– обновления базы знаний, предоставляемой школами;

– повышения уровня преподавания, так как большинство учителей информатики преподают предмет на уровне пользователей. Они могут объяснить, что нужно делать, чтобы программа выдавала нужный результат, но какие процессы происходят в самой машине в момент запуска или компиляции программы, не знают.

– регулирования очерёдности предоставляемой информации, то есть прежде, чем рассказывать детям, как работать в паскале, нужно объяснить, что каждая программа ориентирована на работу на определённых аппаратных комплектующих, находящихся внутри системного блока компьютера и т.д.

УДК 004.057.4

Санцевич С. Н.

СОКЕТЫ В СТЕКЕ ПРОТОКОЛОВ TCP/IP

БНТУ, г. Минск, Республика Беларусь
Научный руководитель: преподаватель С. Г. Липень

Как известно, стек протоколов TCP/IP состоит из 4 уровней:

– прикладной уровень (application layer);

– транспортный уровень (transport layer);

– сетевой уровень (internet layer);

– канальный уровень (link layer).

При передаче данных каждый из этих уровней выполняет свою работу, а её результат отправляет на следующий уровень.

На прикладном уровне работает большинство сетевых приложений. Эти программы имеют свои собственные протоколы обмена информацией.

Однако, приложения могут находиться в разных операционных системах (ОС). Каждая из их имеет свои особенности и, соответственно, с сетью они работают по-разному. Поэто-

му, им нужна некоторая абстракция, которая будет находиться между прикладным уровнем и всей остальной сетью. Абстракция, которой не важен пользовательский протокол.

Эта абстракция и есть *сокет*. Именно сокет выполняет работу одинаково вне зависимости от ОС. Формально, сокет – это программный интерфейс для обеспечения обмена данными между процессами.

Когда в операционную систему UNIX были добавлены средства межпроцессного взаимодействия (Inter-Process Communication, IPC) и сетевого обмена, был заимствован привычный шаблон ввода-вывода. Все ресурсы, открытые для связи, в UNIX и Windows идентифицируются дескрипторами. Эти дескрипторы, или описатели (handles), могут указывать на файл, память или какой-либо другой канал связи, а фактически указывают на внутреннюю структуру данных, используемую операционной системой. Сокет, будучи таким же ресурсом, тоже представляется дескриптором. Следовательно, для сокетов жизнь дескриптора можно разделить на три фазы: открыть (создать) сокет, получить из сокета или отправить сокету и в конце концов закрыть сокет.

Интерфейс IPC для взаимодействия между разными процессами построен поверх методов ввода-вывода. Они облегчают для сокетов отправку и получение данных. Целевой объект задается адресом сокета, следовательно, этот адрес можно указать в клиенте, чтобы установить соединение с целью.

Существуют два основных *типа сокетов* – потоковые сокеты и дейтаграммные:

– потоковые сокеты (stream socket);

– дейтаграммные сокеты (datagram socket).

Потоковый сокет – это сокет с установленным соединением, состоящий из потока байтов, который может быть двунаправленным, т, е. через эту конечную точку приложение может и передавать, и получать данные. Потоковый сокет гаран-

тирует исправление ошибок, обрабатывает доставку и сохраняет последовательность данных. На него можно положиться в доставке упорядоченных, сдублированных данных.

Дейтаграммные сокеты иногда называют сокетами без организации соединений, т. е. никакого явного соединения между ними не устанавливается – сообщение отправляется указанному сокету и, соответственно, может получаться от указанного сокета. Потоковые сокеты по сравнению с дейтаграммными действительно дают более надежный метод, но для некоторых приложений накладные расходы, связанные с установкой явного соединения, неприемлемы.

Трудно переоценить роль сокетов, так как в цепочке шагов по разработке любого приложения, ориентированного на обмен данными по сети, они являются одним из всех ключевых звеньев.

УДК 714

Санцевич С. Н., Бунькевич Д. А.

СПРАЙТЫ В CSS

БНТУ, г. Минск, Республика Беларусь
Научный руководитель: ст. преподаватель Астапчик Н. И.

До того, как в CSS появился псевдокласс: hover, создание ролловера – элемента, который меняет свой вид при наведении курсора – реализовывалось через язык JavaScript. Сейчас это делается намного проще, но есть один недостаток: если в состоянии: hover (то есть при наведении курсора на элемент) должно появиться какое-то фоновое изображение, то оно начинает загружаться в момент наведения курсора, а не при общей загрузке страницы.

Из-за этого может возникать небольшая задержка при появлении картинки в первый раз. И хотя при всех последующих наведениях курсора этой задержки уже нет, многие разработчики задумались над тем, как устранить эту проблему.