Федеральное агентство по образованию Российской Федерации (РФ)

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра Электронных приборов (ЭП)

УТВЕРЖДАЮ
Заведующий кафедрой ЭП
_____ С.М. Шандаров

# Информационные системы

Учебно-методическое пособие
Методические указания к лабораторным работам

Разработчик:
ст. преподаватель каф. ЭП
_____ Е.С. Шандаров

2007

# Содержание

# Введение

Данный курс лабораторных работ посвящен теме «Информационные системы» и включает в себя описание 5 лабораторных работ.

Лабораторные работы по курсу проводятся с использованием программного обеспечения с открытыми кодами, в том числе: сервер Apache, броузер Firefox, язык PHP.

В рамках данного курса студенты осваивают технологии Интернет, создают свой собственный мини-сайт, пишут серверные скрипты.

# Понятие информационных систем. Базовые принципы построения информационных систем на основе технологий Интернет

Современные информационные системы - это комплексные решения , интегрирующие телекоммуникационные и информационные технологии, различные среды передачи информации и ее носители. Информационные системы окружают нас повсюду, они помогают бухгалтерам в начислении зарплаты и расчете налогов, продают авиабилеты и позволяют вносить платежи за мобильную связь, обеспечивают бесперебойную и безаварийную работу технологических установок и даже пытаются управлять целыми производственными комплексами.

При проектировании и создании современных информационных систем, в дальнейшем ИС, могут быть использованы совершенно различные подходы. В данном пособии мы постараемся рассмотреть круг вопросов, связанных с проектированием и созданием информационных систем на базе технологий Интернет. Как нам представляется, именно этот подход на сегодняшний день является наиболее прогрессивным и динамично развивающимся.

Попробуем определить несколько базовых принципов построения информационных систем на основе технологий Интернет.

В первую очередь отметим, что любая ИС содержит в себе следующий набор обязательных компонентов:

- интерфейс пользователя
- подсистема обработки пользовательской информации
- подсистема хранения данных
- канал передачи данных между центром процессинга и приложением пользователя

Технологии Интернет предлагают нам обширный инструментарий для реализации любого из вышеуказанных компонентов. Для обслуживания канала передачи данных воспользуемся семейством протоколов TCP/IP, в качестве

подсистемы обработки пользовательской информации возьмем HTTP-сервер Apache с подключенным языком написания скриптов PHP. С задачей хранения структурированных данных прекрасно справится один из многочисленных SQL-серверов. В качестве пользовательского интерфейса возможно использование как специально написанного программного обеспечения на языке высокого уровня, так и набор стандартных компонентов, предлагаемый нам любым Интернет-броузером и языком гипертекстовой разметки документов HTML.

При этом наша информационная система будет доступна пользователю из любой точки земного шара (во всяком случае если там есть Интернет или работает спутниковая связь), обеспечит высокую скорость обработки информации (благодаря технологии распределенных систем) и сможет защитить наши данные от постороннего вмешательства.

В чем на наш взгляд состоят основные преимущества использования технологий Интернет для построения современных информационных систем?

Выделим несколько ключевых причин

- открытые стандарты
- поддержка любой вычислительной платформы
- возможность использования программного обеспечения с открытым кодом
- относительно простая и эффективная организация многопользовательской работы
- относительно простой способ решения задачи актуализации программного обеспечения
- доступность информационной системы из любой точки мира

## Инструменты необходимые для выполнения лабораторных работ

Как уже отмечалось выше, для выполнения лабораторных работ нам потребуется следующий набор программного обеспечения:

HTTP-сервер Apache

броузер Mozilla Firefox

интерпретатор языка скриптов PHP

любой текстовый редактор (в системе Windows это может быть Notepad, в Linux например kwrite)

Ниже дадим краткое описание данных продуктов

### *HTTP-сервер Apache*

Самый распространенный Web-сервер в мире - это Apache. По данным компании Netcraft (http://www.netcraft.com/Survey/) общее число Web-узлов, работающих под его управлением, к концу 1998 г. достигло 2 млн. (55% общего числа узлов) и постоянно растет. Для сравнения: на долю серверов Microsoft приходится 25%, Netscape -7%. Будучи бесплатной открытой программой, предназначенной для бесплатных же Unix-систем (FreeBSD, Linux и др.), Apache по функциональным возможностям и надежности не уступает коммерческим серверам, а широкие возможности конфигурирования позволяют настроить его для работы практически с любой конкретной системой. Существуют локализации сервера для различных языков, в том числе и для русского.

Для тонкой настройки сервера Apache используется набор конфигурационных файлов. Рассмотрим здесь одну из наиболее важных процедур конфигурирования сервера Apache.

#### Файл httpd.conf

Конфигурационный файл httpd.conf является основным и содержит настройки, связанные с работой Web-сервера, виртуальных серверов, а также

всех его программных модулей. Кроме того, именно в нем настраивается перекодирование русских букв при передаче от сервера к клиенту и обратно.

Директива Port, помещенная в самом начале файла, определяет номер порта для http-сервера; по умолчанию это 80. При необходимости можно приписать серверу другой порт или несколько портов, для чего служит директива Listen.

Директива HostnameLookups с параметром on или off включает или, соответственно, отключает преобразование численных IP-адресов клиентов, получивших документы с сервера, в доменные имена. Такое преобразование несколько замедляет работу сервера, но при числе посещений менее 10 000 в сутки это, как правило, практически не заметно.

Директивы User и Group задают пользователя, который будет администрировать сервер. С точки зрения безопасности нежелательно указывать здесь существующего пользователя, имеющего доступ к каким-либо другим ресурсам или файлам. Лучше создать отдельного пользователя и группу специально для http-сервера, например:

```
User www
Group www
```

Директивы ServerRoot, ErrorLog, CustomLog определяют соответственно корневой каталог http-сервера, путь к журналу регистрации ошибок (error_log) и путь к общему журналу обращений к серверу (access_log).

Директива CacheNegotiatedDocs разрешает кэширование документов, полученных с сервера. По умолчанию этот режим отключен, но, поскольку пропускная способность отечественных Internet-каналов еще долго будет оставлять желать лучшего, хорошо бы его включить: тогда пользователю не придется ждать загрузки картинок при каждом обращении к вашей странице.

**Настройка виртуальных серверов в файле httpd.conf**

В большинстве случаев один http-сервер способен обрабатывать запросы, поступающие на различные, так называемые виртуальные, Web-серверы. Виртуальные серверы могут иметь как один и тот же IP-адрес, но разные доменные имена, так и разные IP-адреса. С точки зрения пользователя второй вариант чуть более предпочтителен, поскольку запрос к серверу, отличающемуся от основного только доменным именем, должен содержать его имя, а некоторые старые браузеры, не поддерживающие протокол HTTP/1.1 (например, Microsoft Internet Explorer 2.0), не включают в запрос эту информацию. Однако такие браузеры выходят из употребления (сейчас их уже менее 0,5% общего числа); с другой стороны, выделение собственного IP-адреса каждому виртуальному серверу может быть неоправданной растратой адресного пространства компании.

Для описания адресов и доменных имен виртуальных серверов служат директивы ServerName, ServerAlias, NameVirtualHost и VirtualHost. Они необходимы, только если вам нужно установить более одного виртуального сервера.

**Описание виртуальных серверов с различными IP-адресами**

```
ServerName ed.tusur.ru

<VirtualHost 192.168.1.1>
DocumentRoot /www/ed.tusur.ru
ServerName ed.tusur.ru
ErrorLog /var/log/error_log.ed.tusur.ru
CustomLog /var/log/access_log.ed.tusur.ru combined
...
</VirtualHost>
```

```
<VirtualHost 192.168.1.3>
DocumentRoot /www/rzi.tusur.ru
ServerName rzi.tusur.ru
ErrorLog /var/log/error_log.rzi.tusur.ru
CustomLog /var/log/access_log.rzi.tusur.ru combined
...
</VirtualHost>
```

В листинге приведен фрагмент конфигурационного файла для случая виртуальных серверов с различными IP-адресами, в следующем листинге - аналогичный фрагмент для случая, когда серверы различаются только доменным именем.

**Описание виртуальных серверов, различающихся только доменным именем**

```
ServerName ed.tusur.ru
NameVirtualHost 192.168.1.1

<VirtualHost 192.168.1.1>
DocumentRoot /www/ed.tusur.ru
ServerName ed.tusur.ru
ErrorLog /var/log/error_log.ed.tusur.ru
CustomLog /var/log/access_log.ed.tusur.ru combined
...
</VirtualHost>

<VirtualHost 192.168.1.1>
DocumentRoot /www/forum.ed.tusur.ru
ServerName forum.ed.tusur.ru
```

```
ServerAlias *.forum.ed.tusur.ru
ErrorLog /var/log/error_log.forum.ed.tusur.ru
CustomLog /var/log/access_log.forum.ed.tusur.ru combined
...
</VirtualHost>
```

Директива ServerName, находящаяся вне секций VirtualHost, определяет имя основного сервера, т. е. сервера, корневой каталог которого задан директивой DocumentRoot в файле srm.conf. Виртуальные серверы наследуют настройки основного; при необходимости специальной настройки соответствующие директивы помещаются в секции VirtualHost, относящейся к данному серверу. Допустимы любые директивы, которые могут встретиться в файлах httpd.conf и srm.conf, например DocumentRoot, ErrorLog, CustomLog, Location, ServerAdmin.

Из последнего листинга видно, как используется директива ServerAlias, если необходимо создать несколько виртуальных серверов с одинаковым содержанием.

## *Язык разметки гипертекстовых страниц HTML*

HTML (HyperText Markup Language - язык маркировки гипертекстов) - язык разметки гипертекстового документа. Говоря проще, это набор средств для описания визуальных свойств (позиция, размер, цвет и т.д.) различных элементов, в частности текста или графики. Под гипертекстовым документом подразумеваются документы с гипертекстовыми ссылками - указателями на другие гипертекстовые документы.

HTML документ в самом простом случае представляет собой один текстовый файл с разширением .htm или .html - никакой разницы между ними нет, можете использовать то, которое Вам больше нравится. Создавать HTML возможно при помощи обычного текстового редактора. Только не стоит использовать программы, которые добавляют форматирование (например, Microsoft Word). Это такие программы в которых можно, например выделить часть текста жирным фоном или вставить картинку - все это испортит HTML файл, для корректного создания которого необходим чисто текстовый редактор.

Язык разметки гипертекста (HyperText Markup Language - HTML) можно использовать для представления:

- гипертекстовых новостей, почты, сопровождающей информации и сопутствующей гиперсреды,
- меню с опциями
- результатов запросов к базам данных
- простых структурированных документов со встроенной графикой
- гипертекстовых обзоров имеющейся информации

Программа World Wide Web (W3) инициирует каналы передачи связной информации по всему земному шару. Язык HTML предоставляет простой формат для предоставления этой информации. Требуется, чтобы все программы, совместимые с W3, могли поддерживать язык HTML. Программа W3 использует протокол Internet (протокол передачи гипертекста - HTTP),

который позволяет передавать кодированную информацию между клиентом и сервером, при этом результат возвращается через расширенное MIME сообщение. Поэтому язык HTML является лишь одним, но довольно важным, из описаний, используемых в программе W3.

Данные в формате HTML похожи на текстовой файл, за исключением того, что некоторые из символов интерпретируются как разметка. Разметка придает документу некую структуру.

Данные представляют собой иерархию элементов. Каждый элемент имеет имя, атрибуты и несет некую информацию. Большинство элементов представлены в документе в виде начальной метки, указывающей имя и атрибуты. Далее следует собственно содержание элемента. И наконец, заканчивает все это конечная метка. Например,

```
<HTML>
    <TITLE>
        Простой блок данных
     </TITLE>
    <H1>
        Пример структуры
    </H1>
 Обычный параграф
      <P>
      <UL>
         <LI>Первая запись, включающая
            <A NAME="URI">
               текст
            </A>
         <LI>Вторая запись
      </UL>
   </HTML>
```

Некоторые элементы языка (такие как P, LI) являются пустыми. Они не имеют поля данных, и ограничиваются лишь начальной меткой.

В остальных элементах поле данных представляет собой набор символов и вложенных элементов. Заметим, что описание HTML DTD фактически накладывает некие ограничения на количество допустимых вложений - большинство элементов не могут быть вложены в другие элементы. Ни один из элементов не может быть вложенным сам в себя рекурсивным образом. Анкеры и выделенные символы могут быть помещены в другие конструкции.

Каждый элемент начинается с метки, меткой же и заканчивается каждый непустой элемент. Начальные метки выделяются символами < и >, а конечные - символами </ и >.

Имя элемента следует в метке сразу за символом открытия <. Имя начинается с буквы, за которой могут следовать еще 33 буквы, цифры, пробела или дефиса. В именах игнорируется разница между прописными и строчными буквами.

Начальная метка позволяет вставить между именем и символом > пробелы и атрибуты. Атрибут состоит из имени, символа равенства и значения. Слева и справа от символа равенства можно оставлять пробелы.

Значение атрибута указывается в виде строки, заключенной в одинарные или двойные кавычки.

Чтобы определить значение атрибута, осуществляется анализ данной строки в формате RCDATA. Например, такой подход позволяет представлять символы кавычек в значении атрибута как обращения к символам по числовому значению. Длина строки со значением атрибута после такого анализа не должна превышать 1024 символов.

Если мы хотим включить в HTML документ комментарий таким образом, чтобы он игнорировался анализатором, необходимо поставить перед ним и после него ограничители <!-- и --> соответственно. Весь текст, расположенный между начальным ограничителем и символами --, будет игнорироваться. Следовательно, комментарии не могут быть вложенными. В заключительном ограничителе между -- и символом > можно вставлять пробелы (но в начальном ограничителе между <! и -- вставки не допускаются). Например,

```
<HEAD>
<TITLE>HTML Guide: Recommended Usage</TITLE>
<!-- $Id: recommended.html,v 1.3 93/01/06 18:38:11
connolly Exp $ -->
</HEAD>
```

## Элементы языка HTML

Здесь приведен список элементов, используемых в языке HTML. Документы должны (но не обязательно) содержать элемент HEAD, за которым следует элемент BODY.

Документы старого типа могут содержать лишь данные обычных элементов HEAD и BODY, причем в любом порядке. Это осуждается, но тем не менее, анализаторы должны воспринимать такое построение документа. Обратите внимание также на статус элементов.

## Общие свойства документа

Элемент **HEAD** содержит всю информацию о документе в целом. Однако он не содержит какого-либо текста. Последний является лишь частью документа и должен находиться в элементе BODY. В элементе заголовка HEAD можно использовать лишь строго заданный набор элементов.

Нижеприведенные элементы определяют общие свойства документа. Они

должны появляться в элементе HEAD. Порядок элементов значения не имеет.

- TITLE - Название элемента.
- ISINDEX - Элемент, посылаемый серверу вместе с документом, предназначенным для информации к поиску. .
- NEXTID - Параметр, используемый текстовыми редакторами для создания , уникальных идентификаторов. , (Устарел и не рекомендуется использовать.).
- LINK - Элемент, определяющий связь этого документа с другими. В , документе может присутствовать несколько элементов LINK.
- BASE - Запись, сделанная на языке URL при фиксировании данного , документа.

**Форматирование текста**

В элементе BODY документа встречаются элементы из приведенного ниже списка. Они выстроены в том порядке, в каком должны подаваться на устройство вывода.

**Заголовки**

(Headings) Язык поддерживает заголовки разделов различных уровней.

**Анкеры**

(Anchors) Части текста, которые формируют начало и/или конец связей в гипертексте, называются, анкерами и формируются меткой A.

**Метки параграфов**

(Paragraph marks) Элемент P указывает на границу между параграфами.

**Стиль адреса**

(Address style) Этот элемент указывает, в каком стиле предстает перед клиентом элемент ADDRESS.

**Выделенный блок текста.**

(Blockquote style) .

**Списки списки, словари и т.д.**

**Преформатированный текст**

(Preformatted text) Части текста, предварительно отформатированные с

использованием шрифта фиксированной ширины.

**Выделение символов**

(Character highlighting) Элементы форматирования, не вызывающие разбиения на параграфы.

**Графика**

IMG - Метка IMG может использоваться для включения в текст графических изображений.

В противоположность элементу HEAD элемент BODY содержит всю ту информацию, из которой собственно и состоит рассматриваемый документ. Порядок следования элементов здесь именно тот, в каком они предстают перед читателем.

**Анкер (элемент A)**

Анкер - это некий текст, который указывает на начало и/или конец связи в гипертексте. Текст между открывающей и закрывающей метками определяет начало связи или указываемое ею место (или и то, и другое вместе). Метка анкера может иметь следующие атрибуты:

- HREF Необязательный. (Адрес гипертекстовой ссылки) Если атрибут HREF установлен, то анкер является точно выверенный текстом - началом соединения. Если читатель выбрал этот текст, то ему (ей) будет представлен другой элемент, чей сетевой адрес определяется значением HREF атрибута. Формат сетевого адреса определяется в другом месте. Такой подход позволяет с помощью формы HREF="#индикатор" ссылаться на другой анкер в том же самом документе. Если же анкер относится к другому документу, атрибут является относительным именем, именем относительно данного документа (либо он указывает базовый адрес, если таковой имеется).
- NAME Необязательный. Если этот атрибут указан, то он позволяет данному анкеру быть местом в документе, на которое ссылается какой-

либо анкер. Значение атрибута является идентификатором анкера. Идентификатор анкера - это произвольная строка текста, которая тем не менее уникальна в пределах рассматриваемого HTML документа. Другие документы тоже могут создавать ссылки именно на этот анкер, помещая его идентификатор в поле адреса документа после символа #.

- REL Необязательный. Атрибут REL может дать взаимоотношение(ия) в описанной ранее связи гипертекста. Значение атрибута - это список значений для взаимоотношений, написанный через запятую. Значения атрибута и их семантика будут регистрироваться комитетом по языку HTML. Если ничего не указано, то по умолчанию предполагается, что взаимоотношениях не несут каких-либо значений. Атрибут REL нельзя применять, если нет атрибута HREF.
- REV Необязательный. Полностью аналогичен атрибуту REL за исключением того, что тип соединения имеет обратную семантику. Связь из анкера A в анкер B с атрибутом REL="X" полностью аналогична связи из B в A с атрибутом REV="X". Анкер может иметь оба атрибута REL и REV.
- URN Необязательный. Если этот атрибут указан, то это определяет универсальный номер ресурса для данного документа.
- TITLE Необязательный. Данный атрибут является чисто информационным. Если атрибут присутствует в анкере, его значение должно (может прим.ред.) совпадать со значением элемента TITLE в документе, чей адрес указан в атрибуте HREF.
- METHODS Необязательный. Значение этого атрибута - строка. Она должна представлять собой список через запятую методов HTTP, которые программа общего пользования в состоянии поддерживать.

Все приведенные выше атрибуты являются необязательными, хотя для того, чтобы анкер действовал, нужны NAME и HREF. См. также описание LINK.

**Пример использования анкеров**

```
See <A HREF="http://info.cern.ch/">CERN</A>'s information
for more details.

A <A NAME=serious>serious</A> crime is one which is
associated with imprisonment.

The Organization may refuse employment to anyone
convicted of a <a href="#serious">serious</A> crime.
```

**Замечание 1**. Универсальные номера для ресурсов (Universal resource numbers - URN) должны обеспечивать распознавание документа в случае обнаружения его дубликатов. Должно существовать программное обеспечение клиента, осуществляющее отсев копий для уже имеющейся информации.

Формат номеров URN обсуждается различными рабочими группами из инженерного подразделения сети Internet (1993). (На сегодняшний день спецификация URN не определена. Прим.ред.)

**Замечание 2**. Атрибут названия для связей (TITLE).

Соединение может содержать атрибут TITLE. Если этот атрибут имеется, он должен давать название документа, чей адрес определен в атрибуте HREF. Есть по крайней мере две причины для использования такого атрибута.

- Программа просмотра может запросить показ названия документа в качестве предварительного условия для его выборки. Например, в виде метки с записью, или маленького ящика, возникающего, когда мышь попадает на анкер или же при вызове документа.
- Некоторые документы не имеют названия, так что использование атрибута названия для связи является для них единственным способом получить название. В основном это документы, не являющиеся размеченным текстом, графикой, текстом и меню для программы Gopher.

Именно так работает упомянутая программа Gopher. Очевидно, что это приводит к дублированию данных и было бы рискованно безоглядно надеяться на то, что атрибут названия у соединения будет корректным и уникальным для соответствующего документа.

**Замечание. 3** Атрибут метода для связи (METHODS).

Анкеры и связи используют атрибут метода для указания действий, которые клиент может применять к объектам. Эти действия более точно формулируются в HTTP протоколе, если таковой применяется. Однако этот атрибут, как и атрибут TITLE, в силу некоторых причин может использоваться для повышения информативности соединения. Например, программа чтения может вызывать различные способы визуализации информации в зависимости от разрешенного в атрибуте метода (например, клиент, осуществляющий поиск, может пользоваться различными иконами).

**Элемент выделения блока (BLOCKQUOTE)**

Элемент BLOCKQUOTE допускает обработку специальным образом текста, выделенного в каком-либо источнике.

Типовая обработка элемента. Типовая обработка может заключаться в дополнительном смещении текста влево или вправо и/или в использовании наклонного шрифта. Элемент BLOCKQUOTE приводит к разбиению текста на параграфы, а также обычно к появлению пустой строки или пробелов между выделенным блоком и предшествующим/ последующим текстом.

Обработка с единым шрифтом может, к примеру, привести к появлению в начале строки символа ">", что соответствует стилю выделения в системе Internet почты.

Пример:

```
I think it ends
<BLOCKQUOTE>Soft you now, the fair Ophelia, Nymph, in thy
orisons,
be all my sins remembered.
</BLOCKQUOTE>
but I am not sure.
```

**Заголовки (Headings)**

Обрабатывается до шести уровней заголовков (Заметим, что узел в гипертексте, как правило, нуждается в меньшем количестве уровней, чем сочинение, чья структура целиком определяется применением заголовков). Элемент заголовка несет в себе все изменения шрифтов, разбиение на параграфы до и после, пробелы, необходимые, например, для обработки заголовка. Язык HTML не требует применения иных средств для выделения символов или разбивки текста на параграфы.

Заголовок H1 относится к самому верхнему уровню и рекомендуется в качестве начального для узла в гипертексте. Предполагается, что текст первого заголовка будет соответствовать запросам клиента, уже производящего анализ связанной с этим узлом информации. Это отличает заголовок (heading) от названия (title), которое должно характеризовать данный узел в более широком плане.

Элементы заголовка: <H1>, <H2>, <H3>, <H4>, <H5>, <H6>.

Было бы отклонением от правил при переходе от заголовка к заголовку пропускать какой-либо уровень, например, ставить после элемента H1 сразу элемент H3. Хотя такая практика и не запрещена, но нежелательна и может привести к странным результатам при написании других реализаций языка HTML.

Пример

```
<H1>This is a heading</H1>
Here is some text
<H2>Second level heading</H2>
Here is some more text
```

- H1 Толстый, очень крупный шрифт, текст центрирован. Между заголовком и последующим текстом вставляется одна или две пустых строки. При выводе на принтер заголовок печатается на новой странице.
- H2 Толстый крупный шрифт. Без отступа. До и после заголовка помещаются одна или две пустых строки.
- H3 Наклонный большой шрифт. До и после заголовка помещаются одна или две пустые строки. С небольшим отступом.
- H4 Толстый нормальный шрифт. Отступ больше, чем в H3. До и после заголовка помещается пустая строка.
- H5 Наклонный нормальный шрифт. Отступ как у заголовка H4. Пустая строка ставится перед заголовком, но не после.
- H6 Толстый шрифт. Отступ такой же, как у обычного текста и больше, чем у H5. Перед заголовком ставится пустая строка.

**IMG: Встроенные изображения**

Статус: дополнительный

Элемент IMG позволяет вставлять информацию из другого документа. Последний обычно является иконкой, маленькой картинкой и т.д. Элемент IMG не предназначен для вставки дополнительного HTML текста.

Те анализаторы гипертекста, которые не могут показывать встроенные изображения, элементы IMG игнорируют. Авторам документов следует взять на заметку, что некоторые анализаторы могут показывать (или печатать на принтере) связанные с данным документом изображения, но не встроенные. Если изображение имеет большое значение, может оказаться более разумным

создать с ним связь, нежели делать это изображение встроенным в гипертекст. Если же изображение является в значительной степени декоративным, более удобным будет применение элемента IMG.

Элемент IMG является пустым (не имеет заключительной метки) и имеет два атрибута:

- SRC Значением этого атрибута является URL документа, который должен быть вставлен в гипертекст. Синтаксис этого атрибута такой же, как и у атрибута HREF для метки A. Атрибут SRC является обязательным.
- ALIGN Это атрибут, принимая значения TOP, MIDDLE или BOTTOM, определяет, верхняя, средняя или нижняя часть изображения должна быть поставлена вровень с текстом.

В тексте анкеров допускается применение элементов IMG.

Пример

```
Warning: < IMG SRC ="triangle.gif">
Thus must be done by a qualified
technician.
< A HREF="Go"><  IMG SRC ="Button"> Press to start</A>
```

**Списки**

Список - это последовательность параграфов, каждому из которых может предшествовать специальная метка или очередной номер. Синтаксис списка:

```
<UL>
<LI> list element
<LI> another list element ...
</UL>
```

Открывающими метками для списка могут быть UL, OL, MENU или DIR.

Сразу за открывающей меткой должен следовать первый элемент списка.

Список элементов, имеющих типовые алгоритмы обработки:

- UL Список многострочных параграфов, обычно разделенных несколькими пробелами и/или размеченный кружками или крупными черными точками.
- OL Этот элемент похож на элемент UL, за исключением того, что параграфы нумеруются.
- MENU Список параграфов меньшего размера. Обычно на одну запись приходится лишь одна строка, а ее стиль более компактен, чем в случае элемента UL.
- DIR Список элементов, чей размер, как правило, не превышает 20 символов. Элементы могут размещаться в несколько колонок на странице, причем ширина такой колонки обычно 24 символа. Намного лучше, если программа обработки в состоянии оптимизировать ширину колонки в зависимости от ширины составляющих ее элементов.

Пример использования

```
<OL>
<LI> When you get to the station, leave
by the southern exit, on platform one.
<LI>Turn left to face toward the mountain
<LI>Walk for a mile or so until you reach the
"Asquith Arms" then
<LI>Wait and see ...
</OL>

< MENU >
<LI>The oranges should be pressed fresh
<LI>The nuts may come from a packet
<LI>The gin must be good quality
```

```
</MENU>

< DIR >
<LI>A-H<LI>I-M
<LI>M-R<LI>S-Z
</DIR>
```

**P: Метка параграфа**

Пустой элемент P служит разделителем параграфов. Конкретная процедура обработки (отступы, инструкции и т.д.) здесь не оговаривается и может зависеть от наличия иных меток, стилей и т.д.

Метка <P> ставится между двумя частями текста для их разделения.

Нет нужды применять <P> для создания пустого места вокруг заголовка, списка, адреса или выделенных элементов, которые уже сами по себе предполагают наличие разделителей параграфов. Создание пустых мест вокруг перечисленных элементов - обязанность программы обработки. Соседство метки разделителя параграфов и такого элемента, который сам автоматически создает разделители параграфов, может привести к непредсказуемым последствиям. Следует избегать того, чтобы метке разделителя параграфов предшествовал или следовал за ней такой элемент.

Обычно метка <P> создает небольшой вертикальный пропуск между параграфами (одна строка или полстроки). Этого не происходит (как правило) в тексте элементов ADDRESS и (даже) PRE. В некоторых версиях в обычном тексте метка <P> может также создавать небольшой отступ слева в первой строке открываемого ею параграфа.

Примеры использования

```
<h1>What to do</h1>
This is a one paragraph.<p>This is a second.
< P >
```

```
This is a third.
```

**Выделение символов**

Элементы выделения позволяют форматировать отдельные части текста особым образом, производить выделение и т.д. Метки выделения не приводят к разбиению на параграфы и могут применяться к отдельным кускам текста внутри параграфов. Как и все метки, не поддерживаемые имеющимися версиями анализаторов языка HTML, эти метки будут игнорироваться, однако размеченный ими текст будет обрабатываться как и любой другой.

Все метки выделения имеют соответствующие им завершающие метки, как в этом примере

```
This is <EM>emphasized</EM> text.
```

Практическая реализация одних стилей выделения более очевидна. Для других - менее. Логические стили можно применять в любом месте, если, к примеру, нет нужды ссылаться в тексте на процедуру форматирования (например, "обязательно использование наклонного шрифта для отдельных частей текста").

**Замечание**

Анализаторы, не способные изобразить какой-либо стиль выделения символов, могут представить его с некоторой потерей качества изображения с применением альтернативного стиля или стиля по умолчанию. Некоторые версии анализаторов могут игнорировать все метки, так что поставщикам информации желательно не придавать меткам выделения большую смысловую нагрузку.

**Физические стили**

- TT  Шрифт фиксированной ширины.

- B Толстый или еще каким-либо образом выделенный шрифт.
- I Наклонный шрифт (или искаженный каким-либо образом, если просто наклон невозможен).
- U Подчеркивание.

**Логические стили**

- EM Выделение символов (обычно наклон шрифта). (смысловое усиление определенного слова или фразы)
- STRONG Более четкое выделение (обычно применение более жирного шрифта). (выделение, привлечение внимания)
- CODE Пример кода. Обычно фиксированный шрифт (не путать с элементом PRE). (формулы, выражения.)
- SAMP Последовательность символов. (названия команд, примеры)
- KBD Текст, набираемый пользователем. Этот стиль применяется в описаниях.
- VAR Имя переменной. (имена переменных в примерах, формулах)
- DFN Пример определения к какому-либо термину. Обычно жирный наклонный шрифт или просто жирный. (Официально считается расширением в версии HTML 2.0)
- CITE Цитата. Обычно наклонный шрифт. (названия документов, выдержки из документов, цитируемые фразы и т.д)

Пример использования

```
This text contains an <em>emphasized</em> word.
<strong>Don't assume</strong> that it will be italic!
It was made using the <CODE>EM</CODE> element. A citation
is
typically italic and was no formal necessary structure:
<cite>Moby Dick></cite> is a book title.
```

**FORM тэг в HTML документах**

Синтаксис

FORM тэг определяет форму для заполнения в HTML документе. В одном документе может быть определено несколько форм для заполнения, но вложенные FORM операторы не разрешены. Формат оператора FORM выглядит следующим образом:

```
<FORM ACTION="url" METHOD="POST">...</FORM>
```

Его атрибуты следующие:

- ACTION - URL сервера запросов, куда будет отослано содержание формы после подтверждения. Если это поле отсутствует, будет использован URL текущего документа.
- METHOD - HTTP/1.0 метод используемый для посылки содержания заполненной формы на сервер. Этот метод зависит от того, как работает конкретный сервер запросов. Настоятельно рекомендуется использование метода POST. Возможные варианты следующие:

    - GET - это метод по умолчанию, который приводит к добавлению содержимого заполненной формы к URL, как и в нормальном запросе.
    - POST при использовании этого метода содержимое заполненной формы пересылается не как часть URL, а как содержимое тела запроса.

- ENCTYPE - задает тип кодирования содержимого заполненной формы. Этот атрибут действует только когда используется метод POST и даже в этом случае имеет только одно возможное значение (которое является значением по умолчанию)- application/x-www-form-urlencoded.

Внутри FORM оператора может находиться все, что угодно, кроме другого оператора FORM. Согласно спецификации, для задания интерфейсных элементов внутри оператора FORM используются тэги INPUT, SELECT, и TEXTAREA.

**INPUT** - Тэг INPUT используется для задания простого элемента ввода внутри FORM. Это одиночный тэг, его ничего не окружает и он не имеет закрывающего тэга - т.е. он используется подобно тэгу IMG.

Атрибуты для тэга INPUT следующие:

TYPE - должен быть один из:

- "hidden" : пользователю не предлагается поля для ввода, но содержимое тэга передается при подтверждении и посылке формы. Это значение может быть использовано для передачи информации состояния при взаимодействии клиента и сервера.
- "image" : картинка, по которой вы можете сделать щелчок мышью или другим указывающим устройством, что приводит к немедленному подтверждению и отсылке формы. Координаты выбранной точки измеряются в точках от верхнего левого угла и возвращаются (наряду с другими компонентами формы) точно так же, как для элемента Image.
- "text" поле ввода текста, значение по умолчанию
- "password" (поле ввода пароля; вводимые символы представляются как звездочки)
- "checkbox" (кнопка, принимающая положения on (включено) и off (выключено))
- "radio" (кнопка, принимающая положения on и off; причем остальные кнопки с тем же параметром NAME ведут себя по принципу "одна из многих")
- "submit" (кнопка, действие которой сводится к отсылке содержимого заполненной формы на сервер запросов)

- "reset" (кнопка, которая устанавливает во всех интерфейсных элементах значения по умолчанию)

NAME - символическое имя (оно не показывается) для этого поля ввода. Это поле должно присутствовать для всех полей ввода кроме "submit" и "reset", т.к. оно используется в строке запроса при идентификации поля ввода при посылке ее на сервер после подтверждения формы.
VALUE - для полей ввода текста или пароля, может быть использовано для задания начального содержания поля. Для checkbox или radio button, VALUE задает значение кнопки, когда она находится в отмеченном состоянии (неотмеченные кнопки опускаются при посылке запроса); значение по умолчанию для checkbox или radio button - "on". Для типов "submit" и "reset", VALUE может быть использовано для задания надписи на этих кнопках.
CHECKED (значение не требуется) - указывает, что данная кнопка типа checkbox или radio button отмечена по умолчанию; это поле работает только для кнопок типа checkbox и radio button.
SIZE - физический размер поля ввода в символах; это поле действует только для элементов ввода текста или пароля. Если не присутствует явно, выставляется 20. Многострочные поля ввода текста могут быть заданы с помощью SIZE=ширина,высота; например SIZE=60,12. Заметим, что SIZE атрибут не должен использоваться для задания многострочных полей ввода, т.к. можно воспользоваться тэгом TEXTAREA.
MAXLENGTH - максимальное количество введенных символов, которые будут приниматься для ввода, верно только для полей ввода текста и пароля (и только в однострочных элементах). По умолчанию - неограниченно. Подразумевается, что поля ввода должны прокручиваться.

**Тэг SELECT**

Внутри <FORM> ... </FORM>, может присутствовать любое количество тэгов SELECT, свободно перемешанных с другими HTML элементами (включая INPUT и TEXTAREA элементы) и текстом (но не дополнительных элементов

FORM). Тэг SELECT во многих графических клиентах представляется как меню или список.

В отличие от INPUT, SELECT имеет открывающий и закрывающий тэги. Внутри оператора SELECT разрешена только последовательность тэгов OPTION, за каждым из которых следует некоторое количество простого текста (без HTML выражений), например:

```
<SELECT NAME="a-menu">
<OPTION> First option.
<OPTION> Second option.
</SELECT>
```

Атрибуты оператора SELECT следующие:

- NAME - символическое имя для этого SELECT элемента. Это поле должно присутствовать, т.к. оно используется при посылке запроса (аналогично оператору INPUT).
- SIZE - если SIZE равен 1 или если этот атрибут опущен, по умолчанию SELECT будет представлен как меню опций Motif. Если SIZE = 2 или более, SELECT будет представлен как окно выбора; значение SIZE тогда будет определять, сколько элементов списка будут видны.
- MULTIPLE - если присутствует (нет значения), задает, что SELECT должен позволять множественный выбор из списка. Наличие MULTIPLE принуждает SELECT быть представленным как список выбора, вне зависимости от значения SIZE.

Атрибуты OPTION следующие:

- SELECTED задает, что эта опция выбрана по умолчанию. Если SELECT позволяет множественный выбор (с помощью атрибута MULTIPLE ), как

SELECTED могут быть помечены несколько опций.

- VALUE – значение переменной, передаваемой из формы

**Тэг TEXTAREA**

Тэг TEXTAREA может быть использован для расположения многострокового поля ввода с необязательным содержимым по умолчанию в форме. Атрибуты тэга TEXTAREA следующие:

- NAME символическое имя поля ввода.
- ROWS число строк в поле ввода( высота).
- COLS число столбцов в поле ввода (ширина).

TEXTAREA имеет полосы прокрутки, так что может быть введено любое количество текста. Элемент TEXTAREA требует и открывающий и закрывающий тэги. TEXTAREA без содержания по умолчанию выглядит примерно так:

```
<TEXTAREA NAME="foo" ROWS=4 COLS=40></TEXTAREA>
```

TEXTAREA с содержанием по умолчанию выглядит так:

```
<TEXTAREA NAME="foo" ROWS=4 COLS=40>
Default contents go here.
</TEXTAREA>
```

Содержание по умолчанию должно быть строгим ASCII текстом. Символы перевода строки воспринимаются (так что в примере выше до и после текста "Default contents go here." будет пустая строка).

## Подтверждение и посылка формы

**Для метода GET**

Когда нажимается кнопка submit, содержимое формы будет добавлено к URL в следующей форме:

```
action?name=value&name=value&name=value
```

(здесь "action" - URL, заданное атрибутом ACTION в тэге FORM, или URL текущего документа, если атрибут ACTION не был задан).

Нестандартные символы в примерах "name" или "value" будут опущены, при этом имеются в виду также "=" and "&". Это означает, что включения "=", которые разделяют имена и значения (names и values), и включения "&", которые разделяют пары name/value, не опускаются.

Для полей ввода текста и пароля, значением будет то, что введет пользователь. Если пользователь оставит это поле пустым, значение value также будет пустым , но в строке запроса будет присутствовать фрагмент "name=".

Для кнопок типа checkbox и radio button, значение value определяется атрибутом VALUE в том случае, когда кнопка отмечена. Неотмеченные кнопки при составлении строки запроса игнорируются целиком. Несколько кнопок типа checkbox могут иметь один атрибут NAME (и различные VALUE), если это необходимо. Кнопки типа radio button предназначены для того, чтобы вести себя по принципу "одна из всех" и должны иметь одинаковый атрибут NAME и различные атрибуты VALUE.

**Для метода POST**

Содержимое формы кодируется точно как для метода GET (см. выше), но вместо добавления содержимого формы к URL, заданной атрибутом формы ACTION, в качестве запроса , содержимое посылается блоком данных как часть операции POST. Если присутствует атрибут ACTION, то значение URL, которое там находится, определяет, куда послать этот блок данных. Этот метод особенно

рекомендуется при посылке больших блоков данных.

### *Язык программирования PHP*

PHP был задуман примерно в конце 1994 года Расмусом Ледорфом (Rasmus Lerdorf). Ранние невыпущенные версии использовались на его домашней странице для того, чтобы следить за тем кто просматривал его интерактивное резюме. Первая используемая версия стала доступна где-то в начале 1995 и была известна как Personal Home Page Tools. Она состояла из очень упрощенного движка синтаксического анализатора, который понимал только несколько специальных макрокоманд и ряд утилит, которые затем были в общем использовании на домашних страницах. Гостевые книги, счетчики и некоторые другие дополнения.

Довольно трудно дать какую-либо жесткую статистику, но отмечено, что к 1996 г. PHP/FI был использован по крайней мере на 15,000 веб-сайтах во всем мире. В середине 1997г. эта цифра выросла до более чем 50,000. В середине 1997г. также наблюдалось изменение в разработке PHP. Из частного любимого проекта Расмуса, которому способствовала горстка людей , это превратилось в намного более организованную рабочую группу. Синтаксический анализатор был заново переписан Зевом Сураски(Zeev Suraski) и Анди Гутмансом(Andi Gutmans), и этот новый синтаксический анализатор стал основой для PHP Версии 3.

## Возможности PHP

### HTTP-аутентификация средствами PHP

HTTP аутентификация в PHP доступна только при использовании модуля Apache. В модуле Apache PHP-скрипт, может использовать функцию Header() для отправки сообщения "Authentication Required" браузеру клиента, вызвав тем самым окно диалога Username/Password. Как только пользователь заполняет поля username и password, URL содержащий PHP-скрипт будет вызван заново с

переменными $PHP_AUTH_USER, $PHP_AUTH_PW и $PHP_AUTH_TYPE содержащими введенную информацию. В данном случае обеспечивается только "Основная" аутификация.

Фрагмент примера сценария, который производит аутентификацию клиента на странице, должен быть следующим:

```
<?php
  if(!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"My Realm\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Text to send if user hits Cancel button\n";
    exit;
  } else {
    echo "Hello $PHP_AUTH_USER.<P>";
    echo "You entered $PHP_AUTH_PW as your password.<P>";
  }
?>
```

Вместо просто распечатывания $PHP_AUTH_USER и $PHP_AUTH_PW, Вы вероятно хотели бы проверить имя_пользователя и пароль для проверки правильности. Возможно, посылая запрос к базе данных, или, ища пользователя в dbm или текстовом файле.

Будьте внимательны при использовании браузера Internet Explorer. Он весьма придирчив к порядку заголовков. Отправка заголовка WWW-Authenticate перед заголовком HTTP/1.0 401 возможно даст аутентификацию в любом случае.

Чтобы предотвратить от записи кем-то сценарий , который определяет пароль для страницы, которая была опознана через традиционный внешний

механизм, PHP_AUTH переменные не будут установлены, если допускается внешнее установление подлинности для той специфической страницы. В этом случае может быть использована переменная $REMOTE_USER чтобы идентифицировать внешне-опознанного пользователя.

Обратите внимание, однако, что вышеупомянутое не защищает от кого - то, кто может управлять не-аутентифицированным URL используя перехваченный пароль из аутентифицированных URL на том же самом сервере.

И Netscape и Internet Explorer очистит локальный кэш окна аутентификации после получения ответа сервера 401. Это эффективно как мера отключения пользователей("logout"), вынуждающая их повторно ввести их username и пароль. Некоторые используют это для отключения пользователя по истечении интервала времени("time out"), или обеспечивают кнопку "Log Out".

**Поддержка file upload**

PHP может принимать файлы, загруженные из любого браузера, отвечающего стандартам RFC-1867 (которыми являются, например, Netscape Navigator 3 или старше, Microsoft Internet Explorer 3 с исправлениями от Microsoft, или старше). Эта возможность позволяет людям загружать файлы. С PHP-аутификацией и функциями манипулирования файлами, вы имеете полный контроль над тем, кому позволять загружать файлы и что должно быть выполнено с файлом, если он был загружен.

Экран загрузки файла может быть организован созданием специальной формы, которая выглядит примерно так:

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_"
METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <INPUT NAME="userfile" TYPE="file">
```

```
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

_URL_ должен указать на php html файл. Скрытое поле MAX_FILE_SIZE должно предшествовать полю ввода файла и означает максимально допустимый размер файла. Значение определяется в байтах. Для этого файла при успешной загрузке будут определены следующие переменные :

- $userfile - Временное имя файла под которым загруженный файл загружается в машину сервера.
- $userfile_name - Исходное имя файла в системе отправителя.
- $userfile_size - Размер загруженного файла в байтах.
- $userfile_type - Тип MIME файла, если броузер предоставил эту информацию. Например может быть "image/gif". Обратите внимание, что компонент вышеупомянутых переменных "$userfile" - это любое значение поля Name тега INPUT с TYPE=file обозначенное в форме загрузки. В приведенном выше примере формы загрузки мы назвали его "userfile".

По умолчанию файлы будут сохранены в заданном по умолчанию временном каталоге сервера. Его можно изменить, установкой переменной среды TMPDIR в среде, в которой PHP выполняется. Хотя, использование при ее установке обращения PutEnv () изнутри сценария PHP не будет работать.

Скрипт PHP, который получает загруженный файл, должен определить, что должно быть выполнено с загруженным файлом. Вы можете, например, использовать переменную $file_size, чтобы отбросить любые файлы, которые являются или слишком маленькими или слишком большими. Вы могли бы использовать переменную $file_type, чтобы отбросить любые файлы, которые не соответствуют некоторым критериям типа. В любом случае, вы должны или удалить файл из временного каталога или переместить эго в другое место.

Файл будет удален из временного каталога в конце запроса, если он не

перемещен или переименован.

## Поддержка HTTP cookie

PHP поддерживает HTTP cookies. Cookies - механизм для сохранения данных в удаленном браузере и, таким образом, - трэкинг или идентификация пользователей. Вы можете устанавливать файлы cookie используя функцию setcookie(). Cookies - часть HTTP заголовка, так что функция SetCookie() должна вызваться прежде чем браузеру послан какая-нибудь информация для вывода. Это - то же самое ограничение, которое касается и функции Header().

Любой cookie, посланный Вам от клиента будет автоматически превращен в переменную PHP точно так же как данные методов GET и POST. Если вы желаете назначить множественные значения одиночному cookie - просто добавьте [] к имени cookie. Для более подробной информации см. функцию setcookie ().

## Поддержка баз данных

PHP поддерживает ряд различных баз данных, и в режиме работы в собственной системе команд и через ODBC, включая:

- Adabas D
- MySQL
- dBase
- Oracle
- Empress
- PostgreSQL
- FilePro
- Solid
- Informix
- Sybase
- InterBase
- Velocis

- mSQL
- Unix dbm

## Регулярные выражения

Регулярные выражения используются для сложного манипулирования строками в PHP. Функции, которые поддерживают регулярные выражения:

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()

Все эти функции принимают строку регулярного выражения как их первый параметр. PHP использует расширенные регулярные выражения POSIX как определено в POSIX 1003.2. Для полного описания регулярных выражений POSIX см. соответствующие разделы руководства (regex), в каталоге regex дистрибутива PHP.

### Пример регулярных выражений

```
ereg("abc",$string);
/* Возвращает 'истина', если "abc"
   найдено в строке $string. */

ereg("^abc",$string);
/* Возвращает 'истина', если "abc"
   найдено в начале строки $string. */

ereg("abc$",$string);
/*  Возвращает 'истина', если "abc"
```

```
   найдено в конце строки $string. */

eregi("(ozilla.[23]|MSIE.3)",$HTTP_USER_AGENT);
/*  Возвращает 'истина', если браузер клиента
   - Netscape 2, 3 или MSIE 3. */

ereg("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)",
     $string,$regs);
/* Помещает три слова - $regs[1], $regs[2] и
$regs[3], разделенные пробелом. */

ereg_replace("^","<BR>",$string);
/* Устанавливает тег <BR> в начало строки $string. */

ereg_replace("$","<BR>",$string);
/* Устанавливает тег <BR> в конец строки $string. */

ereg_replace("\n","",$string);
/* Отсекает символ "возврат каретки" в строке
$string. */
```

**Обработка ошибок**

В PHP есть 4 типа ошибок и предупреждений. Это:

- 1 - Нормальные Ошибки Функции(Normal Function Errors)
- 2 - Нормальные Предупреждения(Normal Warnings)
- 4 - Ошибки Синтаксического Анализатора(Parser Errors)
- 8 - Уведомления(Notices) : предупреждения, которые Вы можете проигнорировать но, которые могут подразумевать ошибки в вашем коде

Эти 4 типа комбинируются при определении ошибки, сообщая уровень.

Ошибка по умолчанию, возвращает уровень 7, который является комбинацией 1 + 2 + 4, или все ошибки за исключением примечаний. Этот уровень может быть изменен в файле php3. ini директивой error_reporting. Он также может быть установлен в вашем файле Apache httpd.conf директивой php3_error_reporting, или же это может быть произведено во времени выполнения сценария, с использованием функции error_reporting ().

Все выражения PHP могут также вызываться с префиксом "@", который выключает сообщение об ошибке, специфичное для этого выражения. Если ошибка произошла во время выполнения такого выражения, и допускается возможность track_errors, Вы можете найти сообщения об ошибках в глобальной переменной $php_errormsg.

## Синтаксис и грамматика

Синтаксис PHP заимствован непосредственно из C, Java и Perl также повлияли на синтаксис данного языка.

## Переход из HTML

Есть три способа выхода из HTML и перехода в "режим PHP кода":

```
1.  <? echo("простейший способ, инструкция обработки
SGML\n"); ?>

2.  <?php echo("при работе с XML документами делайте
так\n"); ?>

3.  <script language="php">
        echo ("некоторые редакторы (подобные FrontPage)
не любят обрабатывающие инструкции");
    </script>;
```

```
4.  <% echo("От PHP 3.0.4 можно факультативно применять
ASP-тэги"); %>
```

## Разделение инструкций

Инструкции (утверждения) разделяются также как в C или Perl - точкой с запятой.

Закрывающий тэг (?>) тоже подразумевает конец утверждения, поэтому следующие записи эквивалентны:

```
<php
    echo "Это тест";
?>

<php echo "Это тест" ?>
```

## HTML Формы (GET и POST)

Когда программой-обработчиком формы является PHP-скрипт, переменные этой формы автоматически доступны для данного скрипта PHP. Например, рассмотрим следующую форму:

```
<form action="foo.php3" method="post">
    Name: <input type="text" name="name"><br>
    <input type="submit">
</form>
```

При активизации формы PHP создаст переменную $name, значением которой будет то содержимое, которое было введено в поле Name: данной формы.

PHP также воспринимает массивы в контексте переменных формы, но только одномерные. Вы можете, например, группировать взаимосвязанные переменные вместе или использовать это свойство для определения значений переменных при множественном выборе на входе:

Более сложные переменные формы:

```
<form action="array.html" method="post">
    Name: <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog
        <option value="guinness">Guinness
        </select>
    <input type="submit">
</form>
```

Если PHP-атрибут track_vars включен, через установку конфигурации track_vars или директивой <?php_track_vars?>, тогда переменные, активизированные посредством методов POST или GET, будут также находиться в глобальных ассоциативных массивах $HTTP_POST_VARS и $HTTP_GET_VARS соответственно.

PHP очевидным образом поддерживает HTTP cookies, как это определено в Netscape's Spec. Cookies являются механизмом хранения данных в удаленном браузере, используемым для поддержки процедуры обмена или идентификации ответа пользователя. Cookies можно устанавливать используя функцию SetCookie(). Cookies являются частью заголовка HTTP, поэтому функция SetCookie() должна вызываться прежде чем какие-либо передаваемые данные

посылаются браузеру. Это такое же ограничение, как и для функции Header(). Любые cookies, посылаемые вам клиентом, автоматически преобразовываются в переменные PHP, также как данные методов GET и POST.

Если необходимо назначить множественные значения одному cookie, просто добавьте квадратные скобки [ ] к имени cookie. Например:

```
SetCookie ("MyCookie[]", "Testing", time()+3600);
```

Учтите, что текущий cookie заменит предыдущий с тем же именем в вашем браузере, если только путь или домен не являются различными. Поэтому, при работе с программами обслуживания карт вы можете использовать для сохранения данных счетчик и посылать его значения дальше и т.п.

Пример функции SetCookie:

```
$Count++;
SetCookie ("Count", $Count, time()+3600);
SetCookie ("Cart[$Count]", $item, time()+3600);
```

## Переменные окружения

PHP автоматически создает переменные окружения, как и обычные переменные.

```
echo $HOME;  /* Показывает переменную окружения HOME,
если она установлена. */
```

Хотя при поступлении информации механизмы GET, POST и Cookie также атоматически создают переменные PHP, иногда лучше явным образом

прочитать переменную окружения, для того чтобы быть уверенным в получении ее правильной версии. Для этого может использоваться функция getenv(). Для установки значения переменной окружения пользуйтесь функцией putenv().

Кроме вышеперечисленных, PHP включает в себя массу различных функций, что делает его незаменимым инструментом при создании информационных систем.

# Лабораторная работа №1. Создание простой HTML-странички

# Лабораторная работа №2. Использование дополнительных элементов HTML для оформления страницы

# Лабораторная работа №3. Создание простейшего PHP-скрипта

# Лабораторная работа №4. Создание HTML-форм и их обработка PHP-скриптом