Надежность программных продуктов

Средства и методы повышения надежности

Принципы и методы обеспечения надежности программ, в соответствии с их целью, можно разделить на четыре группы:

- Предупреждение ошибок,
- Обнаружение ошибок
- Исправление ошибок
- Обеспечение устойчивости к ошибкам

К первой группе относятся принципы и методы, позволяющие минимизировать или вообще исключить ошибки.

Методы второй группы сосредоточивают внимание на функциях самого программного обеспечения, помогающих выявлять ошибки.

К третьей группе относятся функции программного обеспечения, предназначенные для исправления ошибок или их последствий.

Устойчивость к ошибкам (четвертая группа) – это мера способности системы программного обеспеченияпродолжать функционирование при наличии ошибок.

Схема обеспечения надежного ПС

Принципы и методы предупреждения ошибок основной целью имеют не допустить появления ошибок в готовой программе.

С этой целью широко применяют способы так называемого защитного (безопасного) программирования, направленного на уменьшение вероятности ошибок в программах, а также многоверсионное программирование.

Под ***защитным программированием*** понимают ограничение неправильного использования программных объектов. Другими словами, выдвигается требование проектировать и программировать таким образом, чтобы не только гарантировать ожидаемое использование программы в строгом соответствии со спецификациями, но и сделать невозможным ее неправильное использование.

Например, при проектировании системы, в которой взаимодействуют много модулей, можно описать, чтобы какие-то взаимодействия между ними разрешались лишь в определенных ситуациях.

Защитное программирование – технология предупреждения потенциальных ошибок путем проверки в каждом модуле множества допустимых условий.

Во время программирования может быть использовано много технических приемов, чтобы проверить аномалии в управлении или в данных. Вот примеры некоторых из них:

- анализируются граничные значения переменных;
- проверяются размеры, тип и диапазон параметров процедуры ввода данных;
- - параметры “только для доступ к ним чтения” и “считывание-запись” должны быть разделены, и должен быть проверен доступ к ним;
- символьные константы не должны быть доступны для записи;
- -входные переменные и промежуточные (вспомогательные) переменные с физическим значением должны быть проверены на предмет достоверности;

- воздействие выходных переменных должно быть проверено, предпочтительно непосредственным наблюдением связанных с ними изменений состояния системы и т.д.

Простейший метод защитного программирования заключается в использовании специальных ловушек ошибок, рассчитанных на ошибки типа неправильного использования модулей. Разработчика программы не интересует, что будет делать пользователь после того, как получит сообщение о неправильном использовании модуля, но при этом он обязан спроектировать модуль так, чтобы ошибки пользователя не вызывали необратимых изменений в модуле. Таким образом, пользователь, пойманный на неправильном употреблении модуля, принимает корректирующие действия и снова вызывает модуль, не оставляя никаких следов ошибочных вызовов. Иначе говоря, речь идет о таком программировании, когда программный продукт очень трудно или невозможно использовать за пределами области действия его спецификации.

Многоверсионное программирование (N-версионное программирование).

Основная цель : обнаружить и замаскировать остаточные ошибки проекта программного обеспечения в течение выполнения программ, чтобы предотвратить критически опасные отказы системы, и продолжать работу с высокой надежностью.

В многоверсионном программировании спецификация программы реализуется по-разному N раз. Те же самые значения входных данных задаются N версиям, и результаты, произведенные N версиями, сравниваются. Если результат считается достоверным, то он передается компьютеру в качестве выходных данных. N версии могут выполняться параллельно на отдельных компьютерах, альтернативно все версии могут выполняться на одном компьютере.

Выходные результаты подвергаются внутреннему голосованию. Для случаев, где не имеется коллективного согласия, могут быть использованы вероятностные подходы, чтобы максимизировать шанс выбора правильного значения, например, взяв среднее значение, временно заморозив выходные данные до возвращения согласия, и т.д.

При ***дуальном*** программировании (если разрабатываются две версии программы) в случае обнаружения расхождения в результатах, необходимо определить по дополнительным критериям, какой результат правильный и отбросить другой результат.

При N-версионном программировании правильный результат определяется по мажоритарному признаку, т.е. выбирается тот результат, который наблюдается в большинстве вариантов программы. Рассмотренные способы резервирования требуют в 2 или N раз больше времени для вычислений и увеличение объема труда программистов во столько же раз. В связи с этим представляет интерес модифицированное дуальное программирование, при котором наряду с достаточно точной, но сложной основной программой используется менее точная, но простая резервная программа. Если при одинаковых исходных данных результаты работы программ отличаются на величину большую, чем допустимая погрешность, делается предположение, что отказала основная программа, как менее надежная, и в качестве правильного результата принимается результат работы резервной программы. Гарантировать отсутствие ошибок, однако, невозможно никогда.