Применение отладки при разработке программного обеспечения

А. М. Петров

Московский институт стали и сплавов,
Россия, 119049 Москва, Ленинский пр., 4

Статья посвящена отладке, одному из этапов разработки программного обеспечения. Рассматриваются основные методы отладки, их классификация и область применения.

Введение

В современном процессе разработки программного обеспечения довольно значительную часть времени разработчик тратит на отладку. От качества используемого отладочного инструментария и грамотного выбора методов отладки зависит качество и сроки получения программного продукта. Неисправленная вовремя ошибка может свести на нет все усилия по разработке программного продукта. В статье будут рассмотрены некоторые методы отладки, их классификация и область применения.

1. Понятие отладки

Процесс разработки программного обеспечения (ПО) является сложным и многоэтапным. Методы разработки ПО с каждым годом совершенствуются и усложняются. Часто этапы разработки похожи друг на друга и выполняются одними и теми же людьми, поэтому их границы размыты и существует несколько точек зрения на них. Дадим несколько определений и поясним их.

Отладка — этап разработки программного обеспечения, на котором обнаруживают, локализуют и устраняют ошибки.

Тестирование — этап разработки программного обеспечения, на котором обнаруживают факт существования ошибки.

Перечисленные этапы часто переплетаются, но все же разделены — отладка не занимается проверкой на существование ошибок, а в процессе тестирования не следует искать их точное месторасположение и причину.

С развитием программного обеспечения и методов его разработки было создано несколько довольно эффективных методов локализации, устранения и предупреждения ошибки. Дадим краткий обзор основных методов отладки.

Классификация методов отладки:

- протоколирование;
- работа в интерактивном отладчике;
- диагностика в процессе исполнения;
- использование графических средств;
- визуальный просмотр кода;
- анализ crash dump.

Протоколирование — это процесс записи информации о происходящих событиях в протокол. Применительно к программному обеспечению — это запись в хронологическом порядке операций обработки данных в специальный файл, что позволяет впоследствии воссоздать существовавшее на момент записи в протокол состояние программы.

Помимо этого данный способ отладки можно использовать в период эксплуатации программного продукта конечным пользователем. Этот подход прост в реализации и хорошо зарекомендовал себя в системах управления базами данных.

Работа в интерактивном отладчике позволяет:

- устанавливать точки остановки в программе;
- просматривать и редактировать значения переменных;
- исполнять программу по шагам.

Этот метод имеет несколько существенных недостатков:

- Неспособность выявить ошибку синхронизации (например, в многозадачных системах и многопоточных приложениях).
- Чрезвычайно высокая сложность низкоуровневой отладки приложения, написанного на языках высокого уровня.
- Реализации интерактивных отладчиков для высокоуровневых языков часто имеют серьезный недостаток: возможна отладка только так называемой «отладочной» сборки приложения, финальная же сборка может сильно отличаться от фактически отлаживаемой.

Несмотря на все недостатки, это довольно надежный и популярный способ отладки.

Диагностика в процессе исполнения — это различные проверки во время исполнения программы, например:

- проверка состояния и целостности стека;
- проверка переполнения буфера;

- слежение за динамической памятью;
- различные проверки внутренней логики программы.

Код таких проверок может как создаваться самим компилятором в специальных режимах сборки программы, так и быть заложенным в код самой программы (различные макроопределения и asssert()). Также существуют специальные отладочные утилиты, следящие за состоянием памяти.

Этот метод отладки используется только в паре с другим методом, например работой в интерактивном отладчике, так как он способен только на обнаружение некорректного поведения программы, но не на исправление ошибки. Часто с помощью этого метода удается определить только место, где программа ведет себя некорректно, но не причину этой некорректности. Лучше всего этот метод применяется, когда приложение запущено не на машине разработчика, так как позволяет определить поведение приложения в различных условиях работы. В целом метод является скорее тестирующим, чем отладочным.

Использование графических средств. Графические средства не являются самостоятельным методом отладки, они лишь позволяют представить различную информацию более наглядно. Визуализировать можно многое:

- изменение значения какой-то переменной во времени;
- объем используемой памяти и загрузки процессора (известный пример — Диспетчер задач Microsoft Windows);
- структуру сложного объекта (дерева, графа);
- информацию, относящуюся к бизнес-логике приложения, например разметку изображения в задачах распознавания образов.

Кроме того, визуализация производится без прерывания работы приложения, что позволяет видеть весь ход работы в динамике. Очень часто графическая визуализация помогает в анализе эффективности работы программы, поиске узких мест и т. п.

Визуальный просмотр кода представляет собой просмотр и логический анализ кода программного продукта и поиск проблемных мест. Используется вместе со всеми остальными методами отладки. Рекомендуется использовать его всегда, даже до первой компиляции программы. Позволяет избавиться от многих проблем еще на этапе дизайна и разработки.

До возникновения интерактивных средств разработки визуальный просмотр кода и протоколирование были единственными возможными средствами отладки. Развитие компиляторов с функциями проверки синтаксиса и семантики, а также оптимизации кода значительно снизило ценность этого метода. Тем не менее, он остается весьма эффективным средством поиска, устранения и, главное, предупреждения возможных ошибок и некорректного поведения. Только этот метод дает возможность рассмотреть все

сценарии работы приложения и найти потенциально ненадежные места в коде программы.

Для визуального просмотра кода необходимо выполнение нескольких условий:

- Хорошая декомпозиция. Только хорошо спроектированный код, достаточно ясный и простой, с функциями небольшого размера поддается визуальному просмотру. Функция длиной в несколько экранов почти не поддается логическому разбору. В случае переусложненного кода этот объем может сократиться до одной строки, например ((void(*)(void))0)().
- Высокая квалификация программиста, знание особенностей компиляции в ассемблерный код, распределения оперативной и стековой памяти.

Использование визуального просмотра кода хорошо сказывается на коде программного продукта в целом, а также на развитии системы протоколирования и других методов отладки, которые требуют закладывания дополнительной функциональности в код.

Анализ crash dump появился как следствие изобретения механизма обработки исключений. В случае генерации программным продуктом специального исключения, сигнализирующего о неисправимой ошибке, происходит запись всего dump-а памяти приложения (содержимое регистров, состояние стека и др.) в специальный файл для последующего анализа. Используется в сочетании с протоколированием и другими методами отладки.

Каждый из описанных выше методов имеет свою область применения, некоторые методы нерационально или невозможно применять там, где успешно используются другие. Например, при использовании продукта конечным пользователем невозможно применить интерактивный отладчик. Поэтому введем классификацию видов отладки по фактическому месту запуска программного продукта.

2. Классификация видов отладки

Под *непосредственной отладкой* понимается такой процесс отладки, когда отлаживаемое приложение запущено на машине разработчика. Это самый распространенный вид отладки.

Под *удаленной отладкой* понимается такой процесс отладки, когда отлаживаемое приложение запущено на машине пользователя, а отладочный инструмент — на машине разработчика. Удаленная отладка используется:

- при отладке приложения в условиях работы, которые нельзя воспроизвести на компьютере разработчика;
- при отладке ПО для отдельных устройств;

- при отладке кроссплатформенных приложений;
- при отладке распределенных систем.

Очевидно, что нет препятствий к тому, чтобы использовать методы удаленной отладки при непосредственной отладке, но обычно они менее эффективны и связаны с большими временными затратами. Также методы отладки можно сгруппировать по признаку использования дополнительных технических средств:

- не требуют дополнительных средств (визуальный просмотр кода);
- требуют только сравнительно небольших дополнений в исходном коде программного продукта (протоколирование, диагностика в процессе исполнения);
- требуют отдельных самостоятельных программных продуктов-инструментов (интерактивный отладчик, графические средства).

Подробнее рассмотрим отладчики, реализованные в виде отдельных программных продуктов. Они могут обладать следующими свойствами и возможностями:

- возможность просмотра текущих значений переменных;
- возможность изменения текущих значений переменных;
- возможность устанавливать (удалять) контрольные точки и условия остановки программы;
- кроссплатформенность;
- рефлексивность (возможность отладить отладчик им же самим);
- возможность удаленной работы;
- поддержка контекстно-зависимого графического вывода.

Если первые две возможности реализует большинство отладчиков, то остальные свойства встречаются у отладчиков гораздо реже.

3. Место отладки в современном процессе разработки ПО

Рассмотрим каскадную модель разработки (рис. 1).

Можно выделить три этапа, в которых отладка имеет место: реализация, продуктизация, эксплуатация.

Рассмотрим каждый этап подробнее:

Реализация — сам процесс написания кода программы. На этом этапе разработчику уже нужен инструмент контроля корректности работы пусть не целой программы, но ее фрагментов и модулей. Чаще всего используются отладчики, встроенные в среду разработки. На этом этапе уже следует

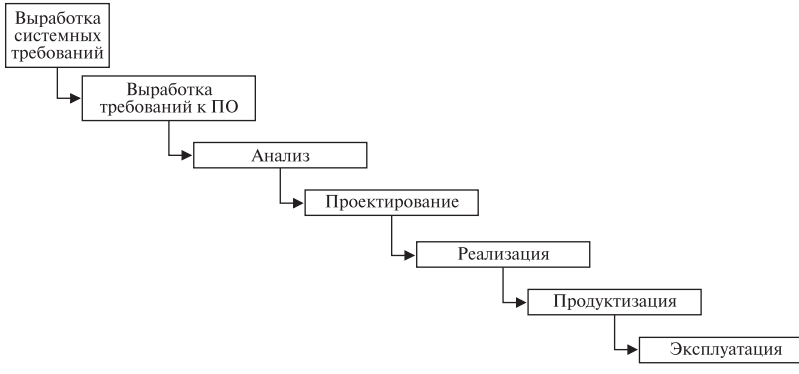

Рис. 1

озаботиться включением в код методов встраиваемого отладчика, если предполагается его использовать. Также рекомендуется использовать метод непосредственного разбора кода.

Продуктизация — процесс выяснения соответствия программного обеспечения заявленным требованиям и приведение к ним. Сюда входят верификационное тестирование, нагрузочное тестирование, отладка.

Часто сюда же входит процесс подбора параметров работы самого программного продукта.

Эксплуатация — процесс непосредственного использования программного продукта. В идеальных условия на этом этапе можно забыть об отладке, если бы не следующие соображения:

- требования к программному продукту могут измениться в любой момент;
- тестирование не может гарантировать абсолютной верности работы программного продукта в любых условиях, всегда может быть что-то пропущено;
- может возникнуть нужда использования программного продукта в условиях, изначально не предусмотренных при разработке.

Литература

Rosenberg J. B. How Debuggers Work: Algorithms, Data Structures, and Architecture. John Wiley & Sons, 1996.