Лабораторный практикум по информатике
Третий уровень

Емельянов Эдуард Владимирович

14 февраля 2007 г.

1. Знакомство с Linux: вход в систему, X, консоль.

1.1. Знакомство с клавиатурой

Если вы знакомы с расположением и назначением клавиш стандартной компьютерной клавиатуры, не читайте этот раздел.

Рассмотрите клавиатуру. Начнем с верхнего правого угла. Здесь расположена клавиша `<Esc>` (произносится - *„искейп”*) — „Отмена”. Ее нажатие зачастую равносильно нажатию кнопки „Отмена” диалоговых окон.

Далее идет ряд **функциональных клавиш** `<F1>` ...`<F12>`, которые имеют особое назначение в различных программах. По стандарту не меняется значение только клавиши `<F1>` — ее нажатие вызывает справочную систему данной программы.

Следующая группа из трех клавиш не представляет особого значения, т.к. они не используются практически ни в каких программах.

Нижняя часть клавиатуры разделена на три горизонтальных группы: текстовая клавиатура, клавиши редактирования текста / управления курсором, цифровая клавиатура, подобная клавиатуре калькулятора.

Текстовая клавиатура подобна клавиатуре печатной машинки за исключением дополнительных клавиш. Клавиша „Забой” (⟵) — крайняя справа в верхнем ряду предназначена для удаления символа слева от курсора. Во втором ряду есть еще одна особая клавиша — „Табуляция” или „Tab” (↹). Ее значение важно при форматировании текстовых документов (если вы хотите получить ровные столбцы из цифр. По краям третьего ряда расположены клавиши `<Caps Lock>` и „Ввод” (`Enter` — ↵).

Клавиша `<Caps Lock>` служит для фиксирования верхнего регистра при вводе букв. Если ее нажать, загорится соответствующий светодиодный индикатор, и вы будете печатать заглавными буквами. Если необходимо ввести 1-2 символа строчными буквами — достаточно нажать одну из клавиш `<Shift>` (они расположены по краям второго ряда снизу) и, удерживая ее — нажать клавишу с нужным символом. Для окончания ввода заглавными буквами, достаточно повторно нажать клавишу `<Caps Lock>`. Если вы печатаете строчными буквами и вам необходимо набрать 1-2 символа заглавными, нет необходимости нажимать `<Caps Lock>` — достаточно нажать нужную клавишу, удерживая `<Shift>`. Аналогично вводятся спецсимволы, расположенные на цифровых клавишах (!@#$%^&*()_+).

При нажатии клавиши `<Enter>` в командной строке выполняется только что введенная команда, в текстовом редакторе она заканчивает текущую строку (абзац).

Для смены раскладки клавиатуры (английская/русская) нужно нажать одновременно клавиши `<Ctrl>` и `<Shift>`.

Клавиши `<Ctrl>` и `<Alt>` используются только в сочетании с другими клавишами для выполнения *клавиатурных макросов*.

Основные сочетания клавиш:

- `<Ctrl> + <Fn>` — переключение на n-й рабочий стол;
- `<Alt> + <F1>` — вызов K-меню (в оболочке KDE);
- `<Alt> + <F2>` — вызов меню „Запуск программы”;

- `<Alt> + <F3>` — меню окна (в графической оболочке);
- `<Alt> + <F4>` — закрыть окно;
- `<Alt> + <↹>` — переключение между открытыми окнами;
- `<Ctrl> + <↹>` — переключение между рабочими столами;
- `<Ctrl> + <Shift>` — смена раскладки клавиатуры;
- `<Ctrl> + <w>` — закрыть дочернее окно в многооконном приложении / закрыть окно;
- `<Ctrl> + <q>` — закрыть все окна многооконного приложения / закрыть окно;
- `<Ctrl> + <s>` — сохранить текущий документ;
- `<Ctrl> + <a>` — выделить все (в текстовых и подобных им документах);
- `<Shift> + <курсорные клавиши>` — выделить блок текста (можно выделять, нажав левую клавишу мыши и передвигая курсор от начала нужного блока до его конца);
- `<Ctrl> + <c>` — скопировать выделенный блок в буфер обмена (специальная область памяти);
- `<Ctrl> + <v>` — вставить в данное места блок из буфера;
- `<Ctrl> + <x>` — вырезать выделенный блок в буфер обмена;
- `<Ctrl> + <z>` — отменить предыдущее действие;
- `<Ctrl> + <Home>` — переместиться в начало документа;
- `<Ctrl> + <End>` — переместиться в конец документа;
- `<Ctrl> + <Page Up>/<Page Down>` — пролистать 1 страницу назад/вперед.

1. 2. Вход в Linux

1. 2.1.Графический интерфейс пользователя (ГИП)

Сразу после загрузки Linux возникает экран приглашения входа в систему. В поле „`Имя пользователя`" нужно ввести свой **логин** — регистрационное имя в системе. Далее система попросит ввести свой пароль. Пока вы его не смените, это — ваш логин, окруженный с двух сторон точками. Например, если ваш логин „`igor_k`", то паролем будет „`.igor_k.`". После входа в систему вы сможете сменить свой пароль командой `passwd` (об этом — позже).

Если введенные логин и пароль были правильными, появится заставка входа, информирующая о процессе загрузки **X** и **оболочки** (X — общеупотребительное название для сервера графического интерфейса пользователя, оболочкой называется выбранный вид интерфейса.) Существует множество графических оболочек. Наиболее популярные среди них — **KDE** и **Gnome**. По умолчанию вашей оболочкой будет KDE, но во время входа в систему можно определить свою оболочку по умолчанию (для этого внизу экрана входа расположены соответствующие клавиши).

После окончания загрузки оболочки вы можете приступать к работе в X. Как можете заметить на первый взгляд, X почти ничем не отличаются от графической оболочки Windows, но, например, KDE имеет куда больше возможностей, чем ГИП W.

Выполните задание (1):

Войдите в KDE, посмотрите, как выглядит эта оболочка. Затем выйдите из нее, нажав в К-меню кнопку „Завершить сеанс". Войдите в Gnome и посмотрите, как выглядит эта среда. выйдя из Gnome, установите понравившуюся вам оболочку по умолчанию. Кроме того, установите по умолчанию русский язык (выбрать эти настройки можно в любой момент до нажатия `<Enter>` после ввода пароля.

1. 2.2.Текстовая консоль

Linux — многопользовательская операционная система, одновременно на ПК с Linux могут работать много человек (естественно, сидя за разными ПК, но выполняя команды на данном), чего нельзя сделать в W. Изначально в Linux не было поддержки X, с тех времен сохранилась возможность работать в реальной консоли (позже вы узнаете, что в X есть ее аналоги — `Xterm` и `Konsole`, которые чем-то похожи на режим сессии DOS в W.)

В Linux существует 6 консолей (терминалов). Перейти на какую-нибудь из них из X можно, нажав сочетание клавиш `<Ctrl> + <Alt> + <F1...6>`. Монитор переключится в текстовый режим и вы увидете приглашение в консоль:

```
ASPLinux releas ...
Kernel ...
... login:
```

Вы можете ввести свой логин, а затем пароль. После этого появится приглашение к работе вида `igor_k at /home/101/igor_k>`. Это — приглашение **командного процессора**, обрабатывающего вводимые команды и выполняющего соответствующие действия. Теперь можно работать в консоли, выполняя различные команды. Конечно, рядовому пользователю это покажется страшным, но истинные *линуксоиды* очень уважают консоль и даже в X работают обычно в виртуальном терминале. Это действительно очень удобно: зная всего 100 – 200 основных команд проще набрать их в консоли, чем искать значки этих приложений в меню. Кроме того, некоторые команды можно выполнить только в консоли.

► Не забывайте, что в Linux всем устройствам соответствуют файлы в каталоге `/dev`, поэтому работа с устройством сводится к открытию соответствующего файла и чтению/записи информации из него. Работу по непосредственной передаче данных устройству выполняет его **драйвер**.

► В Linux существует понятие **стандартного ввода, стандартного вывода и стандартного потока ошибок**. Их назначение в следующем: стандартным вводом называют файл, из которого программа читает входные данные, стандартный вывод — файл, в который программа помещает выходные данные, а в стандартный поток ошибок записываются все ошибки, возникающие при выполнении программы. Если не определено иначе, все три потока по умолчанию работают с одним и тем же файлом — консолью.

Для переключения между консолями используйте сочетания `<Alt> + <F1...6>`, а для переключения в X — `<Alt> + <F7>`.

Выполните задание (2):

Переключитесь в консоль. Посмотрите, как выглядит стандартное приглашение. Просмотрите содержимое своего домашнего каталога (сеанс консоли начинается в этой папке), набрав команду `ls`.

Создайте папку `Work`, набрав `mkdir Work`. В этой папке вы будете сохранять все свои задания. Постепенно, по мере их выполнения, вы освоите Linux, а тогда освоить любую другую ОС вам будет очень просто.

Зайдите в папку Work: `cd Work` и создайте в ней файл, куда в дальнейшем будете заносить информацию о новых командах: `touch Help.txt`.

Просмотреть содержимое файла можно командой `cat`: `cat Help.txt` покажет вам содержимое только что созданного файла (но, так как он пуст, ничего не отобразится). Занесите в этот файл новые команды, завершая каждую строку нажатием `<Enter>`:

```
echo "ls <папка> - вывод содержимого текущей или указанной папки" >> Help.txt
```

```
echo "mkdir <имя> - создание папки"  >> Help.txt
echo "cd <папка> - смена текущей папки"  >> Help.txt
echo "touch <имя> - создание файла"  >> Help.txt
echo "cat <имя файла> - вывод содержимого файла на экран"  >> Help.txt
echo "echo - вывод на экран текста, стоящего после команды"  >> Help.txt
```

► Переключение на русский язык - нажатием клавиш <Ctrl> + <Shift>. Если вы не можете найти клавиши <Shift>: это крайние клавиши второго ряда снизу в группе текствых клавиш.

Обратите внимание на то, что после каждой команды echo мы писали >> Help.txt. Это — **перенаправление вывода команды**. Вместо того, чтобы писать текст на экран, он отображается в файл.

Если после команды, делающей вывод результатов на экран, написать > filename, командный процессор запишет весь вывод этой программы в файл filename, а если такой файл существовал, прежнее его содержимое будет затерто. Если же после команды набрать >> filename, то если файл filename существовал, вывод команды будет дописан в его конец.

► Чтобы не набирать одинаковые команды, полезно использовать клавиши перемещения курсора вверх и вниз, расположенные внизу на клавиатуре между текстовой и вспомогательной цифровой клавиатурами. Эти клавиши помогают передвигаться в истории введенных команд на 1 пункт. Кроме того, клавиши <Page Up> и <Page Down> позволяют передвигаться на 30 команд в истории.

2. Основные элементы графического интерфейса. Файловая система.

2. 1. Работа в X

2. 1.1.Рабочие столы

Самый первый элемент графического интерфейса любой ОС, который бросается в глаза — **рабочий стол** (см. рис. 1). В Linux по умолчанию 4 рабочих стола (можно сделать до 20-ти), переключиться между которыми можно либо нажимая <Ctrl> + ↹, либо – <Ctrl> +<Fn>, где n — номер желаемого рабочего стола. Наличие нескольких рабочих столов очень удобно, так как можно одновременно работать с большим количеством приложений, как бы на нескольких мониторах.

Выполните задание (3):

Настройте свои рабочие столы: разместите на них заставки, настройте хранитель экрана.

Следующий элемент — **панель управления**. В Linux можно создать сколько угодно панелей, лишь бы вам было удобно. Настроить панель можно, выбрав соответствующий пункт из ее контекстного меню. На основной панели расположены: кнопка меню; кнопки

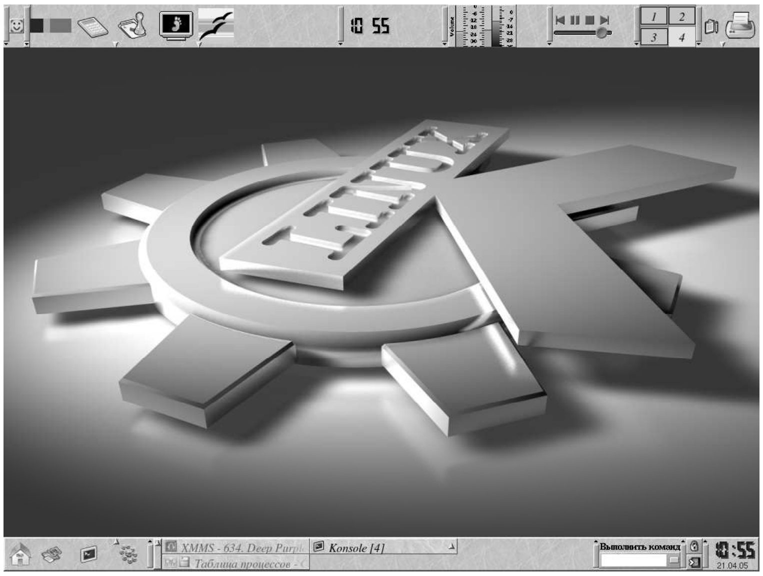

Рис. 1: Рабочий стол

быстрого запуска приложений; **панель задач** — область, в которой расположены кнопки активации открытых окон; апплет выхода и запирания экрана и часы.

2. 1.2.Меню

Раскройте К-меню (см. рис. 2). Ознакомьтесь с ним.

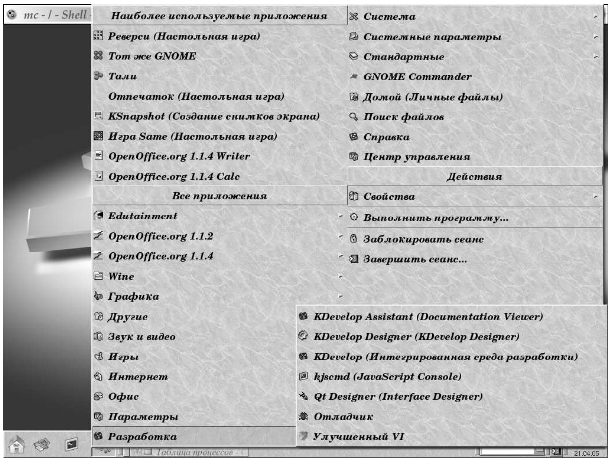

Рис. 2: К-меню

Оно состоит из трех частей:

- **Действия** — выход из KDE, запирание экрана (если вы ненадолго отлучаетесь от компьютера и хотите на время блокировать его), выполнение программы (если вы знаете ее название), свойства — настройка различных параметров KDE.
- **Все приложения** — запуск различных приложений.
- **Наиболее используемые программы** — список программ, которые вы чаще всего запускаете.

Обратите внимание на пункт „Центр управления”. При помощи этой программы вы можете настроить не только KDE, но и различные службы (см. рис. 3).

Выполните задание (4):

Запустите центр управления. На вкладке слева выберите пункт „Внешний вид и темы". Настройте KDE по своему вкусу.

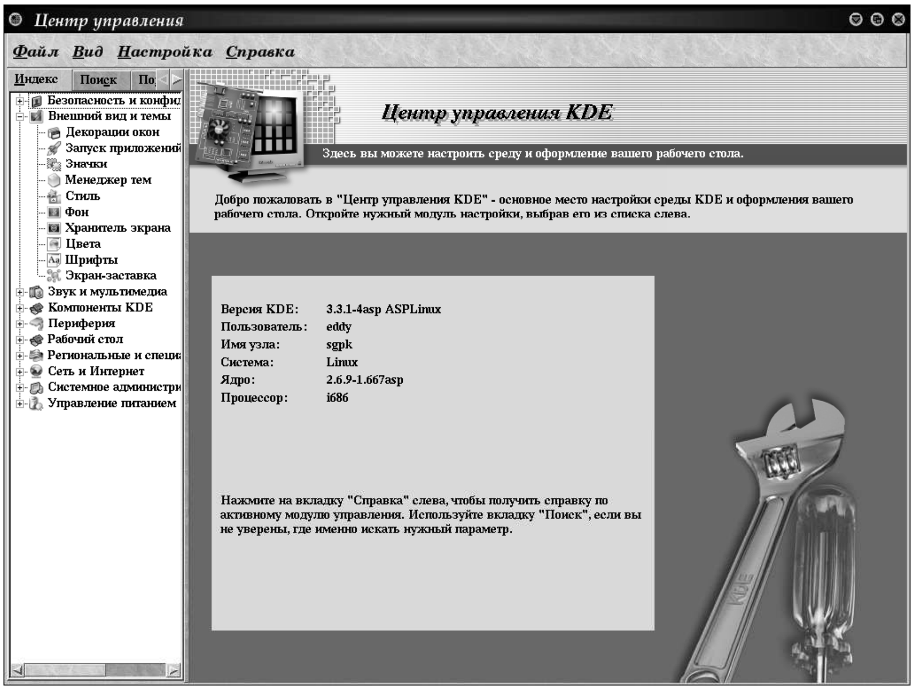

Рис. 3: Центр управления KDE

2. 2. Выполнение команд пользователем

2. 2.1.Вызов программ

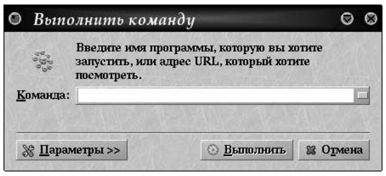

Рис. 4: Окно „Выполнить программу"

Для выполнения единичных команд существует пункт К-меню „Выполнить программу" (см. рис. 4). Вызвать это окно можно также, нажав сочетание клавиш `<Alt> + <F2>`. В появившемся окне введите название команды и нажмите `<Enter>`.

Выполните задание (5):

Попробуйте открыть таким образом браузер Mozilla (он вызывается командой `mozilla`).

2. 2.2.Работа в виртуальной консоли

Для того, чтобы не переключаться постоянно из графической оболочки в консоль существуют **виртуальные консоли**. Самая примитивная из них — вызывается командой `xterm` (см. рис. 5).

Рис. 5: Виртуальный терминал (консоль) Xterm

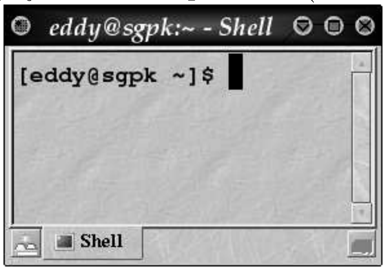

Рис. 6: Виртуальная консоль Konsole

Виртуальная консоль практически аналогична консоли реальной. В ней вы также можете выполнять команды, редактировать файлы и т.п.

Более „красивая” виртуальная консоль — `konsole` (см. рис. 6).

Она уже более приспособлена к X, чем `xterm`, и позволяет пользователю менять свое оформление (см. рис. 7).

Рис. 7: „Прозрачная” тема Konsole

Выполните задание (6):

Запустите программу *konsole* (либо из окна „Выполнить программу”, либо из К-меню →„Система” →„Терминал”, либо же из панели быстрого запуска, нажав на иконку с изображением монитора).

Настройте на свой вкус тему консоли (нажмите правую клавишу мыши, затем выберите из контекстного меню пункт „Настройка”).

Наберите команду `ls`. Она покажет вам список файлов и папок текущего каталога. Следующее задание вы будете выполнять в консоли.

2. 3. Файловая система (ФС). Доступ к ресурсам ФС.

2. 3.1.Консольный доступ

Linux поддерживает много типов ФС. Но наиболее распространены три: `xfs`, `reiserfs` и `ext3fs`. На вашем компьютере разделы жесткого диска отформатированы в reiserfs. В любом случае, в какой бы ФС ни были отформатированы разделы, в Linux существует следующая организация файловой структуры: существует каталог верхнего уровня (**корневой каталог** — /), в котором размещены каталоги следующего уровня.

Другие разделы жесткого диска можно **монтировать** в любой каталог командой `mount`. Вам разрешается монтировать только флоппи- и компакт-диски. Остальные разделы файловой системы в целях безопасности разрешено монтировать только суперпользователю root (администратор компьютера). Более простая процедура монтирования диска — через его иконку на рабочем столе (пункт „Подключить" контекстного меню). Для **отмонтирования** диска используется команда `umount` или пункт „Отмонтировать" (или „Извлечь" — для компакт-диска) контекстного меню иконки диска.

В файловых системах Linux помимо понятий файл и каталог существуют особые файлы: **символическая и жесткая ссылки**. Создать жесткую ссылку можно командой `ln <имя файла> [<имя ссылки>]`, а символическую — `ln -s <имя файла> [<имя ссылки>]`. Символическая ссылка — это файл, в котором записан полный путь к другому файлу. Жесткая ссылка возможна *только внутри того же раздела, на котором размещена цель ссылки*, она представляет собой запись в каталоге, содержащую *файловый дескриптор цели*. Можно сказать, что жесткая ссылка — это копия файла, которая практически не занимает места на диске. Ссылка отличается от копии тем, что если редактировать цель ссылки, то по всем ссылкам получим измененный файл. По понятиям безопасности нельзя создавать жесткие ссылки на папки (иначе может получиться циклическая запись ссылок, в итоге которой из папки нельзя будет выйти).

Выполните задание (7):

Создайте в своем домашнем каталоге `/home/<ваш логин>` файл `link` (если забыли — командой `touch link`). Создайте на него жесткую и символическую ссылки (`ln link h_link` и `ln -s link s_link`). Внесите в этот файл запись: `echo "lalala» link`. Посмотрите, как изменились файлы `s_link` и `h_link: cat s_link; cat h_link`.

Удалите файл `link: rm link`. Теперь посмотрите опять файлы `s_link` и `h_link`. Как видите, в файле — жесткой сслылке ничего не изменилось, а при попытке открыть символическую появилась ошибка: `No such file or directory`, говорящая, что объект ссылки удален.

В каждой папке *всегда существуют по крайней мере две других*: ссылка на саму себя (.) и ссылка на родительскую папку (..). Ссылка на саму себя существует для запуска программ, находящихся в данной папке, а ссылка на родительский каталог — для удобства перемещения (команда `cd ../` перемещает вас на один уровень файловой системы выше).

Конечно, постоянно набирать команды `cd, ls` и т.п. неудобно, поэтому были созданы **файловые менеджеры**. Наиболее популярный консольный файловый менеджер — `mc` (Midnight Commander) (см. рис. 8).

Выполните задание (8):

Запустите mc в виртуальной консоли. В самом низу находится полоса подсказок: нажав на функциональную клавишу с таким номером будет совершено действие, указанное на подсказке.

Откройте для редактирования свой файл `Help.txt` (клавиша <F4>). Запишите в него все новые команды, которые вы узнали за сегодняшний урок.

2. 3.2.Доступ к ФС из X

Просмотреть содержимое ФС из X можно при помощи любого браузера. На панели управ-

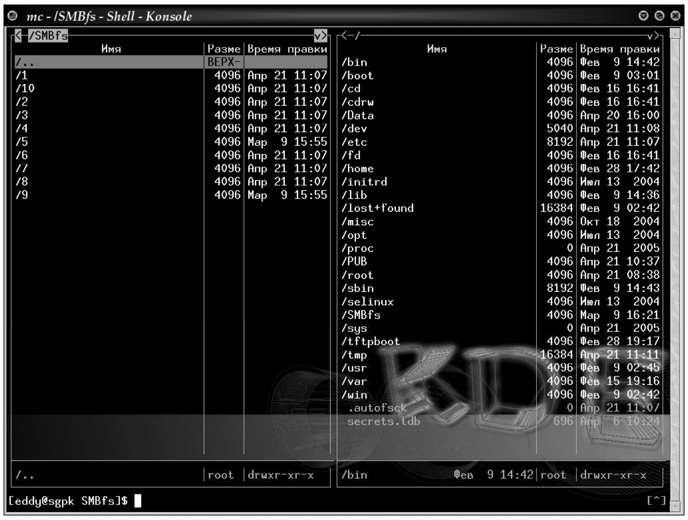

Рис. 8: Файловый менеджер Midnight Commander (`mc`)

ления есть кнопка „Домой”. Нажмите ее. Откроется браузер Konqueror, в котором вы сможете просмотреть содержимое своей домашней папки.

► Запустить Konqueror можно и командой `konqueror`

Переместиться в какую-нибудь папку можно, щелкнув по ней два раза левой клавишей мыши (если вы не меняли настроек действия мыши в „Центре управления”). Кроме того, на левой панели Конкуерора находится **дерево каталогов** для быстрого доступа к ним. Если в каталоге есть подуровни, он отображается с крестиком слева, нажав на который можно увидеть вложенные в него папки.

► Дважды щелкнув на каком-нибудь файле левой клавишей мыши можно открыть его программой по умолчанию для данного типа файлов. Если вы хотите открыть его другой программой, щелкните по файлу правой клавишей мыши и из контекстного меню „Открыть в” выберите программу.

► Можно настроить Konqueror так, чтобы папки открывались одиночным нажатием левой кнопки мыши, тогда чтобы выделить элемент ФС, необходимо навести на него мышь и немного подождать.

Выполните задание (9):

Скопируйте в свой домашний каталог пару песен с учительского компьютера (папка `Music`). Сделайте это при помощи:

1. mc — в левой панели откройте папку `Music`, в правой — домашнюю папку.

   ► Переключаться между панелями можно при помощи клавиши <Tab> (⭾).

   Затем в панели слева выделите нужные файлы клавишей <Insert>, в домашней папке создайте каталог Music (клавиша <F7>), зайдите в него, переместитесь в левую панель и, нажав <F5>, скопируйте файлы в свою папку.

2. При помощи браузера — нажмите кнопку „Домой”, войдите в папку `/Music`. Выделите несколько песен нажатием левой клавиши мыши.

   ► Если при выделении держать нажатой клавишу <Ctrl>, можно выделить объекты, расположенные в произвольном порядке.

   - Нажмите правую клавишу мыши на любом из выделенных файлов. Из контекстного меню выберите пункт „Копировать”. Перейдите в папку, куда нужно скопиро-

вать файлы и нажмите правую клавишу в свободном месте папки. Из контекстного меню выберите „Вставить". Файлы скопируются.
- Проделайте то же самое, но вместо вызова контекстного меню нажмите сочетание `<Ctrl> + <C>`. Перейдите в папку, куда нужно скопировать и нажмите `<Ctrl> + <V>`. Файлы скопированы.

Прослушать песни можно при помощи программы Xmms (`xmms`). В К-меню: „Звук и видео" → „Музыкальный проигрыватель".

3. Набор, редактирование и форматирование текстовых документов.

В текстовых редакторах имеется множество дополнительных функций, ускоряющих и облегчающих работу с текстами:

- расстановка переносов;
- изменение масштаба просмотра документа;
- автоматическая нумерация страниц;
- создание границ вокруг текста и страниц;
- разработка своего стиля и использование его в дальнейшем для быстрого форматирования текста;
- поддержка нескольких словарей для проверки правильности написания;
- работа с макросами;
- поддержка нескольких окон;
- удобная справочная система;
- проверка орфографии, грамматики и синтаксиса;
- настройка панелей инструментов (добавление и удаление кнопок и команд);
- возможность предварительного просмотра перед печатью;
- настройка параметров страницы и поддержка нестандартных форматов бумаги;
- вставка в текст специальных символов.

В Linux есть много разных текстовых редакторов: от примитивных вроде `gedit` до сложных, как Open Office (OO). Gedit предназначен для набора простых текстов, которым не нужно содержать сложного форматирования и т.п. (например, компьютерных программ).

OO — сложная офисная система, предназначенная для набора, редактирования и форматирования текстовых документов любой сложности. Конечно, при помощи этой программы нельзя создавать профессиональные текстовые документы для печати в типографии (такие документы набирают в специальных типографских системах — $\TeX$, $\LaTeX$), но для офисного использования OO подходит как нельзя лучше.

3.1. Ознакомление с OO Writer

Откройте текстовый редактор OO Writer. Сохраните сразу пустой документ в своей рабочей папке.

В открытом редакторе всегда отображаются две панели инструментов: „Стандартная" и „Форматирование". Панели содержат кнопки редактирования и форматирования текста, а также выпадающие списки, в которых можно выбрать стиль текста, его гарнитуру (шрифт) и размер. Для просмотра описания пунктов панели нужно навести на них курсор мыши, не нажимая никаких клавиш. Через небольшой промежуток времени под курсором загорится описание данной кнопки или списка.

Кроме того, в случае возникновения вопросов по нажатию кнопки`<F1>`возникает окно помощи, где можно найти описания и советы. Кроме того, в новых версиях офисных пакетов существует т.н. „Помощник", дающий советы в процессе работы.

Ознакомьтесь с меню и панелями инструментов.

3. 2. Ввод и редактирование текста

Наберите текст:

> Для понятия аснов работы с пирсональным компьютером, прежди всего важна практика.
> Практические занятия емеют огромнейшее значение благодаря им учащиеся приобретают навыки работы на ПК осваивают различные програмные продукты которые могут пригодиться им в дальнейшей жизни не только на работе но и дома. Ведь в наше время очень тежело представить себе человека никоим образом не относящимся к информационно-вычислительной технике.
> Выполнение практических заданий носит характер самообучения. Учащиеся самостоятельно постигают основы работы с ПК, преподаватель же выполняет роль консультанта в случае возникновения трудностей и ошибок.

Обратите внимание на ошибки (они подчеркиваются красной или зеленой волнистой линией), исправьте их. Выполните расстановку переносов в тексте.

3. 3. Форматирование текста

Выделите некоторые элементы текста курсивом, некоторые — жирным, некоторые — подчеркнутым. Задайте разные цвета некоторых слов. Сделайте различное выравнивание абзацев (по центру, слева, справа, по ширине). Например:

> Для понятия основ работы с *персональным компьютером*, прежде всего важна **практика**.
>
> Практические занятия имеют огромнейшее значение*благодаря им учащиеся приобретают навыки работы на ПК осваивают различные програмные продукты* которые могут пригодиться им в дальнейшей жизни не только на работе но и дома.Ведь в наше время очень тежело представить себе человека никоим образом не относящимся к информационно-вычислительной технике.
>
> Выполнение практических заданий *носит характер самообучения.* Учащиеся самостоятельно постигают основы работы с ПК, преподаватель же выполняет роль **консультанта** в случае возникновения трудностей и ошибок.

► Размер символов, а также их гарнитуру можно задать, изменяя значения в выпадающем списке панели инструментов. Начертание символов можно изменить либо нажав кнопки „**Ж (B)**”, „*К (i)*”, „П (U)” на панели инструментов, либо сочетания клавиш: `<Ctrl> + <B>`, `<Ctrl> + <I>`, `<Ctrl + U>` соответственно для жирного, курсивного или подчеркнутого начертания.

Сделайте надпись, использующую нижние и верхние индексы, спецсимволы, например, $X_i^k = \sum X_{ij}^k \cdot y_i$.

► Для того, чтобы введенный символ был верхним или нижним индексом, необходимо выделить его и в контекстном меню (по нажатию правой кнопки мыши) выбрать пункт „Стиль ⇒ Верхний/Нижний индекс” („Style” ⇒„Subscript/Superscript”), где пометить нужный аттрибут символа. Можно добавить на панели инструментов кнопки, создающие верхние/нижние индексы, а можно создать макросы и назначить им какое-нибудь сочетание клавиш.

3. 4. Сноски

Добавьте к некоторым элементам автоматические сноски[1]. Сноски можно маркировать не только подряд, но и задавать произвольный номер[100], еще сноска[2].

3. 5. Колонтитулы

Добавьте в документ верхний и нижний колонтитул. На верхнем колонтитуле отобразите общее количество страниц документа, на нижнем - номер страницы. Отредактируйте документ, задав размер левого поля 2.5см, остальных — 1см. На колонтитулы отведите по 1.5см.

► Для отображения колонтитулов выберите пункт меню „Вставка” ⇒ „Верхний/нижний Колонтитул” („Insert” ⇒„Header”, „Footer”).

3. 6. Использование стилей в документе

Создайте стиль на основе стиля „Основной” („Text Body”) со следующими параметрами:

- Кегль (размер шрифта) — 14 пт;
- Первая строка — красная с отступом 1 см;
- Отступы слева и справа — 0 см;
- Выравнивание — „По ширине”.
- Гарнитура — „Urw Bookman L”

► Для отображения имеющихся стилей выберите пункт меню „Формат” ⇒„Стили” („Format” ⇒„Stylist”) или нажмите `<F11>`.

Дайте стилю название „Мой стиль”. Таким же образом на основе стиля „Заголовок 1” и „Заголовок 2” (Head 1,2) создайте свои собственные стили заголовков 1-го и 2-го уровня. Это пригодится вам в дальнейшем. Выравнивание обоих уровней заголовков создайте по центру, гарнитура — „Times New Roman”, кегль у заголовка 1-го уровня — 30pt, у заголовка 2-го

[1] Вроде такой
[100] А после этого нумерация продолжится с предыдущего числа.
[2] С продолжающейся нумерацией

уровня — 20pt. Заголовок 1-го уровня сделайте жирным, а 2-го — курсивным. До и после заголовков оставьте пространство, равное кеглю заголовков.

Назовите заголовки „Уровень 1” и „Уровень 2”.

Перед набранным до этого текстом создайте заголовок 1-го уровня: „Практическая работа 1”, затем, следом за ним — заголовок 2-го уровня: „Форматирование текста”. Весь остальной текст оформите в стиле „Мой стиль”.

3. 7. Таблицы

Начните задание с новой страницы. Для этого вставьте в документ разрыв страницы.

► Разрыв страницы можно создать двумя способами: либо выбрав из меню „Вставка” пункт „Разрыв” ⇒„Разрыв страницы” („Insert” ⇒„Manual break” ⇒„Page break”), либо нажав сочетание клавиш `<Ctrl> + <Enter>`.

Наберите стилем „Уровень 2” заголовок „Форматирование таблиц”.

Вставьте в документ таблицу — меню „Вставка” ⇒„Таблица” („Insert” ⇒„Table”). Кроме того, таблицу можно вставить при помощи сочетания клавиш `<Ctrl> +<F12>`. При этом появится меню, в котором необходимо определить количество колонок и столбцов новой таблицы, выравнивание по умолчанию в ее ячейках, а также ширину таблицы — фиксированную, по ширине окна, или по размеру содержимого. Нажав кнопку „Автоформат” („Autoformat”) можно задать шаблон предварительного форматирования таблицы.

Для объединения соседних ячеек необходимо выделить их и щелкнуть по выделению правой кнопкой мыши. В открывшемся контекстном меню выбрать пункт „Ячейка” ⇒ „Объединить” („Cell” ⇒„Merge”). Чтобы разбить ячейку необходимо выбрать пункт контекстного меню „Ячейка” ⇒„Разбить” („Split”).

► При работе с таблицами появляется панель инструментов таблицы. Если вы ее не видите, щелкните по синему треугольнику в правой части панели инструментов (это — переключатель панелей).

Создайте таблицу - расписание ваших занятий. Пример:

Расписание занятий гр. ___

№пары	Понедельник	Вторник	Среда	Четверг	Пятница
1	математика	—	физика	—	рус. язык
2	химия	технология	спецпредметы	—	физика
3	информатика	спецпредметы	спецпредметы	—	литература
4	биология	—	химия	—	экология
5	—	—	—	информатика	—
6				физика	
7				рус. язык	
8				литература	

► Для создания невидимых линий между ячейками можно выделить нужную ячейку и, нажав кнопку „Границы” („Borders”) на панели инструментов таблицы, выбрать пункт без границ. В режиме разметки страницы невидимые линии будут отображаться серым цветом. Таким же образом можно изменять стили других границ, стирать ненужные границы.

Чтобы задать какое-то выравнивание в ячейках таблицы необходимо выделить эти ячейки, а затем нажать на панели инструментов кнопку, соответствующую желаемому типу выравнивания. Кроме того, можно нажать на выделении правой кнопкой мыши и из контекстного меню выбрать нужный тип выравнивания.

3. 8. Элементы векторной графики

Вставьте в документ разрыв страницы и наберите стилем „Уровень 1": „Занятие 2". А стилем „Уровень 2" — „Векторная графика".

Создайте в документе какой-либо векторный рисунок, например, блок-схему алгоритма выбора. В OO Writer удобно выбирать уже готовые графические объекты из галереи объектов. В OO Writer существует возможность создать текстовое поле, которое можно вставить из панели „Рисование". Оно обладает большим количеством свойств, сочетание которых позволяет получить полноценный красочный текстовый объект.

► Удобно рисовать, вынеся дополнительную панель рисования: на левом краю окна существует панель дополнительных функций, нажатием кнопок которой вызываются вспомогательные плавающие панели, которые при желании можно встроить в окно.

3. 9. Формулы

На новой странице стилем „Уровень 2" наберите: „Набор и редактирование формул".

3. 9.1.Простые

Вставьте в текст какую-нибудь формулу. Пример:

$$\lim_{x\to 0}\int\limits_0^\pi \frac{\sin x}{x}\,dx = \int\limits_0^\pi dx = \pi$$

► Вставка в текст формулы вызвать из меню „Вставка ⇒„Формула" („Insert" ⇒ „Formula").

3. 9.2.Сложные

Вставьте в текст „многоэтажную" формулу, например,

$$F(\omega, N) = \int\limits_{-\infty}^{\infty} E(t)\cdot\frac{\sin\dfrac{N\omega t}{2\pi}}{\sin\dfrac{\omega t}{2\pi}} e^{-i\omega t}\,dt.$$

► При вставке в текст формулы появляется панель формул. Если она не появится, необходимо в меню „Вид" отметить соответствующий пункт.

3. 10. Автоматические элементы

На новой странице создайте стилем „Уровень 1" заголовок „Автоматические элементы", стилем „Уровень 2" будете создавать подзаголовки, соответствующие названиям выполняемых заданий.

3. 10.1.Использование автоматических списков

Создайте список, содержащий несколько уровней. Используйте различные стили. Например:

Животные:

1. **Домашние:**
    1. *грызуны*
        - мышь
        - крыса
    2. *хищники*
        - собака
        - кошка
2. **Дикие:**
    1. *млекопитающие*
        - кенгуру
        - дельфин
    2. *земноводные*
        - крокодил
        - черепаха

► Для создания списков используйте кнопки . Для задания нового элемента списка просто нажмите „Enter” в конце строки, для создания следующего уровня списка нажмите „Tab” (клавиша слева от буквы q на клавиатуре) после того, как перейдете на новый элемент. Возврат на предыдущий уровень списка – „Shift + Tab”. Редактирование вида маркеров – двойным щелчком левой клавишей мыши на маркере.

3. 11. Оглавление

Если в вашем документе не применялись стили „Уровень 1” и „Уровень 2”, то создайте в нескольких местах документа заголовки. Желательно, чтобы документ состоял из нескольких страниц, и на каждой было хотя бы по одному заголовку 1-го уровня и 2-3 заголовка 2-го уровня.

Вставьте в самом начале текста автоматический элемент „Оглавление”. Оглавление вставляется из меню „Вставка” ⇒„Оглавления и сноски” ⇒„Оглавления и сноски”(„Insert” ⇒ „Indexes and tables” ⇒„Indexes and tables”). Выберите какой-нибудь стиль оглавления, а также количество уровней, которые будут в нем отображаться.

3. 12. Алфавитный указатель

Выделите какое-нибудь слово, например, если вы пользовались классификацией животных, выделите „дикие”. Выберите в меню „Вставка” ⇒ ”Оглавления и сноски” ⇒ „Элемент указателя” („Entry”). В качестве 1-го ключа укажите „Дикие животные”. Укажите, что это – основной элемент. Затем выделите, например, „Кошка”. Опять повторите операцию с выделением указателя, но кроме 1-го ключа укажите 2-й ключ – „Хищники”. Проделайте подобное еще с несколькими словами, затем перейдите в конец документа и вставьте алфавитный указатель также, как вставляли оглавление.

3. 13. Гиперссылки

Вставьте в документ гиперссылку на какой-нибудь рисунок, расположенный в папке „Мои рисунки". Посмотрите, как она работает.

На какой-нибудь странице документа создайте внутреннюю гиперссылку на один из элементов документа (например, какой-нибудь заголовок). Для активации гиперссылки нужно щелкнуть на ней левой клавишей мыши с нажатой клавишей `<Ctrl>`.

4. Электронные таблицы.

► Вспомнить лекцию „Обработка числовой документации"

Для набора табличных данных существует множество специальных программ — электронные таблицы. Конечно, можно набирать таблицы и в текстовом редакторе, но намного удобнее делать это в специальной программе, позволяющей не только набирать и красиво оформлять таблицы, но и производить в них несложные расчеты. Благодаря этому очень удобно создавать различные сметы и отчеты.

Самая важная функция электронных таблиц — создание формул, включающих функцию и диапазон вычисляемых ячеек. Благодаря этому электронные таблицы получили такое широкое применение. Необходимо отметить также возможность прогнозирования значения с помощью анализа "что-если". Это означает, что можно пересмотреть различные варианты данных для достижения результата.

По имеющимся данным можно построить диаграмму.

Диаграммы настраиваются по многим параметрам (тип, размер, шрифт, цвет, оформление и др.). Кроме всего перечисленного в OO Calc имеется множество дополнительных функций, ускоряющих и облегчающих работу с таблицами:

- автоматический подбор высоты строк и ширины столбцов;
- выполнение условного форматирования таблиц;
- наложение защиты на книгу;
- поддержка большинства математических, экономических и других функций;
- возможность сортировки данных таблицы;
- изменение масштаба просмотра документа;
- создание границ вокруг ячеек и таблицы;
- разработка своего стиля и использование его в дальнейшем для
- быстрого форматирования текста;
- поддержка нескольких словарей для проверки правильности написания;
- работа с макросами;
- удобная справочная система;
- проверка орфографии, грамматики и синтаксиса;
- настройка панелей инструментов (добавление и удаление кнопок и команд);
- возможность предварительного просмотра перед печатью;
- настройка параметров страницы и поддержка нестандартных форматов бумаги.

Откройте OO Calc. Ознакомьтесь с меню программы и панелью инструментов.

4. 1. Простые таблицы

Создайте таблицу по образцу:

Экзаменационные оценки				
№п/п	**Ф.И.О.**	**Математика**	**Физика**	**Химия**
1.	Пупкин Вася	2	3	2
2.	Иванов Коля	3	4	4
3.	Сергеев Олег	5	5	4
4.	Никитин Стас	3	3	4
5.	Аверьянов Саша	4	3	3

► Для создания автоматической нумерации поставьте цифру 1 в 1-й ячейке списка, затем выделите ее одним щелчком левой кнопки мыши и протяните на нужное количество ячеек за правый нижний угол (черный квадрат).

Отсортируйте таблицу по алфавиту. Оформите ее границы, сделайте какую-нибудь заливку ячеек таблицы.

► Для сортировки данных выделите диапазон, который необходимо отсортировать, и выберите из меню „Данные" („Data") пункт „Сортировка" („Sort").

4. 2. Таблицы с формулами

Создайте рядом с предыдущей таблицей таблицу 3x30. В первый столбец занесите номер ячейки с данными в таблице (как в предыдущей таблице), во второй столбец занесите значения аргумента — X — начиная от -1.5 и заканчивая +1.5 с шагом 0.1. Для этого введите в первую ячейку -1.5, в ячейке под ней напишите формулу $= Xx + 0.1$, где Xx — кординаты предыдущей ячейки.

► Можно не вводить координаты ячейки, а щелкнуть по ней левой клавишей мыши в процессе набора формулы, или переместиться на нее клавишами управления курсором.

Теперь протяните только что введенную ячейку вниз за правый нижний угол вплоть до конца таблицы.

В третий столбец вставьте какую-нибудь формулу, например, `=SIN(Xx) + 3*TG(Xx)`, где Xx — координаты соседней слева ячейки. Протяните эту ячейку до конца таблицы.

Залейте первый столбец черным цветом, а второй и третий — зеленым. Создайте границы таблицы.

4. 3. Ссылки

Вставьте в какую-нибудь ячейку своего документа гиперссылку на другой документ. Теперь при активации этой ссылки автоматически будет открываться новый документ.

На втором листе документа (см. закладки внизу рабочей области) создайте таблицу „Цены на продуктовые товары". Включите в нее несколько товаров (первый столбец), во втором столбце запишите стоимость товаров в у.е., а в третьем — в рублях. Для автоматического пересчета введите в 1-ю ячейку третьего столбца формулу `=Xx*Yy`. Где Xx — соседняя слева ячейка с ценой в рублях, Yy — ячейка, содержащая курс валюты (поместите ее рядом с таблицей).

Цены на продуктовые товары			
№п/п	**Продукт**	**Цена ($)**	**Цена (р.)**
1.	Сметана (1кг)	1.22	
2.	Хлеб (1 булка)	0.31	
3.	Молоко (1л)	0.43	
4.	Колбаса копченая (1кг)	4.55	
5.	Вареники (0.5кг)	0.95	

Теперь протяните ячейку с формулой вниз. У вас получились нули. Объясните, почему так произошло. Посмотрите, что было записано в ячейки при автозаполнении. Как избавиться от этой проблемы?

► См. лекции (про абсолютные и отностительные ссылки)

Добавьте в таблицу еще 2-3 пункта, протяните формулу, выражающую цену в $ в рублях, чтобы вычислить и их стоимость.

Разметьте границы таблицы, сделайте заливку ячеек, оформьте их.

4. 4. Построение диаграмм и графиков

Откройте или создайте новый файл, содержащий таблицу из двух колонок. В первую занесите координаты X с некоторым постоянным шагом, во вторую — какую-нибудь функцию от X. Пар X — Y должно быть не менее 30. В данной работе будет использоваться эта таблица.

Выделите данную таблицу и постройте по ней гистограмму и график. Для этого выделите колонки с данными осей X и У, затем из меню „Вставка" выберите пункт „Диаграмма" („Insert" ⇒„Chart"). Поставьте флажок на пункте „Первая колонка как подпись оси X" („First column as label"). Другой флажок снимите.

Нажмите кнопку „Далее" („Next"). Выберите тип „График". Убедитесь, что переключатель „Данные в" („Data series in") под окном выбора стоит в положении „Колонках" („Columns"). Нажмите „Далее" и выберите вид графика „Обычный" („Normal"). Нажмите „Далее" и задайте название графика и подписи осей. Снимите флажок с пункта „Легенда" („Legend").

Нажмите „Создать" („Create") и на вашем рабочем листе появится диаграмма.

Создайте какое-нибудь красивое оформление диаграммы, измените шрифты подписей, типы линий, заливку и т.п.

Теперь постройте статистическую таблицу разрядки конденсатора:

t, мс	**I, мкА**	**t, мс**	**I, мкА**	**t, мс**	**I, мкА**	**t, мс**	**I, мкА**
0	150.5	6	7.3	11	1.0	16	0.1
1	100.4	7	5.0	12	0.5	17	0.1
2	55.7	8	3.8	13	0.3	18	0.1
3	35.6	9	2.6	14	0.2	19	0.0
4	19.4	10	2.3	15	0.2	20	0.0

Зная, что конденсатор разряжается по закону $I = I_0 e^{-\dfrac{t}{RC}}$ (∗), а также, что $R = 22$кОм, найдите емкость конденсатора.

Для этого сделаем кое-какие расчеты:

- Т.к. данные таблицы содержат ошибки, нам придется искать средние значения I_0 и C. Прологарифмируем формулу (*): $\ln I = \ln I_0 - \dfrac{t}{RC}$.
- Построим справа от нашей таблицы дополнительный столбец: $\ln I$ (в первую ячейку которого введем формулу `=ln(Xx)`, где `Xx` — значение тока в 0-й момент времени, и растянем до конца таблицы).
  - ▶ В наших вычислениях не надо использовать последние два значения (при $t = 19$ и 20с), иначе они внесут серьезную ошибку в вычисления. Возможно, придется исключить из расчетов и самое первое значение, тогда общее количество данных будет равно 17.
- Искомое значение $C = \dfrac{t}{R(<\ln I_0> - \ln I)}$, усреднив все значения C получим близкую к истине величину. ($<X>$ — обозначение средней величины). Добавим столбец, в который запишем эту формулу: = <t>/22000/(<I0>-<I>), здесь <t>, <I0> и <I> — координаты ячеек, в которых содержатся соответствующие значения времени, логарифма нулевого тока и логарифма тока. Т.к. I0 нам пока не известно, введите вместо <I0> координаты любой пустой ячейки. Запишите в эту ячейку `=ln(150.5)`. Потом мы внесем в нее более точное значение.
  - ▶ Не забывайте, что ссылка на ячейку с адресом $\ln I_0$ не должна изменяться, т.е. это должна быть **абсолютная ссылка**. Например, если Эта величина находится у вас в ячейке `D25`, то ссылку на нее в формуле следует писать так: `$D$29`.

  Чтобы были видны значимые цифры, следует установить цифровой формат ячеек с шестью знаками после запятой (из контекстного меню „Формат ячейки").
- Под заполненным только что столбцом посчитайте среднее значение емкости: `=sum(xX:yY)/18*1000000`. Здесь `(xX:yY)` — весь диапазон емкостей (написав `sum(`, выделите столбец сверху), 18 — количество значений C. Умножать на 1000000 необходимо, чтобы получить значение емкости конденсатора в микрофарадах.
- Рядом создайте столбец, в котором нужно посчитать среднюю ошибку. В первую его ячейку запишите `=(xX*1000000-$y$Y)2̂`, где `Xx` — соседняя слева ячейка, `$y$Y` — ячейка, в которой записано среднее значение емкости. В ячейке снизу запишите сумму ошибок.
- Перебирайте значения I_0 от 145 до 155. Остановитесь на том значении, при котором будет минимальной сумма ошибок.

Теперь вы получили значение емкости конденсатора. Это был пример использования электронных таблиц в физике и математике.

Вообще, для физических и математических расчетов существуют более „продвинутые" электронные таблицы (вроде Origin или LabPlot), в которых намного больше специфических функций: построение и анализ графиков по сложным функциям, выполнение сложных расчетов и т.п.

Для сложных вычислений существует такой „монстр", как Matlab — программа, которая удовлетворит практически все вычислительные и визуализационные потребности современного ученого.

5. Электронные презентации.

Электронные презентации значительно упрощают работу по составлению графического сопровождения различного рода мероприятий.

ОО Impress позволяет:

- создавать профессиональные слайдовые демонстрации, включающие графики, изображения, текст, анимации и прочие объекты;
- при полноэкранном демонстрировании слайдов существует возможность добавления спецэффектов к элементам слайда: тексту, анимациям и изображениям, а также — настраивать эффекты появления самих слайдов на экране;
- создавать в презентациях векторные изображения;
- создавать слайды на основе профессионально оформленных шаблонов;
- создавать различные виды презентации, позволяющие удобно ориентироваться среди уже созданных слайдов в больших презентациях;
- сохранять презентации не только в „родном" формате, но и экспортировать презентацию в формат MS PowerPoint, pdf-формат (являющийся мировым стандартом файлов документации).

5. 1. Создание простой презентации

Запустите ОО Impress. Появится „автопилот", предлагающий создать пустую презентацию, открыть уже имеющуюся презентацию или создать презентацию из шаблона. Создайте пустую презентацию, для этого сразу нажмите кнопку „Создать" („Create"). У вас появится окно „Разметка слайда" („Modify slide"), из которого можно выбрать разметку нового слайда. Для первого слайда выберите второй вариант сверху: текстовое поле (заголовок слайда) вверху слайда и пустое пространство внизу. Нажмите „ОК".

При создании презентации появляется „плавающее" окно „Презентация", позволяющее вставлять и изменять слайды, а также копировать их.

Слайд в виде черного текста на белом фоне смотрится некрасиво, поэтому для начала создайте фон презентации.

► Для задания фона слайдов щелкните по пустому месту на слайде и из контекстного меню выберите „Слайд" ⇒„Настройки страницы" ⇒„Фон" („Slide" ⇒„Page setup" ⇒„Background").

Можно залить фон одним цветом („Color"), градиентной заливкой („Gradient"), штриховкой („Hatching") или рисунком („Bitmap"). Справа вверху окна расположен образец — так будет выглядеть фон слайда после заливки. Внизу расположены элемены выбора цвета/типа заливки.

Выбрав нужную заливку, щелкните „ОК". Появится окно, спрашивающее, применить ли такой фон ко всем слайдам. Нажмите „Нет" („No"), если хотите оформить слайды по-разному.

► Не забывайте о том, что электронные презентации, в основном, демонстрируются на экране в затемненной комнате при помощи проектора, поэтому фон слайдов должен быть темным, а текст - светлым. Глаза менее утомляются, читая светлый текст на темном фоне.

Придумайте тему презентации, или выберите ее из списка: „Отдых в горах", „Виды спорта", „Мой колледж", „Я и мои друзья", „Каникулы моей мечты", „Немного про меня", „Выбор профессии".

На первом слайде напишите название презентации, добавьте какой-нибудь спецэффект для появления названия. Для этого щелкните по тексту левой кнопкой мыши, затем щелкните левой кнопкой по границе текстового поля для того, чтобы войти в режим редактирования не самого текста, а собственно поля. Щелкните по нему правой кнопкой мыши и из контекстного меню и выберите пункт „Эффекты” („Effects”). Появится окно настройки спецэффектов, в котором довольно легко разобраться. Из закладок вверху выбираем тип эффекта, затем из выпадающего меню, расположенном ниже, выбираем группу эффектов, а из нее — из самого большого поля в этом окне выбираем тип эффекта. Самое нижнее выпадающее меню позволяет выбрать скорость эффекта. Выбрав эффект, нажмите кнопку „Применить” („Assign”) — кнопка с зеленой галочкой — для присвоения эффекта элементу. Эффект можно просмотреть и не применяя, нажав кнопку „Просмотр” („Prewiew”), расположенную справа от кнопки „Применить”.

Если не закрывать окно „Эффекты”, можно применять спецэффекты и к другим элементам слайда, щелкая левой кнопкой мыши по этому элементу и присваивая ему какой-нибудь спецэффект.

► Окно „Эффекты” можно вызвать и из пункта меню „Слайд шоу” („Slide show”) ⇒ „Эффекты”.

Сделайте цвет текста таким, чтобы он хорошо контрастировал с фоном слайда. Для этого выделите текст, цвет которого вы хотите изменить, а затем либо из меню „Формат” ⇒„Символ” ⇒„Эффекты шрифта” („Format” ⇒‚Character” ⇒„Font Effects”), либо нажатием кнопки цветовых аттрибутов текста на панели инструментов выбрать цвет текста.

Добавьте в презентацию еще 7-10 слайдов (в вашей итоговой презентации должно быть не менее 8 слайдов).

► Перемещаться между слайдами можно при помощи клавиш <Page Up> и <Page Down>

Оформьте остальные слайды на свой вкус. Помимо текста вставьте в слайды изображения из галереи OO, либо из сетевой папки `pictures` на сервере `sgpk`. Обратите особое внимание на смысловую нагрузку презентации и эргономичное оформление слайдов.

Постарайтесь не перегружать презентацию спецэффектами, а также не размещать на одном слайде большого числа элементов. Элементы должны размещаться рационально, между ними должно быть свободное пространство (около 5% размера соответствующей стороны слайда).

Анимация не должна утомлять зрение (т.е. элементы не должны „скакать” – они должны перемещаться плавно).

5. 2. Создание автоматического показа презентации

Откройте пункт меню „Показ слайдов” („Slide Show”). Настройте автоматическую смену слайдов по времени — подпункт „Настройка презентации” („Slide Show settings) и вид смены слайдов — подпункт „Смена слайдов” („Slide Transition”).

► В окне „Настройка презентации” можно настроить много разных параметров. Обратите внимание на эти настройки.

Вид смены слайдов можно настроить отдельно для каждого слайда, также как и паузу между слайдами.

Сделайте задержку между появлением слайдов 3..5 секунд. Задержку спецэффектов для элементов слайдов создайте в диапазоне 1..3 секунд, чтобы можно было успеть прочитать текст слайда и разглядеть изображения. Не переусердствуйте со спецэффектами: для каждого слайда их должно быть не более 3..5, но и не пренебрегайте разнообразить спецэффекты. Например, если заголовок слайда вылетает сверху, текст слайда может выплывать снизу, а изображения – плавно появляться.

Можно настроить автоматическую смену слайдов и по-другому. Для этого зайдите в меню „Слайд шоу” ⇒„Временные интервалы” („Slide show” ⇒„Rehearse timings”). Запустится слайд шоу, а на экране появится таймер, щелкая на который через желаемый интервал времени, можно настроить соответствующую паузу между слайдами.

5. 3. Анимации и звуковые файлы в презентации

Добавьте к презентации несколько слайдов. Вставьте на один из слайдов анимацию (из галереи, сети, либо создайте ее самостоятельно с помощью Gimp).

Добавьте спецэффекты для появления и исчезновения анимации.

Для другого слайда настройте музыкальное сопровождение (вставьте в него аудиофайл). Можно сделать, чтобы аудиофайл начинал играть сразу после появления слайда, либо через некоторое время.

Вставьте в один из слайдов аудиофайл „Апплодисменты” (его можно найти в галерее).

Завершите оформление слайдов, проверьте функциональность презентации.

6. Графические редакторы.

6. 1. Растровые графические редакторы (Gimp)

Запустите растровый редактор, ознакомтесь с его меню.

Загрузите в редактор какую-нибудь фотографию (можно скачать по сети с компьютера sgpk по адресу `http://www.sgpk/pict`). Скачать изображение можно, щелкнув в браузере

(Konqueror или Mozilla) правой кнопкой на понравившееся изображение и выбрав из контекстного меню пункт „Сохранить как" („Save As").

Для того, чтобы открыть изображение в Gimp, необходимо выбрать в меню „Файл" („File") пункт „Открыть" („Open") или нажмите сочетание клавиш `"Ctrl"+ "O"`. В появившемся диалоговом окне выберите файл, который необходимо открыть.

Испробуйте различные спецэффекты. Ознакомьтесь с возможностями редактора.

Выполните с изображением следующие манипуляции:

- улучшите фокусировку в центре изображения и создайте дефокусировку на краях;
- замените цвет некоторых цветных участков изображения;
- создайте вокруг фотографии красивую рамку;
- разрежбте ее посередине, удалите одну половину, а копию второй отразите зеркально и вставьте на место первой;
- создайте некоторую размытость второстепенных участков изображения, сделайте более резкими основные участки;
- попробуйте сделать фотомонтаж из пары фотографий.

Поэкспериментируйте с другими спецэффектами. Отменить только что выполненное действие можно, нажав сочетание клавиш `"Ctrl"+ "Z"`.

Попробуйте сделать несколько слоев изображения, сделайте верхние слои прозрачными, или же создайте в них некоторые прозрачные области.

Посмотрите, что можно сделать с форографией по адресу http://www.sgpk/ $\Rightarrow$ Документы $\Rightarrow$ Примеры.

Сохраните полученное изображение в своей домашней папке.

6. 2. Векторные графические редакторы (OO Draw, QCad, GnuPlot, Xfig и т.п.)

Создайте векторное изображение в любом векторном редакторе. Поэкспериментируйте с градиентными заливками и трехмерными спецэффектами.

Если вы работаете в Open Office, поэкспериментируйте с трехмерными эффектами. В Linux существуют такие удобные векторные графические редакторы, как Xfig и Dia. Если вы не можете найти их в меню, нажмите сочетание клавиш `,,Ctrl'' + ,,F2''`, в появившемся окне наберите „dia" („xfig") и нажмите `Enter`.

Нарисуйте в векторном редакторе простую схему или чертеж, например, план расположения мебели в компьютерном классе.

Сохраните изображение в домашней папке.

7. Компьютерная модель и ее исследование.

Моделирование — метод познания, состоящий в создании и исследовании моделей. **Модели** позволяют представить *в наглядной форме* объекты и процессы, недоступные для непосредственного восприятия. Модели играют важную роль в проектировании и создании различных технических устройств, механизмов, электрических цепей и т.д.

Все модели можно разбить на два класса: *предметные* (материальные) и *информационные*. Предметные модели воспроизводят свойства объектов в материальной форме, а информационные — в образной или знаковой.

С помощью формальных языков строятся *формальные информационные модели* (математические, логические и др.), а с помощью математических понятий и формул строятся *математичевские модели*. Исследование модели становится более наглядным, если использовать ее **визуализацию** — представить модель в виде чертежа. схемы, изображения и т.п.

Информационные модели делятся на следующие типы:

- Табличные модели представляют собой перечень объектов и их свойств. Типичным примером такой модели является прейскурант, таблица истинности логической схемы и т.п. Визуализация такой модели наиболее наглядна в форме диаграммы или графика.
- Иерархические модели имеют иерархическую структуру: объекты делятся на классы, которые распределяются по степени подчинения друг другу. Наиболее наглядно и естественно представить такую модель в виде иерархического дерева (например, генеалогическое древо).
- Сетевые модели применяются для отражения систем со сложной структурой, в которых связи между элементами имеют произвольный характер. Визуализация такой модели имеет вид графа, в котором отражены взаимосвязи элементов модели в виде стрелок или прямых.

Для исследования модели можно использовать такой алгоритм:

1. Содержательная постановка задачи. Задание необходимых для составления модели условий и параметров.
2. Создание качественной описательной модели. Словесное описание модели и допущений, возможных при формализации.
3. Составление формальной модели. Запись всех необходимых уравнений и условий, составление набросков графиков и т.п.
4. Визуализация модели. Графическое отображение модели.

7. 1. Исследование математической модели.

Выполните задание (10):

Найти графическим методом корень уравнения $x^3 = \sin x$, не имеющее точного алгебраического решения.

Выполнять это задание следует, воспользовавшись следующим алгоритмом:

1. Создайте электронную таблицу Open Office Calc. Сохраните ее под названием `Lesson_7.sxi` в своей рабочей папке.
2. В самой первой ячейке поставьте `-1`, в ячейке под ней запишите формулу `=A1+0.05` и протяните эту ячейку за правый нижний угол до 41-й строки.
3. В ячейке **B1** запишите формулу `=A1`$\hat{3}$, а в **C1** — `=SIN(A1)` и тоже протяните их до 41-й строки.

4. Выделите всю получившуюся таблицу и выберите в меню „Вставить" („Insert") $\Rightarrow$ „Диаграмма" („Diagram").
5. В открывшемся диалоге выбора типа диаграммы поставьте флажок у пункта „Первый столбец как подпись" („First column as…"); нажмите „Далее" („Next").
6. Выберите тип диаграммы — „Линии" („Lines"); укажите, что ряды данных расположены в столбцах (Columns); нажмите „Далее".
7. Выберите „Обычная" („Normal"); нажмите „Далее".
8. Введите название диаграммы и укажите подписи осей; нажмите „Готово" („Ready").

На построенном графике найдите координаты точек пересечения двух кривых. Это и будут искомые корни уравнения.

7. 2. Исследование физической модели.

Выполните задание (11):

Построить физическую модель полета тела, брошенного под углом α к горизонту.

Постановка задачи: в процессе тренировки теннисистов используются автоматы для бросания мяча. Требуется задать автомату необходимую скорость и угол бросания мячика для попадания в определенное место площадки.

Качественная описательная модель. Здесь можно сделать следующие допущения:

- из-за малых, по сравнению с Землей, размеров мяча его можно считать материальной точкой;
- т.к. мячик взлетает на высоту, пренебрежимо малую по сравнению с радиусом Земли, можно считать, что на него действует постоянная сила тяжести, а его движение происходит равноускоренно с ускорением $g = 9.81$ м/с ^{2};
- скорость бросания мяча невелика, так что сопротивлением воздуха можно пренебречь.

Формальная модель. Опишем математически движение мяча. По оси X он движется равномерно, а по оси Y — равноускоренно. Проекции начальной скорости на координатные оси равны:

$$v_x = v_0 \cos\alpha; \qquad v_y = v_0 \sin\alpha.$$

Соответственно, координаты будут меняться по законам:

$$x = v_0 \cos\alpha \cdot t; \qquad y = v_0 \sin\alpha \cdot t - \frac{gt^2}{2}.$$

Когда мяч упадет на площадку, его координата y станет равна 0, следовательно, можно получить время полета мяча:

$$v_0 \sin\alpha \cdot t = \frac{gt^2}{2} \quad \Rightarrow \quad t_1 = 0, \quad t_2 = \frac{2v_0 \sin\alpha}{g}.$$

t_1 нас не интересует, т.к. это — время вылета мяча из автомата. Теперь, подставив найденное t в выражение для x, получим:

$$x = \frac{v_0^2}{g} \sin 2\alpha. \qquad (*)$$

Пусть аппарат обладает возможностью задать мячу одну из трех скоростей: $v_1 = 5$ м/с, $v_2 = 10$ м/с, $v_3 = 20$ м/с. Согласно формуле $(*)$, какими бы ни были углы наклона, для конкретной скорости v_0 расстояние, которое пролетит мяч, лежит в диапазоне $\left[0, \frac{v_0^2}{g}\right]$. Таким образом, если необходимо бросить мяч на расстояние x, вначале следует выбрать необходимую начальную скорость из условия

$$\frac{v_0^2}{g} \geqslant x, \quad \Rightarrow \quad v_0 \geqslant \sqrt{gx}.$$

Итак, если начальная скорость v_1, v_2 или v_3 удовлетворяет этим условиям, вторым шагом будет выбор угла наклона автомата. Следует учесть, что почти для всех точек таких углов будет 2: $\alpha_1 = 45° - \beta$ и $\alpha_2 = 45° + \beta$, где β — некоторый угол.

Вычислить угол можно из формулы $(*)$:

$$\alpha = \frac{1}{2} \arcsin \frac{gx}{v_0^2}.$$

Вполне естественно, что для некоторых расстояний будет возможным задать все три начальные скорости, и для каждой скорости задать по два угла, т.е. попасть в эти места можно при различных шести значениях скорости/угла наклона автомата.

Это задание также будем выполнять в OO Calc.

1. Создайте новый документ или используйте другой лист имеющегося документа.
2. В первый столбец занесите скорости: 5, 10 и 20 м/с. Выделите ячейку `A6` каким-нибудь цветом. В нее вы будете заносить требуемую дальность полета. В первую ячейку второго столбца занесите формулу `=IF(A1>=SQRT($A$6*9,81);A1;0)` и протяните ее вниз до третьей строки. Это — условие, проверяющее, сможет ли при данной начальной скорости мяч долететь до цели.
3. В первую строку столбца `C` занесите формулу `=0,5*ASIN(9,81*$A$6/(B1^2))*180/PI()`. Здесь будет расчитываться значение первого угла. Значение второго угла будет расчитываться в столбце `D`, куда следует занести формулу `=90-C1`. Выделите обе ячейки и протяните вниз до третьей строки.
4. В столбец `F` занесите значения времени от 0 до 4 с интервалом 0,05.
5. В столбцах `G` и `I` будем рассчитывать координату X при полете мяча с максимальной скоростью. Занесите в `G1` формулу `=$A$6*COS($C$3*PI()/180)*F1`. В `I1` занесите формулу `=$A$6*COS($D$3*PI()/180)*F1`.
6. В столбцах `H` и `J` будем рассчитывать координату У мяча. Занесите в ячейку `H1` формулу `=$A$6*SIN($C$3*PI()/180)*F1-4,905*F1^2`, а в ячейку `J1` — формулу `=$A$6*SIN($D$3*PI()/180)*F1-4,905*F1^2`.
7. Протяните первую строку от столбца `G` до столбца `J` до цифры 4 столбца `F`.
8. Теперь внесите в ячейку `A6` число 15 и постройте траекторию полета мяча для двух углов. Для этого выделите столбцы `G` и `H` и выберите в меню „Вставить" ⇒„Диаграмма". Укажите, что подписи оси X расположены в первом столбце. Выберите тип диаграммы „Линии", „Обычная". Аналогично поступите для столбцов `I` и `J`.
9. Теперь измените значение расстояния на 10 и посмотрите, как изменятся графики полета мяча. Для построения графика выделите колонки, в которуы занесены координаты X и У (например, `G` и `H`. Затем выберите пункт меню „Вставка → Диаграмма". В появившемся окне не забудьте поставить флажок „Первый столбец как подпись", далее выберите нужный тип диаграммы и нажмите „OK".

8. Основы программирования в shell.

Shell (оболочка) — командный процессор, принимающий команды пользователя, анализирующий их и выполняющий необходимые действия. Наиболее часто в Linux употребляется оболочка Bash (`sh`). Некоторые команды оболочки вы уже знаете, но бывают случаи, когда приходится по многу раз вводить одни и те же команды, меняя лишь малую их часть. Для упрощения работы пользователя в таких случаях используются командные файлы (**скрипты** shell).

8. 1. Простейшие скрипты.

8. 1.1.Подсчет файлов 1.

Для начала создадим скрипт, считающий количество файлов в текущей папке.

Создайте в своей рабочей папке файл `count` и сделайте его исполняемым (наберите команду `chmod 775 count`). Теперь откройте этот файл для редактирования любым текстовым редактором (если вы работаете в `mc`, наведите выделение на этот файл и нажмите клавишу F4, если командной строке — наберите, например, `kwrite count`).

В самом начале файла наберите строку

```
#!/bin/sh
```

Она необходима для того, чтобы командный процессор распознал данный файл как скрипт. Далее наберите непосредственно текст скрипта:

```
COUNT=-2                                    # Не считаем папки . и ..
for FILE in ‘ls -a‘                         # Начало цикла
do                                          #  по списку файлов в папке
    COUNT=‘expr $COUNT + 1‘                 # Увеличить счетчик на 1
done                                        # Конец цикла
echo "Папка содержит $COUNT файлов и папок"   # Вывод результата
```

- ► Обратите внимание, что ковычки, в которые заключены команды `ls` и `expr`, получаются при нажатии буквы ё (слева от 1) в английской раскладке. Они означают, что заключенный в них текст будет выполнен как команда, а результат ее — подставлен вместо кавычек.
- ► Знаком `#` отмечаются комментарии, все, что написаоно в строке после него, не будет считано командным интерпретатором, поэтому, в принципе, комментарии можно не записывать. Они нужны только для лучшего понимания скрипта. Единственный комментарий, который обязательно следует записать — указание командной оболочке, что для исполнения файла необходимо использовать bash (`#!/bin/sh`).

Здесь `COUNT` — счетчик, который будет содержать количество файлов и папок в текущей папке. Далее в программе следует цикл, в котором переменная `FILE` принимает все значения из листинга текущей папки. На каждый файл переменная `COUNT` увеличивается на 1.

Знак `$` перед именем переменной позволяет получить ее значение, а команда `expr` представляет собой простейший калькулятор командной строки, возвращая значение выражения, стоящего после нее.

Набрав и сохранив программу, выполните ее, для чего в `mc` просто наведите на нее выделение и нажмите Enter, а в командной строке ее можно выполнить двумя способами: набрав либо `sh count`, либо `./count`.

8. 1.2.Подсчет файлов 2.

Теперь заставим нашу программу не просто считать файлы и папки вмести, а определять отдельно количество файлов, папок и исполняемых файлов. Для этого нам понадобится три счетчика. В начале программы обнулим счетчики файлов и исполняемых файлов, а счетчику папок присвоим значение -2, чтобы не считать папки "."и "..".

Кроме того, сделаем так, чтобы программа могла не только считать файлы в текущей папке, но и в любой другой, указанной в параметре команды.

- ▶ **Параметром** команды называют текст, который идет после ее имени. Обычно параметры используются для того, чтобы задавать различные варианты исполнения команды, либо передавать ей какие-либо данные. Например, в предыдущем тексте для вывода списка всех файлов данной папки мы передали команде `ls` параметр `-a`. Для того, чтобы просмотреть содержимое не текущей папки, необходимо передать ее имя как параметр команды `ls`: например, `ls /usr/lib`.
- ▶ **Ключом** команды называют параметр, влияющий на ее выполнение. Обычно все ключи начинаются со знака `'-'` или же `'--'`.

Организуем цикл по списку файлов и папок в данной папке, а затем будем проверять каждый элемент этого списка, прибавляя к соответствующему счетчику 1. Проверка типа выполняется с помощью условного оператора `[]`, проверяющего истинность значения, заключенного в скобках. Оператор `&&` выполняет следующую за ним команду, если предыдущая вернула истинное значение. Каждому типу файлов соответствует свой флаг проверки: `-f` — для любых файлов, `-d` — для папок, `-x` — для исполняемых файлов. Когда будет необходимо применить сложную проверку, будем пользоваться операторами `if` — `then` — `else` — `fi`, означающими „если" — „то" — „иначе" — „конец проверки".

- ▶ Цикл в скриптах bash организуются командой `for`, после которой следует имя переменной, в которую будут заноситься очередные значения цикла, затем — слово `in`, означающее, что значения будут браться из последующего списка, и, наконец, сам список значений, чаще всего его заменяют выводом какой-нибудь программы, как мы и поступим.

После инициализации переменных проверим, ввел ли пользователь имя папки, в котрой необходимо подсчитать количество файлов. Если это имя не введено, будем считать файлы в текущей папке.

- ▶ Доступ к параметрам, сообщенным скрипту в командной строке, осуществляется так же, как и доступ к значениям переменных, за тем исключением, что каждый параметр обозначается цифрой, соответствующей его номеру. Нулевой параметр соответствует названию команды. Пример: `$2` означает доступ ко второму параметру введенной команды.

После завершения цикла выведем содержимое счетчиков на экран.

```
#!/bin/sh

countf=0                    # счетчик файлов
countd=-2                   # счетчик папок
countx=0                    # счетчик исполняемых файлов
```

```
[ "$1" != "" ] && dir="$1/" # по умолчанию будем считать в текущей папке
for file in `ls -a $1`
do
    [ -f "$dir$file" ] && countf=`expr $countf + 1`
    [ -d "$dir$file" ] && countd=`expr $countd + 1`
    [ -x "$dir$file" -a ! -d "$dir$file" ] && countx=`expr $countx + 1`
done
echo -e "count files in $PWD/$1:\n$countd\tfolders\n$countf\tfiles\n$countx\texecutes"
echo "Total: `expr $countf + $countd + $countx` files and folders"
```

► Параметр `-e` говорит команде `echo`, что она должна учитывать специальные параметры, управляюие текстом: `\n` — начать новую строку, `\t` — вставить символ табуляции.

Набрав и сохранив программу, выполните ее, для чего в `mc` просто наведите на нее выделение и нажмите Enter, а в командной строке ее можно выполнить двумя способами: набрав либо `sh count`, либо `./count`.

8. 2. Более сложные скрипты.

8. 2.1.Сбор файлов из разных папок в одну 1.

Иногда бывает необходимо собрать множество файлов, расположенных в разных папках, в одно место. Создадим программу, осуществляющую такие действия. Это полезно, например, при создании фотоальбома с цифрового фотоаппарата.

Так как при чтении списка файлов программа будет регулярно находить папки, файлы из которых надо извлечь, нам придется вызывать эту программу рекурсивно, то есть из самой себя, передавая параметром папку, в которую надо „складывать” файлы. Таким образом, у нашей команды обязательно должен будет быть один параметр. По умолчанию „складывать” файлы будем в папке GATHER, поиск будем начинать с текущей папки.

Вначале проверим, является ли параметр нашей команды полным именем, то есть проверим, начинается ли оно со знака /. Для этого нам поможет команда `grep`, показывающая все вхождения заданного ей параметра в файле или входных данных. Данные на вход команды `grep` перенаправим при помощи *конвейера* — оператора `|`, который переводит стандартный вывод (выходные данные) первой программы на стандартный вход второй (входные данные).

► **Полным именем** (путем) файла называется весь путь к файлу, начиная с корневой папки. Например, `/var/log/messages/1.msg` — полное имя файла, `messages/1.msg` — его имя (путь к нему) относительно папки log, а `1.msg` — краткое имя файла (или же — собственно имя файла).

При проверке имя папки сохраним в переменной tmp. Если оно является полным, присвоим его переменной where, в которой будет храниться полный путь к искомой папке. Если имя является неполным, сделаем его таким, подставив префиксом текущую папку (она хранится в переменной `PWD`), а если gather была запущена вообще без параметров, присвоим переменной where GATHER.

Файлы будем копировать командой `cp`, если вы хотите, чтобы файлы не копировались, а перемещались, необходимо заменить ее на команду `mv`.

```
#!/bin/sh

tmp=`echo "$1" | grep "^/"`                          # Проверка, является ли
if  [ "$tmp" != "" ]; then                           #  параметр $1 полным именем
    where="$1"                                       #  файла.
else                                                 # Если не является, то
    if [ "$1" != "" ]; then                          #  если он не пустой
        where="$PWD/$1"                              # приводим его к полному виду,
    else                                             #  иначе -
        where="$PWD/GATHER"                          #  будем складывать в GATHER.
    fi                                               # Конец проверки
fi                                                   # Конец проверки
[ ! -e $where ] && mkdir $where                      # Если папки назначения нет,
                                                     #  создаем ее.
for file in `ls`                                     # Цикл по списку файлов в
do                                                   #  текущей папке (без скрытых).
    if [ "$PWD/$file" != "$where" -a -d "$file" ]; then # Если наткнулись на папку,
        cd $file                                     #  входим в нее
        gather $where                                #  и собираем из нее файлы,
        cd ../                                       #  а потом выходим назад,
        continue                                     #  прерывая цикл, чтобы
    fi                                               #  не обработать случайно папку
    cp $file $where                                  # Копируем файл в место назначения
done                                                 # Конец цикла
```

8. 2.2.Сбор файлов из разных папок в одну 2.

Теперь усложним нашу программу: сделаем так, чтобы она собирала только картинки jpg и переименовывала их, начиная с 1. Для этого нам понадобится счетчик, в котором будет находиться номер текущей картинки, назовем его `count`. Кроме того, т.к. программа работает рекурсивно, необходимо передавать дочерним процессам (вызываемым основной программой) номер текущей картинки (будем передавать его вторым параметром). Основная программа также должна знать, до какого номера „досчитался" дочерний процесс, т.е. он должен возвращать текущее значение счетчика после завершения своей работы (сделаем это при помощи команды `echo $count`, выводящей в конце программы текущее значение счетчика).

Увеличивать значение счетчика будем опять при помощи команды `expr`. Проверить, является ли файл картинкой jpg можно по его расширению при помощи команды `grep`, перенаправив на ее стандартный вход имя файла, выведенное командой `echo`. Будем искать вхождение выражения `.jpg` в конец имени файла, для этого используем знак `$` в конце выражения, означающий конец строки (начало строки обозначается знаком `^`). Кроме того, следует учесть, что имя файла может быть не только в нижнем регистре, но и в верхнем, т.е. следует проверять как малые, так и большие буквы в расширении. Для того, чтобы подставить одно из нескольких значений в строчном выражении, их пишут в квадратных скобках, не разделяя запятой.

```
#!/bin/sh
```

```
if [ "$2" != "" ]; then           # Если есть второй параметр,
    count="$2"                    #  присваиваем счетчику его значение,
else                              # иначе
    count=1                       # присваиваем счетчику 1.
fi
tmp=`echo "$1" | grep "^/"`
if  [ "$tmp" != "" ]; then
    where="$1"
else
    if [ "$1" != "" ]; then
        where="$PWD/$1"
    else
        where="$PWD/GATHER"
    fi
fi
[ ! -e $where ] && mkdir $where
for file in `ls`
do
    if [ "$PWD/$file" != "$where" -a -d "$file" ]; then
        cd $file
        count=`gather $where $count`  # Присвоим счетчику значение дочернего процесса.
        cd ../
        continue
    fi
    tmp=`echo "$file" | grep ".[jJ][pP][gG]$"`  # Проверим, картинка ли наш файл
    if [ "$tmp" != "" ]; then                  # если картинка, то
        cp $file $where/$count.jpg             # скопируем ее в папку назначения,
        count=`expr $count + 1`                # и увеличим на 1 значение счетчика.
    fi
done
echo $count                                    # В конце выведем текущее значение счетчика.
```

Выполните задание (12):

Наберите этот скрипт и посмотрите, как он работает.

9. Основы программирования на языке С.

9.1. Знакомство с языком С.

9.1.1.Введение.

Для написания компьютерных программ используется большое число языков программирования. Для того, чтобы запустить написанную на одном из них программу, следует

транслировать ее в понятные компьютеру коды. Программ-трансляторов намного больше, чем языков программирования, каждый программист использует тот транслятор, который ему больше нравится и лучше всего удовлетворяет требованиям к готовой программе.

Трансляторы делятся на **компиляторы** и **интерпретаторы**.

Интерпретаторы представляют собой трансляторы реального времени, читающие программный код строка за строкой, анализирующие его и передающие операционной системе необходимые команды. Примером языка программирования, использующего интерпретатор для выполнения программ является BASIC. Недостатки таких программ в том, что для их запуска всегда необходимо запускать в начале программу-интерпретатор, кроме того, на анализ кода тратится большое количество времени, что сильно замедляет выполнение программы.

Более „продвинутыми" являются языки программирования, использующие компиляторы. Компилятор анализирует программный код и создает **исполняемый файл**, содержащий **машинные коды**, понятные операционной системе. Исполняемый файл не нужно анализировать в ходе его выполнения — он загружается в память компьютера и выполняется ОС сразу, что значительно повышает скорость выполнения программы.

Бо́льшая часть кодов всех операционных систем написана на языках C и C++. C первым мы и познакомимся. Второй представляет собой усовершенствованный вариант языка C, значительно упрощающий программисту написание сложных программ.

Файлы, содержащие текст программ на языке C, обычно имеют расширение c, а на языке C++ — cpp, cc или c++. Кроме них существуют **заголовочные файлы**, имеющие расширение h (в C++ — hpp, H или h++). Заголовочные файлы содержат информацию о функциях и переменных, хранящихся в соответствующих c файлах.

9. 1.2.Структура файлов на языке C.

Для трансляции программного кода, написанного на C и C++ в системе Linux используется компилятор `gcc` (аббревиатура, означающая GNU C compiler — компилятор языков C и C++, написанный по лицензии GNU).

Программы на языке C имеют модульную структуру, где каждый модуль заключен в фигурные скобки `{}`. Модулями могут быть как функции, так и тела циклов, операторов выбора и пр.

Функции на языке C имеют следующий вид:

```
TYPE FNAME(ARGLIST){
    FBODY;
    return RET;
}
```

где `TYPE` — **тип данных** `RET`, возвращаемых функцией; `FNAME` — имя функции, оно должно начинаться с буквы или символа `_`, может содержать латинские буквы (компилятор различает прописные и строчные буквы, так что `myFn` и `myfn` — разные имена функций), цифры и символ `_`; `ARGLIST` — **список аргументов**, представляющий собой набор переменных (и описаний их типов), которые должны быть переданы в функцию и/или возвращены ею (если не хватает одного возвращаемого значения `RET`; `FBODY` — **тело функции**, представляющее собой набор команд, которые должна выполнять эта функция.

Все функции в программе разделяются знаком „ ; ", который говорит компилятору, что действие данного оператора закончилось.

Главной функцией программы является функция `main`, возвращающая обычно целое значение `int`. Если компилятор не находит такой функции, он выдает ошибку. Функция `main` может быть единственной в программе, а может и вызывать другие функции.

Более широким является понятие **оператора**, представляющего собой указание ОС совершить какое-то действие. Оператором может быть и функция, и, например, арифметические операции (`+, -, *, /`). Существует множество различных операторов, часть из них вы изучите.

Типы данных. Язык C использует много типов данных, нас будут интересовать следующие:

`int` — целое число;

`float` — число с плавающей точкой (дробное);

`char` — „буква” — целое число, занимающее 1 байт, обычно его используют для хранения букв;

`void` — значение — ничто, этот тип используется для обозначения функций, не возвращающих никакого значения и в некоторых других случаях;

`TYPE*` — указатель на тип `TYPE`, представляющий собой адрес, по которому в памяти размещены данные; чтобы ОС знала, как трактовать эти данные, перед указателем и указывается их тип.

Ответьте на вопрос(ы):

> В языке Pascal помимо функций существует понятие процедуры, выполняющей какое-либо действие, но не возвращающей никаких значений. Как будет выглядеть запись процедуры на языке C?

Чтобы использовать в своей программе переменные, необходимо заранее дать компилятору знать о них, **объявив их** до их использования. Объявление переменной имеет вид, подобный `int A1`, означающий, что далее будет использоваться целая переменная `A1`. Если объявить переменную вне всех функций, она будет **глобальной**, т.е. ее значение во всех функциях программы будет одним и тем же. Если же объявить ее внутри какого-то модуля, при завершении его программа сотрет эту переменную, поэтому ее называют **локальной**.

Если необходимо объявить несколько переменных, их можно перечислить через запятую: `float a, b, c;` эквивалентно записи `float a; float b; float c;`.

Библиотечные функции. В стандартном C существует множество функций, но все же их обычно недостаточно для совершения всего многообразия операций, которое может понадобиться программисту. Бóльшую часть необходимых функций содержат **библиотеки**, представляющие собой уже скомпилированные файлы, содержащие определенный набор функций. Чтобы выполнить такую функцию, необходимо подключить данную библиотеку, чтобы в процессе компиляции специальная программа (linker) включила в код нашей программы код, содержащий данные функции.

Подключение библиотек осуществляется путем указания компилятору ее имени, например, чтобы подключить математическую библиотеку, при компиляции надо указать `gcc` *Имя программы* `-lm`, т.е. поставить ключ `-l`*Имя библиотеки*.

Каждая библиотека содержит набор большого количества функций, собранных по пакетам, подключение которых осуществляется при помощи специальной записи в начале файла, имеющей вид `#include` <*Имя заголовочного файла*> для стандартных библиотек или `#include "`*Имя заголовочного файла*`"` для подключаемых файлов, созданных пользователем (т.е. большой проект можно разбить на несколько мелких файлов и указать главному файлу, что их надо подключить на завершающем этапе компиляции — связке).

► Стандартная библиотека C подключается по умолчанию, поэтому в простых программах не надо указывать колюч `-lstd`.

Основные функции, необходимые для преобразования типов данных и некоторых других действий, содержатся в модуле `stdlib`. Модуль `stdio` позволяет работать с файлами, и, как следствие, многими устройствами (которые, как вы помните, представляются в ОС Linux как файлы), а также с консолью (набранный вами текст попадает в файл стандартного ввода, а отображаемый на экране получается из файла стандартного вывода). Для математических вычислений используется библиотека `math`. Это — основные библиотеки, которые нам понадобятся. Кроме них существует множество специальных библиотек, но для познания основ C они не обязательны.

Конечно, хорошим вариантом могло бы показаться собрать все библиотечные файлы в одну библиотеку и подключать ее ко всем компилирующимся программам, но в этом случае размер полученного исполняемого файла был бы огромен.

9. 2. Простейшие программы.

9. 2.1.Hello World! (Здравствуй, мир!).

Итак, приступим к познанию C со стандартной программы, выводящей на экран надпись "Hello World!". В C существует множество операторов вывода, в данном случае нам больше всего подойдет `printf` (вывод на экран по формату). Если вы хотите узнать что-либо по конкретной стандартной функции, вызовите ее man-страничку командой `man` <команда> (так же, как это можно узнать для любой команды Linux).

Создайте в своей рабочей папке папку „C". Создайте в ней текстовый файл `HelloWorld.c` и откройте его любым редактором (в Bash можно, войдя в эту папку, сразу набрать `mcedit HelloWorld.c` или `kedit HelloWorld.c`, тогда редактор сам создаст нужный файл и откроется для его редактирования.

► Создание папки — `mkdir` *Имя*, создание файла — `touch` *Имя*.

Для того, чтобы вывести текст на экран монитора, нам понадобится только одна библиотека `stdio`, которую мы объявим в начале файла. Программа выполняет простейшие действия, т.е. она будет содержать одну-единственную функцию `main`, которая вызовет функцию вывода на печать и завершится:

```
#include <stdio.h>              // Подключим стандартные функции ввода-вывода

int main(){                     /* Главная функция */
    printf("Hello, World!\n"); // Выведем на экран надпись ,,Привет, мир!''
    return 0;                   // Выход из программы
}
```

► Обратите внимание на то, что в языке C комментариями является весь текст строки, стоящий после символа двойного слэша (`//`), а также весь текст, независимо от количества занимаемых им строк, стоящий между символами `/*` и `*/`.

Запись `\n` говорит о том, что после фразы „Hello, World!" необходимо вставить знак переноса строки, чтобы после окончания программы приглашение Bash не оказалось на одной с ней строчке.

► Программа после своего выполнения должна возвращать 0 или положительное число, что говорит о нормальном ее завершении, если в программе произошла ошибка, возвращается код ошибки (отрицательное число).

Выполните задание (13):

Составьте блок-схему программы.

Набрав и сохранив текст программы, нам следует откомпилировать его. Можно просто набрать `gcc HelloWorld.c`, тогда компилятор создаст исполняемый файл `a.out`, но лучше указать компилятору ключ `-o`, который поместит готовую к выполнению программу в файл, указанный после ключа:

```
gcc HelloWorld.c -o Hello
```

создаст файл Hello. Запустите его командой `./Hello`.

► Указывать `./` перед именем программы нужно для того, чтобы Bash знала, что данный файл нужно искать в текущей папке, а не в папках, в которых по умолчанию хранятся программы.

9. 2.2.Ввод, условный оператор.

Обычно программа должна выполнять какие-то действия в ответ на действия пользователя или введенные им данные. Ввести данные в программу можно как при ее запуске (параметрами командной строки), так и в процессе ее выполнения при помощи функций `getchar`, `scanf` и им подобных.

Функция `getchar` считывает одну букву и возвращает ее код, функция `scanf` имеет такой формат: `scanf("формат список переменных)`, где `формат` — описание типа данных, вводимых пользователем (например, `%d` означает целое число), `список переменных` — разделенный запятыми список переменных, которые должен ввести пользователь (на каждую переменную должен быть указан ее формат, форматы разделяются через пробел).

► Более подробно о стандартных функциях можно узнать из их man-страниц (`man функция`).

Составим программу, запрашивающую у пользователя 2 числа и сообщающую, какое из них больше, или же они равны. Для этого нам понадобится еще и изучить условный оператор, выполняющий определенные действия при выполнении/невыполнении какого-либо условия.

В C структура условного оператора выглядит так:

```
if (Условие) действия1;
else действия2;
```

Если Условие истинно, выполняются действия1 (если их много, следует заключить их в блок `{}`), иначе выполняются действия2.

Выполните задание (14):

Составьте блок-схему алгоритма программы. Если возникнут затруднения, внимательно просмотрите ее текст (ниже).

Теперь создайте файл `if.c` и наберите в нем текст программы.

```
#include <stdlib.h>
int a,b;                                                    // Резервируем память для переменных

int main(){
    printf("Введите значения a и b (через пробел)\n");
    scanf("%d %d",&a,&b);                                   // Запрашиваем данные
    if(a>b) printf("a больше b\n");                         // И проверяем условие
    else{
        if(a<b) printf("b больше a\n");
        else printf("a и b равны\n");}
    return 0;
}
```

Обратите внимание нв знак `&` перед именем переменной. Он означает, что функции передается значение адреса этой переменной, а не ее копия. Если бы мы передавали в функцию саму переменную, то введенное пользователем значение останется внутри функции, а сами переменные не изменятся.

Откомпилируйте файл и запустите его. Проверьте, как он работает при различных вариантах введенных значений.

9. 2.3.Префиксные и постфиксные операции. Циклы.

Допустим, у нас есть какой-то счетчик `i` и нам надо увеличивать его на 1 каждый раз, когда происходит какое-то событие. Можно, конечно, записать `i = i + 1;`, но более наглядно (да и короче) было бы написать, используя оператор **инкремента** `++`: `++i;`. Кроме того, существует еще оператор **декремента** `--`, уменьшающий на 1 значение счетчика: `--i;`.

Для сокращения записей можно использовать операторы `+=`, `-=`, `*=`, `/=` и `%=`. Запись `x  = y` означает то же самое, что `x = x  y`, где знак — одна из операций `+ - * / %`. Оператор `%` возвращает остаток от деления первого числа на второе.

Операторы инкремента и декремента можно ставить как перед переменной (**префикс**), так и после нее (**постфикс**). Важно понять разницу в префиксных и постфиксных операциях. Для этого напишем программку, выдающую числа и результаты префиксных и постфиксных операций с ними. Назовите ее `prefpost.c`. Разберитесь с текстом программы самостоятельно.

```
#include <stdlib.h>
int x,y;

int main(){
    x = 1; y = 1;
    printf("x = %d, y = %d\n", x, y);           // x=1, y=1
    printf("++x = %d, y++ = %d\n", ++x, y++);  // x=2, y=1
    printf("x = %d, y = %d\n", x, y);           // x=2, y=2
    printf("--x = %d, y-- = %d\n", --x, y--);  // x=1, y=2
    printf("x = %d, y = %d\n", x, y);           // x=1, y=1
    return 0;
}
```

Выполните задание (15):

Наберите программу, откомпилируйте ее и посмотрите результат выполнения.

► Здесь вы заметили, что `printf` имеет такой же формат, как и `scanf`, с той лишь разницей, что первая функция выводит данные на дисплей, а вторая — считывает их с него.

По результатам программы видно, что префиксная операция выполняется сразу же, а постфиксная только после выполнения определенных действий с этой переменной.

Циклы. Их предназначение — в многократном выполнении однообразных действий вплоть до выполнения определенных условий.

Для выполнения многократных вычислений с изменяющейся с определенным шагом переменной используется цикл `for`, а для действий, завершающихся лишь после выполнения какого-либо условия — цикл `while`.

Цикл `for` имеет следующий вид: `for(Нач.действия; Усл.; Операция){ Тело цикла }`, где Нач.действия — действия, которые выполняются при самом первом запуске цикла, чаще всего это присваивание переменной цикла определенного значения; Усл. — условие, которое проверяется каждый раз в конце цикла, если оно не выполняется, цикл завершается; Операция — действия, производящиеся каждый раз после невыполнения условия Усл.; Тело цикла — действия, выполняющиеся в цикле. Любой из параметров в скобках цикла может отсутствовать (даже одновременно все три).

Итак, создадим программу, которая будет вычислять факториал числа n, введенного пользователем.

► Факториал n ($n!$) — произведение всех чисел от 1 до n.

Первый вариант `factor1.c` создадим по всем правилам организации цикла, т.е. вначале запросим пользователя ввести число n, затем организуем цикл от 1 до n, в котором будем умножать переменную-ответ на счетчик цикла.

Выполните задание (16):

Составьте блок-схему программы. Вы на собственном опыте убедитесь, что для одной и той же блок-схемы можно создать множество реализаций программы. Наберите, откомпилируйте и посмотрите результаты выполнения всех вариантов программы.

Итак, первый вариант:

```
#include <stdio.h>
int n;

long factor(int n){
    int i;
    long f = 1;
    for(i = 2; i <= n; i++) f *= i;
    return f;
}

int main(){
    printf("Введите значение n\n");
    scanf("%d", &n);
    printf("Факториал %d равен %d\n", n, factor(n));
    return 0;
```

```
}
```

Здесь вводится тип данных `long`, позволяющий вместить большее число, чем тип `int`. Факториал — быстрорастущая функция, поэтому уже при $n = 32$ наша функция вернет неверное значение из-за того, что ответ будет больше, чем может вместиться в тип `long`.

Запустите программу и посмотрите, как она работает.

Теперь модифицируем нашу функцию `factor`. Скопируйте файл `factor1.c` в `factor2.c` командой `cp factor1.c factor2.c` или же сохраните его в текстовом редакторе под новым именем.

Сделаем так, чтобы факториал вычислялся в теле функции. Воспользуемся тем, что в разделе Операция цикла for можно писать любую функцию.

```
long factor(int n){
    long f = 1;
    for(; n > 0; f *= i--);
    return f;
}
```

Теперь мы воспользовались тем, что в функцию передается копия аргумента, т.е. внутри нее мы можем делать с ним все, что угодно.

Цикл `for(; n > 0;)` эквивалентен циклу `while(n > 0)`. Сделаем еще одну модификацию, использующую этот цикл:

```
long factor(int n){
    long f=1;
    while(n > 0) f *= i--;
    return f;
}
```

► Досрочно можно выйти из цикла командой `break`, а заставить его перейти к началу — командой `continue`. Зная английский язык легко можно разобраться с программами.

Выполните задание (17):

Напишите программу, которая будет считать сумму квадратов первых n чисел. Сначала составьте ее блок-схему. Подумайте, сколько переменных понадобится для программы. Не забывайте, что нужно по крайней мере две переменных: для хранения введенного числа n и результата вычисления. Подумайте, нужна ли отдельная переменная для хранения счетчика цикла (посмотрите предыдущие примеры).

► Для вычисления степени числа используется оператор ^. Например, чтобы вычислить x^2, необходимо записать $\mathtt{x}\hat{\mathtt{2}}$. Кроме того, для небольших степеней можно использовать вместо оператора возведения в степень многократное умножение (`x*x`).

► Обратите внимание на представление целых чисел в памяти компьютера. Если результат вычисления будет очень большим (например, когда вы введете 10000, на экране отобразится отрицательное число. Это не ошибка. Просто старший бит числа отражает его знак, а для хранения большого числа может понадобиться заполнить знаковый бит. Чтобы результат отображался всегда правильно, необходимо указать, что данная переменная беззнаковая, указав в ее определении `unsigned`, например, для беззнакового целого `unsigned int`.

9. 2.4.Операции с файлами.

Для операций с файлами существует структура `FILE`. Команда `fopen(char* n, char* m)` (`n` — имя файла (указатель на символьную строку, `m` — режим доступа к файлу — см. `man fopen`) возвращает указатель на эту структуру, который будет использоваться в командах ввода/вывода `fscanf` и `fprintf`. Синтаксис последних команд аналогичен `scanf` и `printf` за исключением того, что первым их аргументом идет указатель на структуру `FILE`.

► **Указатель** — это адрес переменной в памяти компьютера. С указателями следует обращаться осторожно, чтобы не уйти за пределы дозволенной области. Указатели очень удобно передавать вместо аргументов функций, если они занимают много места в памяти, т.к. на аргумент выделяется столько места, сколько он занимает, а указатель требует всего 2-4 байта. Чтобы получить по указателю содержимое данной величины, используется оператор `*`, а чтобы получить адрес переменной — оператор `&`.

В этом задании вы научитесь совершать простые операции с файлами, кроме того, узнаете как передавать данные в программу посредством аргументов командной строки.

Функцию `main` мы всегда записывали, опуская аргументы. На самом деле, полный ее формат — `int main(int argc, void **argv)`, где `argc` — количество аргументов функции, `**argv` — указатель на указатель на структуру, хранящую аргументы командной строки. Доступ к указателю можно получить как через элемент массива (`char* name = argv[2]`), так и путем разыменования указателя (`char* name = *argv++`). Тип `void*` (указатель на ничто) необходимо явным образом преобразовать к любому необходимому типу данных, указав, например, `int main(int argc, char **argv)`.

В нашей программе будет использоваться **массив** — индексируемый набор данных одного типа. В нем будет храниться имя файла, с которым будет работать программа. Объявление `char name[128]` означает, что в памяти выделится блок в 128 байт (т.к. `char` занимает 1 байт), а его имя является указателем на первый элемент (`name[0]`). Обратите внимание, что нумерация массивов начинается с 0, а не с 1.

Итак, напишем программу, которая будет открывать файл (имя которого можно ввести как первый аргумент программы), записывать в него оставшиеся аргументы (если таковые есть), затем спрашивать у пользователя радиус круга и рисовать этот круг в том же файле.

После работы с файлом его обязательно нужно закрыть при помощи функции `fclose`. Вообще-то, при выходе из программы будет очищена вся память и закрыты все ее файлы, но лучше закрывать их явным образом, чтобы в них не попал какой-нибудь „мусор”.

Выполните задание (18):

Наберите данную программу (назовите ее `file.c`). Т.к. программа использует математическую функцию `sqrt`, возвращающую квадратный корень ее аргумента, необходимо сообщить компилятору, что объявление этой функции находится в файле `math.h`. Кроме того, этот файл не входит в стандартную библиотеку, поэтому компилятору нужно указать явно, что необходимо подключить математическую библиотеку ключом `-lm` (ключ пишется до имени файла с программой).

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

FILE *f;
char name[128];
```

```
void printstar(int r, int n){
    int i;
    fprintf(f, "   ");
    for (i = 0; i < n; i++) fprintf(f, " ");
    if (n != r){
        fprintf(f, "*");
        for (i = 2*(r-n)-1; i > 0; i--) fprintf(f, " ");
    }
    fprintf(f, "*\n");
}

int main(char argc, char** argv){
    if (argc < 2){
        printf("Введите имя файла для записи\n");
        scanf("%s", name);
         }
    else strcpy(name, argv[1]);
    if ((f = fopen(name, "w")) == NULL){
        printf("Ошибка открытия файла\n");
        return -1;
        }
    fprintf(f, "Мой файл:\n");
    int a = 1;
    while (++a < argc) fprintf(f, "%s\n", argv[a]);

    unsigned char r;
    printf("Введите радиус круга\n");
    scanf("%d",&r);
    for(a = r; a > 0; a--) printstar(r, r - sqrt((double)(r*r - a*a)));
    for(a = 0; a <= r; a++ ) printstar(r, r - sqrt((double)(r*r - a*a)));
    fclose(f);
    return 0;
}
```

Скомпилируйте программу (`gcc -lm file.c -o file`). Запустите ее сначала без аргументов (`./file`), а затем — с несколькими аргументами (`./file myfile arg1 arg2 arg3`). Посмотрите выходной файл (в первом случае он будет носить имя, которое вы ему дадите в диалоге с программой, а во втором его именем будет первый аргумент командной строки).

Ответьте на вопрос(ы):

- Почему „круг” вытянут вдоль вертикальной оси?
- Почему для присваивания имени `name` первого аргумента командной строки использовалась функция `strcpy`, побуквенно копирующая строки, а не более короткая запись `name=argv[1]`? (подсказка: `name` — статический указатель, жестко привязанный к массиву.)
- Что будет, если в функции `printstar` заменить `fprintf(f, ...)` на `printf(...)`?

- Какова роль функции `printstar`?

Выполните задание (19):

Теперь скопируйте файл с текстом программы в файл `file1.c`, откройте этот файл в текстовом редакторе, и измените его так, чтобы функция печатала в файл сумму квадратов целых чисел от 1 до n, беря значением n первый аргумент командной строки, а именем файла — второй аргумент. Если же программа была запущена без аргументов, число n и имя файла следует запросить у пользователя.

Для получения числа используется функция `atoi` (напр., `int i = atoi(argv[1])`).

10. Программирование на C. Создание интерактивных консольных программ.

Теперь, когда мы научились составлять простые программы, попробуем более сложные. Для взаимодействия пользователя и компьютера используется множество элементов. Самый простой способ — консольный ввод/вывод: программа запрашивает у пользователя необходимые данные, которые тот набирает на клавиатуре. Более сложный — графический. Для запуска графических программ требуется графическая оболочка (X), а также подключение к программе графических библиотек. Программирование графических программ несколько сложнее, чем консольных.

10. 1. Простейшие интерактивные программы.

10. 1.1.Приветствие пользователя.

Выполните задание (20):

Создайте программу, запрашивающую у пользователя его имя и приветствующую его по имени. Для начала составьте блок-схему программы, после этого — приступайте к ее написанию.

Чтобы упростить себе работу, используйте готовые блоки из предыдущих программ.

► Редактировать несколько файлов сразу можно при помощи редактора `gedit`: зайдите в свою рабочую папку с файлами и наберите в командной строке `gedit *.c`. Откроется редактор, в котором каждый файл можно редактировать, активировав соответствующую вкладку.

Итак, для имени вам понадобится символьный массив длиной около 128 символов, ввод имени будет осуществляться при помощи команды `scanf`, а вывод данных — `printf`.

10. 1.2.Приветствие пользователя 2.

Выполните задание (21):

Наверное, вы заметили, что при вводе имени при помощи функции `scanf` ввод производится до первого пробела, т.е. фамилию, имя и отчество так ввести не получится. Функция `fgets` позволяет вводить данные до символа „конец файла” или символа новой строки (enter). Формат команды таков: `fgets(char* BUFFER, int NUM, FILE *NAME)`, где `BUFFER` — указатель на символьный массив (его длина должна быть не меньше `NUM`), `NUM` — максимальное количество введенных символов, `NAME` — указатель на файл, с которого следует производить считывание.

В Linux при запуске программы сразу открывается три файла: `stdin` (ввод), `stdout` (вывод) и `stderr` (поток ошибок). Для ввода имени вам необходимо считывать с файла `stdin`, который и надо будет указать третьим аргументом. Итак, необходимо будет строку считывания имени заменить на `fgets(name, sizeof name, stdin)`, если символьный массив вы назвали `name`.

10. 2. Файловые операции.

Создадим программу, позволяющую составлять „картотеку” пользователей, куда будет заносится их фамилия, имя и дата рождения.

В начале программы объявите переменные для хранения фамилии, имени и даты рождения, например, так: `char firstname[32], lastname[32], birthdate[32]`. Не забудьте подключить библиотеку стандартного ввода/вывода `stdio.h`. Еще нам понадобится переменная-символ, в которой будет храниться введенная команда (`char command`) и указатель на файл-базу данных (`FILE* file`).

Далее, нам понадобятся две функции: распечатывающая весь список (`void printcard()`) и добавляющая пользователя (`void newuser`).

Функция, добавляющая пользователя, должна запросить ввести его фамилию, имя и дату рождения, а затем оператором `fprintf` записать эти данные в файл-базу данных.

Функция, печатающая „картотеку”, должна для начала „перемотать” файл-базу данных на начало (функция `rewind(file)`), затем считать из него первую запись (функция `fscanf`) и в цикле печатать/считывать информацию, вплоть до достижения конца файла. Здесь удобнее использовать цикл `do ... while`:

```
do printf("%s\t%s\t%s\n", firstname, lastname, birthdate);
while(fscanf(file,"%s %s %s\n", firstname, lastname, birthdate)!=EOF);
```

Этот цикл сначала выполняет свое тело, а потом проверяет условие. Нам необходимо считать данные вплоть до конца файла, для этого в операторе проверки условия выполнения цикла `while` мы считываем следующую запись. Если предыдущая запись — последняя, функция `fscanf` возвращает признак конца файла `EOF`, тогда цикл прекращается.

Теперь дошла очередь и до основной программы. Во-первых, откроем файл базы данных (назовите его как угодно, в данном случае откроется файл `card`). Так как при первом запуске файла может не существовать, необходимо, чтобы в этом случае он создался. Если же запуск не первый, файл должен открыться для дописывания данных в его конец. Для этого используем функцию `open` со вторым аргументом `"a+"` — открыть для дописи и чтения: `file=fopen("card "a+");`.

Теперь создадим цикл считывания символа и проверки, является ли введенный символ командой. Этот цикл подолжается до тех пор, пока не будет введено q. Для такого бесконечного цикла удобно использовать скобки `do ... while`.

Проверку введенной команды проще осуществлять при помощи функции `switch`, за которой следует последовательность вариантов `case ... : ... ; break`. Если аргумент функции совпадает с аргументом какого-либо `case`, выполняются все действия после этого `case` вплоть до оператора `break`, выходящего из оператора проверки.

Итак, последний участок функции `main` можно оформить так:

```
do{
    printf("Введите команду\n(n - новый пользователь, p - отобразить картотеку, q - выход):\
    command = getchar();
    switch(command){
        case 'n': newuser(); break;
        case 'p': printcard(); break;
        case 'q': fclose(file); exit(0);
        }
    }
while(command != 'n' || command != 'e' || command != 'q')
```

Выполните задание (22):

Составьте блок-схему данной программы, наберите ее текст, откомпилируйте и запустите. Создайте несколько записей, проверьте реакцию программы на ввод посторонних символов, введите команду „p” и посмотрите, как выглядят все записи.

Выйдите из программы, посмотрите содержимое файла, в котором хранилась ваша база данных.

11. Составление алгоритмов управления станками с ЧПУ.

11. 1. Введение

Современный технологический процесс не обходится без цифровой электроники. Электроникой „напичканы” многие приборы, начиная от наручных часов и заканчивая космическими аппаратами. На производстве тоже внедряется электроника: это и станки с ЧПУ, и роботы, и даже целые роботизированные предприятия, управляемые единой АСУ.

Оперирование такими приборами требует знания элементарной цифровой логики и основ алгоритмизации и программирования.

11. 1.1.Логические элементы.

Все цифрове схемы основаны на трех „китах” — элементарных логических элементах: это элементы **„И”, „ИЛИ”, „НЕ”.** Каждый элемент характеризуется **таблицей истинности** — соответствия выходного сигнала разным значениям входного. Названия элементов вполне логически соответствуют производимым ими операциям:

- Элемент „И” осуществляет логическое **умножение** (**дизъюнкция**). Обозначается значком $\wedge$. Он имеет как минимум два входа и всегда один выход. На выходе будет сигнал логической единицы (*true* или *истина*) лишь в случае, когда **на все** входы поданы сигналы логической единицы. В противном случае на выходе будет сигнал логического нуля (*false* или *ложь*).
  В математике этому элементу соответствует оператор пересечения ($\cap$). В языке C логическое „И” обозначается соответствующим значком (`&`) для **побитового** „И” (когда сравниваются все биты двух чисел) и двойным значком (`&&`) для логического „И”.
  Результат логической операции „И” будет верным лишь в случае верности всех ее аргументов. Установить ее связь с умножением легко, взглянув на таблицу истинности.
- Элемент „ИЛИ” осуществляет логическое **сложение** (**конъюнкция**). Обозначается значком $\vee$. Тоже имеет не менее двух входов и всего один выход. На выходе будет сигнал логического нуля лишь в случае, когда на **все входы** подан сигнал логического нуля. Иначе на выходе сигнал логической единицы.
  Таблица истинности элемента „ИЛИ” не совсем оправдывает операцию сложения ($1 + 1 = 10$, а в таблице истинности $1 \vee 1 = 1$), поэтому существует еще видоизмененный элемент — „Исключающее ИЛИ” ($\oplus$), выход которого отражает младший бит суммы входных сигналов. Для него $1 \oplus 1 = 0$, остальные строки таблицы истинности — как у элемента „ИЛИ”.
  В математике элементу „ИЛИ” соответствует оператор сложения ($\cup$), а в языке C — побитовое „ИЛИ” (`|`) и логическое „ИЛИ” (`||`).
- Элемент „НЕ” осуществляет логическую операцию **инверсии**. Обозначается горизонтальной чертой над символом ($\overline{A}$). Этот элемент всегда имеет всего один вход и один выход. Сигнал на выходе противоположен сигналу на входе.
  Строгого математического эквивалента этому элементу нет. Обычно ему приписывают операцию $\neq$, а в геометрии — геометрического дополнения. На языке C логическая операция „НЕ” обозначается восклицательным знаком (`!`), а побитовая — тильдой (`~`).

Таблица истинности для элементов „И”, „ИЛИ” и „НЕ”

A	B	$A \wedge B$	$A \vee B$	$\overline{A}$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

На основе элементарных логических элементов создаются и более сложные, необходимые для полноценного осуществления математических операций и хранения данных.

Триггер предназначен для хранения одного бита информации. Его выходной сигнал зависит от подачи логической 1 на один из двух входов („Установить в 1” и „ Установить в 0”).

Регистр обеспечивает хранение N бит информации, которые можно считать одновременно (если регистр состоит из N параллельно подключенных триггеров) или поочередно (если он состоит из N последовательно подключенных триггеров). Во втором случае регистр называется **сдвиговым**, т.к. он обеспечивает сдвиг хранящейся информации на один бит вправо (деление на 2) или влево (умножение на 2).

Счетчик обычно имеет один логический вход (куда подаются импульсы, которые он должен считать), один вход сброса (для обнуления счетчика) и несколько логических выходов.

Счетчик содержит триггеры и регистры, считая количество входных импульсов и выдавая на выходе сумму в двоичной системе.

Современные микросхемы могут содержать миллионы и даже миллиарды элементарных логических узлов, связанных в сложные компоненты. Нет нужды знать устройство микросхемы, чтобы работать с ней, но необходимо знать, как оперировать с прибором, содержащим такие микросхемы.

11. 1.2.Датчики и микроконтроллеры в АСУ.

Программируемый микроконтроллер (ПМК) представляет собой специализированную управляющую микро ЭВМ, работающую в режиме реального времени по фиксированной рабочей программе, размещенной в ОЗУ ПМК. ПМК может работать как в паре с ПК, так и отдельно от него.

Обычно ПМК содержит **аналогово-цифровые преобразователи** (АЦП) — для кодирования измеряемых аналоговых сигналов в привычную для ЭВМ форму; **ЦАП** — для обратного преобразования цифровых сигналов в аналоговые; **коммутаторы каналов** — для последовательного опроса цифровых каналов; **ОЗУ**, в котором хранятся исполняемая программа и ее данные; **ПЗУ** — содержацая операционную систему.

11. 2. Программирование ЧПУ.

Для программирования работы станка с ЧПУ необходимо разработать карту технологического процесса, а затем составить алгоритм работы устройства.

Ввиду отсутствия собственно станков, мы будем только составлять алгоритмы технологического процесса.

Выполните задание (23):

Составьте алгоритм работы роботизированного сверлильного станка для сверления десяти металлических заготовок размером 100х200 мм. В них необходимо сделать 5 отверстий диаметром 10 мм.: одно в центре и четыре по углам на расстоянии 20 мм от краев заготовок. Алгоритм оформите в виде блок-схемы.

► Перед выполнением сверления следует проверить соответствие размеров заготовок требуемым размерам. Допуск размеров ± 0.1 мм. Для подсчета готовых изделий следует ввести счетчик, увеличивающийся на 1 каждый раз, как только окончен процесс изготовления очередной детали.

Выполните задание (24):

Теперь представьте, что вы оперируете АСУ конвейера, производящего гайки М3. На входе вы имеете металлический лист толщиной 2.5 мм и размерами 3х3 м. Составьте технологическую карту производства гаек (методом штамповки) из такого листа. После штамповки, и сверления следует отшлифовать гайки, затем — просверлить в них отверстие под резьбу (вспомните, какой диаметр сверла необходимо взять для этого), нарезать резьбу и оцинковать.

Вначале составьте функциональную блок-схему, отмечая каждый техпроцесс функциональным модулем. Затем составьте блок-схемы каждого модуля.

Не забывайте, что автомат, после окончания штампованных заготовок, должен дать команду прессовочному станку произвести следующую их партию.

12. Работа в компьютерных сетях.

12. 1. Работа с протоколом http.

Для просмотра гипертекстовых документов в Linux существует множество браузеров, как консольных, так и графических. Они работают обычно не только с протоколом http, но и с другими протоколами, позволяя, например, скачивать файлы по протоколу ftp, заходить на защищенные сайты по протоколу https, читать новости и т.п.

В интернете существует множество разнообразных ресурсов, содержащих огромное количество документов. Кроме того, интернет предлагает средства текстового общения (форумы, блоги, ICQ, IRC и многое другое), рассылки писем, новостей, телефонии, видеофонии.

12. 1.1. Консольные браузеры: `lynx` и `links`.

Откройте окно виртуальной консоли (например, `konsole` или `xterm`). Запустите в нем сначала браузер Lynx командой `lynx www.sgpk.ru`. Полистайте страницы. Обратите внимание, как браузер реагирует на нажатие тех или иных клавиш. Активация гиперссылки происходит при нажатии клавиши Enter, перемещение между ссылками — клавишами „вверх” и „вниз”.

Выполните задание (25):

Откройте страницу с фреймами (например, ссылку „разные картинки”). Обратите внимание, что фреймы можно просматривать только по отдельности.

Ответьте на вопрос(ы):

Что такое „фрейм” и для чего он нужен?

Выполните задание (26):

Теперь откройте тот же сайт при помощи браузера Links (`links www.sgpk.ru`). Посмотрите, как теперь выглядят страницы с фреймами. Скажите, в чем недостаток текстовых браузеров, а в чем их преимущества перед графическими.

12. 1.2. Графические браузеры.

Откройте любой графический браузер (Mozilla, Konqueror, Firefox) на сайте www.sgpk.ru. Пройдитесь по ссылкам. Обратите внимание на самое главное отличие графических браузеров от консольных.

Выполните задание (27):

Зайдите на форум и оставьте в нем какую-нибудь запись. Сможете ли вы удалить запись после того, как ее добавили?

Выполните задание (28):

Зайдите на хранилище документации. Как сохранить документ? Как открыть его или найти после сохранения?

Выполните задание (29):

Зайдите на хранилище изображений. Как сделать изображение заставкой на рабочем столе?

12. 2. Протокол ftp.

FTP служит для обмена файлами. Для просмотра ресурсов ftp-серверов служат браузеры, а также консольная программа `ftp`.

Выполните задание (30):

Зайдите на ресурс `ftp.sgpk.ru` при помощи `mc`. Для этого, нажав клавишу <F9>, выберите в меню соответствующей панели пункт „FTP-соединение". Активируйте его и введите адрес ресурса. После соединения вы можете просматривать его как локальную папку. Скачайте какой-нибудь файл к себе в домашнюю папку.

Выполните задание (31):

Зайдите на тот же ресурс при помощи графического браузера. Для этого в строке адреса перед адресом ресурса следует указать протокол: `ftp://ftp.sgpk.ru`. Браузер будет выступать в роли графического файлового менеджера.

12. 3. Создание домашней страницы.

Ваши домашние папки хранятся на сервере. Он настроен так, что, если вы создадите в папке `~/www` файл `index.html` с соответствующим содержанием, у вас появится домашняя страничка, выйти на которую можно по адресу `www.sgpk.ru/~`<login>, где <login> — ваше регистрационное имя в системе.

► Краткое обозначение домашней папки — `~/`, поэтому вы можете быстро войти в какой-нибудь свой каталог, набрав `cd ~/каталог`. Для того, чтобы просто войти в домашнюю папку, достаточно набрать команду `cd` без параметров.

Создать домашнюю страничку можно в любом текстовом редакторе, но для этого вы должны знать язык гипертекстовой разметки html. Проще создать этот документ при помощи специальной программы — компоновщика. Например, при помощи компоновщика браузера Mozilla (Mozilla composer).

Выполните задание (32):

Создайте в Mozilla новую страницу компоновщика. Напишите в ней какой-нибудь текст о себе, добавьте изображений. Для более красивого размещения текста и картинок используйте таблицы. Раскрасьте элементы в разные цвета. Название страницы выделите крупным шрифтом.

Сохраните страницу под именем `index.html` в папку `~/www/`. Теперь, если вы все сделали правильно, зайдите на свою домашнюю страницу при помощи любого браузера. Посмотрите ее в нескольких разных браузерах. Объясните причины различия представления одной и той же информации в разных браузерах.