Обработка исключительных ситуаций с использованием библиотеки мониторинга

А.И. Грюнталь[1], К.Г. Нархов[2], А.М. Щегольков[3]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН».
E-mail's: [1]grntl@niisi.ras.ru, [2]kostas@niisi.ras.ru, [3]ashch@niisi.ras.ru

Аннотация: Статья посвящена вопросам реализации средств управления обработкой исключительных ситуаций в многопоточной прикладной программе в среде операционной системы реального времени, функционирующей на многопроцессорной вычислительной системе. Рассматривается архитектура библиотеки мониторинга, а также технологические аспекты ее применения.

Ключевые слова: контролируемое выполнение, исключительная ситуация, исключение, библиотека мониторинга, многопоточная программа, поток управления, сигнал, операционная система реального времени, многопроцессорная система.

Введение

Исключительная ситуация (exception) – состояние внешних данных, устройств ввода-вывода или вычислительной системы в целом, в котором дальнейшее выполнение прикладной программы становится невозможными или бессмысленными. Классические примеры подобных ситуаций: деление на нуль, ошибка чтения данных с внешнего устройства, исчерпание ресурсов (память, дисковое пространство), прием сигнала выключения питания [1].

Программист может парировать подобные ситуации, используя структуры типа **`if-else`**, в которых реализована проверка критических условий выполнения, однако в этом случае алгоритм прикладной программы теряет прозрачность, обрастает многочисленными проверками и блоками обработки ошибок. Также следует принять во внимание, что требуемые для выполнения проверок библиотечные функции (или соответствующие системные вызовы ОС РВ) могут отсутствовать, и программист оказывается перед необходимостью разработки дополнительного функционала.

В состав ОС РВ Багет 2.6 входит гибкий механизм регистрации и обработки исключительных ситуаций на уровне ОС. Использование этого механизма позволяет программисту реализовать принцип контролируемого выполнения [2] в прикладной программе. Однако, чтобы избежать «ложного чувства безопасности» [5], программисту следует максимально точно определить множество возможных исключительных ситуаций и разработать соответствующие достоверные обработчики таких ситуаций (с полным тестовым покрытием, обеспечивающим высокую надежность кода). С точки зрения трудоемкости эта задача сравнима (или более трудоемка) с парированием исключительных ситуаций посредством конструкций **`if-else`**, о котором говорилось выше.

Библиотека мониторинга (БМ) [2, 6] реализует функции, которыми должна обладать система, спроектированная в соответствии с принципами контролируемого выполнения [2, 3, 4]. В соответствии с [2], программа с контролируемым выполнением – это программа со встроенными механизмами, обеспечивающими выполнение программы в критических условиях (например, при поступлении в программу некорректных данных, ошибок в среде исполнения или в самой прикладной программе). Средства контролируемого выполнения прикладной программы содержат механизмы, обеспечивающие контроль состояния выполняемой программы, сравнение параметров состояния прикладной программы с эталоном, а также генерацию управляющих воздействий, в случае отклонения поведения программы от эталона.

Библиотека мониторинга выполняет контроль работоспособности, корректности и целостности прикладной программы на основании данных, получаемых от агентов

мониторинга. При этом регистрация и обработка ошибок являются задачами, решаемыми прикладным программистом, при этом, однако, он может использовать библиотечные функции БМ, в том числе и инструменты протоколирования и восстановления, предоставляемые БМ.

В данной статье рассматриваются приемы и решения использования механизма управления исключительными ситуациями в прикладной программе средствами БМ.

1 Архитектурные особенности обработки исключений с использованием БМ

Архитектура БМ и структура прикладной программы, работающей совместно с библиотекой мониторинга на одном процессоре, рассмотрена в [2]. Структура прикладной программы, работающей совместно с библиотекой мониторинга на многопроцессорной системе, рассмотрена в [6].

На рисунке 1 представлена схема взаимодействия БМ с прикладной программой на уровне функций (API).

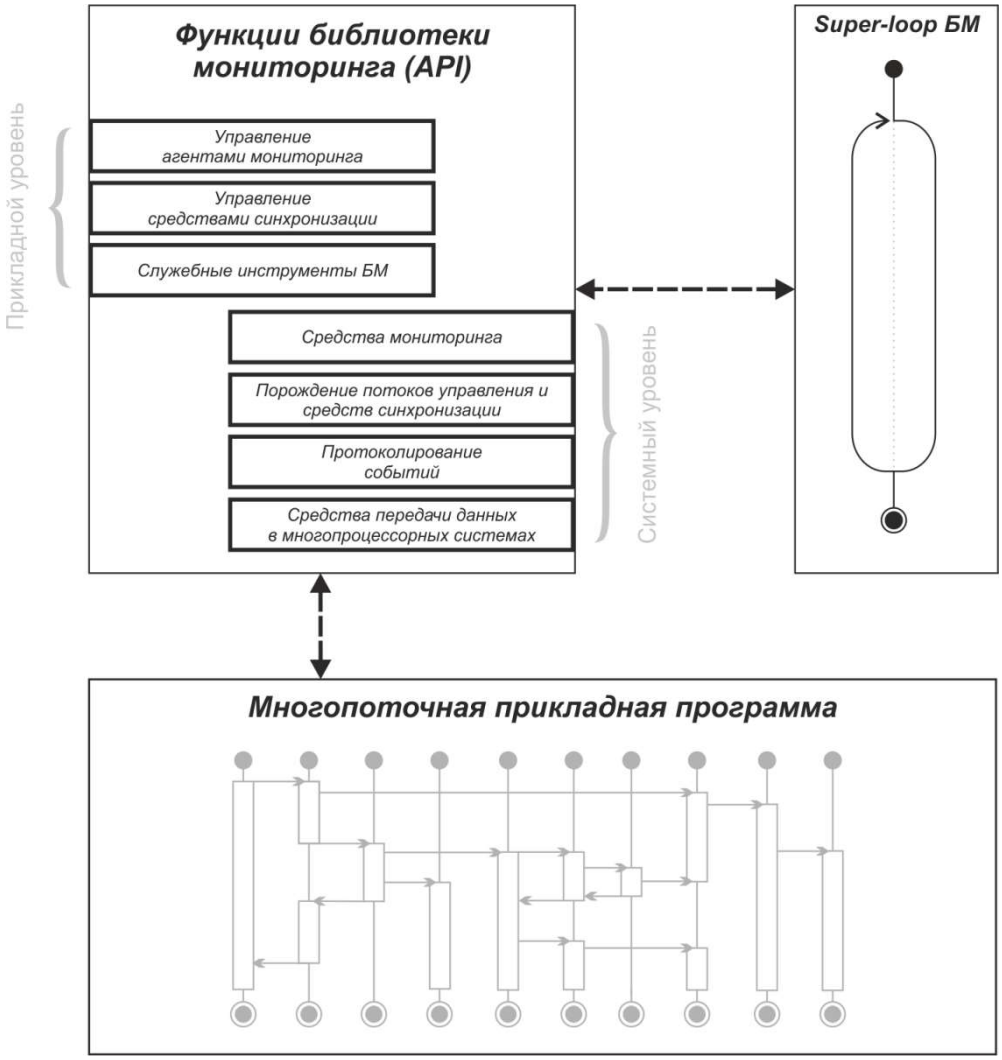

Рисунок 1 – Схема взаимодействия БМ и прикладной программы

При старте ОС РВ управление передается головной функции БМ, реализующей **`super-loop`** [7], в котором выполняется последовательное обращение к функциям БМ системного уровня (см. рисунок 1). Эти функции в соответствии с заданной конфигурацией прикладной программы выполняют порождение потоков

управления, средств синхронизации, мониторинга и протоколирования [6].

Возникновение исключительной ситуаций в потоке управления прикладной программы (т. е. на прикладном уровне) приведет к приостановке этого потока; БМ зарегистрирует отказ потока по таймауту (агент мониторинга в заданном потоке управления перестанет информировать локальный монитор о своей работе), при этом причину отказа, способ восстановления и время отказа установить невозможно.

Для обработки исключительных ситуаций в БМ реализован дополнительный набор функций (так называемый «слой исключений», представленный на рисунке 2).

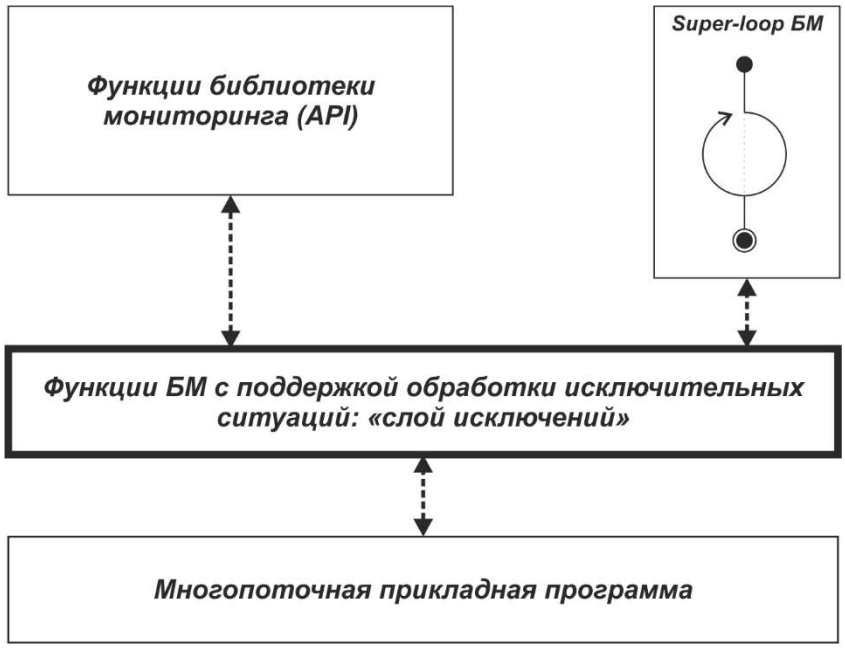

Рисунок 2 – Схема взаимодействия БМ и прикладной программы с дополнительным набором функций

Из рисунка 2 следует, что прикладная программа, а также головная функция БМ, обращаются к API БМ через «слой исключений», функции которого обеспечивают управление исключительными ситуациями (регистрация и обработка). Таким образом, исключительная ситуация, возникшая при выполнении прикладной программы, будет перехвачена БМ, которая запротоколирует общую информацию о возникшей исключительной ситуации и попытается восстановить работоспособность прикладной программы (сформирует реакцию на исключительную ситуацию).

2 Структура функции с обработкой исключительных ситуаций

В соответствии с документацией ОС РВ Багет 2.6 [1] в прикладной программе можно использовать гибкий способ обработки исключительных ситуаций аналогичный конструкции **`TRY-CATCH`** в C++, применимый как в потоках управления, так и в функциях обработки прерываний. Для этой цели предназначены макрооперации **`tryBegin()`**, **`tryCatch()`**, **`tryEnd()`**.

Таким образом, универсальный «скелет» функций из «слоя исключений» БМ соответствует исходному тексту, представленному в листинге 1.

```
#include <try.h>
int _func1_exept( void ) {
    tryBegin();
        func1();
    tryCatch();
      struct tryINFO *p;
      tryInfo(p);
        checkpoint_agent(
                    p->cause);
        return -1;
    tryEnd();
    return 0;
}
```

Листинг 1 – Исходный текст функции БМ с обработкой исключительных ситуаций

В примере из листинга 1 при возникновении исключительной ситуации в функции **func1()** управление передается оператору, непосредственно следующему за **tryCatch()**. Если при выполнении функции **func1()** исключительная ситуация не возникнет, то участок программы между **tryCatch()** и **tryEnd()** не будет исполняться и передается оператору, непосредственно следующему за **tryEnd()**.

3 Агенты мониторинга

Использование библиотеки мониторинга предполагает, что программист выполнил проектирование системы в терминах потоков управления, а именно – определил количество групп функциональных потоков управления, количество потоков управления, входящих в каждую группу, и разработал для каждого потока управления итеративную функцию, выполняющуюся в теле цикла, на языке Си [2]. При этом в итеративных функциях должны быть определены точки останова, и в этих точках должны быть размещены вызовы агентов мониторинга – функции прикладного уровня БМ **ckeckpoint_agent()**.

Реализация «слоя исключений» позволяет упростить интеграцию БМ в прикладную программу для «быстрого старта»: не требуется размещение агентов мониторинга в прикладных функциях. Мониторинг прикладной программы в этом случае заключается в регистрации исключительных ситуаций и самовосстановлении системы.

Следует отметить, что в блоке **tryCatch() - tryEnd()** может быть реализован запуск пользовательского обработчика исключительной ситуации, который должен быть определен для заданной головной функции на этапе конфигурирования БМ [6]. Кроме того, в **tryCatch() - tryEnd()** может быть выполнен стандартный сценарий: протоколирование системных структур БМ, связанных с заданным потоком управления, или повторный запуск потока управления.

4 Отслеживание значений переменных

Механизм обработки исключительных ситуаций позволяет реализовать проверку, например, значений целочисленных переменных непосредственно в процессе выполнения прикладной программы. Эта проверка реализована по аналогии с функцией **assert()**. Метод проверки состоит в том, что в «слой исключений» вводится функция **is()**, в которую в качестве аргумента передается контролируемая переменная (см. листинг 2)

```
int is(int retcode,
const int etalon, int causecode);
```

Листинг 2 – Прототип функции отслеживания значений целочисленных переменных

В функции выполняется соответствующая проверка, и в случае отказа (результат проверки – ложь) возбуждается исключительная ситуация посредством системной функции **throwException()**. Пример реализации функции **is()** представлен в листинге 3.

```
#include <try.h>
void is(int retcode,
        const int etalon,
        int causecode ) {
   if (retcode != etalon) {
     throwException(
           causecode,
           (void*)NULL );
   }
}
```

Листинг 3 – Пример реализации функции **is()**

5 Выводы

Представленная в статье технология управления исключительными ситуациями позволяет повысить надёжность приложений, использующих библиотеку мониторинга, упростить их отладку и увеличить точность регистрации ошибок и отказов. Применение данной технологии приводит к созданию отказоустойчивых программ, гарантирующих обработку исключительных ситуаций. Технология освобождает разработчика от рутинных действий по использованию типовых конструкций операционной системы.

Изложенный в статье подход применим как к однопроцессорным, так и к многопроцессорным системам. Создание библиотеки мониторинга выполнялось в рамках работ ФГУ ФНЦ НИИСИ РАН по технологии разработки систем реального времени.

Handling Exceptions Using the Monitor Library

A. Gryuntal, K. Narkhov, A. Shchegolkov

Abstract: The Article is devoted to the implementation of management tools handling exceptional situations in a multi-threaded application program in the real-time operating system environment, functioning on a multiprocessor computer system. Special attention is paid to the monitor library architecture, as well as the technological aspects of its application.

Keywords: controlled execution, exception, monitor library, multithread program, thread, signal, real-time operating system, multiprocessing system.

Литература

1. Безруков В.Л., Годунов А.Н., Назаров П.Е., Солдатов В.А., Хоменков И.И, Введение в ос2000, Вопросы кибернетики. Информационная безопасность, Операционные системы реального времени, Базы данных/Под ред. чл.-корр. РАН В.Б.Бетелина. - Москва; НИИСИ РАН, 1999, – с. 76-106.
2. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Реализация принципа контролируемого выполнения для прикладных программ реального времени. // Труды НИИСИ РАН. – Том 5 № 2. Построение программ реального времени. ISSN 2225-7349, Москва, 2015, с. 113-121.
3. В.А. Галатенко, Контролируемое выполнение / В.А. Галатенко, К.А. Костюхин, К.А., Н.В. Шмырев – М: НИИСИ РАН, 2012. – 157 с.
4. В.Б. Бетелин, Контролируемое выполнение с явной моделью / В.Б. Бетелин, В.А. Галатенко, К.А. Костюхин – ПРОГРАММИРОВАНИЕ, 2014, N- 6, с. 45-55.
5. T. Cargill. Exception handling: A false sense of security. C++ Report, Nov-Dec 1994. Доступно по адресу: http://ptgmedia.pearsoncmg.com/images/020163371x/supplements/Exception_Handling_Article.html
6. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Библиотека мониторинга для многопоточных программ. // Труды НИИСИ РАН. – Том 7 № 1. Построение программ реального времени. ISSN 2225-7349, Москва, 2017, с. 70-74.
7. M. Nahas and A. M. Nahhas, Ways for implementing highly-predictable embedded systems using time-triggered co-operative (TTC) architectures. INTECH Open Access Publisher, 2012.