Лекции 6. Дестабилизирующие факторы и методы обеспечения надежности программных средств (продолжение)

1. Методы обеспечения надежности ПС (продолжение)

Исправление ошибок

Следующий шаг — методы исправления ошибок; после того как ошибка обнаружена, либо она сама, либо ее последствия должны быть исправлены программным обеспечением. Исправление ошибок самой системой — плодотворный метод проектирования надежных систем аппаратного обеспечения. Некоторые устройства способны обнаружить неисправные компоненты и перейти к использованию идентичных запасных. Аналогичные методы неприменимы к программному обеспечению вследствие глубоких внутренних различий между сбоями аппаратуры и ошибками в программах. Если некоторый программный модуль содержит ошибку, идентичные «запасные» модули также будут содержать ту же ошибку.

Другой подход к исправлению связан с попытками восстано-вить разрушения, вызванные ошибками, например искажения записей в базе данных или управляющих таблицах системы. Польза от методов борьбы с искажениями ограничена, поскольку предполагается, что разработчик заранее предугадает несколько возможных типов искажений и предусмотрит программно реализуемые функции для их устранения. Это похоже на парадокс, поскольку, если знать заранее, какие ошибки возникнут, можно было бы принять дополнительные меры по их предупреждению. Если методы ликвидации последствий сбоев не могут быть обобщены для работы со многими типами искажений, лучше всего направлять силы и средства на предупреждение ошибок. Вместо того, чтобы, разрабатывая операционную систему, оснащать ее средствами обнаружения и восстановления цепочки искаженных таблиц или управляющих блоков, следовало бы лучше спроектировать систему так, чтобы только один модуль имел доступ к этой цепочке, а затем настойчиво пытаться убедиться в правильности этого модуля.

Обеспечение устойчивости к ошибкам

Методы этой группы ставят своей целью обеспечить функционирование программной системы при наличии в ней ошибок. Они разбиваются на три подгруппы: динамическая избыточность, методы отступления и методы изоляции ошибок.

1. Истоки концепции **динамической избыточности** лежат в проектировании аппаратного обеспечения. Один из подходов к динамической избыточности — метод голосования. Данные обрабатываются независимо несколькими идентичными устройствами, и результаты сравниваются. Если большинство устройств выработало одинаковый результат, этот результат и считается правильным. И опять, вследствие особой природы ошибок в программном обеспечении ошибка, имеющаяся в копии программного модуля, будет также присутствовать во всех других его копиях, поэтому идея голосования здесь, видимо, неприемлема. Предлагаемый иногда подход к решению этой проблемы состоит в том, чтобы иметь несколько неидентичных копий модуля. Это значит, что все копии выполняют одну и ту же функцию, но либо реализуют различные алгоритмы, либо созданы разными разработчиками. Этот подход бесперспективен по следующим причинам. Часто трудно получить существенно разные версии модуля, выполняющие одинаковые функции. Кроме того, возникает необходимость в дополнительном программном обеспечении для организации выполнения этих версий параллельно или последовательно и сравнения результатов. Это дополнительное программное обеспечение повышает уровень сложности системы, что, конечно, противоречит основной идее предупреждения ошибок — стремиться в первую очередь минимизировать сложность.

Второй подход к динамической избыточности — выполнять эти запасные копии только тогда, когда результаты, полученные с помощью основной копии, признаны неправильными. Если это происходит, система автоматически вызывает запасную копию. Если и ее результаты неправильны, вызывается другая запасная копия и т. д.

2. Вторая подгруппа методов обеспечения устойчивости к ошибкам называется **методами отступления или сокращенного обслуживания.**

Эти методы приемлемы обычно лишь тогда, когда для системы программного обеспечения существенно важно корректно закончить работу. Например, если ошибка оказывается в системе, управляющей технологическими процессами, и в результате эта система выходит из строя, то может быть загружен и выполнен особый фрагмент программы, призванный подстраховать систему и обеспечить безаварийное завершение всех управляемых системой процессов. Аналогичные средства часто необходимы в операционных системах. Если операционная система обнаруживает, что вот-вот выйдет из строя, она может загрузить аварийный фрагмент, ответственный за оповещение

пользователей у терминалов о предстоящем сбое и за сохранение всех критических для системы данных.

3. Последняя подгруппа — **методы изоляции ошибок**. Основная их идея — не дать последствиям ошибки выйти за пределы как можно меньшей части системы программного обеспечения, так чтобы, если ошибка возникнет, то не вся система оказалась неработоспособной; отключаются лишь отдельные функции в системе либо некоторые ее пользователи. Например, во многих операционных системах изолируются ошибки отдельных пользователей, так что сбой влияет лишь на некоторое подмножество пользователей, а система в целом продолжает функционировать. В телефонных переключательных системах для восстановления после ошибки, чтобы не рисковать выходом из строя всей системы, просто разрывают телефонную связь. Другие методы изоляции ошибок связаны с защитой каждой из программ в системе от ошибок других программ. Ошибка в прикладной программе, выполняемой под управлением операционной системы, должна оказывать влияние только на эту программу. Она не должна сказываться на операционной системе или других программах, функционирующих в этой системе.

В большой вычислительной системе изоляция программ является ключевым фактором, гарантирующим, что отказы в программе одного пользователя не приведут к отказам в программах других пользователей или к полному выводу системы из строя. Основные правила изоляции ошибок перечислены ниже. Хотя в формулировке многих из них употребляются слова «операционная система», они применимы к любой программе (будь то операционная система, монитор телеобработки или подсистема управления файлами), которая занята обслуживанием других программ.

1. Прикладная программа не должна иметь возможности непосредственно ссылаться на другую прикладную программу или данные в другой программе и изменять их.

2. Прикладная программа не должна иметь возможности непосредственно ссылаться на программы или данные операционной системы и изменять их. Связь между двумя программами (или программой и операционной системой) может быть разрешена только при условии использования четко определенных сопряжений и только в случае, когда обе программы дают согласие на эту связь.

3. Прикладные программы и их данные должны быть защищены от операционной системы до такой степени, чтобы ошибки в операционной системе не могли привести к случайному изменению прикладных программ или их данных.

4. Операционная система должна защищать все прикладные программы и данные от случайного их изменения операторами системы или обслуживающим персоналом.

5. Прикладные программы не должны иметь возможности ни остановить систему, ни вынудить ее изменить другую прикладную программу или ее данные.

6. Когда прикладная программа обращается к операционной системе, должна проверяться допустимость всех параметров. Прикладная программа не должна иметь возможности изменить эти параметры между моментами проверки и реального их использования операционной системой.

7. Никакие системные данные, непосредственно доступные прикладным программам, не должны влиять на функционирование операционной системы. Ошибка в прикладной программе, вследствие которой содержимое этой памяти может быть случайно изменено, приводит в конце концов к сбою системы.

8. Прикладные программы не должны иметь возможности в обход операционной системы прямо использовать управляемые ею аппаратные ресурсы. Прикладные программы не должны прямо вызывать компоненты операционной системы, предназначенные для использования только ее подсистемами.

9. Компоненты операционной системы должны быть изолированы друг от друга так, чтобы ошибка в одной из них не привела к изменению других компонентов или их данных.

10. Если операционная система обнаруживает ошибку в себе самой, она должна попытаться ограничить влияние этой ошибки одной прикладной программой и в крайнем случае прекратить выполнение только этой программы.

11. Операционная система должна давать прикладным программам возможность по требованию исправлять обнаруженные в них ошибки, а не безоговорочно прекращать их выполнение.

Реализация многих из этих принципов влияет на архитектуру лежащего в основе системы аппаратного обеспечения.

2. Обработка сбоев аппаратуры

Улучшая общую надежность системы, следует заботиться не только об ошибках в программном обеспечении (хотя надежность программного обеспечения требует наибольшего внимания). Другая сторона, о которой необходимо подумать, — это ошибки во входных данных системы (ошибки пользователя).

Наконец, еще один интересующий нас класс ошибок — сбои аппаратуры. Во многих случаях они обрабатываются самой аппаратурой за счет использования кодов, исправляющих ошибки, исправления последствий сбоев (например, переключением на запасные компоненты) и средств, обеспечивающих устойчивость к ошибкам (например, голосование). Некоторые сбои, однако, нельзя обработать только аппаратными средствами, они требуют помощи со стороны программного обеспечения. Ниже приводится список возможностей, которые часто бывают необходимы в программных системах для борьбы со сбоями аппаратуры.

1. ***Повторное выполнение операций.*** Многие сбои аппаратуры не постоянны (например, скачки напряжения, шум в телекоммуникационных линиях, колебания при механическом движении).

Всегда имеет смысл попытаться выполнить операцию, искаженную сбоем (например, команду машины или операцию ввода-вывода), несколько раз, прежде чем принимать другие меры.

2. ***Восстановление памяти.*** Если обнаруженный случайный сбой аппаратуры вызывает искажение области основной памяти и эта область содержит статические данные (например, команды объектной программы), то последствия сбоя можно ликвидировать, повторно загрузив эту область памяти.

3. ***Динамическое изменение конфигурации.*** Если аппаратная подсистема, такая, как процессор, канал ввода-вывода, блок основной памяти или устройство ввода-вывода, выходит из строя, работоспособность системы можно сохранить, динамически исключая неисправное устройство из набора ресурсов системы.

4. ***Восстановление файлов.*** Системы управления базами данных обычно обеспечивают избыточность данных, сохраняя копию текущего состояния базы данных на выделенных устройствах ввода-вывода, регистрируя все изменения базы данных или периодически автономно копируя всю базу данных. Поэтому программы восстановления могут воссоздать базу данных в случае катас-трофического сбоя ввода-вывода.

5. ***Контрольная точка/рестарт.*** Контрольная точка — это периодически обновляемая копия состояния прикладной программы или всей системы. Если происходит отказ аппаратуры, такой, как ошибка ввода-вывода, сбой памяти или питания, программа может быть запущена повторно с последней контрольной точки.

6. ***Предупреждение отказов питания***. Некоторые вычислительные системы, особенно те, в которых используется энергозависимая память, предусматривают прерывание, предупреждающее программу о предстоящем отказе питания. Это дает возможность организовать контрольную точку или перенести жизненно важные данные во вторичную память.

7. ***Регистрация ошибок.*** Все сбои аппаратуры, с которыми уда-лось справиться, должны регистрироваться во внешнем файле, чтобы обслуживающий персонал мог получать сведения о посте-пенном износе устройств.

Из рассмотренных выше трех подгрупп методов обеспечения устойчивости к ошибкам только третья, изоляция ошибок, применима для большинства систем программного обеспечения.

Важное обстоятельство, касающееся всех четырех подходов, состоит в том, что обнаружение, исправление ошибок и устойчивость к ошибкам в некотором отношении противоположны методам предупреждения ошибок. В частности, обнаружение, исправление и устойчивость требуют дополнительных функций от самого программного обеспечения. Тем самым не только увеличивается сложность готовой системы, но и появляется возможность внести новые ошибки при реализации этих функций. Как правило, все рассматриваемые методы предупреждения и многие методы обнаружения ошибок применимы к любому программному проекту. Методы исправления ошибок и обеспечения устойчивости применяются не очень широко. Это, однако, зависит от области приложения. Если рассматривается, скажем, система реального времени, то ясно, что она должна сохранить работоспособность и при наличии ошибок, а тогда могут оказаться желательными и методы исправления и обеспечения устойчивости. К системам

такого типа относятся телефонные переключательные системы, системы управления технологическими процессами, аэрокосмические и авиационные диспетчерские системы и операционные системы широкого назначения.