***1)Понятие о программном обеспечении ЭВМ.Состав,Структура,многоуровневый иерархический характер.***

**Понятие программного обеспечения.   
Классификация программного обеспечения.**

Программное обеспечение-это совокупность программ, выполненных вычислительной системой.

 К программному обеспечению (ПО) относится также вся область деятельности по проектированию и разработке (ПО):

* технология проектирования программ (нисходящее проектирование, структурное программирование и др.)
* методы тестирования программ.
* методы доказательства правильности программ.
* анализ качества работы программ и др.

Программное обеспечение - неотъемлемая часть ЭВМ. Оно является логическим продолжением технических средств ЭВМ, расширяющие их возможности и сферу использования.

**Классификация программного обеспечения.**

|  |  |
| --- | --- |
|  |  |
|  | http://www.infosgs.narod.ru/23.files/image001.gif |

Существует три категории:

**1) Прикладные программы**, непосредственно обеспечивающие выполнение необходимых пользователям работ.

**2) Системные программы**:

* управление ресурсами ЭВМ.
* создание копий используемой информации.
* проверку работоспособности устройств компьютера.
* выдачу справочной информации о компьютере и др..

**3)** **Инструментальные программные системы**, облегчающие процесс создания новых программ для компьютера.

 Более или менее определенно сложились следующие группы программного обеспечения:

* операционные системы.
* системы программирования.
* инструментальные системы.
* интегрированные пакеты.
* динамические электронные таблицы.
* системы машинной графики.
* системы управления базами данных (СУБД).
* прикладное программное обеспечение.

***2)Понятие об ОС.***

Операционная система (ОС) - комплекс системных и управляющих программ, предназначенных для наиболее эффективного использования всех ресурсов вычислительной системы (ВС) (Вычислительная система - взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации) и удобства работы с ней.

Назначение ОС - организация вычислительного процесса в вычислительной системе, рациональное распределение вычислительных ресурсов между отдельными решаемыми задачами; предоставление пользователям многочисленных сервисных средств, облегчающих процесс программирования и отладки задач. ОС в значительной степени формирует у пользователя представление о возможностях ВС, удобстве работы с ней, ее пропускной способности. Различные ОС на одних и тех же технических средствах могут предоставить пользователю различные возможности для организации вычислительного процесса или автоматизированной обработки данных.

В соответствии с условиями применения различают три режима ОС: пакетной обработки, разделения времени и реального времени. В режиме пакетной обработки ОС последовательно выполняет собранные в пакет задания. В этом режиме пользователь не имеет контакта с ЭВМ, получая лишь результаты вычислений. В режиме разделения времени ОС одновременно выполняет несколько задач, допуская обращение каждого пользователя к ЭВМ. В режиме реального времени ОС обеспечивает управление объектами в соответствии с принимаемыми входными сигналами. Время отклика ЭВМ с ОС реального времени на возмущающее воздействие должно быть минимальным.

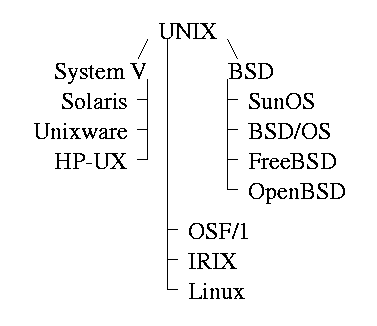
***3)Краткая историческая справка об ОС UNIX.***

ОС Unix была создана в 1969г. несколькими программистами Bell Labs фирмы AT&T "для себя", и работала на компьютере PDP-7. Через некоторое время большая ее часть была переписана с языка ассемблера на язык C, что дало возможность перенести ее на большое количество разных платформ. В настоящее время Unix работает на большинстве существующих архитектур, и для многих из них является основной ОС.

Дальнейшее развитие Unix, разработанного в AT&T и его производные называться System V (пятая версия), сокращенно SysV, иногда используется название "AT&T-версия Unix".

В середине семидесятых годов исходный код Unix попал в университет Беркли, где была создана своя версия Unix, получившая название "BSD Unix" (Berkeley Software Distribution).

В настоящее время большинство вариатнов Unix основаны или на SystemV, или на BSD.



Однако в последнее время предпринимаются усилия по выработке единого стандарта, и наиболее современные варианты Unix нельзя четко отнести ни к той, ни к другой ветви. В их число входят IRIX (компьютеры Silicon Graphics, пример в ИЯФ -- sky), Digital OSF/1 (компьютеры Alpha, пример в ИЯФ -- AxInp), и Linux.

В настоящее время UNIX используются в основном на [серверах](http://ru.wikipedia.org/wiki/Сервер_(аппаратное_обеспечение)), а также как встроенные системы для различного оборудования. На рынке ОС для [рабочих станций](http://ru.wikipedia.org/wiki/Рабочая_станция) и домашнего применения лидером является [Microsoft Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows), UNIX занимает только второе ([Mac OS X](http://ru.wikipedia.org/wiki/Mac_OS_X)), третье ([GNU/Linux](http://ru.wikipedia.org/wiki/GNU/Linux)) и многие последующие места.

UNIX-системы имеют большую историческую важность, поскольку благодаря им распространились некоторые популярные сегодня концепции и подходы в области ОС и [программного обеспечения](http://ru.wikipedia.org/wiki/Программное_обеспечение). Также, в ходе разработки Unix-систем был создан язык [Си](http://ru.wikipedia.org/wiki/Си_(язык_программирования)).

***4)Особенности ОС UNIX(режим работы, инструментальные и сервисные средства, язык С, TCP/P, Internet и пр.)***

* использование простых текстовых файлов для настройки и управления системой;
* широкое применение утилит, запускаемых в [командной строке](http://ru.wikipedia.org/wiki/Интерфейс_командной_строки);
* взаимодействие с пользователем посредством виртуального устройства — терминала;
* представление физических и виртуальных устройств и некоторых средств межпроцессового взаимодействия как [файлов](http://ru.wikipedia.org/wiki/Файл);
* использование [конвейеров]((UNIX)) из нескольких программ, каждая из которых выполняет одну задачу.
* Файловая система древовидная, чувствительная к регистру символов в именах, очень слабые ограничения на длину имён и пути.
* Нет поддержки структурированных файлов ядром ОС, на уровне системных вызовов файл есть поток байтов.
* Широкое использование текстовых файлов для хранения настроек, в отличие от двоичной базы данных настроек, как, например, в Windows.
* Широкое использование утилит обработки текста для выполнения повседневных задач под управлением скриптов.

Сетевые протоколы TCP/P

* Архитектура
* Протокольные модули
* Интерфейсы
* Адресация, DNS
* Маршрутизация,шлюзы

Основные службы Internet

* Почта(SMTP, POP, IMAP, UUCP)
* Передача файлов(FTP)
* Файловые серверы(NFS, SMB)
* Новостные группы(NNTP)
* Удаленные терминал и выполнение(telnet,rsh,ssh)

Сетевая безопасность

* Использование Firewall
* Защита inetd служб, tcpwrapper(tcpd)
* Proxy – серверы

Особая роль языка программирования Си состоит в том, что он, с одной стороны, позволяет писать для UNIX-систем практически столь же эффективный код, что и языки ассемблера, а с другой, является основным средством переноса программ между UNIX-системами. Можно сказать, что Си является машинно-независимым языком ассемблера для UNIX-систем. Это делает его основным средством написания эффективных и переносимых программ для этого класса вычислительных систем.

***5)Основы интерактивной работы в ОС UNIX.Начало и окончание сеанса.Доступ к оперативной документации.***

В операционной системе UNIX при наборе команд, имен файлов и каталогов, используется латинский алфавит, причем различаются заглавные и прописные буквы, т.е. команды и имена файлов должны вводится в том виде, в котором они приводятся в руководствах по UNIX [[5]](http://math.tsu.ru/cluster/other/cluster/literature.htm). Процесс общения с ОС UNIX осуществляется, в основном, при помощи клавиатуры и монитора. После входа в систему на экране можно увидеть одно из двух элементов интерфейса UNIX: командную строку или графическую оболочку.

Командная строка – традиционный способ общения в UNIX. Когда UNIX ожидает команду, на экране высвечивается приглашение – «$». Чтобы дать команду системе, нужно в командной строке набрать имя команды, а затем ее опции и аргументы (могут отсутствовать), разделенные пробелами. После набора ввод команды осуществляется нажатием клавиши ENTER. Команды UNIX имеют структуру

command option(s) argument(s)

Опции обычно сопровождаются знаком «минус» и модифицируют поведение команды. Аргументы – это обычно имена каталогов или файлов.

Заметим, что каталог пользователя находится в папке /home/user\_name/ корневого каталога системы. Названия файлов в ОС UNIX имеют структуру filename.ext, например, Euler.f; myCprog.c; a1.o; document.txt.

**НАЧАЛО СЕАНСА В СИСТЕМЕ UNIX**

**Пароль**

Когда вы впервые входили в систему, вам, наверное, пришлось ввести пароль, после того как вы вели свое входное имя. Однако системный администратор может и не установить пароль для вас, в этом случае приглашение password: не появится на экране.

Ваш пароль - средство безопасности; он защищает ваши файлы от доступа других пользователей. Если ваш системный администратор не установил пароль для вас, вы должны сделать это сами. Хорошо менять свой пароль примерно раз в месяц, даже если ваша система и не требует этого от вас.

Ниже приведен список правил, которым вы должны следовать, выбирая пароль. Пароль должен:

**-** иметь по меньшей мере шесть символов, но значение имеют только первые восемь.

**-** содержать по меньшей мере две буквы

**-** содержать по меньшей мере одну цифру или специальный знак (не используйте символы @ и #; они относятся к уничтожающим символам).

**-** отличаться от входных имен пользователей, отличаться от сдвинутых входных имен и не быть входным именем набранным в обратном порядке.

**-** новый пароль должен отличаться от старого по меньшей мере тремя символами.

Помните, что каждый может знать ваше входное имя, но только вы должны знать свой пароль. Не выбирайте пароль, о котором можно легко догадаться. Следует избегать использования в качестве пароля имен ваших супругов и детей. Также лучше иметь цифру где-нибудь в середине пароля, а не с краю.

Введите команду **passwd** для того чтобы изменить свой пароль или набрать новый.

**Конец сеанса**

Чтобы закончить сеанс, вы должны выйти из системы. Выйти можно двумя способами. Первый способ: нажать одновременно клавишу Control и букву **d**. Это называется **control d** и обозначается в справочнике **^d** или **<CTRL/d>**. После нажатия **^d** на терминале снова появится приглашение **login**:.

Альтернативный метод завершения сеанса - набрать слово **exit** и нажать <RET> после интерактивного символа $.

Если ваш терминал соединен посредством модема, вам может понадобится отключить телефон. Замечание: простое выключение терминала не заканчивает ваш сеанс в системе!

….

….

….

…..

***6)Команды ОС UNIX. Формат команд. Интерпретатор команд.***

**1.** **Простейшие действия ls** - выдать список файлов в текущем каталоге.

**cd [каталог]** -сменить текущий каталог. Если имя каталога не указывается, то текущим становится домашний каталог пользователя.

**cp <что\_копировать> <куда\_копировать>** -копировать файлы.

**mv <что\_перемещать> <куда\_перемещать>** -переместить или переименовать файл.

**ln -s <на\_что\_сделать\_ссылку> <имя\_ссылки>** -создать символическую ссылку.

**rm <файлы>** -удалить файлы.

**mkdir <каталог>** -создать новый каталог.

**rmdir <каталог>** -удалить пустой каталог.

**rm -r <файлы и/или каталоги> (рекурсивное удаление)** -удалить файлы или каталоги и их подкаталоги. ОСТОРОЖНЕЙ с этой командой, поскольку пока у Linux нет системы полного восстановления удаленных файлов (если вы не пользуетесь специальными программами для помещения удаленных файлов в специальный каталог, что-то вроде "корзинки" в ОС Windows).

**cat <имя\_файла>** -вывод содержимого файла на стандартный вывод (по умолчанию - на экран).

Можно записать вводимый на экран текст с помощью следующей последовательности действий:   
**cat > <имя\_файла>  
.   
.   
CTRL/d**

**more <имя\_файла>** -просмотр содержимого длинного текстового файла по страницам.

**less <имя\_файла>** -просмотр содержимого текстового файла с возможностью вернуться к предыдущим страницам. Нажмите q, когда захотите выйти из программы. "less" - аналог команды DOS "more", хотя очень часто "less" бывает более удобной чем "more".

**pico <имя\_файла>** -редактировать текстовый файл с помощью текстового редактора pico.

**lynx** -просмотр файла html или WWW ссылки с помощью текстового браузера Lynx.

**tar -zxvf <файл>** -распаковать архив tgz или tar.gz

**find <каталог>** -name имя\_файла -найти файл с именем "имя файла" и отобразить результат поиска на экране. Поиск начинается с каталога <каталог>; "имя\_файла" может содержать маску для поиска.

**pine** -хорошая текстово - ориентированная программа для чтения электронной почты.

**mc** -запустить программу управления файлами "Midnight Commander" (Выглядит как "Norton Commander", но по своим возможностям ближе к far).

**./Имя\_Программы** -запустить на исполнение исполняемый файл в текущем каталоге, если текущего каталога нет в списке каталогов, указанных в переменной окружения PATH.

**xterm (в X терминале)** -запустить простой терминал в графической оболочке X-windows. Для того чтобы выйти из него, наберите "exit".

**Стандартные команды и команды, дающие информацию по системе команд (всегда набираются в одной строке). Описание:**

**pwd** -вывести имя текущего каталога.

**whoami** -вывести имя под которым Вы зарегистрированы.

**date** -вывести дату и время.

**time <имя программы>** -выполнить программу и получить информацию о времени, нужном для ее выполнения. Не путайте эту команду с date. Например: Я могу определить выполнить команду ls и узнать, как много времени требуется для вывода списка файлов в каталоге, набрав последовательность: time ls

**who** -определить кто из пользователей работает на машине.

**rwho -a** -определение всех пользователей, подключившихся к вашей сети. Для выполнения этой команды требуется, чтобы был запущен процесс rwho.

**ruptime** -какие машины работают в сети, и какие остановлены.

**finger <имя\_пользователя>** -системная информация о зарегистрированном пользователе. Попробуйте: finger <ваш login-name>

**uptime** -количество времени, прошедшего с последней перезагрузки операционной системы.

**ps a** -вывести список текущих процессов в Вашем сеансе работы.

**top** -интерактивный список текущих процессов, отсортированных по использованию центрального процессора.

**uname -a** -вывести информацию о версии операционной системы.

**free** -вывести информацию по использованию памяти.

**df -h** -вывести информацию о свободном и используемом месте на дисках.

**du . -bh | more** -вывод на экран информации о размере файлов и каталогов, начиная с текущего каталога.

**set|more** -вывести текущие значения переменных окружения. (Не для всех shell. Для csh/tcsh - printenv | more, хотя set тоже покажет полезную информацию.)

**echo $PATH** -вывести значение переменной окружения "PATH" Команда echo может использоваться для вывода значений любых переменных окружения. Воспользуйтесь командами set или printenv для получения полного списка.

**Работа с сетью**

**ssh** -обеспечивает безопасное вхождение в удаленный сеанс работы с другой машиной, а также позволяет выполнить заданную команду на удаленной машине без вхождения в сеанс работы:   
**ssh [-l Ваше\_имя\_пользователя\_на\_удаленной\_машине] <имя\_удаленной\_машины>** -вхождение в сеанс на удаленной машине. Используйте имя машины или ее IP адрес. (Вы должны быть зарегистрированы на этой удаленной машине).   
(если Ваше\_имя\_пользователя одинаково на локальной и удаленной машинах, то его можно не набирать, т.е.:   
**ssh <имя\_удаленной\_машины>** -позволит Вам войти в сеанс на удаленной машине)   
**ssh <Ваше\_имя\_пользователя\_на удаленной машине@><имя\_удаленной\_машины> <команда>** -выполнит на удаленном компьютере заданную команду и передаст Вам на экран результат ее выполнения (При установлении соединения по ssh Вы должны будете ввести пароль, который Вы имеете на удаленном компьютере; при этом пароль будет передан по сети в зашифрованном виде, т.е. безопасным образом.)

**scp** -обеспечивает безопасное копирование файлов в сети:   
**scp <имя\_файла\_на\_локальном\_компьютере> <Ваше\_имя\_пользователя\_на удаленной машине>@<имя\_удаленной\_машины>:** -cкопирует файл с локального компьютера в Вашу корневую директорию на удаленном компьютере (наличие ":" в конце команды обязательно).

**telnet <имя\_удаленной\_машины>** -связаться по telnet с другой машиной. Войдите в сеанс работы после установления связи с помощью вашего пароля.

**ftp <имя\_удаленной\_машины>** -связаться по ftp с удаленным компьютером. Этот тип связи хорош для копирования файлов с/на удаленную машину.

Предпочтительно не пользоваться командами **telnet** и **ftp**, а использовать только ssh и scp, поскольку они обеспечивают безопасность сетевого соединения!!!

**hostname -i** -показывает IP адрес компьютера, на котором Вы работаете.

**Некоторые команды администрирования**

**alias ls="ls -Fskb --color"** -создать alias - псевдоним для того чтобы одной командой можно было запустить более сложную комбинацию команд. Поместите создание alias в файл /etc/bashrc если вы хотите, чтобы эти псевдонимы были доступны всем пользователям вашей системы.

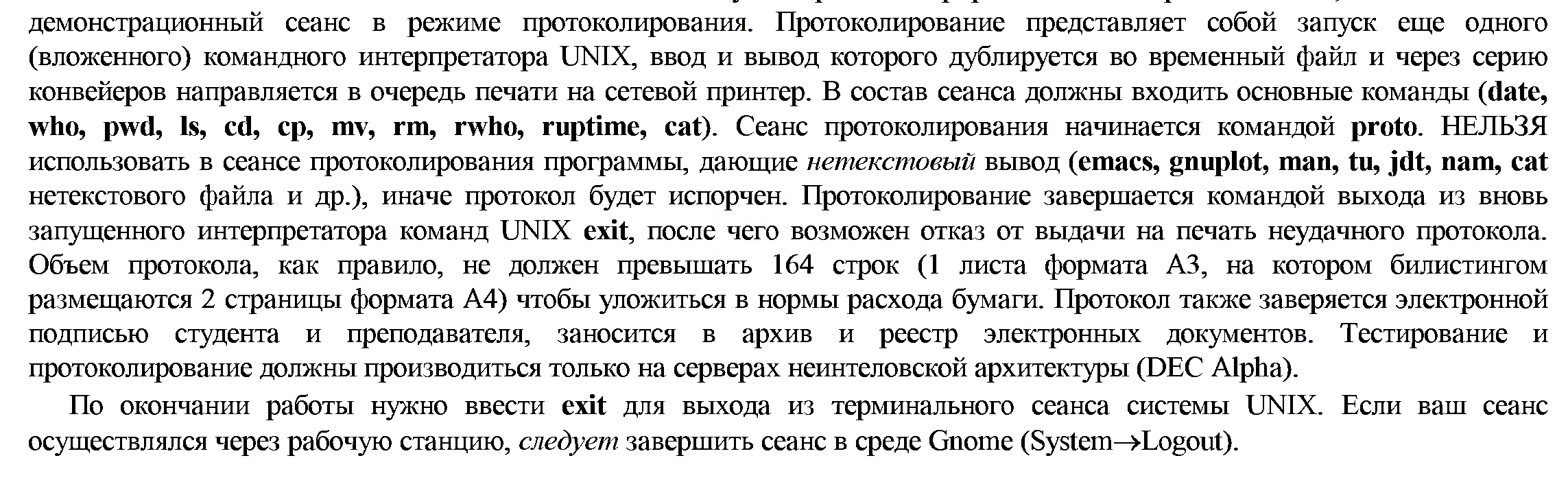
**Для tcsh** -формат определения alias другой:

**alias la 'ls -AF --color=none'**

**kapasswd** -команда для изменения пароля для доступа к файловой системе AFS. При работе на базовом Linux-кластере ЛИТ следует пользоваться только этой командой (а не командой passwd!) для изменения пароля для вхождения в кластер.

**passwd** -изменить свой пароль на каком-либо локальном компьютере.

***7)Протоколирование команд.***

 ***8)Файловая система ОС UNIX (структура, наименование, навигация)***

Файловая система в Unix организована в виде единого дерева директорий. Никаких букв дисков (a:, c:) здесь нет, а если используется несколько физических дисков (например, первый жесткий диск, второй, CD-ROM и т.д.) то они представляются отдельными ветвями на общем дереве.

Директории разделяются косой чертой "/" (а не обратной, как в Dos), например, "/home/inpunix/ivanov/" -- поддиректория "ivanov" в директории "inpunix" директории "home". Самая верхняя директория ("корневая") называется "/".

Какие разрешаются имена:

* Имя файла (и директории) может содержать практически любые символы, включая пробелы (и даже управляющие символы, типа перевода строки, если удастся их ввести). Примеры:

list.txt  
A file with spaces.txt  
Сводный отчет по трудозатратам за 1998й год.tex  
Manual [part 1 of 3].txt

* Символ "." не является особенным, и в имени может быть любое их количество, а может и не быть вовсе. Так, к примеру,

file.with.many.dots  
short  
file\_without\_extension  
.......

являются вполне нормальными именами.

* У программ не требуется какое-либо специальное расширение, типа .exe -- достаточно установить атрибут "x" (см. ниже).
* Заглавные и маленькие буквы различаются; так,

report.txt  
REPORT.TXT  
Report.txt  
RePorT.tXt

это четыре разных имени.

* В отличие от Dos/Windows, нет зарезервированных имен, таких, как con, com1 и т.д. Вместо них есть сответствующие файлы устройств в директории /dev:

|  |  |  |
| --- | --- | --- |
| **Файл в Dos** | **Устройство в Unix** | **Назначение** |
| con | /dev/tty | Консоль |
| nul | /dev/null | Пустота |
| com1 | /dev/ttyS0 | Первый последовательный порт |
| lpt1 | /dev/lp0 | Первый параллельный порт |

* У программ не требуется какое-либо специальное расширение, типа .exe -- достаточно установить атрибут "x" (см. ниже).
* Длина имени может быть до 255 символов; полный путь (включая директории) -- 1024 символа.

Хотя единственный символ, которого не может быть в имени, это "/" (поскольку это разделитель директорий), лучше не слишком увлекаться, и стараться не делать имен с пробелами, и не злоупотреблять различием в регистре букв -- это только внесет лишнюю путаницу. Также лучше не называть файлы по-русски.

В каждой директории есть поддиректория ".", являющаяся ссылкой на саму директорию и "..", являющаяся ссылками директорию уровнем выше.

Файлы, имена которых начинаются c точки (включая . и ..), считаются "неинтересными", и в листинге директории обычно не показываются.

***9)Метасимволы для генерации имен файлов.***

Метасимволы - символы, имеющие специальное значение для интерпретатора :

? \* ; & ( ) | ^ < > <пробел> <табуляция> <возврат\_каретки>

Однако каждый из этих символов может представлять самого себя, если перед ним стоит \. Все символы, заключенные между кавычками ' и ', представляют самих себя. Между двойными кавычками (") выполняются подстановки команд (см п. 2.2) и параметров (см. п. 2.3), а символы \, `," и $ могут экранироваться предшествующим символом \.

После всех подстановок в каждом слове команды ищутся символы \*,?, и [. Если находится хотя бы один из них, то это слово рассматривается как шаблон имен файлов и заменяется именами файлов, удовлетворяющих данному шаблону (в алфавитном порядке). Если ни одно имя файла не удовлетворяет шаблону, то он остается неизменным. Значения указанных символов:

\* - произвольная (возможно пустая) последовательность

символов;

? - один произвольный символ;

[...] - любой из символов, указанных в скобках перечислением

и/или с указанием диапазона;

cat f\* - выдаст все файлы каталога, начинающиеся с "f";

cat \*f\* - выдаст все файлы, содержащие "f";

cat program.? - выдаст файлы данного каталога с

однобуквенными расширениями, скажем "program.c" и "program.o", но

не выдаст "program.com";

cat [a-d]\* - выдаст файлы, которые начинаются с "a", "b",

"c", "d". Аналогичный эффект дадут и команды "cat [abcd]\*" и

"cat [bdac]\*".

***10)Управление файлами(создание, копирование, переименование, сцепление, удаление)***

|  |  |
| --- | --- |
| urbbul1d | file filename(s) - отображает тип файла (например ASCII текст, документ Word и др.) |
| urbbul1d | cat filename(s) - показывает содержание файлов (используется только для текстовых файлов!); |
| urbbul1d | more filename(s) - действует также, как и cat, но позволяет листовать страницы; |
| urbbul1d | head filename - показывает первые десять строк файла filename; |
| urbbul1d | tail filename - показывает последние десять строк файла filename; |
| urbbul1d | wc filename(s) - для каждого текстового файла показывает число байт, букв и линий; |
| urbbul1d | rm filename(s) - уничтожает файлы; |
| urbbul1d | cp filename newname - создает копии файлов с новыми именами; |
| urbbul1d | cp filename(s) dir копирует один или более файлов в другой каталог; |
| urbbul1d | mv oldname newname - изменяет имя файла или каталога; |
| urbbul1d | mv filename(s) dir - перемещает один или более файлов в другой каталог; |
| urbbul1d | find dir –name filename - пытается локализовать файл filename рекурсивно в подкаталоге dir; |

***11)Управление каталогами.***

|  |  |
| --- | --- |
| urbbul1d | pwd – показывает текущий рабочий каталог; |
| urbbul1d | cd dir – установит текущим рабочим каталогом катало с именем dir; |
| urbbul1d | cd – вернуться в каталог /home/user\_name/; |
| urbbul1d | mkdir subdir – создать новый подкаталог с именем subdir; |
| urbbul1d | rmdir subdir – переместить подкаталог с именем subdir, который должен быть пустым; |
| urbbul1d | ls – показать содержание текущего каталога; |
| urbbul1d | ls dir – показать содержание каталога dir; |
| urbbul1d | ls –A - показать все файлы, в том числе и скрытые; |
| urbbul1d | ls –l - показать аттрибуты (владельца, разрешение на доступ, размер файла и время последней модификации); |
| urbbul1d | mv oldname newname - изменить имя подкаталога или переместить его; |
| urbbul1d | cp dirname destination - копировать весь подкаталог dirname в другое место destination |

***12)Переадресация ввода и вывода в ОС UNIX.***

### Перенаправление ввода и вывода

В системе UNIX некоторые команды ожидают ввод только с клавиатуры (стандартный ввод) и большинство команд отображают свой вывод на экране терминала (стандартный вывод). Однако система UNIX позволяет вам перенаправлять ввод и вывод в файлы и программы, т.е. вы можете сказать shell:

* взять ввод из файла, а не с клавиатуры;
* послать вывод в файл, а не на терминал;
* использовать программу как исходные данные для другой программы.

#### 8.3.1. Перенаправить ввод: знак <

Чтобы перенаправить ввод, укажите в командной строке после знака "меньше чем" (<) имя файла:

command < имя\_файла<CR>

#### 8.3.2. Перенаправить вывод: знак >

Чтобы перенаправить вывод, укажите в командной строке после знака "больше чем" (>) имя файла:

command > имя\_файла<CR>

Примечание. Если вы перенаправите вывод в уже существующий файл, то вывод вашей команды заменит содержимое существующего файла.

Перед тем, как перенаправить вывод команды в конкретный файл убедитесь, что этот файл не существует. shell не предупреждает, что выполняет перезапись существующего файла.

Чтобы убедиться, что файл с запланированным именем не существует, воспользуйтесь командой ls с аргументом "имя\_файла". Если этот файл не существует, то ls выдаст сообщение, что файл не найден в текущем справочнике. Например, проверка существования файлов temp и junk даст следующий результат:

$ ls temp<CR>

temp

$ ls junk<CR>

junk: no such file or directiry

$

Это означает, что вы можете назвать свой файл junk, но не можете использовать в качестве имени temp, если не хотите потерять содержимое существующего файла.

#### 8.3.3. Добавить вывод в существующий файл

Чтобы добавить вывод в существующий файл и не разрушить его, вы можете воспользоваться символом >>:

command >> имя\_файла<CR>

В результате выполнения команды вывод будет добавлен в конец существующего файла. Если файл не существует, то он будет создан.

***13)Конвейеры команд***

Конвейер. Взять стандартный вывод одной программы и использовать как стандартный ввод другой. Это одна из самых богатых возможностей ОС UNIX; ее можно использовать для создания собственных команд. Предположим, Вам нужен список всех файлов, принадлежащих **edk**. Эту операцию нельзя выполнить с помощью ключей команды **ls**, но, используя конвейер, можно объединить команды **ls -l** и **grep**:

% ls -ls | grep "edk"

Команда ls -l выдает список всех файлов, включая информацию о владельцах, а grep извлекает все пункты, содержащие строку **edk**.

*Конвейеры* — это возможность нескольких программ работать совместно, когда выход одной программы непосредственно идет на вход другой без использования промежуточных временных файлов. Синтаксис:

команда1 | команда2

Выполняет команду1 используя её поток вывода как поток ввода при выполнении команды2, что равносильно использованию двух перенаправлений и временного файла:

команда1 > ВременныйФайл

команда2 < ВременныйФайл

rm ВременныйФайл

***14)Фоновые процессы: запуск, остановка, слежение. Приостановка и возобновление emacs.***

***Фоновой процесс — это процесс, который работает в фоне, на заднем плане. Имеется в виду, что оболочка операционной системы, которая выполняет фоновый процесс, не ждёт завершения или окончания процесса, как это происходит с обычными программами. Оболочка может запустить ещё много процессов сразу после запуска одного фонового так, что они будут выполняться одновременно.***

Типичными фоновыми процессами, выполняющимися в системе, являются обработчики событий и системные службы.

Оболочка ОС UNIX подразделяет запущенные ей группы процессов на «переднего плана», «фоновые» и «приостановленные», и поддерживает перевод групп процессов из одного из выше названных классов в другой. Для этого используется & (амперсенд) в конце командной строки, клавиатурная комбинация Ctrl-Z (приостанавливает текущую группу процессов переднего плана), и команды jobs, fg (переносит описанное задание на передний план и делает его текущим) и bg (размещает описанное задания на заднем плане).

Обычно для вызова Emacs достаточно просто набрать `emacs' в оболочке. Если запускать Emacs из окна с оболочкой в системе X Windows, следует запускать его в фоновом режиме с помощью `emacs&'. Тогда Emacs не свяжет окно оболочки, что позволит выполнять другие команды, пока Emacs работает в своих X-окнах.

Приостановка означает временную остановку Emacs и возврат управления его родительскому процессу (обычно это оболочка), она позволяет позднее возобновить редактирование в том же задании Emacs. Чтобы приостановить Emacs, следует набрать C-z (suspend-emacs). Это вернет пользователя обратно в оболочку, из которой вызывали Emacs. Можно возобновить Emacs с помощью команды `%еmacs' в большинстве обычных оболочек.

***15)Понятие о программировании на ИКЯ.***

***Командные языки и командные интерпретаторы***

Как и в большинстве интерактивных систем, традиционный интерфейс с пользователем ОС UNIX основан на использовании командных языков. Выражаясь несколько тавтологично, можно сказать, что командный язык - это язык, на котором пользователь взаимодействует с системой в интерактивном режиме. Такой язык называется командным, поскольку каждую строку, вводимую с терминала и отправляемую системе, можно рассматривать как команду пользователя по отношению к системе. Одним из достижений ОС UNIX является то, что командные языки этой операционной системы являются хорошо определенными (не очень хороший русский термин, соответствующий совершенно однозначному английскому термину well-defined) и содержат много средств, приближающих их к языкам программирования.

## Общая характеристика командных языков

В этом пункте и далее в данном разделе мы будем более конкретно говорить о командных языках семейства shell. Основное назначение этих языков (их разновидностей существует достаточно много, но мы рассмотрим только три наиболее распространенные варианта - Bourne-shell, C-shell и Korn-shell) состоит в том, чтобы предоставить пользователям удобные средства взаимодействия с системой.

Bourne-shell

Bourne-shell является наиболее распространенным командным языком (и одновременно командным интерпретатором) системы UNIX. Вот основные определения языка Bourne-shell (конечно, мы приводим неформальные определения, хотя язык обладает вполне формализованным синтаксисом):

* Пробел - это либо символ пробела, либо символ горизонтальной табуляции.
* Имя - это последовательность букв, цифр или символов подчеркивания, начинающаяся с буквы или подчеркивания.
* Параметр - имя, цифра или один из символов \*, @, #, ?, -, $, !.
* Простая команда - последовательность слов, разделенных пробелами. Первое слово простой команды - это ее имя, остальные слова - аргументы команды (имя команды считается ее нулевым аргументом - см. п. 5.2.1).
* Метасимволы. Аргументы команды (которые обычно задают имена файлов) могут содержать специальные символы (метасимволы) "\*", "?", а также заключенные в квадратные скобки списки или указанные диапазоны символов. В этом случае заданное текстовое представление параметра называется шаблоном. Указание звездочки означает, что вместо указанного шаблона может использоваться любое имя, в котором звездочка заменена на произвольную текстовую строку. Задание в шаблоне вопросительного знака означает, что в соответствующей позиции может использоваться любой допустимый символ. Наконец, при использовании в шаблоне квадратных скобок для генерации имени используются все указанные в квадратных скобках символы. Команда применяется в цикле для всех осмысленных сгенерированных имен. Метасимволы – символы, порождающие другие
* Значение простой команды - это код ее завершения, если команда заканчивается нормально, либо 128 + код ошибки, если завершение команды не нормальное (все значения выдаются в текстовом виде).
* Команда - это либо простая команда, либо одна из управляющих конструкций (специальных встроенных в язык конструкций, предназначенных для организации сложных shell-программ).
* Командная строка - текстовая строка на языке shell.
* shell-процедура (shell-script) - файл с программой, написанной на языке shell.
* Конвейер - последовательность команд, разделенных символом "|". При выполнении конвейера стандартный вывод каждой команды конвейера, кроме последней, направляется на стандартный вход следующей команды. Интерпретатор shell ожидает завершения последней команды конвейера. Код завершения последней команды считается кодом завершения всего конвейера.
* Список - последовательность нескольких конвейеров, соединенных символами ";", "&", "&&", "||", и, может быть, заканчивающаяся символами ";" или "&". Разделитель между конвейерами ";" означает, что требуется последовательное выполнение конвейеров; "&" означает, что конвейеры могут выполняться параллельно. Использование в качестве разделителя символов "&&" (и "||") означает, что следующий конвейер будет выполняться только в том случае, если предыдущий конвейер завершился с кодом завершения "0" (т.е. абсолютно нормально). При организации списка символы ";" и "&" имеют одинаковые приоритеты, меньшие, чем у разделителей "&&" и "||".
* В любой точке программы может быть объявлена (и установлена) переменная с помощью конструкции "имя = значение" (все значения переменных - текстовые). Использование конструкций $имя или ${имя} приводит к подстановке текущего значения переменной в соответствующее слово.
* Предопределенными переменными Bourne-shell, среди прочих, являются следующие:
* HOME - полное имя домашнего каталога текущего пользователя;
* PATH - список имен каталогов, в которых производится поиск команды, при ее указании коротким именем;
* PS1 - основное приглашение shell ко вводу команды;

и т.д.

* Вызов любой команды можно окружить одиночными кавычками (`), и тогда в соответствующую строку будет подставлен результат стандартного вывода этой команды.
* Среди управляющих конструкций языка содержатся следующие: for и while для организации циклов, if для организации ветвлений и case для организации переключателей (естественно, все это специфически приспособлено для работы с текстовыми значениями).

C-shell

Командный язык C-shell главным образом отличается от Bourne-shell тем, что его синтаксис приближен к синтаксису языка Си (это, конечно, не означает действительной близости языков). В основном, C-shell включает в себя функциональные возможности Bourne-shell. Если не вдаваться в детали, то реальными отличиями C-shell от Bourne-shell является поддержка протокола (файла истории) и псевдонимов.

В протоколе сохраняются введенные в данном сеансе работы с интерпретатором командные строки. Размер протокола определяется установкой предопределенной переменной history, но последняя введенная командная строка сохраняется всегда. В любом месте текущей командной строки в нее может быть подставлена командная строка (или ее часть) из протокола.

Кроме того, в C-shell по сравнению с Bourne-shell существенно расширен набор предопределенных переменных, а также введены более развитые возможности вычислений (по-прежнему, все значения представляются в текстовой форме).

Редактирование текстов

2. Строчное редактирование. Концепция. Пример: основные возможности стандартного строчного редактора ОС UNIX ed.

Понятие о редакторе тексwтов. Основные типы редакторов текстов (строчный, экранный, графический, пакетные и текстовые процессоры).

Текстовый редактор — компьютерная программа, предназначенная для работы с текстовыми файлами — их создания, просмотра, внесения изменений, распечатки и т. п.

· Строчный редактор - тип текстовых редакторов, позволяющий выполнять редактирование только в указанной строке. Строчный редактор не дает такой свободы перемещения как полностраничный редактор, а работает с указателями строк и положения курсора в строке. Пользователь вводит необходимые изменения внизу экрана.

· Экранный редактор - текстовый редактор, который позволяет отображать и редактировать текст на экране с использованием позиции курсора.

· Графический редактор — специализированная программа, предназначенная для создания и обработки изображений. Растровые графические редакторы позволяют пользователю рисовать и редактировать изображения на экране компьютера, а также сохранять их в различных растровых форматах.

· Текстовый процессор — компьютерная программа, используемая для написания и модификации документов, компоновки макета текста и предварительного просмотра документов в том виде, в котором они будут напечатаны. Текстовые процессоры ориентированы на оформление и форматирование текстов и внедрение в них сторонних объектов (шрифтов, таблиц, формул, графиков т. п.). Поскольку в текстовом формате не предусмотрено хранение информации об оформлении текста, текстовые процессоры работают либо с файлами, в которых тексты представлены в обрамлении какого-либо языка разметки вроде HTML, либо с файлами в собственных «двоичных» форматах.

***2. Строчное редактирование. Концепция. Пример: основные возможности стандартного строчного редактора ОС UNIX ed.***

*Ed* ***— первый стандартный текстовый редактор операционной системы UNIX, применялся в начале 1970-х. Имеет командно-ориентированный интерфейс, поскольку создавался в те времена, когда мониторов не существовало и стандартным средством ввода-вывода был телетайп (электромеханическая печатная машина, используемая для передачи между двумя абонентами текстовых сообщений по простейшему электрическому каналу).***

Построчный редактор становится доступным при помощи команды ed и является разновидностью программ для подготовки текстовых файлов. Он называется построчным, т.к. манипулирует с текстом по принципу "строка-за-строкой". Это означает, что вы должны указать номер строки, текст которой вы хотите изменить. Затем ed печатает эту строку на экране и вы можете ее модифицировать.

Имеется возможность замены слова или части слова в тексте как для одной указанной строки, так и для части файла или всего файла. Кроме того, можно просмотреть текстовый файл, напечатать его частично или целиком, напечатать строки, содержащие определенное слово и т.д.

В общем случае команды редактора ed состоят из имени команды, которому могут предшествовать один или два номера строк, и после которого для команд "e", "r" и "w" может указываться имя файла. В строке может присутствовать только одна команда, однако команда "p" может следовать после любой другой команды, кроме "e", "r", "w" и "q".

a -

Дописать строки в буфер редактора после текущей строки, (если не задана другая строка). Дозапись продолжается до тех пор, пока не будет введена команда ".<CR>". Текущей становится последняя дописанная строка. В этой команде допускается адрес "0", при этом текст помещается в начало буфера.

c -

Заменить указанные строки новым текстом, следующим после команды "c". Ввод новых строк завершается ".<CR>". Если номера строк в команде не указаны, заменяется текущая строка. Текущей становится последняя замененная строка или (если строк замены не было) строка, предшествующая удаленным строкам.

d -

Удалить указанные строки. Если номера строк не заданы, удаляется текущая строка. Текущей становится первая неудаленная строка, кроме случая, когда удаляется последняя строка буфера. В этом случае текущей становится новая последняя строка буфера.

e -

Релактирование нового файла. Предшествующее содержимое буфера теряется, так что предварительно полезно ввести команду "w". Текущей становится последняя строка буфера.

E -

Команда выполняет те же действия, что и "e", за исключением того, что не выдается сообщение, если после последнего изменения буфера не была выдана команда записи "w".

f -

Выдать текущее имя файла. Если после команды "f" указывается имя файла, текущее имя файла будет заменено этим именем.

3. Экранное редактирование. Концепция. Примеры экранных редакторов ОС UNIX (vi, vim, emacs, ted).

Редактор Viявляется мощным средством для создания и редактирования файлов. Он предназначен для работы на видеотерминале и использует окна, посредством которых вы можете просматривать текст файла. Несколько простых команд позволяют вам сделать изменения в тексте, которые очень быстро отображаются на экране. Редактор может отобразить одну или несколько строк текста. Он позволяет вам передвигать курсор в любую точку на экране или в файле, создавать, изменять или удалять текст. Вы можете также использовать некоторые команды построчного редактора, такие как глобальные команды, которые позволяют вам изменять множество появлений некоторой строки символов, используя одну команду. Чтобы двигаться по файлу, вы можете прокручивать текст вперед или назад.

Vim — свободный текстовый редактор, созданный на основе более старого vi. Ныне это один из мощнейших текстовых редакторов с полной свободой настройки и автоматизации, и созданным благодаря этому расширениям и надстройкам. Пользовательский интерфейс Vim’а может работать в чистом текстовом (консольном) режиме. Одна из главных особенностей редактора — применение двух основных, вручную переключаемых, режимов ввода: командного (после запуска редактор находится в нём) и текстового (режим непосредственного редактирования текста, аналогичный большинству «обычных» редакторов).

Emacs - один из наиболее мощных и широко распространённых редакторов, используемых в мире Unix. По популярности он соперничает с редактором vi и его клонами. В зависимости от ситуации, Emacs может быть:

* текстовым редактором;
* программой для чтения почты и новостей Usenet;
* интегрированной средой разработки (IDE);
* операционной системой;
* всем, чем угодно.

Всё это разнообразие достигается благодаря архитектуре Emacs, которая позволяет расширять возможности редактора при помощи языка Emacs Lisp.

Редактор Ted задуман автором (Mark de Does) как простой текстовый редактор, работающий под X Window в Unix/Linux системах. Основное преимущество Ted - это возможность не только вводить и редактировать текст, но и обеспечивать простейшие возможности форматирования текста, изменять шрифты, вставлять в текст таблицы и рисунки

4. Краткая историческая справка по редактору emacs. Особенности реализации.

Краткая историческая справка по редактору Emacs. Особенности реализации.

Текстовый редактор Emacs разработан выдающимися программистами (Ричардом Сталлманом и др.) в Массачусетском технологическом институте и включен в качестве стандартного текстового редактора в состав программного обеспечения GNU-проекта. Emacs легко переносим, версии Emacs существуют практически на всех платформах. Emacs написан на Си (интерфейная часть) и на диалекте Лиспа (функциональная часть), что обеспечивает его гибкость вплоть до полной перепрограммируемости под другой редактор. Emacs поддерживает X Window System.

Интересно, что слово «Emacs» берет начало в аббревиатуре «Editor MACroS», наборе макросов для редактора TECO, написанном Столлманом и другими в 1976 году.

Основу идеологии Emacs составляют принципы «всё в одном», расширяемости, настраиваемости под пользователя и документированности.

· Функциональность Emacs можно разделить на три уровня: базовая, основной режим и дополнительные режимы. Сочетание этих трёх уровней позволяет выполнять огромное количество задач (в том числе и не связанных напрямую с редактированием текста), не выходя из редактора.

· Пользователи могут сами создавать части Emacs, от отдельных функций до новых основных режимов. При этом можно переопределять любые Elisp-функции, в том числе и те, что являются частью самого редактора. Так что можно легко модифицировать функциональность Emacs, просто соответствующим образом «подправив» некоторые функции.

· Emacs написан с учётом того, что разные пользователи имеют разные предпочтения. Различные параметры редактирования можно изменять, используя Elisp-переменные. Кроме того, для удобства, а также для тех, кто не знает Elisp, предусмотрен режим настройки, где значения параметров можно вводить в специальные поля или выбирать из списков.

· Emacs обладает подробной документацией, включая и документацию по Elisp. Но особенностью Emacs является то, что каждая функция или переменная может быть снабжена документацией при своём определении. Такую документацию можно просмотреть просто указав имя функции или переменной.