «кучи» является достаточно медленной операцией, однако она укорачивает ссылочные цепочки фрагментов, что положительно сказывается на производительности при дальнейшей работе с ней [4].

В объектно-ориентированных языках организуется специальный метод, называемый «сборкой мусора», суть которого состоит в контроле над ненужными объектами в динамической памяти. Фрагмент памяти считается свободным, если нет ни одной ссылки, указывающей на объект, память под который была выделена в виде данного фрагмента. При этом процесс «сборки мусора» проводится только в момент, когда в «куче» обнаружено отсутствие свободной области памяти требуемого размера [5].

В ряде операционных систем, в частности семейства Windows, допускается использование нескольких «куч» памяти, что позволяет получить ряд преимуществ в зависимости от приложений и структур данных, которые они помещают в эти «кучи» [6]. При организации нескольких «куч» памяти вводятся дескрипторы, которые описывают их размер, максимально выделяемый размер фрагментов памяти, параметры доступа и другие особенности, связанные со специализацией «куч» (рис. 2).

В целях повышения эффективности процесса выделения динамической памяти «кучи» специализируют по определенным критериям. Основным критерием специализации является размер выделяемых фрагментов. Если заранее известно, что из «кучи» будут выделяться фрагменты строго фиксированной длины, то появляется возможность построить алгоритмы быстрого поиска свободного фрагмента. И даже в минимальном случае специализация областей памяти на «кучи» для мелких и крупных фрагментов позволит провести их дефрагментацию и повысить скорость выделения памяти.

При организации поиска в «куче» памяти свободного фрагмента подходящего размера в БПО спутников связи и навигации применяется принцип поиска до нахождения первого подходящего фрагмента, обеспечивающий быстрое нахождение свободного фрагмента. Если выделение нового фрагмента оставит свободную область раз-

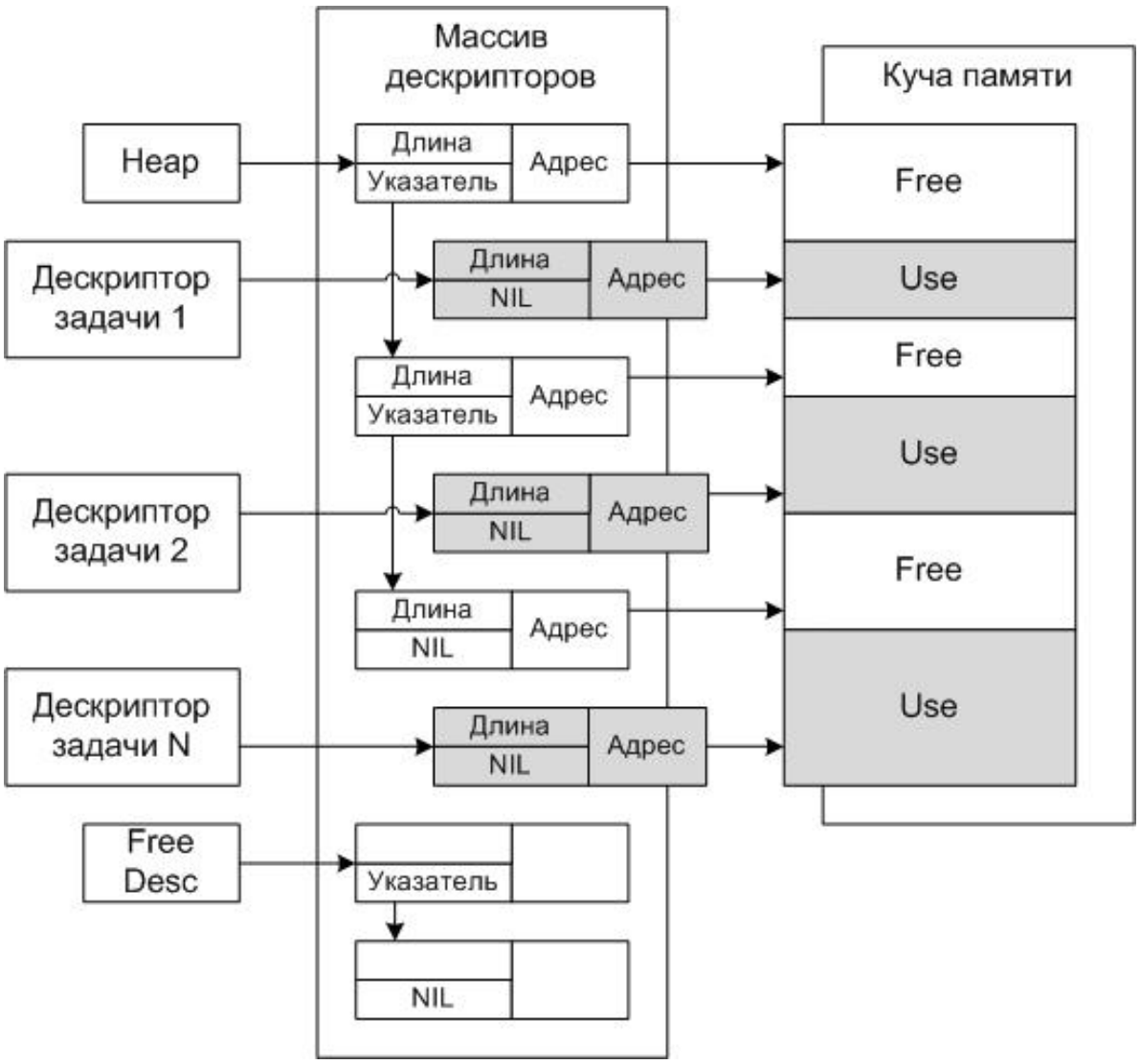


Рис. 2. Структура «кучи» памяти при организации фрагментов с дескрипторами

мером меньше 128 байт, то фрагмент памяти будет выделяться полностью без деления на область требуемого размера и остатка от нее.

Динамическая память в БПО используется:

– в ОС для выделения памяти под организацию стека при ее запуске на выполнение;

– для хранения динамических данных во время выполнения прикладных задач;

– для поддержки конструкций языка Модула-2, работающих с динамической памятью.

Для организации работы с динамической памятью ОС организует следующие объекты:

– MemoryHeap – непосредственно выделенную область динамической памяти;

– Array_DMem – массив дескрипторов фрагментов памяти;

– Heap – адрес начала списка фрагментов свободной динамической памяти;

– FreeDesc – адрес начала списка свободных дескрипторов фрагментов памяти.

Фрагментом памяти называется область памяти, описываемая дескриптором. В дескрипторе фрагмента находятся адрес расположения области памяти и ее длина. Все дескрипторы фрагментов собраны в массиве дескрипторов Array_DMem.

Свободная динамическая память представлена в виде однонаправленного списка фрагментов свободной памяти, упорядоченных по возрастанию адресов. Адрес начала списка определяется переменной heap.

Свободные (незадействованные) дескрипторы фрагментов также организованы в однонаправленный список, адрес начала которого находится в переменной FreeDesc. При необходимости задействовать дескриптор фрагмента он берется первым из списка. При возврате дескриптора после использования он ставится первым элементом.

Операционная система спутников связи и навигации реализует следующие основные функции при работе с динамической памятью:

– выделение памяти;

– возврат памяти;

– инициализацию;

– защиту от искажения;

– документирование;

– настройку для конкретного применения.

Процесс выделения памяти начинается с поиска свободного фрагмента, размер которого больше или равен запрашиваемому. Найденный фрагмент корректируется путем уменьшения размера описываемой им памяти, выделяемой с конца описываемой им области, после чего выделенная область оформляется свободным дескриптором фрагментов памяти. В дескрипторе задачи указывается адрес дескриптора полученного фрагмента. Для борьбы с фрагментацией динамической памяти вводится константа «Размер минимального свободного фрагмента». Если выделение нового фрагмента памяти породит свободную часть, размер которой меньше указанной константы, то фрагмент памяти предоставляется задаче полностью, без деления на две части.

При возврате памяти после ее использования дескриптор фрагмента ставится в список свободных в соответствии с адресом области, которую он описывал. При этом стоящие подряд свободные фрагменты сливаются, формируя один дескриптор свободного фрагмента памяти, описывающий область суммарного размера, и один незадействованный дескриптор, возвращаемый в список свободных дескрипторов.

Операция инициализации динамической памяти обеспечивает начальную подготовку объектов управления памяти к работе. При инициализации динамической памяти в БПО вся область динамической памяти описывается одним фрагментом свободной памяти (список Heap из одного элемента). Остальные дескрипторы фрагментов организуются в список незадействованных.

При работе системы управления динамической памятью ее данные контролируются на корректность, адреса дескрипторов фрагментов проверяются на принадлежность массиву, а содержимое фрагментов – на корректность адреса и длины описываемой им области. При обнаружении искажения выполняется инициализация системы управления динамической памятью с формированием контрольно-отчетной информации, при этом все задачи снимаются с обслуживания.

В процессе работы системы управления динамической памятью формируется следующая контрольно-отчетная информация:

– размер используемой памяти из «кучи»;

– максимальная величина этой памяти;

– число выделенных фрагментов памяти;

– максимальное число выделенных фрагментов памяти.

Система управления памятью при генерации ОС настраивается для конкретного применения путем указания констант настроек:

– размера «кучи» памяти;

– числа дескрипторов фрагментов памяти;

– минимального размера выделяемых фрагментов.

Существующая ОС достаточно хорошо реализует все основные функции по управлению динамической памятью, однако при всех ее достоинствах она ограничена в количестве дескрипторов фрагментов памяти, что приводит к невозможности выделения нового фрагмента даже в случае наличия свободной памяти.

Разрешить эту проблему может такая схема организации фрагментов памяти, при которой дескриптор фрагмента является частью самого фрагмента в виде его заголовка (рис. 3). Однако основное функциональное назначение динамической памяти в БПО состоит в выделении памяти под стеки запускаемых на выполнение задач, в случае же переполнения стека задачей происходит разрушение списка фрагментов, так как записываемая в стек информация затирает заголовок идущего следом фрагмента, что приводит к нарушению вычислительного процесса. Поэтому в настоящее время такая схема организации динамической памяти в БПО не применяется. Однако ее можно использовать, создав следующие «кучи»:

– системную «кучу» для динамического создания системных объектов, в том числе дескрипторов фрагментов памяти с организацией фрагментов с заголовками. Системная «куча» эффективнее с точки зрения быстроты выделения и надежнее с точки зрения отсутствия опасности ее искажения;

– «кучу» для стеков задач и запрашиваемой пользователями памяти, которая организуется по существующей схеме с дескрипторами системной «кучи».

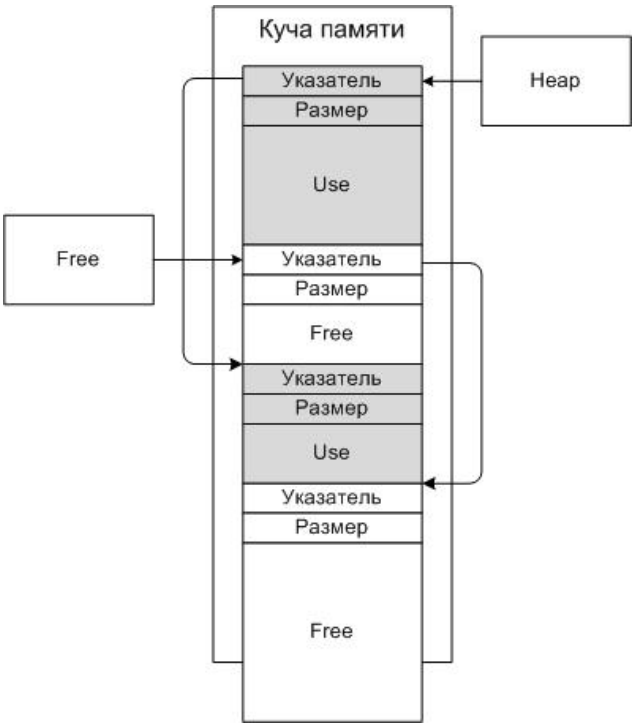


Рис. 3. Структура «кучи» памяти
с организацией фрагментов с заголовками

В процессе проведения испытаний были получены качественные характеристики эффективности рассматриваемых схем организации динамической памяти, которые свидетельствуют о преимуществе схемы с применением специализации «куч» над схемой организации динамической памяти с применением фрагментов с дескрипторами (см. таблицу).

Схема специализации с разделением динамической памяти на «кучу» для стеков и «кучу» для системных целей приводит к ускорению работы с динамической памятью за счет уменьшения длины списков выделенных фрагментов, а также разрешает проблему невозможности выделения памяти из «кучи» в случае наличия в ней свободной области требуемого размера при отсутствии свободного дескриптора фрагмента.

Библиографические ссылки

1. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация / под общ. ред. А. Матросова. СПб. : Питер, 2002.

2. Иртегов Д. В. Введение в операционные системы. 2-е изд. СПб. : БХВ-Петербург, 2008.

3. Цикритзис Д., Бернстайн Ф. Операционные системы. М. : Мир, 1977.

4. C# 2005 и платформа .NET 3.0 для профессионалов / К. Нейгел, Б. Ивьен, Д. Глин и др. М. : Вильямс, 2008.

5. Рихтер Дж. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Мастер-класс. СПб. : Питер, 2007.

6. Харт Д. М. Системное программирование в среде Windows. 3-е изд. М. : Вильямс, 2005.

Временные характеристики функций библиотек обслуживания памяти, реализующие схемы организации динамической памяти (значения приведены в тактах)

Функция	Схема с применением фрагментов с дескрипторами			Схема с применением специализации «куч»		
	Мин.	Ном.	Макс.	Мин.	Ном.	Макс.
Запрос памяти	2 199	3 210	3 535	2 184	2 520	3 140
Освобождение памяти	3 454	3 454	3 891	3 015	3 015	3 580
Инициализация	5 004			5 052		

C. A. Kondratev, N. N. Shumakov

DYNAMIC MEMORY CONTROL IN NAVIGATION AND COMMUNICATION SATELLITES ON-BOARD SOFTWARE

The article considers the basic principles of dynamic memory control in satellites on-board software, and proposes some changes increasing effectiveness and reliability of dynamic memory control and allocation.

Keywords: dynamic memory control.