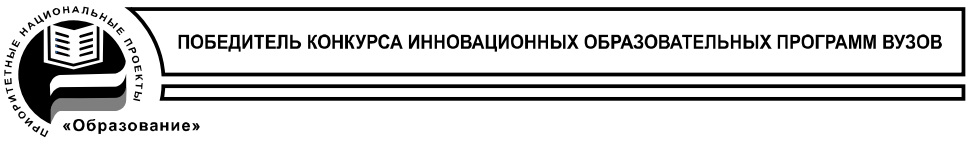
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ



А.О. Казначеева

ОСНОВЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Учебное пособие

Санкт-Петербург

2009

УДК 004.9

Казначеева А. О. Основы информационных технологий. Учебное пособие. – СПб.: СПбГУ ИТМО, 2009. – 44 с.

Рецензент: д.т.н., проф. Сизиков В.С., СПбГУ ИТМО

В учебном пособии рассмотрены основы теории представления алгоритмов, структура операционных систем Linux и Unix, архитектура и протоколы компьютерных сетей, принципы формирования локальных и глобальных сетей и передачи информации. Большое внимание уделено формам представления графической информации, современным методам обработки изображений (линейные и нелинейные фильтры), критериям оценки их качества. В пособии рассмотрены основные команды языка HTML, используемые для создания документов в глобальной сети Internet.

Пособие предназначено для обучения студентов в рамках основной образовательной программы по направлению подготовки бакалавров 200100 «Приборостроение» и в рамках программы подготовки дипломированных специалистов по специальности 200101.65 «Приборостроение» по дисциплине ЕН.Р.02 «Информатика (специальные разделы). Печатается при поддержке РФФИ, грант №08-08-00922а.

Утверждено к печати Ученым советом ФТМиТ, протокол №4 от 10.02.09.

В 2007 году СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007–2008 годы. Реализация инновационной образовательной программы «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий» позволит выйти на качественно новый уровень подготовки выпускников и удовлетворить возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях экономики.

Оглавление

ВВЕДЕНИЕ

В последнее десятилетие информационные технологии стали одним из важнейших факторов, влияющих на развитие общества. Информатизация охватывает экономическую и социальную сферы, науку, образование, культуру, сделав необходимыми теоретические и практические знания в области информационных технологий для специалистов различного профиля. В широком смысле *информатика* есть наука о вычислениях, хранении и обработке информации и включает дисциплины, так или иначе относящиеся к вычислительным машинам, компьютерным системам и сетям: вычислительная техника, анализ алгоритмов, разработка языков программирования и компиляторов, теория вычислимости, искусственный интеллект.

Задачами информатики являются решение научных и инженерных проблем, решение специфических задач с максимальной эффективностью, хранение и восстановление информации, взаимодействие оператора и программ и т.п. Практические результаты информатики постоянно совершенствуются и способствуют научно-техническому прогрессу и развитию общества. В настоящее время термин "информатика" все чаще заменяется более содержательным термином "информационные технологии" (ИТ), обозначающим с одной стороны, разработку, проектирование и производство компьютеров, периферии и элементной базы для них, сетевого оборудования, алгоритмического и системного программного обеспечения, а с другой – их применение в приборах и системах самого различного назначения.

Специалисты в области технических наук должны владеть базовыми понятиями об алгоритмах и алгоритмических моделях, формальной грамматикой представления алгоритмов, знать особенности различных языков программирования, иметь представление об основных утилитах и командах операционных систем Unix и Linux, используемых в современных приборных комплексах. Помимо этих вопросов в учебном пособии рассмотрены формы представления графической информации, основные методы обработки двумерных сигналов и критерии оценки полученных результатов, часто используемые при обработке графической измерительной информации.

Учебное пособие представляет собой часть учебно-методического комплекса дисциплин ЕН.Р.02 "Информатика (специальные разделы)" и ОПД.Ф.09 "Компьютерные технологии в приборостроении", ФТД.02 "Искусственный интеллект в приборостроении" учебного плана подготовки бакалавров по направлению 200100 "Приборостроение".

АЛГОРИТМЫ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Алгоритмы и их свойства

В повседневной практике специалисты сталкиваются с множеством разноплановых задач (прогнозирование результатов, построение оптимальной модели, описание траектории движущегося объекта), порядок решения которых имеет общие моменты: выделение системы, построение ее информационной модели и установка порядка обработки данных. В общем случае обработка состоит в преобразовании по некоторым правилам исходных данных, в результате чего появляется новая информация.

Алгоритм – это точно определенная (однозначная) последовательность простых (элементарных) действий, обеспечивающих решение любой задачи из некоторого класса [1, 17]. Алгоритмам характерны следующие общие свойства:

1) дискретность – алгоритм можно разделить на отдельные шаги (действия), выполнение каждого из которых возможно только после завершения всех операций на предыдущем шаге;

2) детерминированность – совокупность промежуточных величин на любом шаге однозначно определяется системой величин, имевшихся на предыдущем шаге (результат алгоритма определяется только входными данными и шагами самого алгоритма);

3) элементарность шагов – закон получения последующей системы величин из предыдущей должен быть простым и локальным;

4) направленность – если способ получения последующих величин из каких-либо исходных не приводит к результату, то должно быть указано, что следует считать результатом алгоритма;

5) массовость – начальная система величин может выбираться из некоторого множества (т.е. один алгоритм может применяться для решения класса задач).

Уточнение понятия алгоритма происходит в рамках алгоритмических моделей. Модель определяет набор средств, использование которых допустимо при решении задач.

Алгоритмические модели могут быть теоретическими и практическими. Теоретические модели более универсальные (позволяют описать любой алгоритм) и максимально простые (используют минимум средств для решения задачи). Требование простоты важно для выделения необходимых элементов и свойств алгоритма и обеспечивает доказательство общих утверждений об этих свойствах. В практических и прикладных моделях более значимым является удобство программирования и эффективность вычислений, поэтому их средства (набор элементарных шагов) намного сложнее и разнообразнее, что часто затрудняет теоретический анализ.

Развитие и совершенствование компьютерной техники повлекло создание различных алгоритмов, обеспечивающих решение многочисленных прикладных задач. Алгоритмические задачи делятся на два класса:

1) вычисления значений функции;

2) распознавание принадлежности объекта заданному множеству.

Обычно наиболее эффективным для решения данной задачи считают алгоритм, обеспечивающий наиболее быстрое получение результата, поскольку на практике именно ограничения по времени являются доминирующим фактором. Алгоритмы, предназначенные для решения однотипных задач, называются эквивалентными. Поскольку исполнение любого алгоритма требует определенного объема машинной памяти и времени центрального процессора, то понятие эффективности алгоритма связано со всеми вычислительными ресурсами, необходимыми для работы алгоритма.

Формализация представления алгоритмов

Абстрактные алгоритмические модели используются лишь при построении теории и доказательстве общих свойств алгоритмов, но для практических целей такое представление чаще всего неудобно. Поскольку любой алгоритм является набором входных, промежуточных и выходных данных, то для его описания и системы правил преобразования служит определенный язык. Естественные языки являются изменчивыми, неоднозначными и избыточными и не подходят для записи алгоритмов, требующих однозначной определенности. Наиболее простой путь устранения этих недостатков – построение искусственных языков со строгим синтаксисом и полной смысловой определенностью. Такие языки получили название *формальных*. В любом языке можно выделить две составляющих – синтаксис и семантику.

Синтаксис (грамматика языка) – совокупность правил, согласно которым в данном языке стоятся конструкции.

Семантика – смысловая сторона языка, соотносит единицы и конструкции языка с некоторым внешним миром, для описания которого язык используется.

Синтаксис формального языка задается некоторой системой правил, которая из небольшого набора исходных конструкций порождает все допустимые их комбинации, т.е. язык образуется как множество разрешенных правилами сочетаний исходных конструкций. Кроме того, синтаксис содержит формулировку условия, которое выполняется для законченных конструкций языка и не выполняется в противном случае. Наиболее наглядным способом описания формального языка является синтаксическая диаграмма.

Синтаксическая диаграмма – схема (графическое представление) описания какого-либо нетерминального символа языка-объекта. Схема всегда имеет один вход и один выход, а её элементы соединяются между собой направленными линиями, указывающими порядок следования объектов в определенном нетерминальном символе (рис. 1).

В представлении алгоритмов можно выделить две основные формы: символьную (словесную) и графическую.

Основным способом представления алгоритмов является строчная форма записи, представляющая собой последовательность строк, каждая из которых содержит описание одного или нескольких элементарных действий. Ло-

гика алгоритма (порядок действий) задается в явном виде путём указания метки последующей строки (в виде порядковых чисел или букв), или в неявном – по умолчанию передается строке, следующей за выполненной. Данный способ позволяет записать алгоритмическую нотацию для любого исполнителя – как человека, так и технического устройства. Недостатком строчной формы является неудобство целостного восприятия его логической структуры.

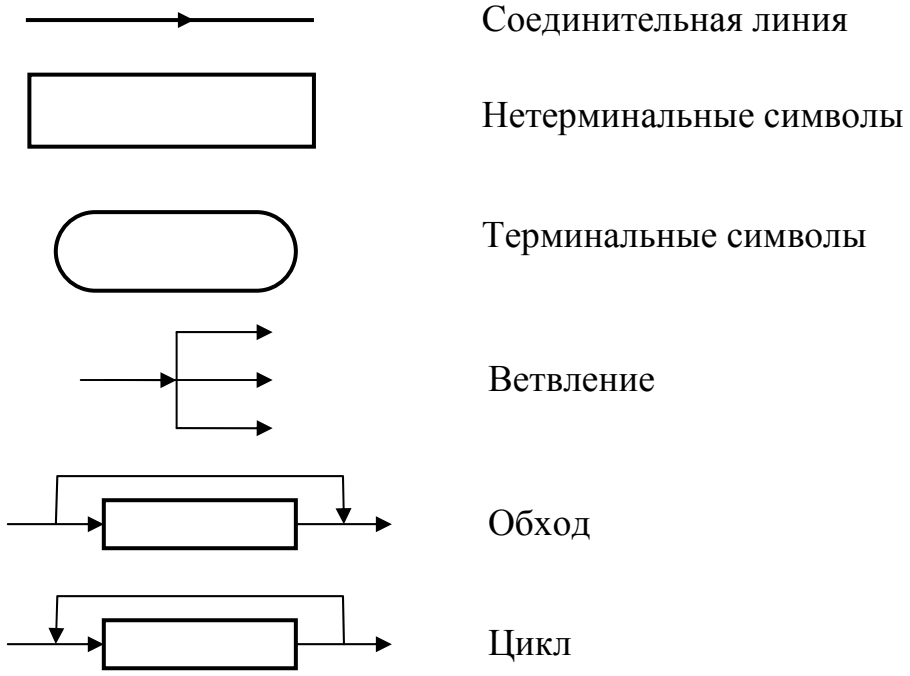


Рис. 1. Элементы синтаксической диаграммы

Формами строчной записи алгоритмов являются:

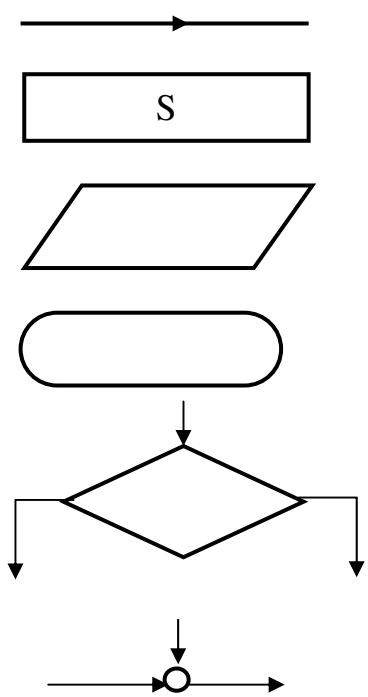
1) пошагово-словесная форма – пронумерованная последовательность строк, содержащих описания конкретных действий на естественном языке;

2) формула – строчная запись действий, обеспечивающих обработку числовых, символьных или логических данных;

3) псевдокод – ориентированный на исполнителя «человек» частично формализованный язык, позволяющий записывать алгоритмы в форме, близкой к англоподобным языкам программирования;

4) язык программирования – искусственный формализованный язык, предназначенный для записи алгоритма для исполнителя «компьютер», метаязыком которого является естественный язык.

Графическая форма записи или блок-схема для представления отдельных блоков алгоритма использует набор геометрических фигур (рис. 2) и предназначена только для исполнителя «человек», что является её основным недостатком. Достоинство данной формы записи заключается в наглядности: блок-схема позволяет охватить весь алгоритм сразу, отследить различные варианты его исполнения, позволяет сделать записи, как на естественном, так и на формальном языках.



Соединительная линия

Обработка данных

Ввод-вывод данных

Начало или конец алгоритма

Проверяемое логическое условие

Объединение

Рис. 2. Элементы блок-схемы при графической записи алгоритма

Классификация языков программирования

Язык программирования – искусственный формализованный язык, представляющий собой набор ключевых слов (словарь) и система правил (грамматических и синтаксических) для конструирования операторов, состоящих из групп или строк чисел, букв, знаков препинания и других символов, с помощью которых люди могут сообщать компьютеру набор команд. Язык программирования строго фиксирует изображение управляющих структур, описание допустимых действий, синтаксические правила построения сложных структур. По сложности языки программирования разделяют на:

- языки низкого уровня (машинно-ориентированные);
- языки высокого уровня (машинно-независимые).

К машинно-ориентированным языкам относятся:

- машинный язык (язык машинных кодов) – совокупность команд, интерпретируемых и исполняемых компьютером; каждый оператор является машинной командой, а данные в ОЗУ размещены по абсолютным значениям адресов;
- ассемблер (макроассемблер) – язык символического кодирования, где операторами являются машинные команды, которым приписываются искусственные обозначения, а в качестве операндов используются символические имена адресов в ОЗУ.

Пример команд ассемблера:

CLA – очистить один из регистров сумматора (аккумулятор);

ADD – сложение содержимого ячейки, номер которой написан после команды, с содержимым аккумулятора (результат остается в аккумуляторе);

MOV – перемещение содержимого аккумулятора в ячейку с указанным номером;

HLT – стоп.

Запись алгоритма происходит в редакторе с ASC-кодировкой. Для

преобразования текста в последовательность машинных команд необходима промежуточная программа – *компилятор*. На этапе компиляции производится распределение данных в ОЗУ, при этом вместо имен переменных подставляются относительные адреса ячеек, в которых располагаются данные. Абсолютные данные присваивает операционная система при размещении программы в ОЗУ компьютера перед ее использованием.

Поскольку ассемблер – машинно-зависимый язык, то записанная на нем программа может выполняться только на той технике (тем типом процессора), ассемблер которого был использован. Этот недостаток отсутствует у языков высокого уровня, которые ориентированы не на систему команд той или иной машины, а на систему операторов, характерных для записи определенного класса алгоритмов (операторы присваивания, условные операторы, циклы, операторы ввода-вывода).

Таблица 1

Различия парадигм программирования

Парадигма программирования	Представление программ и данных	Исполнение программы	Связь частей программы между собой
Процедурное	Программа и данные представляют собой не связанные друг с другом элементы	Последовательное выполнение операторов	Возможна только через совместно обрабатываемые данные
Объектно-ориентированное	Данные и методы их обработки инкапсулированы в рамках единого объекта	Последовательность событий и реакций объектов на эти события	Отдельные части программы могут наследовать методы и элементы данных друг у друга
Логическое	Данные и правила их обработки объединены в рамках единого логического и структурного образования	Преобразование логического образования в соответствии с логическими правилами	Разбиение программы на отдельные независимые части затруднительно

По функциональному назначению языки программирования высокого уровня разделяют на:

1) проблемно-ориентированные – предназначены для решения специфических задач из некоторой отрасли знаний:

- Fortran (formula translator) – язык решения сложных научных и инженерных задач, первый язык высокого уровня;
- COBOL (common business oriented language) – язык для решения экономических и коммерческих задач;
- Algol (algorithmic language) – языки решения научно-технических задач;

- LISP (list processing language) – язык для решения задач искусственного интеллекта;

2) универсальные – позволяют решить любую задачу, хотя трудоемкость решения в разных языках будет отличаться [11]:

- PASCAL (Philips automatic sequence calculator);
- BASIC (Beginner ALL-purpose symbolic instruction code);
- C/C++;
- Jawa;
- современные среды визуального программирования DELPHI, Visual Basic.

Другой признак классификации – парадигма (концепция) программирования – совокупность основополагающих идей и подходов, определяющих модель представления данных и их обработки, а также методологии программирования (табл. 1).

В настоящее время в распоряжении программиста имеется весьма обширный спектр языков-инструментов, из которых для любой конкретной задачи можно выбрать тот, что позволит решить ее оптимальным путем [6].

Операционные системы

Операционная система (ОС) – комплекс системных и управляющих программ, предназначенных для наиболее эффективного использования всех ресурсов вычислительной системы и удобства работы с ней.

В программном обеспечении ОС занимает основное положение, поскольку осуществляет планирование и контроль всего вычислительного процесса, рациональное распределение ресурсов между отдельными решаемыми задачами; предоставление пользователям сервисных средств, облегчающих процесс программирования и отладки задач. Различные ОС на одних и тех же технических средствах могут предоставить пользователю различные возможности для организации вычислительного процесса или автоматизированной обработки данных [12].

UNIX

Операционная система Unix родилась случайно в конце 60-хгг. Кен Томпсон написал несложную компьютерную игру для ламповых ЭВМ, но поскольку ему не понравилась ни ее скорость, ни стоимость, он переписал ее для работы на компьютере DEC и разработал целую операционную систему. В 1972 Unix работала на 10 компьютерах Массачусетского университета, а в 1973 Томсон и Ритчи переписывают ядро с языка ассемблер на язык С. Рост Unix частично обязан легкости переноса языка С на другие платформы. И хотя язык С не так быстр, как ассемблер, он более гибок и мобилен при переходе от одной системы к другой.

Unix состоит из двух ключевых компонентов – ядра (**kernel**) и оболочки (**shell**) или командного процессора. Оболочки действуют как интерпретаторы между пользователем и ядром. Структура файлов и каталогов ОС Unix представляет модель перевернутого дерева, как и в других ОС [9, 18].

Корневой каталог (**root** derectory) является первым слоем файловой

системы и обозначается как прямой слэш **/**. В Unix слово root используется как обозначение всего начального или высшего. Корневой пользователь (root user, superuser) может изменять всё в файловой системе без ограничений. Внутри нормального системного корневого каталога должен быть только один файл – **kernel** – загружаемый файл ОС.

UNIX-системы содержат следующие стандартные подкаталоги: bin, dev, etc, lib, shlib, tcb, tmp, usr, var. Остановимся на них подробнее.

Каталог **bin** содержит двоичные (binary) и выполняемые файлы, многие команды и утилиты, в т.ч. логины, пароли, общие команды (find, kill,date и др.) и доступен для всех пользователей.

Каталог **dev** содержит информацию об устройствах системы: ***clock*** (системные часы), ***dsk*** (жесткий диск), ***null*** (образ устройства, с помощью которого ошибки могут быть переназначены в целях предотвращения их вывода на экран или в файл). Файлы каталога делятся по типу устройства, за которое они отвечают и начинаются с символов **b** (файлы блочных устройств, способные считывать данные блоками, например, ленточные накопители) или **c** (файлы символьных устройств подобные терминалу, характеризуются способностью чтения по символу).

Каталог **etc** применяется для указания продолжения списка подобными значениями и информация из него не переносится на другие компьютеры. В каталоге находятся файлы: **inittab** (информация о доступных устройствах), **issue** (текст, появляющийся на экране при входе в систему), **magic** (база данных типов файлов, распознаваемых системой) **motd** (дата, появляющаяся при каждой регистрации входа в систему), **profile** (действия, автоматически выполняемые при каждом начатом сеансе работы), **shadow** (зашифрованные пароли из файла паролей), **timezone** (информация о часовом поясе).

Каталог **lib** содержит библиотеки для различных компиляторов системы. Каталоги **lost** и **found** хранят удаленные системой поврежденные файлы.

Каталог **shlib** (shared library) содержит версии общей библиотеки программ, используемых компиляторами.

В каталоге **tcb**, отвечающем за вход в систему и пароли, находятся файлы, необходимые для поддержки системной защиты.

Копии файлов, открытых или используемых системой, помещаются в каталог **tmp**.

Каталог **usr** хранит данные об особенностях конкретных пользователей. Его подкаталоги: **adm** (главный административный каталог – используется для мониторинга, ведения учетных данных пользователей, отчетности об ошибках и использовании файлов, содержит сценарии инсталляции); **bin** (содержит выполняемые двоичные файлы); **include** (хранит неоткомпелированные процедуры, обычно используемые Unix при компиляции ядра); **lib** (содержит таблицы, необходимые для поддержания системной работы, в т.ч. файлы и каталоги, влияющие на работу терминалов, учетные данные пользователей); **man** (руководства, описывающие работу каждой команды Unix); **spool** (временное хранилище данных, поступающих в систему из

произвольного источника).

Каталог **var** содержит информацию, характерную для конкретной системы (в отличии от **usr**).

Операции в UNIX-системах осуществляются путём ввода команд (табл. 2), синтаксис которых доступен в файлах помощи (директория man).

Таблица 2

Основные команды в UNIX-системах

Команда	Описание
cd	смена каталога cd / - в корневой каталог
..	на один уровень назад
pwd	выводит имя текущего каталога
ls	выводит содержимое каталога
-a	выводит все файлы, в т.ч. скрытые
-C	выводит данные в несколько колонок
-F	показывает тип файла (каталог или выполняемый файл)
-l	выводит данные в длинном формате
-d	показывает только имена каталогов
-R	выполняет рекурсивный просмотр
-r	выводит список в обратной последовательности
-t	файлы по времени модификации
-u	упорядочивает файлы по времени доступа
touch имяфайла	создание файла
cp	копирование файла
cat	просмотр небольших текстовых файлов целиком
wc файл	счётчик слов
more файл	поэкранный просмотр файлов
pg файл	поэкранный просмотр файлов в фоновом режиме
rm	удаление файла (необратимый процесс)
mv	перемещение и переименование файла mv старое_имя новое_имя mv старое_имя /новый_каталог/новое имя
ln	файл, известный под разными именами ln старое_имя новое_имя

Linux

Linux (полное название GNU/Linux) – общее название UNIX-подобных ОС на основе одноимённого ядра и собранных для него библиотек и системных программ, разработанных в рамках проекта GNU. В отличие от большинства других операционных систем, GNU/Linux не имеет единой «официальной» комплектации, поэтому перед началом её установки целесообразно выяснить, совместимо ли с ней аппаратное обеспечение компьютера. Все физические устройства компьютера с ОС Linux отображаются в каталог **/dev** файловой системы Linux [16]. Для входа в систему необходима идентификация пользователя (права администратора дает команда **su**). При подключении пользователя запускается новая командная оболочка (**shell**) и появляется строка, содержащую символ "**$**" (далее, этот символ

будет обозначать командную строку). В Linux доступны командные оболочки **bash** (самая распространенная), **pdksh** (аналог оболочки в UNIX системах), **tcsh** (версия C shell) и **zsh**, каждая из которых имеет свой синтаксис. Для проверки используемой оболочки служит команда:

echo $shell

Файловая система ОС Linux является единым деревом с размещением файлов в зависимости от того, к какой программе они относятся. Основными частями Linux являются файловые системы **root**, **/usr**, **/var** и **/home**. Каталог **root**, обозначаемый "**/**", содержит все файлы, необходимые для загрузки системы, средства для восстановления поврежденных файловых систем и для работы с резервными копиями.

Корневой каталог Linux обычно не содержит файлов, хотя в нем может находиться системный файл ядра (обычно он называется **/vmlinuz**), загружаемый в память при старте системы. Остальные файлы располагаются в следующих подкаталогах: **/bin** (командные оболочки, основные утилиты), **/boot** (ядро системы), **/dev** (псевдофайлы устройств, позволяющие работать с ними напрямую), **/etc** (файлы конфигурации), **/home** (личные каталоги пользователей), **/lib** (системные библиотеки, модули ядра), **/mnt** (каталоги для монтирования файловых систем сменных устройств и внешних файловых систем), **/proc** (виртуальная файловая система), **/root** (личный каталог администратора системы), **/sbin** (системные утилиты), **/usr** (программы и библиотеки, доступные пользователю, и документация в /usr/share/doc), **/var** (рабочие файлы программ, очереди, журналы), **/tmp** (временные файлы). Текущий каталог обозначается "**.**".

В каталоге **/dev** находятся файлы устройств, создаваемые при установке системы, а затем с помощью файла **/dev/MAKEDEV**. Файл /dev/MAKEDEV.local используется при создании локальных файлов устройств или ссылок (т.е. не соответствующих стандарту MAKEDEV).

Каталог **/etc** содержит командные файлы, выполняемые при запуске системы или при смене ее режима работы **(/etc/rc**), базу данных пользователей (**/etc/passwd**, **/etc/group**), список файловых систем (**/etc/fstab, /etc/mtab**), файлы конфигурации (**/etc/inittab, /etc/login.defs**), список рабочих оболочек (**/etc/shells**), описания форматов файлов (**/etc/magic**). Здесь же хранится теневая база данных пользователей (**/etc/shadow**), при этом информация из файла **/etc/passwd** перемещается в **/etc/shadow**, который не доступен для чтения всем, кроме пользователя **root**. Список терминалов, с которых может подключаться к системе пользователь **root**, хранится в подкаталоге **/etc/securetty**.

В файловой системе **/home** хранятся личные каталоги пользователей, что упрощает резервное копирование информации.

Каталог **/mnt** содержит узлы монтирования для временных файловых систем и может быть разбит на несколько подкаталогов (например, каталог **/mnt/dosa** может использоваться для доступа к дисководу с применением файловой системы MS-DOS, а **/mnt/exta** – для доступа с системой ext2fs).

Файловая система **/proc** является виртуальной (ядро создает ее в памяти компьютера) и предоставляет информацию о текущем состоянии системы (изначально только о процессах). Для каждого процесса существует

отдельный каталог в **/proc**, именем которого является его числовой идентификатор (например, **/proc/1** – каталог, содержащий информацию о процессе номер 1). В каталоге хранится информация о процессоре (**/proc/cpuinfo**), список драйверов устройств, встроенных в действующее ядро (**/proc/devices**), задействованные в данный момент порты ввода/вывода (**/proc/ioports**), загруженность системы (**/proc/loadavg**), информация об использовании памяти (**/proc/meminfo**) и сетевых протоколах (**/proc/net**), статистическая информация о работе системы (**/proc/stat**) и др.

В каталоге **/usr** находятся все команды, программы, библиотеки, страницы руководств (**/usr/man**, **/usr/info**, **/usr/doc**) и другие файлы, требуемые для нормального функционирования системы и помещаемые туда при её установке. Отдельно устанавливаемые пакеты программ и другие файлы размещаются в каталоге **/usr/local**. Ни один из файлов этой системы не должен быть специфичным для какой-либо отдельной машины и не должен быть изменен при обычной работе системы (режим read-only). Обычно файловая система **/usr** достаточно большая по объему. Неизменяемые файлы данных для программ и подсистем, включая некоторые конфигурационные файлы, расположены в подкаталоге **/usr/lib**

Каталог **/var** содержит файлы, изменяемые во время работы системы: изменяемые данные для программ, установленных в **/usr/local** (**/var/local**), файлы, изменяемые при нормальном функционировании системы (**/var/lib**), буферные каталоги для почты, новостей и т.д. (**/var/spool**), журнальные файлы различных программ (**/var/log**), временные файлы (**/var/tmp**), файлы-защёлки (**/var/lock**). Она специфична для каждого компьютера и не может быть разделена в сети между несколькими машинами.

Смена каталога осуществляется командой **cd** (change directory), которая работает как с относительными (например, **cd docs/** для перехода в подкаталог **docs/**), так и с абсолютными путями (например, **cd /usr/bin/** для перехода в каталог **/usr/bin**). Варианты команды:

cd .. возврат в родительский каталог;
cd - возврат в предыдущий каталог;
cd переход в домашний каталог.

Команда **ls** (list) выдает список файлов в текущем каталоге и имеет две основные опции: **-a** (просмотр всех файлов, включая скрытые); **-l** (отображение более подробной информации).

Команда **mkdir** позволяет создать новый каталог в указанном существующем подкаталоге системы.

Команда **rm** имя_файла служит для удаления файлов без возможности восстановления. Возможные параметры: **-i** (запрос на удаление файла); **-r** (удаление включая подкаталоги и скрытые файлы). Например, удаление всех файлов html в каталоге html: rm -i ~/html/*.html. для удаления пустого каталога служит команда **rmdir** имя_каталога.

Команда **ps** выводит список текущих процессов с указанием имени процесса и его номера (колонка PID). Номер процесса может использоваться для его завершения командой **kill** (kill номер_процесса). Если желаемый эффект не достигнут, используют команду kill -9 номер_процесса.

КОМПЬЮТЕРНЫЕ СЕТИ

С развитием вычислительной техники появились и приобрели широкое использование системы физического соединения двух или более компьютеров – *компьютерные сети*. По территориально-организационным признакам (количеству машин и расстоянию между ними) компьютерные сети принято разделять на локальные (local area network, LAN) и глобальные (wide area network, WAN) [8, 10]. Примерами локальных могут быть сети вуза, предприятия, нескольких фирм, находящихся недалеко друг от друга. Глобальные сети (например, Internet) распространяют свое действие по всему миру и используют все каналы связи, включая спутниковые.

Архитектурный принцип построения большинства сетей называется "клиент-сервер". Сервер – компьютер сети, предоставляющий свои программные и аппаратные ресурсы пользователям сети для хранения данных, выполнения программ и других услуг. Клиент – компьютер сети, пользующийся услугами сервера; в его роли часто выступают программы, имеющие доступ к информационным ресурсам или устройствам сервера.

Термины "клиент" и "сервер" используются для обозначения как программных, так и аппаратных средств.

Для передачи данных в сети используются специальные стандарты, обеспечивающие их совместимость – *сетевые протоколы*. Примером универсального протокола является семейство TCP/IP, широко применяющееся во всем мире для объединения компьютеров в сеть Internet которая состоит из множества сетей различной физической природы, от локальных сетей типа Ethernet и Token Ring, до глобальных сетей типа NSFNET. История его возникновения связана с задачей, поставленной после второй мировой войны правительством США. Требовалось создать единую сеть, которая могла бы своими средствами находить маршруты передачи данных, а также в случае повреждения некоторых каналов связи перенаправлять поток информации по другим каналам. При реализации этого проекта были созданы отдельные представители семейства протоколов TCP/IP.

Сетевой протокол TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) – сетевой протокол, используемый в Internet и в большинстве UNIX-систем, включающий протокол транспортного уровня (TCP) и протокол сетевого уровня (IP), отвечающий за передачу блоков данных. Первоначально используемый только в UNIX-сетях, сейчас TCP/IP применяется практически во всех видах локальных сетей, что значительно уменьшает количество проблем, связанных с потоками данных. TCP/IP включает в себя File Transfer Protocol (FTP), Terminal Emulation (TELNET) и Simple Transfer Protocol (SMTP).

Протоколы семейства TCP/IP можно представить в виде четырехуровневой модели. Нижним является уровень сетевого интерфейса, содержащий протоколы, обеспечивающие взаимодействие TCP/IP с физической сетью. Компоненты на уровне сетевого интерфейса отвечают за отправку в сеть и прием из сети кадров, содержащих пакеты информации.

Компоненты второго (межсетевого) уровня решают задачи адресации сообщений, преобразования логических адресов и имен в физические, управления подсетями, определения маршрутов от источника сообщения к узлу назначения. К протоколам этого уровня относятся:

1) Internet Protocol (IP) – протокол Internet, отвечающий за доставку пакетов данных и обеспечивающий адресацию узлов и маршрутизацию данных между ними;

2) Address Resolution Protocol (ARP) – протокол, позволяющий преобразовывать логические IP-адреса узлов в MAC-адреса канального уровня;

3) Internet Control Message Protocol (ICMP) – протокол управляющих сообщений, осуществляющий отправку извещений об ошибках доставки пакетов;

4) Internet Group Management Protocol (IGMP) – протокол управления группами, обеспечивающий поддержку узлами функций управления маршрутизацией.

Третий (транспортный) уровень обеспечивает непосредственное взаимодействие узлов и включает два протокола:

1) Transmission Control Protocol (TCP) обеспечивает надежное соединение и используется прикладными приложениями, передающими большие объемы данных за одну операцию, а также приложениями, которым необходимо подтверждение приема данных. TCP обеспечивает сегментацию сообщений с выявлением и устранением ошибок, освобождая приложения от выполнения этих действий.

2) User Datagram Protocol (UDP) обеспечивает передачу данных и не гарантирует доставку пакетов. Приложения, использующие UDP, обычно передают небольшие объемы данных за одну операцию и должны самостоятельно выполнять контроль ошибок и их исправление.

Модули TCP, UDP и драйвер Ethernet являются мультиплексорами $n \times 1$ (переключают несколько входов на один выход), а также демультиплексорами $1 \times n$.

Верхний уровень или уровень приложения открывает доступ к сетевым компонентам. На этом уровне функционирует множество приложений и протоколов TCP/IP, включая FTP, Telnet, DNS (Domain Name Service) и SNMP (Simple Network Management Protocol). Уровень приложения включает программные интерфейсы API (Application Programming Interface), позволяющие несетевым приложениям взаимодействовать через сеть (Windows Sockets, NetBIOS).

Архитектура TCP/IP предназначена для объединенной сети, состоящей из соединенных шлюзами разнородных пакетных подсетей и машин. Если необходимо передать пакет между машинами, входящими в разные подсети, то машина-отправитель посылает пакет в соответствующий шлюз, а оттуда пакет направляется по определенному маршруту через систему шлюзов и подсетей, пока не достигнет шлюза, подключенного к той же подсети, что и машина-получатель. Проблема доставки пакетов в такой системе решается путем реализации во всех узлах и шлюзах межсетевого

протокола IP. По существу, межсетевой уровень является базовым элементом во всей архитектуре протоколов, обеспечивая возможность стандартизации протоколов верхних уровней.

В любой корректно созданной локальной сети при условии ее подключения к Internet, протокол TCP/IP позволяет работать с любым компьютером в мире. Для успешной передачи данных в заголовке протокола TCP/IP указываются: IP-адрес отправителя, IP-адрес получателя, номер порта (номер прикладной программы, для которой этот пакет предназначен).

IP-адрес

Одна из основных задач, решаемых при объединении нескольких компьютеров, это получение одного или нескольких официальных сетевых номеров, служащих для идентификации компьютера. При использовании протокола TCP/IP как сами сети, так и конечные устройства (персональные компьютеры, коммуникационные серверы, маршрутизаторы и т.д.) имеют свой уникальный адрес или IP-адрес – 32-битное число (записываемое в десятичном или шестнадцатиричном формате), состоящее из адреса сети, в которой располагается устройство, и адреса хоста:

$$\frac{X}{0\text{-}223} \cdot \frac{X}{0\text{-}255} \cdot \frac{X}{0\text{-}255} \cdot \frac{X}{0\text{-}255}$$

При этом если некоторое устройство, например маршрутизатор, имеет несколько физических интерфейсов, то каждый из них должен иметь уникальный IP-адрес. Данная схема адресации описывает не само устройство в сети, а определенное соединение данного устройства с сетью. Иногда поле номера сети в адресе называют сетевым префиксом и в каждой определенной сети все хосты имеют один и тот же сетевой префикс и уникальные номера хостов.

Уникальность всех существующих IP-адресов обеспечивается их централизованным назначением организацией Internet Network Information Center (InterNIC), европейское отделение которой расположено в Амстердаме. До апреля 1993 года назначение IP-адресов осуществлялось организацией Network Information Center (NIC), которая в настоящее время выполняет запросы только для сетей Defense Data Network (DDN), используемых в военных целях. InterNIC назначает только сетевую часть адреса или сетевой префикс, оставляя ответственность за назначения номеров хостов в этой сети организации, запросившей данный адрес.

Для проверки работоспособности компьютера в сети можно использовать команду **ping** (**ping** ip-адрес_компьютера), отправляющую указанному компьютеру небольшой пакет данных; возврат им пакета позволяет судить о связи между двумя хостами. Пример команды:

ping 192.168.11.1

В случае отсутствия связи между компьютерами показать и проверить маршрут прохождения информации между ними можно командой **tracert** (traceroute) со следующим синтаксисом: **tracert** ip-адрес компьютера.

Классы адресов и их маски

Компьютерная сеть представляет собой ряд из 2^n идущих подряд IP-адресов, самый младший из которых резервируется и называется IP-адресом сети. Адрес сети используется в случае, если требуется указать всю сеть целиком. Каждый компьютер в сети TCP/IP имеет адреса трех уровней:

1) физический адрес узла, определяемый технологией построения сети (для узлов, работающих в локальных сетях Ethernet, – это MAC-адрес сетевой платы или порта маршрутизатора, назначаемый производителем оборудования, и состоящий из шести байтов: старшие три байта – идентификатор фирмы-производителя, младшие три байта уникальны и назначаются самим производителем);

2) четырехбайтный IP-адрес, используемый на сетевом уровне эталонной модели OSI;

3) символьный идентификатор – имя, которое может произвольно назначаться администратором и служить для упрощения взаимодействия с удаленным хостом.

Маска сети – это число, двоичная запись которого содержит единицы в разрядах, интерпретируемых как номер сети, и фактически задает число адресов в ней (например, 8бит – 256 адресов, 6 бит – 64 адреса). Маска записывается, например, в виде:

255.255.255.192 - маска на 64 адреса
255.255.255.0 - маска на 256 адресов
255.255.0.0 - маска на 64Kb адресов

Broadcast-адрес сети – самый старший адрес в сети, резервируемый для передачи сообщений типа "всем-всем-всем".

Для обеспечения гибкости в назначении адресов компьютерным сетям разработчики определили, что адресное пространство протокола IP должно быть разделено на классы (табл. 3), каждый из которых фиксирует границу между сетевым префиксом и номером хоста в разных точках 32-разрядного адреса. При этом адреса 0, 127, 255 являются специальными и обычно не используются.

Таблица 3

Классы сетей и адреса

Класс	Диапазон значений первого октета (адрес сети)	Маска сети	Возможное количество сетей	Вид IP-адресов	Возможное количество узлов
A ("огромные" сети)	1-126	255.0.0.0	126	125.*.*.*	16777214
B ("средние" сети)	128.0 - 191.255	255.255.0.0	16382	136.12.*.*	65534
C ("малые" сети)	192.0.0-255.254.255	255.255.255.0	2097150	195.136.12.*	254
D ("multicast-сети")	224-239		-		268435456
E ("экспериментальные")	240-250		-		134217728

Адреса класса A предназначены для идентификации устройств в

крупных сетях и каждый из них имеет 8-разрядный префикс сети, старший бит которого равен "0", а следующие семь бит используются для определения номера сети и так же обозначаются, как "/8". Для обозначения номера хоста служат оставшиеся 24 бита. В настоящий момент все адреса класса А уже выделены, так что получить его вряд ли возможно. Так как адресный блок класса А может содержать максимум 2^{31} индивидуальных адреса, а в протоколе IP может поддерживаться максимум 2^{32} адреса, то адресное пространство класса А занимает 50% общего адресного пространства протокола IP.

Адреса класса В предназначены для сетей среднего размера, например, в крупной организации. Сети класса В имеют 16-разрядный префикс сети, в котором два старших бита равны "10", а следующие 14 бит используются для определения номера сети. Для задания номера хоста служат оставшиеся 16 бит. Сети класса В так же обозначаются, как "/16". Так как весь адресный блок класса В содержит максимум 2^{30} индивидуальных адресов, он занимает 25% общего адресного пространства в протоколе IP.

Адреса класса С предназначены для сетей с небольшим числом компьютеров. Каждая сеть класса С имеет 24-разрядный префикс сети, в котором три старших бита равны "110", а следующие 21 бит используются для определения номера сети. Для определения номера хоста служат оставшиеся 8 бит. Сети класса С так же обозначаются, как "/24". Весь адресный блок класса С может содержать максимум 2^{29} (536 870 912) индивидуальных адреса и занимает 12,5% общего адресного пространства в протоколе IP.

Помимо этих трех наиболее популярных классов адресов существует еще два дополнительных класса – D и E. В классе D старшие четыре бита равны "1110"; этот класс используется для поддержки многоадресной передачи данных. В классе E старшие четыре бита равны "1111", и этот класс зарезервирован для экспериментального использования.

Если сеть используется только внутри организации, и машины не будут иметь непосредственного выхода в интернет, то адреса можно назначать произвольно.

The Internet Assigned Numbers Authority зарезервировал три блока IP адресов для использования во внутренних сетях:

10.0.0.0 - 10.255.255.255	класс А
172.16.0.0 - 172.31.255.255	класс В
192.168.0.0 - 192.168.255.255	класс С

Роутеры большинства ISP эти адреса не маршрутизируют, что повышает безопасность последних.

Помимо возможности направленной передачи информации определенному хосту отправитель может передать данные всем хостам в указанной сети. В протоколе IP существует два типа широковещания: направленное и ограниченное. Направленное широковещание позволяет хосту в удаленной сети передать одну датаграмму, причем она может проходить через маршрутизаторы в распределенной сети и будет доставлена всем хостам в адресованной сети, но не в промежуточных сетях. При направленном ши-

роковещании необходимо знание номера целевой сети.

Вторая форма широковещания обеспечивает передачу данных в пределах сети отправителя, независимо от указанного IP-адреса. Датаграмма с ограниченным широковещательным адресом никогда не будет проходить через маршрутизаторы и будет рассылаться в пределах данной сети.

Архитектура локальной сети

Организация подсетей решает проблему роста таблиц маршрутизации за счет того, что структура подсетей корпоративной сети невидима за пределами организации. Маршруты из Internet до любой подсети данного IP-адреса одинаковы вне зависимости от того, в какой подсети расположен получатель. Это стало возможным благодаря тому, что все подсети данного номера сети используют один сетевой префикс, но разные номера подсетей. Маршрутизаторы в частной сети должны различать отдельные подсети, но в Internet все данные подсети определены единственной записью в таблицах маршрутизации. Это позволяет администратору частной сети вносить любые изменения в её логическую структуру без влияния на размер таблиц маршрутизации. Формирование подсетей позволяет также решить вторую проблему, связанную с выделением организации нового сетевого номера или номеров при ее росте.

Организация подсетей даёт следующие преимущества:

- ограничение размера глобальных таблиц маршрутизации в сети Internet;
- создавать дополнительные подсети без получения новых номеров сетей.

Первым шагом в процессе планирования является определение максимального количества требуемых подсетей с учётом возможного увеличения их числа в будущем. Данное число округляется до ближайшей степени двойки. На втором шаге проверяется факт существования достаточного количества адресов хостов в наибольшей подсети организации. В заключении убеждаются, что выделенный организации класс адресов предоставляет необходимое для определения подсетей количество бит.

В ОС Unix конфигурация основных сетевых параметров компьютера осуществляется следующим образом: имя хоста, IP-адрес и параметры сети, адрес DNS сервера.

HP/UX: /etc/set_parms initial
Linux Slackware: net_config
Solaris: /etc/???

Если необходимо настроить обмен сетевыми пакетами для машин с адресами 198.8.2.* через адрес 198.8.2.1, то используется команда:

route add net 198.8.2.0 198.8.2.1 netmask 255.255.255.0 0

Сетевые пакеты для IP-адресов, которые не лежат в рассматриваемой локальной сети, можно переправлять на машину, которая осуществит их дальнейшую передачу (например, с адресом 198.8.2.107):

route add default 198.8.2.107 1

Последний аргумент команды route – метрика. Результаты конфигурации можно посмотреть с помощью следующих команд:

```
netstat –rn          # распечатать таблицу маршрутизации
ping -s fedfond      # "прозвонить" узлы сети
ping -s fedfond-gate
ping -s 198.8.2.107
netstat -i           # статистика о работе сетевых интерфейсов
```

Узнать hardware адрес ethernet-карты можно командами ifconfig (Linux), lanscan (HP-UX):

```
ping [-s] 123.456.789.255
arp -a
```

Иногда удобнее называть машины по именам, а не числами. В маленьких сетях информация о соответствии имен IP-адресам хранится в файлах "hosts" на каждом узле, в больших – на сервере и доступна по сети:

```
223.1.2.1     alpha
223.1.2.2     beta
223.1.2.3     gamma
223.1.3.2     epsilon
223.1.4.2     iota
```

В большинстве случаев файлы "hosts" могут быть одинаковы на всех узлах. Если узел имеет несколько IP-адресов, то при получении пакета данных он опознает любой из своих IP-адресов. Компьютерные сети также могут иметь имена и в этом случае файл "networks" содержит их номера и соответствующие им имена сетей:

```
223.1.2       development
223.1.3       accounting
223.1.4       factory
```

Файлы hosts и networks используются командами администрирования и прикладными программами и облегчают использование сети internet.

Выбор сетевого интерфейса для отправки пакета данных осуществляет по IP-таблице маршрутов. Ключом поиска служит номер сети, выделенный из IP-адреса места назначения пакета данных.

Таблица маршрутов содержит по одной строке для каждого маршрута и состоит из следующих столбцов: номер сети, флаг прямой или косвенной маршрутизации, IP-адрес шлюза, номер сетевого интерфейса. Эта таблица используется IP-модулем при обработке каждого отправляемого пакета данных и в большинстве систем может быть изменена с помощью команды "route". Содержание таблицы маршрутов определяется менеджером сети, поскольку именно он присваивает машинам IP-адреса.

Простейший способ маршрутизации состоит в установке с помощью специальных команд маршрутов на этапе запуска системы. Этот метод можно применять в относительно маленьких IP-сетях.

На практике большинство машин автоматически формирует таблицы маршрутов: например, UNIX добавляет записи об IP-сетях, к которым есть непосредственный доступ. Стартовый файл может содержать команды:

ifconfig ie0 128.6.4.4 netmask 255.255.255.0

Также в стартовом файле могут быть команды, определяющие маршруты доступа к другим IP-сетям:

route add 128.6.2.0 128.6.4.1 1

route add 128.6.6.0 128.6.5.35 0

Первый адрес в командах является IP-адресом сети, второй адрес – шлюз, используемый для доступа к данной IP-сети, а третий параметр является метрикой. Метрика характеризует удаленность описываемой IP-сети и показывает количество шлюзов на пути между двумя IP-сетями. Маршруты с метрикой 1 и более определяют первый шлюз на пути к IP-сети; метрика 0 показывает, что никакой шлюз не нужен.

Можно определить маршрут по умолчанию, используемый, если IP-адрес места назначения не встречается в таблице маршрутов явно. Обычно это IP-адрес шлюза, который имеет достаточно информации для маршрутизации IP-пакетов со всеми возможными адресами назначения.

Если IP-сеть имеет всего один шлюз, то нужно установить единственную запись в таблице маршрутов, указав этот шлюз как маршрут по умолчанию.

Адресные протоколы

Адресный протокол ARP (address resolution protocol) – это протокол низкого уровня для отображения IP-адресов в Ethernet-адреса, поддерживаемый на уровне ядра и/или драйвера сетевой платы и работающий по принципу broadcast. Отображение выполняется только для отправляемых IP-пакетов, поскольку только в момент отправки создаются заголовки IP и Ethernet:

arp -a # распечатать известные нам IP-адреса и их eth-адреса

Преобразование адресов выполняется путем поиска в ARP-таблице, которая хранится в памяти компьютера и содержит IP- и Ethernet-адреса для каждого узла сети. Если требуется преобразовать IP-адрес в Ethernet-адрес, то ищется запись с соответствующим IP-адресом.

Таблица 4

Пример ARP-таблицы

IP-адрес	Ethernet-адрес
223.1.2.1	08:00:39:00:2F:C3
223.1.2.3	08:00:5A:21:A7:22
223.1.2.4	08:00:10:99:AC:54

ARP-таблица необходима поскольку IP-адреса и Ethernet-адреса выбираются независимо друг от друга. IP-адрес выбирает менеджер сети с учетом положения машины в сети internet. Если машину перемещают в другую часть сети internet, то ее IP-адрес меняется. Ethernet-адрес выбирает производитель сетевого интерфейсного оборудования (сетевой платы) из выделенного для него по лицензии адресного пространства. ARP-таблица заполняется автоматически модулем ARP, по мере необходимости, и новые записи в ней появляются спустя миллисекунды после запроса.

При отсутствии в сети машины с искомым IP-адресом, ARP-ответа не будет и запись в ARP-таблице будет отсутствовать, а IP-протокол будет уничтожать пакеты данных, направляемые по этому адресу. Протоколы верхнего уровня не могут отличить повреждение сети Ethernet от отсутствия машины с искомым адресом. Некоторые реализации IP и ARP не ставят в очередь IP-пакеты на время ожидания ARP-ответов. Вместо этого IP-пакет просто уничтожается, а его восстановление возлагается на модуль TCP или прикладной процесс, работающий через UDP. Такое восстановление выполняется с помощью таймаутов и повторных передач, которые проходят успешно, так как первая попытка уже вызвала заполнение ARP-таблицы. Каждая машина имеет отдельную ARP-таблицу для каждого своего сетевого интерфейса.

Обратный протокол преобразования адресов RARP (reverse address resolution protocol) осуществляет преобразование аппаратного (MAC) адреса в IP-адрес. Протокол RARP предполагает наличие специального сервера, обслуживающего RARP-запросы и хранящего базу данных о соответствии аппаратных адресов протокольным, и позволяет передать IP-адрес от редко используемого хоста другому узлу; измененная информация сохраняется сетевыми станциями в их ARP-таблицах.

Для обмена информацией о маршрутизации используются: внутренний протокол роутинга RIP (routing information protocol); BGP (border gateway protocol) и EGP (external gateway protocol), осуществляющие роутинг между автономными системами; протокол обмена управляющими сообщениями ICMP (internet control message protocol) передает сообщения об ошибках в TCP/IP (например "port unreachable"), используется командами ping, traceroute, может передавать сообщение о нерациональном маршруте типа "redirect".

Для обмена информацией о символических именах хостов, пользователей, группах пользователей и пр. используются протоколы DNS и NIS/YP.

Протокол DNS (domain name system) является служебным протоколом прикладного уровня и позволяет использовать символические имена хостов помимо цифровых IP-адресов, а также организовывать структуру дерева наименований доменов. На DNS-сервере хранится база имен хостов в домене, а на остальных хостах для определения адреса по имени используются функции библиотеки "resolver" – gethostbyname, gethostbyaddr, которые обращаются по сети к ближайшему серверу DNS, и считывают с него IP-адрес машины по ее имени. Домены верхнего уровня назначаются для каждой страны, а также на организационной основе, а их имена должны следовать международному стандарту ISO 3166.

Протокол NIS/YP (network information system) позволяет хранить на одном NIS-сервере информацию, единую для всей локальной сети: имена хостов, имена и права пользователей, групп пользователей, название их домашних каталогов, символические имена портов и т.д. NIS/YP содержит помимо имен хостов несколько других информационных баз, но зато поддерживается только в рамках сети одной организации.

Протоколы прикладного уровня

Протоколы TCP и UDP предоставляют разные услуги прикладным процессам, большинство которых пользуются только одним из них. Если нужна гарантированная и эффективная доставка по длинному и ненадежному каналу передачи данных, то предпочтение отдается TCP. Он освобождает прикладные процессы от необходимости использовать таймауты и повторные передачи для обеспечения надежности. Типичными прикладными процессами, использующими TCP, являются FTP и TELNET, а также система X-Window, rcp и другие "r-команды". Недостатком TCP являются высокие требования к производительности процессора и большой пропускной способности сети. Внутренняя структура модуля TCP гораздо сложнее структуры модуля UDP.

Если нужна эффективная доставка датаграмм в быстрых сетях с короткими соединениями, то предпочтительно использование UDP. К заголовку IP-пакета он добавляет два поля, одно из которых (поле "порт") обеспечивает мультиплексирование информации между разными прикладными процессами, а другое поле ("контрольная сумма") позволяет поддерживать целостность данных. UDP используются системами NFS и SNMP.

Если решаемые задачи не относятся к рассмотренным категориям, то предпочтений в выборе транспортного протокола нет. Однако прикладные программы могут устранять недостатки выбранного протокола. В сетях с TCP/IP доступны прикладные протоколы TELNET, FTP, X-Window, SNMP.

1. Протокол TELNET позволяет обслуживающей машине рассматривать все удаленные терминалы как стандартные "сетевые виртуальные терминалы" строчного типа, работающие в коде ASCII, а также обеспечивает возможность согласования более сложных функций (например, локальный или удаленный эхо-контроль, страничный режим, высота и ширина экрана и т.д.). Работа с TELNET довольно проста: после ввода команды «telnet delta» пользователь получает приглашение на вход в машину delta. Протокол TELNET существует давно и широко распространен, и существует множество его реализаций для разных ОС.

2. Протокол передачи файлов FTP (file transfer protocol) является одним из старейших протоколов семейства TCP/IP и распространен также широко как TELNET. Пользователь FTP может посмотреть каталог удаленной машины, перейти из одного каталога в другой, а также скопировать один или несколько файлов. Существует множество реализаций FTP для различных ОС, которые хорошо взаимодействуют между собой.

3. Протокол передачи почты SMTP (simple mail transfer protocol) поддерживает передачу сообщений между произвольными узлами сети internet. Имея механизмы промежуточного хранения почты и механизмы повышения надежности доставки, протокол SMTP допускает использование различных транспортных служб и может работать даже в сетях, не использующих протоколы TCP/IP. Протокол SMTP обеспечивает как группирование сообщений в адрес одного получателя, так и создание нескольких копий сообщения для передачи в разные адреса.

4. Для работы с удаленными машинами предназначена серия r-команд (от remote – удаленный). Например, команда **rcp** является аналогом команды **cp** и предназначена для копирования файлов между машинами. Для передачи файла на узел delta достаточно ввести **rcp file.c delta**. Вход в удаленную систему осуществляется с помощью команды **rlogin** (rlogin delta) Команды r-серии используются главным образом в ОС UNIX и избавляют пользователя от необходимости набирать пароли при подключении к удаленной системе.

5. Сетевая файловая система NFS (network file system) была разработана компанией Sun Microsystems Inc. NFS использует транспортные услуги UDP и позволяет монтировать в единое целое файловые системы нескольких машин с ОС UNIX. Бездисковые рабочие станции получают доступ к дискам файл-сервера так, как если бы это были их локальные диски. NFS значительно увеличивает нагрузку на сеть. Однако если пропускная способность сети позволяет NFS нормально работать, то пользователи получают большие преимущества.

6. Протокол управления сетью SNMP (simple network management protocol) работает на базе UDP и предназначен для использования сетевыми управляющими станциями, позволяя им собирать информацию о сети internet. Протокол определяет формат данных, а их обработка и интерпретация остаются на усмотрение управляющих станций или менеджера сети.

Взаимодействие между прикладными процессами и модулями UDP и TCP осуществляется через порты, имеющие номера от 0 до 65536. Например, FTP-сервер использует конкретный TCP-порт, поэтому другие приложения могут связаться с ним. Прикладной процесс, предоставляющий услуги другим прикладным процессам (сервер), ожидает поступления сообщений в порт, специально выделенный для этих услуг. Сообщения должны содержать запросы на предоставление услуг и отправляются процессами-клиентами. Данные, отправляемые прикладным процессом через модуль UDP, достигают места назначения как единое целое. Размер каждого записанного сообщения будет совпадать с размером прочитанного. Протокол UDP сохраняет границы сообщений, определяемые прикладным процессом, не объединяя несколько сообщений в одно, и не делит одно сообщение на части.

Номера портов для приложений клиентов динамически назначаются операционной системой при обработке запроса на обслуживание. Для отдельных приложений группой Internet Assigned Numbers Authority (IANA) выделяются общеизвестные номера портов в интервале от 1 до 1024 (их список приведен в RFC 1700). Например, сервер SNMP всегда ожидает поступлений сообщений в порт 161 и при получении услуги запрос посылается в UDP-порт 161. В каждом узле может быть только один сервер SNMP, так как существует только один UDP-порт 161. Сервер TELNET использует порт номер 23 и услуги от сервера можно получать, установив его соединение с TCP-портом 23. Узнать номера портов можно, просмотрев файл systemrootSystem32DriversEtcServices.

КОМПЬЮТЕРНАЯ ГРАФИКА

Компьютерная графика является одной из форм представления информации и представляет собой создание, хранение и обработку моделей объектов и их изображений. Существует два вида компьютерной графики: векторная, растровая, а также полутоновая, основное отличие которых заключается в принципе хранения данных [2, 14].

Векторная и растровая графика

Векторная графика (или геометрическое моделирование) – это использование геометрических примитивов, таких как точки, линии, сплайны и многоугольники, описываемых с помощью математических выражений. По своей сути любое изображение можно разложить на множество простых объектов (графические примитивы, контуры, заливка). Например, для описания окружности в векторном формате, необходимы радиус, координаты центра окружности, цвет и толщина контура, цвет заполнения.

Типичные примитивные объекты, используемые в векторной графике, это линии, многоугольники, окружности и эллипсы, кривые Безье, сплайны, текст (в компьютерных шрифтах, таких как TrueType, каждая буква создаётся из кривых Безье).

Векторная форма представления информации имеет ряд достоинств, определяющих высокое качество изображения:

- параметры объектов хранятся в файле и могут быть изменены, а перемещение, масштабирование, вращение, заполнение и т.д. не ухудшат качества изображения, поскольку описывающие его формулы постоянны, а меняется только коэффициент пропорциональности;
- размер файла не зависит от величины объекта;
- при увеличении или уменьшении объектов толщина линий может оставаться постоянной.

Для векторной графики характерны следующие недостатки:

- не каждый объект можно представить в векторном виде, а объем памяти и времени отображения зависит от числа объектов и их сложности. Размер векторного файла сложной геометрической фигуры может быть значительно больше, чем его "растровый" аналог из-за сложности описывающих изображение формул;
- перевод векторной графики в растр достаточно прост, но обратное преобразование чаще не возможно (трассировка растра обычно не обеспечивает высокого качества векторного рисунка);
- векторная графика предпочтительна для изображений, не имеющих большого числа полутонов и оттенков (например, для оформления текстов, создания логотипов и т.д.).

Векторные графические редакторы позволяют вращать, перемещать, отражать, растягивать, выполнять основные аффинные преобразования над объектами и комбинировать примитивы в более сложные объекты. Векторная графика идеальна для простых или составных рисунков, которые должны быть аппаратно-независимыми. Модель векторной графики ис-

пользуют такие редакторы как PostScript и PDF. К векторным графическим редактором относятся Adobe Illustrator, CorelDRAW, Macromedia FreeHand, Xara Xtreme, Strokes Maker, MathCAD, Matlab.

Основными форматами для векторного представления изображений являются SVG, WMF и CDR.

Формат SVG (scalable vector graphics) это язык разметки масштабируемой векторной графики, предназначенный для описания двухмерной векторной и смешанной векторно-растровой графики в формате XML. Достоинствами SVG-графики являются:

1) текстовый формат файлов, что позволяет читать их и редактировать при помощи обычных текстовых редакторов, а также просматривать код файла. Размер SVG файлов обычно меньше, чем для сравнимых по качеству изображений в форматах JPEG или GIF;

2) масштабируемость любой части изображения без потери качества и возможность применения фильтров – специальных модификаторов для создания эффектов, подобных используемым при обработке растровых изображений (размытие, сложные системы трансформации и др.);

3) возможно использование растровой графики в форматах PNG, GIF или JPG;

4) текст не является изображением, поэтому к нему применимы все соответствующие операции;

5) поддерживает анимированную и интерактивную графику.

WMF (Windows metafile) это универсальный формат векторных графических файлов для Windows приложений, используемый для хранения коллекции графических изображений Microsoft Clip Gallery. Формат разработан Microsoft и является неотъемлемой частью Windows, так как сохраняет последовательность аппаратно-независимых функций GDI (Graphical Device Interface), непосредственно выводящих изображение в заданный контекст графического устройства (на экран, на принтер и т.п.). На платформе Macintosh аналогичную роль играет формат PICT.

Растровое цифровое изображение – это файл данных или структура, представляющая прямоугольную сетку пикселов или точек цветов на отображающих устройствах. При использовании растровой графики важным элементом является размер полотна, тип цветопередачи (например, RGB), количество используемых цветов. Т.к. пиксели имеют очень маленький размер, то при высокой разрешающей способности человеческий глаз не видит структуру изображения. При увеличении растрового изображения компьютер увеличивает размер матрицы и добавляет пиксели, усредняя значения исходных точек. При уменьшении изображения происходит обратный процесс – компьютер удаляет лишние пиксели. Это определяет главный недостаток растровой графики – качество изображения зависит от его размеров.

Растровую графику следует применять для изображений с фотографическим качеством, на которых присутствует множество цветовых переходов. Размер файла, хранящего растровое изображение, зависит от двух

факторов: размера изображения и глубины цвета изображения (чем больше цветов, тем больше размер файла).

Разновидностью растровой графики является пиксельная графика: рисование картинки по элементарным неделимым точкам, которые могут принимать только один цвет. Такая форма графики позволяет минимизировать размер файла и обычно применяется для изготовления кнопок, небольших баннеров и т.п.

Достоинства растровой графики:

- позволяет создать (воспроизвести) практически любой рисунок, в том числе точно передать эффект перехода от одного цвета к другому, что практически невозможно в векторной графике (файл размером 1 МБ в формате BMP будет иметь размер 200 МБ в векторном формате);
- широкая распространённость.

Недостатками растровой графики являются большой размер файла даже несмотря на сжатие и потери качества при увеличении изображения.

К растровым графическим редакторам относятся Adobe Photoshop, GIMP, Microsoft Paint.

К растровой графике относятся файлы в форматах BMP, TIFF, GIF, PNG, JPEG.

Растровое изображение Windows или BMP (Windows bitmap) является собственным форматом графического редактора Microsoft Paint. BMP-файлы могут представлять до 16 млн цветов и иметь большой объем, хотя в случае пиктограммы размер BMP-файла обычно оказывается меньше, чем размер соответствующего GIF- или JPEG-файла. Допускает применение алгоритма последовательного сжатия без потерь RLE, однако не все графические программы распознают сжатые BMP-файлы. Формат BMP используется для иллюстраций в справочных системах, фоновых изображений.

Формат TIFF (tagged image file format) предназначен для хранения изображений с большой глубиной цвета, используется при сканировании, распознавании текста, в полиграфии.

Формат для обмена изображениями GIF (graphics interchange format) способен хранить сжатые без потерь изображения в формате до 256 цветов, и предназначен, в основном, для чертежей и графиков. Разработанный в 1987г для передачи растровых изображений по сетям, он позволяет хорошо сжимать файлы, в которых много однородных заливок и повторяющихся участков (логотипы, надписи, схемы). Изображение в формате GIF хранится построчно и позволяет создавать анимационные эффекты.

Формат PNG (portable network graphics) использует для хранения графической информации со сжатием без потерь (в отличие от JPEG) и спроектирован для улучшения и замены устаревшего и более простого формата GIF, а также в некоторой степени для замены значительно более сложного формата TIFF.

Аппаратно-независимый алгоритм сжатия JPEG основан не на поиске одинаковых элементов, а на вычислении разницы между пикселами. Кодирование данных происходит в несколько этапов. Сначала графические дан-

данные конвертируются в цветовое пространство, затем отбрасывается 1/2-3/4 информации о цвете (в зависимости от алгоритма) и анализируются блоки 8x8 пикселов. Для каждого блока формируется набор чисел, первые из которых представляют цвет блока в целом, а последующие – отражают детали. Спектр деталей базируется на зрительном восприятии человека, поэтому крупные детали более заметны. Затем в зависимости от выбранного уровня качества, отбрасывается определенная часть чисел, представляющих детали изображения и проводится кодирование методом Хафмана для более эффективного сжатия конечных данных. Восстановление данных происходит в обратном порядке. Таким образом, чем выше уровень компрессии, тем больше данных отбрасывается и тем ниже качество.

JPEG лучше сжимает растровые фотографии, чем логотипы или схемы, т.к. в них больше полутоновых переходов, а при однотонной заливке появляются нежелательные помехи. Лучше сжимаются и с меньшими потерями большие изображения для web или с высоким разрешением (от 200 dpi), чем с низким (72-150 dpi), т.к. в каждом блоке 8x8 пикселов переходы получаются более мягкие, за счет того, что самих блоков больше. Использование JPEG-сжатия нежелательно для изображений, где важны нюансы цветопередачи. В формате JPEG следует сохранять только конечный вариант работы, т.к. каждое повторное сохранение приводит к потерям данных.

Методы обработки изображений

Цифровое монохромное изображение представляет собой матрицу A целых неотрицательных чисел размера $M \times N$, где каждый элемент – это точка на плоскости:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix},$$

где $a_{i,j}$ – элемент изображения; (i, j) – координаты пикселя на плоскости.

Количество оттенков серого (градаций яркости) L определяется разрядностью k цифрового изображения, т.е. количеством бит, отводимых для хранения информации о яркости каждого пикселя:

$$L = 2^k.$$

Яркостным или полутоновым разрешением называется мельчайшее различимое изменение яркости (т.е. число градаций серого L). Значения элементов матрицы изображения могут быть расположены в интервале $[0, L-1]$. Значение 0 соответствует черному цвету, а значение $L-1$ – белому.

Интервал значений яркости $[L_1, L_2]$ называют динамическим диапазоном изображения. Изображения, диапазон яркости которых занимает значительную часть всего диапазона уровней серого (от 0 до $L-1$), называются изображениями с большим динамическим диапазоном и имеют высокий контраст. Изображения с малым динамическим диапазоном обычно выглядят тусклыми и размытыми.

Пространственное разрешение определяется двумя факторами: количеством пикселов в изображении (т.е. $M \times N$) и размером мельчайших различимых деталей на изображении.

Наиболее распространённым способом обработки изображений является фильтрация, представляющая собой операцию, позволяющую по некоторым правилам получить изображение того же размера, что и исходное:

$$g(x, y) = T[f(x, y)]$$

где $f(x, y)$ – входное изображение; $g(x, y)$ – обработанное изображение; T – оператор над f, определенный в некоторой окрестности точки (x, y).

Обычно интенсивность (цвет) каждого пикселя результирующего изображения обусловлена интенсивностями (цветами) пикселов, расположенных в некоторой его окрестности в исходном изображении. Поскольку результат обработки каждого элемента изображения зависит только от яркости этого же элемента, методы данной категории относят к процедурам поэлементной обработки.

Увеличение размеров окрестности приводит к большей гибкости. Один из основных подходов в такой постановке базируется на использовании так называемых масок (фильтров, окон), представляющих собой небольшой (например, 3×3 элемента) двумерный массив, значение коэффициентов которого определяет вид процесса обработки (например, повышение резкости изображения) [13].

Линейные и нелинейные фильтры

Линейная фильтрация зашумленного изображения f, имеющего размеры $M \times N$, с помощью вещественнозначной функции h, заданной на растре (ядра фильтра размерами $m \times n$), производится при помощи операции дискретной свертки:

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} h(s,t) f(x+s, y+t),$$

где $a = (m\text{-}1)/2$; $b = (n\text{–}1)/2$; h – маска; f – исходное изображение; g – обработанное изображение.

Фильтрация основана на перемещении маски фильтра от точки к точке изображения, в каждой из которых отклик вычисляется как сумма произведений коэффициентов фильтра на соответствующие значения пикселов изображения.

Сглаживающие фильтры снижают локальную контрастность изображения, размывая его, и применяются для шумоподавления. Увеличение окна фильтрации приведет к снижению усредненной интенсивности шума и размытию деталей изображения (например, образом белой точки на черном фоне будет равномерно серый квадрат). Чаще всего используются усредняющие (сглаживающие) маски:

$$h = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \; h = \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Гауссовский сглаживающий фильтр дает эффективное шумоподавление и

представляет собой свертку изображения по строкам и по столбцам с функцией:

$$F_{Gauss}(i,j)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right).$$

Коэффициент пропорциональности и степень размытия, определяются параметром σ. Образом точки при гауссовой фильтрации будет симметричное размытое пятно, с убыванием яркости от середины к краям, поэтому вблизи границ такой фильтр применять нельзя.

Контрастоповышающие фильтры повышают резкость изображения за счет подчеркивания разницы между интенсивностями соседних пикселов путём удаления этих интенсивностей друг от друга. Эффект будет тем сильней, чем больше значение центрального члена ядра. Недостатком такой фильтрации являются заметные светлые и менее заметные темные ореолы вокруг границ. Чаще используемые маски:

$$h_{contr1}=\begin{bmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{bmatrix};\ h_{contr2}=\begin{bmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{bmatrix}.$$

Разностные фильтры – это линейные фильтры, задаваемые дискретными аппроксимациями дифференциальных операторов (по методу конечных разностей), используемые, как правило, в задачах поиска границ на изображении. Определение границ по *x*-координате $\frac{\partial}{\partial x}$ для непрерывных функций получают взятием производной или для дискретных изображений использованием линейных фильтров Прюита и Собеля:

$$h_{prewitt}=\begin{bmatrix}-1&0&1\\-1&0&0\\-1&0&1\end{bmatrix},\ h_{sobel}=\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}.$$

Фильтры для определения границ по *y*-координате $\frac{\partial}{\partial y}$ получают путем транспонирования матриц. В результате применения разностных фильтров получается изображение со средним значением пикселя близким к нулю (сумма элементов ядра равна нулю). Определение границ любой ориентации производится с помощью оператора Лапласа $\nabla^2 f=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}$, а общий алгоритм улучшения изображения сводится к следующему:

$$g(x,y)=\begin{cases}f(x,y)-\nabla^2 f(x,y), & \textit{если}\quad h(0,0)<0\\ f(x,y)+\nabla^2 f(x,y), & \textit{если}\quad h(0,0)\geq 0\end{cases},$$

где $h(0, 0)$ – значение центрального элемента маски.

$$h=\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix};\ h=\begin{bmatrix}1&1&1\\1&-8&1\\1&1&1\end{bmatrix};\ h=\begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix};\ h=\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}.$$

Метод нерезкого маскирования заключается в вычитании из изображения его расфокусированной копии и ведет свое начало из фотографии. Основным результатом такой фильтрации становится подчеркивание кон-

туров: светлые участки на границе становятся еще более светлыми, а темные – еще более темными.

Нелинейным называют фильтр, выход которого не является линейным оператором от входного сигнала. Такие фильтры широко используются в технике, электронике, теории управления и обработке сигналов и изображений [4, 7]. Самым простым примером нелинейного фильтра служит пороговая фильтрация, результатом которой является бинарное изображение:

$$B(x, y) = \begin{cases} 1, & \text{если} \quad A(x, y) > \gamma \\ 0, & \text{иначе} \end{cases},$$

где γ – порог фильтрации.

При *ранговой фильтрации* пиксели исходного изображения, соответствующие ненулевым элементам маски фильтра, сортируются в порядке возрастания и центральному элементу маски присваивается значение с заданным порядковым номером (рангом) [4]. Выбор минимального ранга позволяет создавать эффект расфокусировки (эрозии), а максимального – эффект фокусировки (уточнения) обрабатываемого изображения.

Медианная фильтрация это метод низкочастотной фильтрации изображений, позволяющий убрать резкие выбросы («импульсный» шум) и являющийся частным случаем ранговой фильтрации. Для каждого положения окна (маски размером $m \times n$) все значения пикселов сортируются в порядке возрастания, и за новое значение фильтруемой точки принимается значение элемента в середине списка (медиана):

$$B_{median}(x, y) = median\{N(x, y)\}.$$

Медианный фильтр не влияет на наклонные участки и резкие перепады яркости на изображениях. Недостатком метода является относительно слабая эффективность при фильтрации флуктуационного шума и размытие контуров изображения при увеличении размера маски.

Морфологические операторы – фильтры, применяемые для морфологического анализа бинарных изображений, когда они рассматриваются как вид двумерной геометрической фигуры (пиксели, равные 1, считаются лежащими внутри фигуры, а равные 0 – вовне). Морфологический анализ применяется в задачах распознавания образов. Примером служат фильтры минимум и максимум, дающие в результате фильтрации минимальное и максимальное значения пикселов окрестности:

$$B_{\min}(x, y) = \min\{N(x, y)\},$$
$$B_{\max}(x, y) = \max\{N(x, y)\}.$$

Адаптивная (оконная) фильтрация Винера применяется для подавления на изображениях гауссовского белого шума [4]. Метод учитывает статистические особенности изображения в пределах выбранного окна, в частности среднее значение яркости пикселов и ее среднеквадратическое отклонение. Если в выбранном окне дисперсия яркостей большая, то фильтр Винера дает небольшое сглаживание и наоборот. Адаптивный фильтр более избирательный, чем сопоставимый линейный фильтр, сохраняющий высокочастотные компоненты сигнала, но требует большего время вычисления.

Вейвлет-анализ

Новые способы обработки изображений стали возможны с развитием теории вейвлетов, которые позволяют с высокой точностью представлять сложные сигналы и изображения [15]. Вейвлет-преобразование сигнала $s(t)$ состоит в его разложении по базису, сконструированному из обладающей определенными свойствами функции (вейвлета) $\psi_k(t)$, посредством масштабных изменений и переносов:

$$s(t) = \sum_k C_k \psi_k(t).$$

Каждая из функций этого базиса характеризует как определенную пространственную (временную) частоту, так и ее локализацию в физическом пространстве (времени).

Вейвлет-представление сигнала заключается в его разбивке на две составляющие – грубую (аппроксимирующую) и детализирующую – с последующим их уточнением итерационным методом. Каждый шаг такого уточнения соответствует определенному уровню декомпозиции и реставрации сигнала.

В основе непрерывного вейвлет-преобразования лежит использование двух непрерывных и интегрируемых по всей оси t (или x) функций:

– вейвлет-функции $\psi(t)$ с нулевым значением интеграла ($\int_{-\infty}^{+\infty} \psi(t)dt = 0$), определяющей детали сигнала и порождающей детализирующие коэффициенты;

– масштабирующей функции $\varphi(t)$ с единичным значением интеграла ($\int_{-\infty}^{+\infty} \varphi(t)dt = 1$), определяющей грубое приближение сигнала и порождающей коэффициенты аппроксимации.

Аппроксимирующие функции $\varphi(t)$ присущи не всем вейвлетам, а только тем, которые относятся к ортогональным. Детализирующая функция $\psi(t)$ создается на основе базисной функции $\psi_0(t)$, которая, как и $\psi(t)$, определяет тип вейвлета. Базисная функция должна обеспечивать выполнение двух основных операций:

– смещение по оси времени $t - \psi_0(t-b)$ при $b \in \Re$;

– масштабирование $-a^{-1/2}\psi_0\left(\frac{t}{a}\right)$ при $a > 0$ и $a \in \Re^+ - \{0\}$.

где a – параметр, определяющий ширину этого пакета; b – его положение. Следующее выражение задает оба этих свойства вейвлет-функции $\psi(t)$:

$$\psi(t) \equiv \psi(a,b,t) = a^{-1/2}\psi_0\left(\frac{t-b}{a}\right).$$

Для проведения вейвлет-коррекции необходимо определить двухмерные масштабные функции и двухмерные вейвлеты. Для любой масштабирующей функции φ и соответствующего ей вейвлета ψ построим двухмерную аппроксимирующую функцию и три двухмерных вейвлета, используя тензорное произведение:

$$\varphi(x, y) = \varphi(x)\varphi(y),$$
$$\psi^H(x, y) = \psi(x)\varphi(y),$$
$$\psi^V(x, y) = \varphi(x)\psi(y),$$
$$\psi^D(x, y) = \psi(x)\psi(y).$$

Эти вейвлеты измеряют вариации значений функции (изменения яркости изображений) по разным направлениям: ψ^H измеряет вариации вдоль столбцов (связанные, например, с горизонтальными краями объектов), ψ^V – вдоль строк (вертикальные края) и ψ^D – вдоль диагоналей.

В общем случае вейвлет-коррекция включает следующие шаги:

1) выбор уровня аппроксимации j;

2) выбор глубины разложения N и нахождение коэффициентов разложения $\{cA_N, cD_N, cD_{N-1}, ..., cD_1\}$, определяющих приближение функции $f(x)$ в заданном масштабе;

3) восстановление исходной функции $f(x)$ при помощи обратного дискретного вейвлет-преобразования, с использованием полученных коэффициентов разложения:

$$f(x) \approx \sum_{k \in Z} a_{j-N,k} \varphi_{j-N,k}(x) + \sum_{k \in Z} d_{j-N,k} \Psi_{j-N,k}(x) + ... + \sum_{k \in Z} d_{j-1,k} \Psi_{j-1,k}(x).$$

Высокочастотная, или детальная, часть характеризует высокочастотную в вертикальном направлении составляющую изображения. Низкочастотная часть, или приближение, содержит информацию о низких в вертикальном направлении частотах. К обеим частям изображения затем применяется процедура фильтрации по столбцам и прореживание. Это дает на выходе четыре изображения (четыре части исходного изображения).

Вейвлеты применяются в задачах распознавания образов, при обработке и синтезе различных сигналов, при анализе изображений различной природы, для сжатия больших объемов информации.

Оценка визуального качества

Качество изображения определяется большим количеством технических характеристик системы: отношением сигнал/шум и статистическими характеристиками шума, градационными характеристиками, спектральными (цветовыми) характеристиками, интервалами дискретизации и т.д. [5].

Основной метод оценки качества изображения состоит в использовании критерия нормального распределения. Оптимальное, с точки зрения субъективного восприятия, изображение имеет нормальное распределение яркостей его элементов. В этом случае оценка качества изображения проводится по степени отклонения реального распределения яркостей элементов изображения от нормального:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (x_{i,j} - \bar{x}_{i,j})^2}{\sum_{i=1}^{m} \sum_{j=1}^{n} \bar{x}^2_{i,j}}}$$

где m, n – количество строк и столбцов матрицы изображения; $x_{i,j}$, $\bar{x}_{i,j}$ – интенсивность пиксела откорректированного и эталонного изображения, соответственно.

Результаты оценки качества изображения, полученные по данному методу, хорошо коррелируют с субъективной оценкой визуального качества изображения.

Часто для оценки качества изображений используют пиковое соотношение сигнал/шум *PSNR* (peak signal-to-noise ratio):

$$PSNR = 20 \cdot \lg \frac{L_{\max}}{\sqrt{\frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=}^{n} (x_{i,j} - \bar{x}_{i,j})^2}}.$$

Чем ближе отфильтрованное изображение к оригинальному, тем больше значение соотношение *PSNR*, и тем лучше качество работы алгоритма. В то же время, значение *PSNR* более размытого изображения, в котором вместе с шумом были удалены мелкие детали, может быть выше, чем для изображения с более точно подавленным шумом. Визуальные оценки при этом покажут предпочтительность второго изображения

Для оценки качества изображений используют и другие критерии, например, норму Минковского:

$$L_p = \left\{ \frac{1}{MN} \sum_{i=1}^{M} \sum^{N} \left| x_{ij} - \bar{x}_{ij} \right|^p \right\}^{1/p},$$

где p=1, 2, 3, … .

Оценить уровень шума изображения позволяет нормированная абсолютная погрешность:

$$NAE = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} \left| x_{ij} - \bar{x}_{ij} \right|}{\sum_{i=1}^{M} \sum_{j=1}^{N} \left| x_{ij} \right|}.$$

Верность изображения можно оценить, используя формулу

$$IF = 1 - \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} \left(x_{ij} - \bar{x}_{ij} \right)^2}{\sum_{i=1}^{M} \sum_{j=1}^{N} x_{ij}^2}.$$

Вышеприведенные оценки качества удобны в использовании, но далеко не всегда позволяют объективно оценить качество изображения, особенно с точки зрения визуального восприятия.

Большинство известных подходов к оценке качества изображения использует лишь один параметр. Увеличение количества рассматриваемых параметров качества изображения позволяет повысить точность оценки, но вместе с тем повышает вычислительную сложность метода.

HTML-ДОКУМЕНТЫ

Язык разметки гипертекста HTML (hypertext markup language) это стандартный язык для создания гипертекстовых документов в глобальной сети Internet [3, 6]. Гипертекстовый документ представляет собой текстовый файл, имеющий специальные метки (теги) для отображения содержимого файла на экране компьютера. С помощью таких меток можно выделять заголовки документа, изменять цвет, размер и начертание букв, вставлять графические изображения и таблицы. Основным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа гиперссылок – специальных конструкций языка HTML, позволяющих переходить к просмотру другого документа.

Структура документа HTML

Каждый документ HTML имеет идентифицирующую его строку заголовка, который задается с помощью операторов <TITLE> и </TITLE> и отображается в главном окне навигатора. Размер строки заголовка не ограничен, однако если удаленный пользователь работает в видеорежиме с низким разрешением (например, 640×480 пикселов), длинная строка заголовка страницы может не поместиться в окне просмотра.

Документ HTML состоит из двух основных частей – заголовка документа и тела документа. Заголовок документа выделяется тегами <HEAD> и </HEAD>, может содержать другие операторы и занимать несколько строк. Тело документа выделяется тегами <BODY> и </BODY>, занимает одну или несколько строк и отображается во внутренней области окна навигатора. Структуру документа HTML можно представить как:

```
<HTML>
<HEAD>
   . . . (текст заголовка документа)
</HEAD>
<BODY>
  . . . (тело документа)
</BODY>
</HTML>
```

Параграфы текста html-страницы выделяются тегами <P> и </P>:

```
<BODY>
  <P>Первый параграф текста</P>
  <P>Второй параграф текста</P>
</BODY>
```

Оператор </P> является необязательным и может иметь параметры, определяющие различные характеристики параграфа: ALIGN (выравнивание параграфа по горизонтали); LEFT (выравнивание по левой границе); CENTER (центрирование); RIGHT (выравнивание по правой границе).

Выравнивание строк параграфа можно задать следующими путями:

```
<P ALIGH=CENTER> Текст параграфа
<P><CENTER> Текст параграфа </CENTER>
```

Форматирование текста и символов

Разметка HTML-документа и управление его отображением осуществляется с помощью тегов, определяющих вид текста, преобразование его компонент в гипертекстовые ссылки, графические или мультимедийные объекты, которые должны быть включены в документ. Графическая и звуковая информация, включаемая в HTML-документ, хранится в отдельных файлах. Для файлов, содержащие HTML-документы, приняты расширения .htm или .html.

К форматированию текста относится набор средств, позволяющих задавать размер и стиль начертания шрифта, цвет, а также логическое форматирование. Базовый шрифт определяется оператором <BASEFONT…>, а размер шрифта задается параметром SIZE (по умолчанию SIZE=3):

<P><BASEFONT SIZE=3>Размер шрифта равен 3

Значение параметра SIZE размера базового шрифта соответствует следующим размерам символов в пикселах: 1 – 9пт; 2 – 10 пт; 3 – 12 пт; 4 – 14 пт; 5 – 18 пт; 6 – 24 пт; 7 – 36 пт.

Оператор <FONT…> с параметром SIZE позволяет изменить текущий размер шрифта относительно заданного базового значения. В качестве значения для параметра SIZE можно указывать знаки + или –, например:

<P><BASEFONT SIZE=3>Размер базового шрифта равен 3

<P><FONT SIZE=+3>Размер шрифта равен 6.

Увеличение или уменьшение высоты символов текста осуществляется с помощью операторов <BIG> и <SMALL>, используемых в паре с операторами </BIG> и </SMALL>.

Цвет элементов html-документа определяется соответствующими операторами, в качестве параметров которых необходимо задавать либо численное значение отдельных компонент цвета, либо символическое название цвета. Численное значение указывается в виде #RRGGBB, где RR определяет содержание красной компоненты цвета, GG – зеленой, а BB – голубой. Значение каждой из компонент может изменяться в диапазоне от 0 до FF. В режиме VGA цвет можно указывать символически: Aqua (морская волна, #00FFFF); Black (черный, #000000); Blue (голубой, #0000FF); Fuchsia (малиновый, #FF00FF); Gray (серый; #808080); Green (зеленый, #00FF00); Lime (ярко-зеленый, #008000); Maroon (темно-красный, #800000); Navy (темно-синий, #000080); Olive (оливковый, #808000); Purple (пурпурный; #800080); Red (красный, #FF0000); Silver (серебряный, #C0C0C0); Teal (темная морская волна, #008080); White (белый, #FFFFFF); Yellow (желтый, #FFFF00).

Изменение цвета фона html-документа осуществляется с помощью параметра BGCOLOR оператора <BODY>:

<BODY BGCOLOR=#FFFF00>

Цвет текста задается в операторе <FONT…> параметром COLOR:

<FONT COLOR=RED>

HTML-операторы физического форматирования символов текста определяют внешний вид символов явным образом и являются парными (табл. 5).

Таблица 5

Основные операторы физического форматирования текста

Оператор	Описание
<B>, </B>	Выделение жирным шрифтом
<I>, </I>	Выделение наклонным шрифтом
<U>, </U>	Выделение подчеркиванием
<STRIKE>, </STRIKE>	Выделение перечеркиванием
<TT>, </TT>	Шрифт с фиксированной шириной букв
<BIG>, </BIG>	Текст с крупным размером букв
<SMALL>, </SMALL>	Текст с малым размером букв
<BLINK>, </BLINK>	Мигающий текст (только в Netscape Navigator)
_,	Подстрочный индекс
[,]	Надстрочный индекс

Операторы логического форматирования предназначены для указания семантического (смыслового) назначения оформляемого текста. Операторы данной группы являются парными и определяют не способ оформления, а тип информации, подлежащей выделению (табл. 6).

Таблица 6

Основные операторы логического форматирования текста

Оператор	Описание
<CITE>,	Цитата
<EM>, </EM>	Текст, имеющий особое значение
<STRONG>, </STRONG>	Сильное выделение текста
<KBD>, </KBD>	Текст, введенный пользователем
<CODE>, </CODE>	Листинг программы
<SAMP>, </SAMP>	Последовательность литералов
<VAR>, </VAR>	Имя переменной
<!-- ... -->	Комментарий

По возможности следует использовать вместо физического логическое форматирование символов, так как оно позволяет пользователю самостоятельно выбирать способ оформления указанных логических составляющих текста. Тег <KBD> предназначен для выделения текста шрифтом с фиксированной шириной символов (например, образец команд, введенных пользователем). Тег <CODE> предназначен для публикации листингов программ в документах HTML и предполагает использование нежирного шрифта с фиксированной шириной символов. Тег <SAMP> позволяет выделить отдельные слова или последовательность литеральных символов. Тег <VAR> предназначен для выделения имен переменных или функций и обычно используется при описании листингов программ.

Вставка в документ HTML одной или нескольких строк комментария, не отображаемых навигатором, осуществляется операторами <!-- ... --> и <COMMENT> (например, <!-- Текст однострочного комментария -->).

Для удобства оформления документов HTML, как правило, используются файлы таблиц стилей, которые содержат определения стилей. Определение стиля начинается с имени класса, вслед за которым в фигурных скобках перечисляются параметры стиля. Отдельные параметры стилевого оформления задаются своими именами, после которых через символ двоеточия следует значение параметра. Параметры отделяются символом точка с запятой. Например, определение стиля H1 для шрифта размером 24 пункта с утолщением, красного цвета и с отступом с левой стороны на 10 процентов от ширины экрана будет выглядеть:

H1 { font-size: 24; font-weigt: bold; color: red; margin-left: 10% }

Списки и специальные символы

В html-документах возможно использование двух типов списков: упорядоченных и неупорядоченных. Упорядоченные (нумерованные) списки создаются с помощью парных операторов <OL> и </OL>. Атрибут TYPE, указанный после открывающего список оператора <OL> устанавливает тип маркера: A – маркер в виде прописных букв, a – маркер в виде строчных букв, I – маркер в виде больших римских цифр, i – маркер в виде маленьких римских цифр, 1 – маркер в виде арабских цифр.

Для создания неупорядоченного списка предназначен оператор <UL>, который используется в паре с оператором </UL>, закрывающем список. Каждая строка в списке отмечается оператором <LI>, в которую может быть добавлен атрибут, определяющий внешний вид символа, используемого для выделения строки в списке: disk – круглая жирная точка (по умолчанию), circle – окружность, square – маленький черный квадрат.

Таблица 7

Оформление списков в html-документах

html-код	Вид html-страницы
<OL type=”1”> <LI> Фивы <LI> Гиза <LI> Мемфис </OL>	1. Фивы 2. Гиза 3. Мемфис
<UL type="circle"> <LI>Эхнатон <LI type="square">Тутанхамон <LI>Нефертити </UL>	○ Эхнатон ▪ Тутанхамон ○ Нефертити

В языке HTML определены четыре специальных символа, предназначенных для служебных целей – это символы <, >, & и “. Эти символы нельзя вставлять в обычный текст, вместо этого необходимо использовать замену: < > & " соответственно.

Списки и элементы списков являются блочными элементами, т.е. перед ними и после них автоматически добавляются пустые строки.

Таблицы в документах HTML

Таблицы в документах HTML определяются между командами <TABLE> и </TABLE> и могут быть вложены друг в друга. В простейшем случае строки таблицы ограничиваются тегами <TR> и </TR>, а столбцы – тегами <TD> и </TD>:

```
<TABLE frame=border BORDERCOLOR=black>
 <TR><TD>п/п</TD><TD>Пирамида</TD><TD>Высота</TD></TR>
 <TR><TD>1</TD><TD>Хеопса</TD><TD>146,6м</TD></TR>
 <TR><TD>2</TD><TD>Хефрена</TD><TD>143,5м</TD></TR>
</TABLE>
```

Результат будет выглядеть следующим образом:

п/п	Пирамида	Высота
1	Хеопса	146,6м
2	Хефрена	143,5м

Атрибуты, определяющие внешний вид таблицы в целом, задаются следующими параметрами:

ALIGN – задает положение данных в таблице по горизонтали (left – таблица выравнивается по левому краю окна просмотра; center - центрирование таблицы; right - выравнивание по правому краю; justify - выравнивание по ширине);

BACKGROUND – использование в качестве фона растрового графического изображения;

BGCOLOR – цвет фона для таблицы;

BORDER – установка толщины рамки (в пикселах);

BORDERCOLOR – установка цвета рамки;

BORDERCOLORDARK – темный цвет, используемый для трехмерного выделения рамки;

BORDERCOLORLIGHT – светлый цвет, используемый для трехмерного выделения рамки;

CELLPADDING – расстояние между краями ячейки таблицы и содержимым этой ячейки;

CELLSPACING – расстояние между данными ячейки и ее границами, или между ячейками таблицы;

COLS – количество столбцов в таблице;

FRAME – внешний вид рамки вокруг таблицы (BORDER – рамка отображается со всех сторон таблицы (по умолчанию); VOID – внешняя рамка не отображается; ABOVE – верхняя рамка; BELOW – нижняя рамка; HSIDES – верхняя и нижняя рамка; LHS –левая рамка; RHSXE "RHS" – правая рамка; VSIDESXE "VSIDES" – рамка слева и справа; BOXXE "BOX" –только внешняя рамка);

NOWRAP – отмена переноса слов на следующую строку;

RULES – внешний вид линий между ячейками таблицы (NONE – без разделительных линий между ячейками; GROUPS – горизонтальные разделительные линии между всеми группами таблиц; ROWS - горизонтальные линии между всеми строками; COLS – вертикальные разделительные

линии между столбцами; ALL – все разделительные линии таблицы);

VALIGN – положение таблицы по вертикали (top – выравнивание по верхней границе; middle – выравнивание по середине окна; bottom – выравнивание по нижней границе; baseline – выравнивание по базовой линии текста);

WIDTH – ширина таблицы в пикселах или в процентах от ширины окна просмотра навигатора.

Оператор <TR> предназначен для создания строк таблицы, а его заданные атрибуты подавляют атрибуты таблицы. Внешний вид строки определяют параметры: ALIGN, BORDERCOLOR, BORDERCOLORDARK, BORDERCOLORLIGHT, NOWRAP, VALIGN.

Ячейки таблицы задаются с помощью оператора <TD>, атрибуты которого подавляют атрибуты строки и таблицы. Форматирование ячеек осуществляется с помощью параметров ALIGN, BORDERCOLOR, BACKGROUND, BGCOLOR, BORDERCOLORDARK, BORDERCOLORLIGHT, COLSPAN (значение параметра определяет количество объединяемых соседних ячеек в одной колонке), HEIGHT (высота ячейки в пикселах), NOWRAP, ROWSPAN (количество объединяемых соседних ячеек в строке таблицы), VALIGN, WIDTH.

Изображения и анимация

Язык html позволяет выводить на страницы стационарные изображения, анимированные изображения, а также бегущий текст.

Вставка изображений осуществляется с помощью тега <img> с атрибутами: src – URL-адрес файла-изображения; alt – текстовое описание изображения в области вывода изображения или всплывающая подсказка; height, width – высота и ширина выводимого изображения (атрибут позволяет быстрее загружать страницу); border – ширина рамки вокруг изображения; vspace, hspace – расстояние по вертикали и по горизонтали от изображения до кромки текста, соответственно; align – положение текста относительно изображения.

Если тег <img src> указывает на несуществующее изображение, веб-приложение заменяет его фиктивным изображением. Некоторые вебмастеры, чтобы составить из нескольких рисунков один, используют таблицы без рамок.

Создание бегущей строки осуществляется с помощью тегов <marquee> и </marquee>. Атрибуты открывающего тега задают параметры анимации: height, width – высота и ширина бегущей строки в пикселах или процентах от ширины и высоты окна, соответственно; bgcolor – цвет бегущей строки; behavior – тип движения (scroll – прокрутить; slide – слайды; alternate – поочередно менять направление), direction – направление движения (left – влево, right – вправо по строке); scrollamount и scrolldelay – число пикселов, на которое должно переместиться изображение за заданное число миллисекунд, соответственно; loop – число повторов анимации (по умолчанию – непрерывно).

ЛИТЕРАТУРА

1. *Алексеев В.Е., Таланов В.А.* Графы и алгоритмы. Структуры данных. Модели вычислений. – М.: Бином, 2006.

2. *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. – М.: Техносфера, 2006.

3. *Дуванов А.А.* Web-конструирование. Элективный курс. – СПб.: БХВ-Петербург, 2006.

4. *Дьяконов В.П.* MATLAB 6.5 SP1/7/7 SP1 + Simulink 5/6. Работа с изображениями и видеопотоками. – М.: СОЛОН-Пресс, 2005.

5. *Журавель И.М.* Краткий курс теории обработки изображений. // matlab.exponenta.ru/imageprocess/book2/index.php

6. *Коржинский С.Н.* Настольная книга Web-мастера: эффективное применение HTML, CSS и JavaScript / С.Н. Коржинский . – 2-е изд., испр. и доп . – М.: КноРус. 2000.

7. *Куприянов М.С., Матюшкин Б.Д.* Цифровая обработка сигналов. Процессоры, алгоритмы, средства проектирования. – Изд. 2-е, перераб. и доп. – СПб.: Политехника, 2000.

8. *Левричук Ю.П., Охинченко Е.П., Сотников А.Д., Фоменко Т.А.* Информатика. Часть 2. Интернет-технологии: методические рекомендации и задания к лабораторным работам и курсовому проектированию. [Электронный ресурс]. http://dvo.sut.ru/libr/ite/i280levc/

9. *Магда Ю.С.* Unix. - СПб.: БХВ-Петербург, 2006.

10. *Максимов Н.В., Попов И.И.* Компьютерные сети. Изд. 2-е, перераб. и доп. – М.: Форум - ИнфраМ, 2007.

11. *Павловская Т.А.* Паскаль. Программирование на языке высокого уровня: учебник для вузов. – СПб.: Питер, 2007.

12. *Патрыка Т.Я., Попов И.И.* Операционные системы, среды и оболочки. – М.: Форум - ИнфраМ, 2007.

13. *Прэтт Э.* Цифровая обработка изображений. В 2-х книгах – М.: Мир, 1982.

14. *Рейнбоу В.* Компьютерная графика. Энциклопедия. – СПб.: Питер, 2003.

15. *Смоленцев Н.К.* Основы теории вейвлетов. Вейвлеты в MATLAB. – М.: ДМК Пресс, 2005.

16. *Соломенчук В.* Linux. Экспресс курс. – СПб.: БХВ-Петербург, 2006.

17. *Стариченко Б.Е.* Теоретические основы информатики: Учебное пособие для вузов. – 2-е изд. перераб. и доп. – М.: Горячая линия - Телеком, 2003.

18. *Хейр К. и др.* Внутренний мир Unix. Пер. с англ. – Киев: ДиаСофт, 1998.

В 2007 году СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007–2008 годы. Реализация инновационной образовательной программы «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий» позволит выйти на качественно новый уровень подготовки выпускников и удовлетворить возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях экономики.

КАФЕДРА ИЗМЕРИТЕЛЬНЫХ ТЕХНОЛОГИЙ И КОМПЬЮТЕРНОЙ ТОМОГРАФИИ

Кафедра измерительных технологий и компьютерной томографии, в прошлом кафедра часового производства и приборов точной механики, была создана одновременно с основанием Университета, который ведет свою историю от образования в 1900 г. Ремесленного училища цесаревича Николая. Основателем кафедры был Норберт Болеславович Завадский – первый заведующий механико-оптическим отделением этого училища.

В 1920 г. механико-оптическое отделение было реорганизовано в техникум точной механики и оптики, который с 1925 г. начал подготовку инженеров-приборостроителей. В дальнейшем техникум был преобразован в учебный комбинат, ФЗУ и в 1933 г. стал институтом точной механики и оптики. В течение этого времени кафедрой заведовал профессор Н.Б. Завадский, который преподавал дисциплины, связанные с точной механикой. В 1930 г. кафедру возглавил Лаврентий Павлович Шишелов. На кафедре читались дисциплины «Теория часовых механизмов», «Электроизмерительные приборы», «Механические приборы». В 1935 г. из состава кафедры выделилось направление гироскопических устройств. Была образована отдельная кафедра навигационных приборов. В 1940 г. на кафедре защитил кандидатскую диссертацию Захар Маркович Аксельрод, впоследствии доктор технических наук, возглавивший кафедру во время войны.

После войны кафедра приборов точной механики выпускала специалистов по часовому производству и производству точного измерительного инструмента. На кафедре читались курсы «Приборы времени», «Приборы для измерения скоростей и ускорений», «Тахометры», «Основы конструирования приборов точной механики». С 1976 г. кафедру возглавил Борис Александрович Арефьев, известный специалист в области автоматического управления и газовых опор. В это время на кафедре велась подготовка специалистов по специальности «Приборы точной механики».

С 1985 г. кафедрой заведовал профессор Владислав Александрович Иванов. В связи с развитием техники и потребностью в выпуске инженерных кадров по разработке и эксплуатации томографов в 1996 г. кафедра начала подготовку инженеров по специализации «Компьютерная томография».

В 2005 г. кафедру возглавила д.т.н. Мария Яковлевна Марусина. В настоящее время на кафедре ведется подготовка бакалавров, специалистов и магистров по направлению «Приборостроение» по специализациям «Информационно-измерительные комплексы» и «Томографические методы диагностики».

Анна Олеговна Казначеева

Основы информационных технологий

Учебное пособие

В авторской редакции
Дизайн обложки А.О. Казначеева
Редакционно-издательский отдел
Санкт-Петербургского государственного университета
информационных технологий, механики и оптики
Зав. РИО Н.Ф. Гусарова
Лицензия ИД № 00408 от 05.11.1999
Подписано к печати 03.2009
Отпечатано на ризографе Тираж 100 экз. Заказ № 2043