ВикипедиЯ

Отладка программы

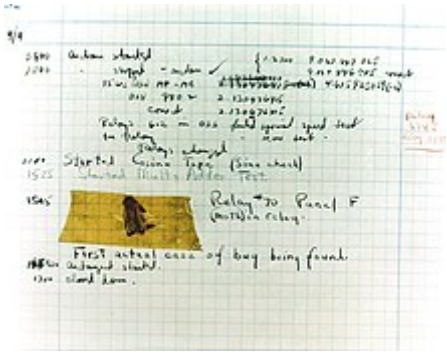
Материал из Википедии — свободной энциклопедии

Отла́дка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится:

- узнавать текущие значения переменных;
- выяснять, по какому пути выполнялась программа.

Существуют две взаимодополняющие технологии отладки.

- Использование отладчиков — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.
- Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода — на экран, принтер, громкоговоритель или в файл. Вывод отладочных сведений в файл называется журналированием.



Запись в журнале компьютера из Марк II, с мотыльком, приклеенным к странице

Содержание

Место отладки в цикле разработки программы

Инструменты

Инструменты отладки

Инструменты, снижающие потребность в отладке

Безопасность программного кода и отладка

См. также

Примечания

Литература

Ссылки

Место отладки в цикле разработки программы

Типичный цикл разработки, за время жизни программы многократно повторяющийся, выглядит примерно так:

1. Программирование — внесение в программу новой функциональности, исправление существующих ошибок.

2. Тестирование (ручное или автоматизированное; программистом, тестировщиком или пользователем; «дымовое», в режиме чёрного ящика или модульное…) — обнаружение факта ошибки.
3. Воспроизведение ошибки — выяснение условий, при которых ошибка случается. Это может оказаться непростой задачей при программировании параллельных процессов и при некоторых необычных ошибках, известных как гейзенбаги.
4. **Отладка** — обнаружение причины ошибки.

Инструменты

Способности программиста к отладке — это, по-видимому, важнейший фактор в обнаружении источника проблемы, но сложность отладки сильно зависит от используемого языка программирования и инструментов, в частности, отладчиков.

Инструменты отладки

Отладчик представляет из себя программный инструмент, позволяющий программисту наблюдать за выполнением исследуемой программы, останавливать и перезапускать её, прогонять в замедленном темпе, изменять значения в памяти и даже, в некоторых случаях, возвращать назад по времени.

Также полезными инструментами в руках программиста могут оказаться:

- Профилировщики. Они позволят определить, сколько времени выполняется тот или иной участок кода. Анализ покрытия позволяет выявить неисполняемые участки кода.
- API логгеры позволяют отследить взаимодействие программы и Windows API при помощи записи сообщений Windows в лог.
- Дизассемблеры позволяют посмотреть ассемблерный код исполняемого файла
- Снифферы помогут отследить сетевой трафик, генерируемый программой
- Снифферы аппаратных интерфейсов позволяют увидеть данные, которыми обмениваются система и устройство.
- Логи системы.

Использование языков программирования высокого уровня обычно упрощает отладку, если такие языки содержат, например, средства обработки исключений, сильно облегчающие поиск источника проблемы. В низкоуровневых языках ошибки могут приводить к незаметным проблемам — например, повреждениям памяти и утечкам памяти. Тогда бывает довольно трудно определить, что стало первоначальной причиной ошибки. В этих случаях могут потребоваться сложные приёмы и средства отладки.

> «Наш личный выбор — стараться не использовать отладчики, кроме как для просмотра стека вызовов или же значений пары переменных. Одна из причин этого заключается в том, что очень легко потеряться в деталях сложных структур данных и путей исполнения программы. Мы считаем пошаговый проход по программе менее продуктивным, чем усиленные размышления и код, проверяющий сам себя в критических точках.
>
> Щёлканье по операторам занимает больше времени, чем просмотр сообщений операторов выдачи отладочной информации, расставленных в критических точках. Быстрее решить, куда поместить оператор отладочной выдачи, чем проходить шаг за шагом критические

> участки кода, даже предполагая, что мы знаем, где находятся такие участки. Более важно то, что отладочные операторы сохраняются в программе, а сессии отладчика переходящи.
>
> Слепое блуждание в отладчике, скорее всего, непродуктивно. Полезнее использовать отладчик, чтобы выяснить состояние программы, в котором она совершает ошибку, затем подумать о том, как такое состояние могло возникнуть. Отладчики могут быть сложными и запутанными программами, особенно для новичков, у которых они вызовут скорее недоумение, чем принесут какую либо пользу…»
>
> «Отладка сложна и может занимать непредсказуемо долгое время, поэтому цель в том, чтобы миновать большую её часть. Технические приёмы, которые помогут уменьшить время отладки, включают хороший дизайн, хороший стиль, проверку граничных условий, проверку правильности исходных утверждений и разумности кода, защитное программирование, хорошо разработанные интерфейсы, ограниченное использование глобальных переменных, автоматические средства контроля и проверки. Грамм профилактики стоит тонны лечения.»
>
> — Брайан Керниган и Роб Пайк

Инструменты, снижающие потребность в отладке

Другое направление — сделать, чтобы отладка нужна была как можно реже. Для этого применяются:

- Контрактное программирование — чтобы программист подтверждал другим путём, что ему на выходе нужно именно такое поведение программы. В языках, в которых контрактного программирования нет, используется самопроверка программы в ключевых точках.
- Модульное тестирование — проверка поведения программы по частям.
- Статический анализ кода — проверка кода на стандартные ошибки «по недосмотру».
- Высокая культура программирования, в частности, паттерны проектирования, соглашения об именовании и прозрачное поведение отдельных блоков кода — чтобы объявить себе и другим, каким образом должна вести себя та или иная функция.
- Широкое использование проверенных внешних библиотек.

Безопасность программного кода и отладка

В программном коде может быть так называемое недокументированное поведение — серьёзные ошибки, которые не проявляются при нормальном ходе выполнения программы, однако весьма опасны для безопасности всей системы в случае целенаправленной атаки. Чаще всего это результат ошибок программиста. Наиболее известные примеры — это SQL-инъекция и переполнение буфера. В данном случае задача отладки это:

- Выявление недокументированного поведения системы
- Устранение небезопасного кода

Выделяют такие методы:

- статический анализ кода. На этой фазе программа сканер ищет последовательности в исходном тексте, соответствующие небезопасным вызовам функций и т. д. Фактически

идет сканирование исходного текста программы на основе специальной базы правил, которая содержит описание небезопасных образцов кода.
- фаззинг. Это процесс подачи на вход программы случайных или некорректных данных и анализ реакции программы.
- Reverse engineering (Обратная инженерия). Этот случай возникает, когда независимые исследователи ищут уязвимости и недокументированные возможности программы.

См. также

- Отладчик
- Отладчик ядра
- Стек вызовов
- Утечка памяти
- Точка останова
- Тестирование программного обеспечения
- Программирование
- Отладочные символы

Примечания

Литература

- Стив Магьюир, «Создание надёжного кода» (Steve Maguire. *Writing Solid Code*. Microsoft Press, 1993)
- Стив Мак-Коннел, «Совершенный код» (Steve McConnel. *Code Complete*. Microsoft Press, 1993)

Ссылки

- AMD64 Instruction-Level Debugging With dbx (http://www.oracle.com/technetwork/server-storage/solarisstudio/documentation/amd64-dbx-364568.html) (англ.)
- AMD64 Instruction-Level Debugging with Sun Studio dbx (http://www.oracle.com/technetwork/server-storage/solaris10/x64-dbx-140481.html) (англ.)
