ИНЖЕНЕРНОЕ ПРОГРАММИРОВАНИЕ ЛЕКЦИЯ 4

АРХИТЕКТУРА ПРОГРАММНОГО СРЕДСТВА

ПЛАН:

1. Понятие архитектуры и задачи ее описания
2. Основные классы архитектур программных средств
3. Взаимодействие между подсистемами и архитектурные функции
4. Контроль архитектуры программных средств

Понятие архитектуры программного средства.

- *Архитектура* ПС – это представление ПС как системы, состоящей из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают обычно отдельные программы. Разработка архитектуры является первым этапом борьбы со сложностью ПС, на котором реализуется принцип выделения относительно независимых компонент.
- Основные задачи разработки архитектуры ПС: Выделение программных подсистем и отображение на них внешних функций (заданных во внешнем описании) ПС;
- определение способов взаимодействия между выделенными программными подсистемами.
- С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация и функциональных спецификаций.

Основные классы архитектур программных средств.

Различают следующие основные классы архитектур программных средств:

- цельная программа;
- комплекс автономно выполняемых программ;
- слоистая программная система;
- коллектив параллельно выполняемых программ.

- *Цельная программа* представляет вырожденный случай архитектуры ПС: в состав ПС входит только одна программа. Такую архитектуру выбирают обычно в том случае, когда ПС должно выполнять одну какую-либо ярко выраженную функцию и ее реализация не представляется слишком сложной. Естественно, что такая архитектура не требует какого-либо описания (кроме фиксации класса архитектуры).

Комплекс автономно выполняемых программ состоит из набора программ, такого, что:

- любая из этих программ может быть активизирована (запущена) пользователем;
- при выполнении активизированной программы другие программы этого набора не могут быть активизированы до тех пор, пока не закончит выполнение активизированная программа;
- все программы этого набора применятся к одной и той же информационной среде.

Слоистая программная система состоит из некоторой упорядоченной совокупности программных подсистем, называемых *слоями*, такой, что:

- на каждом слое ничего не известно о свойствах (и даже существовании) последующих (более высоких) слоев;
- каждый слой может взаимодействовать по управлению (обращаться к компонентам) с непосредственно предшествующим (более низким) слоем через заранее определенный интерфейс, ничего не зная о внутреннем строении всех предшествующих слоев;

- каждый слой располагает определенными ресурсами, которые он либо скрывает от других слоев, либо предоставляет непосредственно последующему слою (через указанный интерфейс) некоторые их абстракции.

Таким образом, в слоистой программной системе каждый слой может реализовать некоторую абстракцию данных. Связи между слоями ограничены передачей значений параметров обращения каждого слоя к смежному снизу слою и выдачей результатов этого обращения от нижнего слоя верхнему. Недопустимо использование глобальных данных несколькими слоями.

- **В качестве примера рассмотрим использование такой архитектуры для построения операционной системы. Такую архитектуру применил Дейкстра при построении операционной системы THE [6.2]. Эта операционная система состоит из четырех слоев (см. рис. 6.1). На нулевом слое производится обработка всех прерываний и выделение центрального процессора программам (процессам) в пакетном режиме.**

- **Только этот уровень осведомлен о мультипрограммных аспектах системы. На первом слое осуществляется управление страничной организацией памяти. Всем вышестоящим слоям предоставляется виртуальная непрерывная (не страничная) память. На втором слое осуществляется связь с консолью (пультом управления) оператора. Только этот слой знает технические характеристики консоли.**

На третьем слое осуществляется буферизация входных и выходных потоков данных и реализуются так называемые абстрактные каналы ввода и вывода, так что прикладные программы не знают технических характеристик устройств ввода и вывода.

Рис. 6.1. Архитектура операционной системы THE.

Коллектив параллельно действующих программ представляет собой набор программ, способных взаимодействовать между собой, находясь одновременно в стадии выполнения. Это означает, что такие программы, во-первых, вызваны в оперативную память, активизированы и могут попеременно разделять по времени один или несколько центральных процессоров. Обычно взаимодействие между такими процессами производится путем передачи друг другу некоторых сообщений.

Простейшей разновидностью такой архитектуры является конвейер. Возможности для организации конвейера имеются, например, в операционной системе UNIX [6.3]. *Конвейер* представляет собой последовательность программ, в которой стандартный вывод каждой программы, кроме самой последней, связан со стандартным вводом следующей программы этой последовательности (см. рис. 6.2).

Конвейер обрабатывает некоторый поток сообщений. Каждое сообщение этого потока поступает на ввод первой программе, которая переработанное сообщение передает следующей программе, а сама начинает обработку очередного сообщения потока. Таким же образом действует каждая программа конвейера.

Таким же образом действует каждая программа конвейера: получив сообщение от предшествующей программы и, обработав его, она передает переработанное сообщение следующей программе и приступает к обработке следующего сообщения. Последняя программа конвейера выводит результат работы всего конвейера (результирующее сообщение). Таким образом, в конвейере, состоящим из n программ, может одновременно находиться в обработке до n сообщений

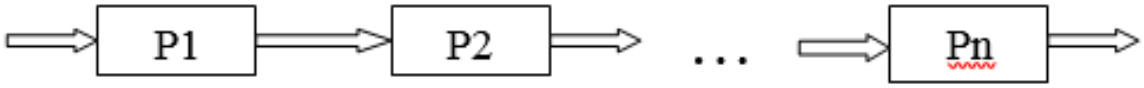

Рис. 6.2. Конвейер параллельно действующих программ.

В общем случае коллектив параллельно действующих программ может быть организован в систему с портами сообщений. *Порт сообщений* представляет собой программную подсистему, обслуживающую некоторую очередь сообщений: она может принимать на хранение от программы какое-либо сообщение, ставя его в очередь, и может выдавать очередное сообщение другой программе по ее требованию.

Сообщение, переданное какой-либо программой некоторому порту, уже не будет принадлежать этой программе (и использовать ее ресурсы), но оно не будет принадлежать и никакой другой программе, пока в порядке очереди не будет передано какой-либо программе по ее запросу. Таким образом, программа, передающая сообщение не будет находиться в стадии ожидания пока программа, принимающая это сообщение, не будет готова его обрабатывать.

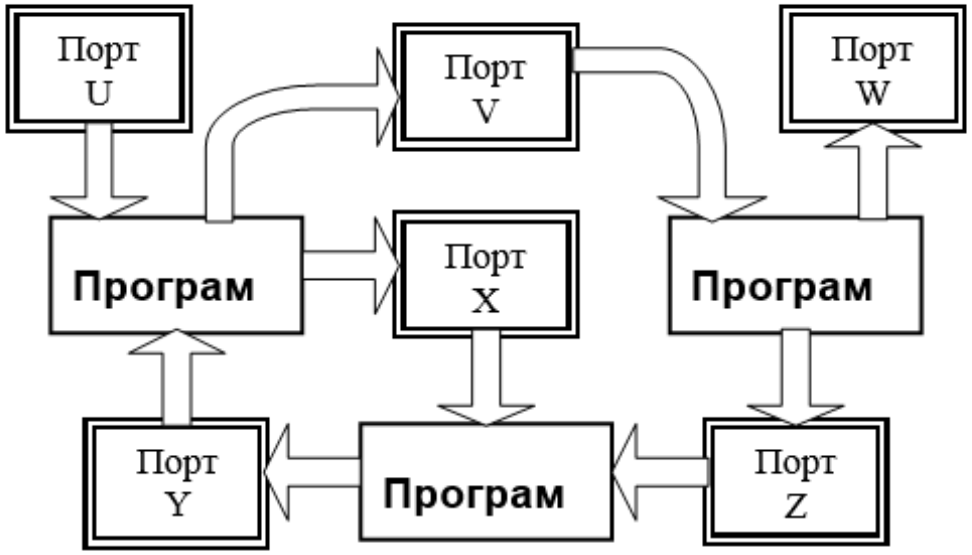

Рис. 6.3. Пример программной системы с портами сообщений.

СПАСИБО ЗА ВНИМАНИЕ