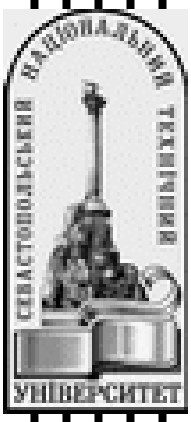
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
Севастопольский национальный технический университет



МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению расчетно-графического задания
по дисциплине «Распределенные системы и сети»
для студентов направления 6.050102 «Компьютерная инженерия»,
специальности 7(8).05010201 «Компьютерные системы и сети»
всех форм обучения

Севастополь
2013

УДК 004.75

Методические указания к выполнению расчетно-графического задания по дисциплине «Распределенные системы и сети» для студентов направления 6.050102 «Компьютерная инженерия», специальности 7(8).05010201 «Компьютерные системы и сети» всех форм обучения / Сост. Н.Л. Корепанова, К.С. Ткаченко. — Севастополь: Изд-во СевНТУ, 2013. — 16 с.

Методические указания содержат требования к выполнению расчетно-графического задания и оформлению отчета к нему, варианты индивидуальных заданий, контрольные вопросы по дисциплине «Распределенные системы и сети».

Методические указания предназначены для студентов направления 6.050102 «Компьютерная инженерия», специальности 7(8).05010201 «Компьютерные системы и сети» всех форм обучения.

Методические указания рассмотрены и утверждены на заседании кафедры кибернетики и вычислительной техники (протокол № _9 от 15.05.2013 г.)

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент:
Грушун А.И., доцент кафедры технической кибернетики, доцент, канд. техн. наук.

Ответственный за выпуск:
Брюховецкий А.А., и.о. зав. кафедрой кибернетики и вычислительной техники, доцент, канд. техн. наук.

СОДЕРЖАНИЕ

	Цель работы	3
1.	Задание 1. Internet-сокеты	4
2.	Задание 2. Устройства генерирования случайных чисел	6
3.	Задание 3. Функция ioctl()	7
4.	Задание 4. Извлечение информации из файловой системы /proc	9
5.	Требования к выполнению отчета и критерии оценки выполнения	10
6.	Варианты индивидуальных заданий	11
7.	Контрольные вопросы	13
	Библиографический список использованной литературы	16

Цель работы

Целью методических указаний (МУ) к выполнению расчетно-графического задания (РГЗ) по дисциплине «Распределенные системы и сети» (РСС) является изучение студентами теоретических основ и принципов построения современных и перспективных вычислительных сетей, основ программирования на специальных языках, операционных сетевых систем, таких как LINUX, UNIX, а также получение основных сведений о компьютерных сетевых технологиях [1—10]. Изучение материала данной дисциплины создает основу использования распределенных и компьютерных сетей в процессе решения специальных задач обработки и анализа информации в различных сферах деятельности.

В результате выполнения РГЗ студент должен знать: тенденции развития структур операционных систем LINUX, UNIX; основные типы структур сетей ЭВМ и примеры их реализации; основные стандарты вычислительных сетей; принципы организации сетей передачи данных и сетевых протоколов; структуру пакетов и кадров стандарта X25 и основные режимы обмена данными; особенности построения и области использования локальных сетей (IEEE.802.4, 802.5, 802.7); структуру сети ETHERNET и ее сетевых контролеров; основы построения сетей управления производством (MAP/TOP); средства объединения вычислительных сетей; наиболее распространенное программное обеспечение для распределенной обработки экономической информации.

В результате выполнения РГЗ студент должен уметь: по техническим требованиям выбрать структуру сети или системы при использовании ОС LINUX , режим их функционирования; разработать структурные и функциональные схемы узлов сети, оценить спроектированный проект; инсталлировать локальную вычислительную сеть, настраивать ее конфигурацию; работать с вычислительной сетью; разрабатывать сетевое прикладное программное обеспечение. В результате выполнения РГЗ студент должен ознакомиться: с основными методами компьютерной обработки информации на базе ОС LINUX с использованием современных программных

систем, с основными техническими и программными средствами сетей на базе LINUX, с основами алгоритмизации вычислительных процессов, с основными сведениями о компьютерных сетевых средах.

Выполнение РГЗ по РСС способствует освоению студентами, практически всех последующих дисциплин, так как современное преподавание в качестве основного инструмента использует вычислительные и программные средства сетей.

1. Задание 1. Internet-сокеты

UNIX-сокеты используются для организации взаимодействия двух процессов, выполняющихся на одном компьютере. С другой стороны, Internet-сокеты позволяют соединять между собой процессы, работающие на разных компьютерах. Пространству имен Internet соответствует константа PF_INET. Internet-сокеты чаще всего работают по протоколам TCP/IP. Протокол IP (Internet Protocol) отвечает за низкоуровневую доставку сообщений, осуществляя при необходимости их разбивку на пакеты и последующую компоновку. Доставка пакетов не гарантируется, поэтому они могут исчезать или приходить в неправильном порядке. Каждый компьютер в сети имеет свой IP-адрес. Протокол TCP (Transmission Control Protocol) функционирует поверх протокола IP и обеспечивает надежную доставку сообщений, ориентированную на установление соединений. Адрес Internet-сокета состоит из двух частей: адреса компьютера и номера порта. Эта информация хранится в структуре типа sockaddr_in. В поле sin_family необходимо записать константу AF_INET, указывающую на то, что адрес принадлежит пространству имен Internet. В поле sin_addr хранится IP-адрес компьютера в виде 32-разрядного целого числа. Благодаря номерам портов можно различать сокеты, создаваемые на одном компьютере.

Требуется разработать программу, которая запрашивает ресурс у Web-сервера, адрес которого указан в командной строке. Варианты заданий приведены в таблице 2. Образец выполнения приводится в листинге 1. Программа извлекает имя Web-сервера. Далее вызывается функция gethostbyname(), которая преобразует имя сервера в числовое представление. После этого программа подключает потоковый (TCP) сокет к порту 80 сервера. Web-серверы общаются по протоколу HTTP (Hypertext Transfer Protocol), поэтому программа посылает команду GET, в ответ на которую сервер возвращает текст начальной страницы.

Листинг 1 — Чтение страницы с Web-сервера

```
#include <stdlib.h>
#include <stdio.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/socket.h>
```

Листинг 1 — Чтение страницы с Web-сервера (продолжение)

```
#include <unistd.h>
#include <string.h>

void get_home_page (int socket_fd)
{
  char buffer[10000];
  ssize_t number_characters_read;

  sprintf (buffer, "GET /\n");
  write (socket_fd, buffer, strlen (buffer));
  while (1) {
    number_characters_read = read (socket_fd, buffer, 10000);
    if (number_characters_read == 0)
      return;
    fwrite (buffer, sizeof (char), number_characters_read, stdout);
  }
}

int main (int argc, char* const argv[])
{
  int socket_fd;
  struct sockaddr_in name;
  struct hostent* hostinfo;

  socket_fd = socket (PF_INET, SOCK_STREAM, 0);
  name.sin_family = AF_INET;
  hostinfo = gethostbyname (argv[1]);
  if (hostinfo == NULL)
    return 1;
  else
    name.sin_addr = *((struct in_addr *) hostinfo->h_addr);
  name.sin_port = htons (80);

  if (connect (socket_fd, &name, sizeof (struct sockaddr_in)) == -1) {
    perror ("connect");
    return 1;
  }
  get_home_page (socket_fd);

  return 0;
}
```

2. Задание 2. Устройства генерирования случайных чисел

Специальные устройства /dev/random и /dev/urandom предоставляют доступ к средствам генерирования случайных чисел, встроенным в ядро Linux. Большинство аналогичных программных функций, например функция rand() стандартной библиотеки языка C, в действительности генерируют псевдослучайные числа. Такие числа имеют некоторые свойства случайных последовательностей, но их можно воспроизвести: достаточно задать то же самое инициализирующее значение, чтобы получить одинаковую последовательность чисел. Такое поведение неизбежно, ведь внутренняя работа компьютера жестко определена и предсказуема. Но в ряде приложений это крайне нежелательно. Например, можно взломать криптографический шифр, если воспроизвести последовательность случайных чисел, лежащих в его основе.

Чтобы получить настоящие случайные числа, необходим внешний «источник хаоса». Ядро Linux использует в качестве такого источника пользователя. Замеряя задержки между действиями пользователя, в частности нажатиями клавиш и перемещениями мыши, ядро способно генерировать непредсказуемый поток действительно случайных чисел. Получить доступ к этому потоку можно путем чтения из устройств /dev/random и /dev/urandom.

Разница между устройствами проявляется, когда запас случайных чисел в ядре Linux заканчивается. Если попытаться прочесть большое количество байтов из устройства /dev/random и при этом не выполнять никаких пользовательских действий (не нажимать клавиши, не перемещать мышь и т.п.), система заблокирует операцию чтения. Только когда пользователь проявит какую-то активность, система сгенерирует дополнительные случайные числа и передаст их программе. В противоположность этому операция чтения из устройства /dev/urandom никогда не блокируется. Если в системе кончаются случайные числа, Linux использует криптографический алгоритм, чтобы сгенерировать псевдослучайные числа из последней цепочки случайных байтов.

В листинге 2 показана функция, которая генерирует случайное число, читая байты из файла /dev/random. Помните, что операция чтения из этого файла окажется заблокированной в случае нехватки случайных чисел. Если важна скорость работы функции, воспользуйтесь файлом /dev/urandom.

Требуется разработать на основе листинга 2 в соответствии с вариантом задания из таблицы 2 программу генерации случайных чисел.

Листинг 2 — Генерирование случайного числа с использованием /dev/random

```
#include <assert.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
```

Листинг 2 — Генерирование случайного числа с использованием /dev/random (продолжение)

```
int random_number (int min, int max)
{
  static int dev_random_fd = -1;

  char* next_random_byte;
  int bytes_to_read;
  unsigned random_value;

  assert (max > min);

  if (dev_random_fd == -1) {
    dev_random_fd = open ("/dev/random", O_RDONLY);
    assert (dev_random_fd != -1);
  }

  next_random_byte = (char*) &random_value;
  bytes_to_read = sizeof (random_value);
  do {
    int bytes_read;
    bytes_read = read (dev_random_fd, next_random_byte, bytes_to_read);
    bytes_to_read -= bytes_read;
    next_random_byte += bytes_read;
  } while (bytes_to_read > 0);

  return min + (random_value % (max - min + 1));
}
```

3. Задание 3. Функция ioctl()

Linux, как и большинство операционных систем, взаимодействует с аппаратными устройствами посредством модульных программных компонентов, называемых драйверами. Драйвер скрывает от операционной системы детали взаимодействия с устройством и предоставляет в распоряжение системы стандартный интерфейс обращения к устройству. Драйвер может подключаться к ядру статически либо по запросу в виде модулей. Он недоступен напрямую пользовательским процессам. Имеются специальные механизмы — файловые объекты, позволяющие процессам взаимодействовать с драйверами, а через них и с аппаратными устройствами. Такие объекты являются частью операционной системы, поэтому программы могут открывать их, читать из них данные и осуществлять в них запись точно так же, как с обыкновенными файлами.

Файлы устройств не являются обычными файлами: с ними не связаны блоки данных на диске. Данные, помещаемые в такой файл или извлекаемые из него передаются соответствующему драйверу устройства или принимаются от него, а драйвер осуществляет обмен данными с обслуживаемым устройством.

Существует два типа устройств: символьные, которые читают и записывают данные в виде потока байтов (накопители на магнитной ленте, терминалы, звуковые платы и прочее) и блочные, которые читают и записывают данные блоками фиксированного размера (накопители на жестких и гибких магнитных дисках, флеш-память и прочее).

Устройство идентифицируется двумя числами: старшим номером устройства и младшим номером устройства. Старший номер соответствует драйверу устройства (соответствие задано в исходном коде ядра), младший — отдельному устройству.

Системный вызов ioctl() — это универсальное средство управления аппаратными устройствами. Первым аргументом функции является дескриптор файла того устройства, которым требуется управлять. Второй аргумент — это код запроса, обозначающего выполняемую операцию. Разным устройствам соответствуют разные запросы. В зависимости от запроса функции ioctl() могут потребоваться дополнительные аргументы. Многие коды запросов перечислены на man-странице ioctl_list. При работе с функцией ioctl() нужно хорошо понимать, как работает драйвер соответствующего устройства.

В листинге 3 представлена программа, которая запрашивает извлечение компакт-диска из дисковода CD-ROM. Программа принимает единственный аргумент командной строки: имя дисковода CD-ROM. Программа открывает файл устройства и вызывает функцию ioctl() с кодом запроса CDROMEJECT. Этот код определен в файле <linux/cdrom.h> и служит устройству указанием извлечь компакт-диск из дисковода. Например, если в системе имеется IDE-дисковод CD-ROM, подключенный в качестве главного устройства к дополнительному IDE-контроллеру, соответствующий файл устройства будет называться /dev/hdc.

Требуется разработать программу, выполняющую работу с приводом CD-ROM, на основании листинга 3 и таблицы 3.

Листинг 3 — Извлечение компакт-диска из привода

```
#include <fcntl.h>
#include <linux/cdrom.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main (int argc, char* argv[])
{
  int fd = open (argv[1], O_RDONLY);
```

Листинг 3 — Извлечение компакт-диска из привода (продолжение)

```
  ioctl (fd, CDROMEJECT);
  close (fd);

  return 0;
}
```

4. Задание 4. Извлечение информации из файловой системы /proc

Специальная файловая система /proc не связана с аппаратным устройством, файлам в ней не соответствуют реальные файлы на физическом устройстве. Но через эти файлы обеспечивается доступ к параметрам, служебным структурам и статистической информации ядра. Содержимое таких файлов генерируется ядром динамически в процессе чтения из файла. Осуществляя запись в определенные файлы можно менять конфигурацию работающего ядра системы. Поскольку содержимое файла создается ядром «на лету» понятие размера файла не применимо, а время модификации равно времени обращения к нему.

Большинство элементов файловой системы /proc выдает информацию в отформатированном виде. Файл /proc/cpuinfo содержит информацию о процессоре (или процессорах, если это многопроцессорный компьютер). Выходная информация представляется в виде таблицы значений, по одному на строку. Каждое значение сопровождается символическим идентификатором.

Требуется на основе листинга 4 и варианта задания из таблицы 3 разработать программу определения заданного вариантом параметра процессора (процессоров).

Листинг 4 — Определение частоты процессора путем анализа файла /proc/cpuinfo

```
#include <stdio.h>
#include <string.h>

float get_cpu_clock_speed ()
{
  FILE* fp;
  char buffer[1024];
  size_t bytes_read;
  char* match;
  float clock_speed;

  fp = fopen ("/proc/cpuinfo", "r");
  bytes_read = fread (buffer, 1, sizeof (buffer), fp);
  fclose (fp);
  if (bytes_read == 0 || bytes_read == sizeof (buffer))
```

Листинг 4 — Определение частоты процессора путем анализа файла /proc/cpuinfo (продолжение)

```
    return 0;
  buffer[bytes_read] = '\0';
  match = strstr (buffer, "cpu MHz");
  if (match == NULL)
    return 0;
  sscanf (match, "cpu MHz : %f", &clock_speed);
  return clock_speed;
}

int main ()
{
  printf ("CPU clock speed: %4.0f MHz\n", get_cpu_clock_speed ());
  return 0;
}
```

5. Требования к выполнению отчета и критерии оценки выполнения

Отчёт о выполнении расчетного задания представляется в виде документа, выполненного на листах формата А4, либо по решению преподавателя — стандартной тетради, включающего титульный лист стандартного образца, содержание расчетного задания с указанием страниц, цель работы, тексты отдельных заданий и результаты выполнения расчетов с обязательными пояснениями, библиографический список использованной литературы. На титульном листе обязательно должны присутствовать номер зачетной книжки, номер студента в списке группы и номер выполняемого варианта. Текст выполнения каждого задания должен содержать: 1. постановку задачи по варианту; 2. описание действий, необходимых для решения задачи; 3. листинги программ; 4. результаты функционирования программ; 5. анализ результатов, выводы по проделанной работе. Отчёт может быть выполнен в рукописном виде, в машинописном виде, в виде текста, набранного на компьютере. Аккуратность в выполнении отчета строго обязательна.

Оценка за РГЗ выставляется на основе следующих квалификационных характеристик: отлично (А) — результаты выполнения РГЗ свидетельствуют о глубоком и всестороннем знании методик дисциплины, грамотном изложении ответов, умении на основе теоретических знаний и полученных навыков квалифицированно оценивать результаты расчетов и делать логические выводы;

хорошо (В, С) — результаты выполнения РГЗ свидетельствуют о твердом и достаточно полном знании методик дисциплины, отсутствии существенных неточностей при их использовании, умении правильно оценивать результаты расчетов и делать логические выводы;

удовлетворительно (D, E) — РГЗ выполнено при посредственном знании методик дисциплины, отсутствии грубых ошибок при их использовании, умении в основном правильно оценивать результаты расчетов и делать логические выводы;

неудовлетворительно (F, Fx) — результаты выполнения РГЗ содержат грубые ошибки, свидетельствующие о полном незнании методик дисциплины, отсутствии навыков расчетов и неумении дать им адекватную оценку.

6. Варианты индивидуальных заданий

Задания выполняются непосредственно в операционной системе Debian GNU/Linux, версия не ниже 5.0.3, с использованием трансляторов Gnu Compilers Collection.

Компилятор языка C называется gcc. При компиляции исходного файла нужно указывать опцию -c. Для компиляции в командной строке файла main.c нужно выполнить команду «gcc -c main.c».

Компилятор языка C++ называется g++. При компиляции исходного файла нужно указывать опцию -c. Для компиляции в командной строке файла main.cpp нужно выполнить команду «g++ -c main.cpp».

После того, как исходные коды программ скомпилированы, необходимо скомпоновать полученные объектные файлы. Если все файлы написаны на C, то можно использовать gcc, иначе — g++. Например, «gcc -o main main.o».

Запуск осуществляется командой «./main».

Исходный код программы для задания сохранять в каталоге суперпользователя root (/root) под именем ГГССНВВ.c, где ГГ — номер группы, СС — номер студента в списке группы, Н — номер задания (1—4), ВВ — номер варианта из таблиц 1—4. Для редактирования исходного кода рекомендуется использовать редактор nano, для компилирования и компоновки — команду «gcc –o ГГССНВВ ГГССНВВ.c». Например, студент группы М41д, 10 по списку, выполняющий 3 задание с вариантом 01 вводит в терминале следующие команды:

```
# nano 4110301.c

# gcc -o 4110301 4110301.c
```

При выполнении индивидуальных вариантов рекомендуется широко использовать возможности справочной системы, доступ к которой обеспечивается с использованием команд «man команда» и «info команда».

Вариант индивидуального задания определяется как остаток от деления последних двух цифр зачетной книжки на количество вариантов в таблице.

Таблица 1 — Варианты индивидуальных заданий к заданию № 1 «Internet-сокеты»

№ варианта	Маска адреса клиента	Адрес сервера	Имя ресурса
1.	255.0.0.0	10.0.1.100	abc.html
2.	255.255.0.0	120.120.1.100	index.html
3.	255.255.255.0	129.129.1.100	photo.html
4.	255.0.0.0	190.190.1.100	abc.html
5.	255.255.0.0	193.193.1.100	index.html
6.	255.255.255.0	220.220.1.100	photo.html
7.	255.0.0.0	10.0.1.100	abc.html
8.	255.255.0.0	120.120.1.100	index.html
9.	255.255.255.0	129.129.1.100	photo.html
10.	255.0.0.0	190.190.1.100	abc.html
11.	255.255.0.0	193.193.1.100	index.html
12.	255.255.255.0	220.220.1.100	photo.html
13.	255.0.0.0	10.0.1.100	abc.html
14.	255.255.0.0	120.120.1.100	index.html
15.	255.255.255.0	129.129.1.100	photo.html
16.	255.0.0.0	190.190.1.100	abc.html
17.	255.255.0.0	193.193.1.100	index.html
18.	255.255.255.0	220.220.1.100	photo.html
19.	255.0.0.0	10.0.1.100	abc.html
20.	255.255.0.0	120.120.1.100	index.html

Таблица 2 — Варианты индивидуальных заданий к заданию № 2 «Устройства генерирования случайных чисел»

№ варианта	Минимум	Максимум	Длина выборки	№ варианта	Минимум	Максимум	Длина выборки
1.	34	96	11	11.	32	61	19
2.	32	55	12	12.	48	51	12
3.	40	52	14	13.	49	83	16
4.	28	66	19	14.	28	75	17
5.	4	85	10	15.	30	87	18
6.	50	53	14	16.	17	70	18
7.	31	78	18	17.	36	64	19
8.	4	66	17	18.	4	91	15
9.	4	85	13	19.	24	92	11
10.	27	73	11	20.	12	66	16

Таблица 3 — Варианты индивидуальных заданий к заданию № 3 «Функция ioctl()»

№ варианта	Идентификатор функции	Номер функции
1.	CDROMPAUSE	0x00005301
2.	CDROMRESUME	0x00005302
3.	CDROMPLAYMSF	0x00005303
4.	CDROMPLAYTRKIND	0x00005304
5.	CDROMREADTOCHDR	0x00005305
6.	CDROMREADTOCENTTY	0x00005306
7.	CDROMSTOP	0x00005307
8.	CDROMSTART	0x00005308
9.	CDROMEJECT	0x00005309

Таблица 4 — Варианты индивидуальных заданий к заданию № 4 «Извлечение информации из файловой системы /proc»

№ варианта	Поле
1.	Номер процессора
2.	Производитель
3.	Семейство
4.	Модель
5.	Модификация
6.	Установленные флаги
7.	Доступные функции
8.	Частота
9.	Размер кэш-памяти
10.	Ошибка деления
11.	Ошибка останова
12.	Наличие FPU
13.	Уровень CPUID
14.	Скорость в поглощающем цикле

7. Контрольные вопросы

1. Многопользовательские системы.
2. Основные принципы распределения.
3. Коммуникации и технические свойства коммуникаций.
4. Основные элементы коммуникаций.
5. Базовая модель OSI.
6. Классификация сетей.
7. Методы кадрирования информации.
8. Знаковое кодирование.
9. Кодирование с изменением длины.
10. Флаговое кодирование.

11. Формат кадров.
12. Маршрутизация. Методы маршрутизации.
13. Стандарт IEEE 802.1, IEEE 802.3.
14. Стандарт IEEE 802.5
15. Оптоволоконный интерфейс FDDI.
16. Уровень MAC и LLC.
17. Режимы работы модемов.
18. AT-команды.
19. S- регистры модемов.
20. Командный режим работы модемов.
21. Режим передачи данных.
22. Классификация команд современных модемов.
23. Сетевой уровень.
24. Транспортный уровень.
25. Управление информационным потоком в сети.
26. Управление информационным потоком в канале.
27. Широкооконная передача данных.
28. Механизм тайм-аута передачи информации.
29. Управление потоком в сети.
30. Управление потоком ЭВМ - коммуникационный канал.
31. Работа с ФС Novell Netware.
32. Защита ФС в ОСС.
33. Уровень защиты по атрибутам.
34. Управление потоком ЭВМ-ЭВМ.
35. Управление потоком процесс- процесс.
36. Защита от перегрузок и блокировок в сети.
37. Сеансы, характеристики сеансов.
38. Интерактивный режим передачи.
39. Коммуникации каналов, передача с промежуточным накоплением.
40. Датаграммный режим передачи.
41. Сеансовый режим передачи.
42. Протоколы среднего уровня.
43. Протокол FDDI.
44. Протокол IPX/SPX.
45. Протокол CSDA/DS.
46. Протоколы высокого уровня.
47. Сеансовый уровень.
48. Уровень представления.
49. Прикладной уровень.
50. Сетевые адаптеры. Классификация. Характеристики.
51. Типы сетевых ОС.
52. OSI.
53. Конфигурация системы.
54. Средства обеспечения безопасности.
55. Сетевые утилиты.

56. Утилиты ФС.
57. Средства обеспечения производительности ОС.
58. Разделение ресурсов в сети.
59. Уровень защиты по правам.
60. Уровень защиты по регистрации и паролю.
61. Архитектура распределенной сети.
62. Основные характеристики сети передачи данных.
63. Оценка средней задержки сообщения в сети передачи данных.
64. Синтез сети передачи данных по критерию минимальной средней задержки.
65. Топология «полный граф».
66. Структура «звезда».
67. Топология типа «кольца».
68. Топология «решетка» («сетка»).
69. Системные характеристики в Linux.
70. Программные характеристики Linux.
71. Распространение программных продуктов.
72. Требования к оборудованию.
73. Базовые концепции.
74. Установка ОС Linux.
75. Сети TCP/IP, UUCP.
76. Межсетевые протоколы.
77. Протоколы передачи данных.
78. Пользовательские протоколы дейтаграмм.
79. Библиотеки сокетов.
80. Понятие порта.
81. Безопасность системы.
82. Сеть на базе TCP/IP.
83. Сетевые интерфейсы, адреса, маршрутизация.
84. Протокол управляющего сообщения.
85. Конфигурация ядра, драйвера PLIP,PPP, SLIP.
86. Настройка последовательных каналов связи, настройка сети TCP/IP.
87. Служба имен и конфигурация резольвера.
88. Протокол PPP.
89. Брандмауэры TCP/IP.
90. Учет трафика.
91. Маскировка IP.
92. Преобразование сетевых адресов.
93. Основные сетевые приложения.
94. Сетевая информационная система.
95. Сетевая файловая система. IPX, NCP
96. Управление TAYLOR UUCP.
97. Электронная почта, Sendmail, электронные новости.
98. Протокол NNTP и демон nntpd.
99. Работа с файловой системой LINUX.

100. Диспетчеризация выполнения процессов.
101. Настройка прокси-сервера.
102. Настройка DNS на машинах под управлением ОС LINUX.
103. Настройка маршрутизатора на базе ОС LINUX.
104. Создание профилей пользователей и групп пользователей.

Библиографический список использованной литературы

1. Алексеев В.Е. Введение в ОС LINUX / В.Е. Алексеев. — М.: Высшая школа, 1984. — 137 с.
2. Фигурнов В.Э. IBM PC для пользователя / В.Э. Фигурнов. — М.: «ФиС», 1991. — 240 с.
3. Брябрин В.М. Программное сетевое обеспечение / В.М. Брябрин. — М: Наука, 1988. — 272 с.
4. Бойко В.В. Проектирование БД под LINUX / В.В. Бойко, В.М. Савинков. — М.: «ФиС», 1989. — 351 с.
5. Хофман П. Internet / П. Хофман. — К.:Диалектика,1996. — 224 с.
6. Фратер Г. Руководство системного администратора LINUX / Г. Фратер. — К.: Диалектика, 1996. — 524 с.
7. Палмер С. UNUX / С. Палмер. — К.: Диалектика, 1997. — 386 с.
8. Дейт К. Введение в системы баз данных / К. Дейт. — М.: Наука, 1980. — 572 с.
9. Каратыгин В.П. Программирование в среде UNIX / В.П. Каратыгин. — К.: Диалектика, 1995. — 426 с.
10. Митчелл М. Программирование для Linux. Профессиональный подход / М. Митчелл, Дж. Оулдем, А. Самьюэл. — М.: Издательский дом «Вильямс», 2003. — 288 с.

Заказ № _____ от «___» _____ 2013 г. Тираж _____ экз.
Изд-во СевНТУ