МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДАЮ
Зав. кафедрой АСУ, профессор

 А.М. Кориков

СОВРЕМЕННЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Тема 6. Развитие сетевой инфраструктуры ОС

Учебно-методическое пособие

для студентов уровня основной образовательной программы: **магистратура**
направление подготовки: **09.04.01 - Информатика и вычислительная техника**

Разработчик
доцент кафедры АСУ

В.Г. Резник

2016

Резник В.Г.

Современные операционные системы. Тема 6. Развитие сетевой инфраструктуры ОС. Учебно-методическое пособие. – Томск, ТУСУР, 2016. – 30 с.

Учебно-методическое пособие предназначено для изучения темы №6 по дисциплине «Современные операционные системы» для студентов уровня основной образовательной программы магистратура направления подготовки: 09.04.01 «Информатика и вычислительная техника».

Оглавление

Введение

Тема 6 является завершающей в данной дисциплине. Современная сетевая архитектура ОС пронизывает не только программное обеспечение самой ОС, но и практически все приложения ЭВМ. Не пытаясь охватить все современные достижения в области сетевых технологий, поскольку эти темы должны изучаться в специализированных курсах, теоретический материал данного пособия рассматривает следующие вопросы:

- Сети TCP/IPv4 и TCP/IPv6.
- Проводные и беспроводные сети.
- Сетевые настройки ОС.
- Назначение и функции NetworkManager (netctl).
- Сетевые настройки systemd.
- Сети и графическая система X11.
- Взаимодействие ЭВМ через сетевой графический интерфейс.

Хорошо видно, что учебный материал данной темы в основном опирается на стек протоколов TCP/IP, но основные теоретические концепции распространяются и на другие протоколы. Такой подход выбран потому, что базовый упор делается на связь сетевых технологий с концепциями управления системными ресурсами ОС и современной технологией взаимодействия процессов.

Лабораторная работа, закрепляющая теоретические знания и обеспечивающая получение ряда практических навыков управления сетевыми ресурсами ОС, построена с учетом особенностей сетевой архитектуры кафедры АСУ.

Сама методика проведения лабораторных работ предполагает использование специального дистрибутива ОС УПК АСУ, которое установлено в компьютерных классах кафедры АСУ.

1 Тема 6. Развитие сетевой архитектуры ОС

1 января 1970 года считается датой рождения ОС *UNIX*.
В 1974 году, корпорация IBM, для своих мэйнфреймов, создала сетевую архитектуру (*SNA*), которая обеспечивала взаимодействие трех типов:

- терминал-терминал;
- терминал-компьютер;
- компьютер-компьютер.

В 1976 году, для ОС UNIX, появляется первое сетевое приложение: программа *UUCP* (*Unix to Unix Copy Program*).
В 1983 году, МО США (*Department of Defense*), в качестве стандарта для своей сети ARPANET, выбирает стек протоколов TCP/IP, а саму сеть разделяет на две части:

- *MILNET* - для военного ведомства США;
- *новую ARPANET*, которую стали называть *Internet*.

В настоящее время, наличие программного обеспечения для поддержки стека протоколов TCP/IP (сети Internet) стало нормой практически для всех ОС.
Именно этому стеку протоколов, с основной проекцией на ОС Linux, посвящен основной материал данной темы.

1.1 Сети TCP/IPv4 и TCP/IPv6

В соответствии с моделью ***DoD*** (*Department of Defense*), сеть ARPANET должна представлять *стек протоколов TCP/IP*, включающая в себя четыре уровня:

- прикладной уровень (*Application layer*);
- транспортный уровень (*Transport layer*);
- сетевой уровень (*Internet layer*);
- канальный уровень (*Link layer*).

TCP (*Transmision Control Protocol*) — протокол *синхронного* (с соединением) взаимодействия ЭВМ (хостов) на транспортном уровне, отвечающий за надежную передачу данных между двумя *портами*.
Порт — логическая точка взаимодействия на транспортном уровне, обычно ассоциированная с приложением ОС.

IP (*Internet Protocol*) — протокол *асинхронного* (без соединения) взаимодействия ЭВМ (хостов) на сетевом уровне, отвечающий за адресацию сетей и хостов.

Замечание
На транспортном уровне модели DoD присутствует также протокол ***UDP***.
UDP (*User Datagram Protocol*) — протокол *асинхронной* (без соединения) передачи данных на транспортном уровне между двумя *портами* хостов.

Стек протоколов TCP/IP — иерархически организованный набор сетевых протоколов, в котором:

- протоколы ***TCP*** и ***IP*** объявляются как главные (базовые);
- протоколы верхнего уровня, в модели ***DoD***, инкапсулируют протоколы нижних уровней, как показано на рисунке 1.1 для протокола ***UDP***.

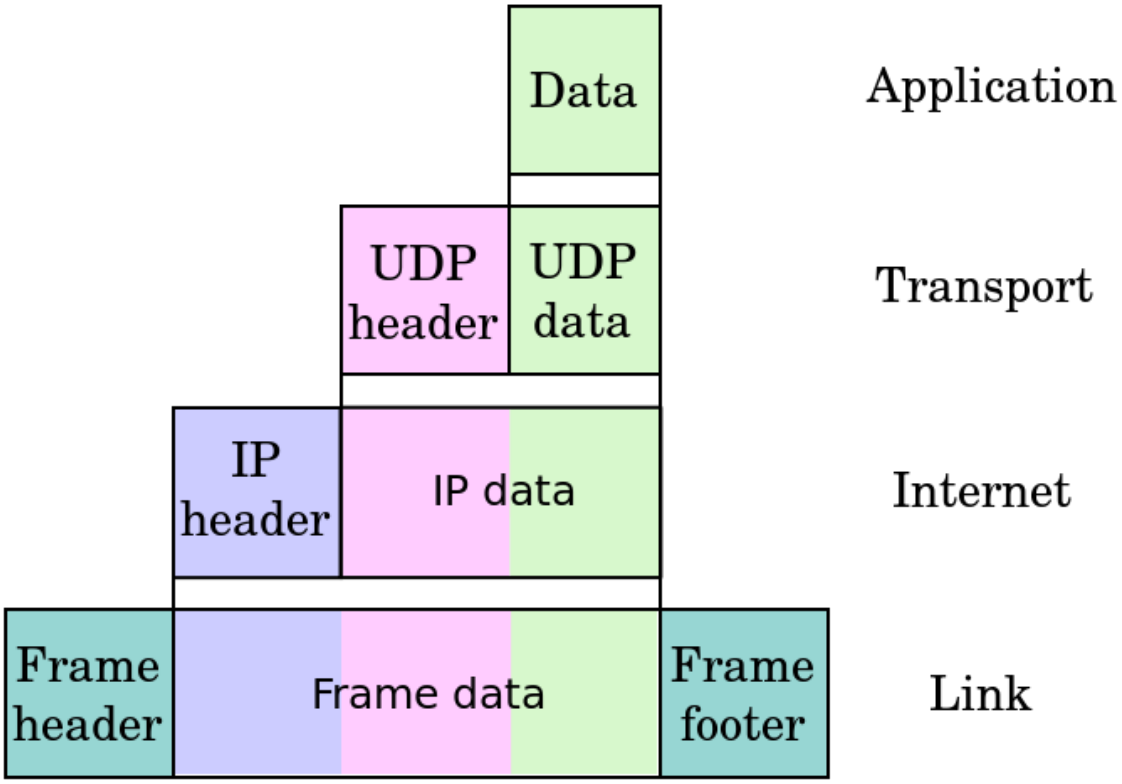

Рисунок 1.1 — Инкапсуляция протоколов по уровням в модели DoD, на примере протокола UDP [рисунок станицы Википедии: https://ru.wikipedia.org/wiki/Инкапсуляция_(компьютерные_сети)]

Современная модель стека протоколов TCP/IP разделена, на сетевом уровне, на две группы:

- ***IPv4*** — классическая модель протокола сетевого уровня;
- ***IPv6*** — расширенная модель протокола сетевого уровня.

1.1.1 Стек протоколов TCP/IPv4

Стек протоколов TCP/IP версии 4 был стандартизирован ***в 1981 году***, в виде документов, называемых ***RFC*** (Request For Comment).
Полный переход сети ARPANET на новые протоколы был завершен ***в 1982 году***.
В 1984 году, международная стандартизирующая организация (***ISO***) создала *модель взаимодействия открытых систем* (***OSI***, *Open System Interconnection*), которая стала базовым средством описания различных стеков протоколов.

На рисунке 1.2, наглядно представлено все многообразие протоколов стека, сфор-

мированного на базе ***IPv4*** и отраженного на модели ***DoD*** и ***OSI***.

	TCP/IP	OSI	
7	**Прикладной**	**Прикладной**	напр., HTTP, SMTP, SNMP, FTP, Telnet, SSH, SCP, SMB, NFS, RTSP, BGP
6		**Представления**	напр., XDR, AFP, TLS, SSL
5		**Сеансовый**	напр., ISO 8327 / CCITT X.225, RPC, NetBIOS, PPTP, L2TP, ASP
4	***Транспортный***	**Транспортный**	напр., TCP, UDP, SCTP, SPX, ATP, DCCP, GRE
3	***Сетевой***	**Сетевой**	напр., IP, ICMP, IGMP, CLNP, OSPF, RIP, IPX, DDP, ARP
2	***Канальный***	**Канальный**	напр., Ethernet, Token ring, HDLC, PPP, X.25, Frame relay, ISDN, ATM, SPB, MPLS
1		**Физический**	напр., электрические провода, радиосвязь, волоконно-оптические провода, инфракрасное излучение

Рисунок 1.2 — Распределение протоколов по уровням моделей OSI и DoD [рисунок станицы Википедии: https://ru.wikipedia.org/wiki/TCP/IP]

Вся классическая сетевая инфраструктура ОС сформирована и построена на основе свойств протокола ***IPv4***:

- ***датаграммы*** передаваемых пакетов содержат *заголовок* и *данные*, суммарный размер которых не превышает 65536 байт;
- ***заголовок*** IP-датаграммы, минимальным размером 20 байт, содержит *адрес источника* и *адрес получателя*;
- ***адреса*** IP-датаграммы, размером 32 бита, включают *адрес сети* и *адрес хоста*, определяемые *битовой маской*;
- ***битовая маска*** состоит из 1, в старших битах адреса: определяет адрес сети, и 0, в младших битах адреса: определяет адрес хоста;
- ***поддержка адресации*** осуществляется двумя типами протоколов: *IGP* и *EGP*;

- ***IGP*** — внутренний протокол маршрутизации, используемый внутри *автономной системы*;
- ***EGP*** — внешний протокол маршрутизации, используемый между *автономными системами*;
- ***автономная система*** — сеть или система сетей, имеющая единую административную и общую маршрутную политику;
- ***имеются технологии*** *классификации адресов*, *локализации адресов* и *переадресации адресов*;
- ***классификация адресов*** представлена пятью классами типов сетей: A, B, C, D и E;
- ***локализация адресов*** — часть адресов, выделенная в каждом классе, которые не маршрутизируются протоколами ***EGP***;
- ***переадресация адресов*** — ***NAT*** (*Network Address Translation*) — трансляция локальных адресов во внешние и обратно, используемая для выхода во внешние сети из локальных сетей;
- ***имеется служба*** именования адресов, представленная набором *DNS-серверов* и организацией *ICANN*;
- ***DNS-сервер — (Domain Name Server)*** — специальный сервер сети Internet, который по имени адреса возвращает его цифровое значение;
- ***ICANN*** (***Internet*** *for* ***Assigned Names*** *and* ***Numbers***) - международная некоммерческая организация, созданная ***18 сентября 1998 года*** для регулирования вопросов, связанных с доменными именами, IP-адресами и другими аспектами функционирования Internet.

1.1.2 Стек протоколов TCP/IPv6

В 1996 году появляется протокол *IPv6*, разработанный открытым международным сообществом *IETF*.
IPv6 - (*Internet Protocol version 6*) — новая версия протокола ***IP***, призванная решить проблемы предыдущей версии (***IPv4***), за счет использования длины адреса ***128 бит*** вместо ***32 бит***.
IETF (*Internet Engineering Task Force*) - **Инженерный совет Интернета** — открытое международное сообщество проектировщиков, учёных, сетевых операторов и провайдеров, созданное ***в 1986 году*** и занимающееся развитием протоколов и архитектуры сети Internet.

Цель проекта — обеспечить адресацией Internet-сообщество, проблемы с которым ожидались в ближайшие десятилетия.
8 июня 2011 года состоялся «*Международный день IPv6*» - мероприятие по тестированию готовности мирового Internet-сообщества к переходу с ***IPv4*** на ***IPv6***.
6 июня 2012 года состоялся «*Всемирный запуск IPv6*» - тестирование ***IPv6*** признано успешным.
Протокол ***IPv6*** имеет значительные улучшения по сравнению с протоколом ***IPv4***, что хорошо показано на рисунке 1.3.

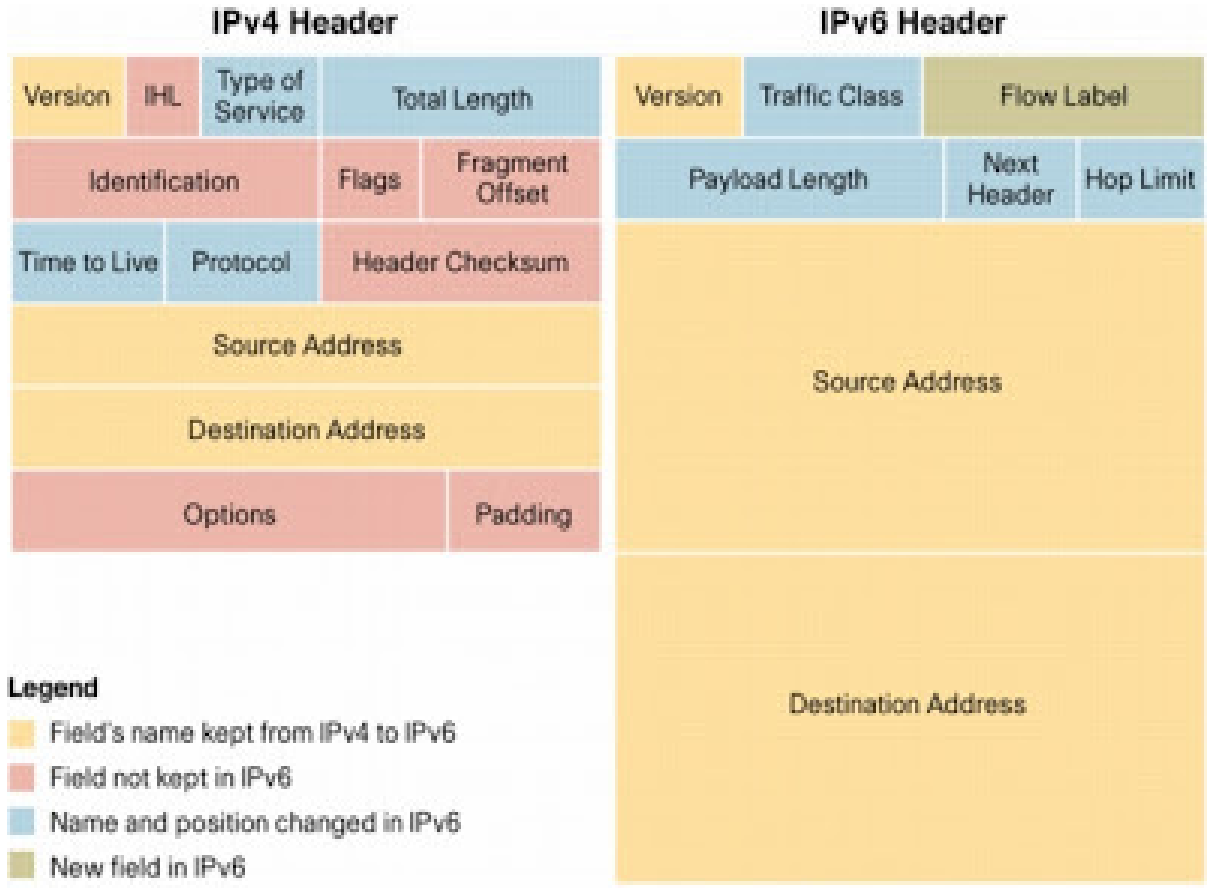

Рисунок 1.3 — Заголовки протоколов IPv4 и IPv6 [рисунок сайта: https://habrahabr.ru/post/253803/]

Длина заголовка IPv4 — от *20* до *60* байт, в зависимости от наличия опций.
Длина заголовка IPv6 — ровно *40* и имеет гораздо меньше полей, упрощающих его использование и маршрутизацию:

- удалено поле ***Fragment Offset***, означающее, что маршрутизаторы теперь не должны фрагментировать пакеты, а использовать технологию *Path MTU discovery*;
- удалено поле ***Header Checksum***, означающее, что маршрутизаторы теперь не должны вычислять контрольные суммы заголовков пакетов, а использовать контрольные суммы протоколов нижних уровней.

MTU (*Maximum Transmission Unit*) — максимальный размер *полезного блока данных* одного пакета, который может быть передан без фрагментации.
Минимальный MTU — минимальный размер полезного блока данных передающего устройства, который сейчас поднят ***до 1280 байт***.
Path MTU discovery — технология согласования размеров передаваемых пакетов данных, разработанная ***в 1988 году*** (*RFC 1191*) и предполагающая передачу данных между взаимодействующими сторонами без промежуточной фрагментации.

На рисунке 1.4 показан один из способов определения MTU ЭВМ:

- MTU интерфейса *loopback*: lo = 65536 байт;

- MTU интерфейса *ethernet-карты*: enp0s25 = 1500 байт;

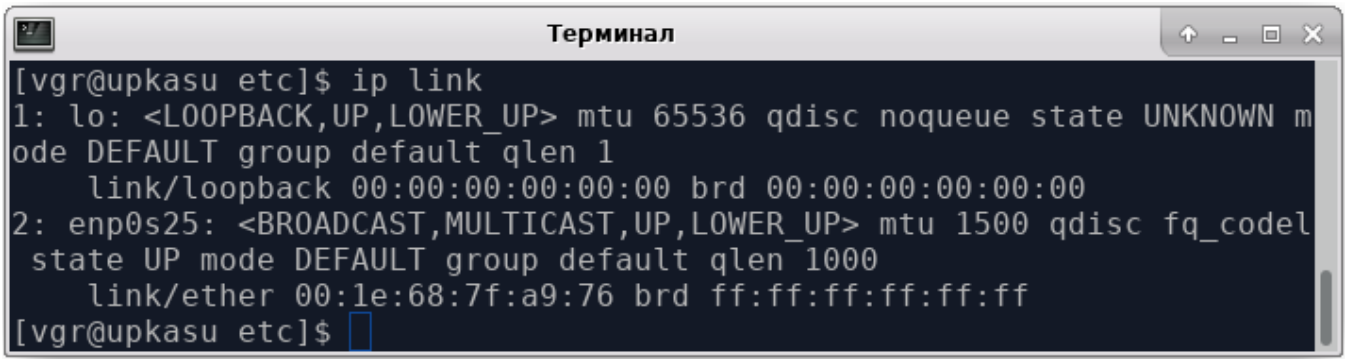

Рисунок 1.4 — MTU сетевых интерфейсов lo и enp0s25

Для представления адреса, протокол ***IPv6*** использует специальную нотацию, предполагающую:

- *представление* 16-байтового адреса с помощью 8 групп, по 2 байта каждая, и *запись* их 16-ричными цифрами, разделенными символом «двоеточия», например:

```
2001:0db8:11a3:09d7:0:0:07a0:765d
```

или

```
2001:0db8:11a3:09d7::07a0:765d
```

- *указание* цифрового ***URL***-адреса — в квадратных скобках, нпример:

```
http://[2001:0db8:11a3:09d7::07a0:765d]:8080/
```

причем, *пропуск нулей в адресе может использоваться только один раз* — по причине сохранения однозначности записи;

- *в специальных записях* (в конфигурационных файлах и других) используется указание длины префикса в битах, разделяющих адрес сетей (подсетей) от адреса хостов, например:

```
2001:0db8:11a3:09d7::07a0:765d/64
```

означает, что первые 8 байт — адресуют сеть, а остальные 8 байт — хост;

- *выделен набор зарезервированных адресов*, представленных в таблице 1.1, которые имеют специальное предназначение — аналогичное протоколу ***IPv4***.

Таблица 1.1 - Зарезервированные адреса IPv6

IPv6 адрес/длина префикса (бит)	***Описание***	***Заметки***
::/128	—	адрес 0.0.0.0 в IPv4
::1/128	Адрес loopback	адрес 127.0.0.1 в IPv4
::ffff:xx.xx.xx.xx/96	Адрес Ipv4 для отображения на IPv6	Нижние 32 бита адреса IPv4 для хостов, не поддерживающих IPv6.
64:ff9b::/96	NAT64	Зарезервирован для доступа из подсети IPv6 к публичной сети IPv4 через механизм трансляции NAT64
2001::/32	Teredo	Зарезервирован для туннелей Teredo в (RFC 4380)
2001:db8::/32	Документирование	Зарезервирован для примеров в документации в (RFC 3849)
2002::/16	6to4	Зарезервирован для туннелей 6to4 в (RFC 3056)
fe80:: — febf::/10	link-local	Аналог 169.254.0.0/16 в IPv4
fc00::/7	Unique Local Unicast	Аналог локальных сетей (RFC 4193)
ff00::/8	multicast	

Подробное изучение протоколов ***IPv4*** и ***IPv6*** выходит за рамки нашего курса, тем более, что они достаточно хорошо описаны в сети Internet и студент может самостоятельно начать их изучать, например, начиная с адресов:

- https://ru.wikipedia.org/wiki/IPv4: IPv4 — Википедия;
- https://ru.wikipedia.org/wiki/IPv6: IPv6 — Википедия;
- https://habrahabr.ru/post/253803/: IPv6 — это весело. Часть 1.

1.2 Проводные и беспроводные сети

С точки зрения современной архитектуры ОС, нет особого различия между *проводными* и *беспроводными сетями*:

- любая сеть реализуется через конкретный интерфейс, который виден в пространстве пользователя как *символьное устройство* (сетевое устройство);
- работа с сетевым устройством обеспечивается *соответствующим драйвером*, который реализует как интерфейс символьного устройства, так и протоколы канального уровня, поддерживаемые ОС на уровне ядра.

С другой стороны, историческое развитие аппаратной части сетей, а также сетевого ПО ОС, до сих пор выделяет различные *сетевые технологии* как в плане их администрирования, так и в планах их практического использования.

Первыми прототипами сетевых устройств стали последовательные устройства, такие как COM-порты для ЭВМ типа IBM PC. Эти устройства известны как */dev/ttySN* или ***serial tty***. Они, собственно говоря, и являются прототипами виртуальных терминалов */dev/ttyN*.
В 1984 году стал широко применяться протокол *SLIP*, который обеспечивал соединение двух компьютеров, через последовательные COM-порты, на уровне протоколов ***TCP/IP***.
Затем, был реализован протокол канального уровня *PPP*, поверх которого можно было реализовывать полноценные сети ***TCP/IP*** на основе последовательных каналов связи.
SLIP (***Serial Line Internet Protocol***) — устаревший сетевой протокол канального уровня для доступа к сетям стека TCP/IP через низкоскоростные линии связи путем простой инкапсуляции IP-пакетов.
PPP (***Point-to-Point Protocol***) — двухточечный протокол канального уровня (***Data Link***), в классификации сетевой модели OSI.

В 1973 году, был разработан прототип протокола ***Ethernet***, обеспечивающий передачу данных по коаксикальному кабелю со скоростью *2.94 Мбит/с*.
В 1982 году, появляется протокол ***Ethernet II***, обеспечивающий передачу данных по коаксикальному кабелю со скоростью *10 Мбит/с*.
В 1983 году, начинается серия стандартов ***IEEE 802.3*** появляется «***Толстый Ethernet***», обеспечивающий скорость *10 Мбит/с*.
В 1987 году – стандарт ***802.3d*** на ***FOIRL*** (*Fiber-Optic Inter-Repeater Link*), волоконно-оптическая связь между повторителями.
В 1990 году – стандарт ***802.3i*** на *10 Мбит/с по витой паре* и так далее.

Собственно, это направление и стало называться *проводными сетевыми интерфейсами* (***Wire Interfaces***).
В ядре ОС, такие устройства стали обозначаться как */dev/ethN*.
Для них стали разрабатываться свои утилиты и другой soft, обеспечивающий настройку и использование проводных сетей.

С 1997 года, начинается серия ***IEEE 802.11*** — набор стандартов связи для коммуникации *в беспроводной локальной сетевой зоне* для частотных диапазонов 0,9; 2,4; 3,6 и 5 ГГц (***Wireless Interfaces***).
Для них стали разрабатываться свои технологии широко известные как ***Bluetooth*** («Синий зуб»), ***Wi-Fi*** (*Hi-Fi*, *High Fidelity* — высокая точность) и другие.
В ядре ОС, такие устройства стали обозначаться как */dev/wlanN*.
Для того, чтобы не создавать проблем ОС при работе в сети, для них также стали разрабатывать свои утилиты и сопутствующий им soft.

Замечание

Ряд устройств, например, устройства мыши могут использовать безпроводное соединение с ЭВМ, но они, в ядре ОС, не рассматриваются как сетевые устройства.

1.3 Сетевые настройки ОС

Сетевая настройка современных ОС — достаточно разнообразна:

- *настройки* BIOS и UEFI для загрузки из сети;
- *определение сетевых устройств* ЭВМ после загрузки ядра ОС;
- *загрузка элементов ОС* на стадии монтирования корневой файловой системы (работа с временной файловой системой *initrd*);
- *формирование сетевой инфраструктуры* для многопользовательского режима работы ОС;
- *конфигурирование сетевых приложений* ОС и ее сетевой графической системы.

В данном подразделе рассматриваются два вопроса:

- *набор стандартных файлов конфигурации*, которые так или иначе присутствуют во всех ОС, поскольку отражают общую сетевую архитектуру стека протоколов TCP/IP;
- *краткий обзор средств управления* сетевыми настройками современных ОС, часть из которых рассматривается более подробно в следующих подразделах данного пособия.

1.3.1 Стандартные файлы конфигурации

Основной перечень файлов конфигурации сетевой инфраструктуры ОС представлен в таблице 1.2.

Таблица 1.2 — Основные файлы конфигурации сети ОС

Файл	***Функция***
/etc/hosts	Связывает *хост-имена* с их IP-адресами
/etc/networks	Связывает *доменные имена* с адресами сетей (*необязательный*)
/etc/hostname	Содержит *хост-имя* вашей системы
/etc/host.conf	*Опции* конфигурирования
/etc/resolv.conf	Содержит *список серверов* доменных имен

Файл ***hosts***, содержимое которого показано на рисунке 1.5, в каждой строке определяет соответствие некоторого *цифрового адреса* и *имени хоста*, которое может использоваться сетевыми программами, когда они обращаются к ЭВМ по именным адресам.
Фактически, файл ***hosts*** ранее использовался как простейший *сервер имен* (***DNS***-сервер), возвращающий сетевому ПО, использующему сетевые имена, цифровой адрес.
Каждый раз, когда программа использует сетевое имя, то сначала просматривается файл */etc/hosts* и, если нужное имя отсутствует, то обращение идет к DNS-серверам, адреса которых указаны в файле */etc/resolv.conf*.

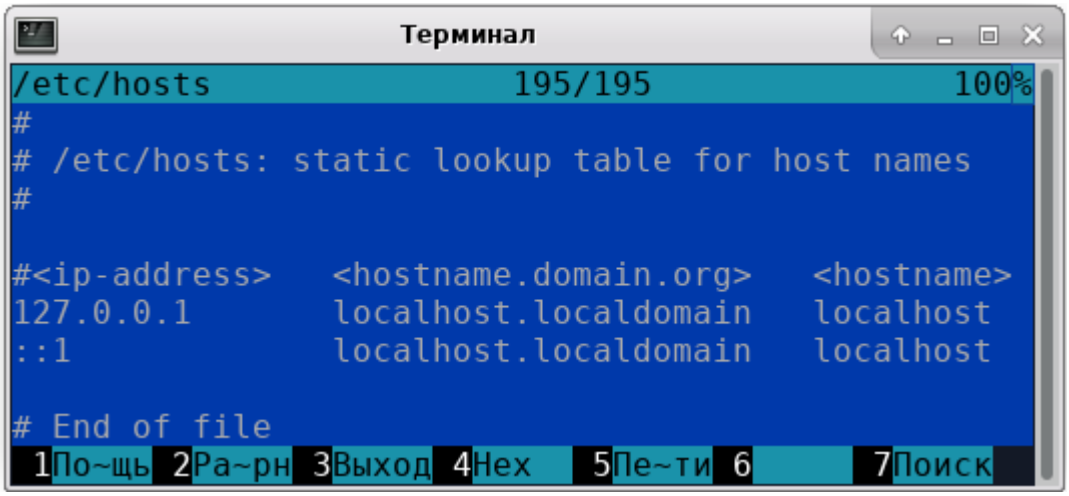

Рисунок 1.5 — Содержимое файла /etc/hosts

Как видно из рисунка 1.5, файл ***hosts*** содержит только стандартные настройки, соответствующие интерфейсу *loopback*, причем эти настройки сделаны как для протокола ***IPv4***, так и для протокола ***IPv6***.

Файл ***networks*** содержит имена сетей и соответсвующие им адреса, например:

```
    loopback 127.0.0.0
    trek.com 199.35.209.0
```

Замечание

В современных ОС, файл ***networks*** можно не использовать, поскольку имена сетей или выбираются *по умолчанию* или определяются *автоматически*, с помощью протоколов маршрутизации. Причина этого состоит в том, что во все возрастающем многообразии сетей и их способов адресации практически невозможно, в локальных условиях, адекватно настроить эти имена.

Файл ***hostname*** содержит сетевое *имя хоста* и является именем ЭВМ в сети, так как его идетифицирует загруженная ОС.
Напрямую, это имя не связано только со стеком протокола TCP/IP, но интенсивно используется ядром и другим системным ПО ОС, поэтому, если изменить это имя, то следует перезагрузить ОС.

В файле **host.conf**, показанном на рисунке 1.6, содержатся *опции* программы-определителя имен, уточняющие саму процедуру (алгоритм, последовательность) такого определения.

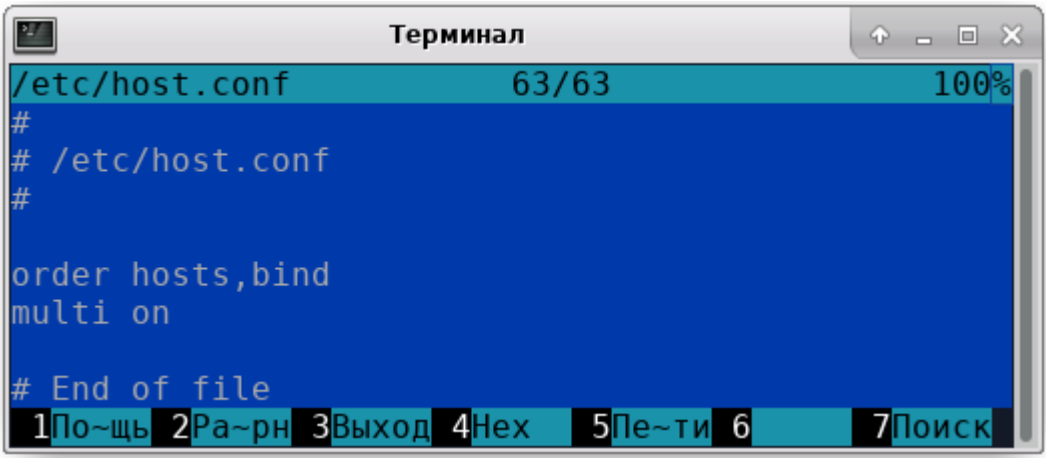

Рисунок 1.6 — Содержимое файла /etc/host.conf

Опции, как показано в таблице 1.3, указывают определителю имен, каким сервисом и как он должен пользоваться.

Таблица 1.3 — Опции определителя имен, доступные в файле host.conf

Опция	***Описание***	
order	*Задает последовательность методов преобразования имен*	
	hosts	Проверяется наличие имени в локальном файле */etc/hosts*
	bind	Запрашивается адрес у сервера имен DNS
	nis	Для получения адреса используется база данных центра сетевой информации (NIS)
	alert	Проверяет наличие в локальной системе адресов удаленных узлов, пытающихся получить к ней доступ; устанавливается и отменяется ключевыми словами **on** и **off**
nospoof	Подтверждает правильность адресов удаленных узлов, пытающихся получить доступ к локальной системе	
trim	Удаляет имя домена из полного имени и проверяет наличие только хост-имени. Позволяет использовать вместо IP-адреса не полное имя *хост.домен.расширение*, а просто хост-имя, указанное в файле **hosts.**	
multi	Позволяет хост-машине иметь несколько IP-адресов в локальном файле **hosts**. Включается и выключается ключевыми словами **on** и **off**	

Файл ***resolv.conf*** содержит перечень имен ***DNS***-серверов, к которым обращается сетевое ПО ЭВМ для разрешения сетевых имен.
В общем случае, как показано в таблице 1.4, файл ***resolv.conf*** может содержать записи трех видов.

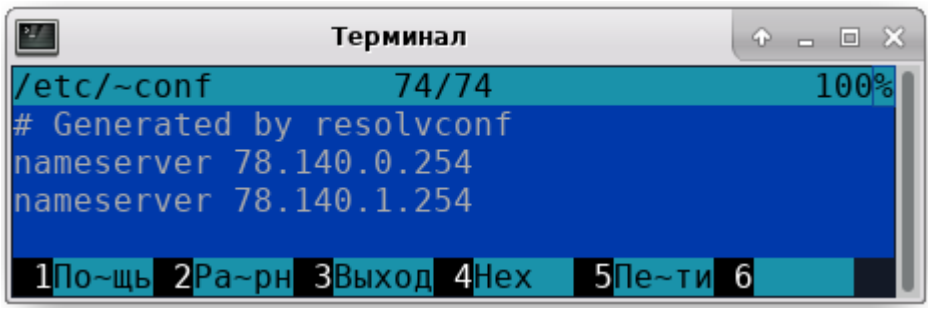

Рисунок 1.7 — Содержимое файла /etc/resolv.conf

Таблица 1.4 — Типы записей файла /etc/resolv.conf

Тип	***Назначение***
domain	Вводится *доменное имя* локальной системы
search	Приводится *список доменов* на тот случай, если задается только хост-имя
nameserver	*Цифровой адрес* ***DNS***-сервера (отдельно на каждый сервер)

Замечание

Дополнительные файлы сетевых настроек зависят от используемого ПО сетевого обеспечения.

1.3.2 Управление сетевыми настройками ОС

Традиционно, для настройки сети и сетевых интерфейсов в системах UNIX, а затем и в системах Linux, использовался набор утилит под общим названием *Net-Tools*.
В этот набор входили такие утилиты как *ifconfig*, *netstat*, *route*, *arp* и ряд других, которые использовали для доступа к ядру ОС набор системных вызовов *ioctl()*.
Указанный набор утилит, по мере развития сетевого ПО, стал широко применяться и в других ОС, хотя в некоторых случаях под другими названиями и модификациями.

Со временем, благодаря идеям и наработкам Алексея Кузнецова стал применяться альтернативный интерфейс, под общим названием *Netlink*.
Netlink — *сокет-ориентированный интерфейс*, широко использующий средства межпроцессного взаимодействия процессов (IPC), вместо системных вызовов *ioctl()*, для целей доступа к сетевому оборудованию через ядро ОС.
На базе этого интерфейса стал разрабатываться новый набор утилит под общим названием *iproute*.
В настоящее время, в ОС Linux применяется набор утилит *iproute2*.

iproute2 — *новый набор утилит* сетевой настройки и управления сетевыми харак-

теристиками ОС, использующая *интерфейс Netlink* для обеспечения современной коммуникации с ядром ОС.

Основу пакета *iproute2* составляет утилита ***ip***, функциональные возможности которой, в сравнении со старыми утилитами, приведены в таблице 1.5.

Таблица 1.5 - Соответствие утилит net-tools и пакета iproute2

Net-tools	***Iproute2***	***Описание***
`ifconfig`	`ip addr` `ip link`	Настройка сетевого адреса, включение и выключение интерфейса
`route`	`ip route`	Управление таблицами маршрутизации
`arp`	`ip neigh`	Управление ***ARP***-кэшем
`iptunnel`	`ip tunnel`	Настройка тоннелей
`nameif`	`ifrename`	Переименование сетевого интерфейса
`ipmaddr`	`ip maddr`	Настройка ***мультикаст***-групп рассылки

`netstat`	`ip -s`
	`ss`
	`ip route`
		Отображение различной сетевой статистики

Замечание

Многие старые утилиты до сих пор широко используются в различных ОС, поэтому их знание также необходимо при работе с системным сетевым ПО.

Кроме указанного выше, широко используются и другие программные средства, среди которых мы выделим *профайловое ПО*, представленное в таблице 1.6 и отдельно рассматриваемое в последующих разделах данного пособия.

Таблица 1.6 — Сетевые инструменты, использующие профайлы

Инструмент	***Описание***
netcfg	**Устаревший** инструмент командной строки, используемый для настройки сетевых подключений через профили.
netctl	**Нативный проект** Arch Linux для настройки сети, использующий интерфейс командной строки для настройки сетевых подключений через профили.
NetworkManager	Программа, облегчающая определение и конфигурацию средств для автоматического подключения к сети. Функционал NetworkManager полезен как для беспроводных, так и проводных сетей.
systemd-networkd	Системный демон, управляющий сетевыми настройками, по мере их появления и обнаружения.

1.4 Назначение и функции NetworkManager (netctl)

Современные ОС Linux широко используют два основных инструмента настройки сетевого ПО ЭВМ: *netctl* и *NetworkManager*:

- **netctl** - инструмент командной строки, используемый для настройки и управления сетевыми подключениями *через профили*; он является нативным проектом для ОС ***Arch Linux***;
- **NetworkManager** - это набор управляющих программ, облегчающих определение и конфигурацию средств для автоматического подключения к сети; был разработан компанией ***Red Hat***, и в настоящее время поддерживается проектом ***GNOME***.

Важная особенность этих проектов — полная совместимость с главным управляющим менеджером ОС Linux — проектом ***systemd***.

Замечание

ПО ***NetworkManager*** поддерживает графический интерфейс, поэтому обычно встраивается в интерфейс рабочих столов ОС.
В ОС УПК АСУ *этот пакет не установлен*, поэтому сосредоточимся на описании ***netctl***.

Инструментальный пакет *netctl* пришел на смену устаревшему пакету *netcfg* по двум основным причинам:

- ***смена парадигмы*** взаимодействия сетевых инструментальных средств с ядром ОС, которая предполагает широкое использование *контейнерных технологий*, основанных на сокетах и поддерживаемых низкоуровневым системным пакетом *iproute2*;
- ***необходимость полноценного взаимодействия*** с *systemd* — новой управляющей системой ОС Linux.

Netctl является инструментом командной строки, основанной на *профайлах*.
Профайл — текстовый файл (конфигурационный файл), поддерживающий специальный язык системы ***netctl***.
Каждый профайл описывает один из типов сетевого подключения — *сетевого интерфейса*.
Сетевой интерфейс — это общее название сетевого взаимодействия, включающее:

- *аппаратное обеспечение* сетевого интерфейса: проводное, безпроводное и другие особенности, характеризующие *тип его реализации*, например, ***Wi-Fi***, ***PPP***, ***Ethernet*** и другие;
- *статическое* или *динамическое* подключение;
- *автоматическое* или *неавтоматическое* подключение.

Для подробного изучения всех особенностей языка описания настроек профайлов пакета netctl следует воспользоваться: ***man netctl*** и ***man netctl.profile***.

Для размещения профайлов выделена специальная директория */etc/netctl*, общее содержимое которой показано в таблице 1.7.

Таблица 1.7 — Общее содержимое директории /etc/netctl

Имя	***Назначение***
имя_профайла	Имя текстового файла, в котором дано описание отдельного сетевого интерфейса. *Не должно содержать* расширений типа: *.action*, *.conf* или *.service*.
exemples	Директория, в которой размещены примеры различных профайлов интерфейсов.
hooks	Директория с запускаемыми сценариями языка bash, которые все выполняются, обрабатывая профили интерфесов, при их вызове.
interfaces	Директория с запускаемыми сценариями языка bash, имена которых совпадают с именами профайлов интерфеса и отдельно запускаются при вызове (обработке) его.

Непосредственное использование инструментария *netctl* состоит в копировании наиболее подходящего профайла из директории */etc/netctl/exemples* в директорию */etc/netctl* и последующее редактирование нужных настроек.
Для примера, на рисунке 1.8 представлено содержимое файла ***ethernet-dhcp***.

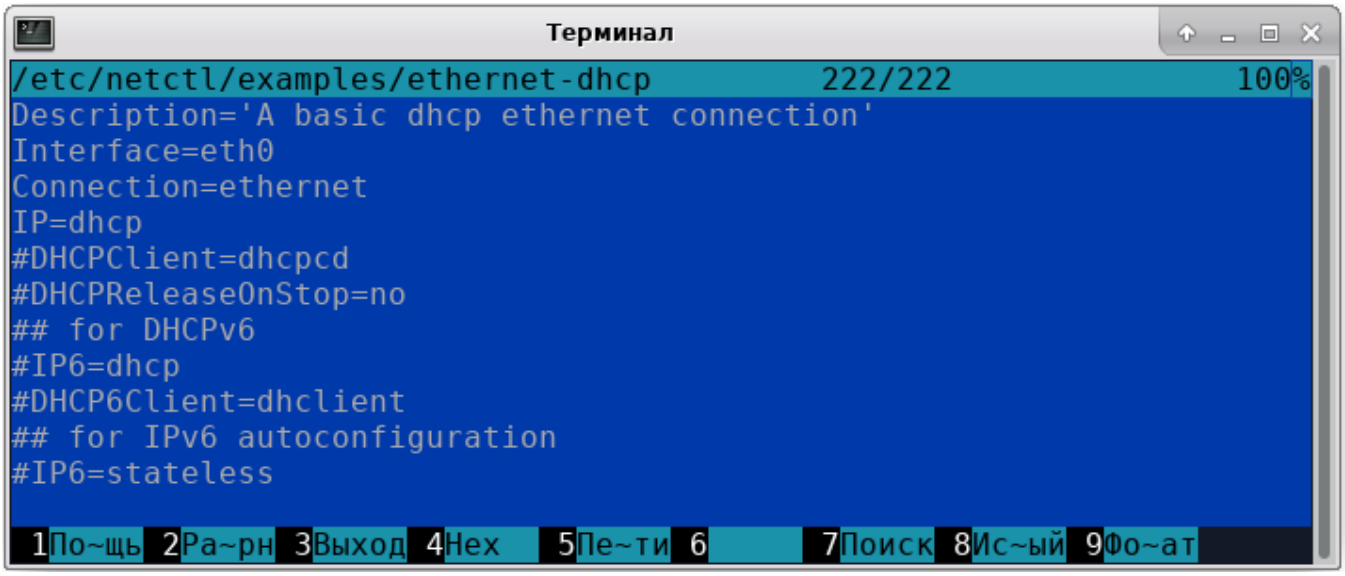

Рисунок 1.8 — Исходное содержимое профайла ethernet-dhcp

Хорошо видно, что в этом файле определены только три главных параметра минимально необходимые для подключения к локальной сети и управляемые сетевым сервисом *dhcp*:

- *Interface=eth0* — сетевой интерфейс, соответствующий первому сетевому устройству ЭВМ;
- *Connection=ethernet* — тип подключения: проводной Ethernet;

- *IP=dhcp* — тип подключения к сети, указывающий, что ЭВМ будет обращаться к ***dhcp***-серверу, расположенному в локальной сети, который должен передать ей необходимые сетевые настройки.

Указанный профайл относится к варианту *динамического подключения* ЭВМ к сети, что предполагает передачу от сервера к ЭВМ минимального состава следующих сетевых настроек:

- *цифрового адреса* ЭВМ, для протокола IPv4;
- *маски* сети;
- *цифрового адреса* основного маршрутизатора;
- *цифрового адреса* хотя бы одного DNS-сервера, обслуживающего сетевые имена хостов.

Если поместить указанный профайл в директорию */etc/netctl*, - это будет основанием для системного ПО использовать данный профайл для настройки ЭВМ.

Замечание

В том виде, как показано на рисунке 1.8, профайл не будет работать правильно. Это обусловлено двумя причинами.

Первая причина состоит в том, что интерфейса *eth0* в современных ОС может и не быть. Это хорошо показано на рисунке 1.9, на котором, с помощью команды ***ip link***, выведен список интерфейсов конкретного компьютера. В данном примере, необходимо интерфейс *eth0* заменить на интерфейс *enp0s25*.

Вторая причина состоит в том, что главному менеджеру процессов ***systemd*** необходимо указать, когда стартовать настройку сети: или сечас или в процессе загрузки ОС. Это делается специальными командами, которые будут рассмотрены в лабораторной работе.

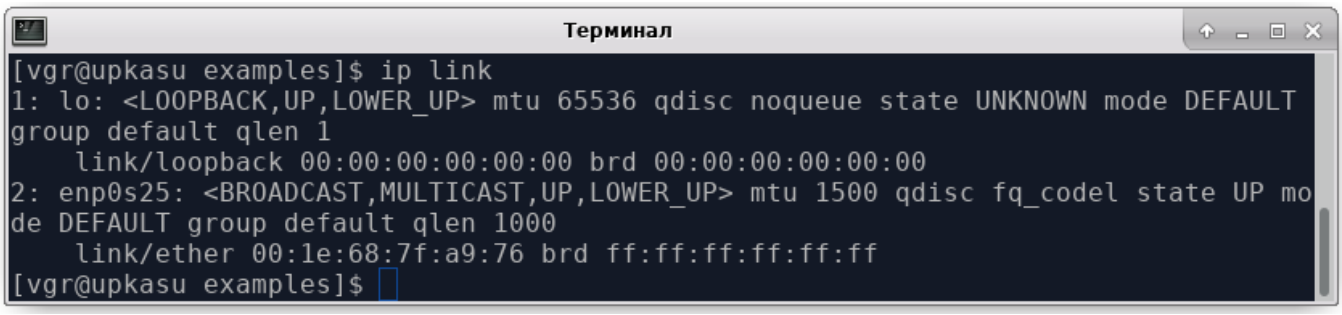

Рисунок 1.9 — Список доступных сетевых интерфейсов конкретного компьютера

1.5 Сетевые настройки systemd

В ноябре 2013 года разработчики *systemd* объявили о создании нового сервиса под названием *systemd-networkd*.
Цель разработки — заменить многобразие различных способов настроек сетевых интерфейсов единим подходом, совместимым с общей идеологией проекта systemd, и обеспечить администраторов ОС общим инструментом, позволяющим внедрять контейнерные технологии.

За нормальную работу сети отвечают два серверных процесса:

- *systemd-networkd* — обнаруживает (естественно с помощью *systemd-udevd*), а затем настраивает и отслеживает сетевые устройства;
- *systemd-resolved* — обеспечивает работу с DNS-серверами.

Файлы конфигурации данного сервиса находятся в директориях:

- */usr/lib/systemd/network* — базовая директория, содержащая файлы конфигурации поставляемые с дистрибутивом;
- */etc/systemd/network* — директория модификации настроек, содержащая файлы конфигурации изменяемые администратором ОС.

Используются следующие типы файлов конфигурации:

- *.link* – описывают физические параметры каждого интерфейса: имя, MAC, MTU и другие;
- *.network* – описывают параметры сети: IP, маршруты, DNS и другие;
- *.netdev* – описывают виртуальные интерфейсы, мосты.

Замечание

Полное изучение интерфеса и возможностей этого сервиса выходит за рамки нашего курса обучения, поэтому ограничимся лишь следующими замечаниями:

- сервис *systemd-networkd* содержит полный набор настроек необходимых для инициации и поддержки сетей различной конфигурации и интерфейсов, но во многих случаях он излишен и может конфликтовать со старыми средствами настройки сетей;
- сервис *systemd-networkd* - достаточно компактный и занимает порядка 2 Мбайт, что выгодно отличает его, например, от пакета *NetworkManager*, который занимает 20 Мбайт;
- сервис *systemd-networkd* не имеет графического интерфеса, поэтому он полезен в первую очередь администраторам серверов и мало удобен для рабочих станций;
- сервис *systemd-networkd* имеет уникальный и во многом сложный синтаксис, что требует его специального изучения;
- сервис *systemd-networkd* полностью совместим с системой управления процессами systemd, что определяет его будущую перспективность.

1.6 Сети и графическая система X11

Как уже было изучено в теме 5, графическая система X11 является сетевой системой, которая позволяет:

- *запустить X-сервер на локальной машине* и перевести виртуальный терминал в графический режим, позволяющий создавать окна для графических приложений и работать с ними;
- *прослушивать сеть* (TCP/IP) и принимать соединения от графических приложений, запущенных на других ЭВМ, обеспечивая на локальной машине работу с ними.

Обычно, при запуске X-сервера предполагается, что:

- *сервер запускается* одновременно с каким-либо графическим приложением, обеспечивающим работу пользователя на конкретном виртуальном терминале, иначе он завершит работу по тайм-ауту;
- *официально*, для внешнего подключения графических приложений используется порт ***6000***;
- *неофициально*, для внешнего подключения графических приложений могут использоватся порты ***6000-6005***, что обязательно нужно отобразить в конфигурационном файле ***/etc/services***.

Специально, для запуска X-сервера и сопутствующих ему приложений используются:

- утилита */bin/xinit*, вариант применения которой показан на рисунке 1.10;
- сценарий */bin/startx*, обеспечивающий учет общих вариантов запуска X-сервера, с учетом других конфигурационных файлов;
- директория */etc/X11*, показанная на рисунке 1.11, в которой содержатся файлы конфигурации для сценария /bin/startx и графической системы X11, в целом.

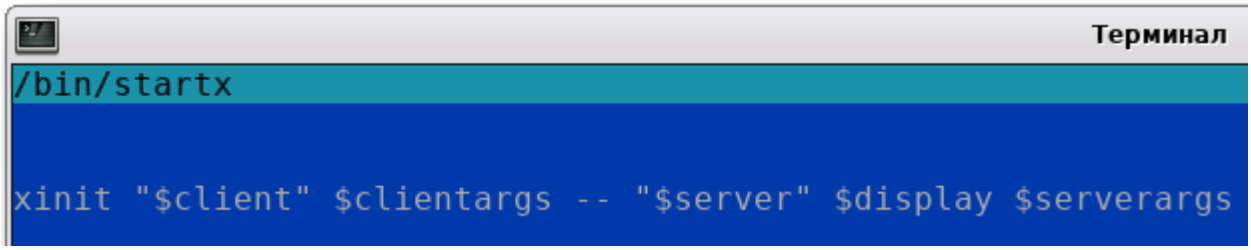

Рисунок 1.10 — Использование утилиты xinit

Аргументы утилиты ***xinit*** разделены на две части:

- *до аргумента «--»*, указывается запуск приложения X-клиента;
- *после аргумента «--»*, указывается запуск приложения X-сервера.

Базовые конфигурационные файлы, как клиента, так и сервера, располагаются в директории ***/etc/X11/xinit/***, показанной на рисунке 1.11.

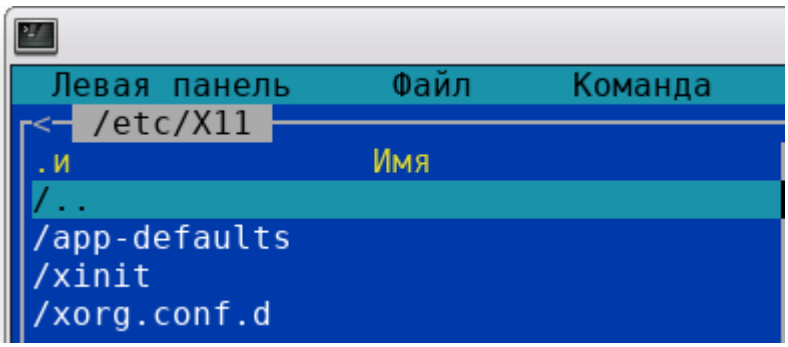

Рисунок 1.11 — Базовая директория конфигурационных файлов системы X11

Содержимое директории ***/etc/X11/xinit/***, показано на рисунке 1.12.
Здесь:

- *xinitrc* — сценарий проверки и запуска графических приложений на локальной машине, в процессе запуска X-сервера;
- *xserverrc* — сценарий непосредственного запуска X-сервера;
- *xinitrc.d* — директория со сценариями, запускаемыми сценарием ***xinitrc***.

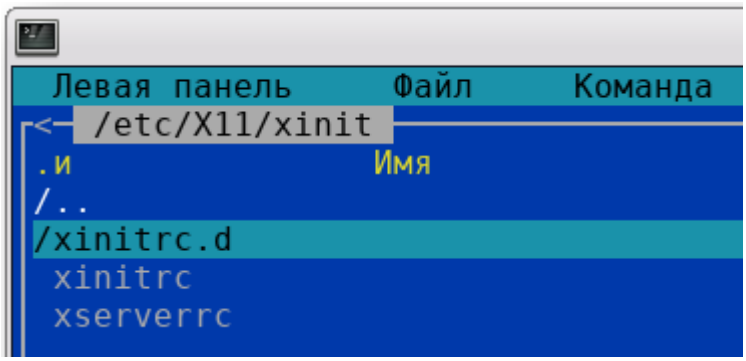

Рисунок 1.12 — Содержимое директории /etc/X11/xinit

На рисунке 1.13 показано содержимое сценария запуска X-сервера.

Рисунок 1.13 — Содержимое сценария запуска X-сервера

Хорошо видно, что:

- *сервер запускается* с ключем ***-nolisten***, что указывает на отсутствие поддержки внешних сетевых соединений;
- *если ключ заменить* на ***-listen***, то сервер будет ожидать соединения на порту, указанному системной переменной ***DISPLAY***;
- *ключевой параметр* ***"$@"*** - содержит последовательность оставшихся опций сервера, включая содержимое переменной ***DISPLAY***.

Чтобы прикладные программы «знали» на каком компьютере и на каком дисплее находится X-сервер, используется переменная окружения ***DISPLAY***, которая содержит информацию в формате:

```
<IP-адрес X-сервера>:<номер дисплея>.<номер экрана дисплея>
```

В таблице 1.8 приведены ряд примеров задания системной переменной ***DISPLAY***.

Таблица 1.8 - Примеры задания значений переменной DISPLAY

Значение переменной DISPLAY	**Номер порта соединения**	**Пояснение**
:0	--	X-сервер на локальном компютере, дисплей №0, экран №0
:1.0	--	X-сервер на локальном компютере, дисплей №1, экран №0
asu.tusur.ru:4	6004	X-сервер на компютере asu.tusur.ru, дисплей №4, экран №0, номер UDP-порта - 6004
192.168.1.17:2	6002	X-сервер на компютере 192.168.1.17, дисплей №0, экран №0, номер UDP-порта - 6002

Многие прикладные программы имеют специальный параметр **-display**, после кото-рого можно указать IP-адрес и номер дисплея вывода.
Классический пример, - запуск программы ***xterm*** в виде:

```
xterm -display foo:1<Enter>
```

выведет консольную программу на дисплей **№1** компьютера **foo**.

Если на компьютере запущено несколько X-серверов, то следует воспользоваться «*горячими клавишами*»:

```
<Ctrl>+<Alt>+<Backspace> - приведет к остановке X-сервера;
<Ctrl>+<Alt>+<F#> - переключение между консолями Linux.
```

Замечание

Конкретный запуск X-серверов зависит от разработчиков конкретного дистрибутива. Например, дистрибутив ОС УПК АСУ запускает X-сервер с помощью специальной программы дисплейного менеджера LightDM.
Со всеми возможностями запуска X-сервера можно познакомиться в руководстве ***man***:

- *man xinit* — описание и параметры использования утилиты ***xinit***;
- *man Xserver* — описание и параметры запуска X-сервера.

1.7 Взаимодействие ЭВМ через сетевой графический интерфейс

Графическая система X Window System позволяет взаимодействовать графическим приложениям различных ЭВМ посредством протокола обмена X11. Для такого взаимодействия необходимо решить две проблемы:

- *на локальном компьютере*, - разрешить X-серверу ***прослушивать*** и ***устанавливать соединение*** с удаленными ЭВМ, что требует прав администратора ОС;
- *на удаленном компьютере*, - запустить графическое приложение, правильно указав ***адрес*** и ***порт*** уже запущенного X-сервера.

Решив указанные проблемы, можно на локальном компьютере работать с приложениями, запущенными на удаленном компьютере.

Первая проблема решается получением прав пользователя *root* и запуска утилиты ***xhost*** в формате:

`xhost [±] адрес_удаленной_ЭВМ …`, где

- *адрес_удаленной_ЭВМ* — IP-адрес в любом формате;
- *«+»* - *разрешает доступ* на соединение X-сервера с удаленной ЭВМ;
- *«-»* - *запрещает доступ* на соединение X-сервера с удаленной ЭВМ.

Вторая проблема связана с правильным запуском графического приложения на удаленной ЭВМ, что может быть решено двумя способами:

- *сообщить оператору* (администратору) удаленной ЭВМ параметры запуска графического приложения, правильно указав содержание системной переменной ***DISPLAY***;
- *установить терминальное соединение*, с локальной машины на удаленную ЭВМ, и самостоятельно запустить графическое приложение с нужными параметрами запуска.

Замечание

Получив терминальный доступ к удаленной ЭВМ, следует:

- или установить правильно системную переменную среды — ***DISPLAY***;
- или воспользоваться ключем ***-display***, если он имеется у приложения.

2 Лабораторная работа №6

Лабораторная работа №6 является последним практическим занятием в курсе «Современные операционные системы».
Цель работы — освоение ряда современных сетевых инструментальных средств, используемых в ОС Linux, на примере дистрибутива ОС УПК АСУ.
Особенности выполнения лабораторной работы обусловлены двумя факторами:

- компьютеры учебного класса 439 ФЭТ подключены к сети ***ethernet*** кафедры АСУ, управляются администраторами этой кафедры и ориентированы на интерфейс ***dhcp***, который контролируется по ***MAC***-адресам сетевых карт;
- личные компьютеры студентов могут быть подключены к сети г. Томска через интерфейсы ***Wi-Fi***, при наличии разрешения на такой доступ.

В целом, лаборторная работа лишь демонстрирует некоторые сетевые возможности современных ОС и не рассчитана на систематическое изучение всех, связанных с такими возможностями, вопросов.

2.1 Настройка сетевого интерфейса ЭВМ

Теоретическое описание сетевых настроек ЭВМ приведено в подразделах 1.3 -1.4 данного руководства.
С помощью руководства ***man*** следует изучить назначение и варианты применения утилит: ***ifconfig***, ***ip***, ***hostname*** и ***ping***.
Перед выполнением настроек, следует удалить все файлы в директории */etc/netctl* (*директории не удалять!*), после чего перезапустить ЭВМ.

2.1.1 Настройка dhcp-интерфейса ethernet

В терминале, с помощью утилит ***ifconfig*** или ***ip***, определить наличие и имя сетевого интерфейса ***ethernet***.
В директории выбрать */etc/netctl/exemples* нужный конфигурационный файл примера, например, *ethernet-dhcp* и скопировать его в директорию */etc/netctl*.
Отредактировать файл *ethernet-dhcp*, указав правильное имя сетевого интерфейса.
Выполнить команду запуска сети:

```
netctl start ethernet-dhcp
```

С помощью утилиты ***hostname*** необходимо определить имя компьютера, а затем: определить ***IP***-адрес и проверить работу сети командой:

```
ping -4 имя-компьютера
```

2.1.2 Настройка Wi-Fi-интерфейса notebook (laptop)

В принципе, настройка Wi-FI-интерфейса проводится также, как и настройка проводной сети, только параметров настройки и их правильная установка выходят за рамки данного курса.

В дистрибутиве имеется утилита ***wifi-menu***, работающая в псевдографике и поддерживаемая соответствующим пакетом ***dialog***. Утилита появилась еще в старом пакете ***netcfg***, но достаточно хорошо работает и в новых условиях.
Использование утилиты осуществляется следующим образом:

- запускается виртуальный терминал, в котором набирется команда перехода в режим администратора:

```
sudo -i
```

- от имени пользователя ***root*** запускается утилита, командой:

```
wifi-menu
```

- дальнейшие действия выполняются в соответствии с диалогом.

2.2 Отображение интерфейса thunar на удаленной ЭВМ

Данная часть лабораторной работы предназначена для демонстрации возможности отображения графического интерфеса приложений на удаленных ЭВМ.
Теоретическая часть работы опирается на учебный материал, изложенный в подразделах 1.6 — 1.7 данного пособия.

Как было отмечено ранее, графические приложения, отображающие свой интерфейс на сервер X11, используют:

- или ключевой параметр ***--display***;
- или значение системного параметра ***DISPLAY***.

В дистрибутиве ОС УПК АСУ, таким приложением является файловый менеджер thunar, аргументы запуска которого можно изучить:

```
man thunar
```

Замечание

Такими же свойствами обладает приложение ***mousepad***. Чтобы убедиться в этом, - запустите в терминале команду:

```
mousepad --help
```

2.2.1 Тестирование запуска на локальной ЭВМ

С помощью утилиты ***hostname*** необходимо определить имя компьютера, а затем: определить ***IP***-адрес и проверить работу сети командой:

```
ping -4 имя-компьютера
```

Разрешить сетевой доступ с локального компьютера, командой:

```
sudo xhost +IP-адрес_локальной_ЭВМ
```

С помощью комбинации клавиш ***Ctrl-Alt-F3*** — перейти на виртуальный терминал */dev/tty3*, на котором — зайти пользователем ***asu***. Затем, нужно выполнить команду:

```
thunar --display=IP-адрес_локальной_ЭВМ
```

С помощью комбинации клавиш ***Alt-F7*** — вернуться на рабочий стол графического режима работы и оценить полученный результат.

2.2.2 Тестирование запуска на удаленной ЭВМ

Данная часть работы выполняется в паре с удаленным пользователем.

Узнать у удаленного пользователя ***IP***-адрес его ЭВМ и проверить ее доступность в сети командой:

```
ping -4 IP-адрес_удаленной_ЭВМ
```

Разрешить сетевой доступ с удаленного компьютера, командой:

```
sudo xhost +IP-адрес_удаленной_ЭВМ
```

Попросить удаленного пользователя выполнить в терминале команду:

```
thunar --display=IP-адрес_вашей_локальной_ЭВМ
```

Оценить полученный результат.
Результаты выполненных работ отразить в личном отчете.

Список использованных источников

1 Резник В.Г. Современные операционные системы. Самостоятельная и индивидуальная работа студента. Учебно-методическое пособие. – Томск, ТУСУР, 2016. – 15 с.

2 Гордеев А.В. Операционные системы: учебное пособие для вузов. ─ Спб.: Питер, 2004. ─ 415с.

3 Таненбаум Э. Современные операционные системы. - СПб.: Питер, 2007. - 1037с.

4 Резник В.Г. Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux. Учебно-методическое пособие. – Томск, ТУСУР, 2017. – 38 с.

5 Резник В. Г., Операционные системы: Учебное пособие для студентов направления 09.03.01, «Информатика и вычислительная техника» [Электронный ресурс] / Резник В. Г. — Томск: ТУСУР, 2016. — 183 с. — Режим доступа: https://edu.tusur.ru/publications/6261.

6 Резник В. Г. Операционные системы. Часть 2: Учебное пособие для студентов направления 09.03.01, «Информатика и вычислительная техника» [Электронный ресурс] / Резник В. Г. — Томск: ТУСУР, 2016. — 216 с. — Режим доступа: https://edu.tusur.ru/publications/6262.

Учебное издание

Резник Виталий Григорьевич

СОВРЕМЕННЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Учебно-методическое пособие предназначено для изучения темы №6 по дисциплине «Современные операционные системы» для студентов уровня основной образовательной программы магистратура направления подготовки: 09.04.01 «Информатика и вычислительная техника».

Учебно-методическое пособие

Усл. печ. л. . Тираж . Заказ .
Томский государственный университет
систем управления и радиоэлектроники
634050, г. Томск, пр. Ленина, 40