Операционная система Linux

Основные характеристики ОС Linux

- Реальная многозадачность.
- Многопользовательский доступ.
- Свопирование оперативной памяти на диск.
- Страничная организация памяти.
- Загрузка выполняемых модулей "по требованию".
- Совместное использование исполняемых программ.
- Общие библиотеки.
- Возможность запуска исполняемых файлов других ОС.
- Поддержка различных форматов файловых систем.
- Сетевые возможности.
- Работа на разных аппаратных платформах.
- 100%-ное соответствие стандарту POSIX 1003.1. Частичная поддержка возможностей System V и BSD.
- System V IPC.

Основные характеристики ОС Linux

- В силу того, что исходные коды Linux распространяются свободно и общедоступны, к развитию системы с самого начала подключилось большое число независимых разработчиков. Благодаря этому на сегодняшний момент Linux - самая современная, устойчивая и быстроразвивающаяся система, почти мгновенно вбирающая в себя самые последние технологические новшества. Она обладает всеми возможностями, которые присущи современным полнофункциональным операционным системам типа UNIX. Приведем краткий список этих возможностей.

Реальная многозадачность

- Все процессы независимы; ни один из них не должен мешать выполнению других задач. Для этого ядро осуществляет режим разделения времени центрального процессора, поочередно выделяя каждому процессу интервалы времени для выполнения. Это существенно отличается от режима "вытесняющей многозадачности", реализованной в Windows 95, когда процесс должен сам "уступить" процессор другим процессам (и может сильно задержать их выполнение).

Многопользовательский доступ

- Linux - не только многозадачная ОС, она поддерживает возможность одновременной работы многих пользователей. При этом Linux может предоставлять все системные ресурсы пользователям, работающим с хостом через различные удаленные терминалы.

Свопирование оперативной памяти на диск

- Свопирование оперативной памяти на диск позволяет работать при ограниченном объеме физической оперативной памяти; для этого содержимое некоторых частей (страниц) оперативной памяти записываются в выделенную область на жестком диске, которая трактуется как дополнительная оперативная память. Это несколько снижает скорость работы, но позволяет организовать работу программ, требующих большего объема ОЗУ, чем фактически имеется в компьютере.

Страничная организация памяти

- Системная память Linux организована в виде страниц объемом 4K. Если оперативная память полностью исчерпана, ОС будет искать давно не использованные страницы памяти для их перемещения из памяти на жесткий диск. Если какие-либо из этих страниц становятся нужны, Linux восстанавливает их с диска. Некоторые старые Unix-системы и некоторые современные платформы (включая Microsoft Windows) переносят на диск все содержимое ОП, относящееся к неработающему в данный момент приложению, (т. е. ВСЕ страницы памяти, относящиеся к приложению, сохраняются на диске при нехватке памяти) что менее эффективно.

Загрузка выполняемых модулей "по требованию"

- Ядро Linux поддерживает выделение страниц памяти по требованию, при котором только необходимая часть кода исполняемой программы находится в оперативной памяти, а не используемые в данный момент части остаются на диске.

Совместное использование исполняемых программ

- Если необходимо запустить одновременно несколько копий какого-то приложения (либо один пользователь запускает несколько идентичных задач, либо разные пользователи запускают одну и ту же задачу), то в память загружается только одна копия исполняемого кода этого приложения, которая используется всеми одновременно исполняющимися идентичными задачами.

Общие библиотеки

- Библиотеки - наборы процедур, используемых программами для обработки данных. Существует некоторое количество стандартных библиотек, используемых одновременно более чем одним процессом. В старых системах такие библиотеки включались в каждый исполняемый файл, одновременное выполнение которых приводило к непродуктивному использованию памяти. В новых системах (в частности, в Linux), обеспечивается работа с динамически и статически разделяемыми библиотеками, что позволяет сократить размер отдельных приложений.

Возможность запуска исполняемых файлов других ОС

- Linux не является первой в истории операционной системой. Для ранее разработанных ОС, включая DOS, Windows 95, FreeBSD или OS/2, разработана масса различного, в том числе очень полезного и очень неплохого программного обеспечения. Для запуска таких программ под Linux разработаны эмуляторы DOS, Windows 3.1 и Windows 95. Более того, фирмой Vmware разработана система "виртуальных машин", представляющая собой эмулятор компьютера, в котором можно запустить любую операционную систему. Имеются аналогичные разработки и у других фирм. ОС Linux способна также выполнять бинарные файлы других Intel-ориентированных Unix-платформ, соответствующих стандарту iBCS2 (intel Binary Compatibility).

Поддержка различных форматов файловых систем

- Linux поддерживает большое число форматов файловых систем, включая файловые системы DOS и OS/2, а также современные журналируемые файловые системы. При этом и собственная файловая система Linux, которая называется Second Extended File System (ext2fs), позволяет эффективно использовать дисковое пространство.

Сетевые возможности

- Linux можно интегрировать в любую локальную сеть. Поддерживаются все службы Unix, включая Networked File System (NFS), удаленный доступ (telnet, rlogin), работа в TCP/IP сетях, dial-up-доступ по протоколам SLIP и PPP, и т. д.. Также поддерживается включение Linux-машины как сервера или клиента для другой сети, в частности, работает общее использование (sharing) файлов и удаленная печать в Macintosh, NetWare и Windows.

- **Работа на разных аппаратных платформах**
- **100%-ное соответствие стандарту POSIX 1003.1. Частичная поддержка возможностей System V и BSD**
- **System V IPC**

  Linux использует технологию IPC (InterProcess Communication) для обмена сообщениями между процессами, использования семафоров и общей памяти.

Работа под Линуксом

Вход в систему

- login: вводим "root" и нажимаем клавишу <Enter>
- Password:

- **[root]# useradd student**

- Для того, чтобы система разрешила работать пользователю с именем jim, надо задать ему пароль. Для этого вводим команду
- **[root]# passwd student**
- Появится строка
- **New UNIX password:**
- Вводите пароль. После того, как вы завершите ввод нажатием клавиши <Enter>, система попросит ввести его повторно:
- **Retype new UNIX password:**

- **whoami** Сообщает имя, с которым вы вошли в систему в данном сеансе работы
- **w** или **who** Сообщает, какие пользователи работают в данный момент в системе
- **pwd** - Сообщает имя текущего каталога
- **ls –l** - Выдает список файлов и подкаталогов текущего каталога
- **cd <имя_каталога>** - Осуществляет смену текущего каталога
- **ps ax** - Выдает список выполняющихся процессов

- Оболочка bash является не только командным процессором, но и мощным языком программирования. В ней имеется целый ряд встроенных (внутренних) команд и операторов, а, кроме того, в качестве команды может использоваться любая программа, хранящаяся в виде файла на диске.
- Список встроенных команд можно получить по команде help.
- Детальную информацию по конкретной встроенной команде выдает та же команда help с указанием в качестве параметра имени встроенной команды, например: help cd.

- Входить в систему под именем суперпользователя не рекомендуется, поскольку любое неосторожное действие суперпользователя может привести к нежелательным последствиям. Входя под именем простого пользователя, вы, по крайней мере, не можете по неосторожности удалить или испортить системные файлы. В то же время, имеется ряд действий (например, монтирование файловых систем), выполнить которые может только суперпользователь. Не перезагружать же каждый раз компьютер! Именно в таких ситуациях выручает команда su. Достаточно ввести команду su и текущая оболочка (так и хочется сказать "система") запустит для вас новый экземпляр оболочки, в который вы попадете уже с правами пользователя root. Естественно, что для этого вам придется (в ответ на соответствующий запрос) ввести пароль этого пользователя. Закончив выполнять администраторские действия, выйдите из оболочки, и вы снова станете непривилегированным пользователем с отведенными ему полномочиями.

- Если ввести символ табуляции после того, как введена одна из команд и пробел, оболочка предполагает, что вы ищете имя файла, который должен вводиться как параметр команды, и выдает в качестве подсказки список файлов текущего каталога. Если же достаточная часть имени файла введена, то заканчивается ввод этого имени в командную строку.
- Аналогичным образом можно пытаться угадывать окончания переменных окружения, если вместо клавиши <Tab> воспользоваться комбинацией <Esc>+<$>.

Справка в Linux

- **man <команда>**

- Но для того, чтобы получить нужную информацию, нужно еще знать, что искать. В таком случае могут помочь команды **whatis** и **apropos**.
- Команда **whatis** производит контекстный поиск заданного ключевого слова (шаблона) в базе данных, содержащей перечень системных команд с кратким описанием команды. Выводятся только точные совпадения с ключевым словом.
- Команда **apropos** производит поиск по фрагментам слов. Аналогично команде apropos работает команда man с параметром -k. Попробуйте, например,
- **[user]$ man -k net**

Команда info

- Команда **info** является некоторой альтернативой команде **man**. Для получения информации по отдельной команде надо задать в командной строке info с параметром, являющимся именем интересующей вас команды, например,
- **[user]$ info man**

Команда locate

- По этой команде производится поиск всех файлов, имена которых содержат заданный шаблон. Например, по команде locate net будет найдена масса имен файлов, в названиях которых встречается подстрока "net". В шаблоне могут применяться метасимволы *, ? , []. Однако команда locate производит поиск не по каталогам файловой системы, а в специально созданной базе имен файлов, которую надо вначале создать (и иногда обновлять) командой updatedb.
- В некоторых дистрибутивах (например, в ALTLinux) вместо locate имеется команда slocate, которая сама создает для себя базу имен файлов (после запуска с соответствующим параметром).

Файлы и их имена

- Имена файлов в Linux могут иметь длину до 255 символов и состоять из любых символов, кроме символа с кодом 0 и символа / (слэша). Однако имеется еще ряд символов, которые имеют в оболочке shell специальное значение и которые поэтому не рекомендуется включать в имена. Это следующие символы:

  ! @ # $ & ~ % * ( ) [ ] { } ' " \ : ; > < ` пробел.

- Можно заключить имя файла или каталога с такими символами в двойные кавычки. Например, для создания каталога с именем "My old files" следует использовать команду:

  [user]$ mkdir "My old files"

  так как команда

  [user]$ mkdir My old files

  создаст каталог с именем "My".

- В Linux различаются символы верхнего и нижнего регистра в именах файлов.
- Поэтому **FILENAME.tar.gz** и **filename.tar.gz** вполне могут существовать одновременно и являться именами разных файлов.

Каталоги

- файлы группируются в каталоги, которые, в свою очередь, могут быть включены в другие каталоги. В результате получается иерархическая структура каталогов, начинающаяся с корневого каталога. Каждый (под)каталог может содержать как отдельные файлы, так и подкаталоги.
- Иерархическую структуру каталогов обычно иллюстрируют рисунком "дерева каталогов", в котором каждый каталог изображается узлом "дерева", а файлы - "листьями".

- В MS Windows или DOS каталоговая структура строится отдельно для каждого физического носителя (т. е., имеем не отдельное "дерево", а целый "лес") и корневой каталог каждой каталоговой структуры обозначается какой-нибудь буквой латинского алфавита (отсюда уже возникает некоторое ограничение). В Linux (и UNIX вообще) строится единая каталоговая структура для всех носителей, и единственный корневой каталог этой структуры обозначается символом "/". В эту единую каталоговую структуру можно подключить любое число каталогов, физически расположенных на разных носителях (как говорят, "смонтировать файловую систему" или "смонтировать носитель").

- Имена каталогов строятся по тем же правилам, что и имена файлов. И, вообще, каталоги в принципе ничем, кроме своей внутренней структуры (до которой ОС уже есть дело) не отличаются от "обычных" файлов, например, текстовых.
- Полным именем файла (или путем к файлу) называется список имен вложенных друг в друга подкаталогов, начинающийся с корневого каталога и оканчивающийся собственно именем файла. При этом имена подкаталогов в этом списке разделяются тем же символом "/", который служит для обозначения корневого каталога.
- В каждый момент времени пользователь работает с одним экземпляром оболочки shell и эта оболочка хранит значение так называемого "текущего" каталога, т. е. того каталога, в котором пользователь сейчас работает. Имеется специальная команда, которая сообщает вам значение текущего каталога - **[user]$ pwd**.

- Кроме текущего каталога для каждого пользователя определен еще его "домашний каталог" - каталог, в котором пользователь имеет все права: может создавать и удалять файлы, менять права доступа к ним и т. д. В каталоговой структуре Linux домашние каталоги пользователей обычно размещаются в каталоге /home и имеют имена, совпадающие с именем пользователя. Например, /home/student. Каждый пользователь может обратиться к своему домашнему каталогу с помощью значка ~, т. е., например, пользователь student может обратиться к каталогу /home/student /doc как к ~/doc. Когда пользователь входит в систему, текущим каталогом становится домашний каталог данного пользователя.

- Для изменения текущего каталога служит команда cd. В качестве параметра этой команде надо указать полный или относительный путь к тому каталогу, который вы хотите сделать текущим. Понятие полного пути очевидно, а понятие относительного пути требует дополнительного пояснения. Относительным путем называется перечисление тех каталогов, которые нужно пройти в "дереве каталогов", чтобы перейти от текущего каталога к какому-то другому каталогу (назовем его целевым). Если целевой каталог, т. е. каталог, который вы хотите сделать текущим, расположен ниже текущего в структуре каталогов, то сделать это просто: вы указываете сначала подкаталог текущего каталога, затем подкаталог того каталога и т. д., вплоть до имени целевого каталога. Если же целевой каталог расположен выше в каталоговой структуре, или вообще на другой "ветви" дерева, то ситуация несколько сложнее.

- Для каждого каталога (кроме корневого) в дереве каталогов однозначно определен "родительский каталог". В каждом каталоге имеются две особых записи. Одна из них обозначается просто точкой и является указанием на этот самый каталог, а вторая запись, обозначаемая двумя точками, - указатель на родительский каталог. Эти имена из двух точек и используются для записи относительных путей. Чтобы сделать текущим родительский каталог, достаточно дать команду
- **[user]$ cd ..**
- А чтобы перейти по дереву каталогов на два "этажа" вверх, откуда спуститься в подкаталог **kat1/kat2** надо дать команду
- **[user]$ cd ../../kat1/kat2**

Команда ls

- Если дать команду **ls** без параметров, то выводятся только имена файлов текущего каталога. Если нужно просмотреть содержимое не текущего, а какого-то другого каталога, надо указать команде **ls** полный или относительный путь к этому каталогу.
- Кроме имени файла (или подкаталога) запись о нем в соответствующем каталоге содержит еще массу информации об этом файле. Для того, чтобы получить эту информацию, надо использовать дополнительные параметры команды ls. Если дать команду **ls** с параметром **-l**, то будут выданы не только имена файлов, но также данные о правах доступа к файлу (подробнее о правах будет рассказано ниже), количество жестких ссылок или имен файла (для каталога указывается число дополнительных блоков), имя владельца файла и группы файла, его размер и дата последней модификации.

Команда ls

- При задании параметра -t сортировка файлов будет производиться не по именам, а по времени модификации файла.
- Задание параметра -u приводит к тому, что вместо времени модификации файла будет выводиться время последнего доступа к файлу.
- Параметр -r меняет порядок сортировки на обратный (используется вместе с параметрами -l и -t).

Параметры можно перечислять как отдельно:

- **[user]$ ls -l -t -r**

так и объединять:

- **[user]$ ls -ltr**

Права доступа к файлам и каталогам

- Поскольку Linux - система многопользовательская, вопрос об организации разграничения доступа к файлам и каталогам является одним из существенных вопросов, которые должна решать операционная система.

- В основе механизмов разграничения доступа лежат имена пользователей и имена групп пользователей. Вы уже знаете, что в Linux каждый пользователь имеет уникальное имя, под которым он входит в систему (логируется). Кроме того, в системе создается некоторое число групп пользователей, причем каждый пользователь может быть включен в одну или несколько групп. Создает и удаляет группы суперпользователь, он же может изменять состав участников той или иной группы.

- **[user]$ ls -l /bin/ls**

  -rwxr-xr-x 1 root root 49940 Sep 12 1999 /bin/ls

- Вы видите, что в данном случае владельцем файла является пользователь root и группа root. Но нас сейчас в выводе этой команды больше интересует первое поле, определяющее тип файла и права доступа к файлу. Это поле в приведенном примере представлено цепочкой символов **-rwxr-xr-x**. Эти символы можно условно разделить на 4 группы.
- Первая группа, состоящая из единственного символа, определяет тип файла. Этот символ в соответствии с возможными типами файлов, рассмотренными в предыдущем разделе, может принимать такие значения:
- **-** = - обычный файл;
- **d** = - каталог;
- **l** = - символическая ссылка (link).

Далее следуют три группы по три символа **rwx r-x r-x**, которые и определяют права доступа к файлу соответственно для владельца файла, для группы пользователей, которая сопоставлена данному файлу, и для всех остальных пользователей системы. В нашем примере права доступа для владельца определены как rwx, что означает, что владелец (root) имеет

- право читать файл (**r**),
- производить запись в этот файл (**w**),
- и запускать файл на выполнение (**x**).

Замена любого из этих символов прочерком будет означать, что пользователь лишается соответствующего права. В том же примере мы видим, что все остальные пользователи (включая и тех, которые вошли в группу root) лишены права записи в этот файл, т. е. не могут файл редактировать и вообще как-то изменять.

Команда **chmod**

Для изменения прав доступа к файлу используется команда **chmod**. Ее можно использовать в двух вариантах. В первом варианте вы должны явно указать, кому какое право даете или кого этого права лишаете:

[user]$ chmod wxp имя-файла где вместо символа w подставляется

- либо символ u (т. е. пользователь, который является владельцем);
- либо g (группа);
- либо o (все пользователи, не входящие в группу, которой принадлежит данный файл);
- либо a (все пользователи системы, т. е. и владелец, и группа, и все остальные).

Вместо x ставится:

- либо + (предоставляем право);
- либо - (лишаем соответствующего права);
- либо = (установить указанные права вместо имеющихся).

Вместо p - символ, обозначающий соответствующее право:

- r (чтение);
- w (запись);
- x (выполнение).

Команда **chmod**

- Вот несколько примеров использования команды chmod:
- **[user]$ chmod a+x file_name**

  предоставляет всем пользователям системы право на выполнение данного файла.
- **[user]$ chmod go-rw file_name**

  удаляет право на чтение и запись для всех, кроме владельца файла.
- **[user]$ chmod ugo+rwx file_name**

  дает всем права на чтение, запись и выполнение.

- Если опустить указание на то, кому предоставляется данное право, то подразумевается, что речь идет вообще обо всех пользователях, т. е. вместо
- **[user]$ chmod a+x file_name**

  можно записать просто
- **[user]$ chmod +x file_name**

Второй вариант задания команды chmod

- Второй вариант задания команды chmod (он используется чаще) основан на цифровом представлении прав. Для этого мы кодируем символ r цифрой 4, символ w - цифрой 2, а символ x - цифрой 1. Для того, чтобы предоставить пользователям какой-то набор прав, надо сложить соответствующие цифры. Получив, таким образом, нужные цифровые значения для владельца файла, для группы файла и для всех остальных пользователей, задаем эти три цифры в качестве аргумента команды chmod (ставим эти цифры после имени команды перед вторым аргументом, который задает имя файла). Например, если надо дать все права владельцу (4+2+1=7), право на чтение и запись - группе (4+2=6), и не давать никаких прав остальным, то следует дать такую команду:
- [user]$ chmod 760 file_name
- Если вы знакомы с двоичным кодированием восьмеричных цифр, то вы поймете, что цифры после имени команды в этой форме ее представления есть не что иное, как восьмеричная запись тех самых 9 бит, которые задают права для владельца файла, группы файла и для всех пользователей.

Команда mkdir

- Команда **mkdir** позволяет создать подкаталог в текущем каталоге. В качестве аргумента этой команде надо дать имя создаваемого каталога. Во вновь созданном каталоге автоматически создаются две записи: . (ссылка на этот самый каталог) и .. (ссылка на родительский каталог).
- Чтобы создать подкаталог, вы должны иметь в текущем каталоге право записи.
- Можно создать подкаталог не в текущем, а в каком-то другом каталоге, но тогда необходимо указать путь к создаваемому каталогу:
- **[user]$ mkdir /home/kos/book/glava5/part1**
- Команда mkdir может использоваться со следующими опциями:
- **-m mode** - задает режим доступа для нового каталога (например, -m 755);
- **-p** - создавать указанные промежуточные каталоги (если они не существуют).

Команда cp

Команду cp можно применять в одной из двух форм:

- **[user]$ cp [options] source destination**
- **[user]$ cp [options] source_directory new_directory**

В первом случае файл или каталог source копируется, соответственно, в файл или каталог destination, а во втором случае файлы, содержащиеся в каталоге source_directory копируются в каталог new_directory. Для копирования надо иметь права на чтение файлов, которые копируются, и права на запись в каталог, в который производится копирование.

Если в качестве целевого указывается существующий файл, то его содержимое будет затерто, поэтому при копировании надо соблюдать осторожность.

Впрочем, можно использовать команду **cp** с опцией **-i**, тогда перед перезаписью существующего файла будет запрашиваться подтверждение (очень рекомендую вам всегда использовать эту опцию!).

Команда mv

- Если вам необходимо не скопировать, а переместить файл из одного каталога в другой, вы можете воспользоваться командой mv. Синтаксис этой команды аналогичен синтаксису команды cp. Более того, она сначала копирует файл (или каталог), а только потом удаляет исходный файл (каталог).
- Команда mv может использоваться не только для перемещения, но и для переименования файлов и каталогов (т. е. перемещения их внутри одного каталога). Для этого надо просто задать в качестве аргументов старое и новое имя файла:
- **[user]$ mv oldname newname**
- Но учтите, что команда mv не позволяет переименовать сразу несколько файлов (используя шаблон имени), так что команда **mv *.xxx *.yyy** не будет работать.
- При использовании команды **mv**, также как и при использовании **cp**, не забывайте применять опцию **-i** для того, чтобы получить предупреждение, когда файл будет перезаписываться.

Команды rm и rmdir

Для удаления ненужных файлов и каталогов в Linux служат команды

- **rm** (удаляет файлы) и
- **rmdir** (удаляет пустой каталог).
- Если вы попытаетесь использовать команду rm (без всяких опций) для удаления каталога, то будет выдано сообщение, что это каталог, и удаления не произойдет. Для удаления каталога надо удалить в нем все файлы, после чего удалить сам каталог с помощью команды rmdir. Однако можно удалить и непустой каталог со всеми входящими в него подкаталогами и файлами, если использовать команду rm с опцией -r.

Если вы дадите команду rm *, то удалите все файлы в текущем каталоге. Подкаталоги при этом не удалятся. Для удаления как файлов, так и подкаталогов текущего каталога надо тоже воспользоваться опцией -r.

Однако всегда помните, что в Linux нет команды восстановления файлов после их удаления (даже если вы спохватились сразу же после ошибочного удаления файла или каталога)!

Команда find и шаблоны для имен файлов

- Общий синтаксис команды find имеет следующий вид:
- **find [список_каталогов] критерий_поиска**
- Чаще всего поиск проводится по именам файлов, как это показано в предыдущем примере, т. е. "критерий_поиска" задается как "-name имя_файла".
- **[user]$ find /usr/share/doc /usr/doc /usr/locale/doc -name instr.txt**
- Чаще всего шаблоны имен файлов строятся с помощью специальных символов "*" и "?". Значок "*" используется для замены произвольной строки символов. В Linux
- "*****" - соответствует всем файлам, за исключением скрытых;
- "**.***" - соответствует всем скрытым файлам (но также текущему каталогу "." и каталогу уровнем выше "..": не забывайте об этом!);
- "***.***" - соответствует только тем файлам и каталогам, которые имеют "**.**" в середине имени, или оканчиваются на точку;

- Значок ? заменяет один произвольный символ, поэтому index?.htm будет соответствовать именам index0.htm, index5.htm и indexa.htm.
- Кроме "*" и "?" в Linux при задании шаблонов имен можно использовать квадратные скобки [], в которых дается либо список возможных символов, либо интервал, в который должны попадать возможные символы. Например, [abc]* соответствует всем именам файлов, начинающимся с a, b, c; *[I-N1-3] соответствует файлам, имена которых оканчиваются на I, J, K, L, M, N, 1, 2, 3.

Сравнение файлов

- Стоит воспользоваться командой diff для получения полного отчета о том, каковы же различия в интересующих нас файлах. Для получения отчета достаточно указать команде, какие именно файлы сравнивать:
- **[user]$ diff paper.old paper.new**
- Отчет о выявленных различиях будет выдан на стандартный выход. Естественно, его лучше перенаправить в файл:
- **[user]$ diff paper.old paper.new >paper.diff**

Компилятор GNU gcc

- Если планируется установка программного обеспечения, которое распространяется в исходном открытом коде, то понадобится специальная программа-компилятор для создания из него выполняемого файла. Многие программы для ОС Linux, написаны на языке программирования С. Таким образом, на вашем Linux должен быть установлен компилятор языка С. Самый широко распространенный компилятор для Linux — это GNU — компилятор языка С (gcc).
- Чтобы определить, какая версия gcc используется в вашей системе, в командной строке задайте опцию --version:
- **[user]$ gcc --version**

Утилита make

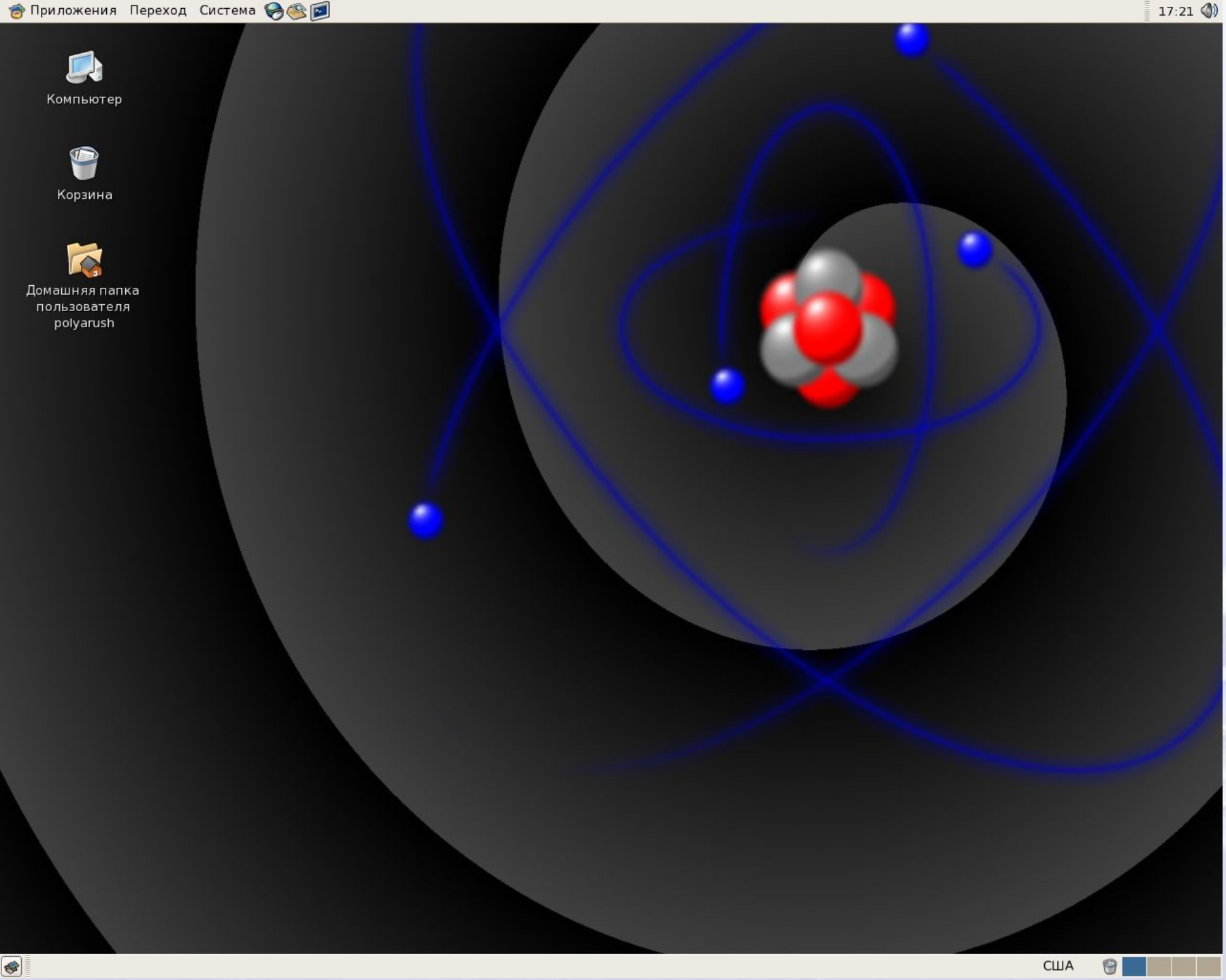
- Со временем проекты на C и C++ становились все сложнее. Приходилось работать с несколькими исходными файлами, каждый из которых имел свой файл заголовка. Компиляция отдельных исходных файлов порождала несколько объектных файлов, которые затем необходимо было связать в определенной комбинации для получения выполняемых файлов, а это довольно громоздкая и неприятная работа. Для упрощения этих задач большинством компиляторов языка C и C++ используется утилита **make**.
- Главной составляющей утилиты **make** является **Makefile**. В нем указывается алгоритм компиляции исходного кода для создания выполняемой программы

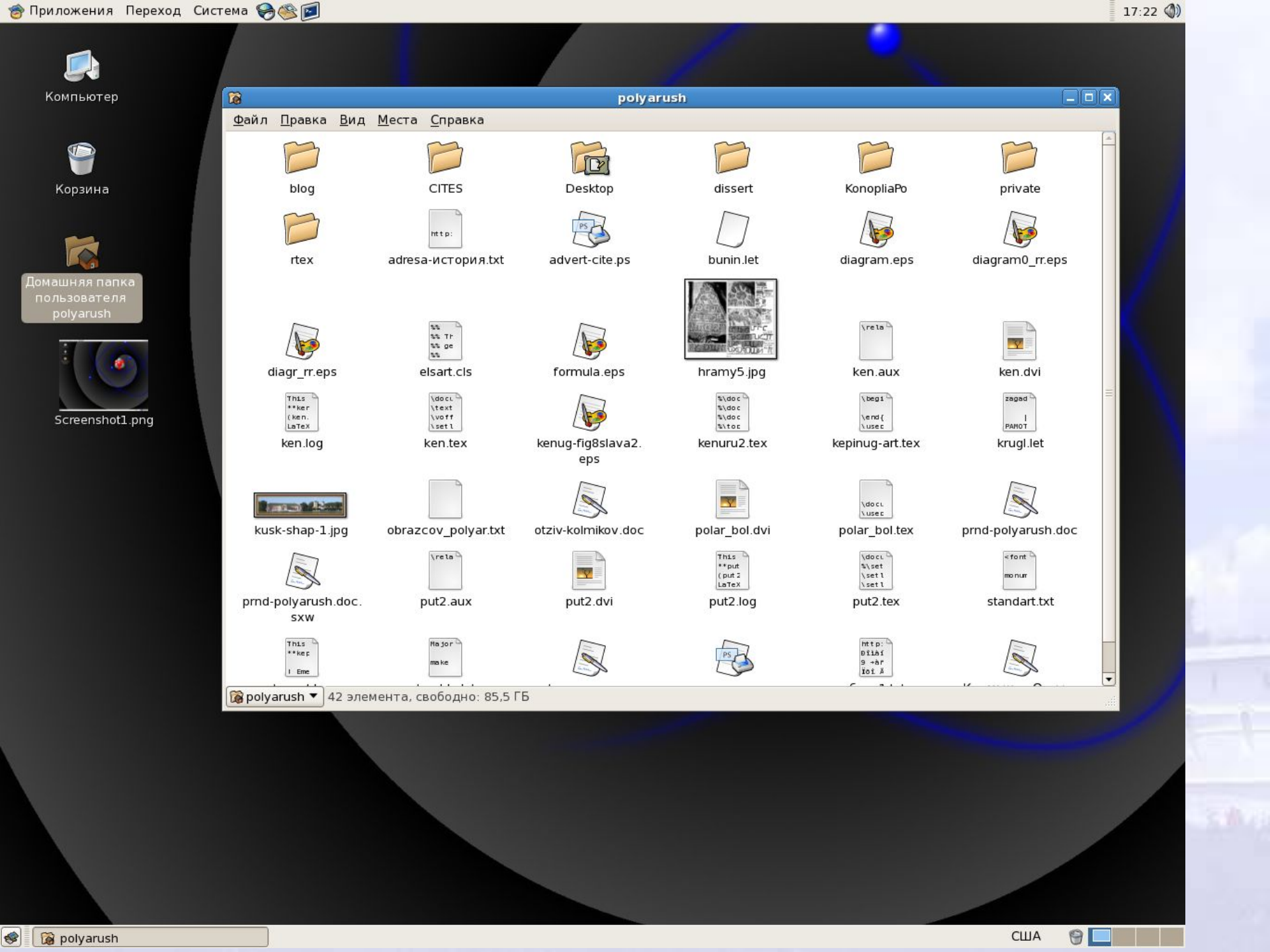
```
CC = gcc
CXX = g++ $(CXXFLAGS)
FORT = g77 -c -O2

LIBDIRS =
CFLAGS = -Wall -O2
.F.o:
    $(FORT) $<
.f.o:       $(FORT) $<
VPATH = /istra/daq/v07_6_1/include
dirs = $(subst :, ,$(VPATH))
incl_dirs = $(patsubst %,-I%,$(dirs))
CXXFLAGS =  $(incl_dirs)

ifeq ($(MAKECMDGOALS),9)
VPATH = /istra/daq/v09/include
dirs = $(subst :, ,$(VPATH))
incl_dirs = $(patsubst %,-I%,$(dirs))
CXXFLAGS =  $(incl_dirs)
endif
```

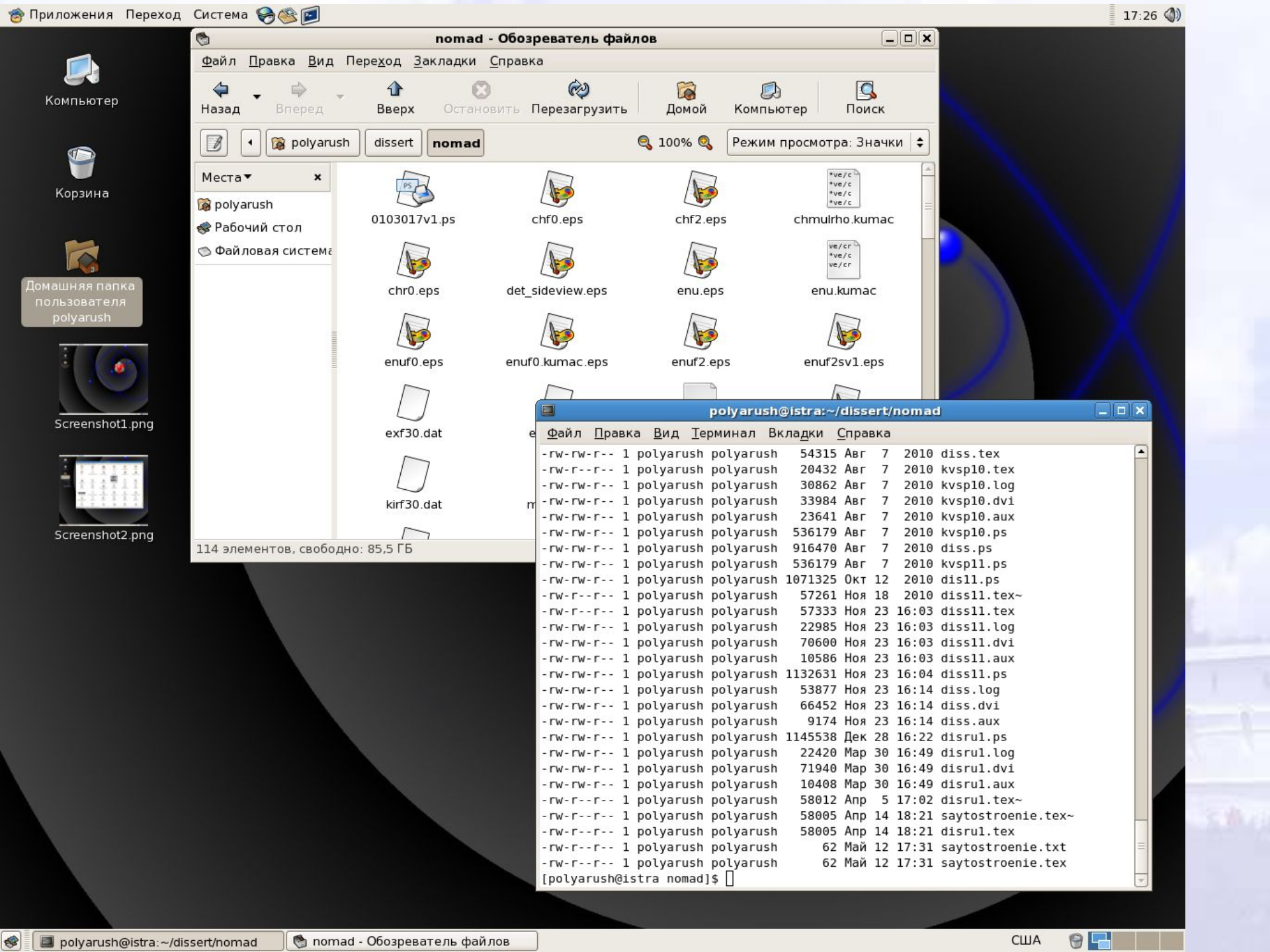
```
RCS = main.cc \
 IQ_SubDet.c \
IQservice.c \
minpro.f \
 select_new.F \
 fits_new.F \
analys.F \
restpol.F \
service.F \
 ecal_new.F    \
fmuon_new.F  \
shadow.F \
cfits.F \
noise.F \
search_sp2.F \
mc_read.F
OBJS = analys.o \
minpro.o \
select_new.o \
fits_new.o \
IQ_SubDet.o \
service.o   \
IQservice.o \
 fmuon_new.o     \
ecal_new.o \
shadow.o \
cfits.o \
noise.o \
search_sp2.o \
restpol.o \
mc_read.o
```

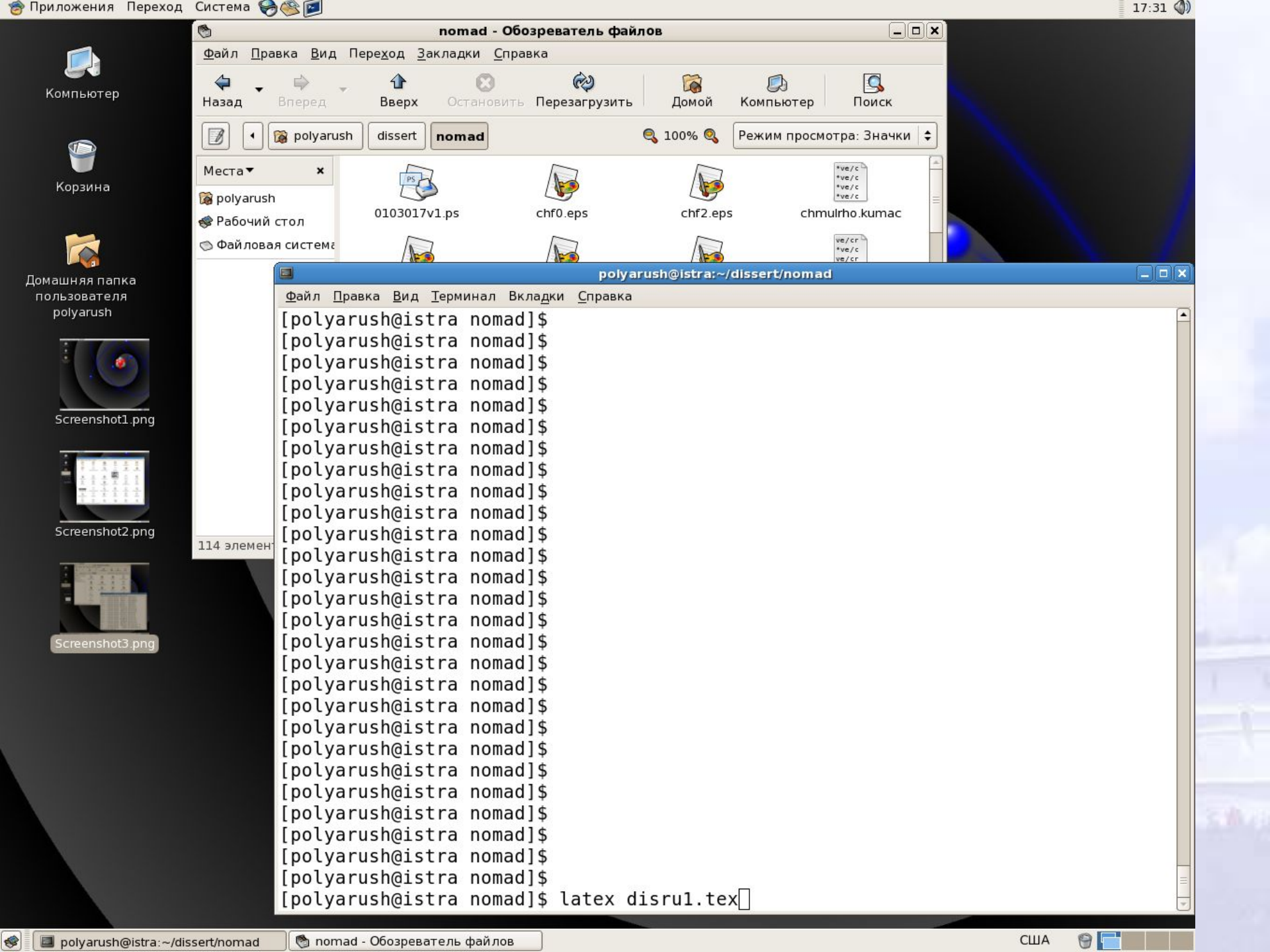
- 6: main6

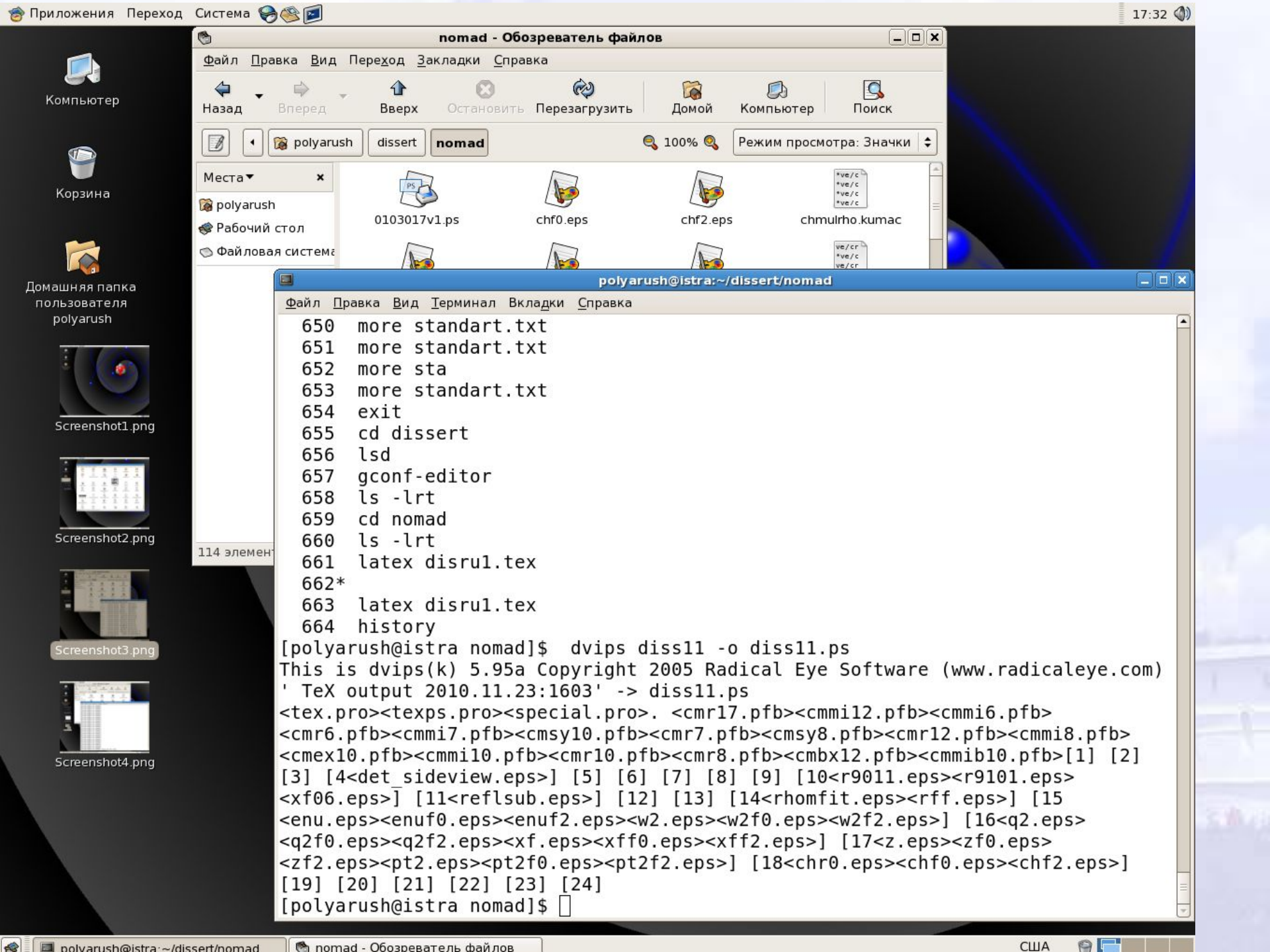
- main6: $(OBJS) main.o
- $(CXX) $(CFLAGS) $(LIBDIRS) $(OBJS) main.o -o analys \ -L/istra/prod/daq/v07_6_1/lib -ldaq `/cern/2004/bin/cernlib mathlib` -lg2c9: main9 main
- 9: $(OBJS) main.o
- $(CXX) $(CFLAGS) $(LIBDIRS) $(OBJS) main.o -o analys \ -L/istra/daq/v09/lib -ldaq `/cern/2004/bin/cernlib mathlib` -lg2c
- clean: rm -f *.o


Приложения
Переход
Система
17:21
Компьютер
Корзина
Домашняя папка пользователя polyarush
США


Приложения Переход Система
17:22
Компьютер
Корзина
Домашняя папка пользователя polyarush
Screenshot1.png
polyarush
Файл Правка Вид Места Справка
blog
CITES
Desktop
dissert
KonopliaPo
private
rtex
adresa-история.txt
advert-cite.ps
bunin.let
diagram.eps
diagram0_rr.eps
diagr_rr.eps
elsart.cls
formula.eps
hramy5.jpg
ken.aux
ken.dvi
ken.log
ken.tex
kenug-fig8slava2.eps
kenuru2.tex
kepinug-art.tex
krugl.let
kusk-shap-1.jpg
obrazcov_polyar.txt
otziv-kolmikov.doc
polar_bol.dvi
polar_bol.tex
prnd-polyarush.doc
prnd-polyarush.doc.sxw
put2.aux
put2.dvi
put2.log
put2.tex
standart.txt
polyarush
42 элемента, свободно: 85,5 ГБ
polyarush
США


Приложения Переход Система
17:26
Компьютер
Корзина
Домашняя папка пользователя polyarush
Screenshot1.png
Screenshot2.png
nomad - Обозреватель файлов
Файл Правка Вид Переход Закладки Справка
Назад Вперед Вверх Остановить Перезагрузить Домой Компьютер Поиск
polyarush dissert nomad
100% Режим просмотра: Значки
Места
polyarush
Рабочий стол
Файловая система
0103017v1.ps
chf0.eps
chf2.eps
chmulrho.kumac
chr0.eps
det_sideview.eps
enu.eps
enu.kumac
enuf0.eps
enuf0.kumac.eps
enuf2.eps
enuf2sv1.eps
exf30.dat
kirf30.dat
114 элементов, свободно: 85,5 ГБ
polyarush@istra:~/dissert/nomad
Файл Правка Вид Терминал Вкладки Справка
-rw-rw-r-- 1 polyarush polyarush 54315 Авг 7 2010 diss.tex
-rw-r--r-- 1 polyarush polyarush 20432 Авг 7 2010 kvsp10.tex
-rw-rw-r-- 1 polyarush polyarush 30862 Авг 7 2010 kvsp10.log
-rw-rw-r-- 1 polyarush polyarush 33984 Авг 7 2010 kvsp10.dvi
-rw-rw-r-- 1 polyarush polyarush 23641 Авг 7 2010 kvsp10.aux
-rw-rw-r-- 1 polyarush polyarush 536179 Авг 7 2010 kvsp10.ps
-rw-rw-r-- 1 polyarush polyarush 916470 Авг 7 2010 diss.ps
-rw-rw-r-- 1 polyarush polyarush 536179 Авг 7 2010 kvsp11.ps
-rw-rw-r-- 1 polyarush polyarush 1071325 Окт 12 2010 dis11.ps
-rw-r--r-- 1 polyarush polyarush 57261 Ноя 18 2010 diss11.tex~
-rw-r--r-- 1 polyarush polyarush 57333 Ноя 23 16:03 diss11.tex
-rw-rw-r-- 1 polyarush polyarush 22985 Ноя 23 16:03 diss11.log
-rw-rw-r-- 1 polyarush polyarush 70600 Ноя 23 16:03 diss11.dvi
-rw-rw-r-- 1 polyarush polyarush 10586 Ноя 23 16:03 diss11.aux
-rw-rw-r-- 1 polyarush polyarush 1132631 Ноя 23 16:04 diss11.ps
-rw-rw-r-- 1 polyarush polyarush 53877 Ноя 23 16:14 diss.log
-rw-rw-r-- 1 polyarush polyarush 66452 Ноя 23 16:14 diss.dvi
-rw-rw-r-- 1 polyarush polyarush 9174 Ноя 23 16:14 diss.aux
-rw-rw-r-- 1 polyarush polyarush 1145538 Дек 28 16:22 disru1.ps
-rw-rw-r-- 1 polyarush polyarush 22420 Мар 30 16:49 disru1.log
-rw-rw-r-- 1 polyarush polyarush 71940 Мар 30 16:49 disru1.dvi
-rw-rw-r-- 1 polyarush polyarush 10408 Мар 30 16:49 disru1.aux
-rw-r--r-- 1 polyarush polyarush 58012 Апр 5 17:02 disru1.tex~
-rw-r--r-- 1 polyarush polyarush 58005 Апр 14 18:21 saytostroenie.tex~
-rw-r--r-- 1 polyarush polyarush 58005 Апр 14 18:21 disru1.tex
-rw-r--r-- 1 polyarush polyarush 62 Май 12 17:31 saytostroenie.txt
-rw-r--r-- 1 polyarush polyarush 62 Май 12 17:31 saytostroenie.tex
[polyarush@istra nomad]$
polyarush@istra:~/dissert/nomad
nomad - Обозреватель файлов
США


Приложения Переход Система
17:31
Компьютер
Корзина
Домашняя папка пользователя polyarush
Screenshot1.png
Screenshot2.png
Screenshot3.png
nomad - Обозреватель файлов
Файл Правка Вид Переход Закладки Справка
Назад Вперед Вверх Остановить Перезагрузить Домой Компьютер Поиск
polyarush dissert nomad
100% Режим просмотра: Значки
Места
polyarush
Рабочий стол
Файловая система
0103017v1.ps
chf0.eps
chf2.eps
chmulrho.kumac
114 элемент
polyarush@istra:~/dissert/nomad
Файл Правка Вид Терминал Вкладки Справка
[polyarush@istra nomad]$ latex disru1.tex
polyarush@istra:~/dissert/nomad
nomad - Обозреватель файлов
США


Приложения Переход Система
17:32
Компьютер
Корзина
Домашняя папка пользователя polyarush
Screenshot1.png
Screenshot2.png
Screenshot3.png
Screenshot4.png
nomad - Обозреватель файлов
Файл Правка Вид Переход Закладки Справка
Назад Вперед Вверх Остановить Перезагрузить Домой Компьютер Поиск
polyarush dissert nomad
100% Режим просмотра: Значки
Места
polyarush
Рабочий стол
Файловая система
0103017v1.ps
chf0.eps
chf2.eps
chmulrho.kumac
114 элемент
polyarush@istra:~/dissert/nomad
Файл Правка Вид Терминал Вкладки Справка
650 more standart.txt
651 more standart.txt
652 more sta
653 more standart.txt
654 exit
655 cd dissert
656 lsd
657 gconf-editor
658 ls -lrt
659 cd nomad
660 ls -lrt
661 latex disru1.tex
662*
663 latex disru1.tex
664 history
[polyarush@istra nomad]$ dvips diss11 -o diss11.ps
This is dvips(k) 5.95a Copyright 2005 Radical Eye Software (www.radicaleye.com)
' TeX output 2010.11.23:1603' -> diss11.ps
<tex.pro><texps.pro><special.pro>. <cmr17.pfb><cmmi12.pfb><cmmi6.pfb>
<cmr6.pfb><cmmi7.pfb><cmsy10.pfb><cmr7.pfb><cmsy8.pfb><cmr12.pfb><cmmi8.pfb>
<cmex10.pfb><cmmi10.pfb><cmr10.pfb><cmr8.pfb><cmbx12.pfb><cmmib10.pfb>[1] [2]
[3] [4<det_sideview.eps>] [5] [6] [7] [8] [9] [10<r9011.eps><r9101.eps>
<xf06.eps>] [11<reflsub.eps>] [12] [13] [14<rhomfit.eps><rff.eps>] [15
<enu.eps><enuf0.eps><enuf2.eps><w2.eps><w2f0.eps><w2f2.eps>] [16<q2.eps>
<q2f0.eps><q2f2.eps><xf.eps><xff0.eps><xff2.eps>] [17<z.eps><zf0.eps>
<zf2.eps><pt2.eps><pt2f0.eps><pt2f2.eps>] [18<chr0.eps><chf0.eps><chf2.eps>]
[19] [20] [21] [22] [23] [24]
[polyarush@istra nomad]$
polyarush@istra:~/dissert/nomad
nomad - Обозреватель файлов
США

section 9 the multiplicities measured in this analysis are compared with the results of previous experiments; summary and conclusions are given in section 10 .

2 The experimental setup

2.1 The NOMAD detector

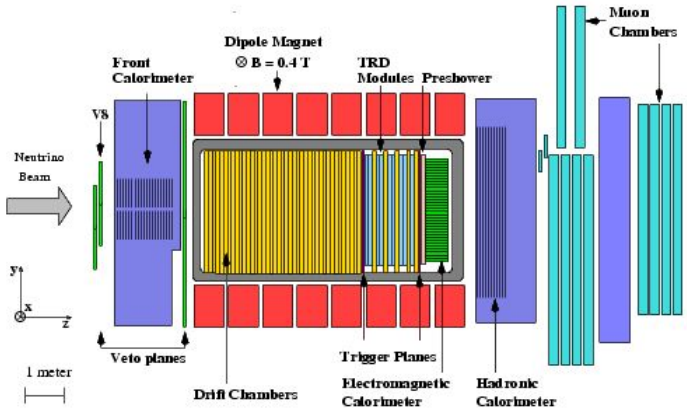


Рис. 1: A sideview of the NOMAD detector.

NOMAD (see Fig. 1) is described in [34]. It consists of a number of sub-detectors most of which are located inside a 0.4 T dipole magnet with an inner volume of $7.5\times3.5\times3.5$ m^3: an active target made of drift chambers (DC) [35] with a mass of 2.7 tons (mainly carbon), an average density of 0.1 g/cm^3 and a total length of about one radiation length ($\sim 1.0X_0$), followed by a transition radiation detector (TRD) [36], a preshower detector (PS) and a lead glass electromagnetic calorimeter (ECAL) [37]. The low density and the good instrumentation of the