Семинар 1. Работа пользователей с ОС UNIX в терминальном режиме

1. Терминальный режим

Терминальный режим работы с компьютером, управляемым ОС UNIX (Linux), подразумевает прямое/сетевое подключение терминала/компьютера пользователя к этому компьютеру (Рис. 1).

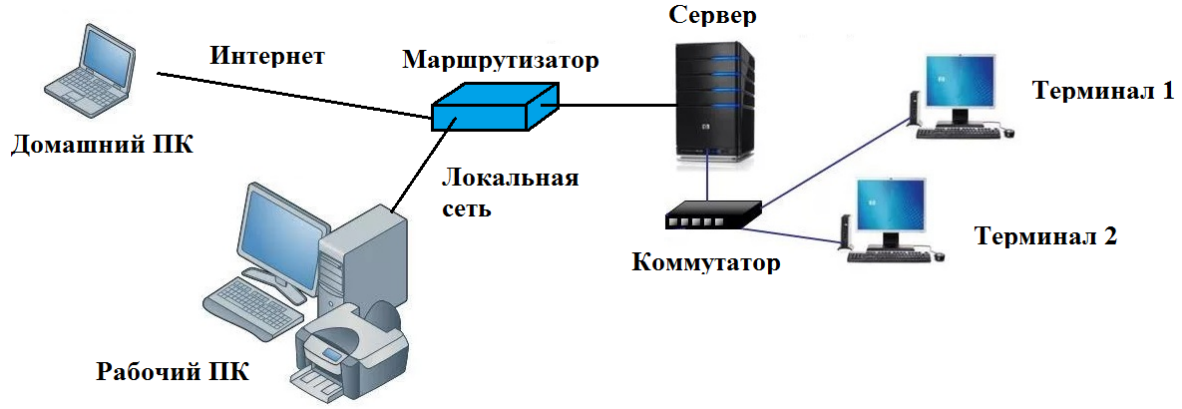

Рисунок 1. Схема работы пользователей с компьютером, управляемым ОС UNIX.

2. Устройство пользовательских аккаунтов

Все пользователи в ОС UNIX имеют персональные учетные записи - аккаунты.
Пример. Имеется пользователь – Иванов Иван Иванович.
Системный администратор для работы Иванова И.И.:
- создал учетную запись (аккаунт, логин, login): **ivanov**
- задал стартовый пароль для входа (password): **Iv@Nov#2022**
- создал домашний каталог: **/home/ivanov**
- привязал к логину требуемую командную оболочку (Shell): **/bin/bash**
- разрешил вход по протоколу ssh.

3. Работа пользователя в терминальном режиме

Работа пользователя начинается с процедуры авторизации на сервере организации (см. Рис. 2-4) с помощью программ telnet, rlogin, rsh, ssh (в ОС Windows можно использовать программу putty.exe с сайта http://putty.org). В случае правильности всех необходимых данных (логин – специальное имя пользователя, пароль, ключ доступа) пользователь получает удаленное терминальное окно, обслуживаемое командным процессором shell.

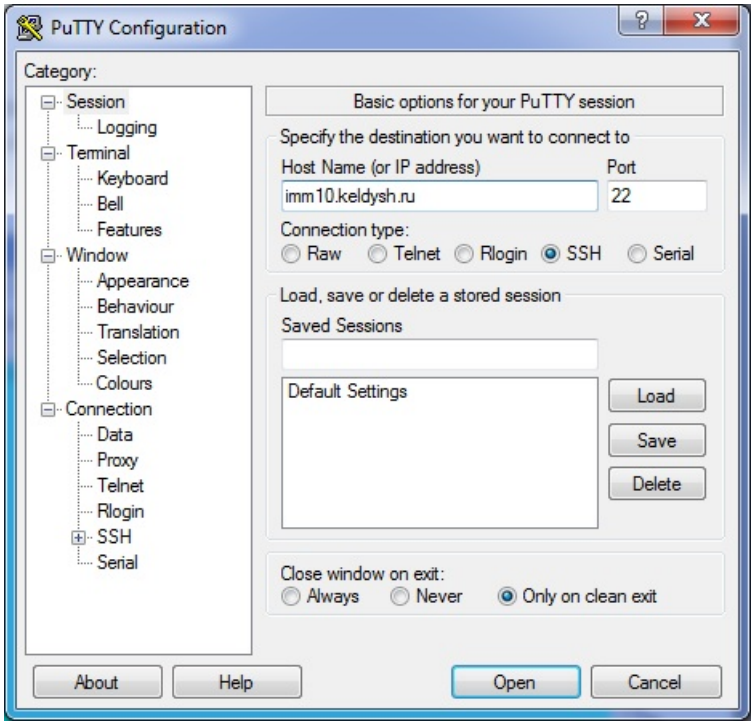

Рисунок 2. SSH-клиент putty.

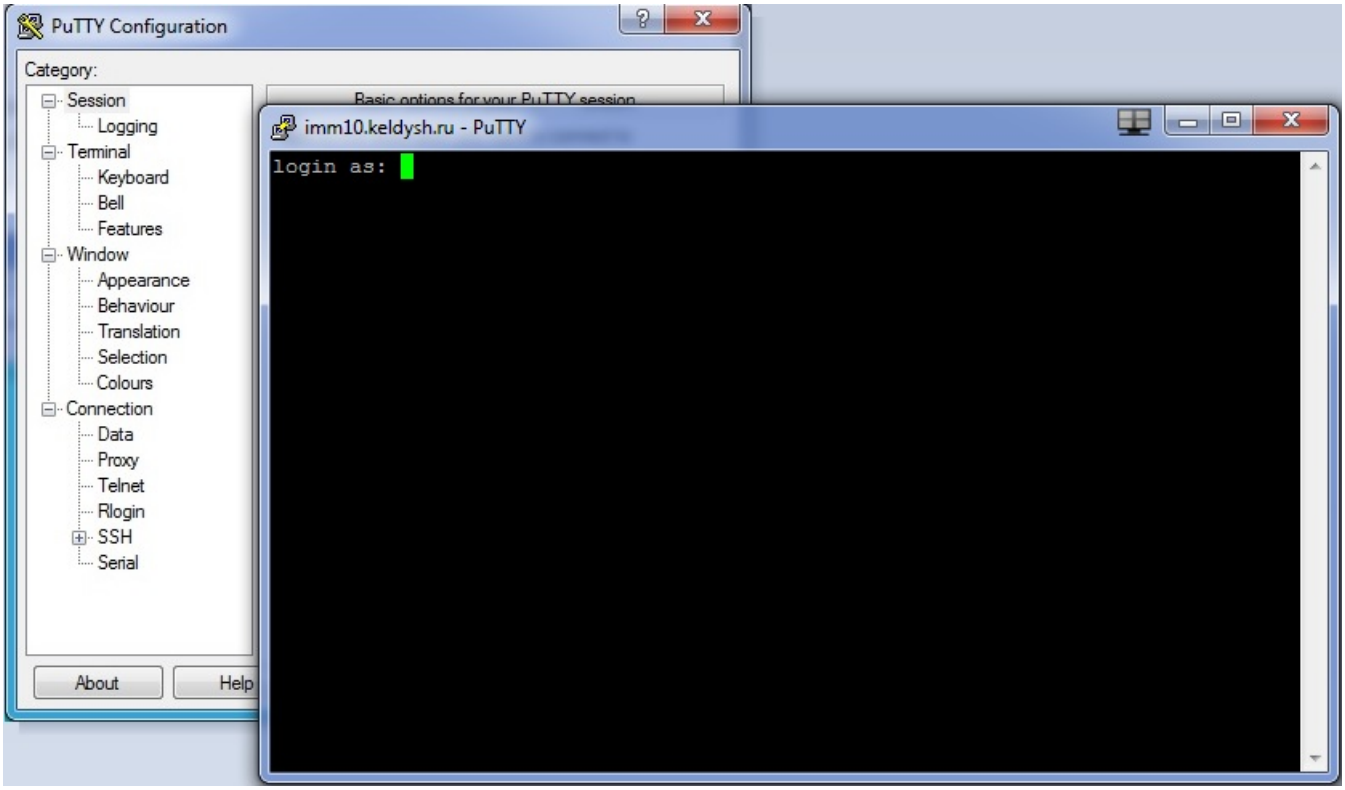

Рисунок 3. Вид удаленного терминального окна перед входом.

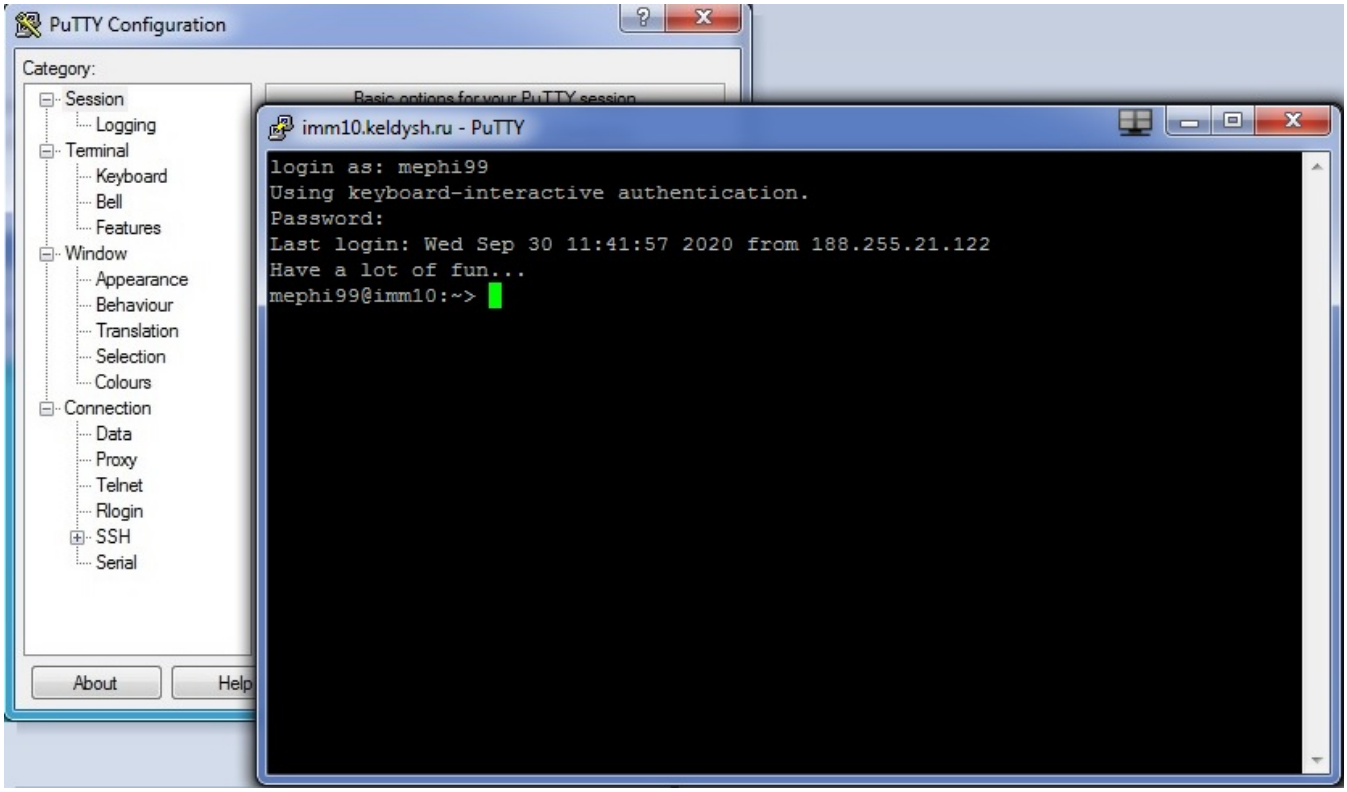

Рисунок 4. Вид удаленного терминального окна после авторизации.

4. Программы и команды

Программа – это набор инструкций для компьютера, хранящийся либо на внутреннем или внешнем носителе в виде исполняемого файла. Программы, которые могут быть выполнены компьютером без предварительной трансляции, называются исполняемыми программами или ***командами***. Пользователю системы UNIX доступно множество стандартных программ и инструментальных средств (пакетов программ).

Ввод и анализ команд пользователя (см. Рис. 5) осуществляют совместно строковый редактор и командный процессор (shell или его разновидности: sh, csh, tcsh, zcsh, bash и др.).

Чтобы ваш запрос был понятен системе UNIX вы должны ввести команду в корректном формате или синтаксисе командной строки. Этот синтаксис определяет порядок, в котором вы вводите компоненты командной строки. И вы должны расположить все составные части командной строки в требуемом синтаксисом порядке, иначе shell не сможет интерпретировать ваш запрос.

Пример синтаксиса командной строки:

> command option(s) argument(s) <CR>

Для каждой командной строки необходимо ввести как минимум два компонента: имя команды и клавишу <RETURN> или <ENTER> (Обозначение <CR> используется в документации как инструкция для нажатия этой клавиши). Командная строка может также содержать ключи и аргументы. В указанном примере синтаксиса командной строки:

command – это имя программы, которую вы хотите выполнить;

option – ключи, которые указывают как запустить команду;

argument – указывает на данные, которые эта команда обрабатывает, обычно это имя каталога или файла.

В командной строке, которая включает ключи и/или аргументы, каждый компонент отделяется друг от друга по крайней мере одним пробелом. Если аргумент содержит пробел, его надо заключить в двойные кавычки. Например, если аргумент sample 1, то необходимо указать его как "sample 1". Если не поставить двойные кавычки, то командный процессор shell будет интерпретировать sample и 1 как два отдельных аргумента. Некоторые команды позволяют указать несколько аргументов в одной командной строке:

>	ls	-l -i	file1 file2 file3	<CR>
приглашение	команда	ключи	аргументы	конец строки (команды)

Здесь команда ls использует два ключа -l и -i и три аргумента file1, file2 и file3. Ключ -l обеспечивает информацию в длинном формате, включая режим, владельца и размер, а ключ -i печатает номер inode. Командный процессор позволяет вам группировать ключи, например -li, и выводить их в любом порядке. Этого нельзя делать с аргументами.

В конце командной строки необходимо нажать клавишу <CR>.

Чтобы выполнить команду, введите командную строку, когда на экране появится приглашение (например, символ # или >). Shell рассмотрит вашу команду, найдет соответствующую ей программу и перенаправит запрос ядру ОС. Ядро ОС запустит программу с необходимыми аргументами и тем самым выполнит вашу команду. После завершения выполнения команды shell сигнализирует, что готов выполнить следующую команду, напечатав приглашение (см. Рис. 5).

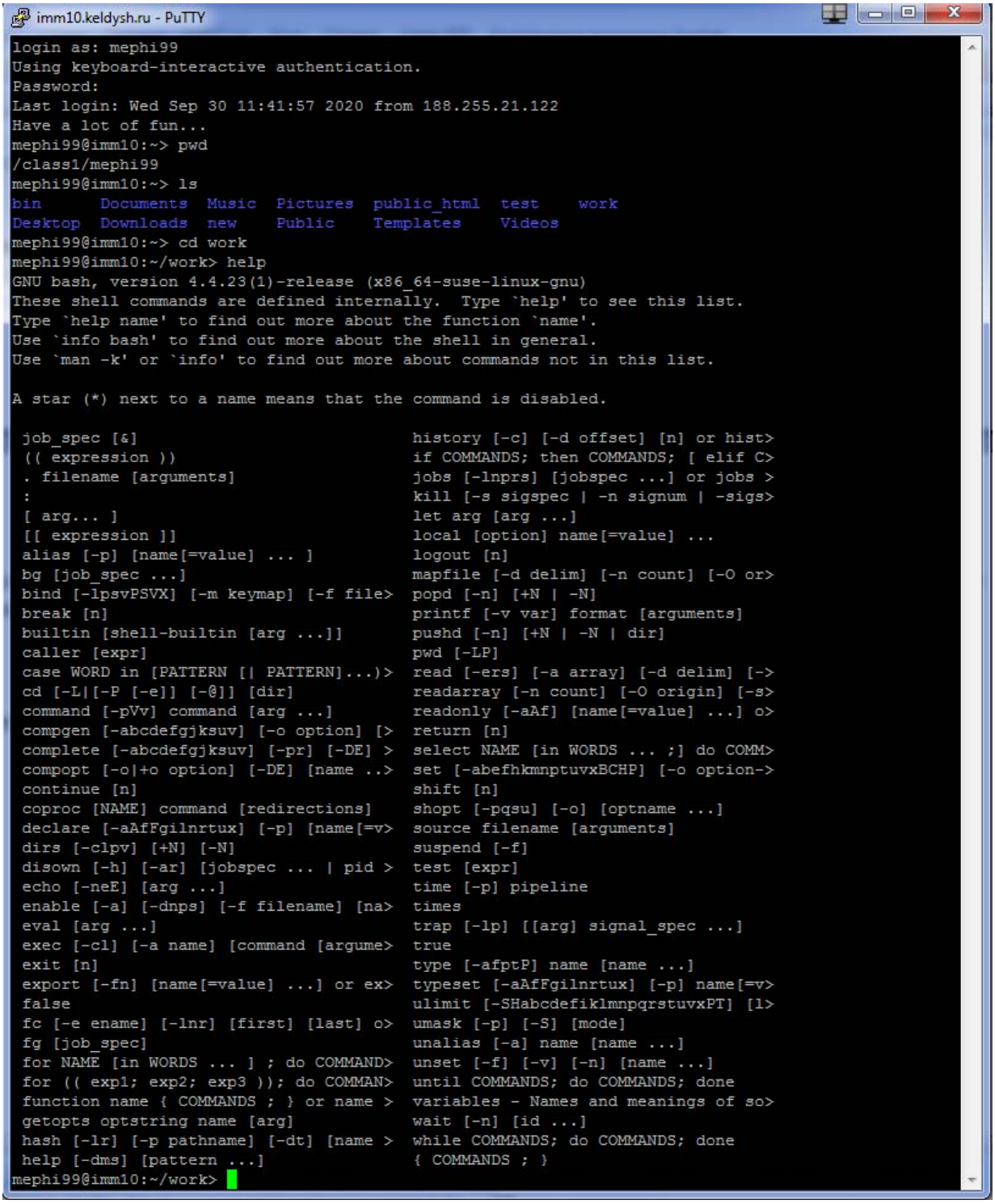

Рисунок 5. Ввод команд.

5. Язык и команды shell

Главная задача shell – это анализ команд пользователя в терминальном режиме и передача их на выполнение ядру ОС, а также обработка так называемых пакетных файлов (скриптов или сценариев). В последнем случае shell выступает в качестве интерпретатора команд специального языка. Язык shell содержит стандартные средства для объявления переменных и функций, организации циклов обработки данных, ветвлений сценария и т.п. Файлы со сценариями являются текстовыми и часто имеют расширение ".sh". Язык shell часто используется в UNIX-подобных системах при создании различных сценариев работы, в частности, сценариев автоматического конфигурирования исходных кодов программ перед их компиляцией.

Отличительная особенность языка shell – многие операции, которые в традиционных языках программирования являются встроенными, выполняются с помощью вызова внешних программ. Поэтому все команды shell можно подразделить на внутренние и внешние.

Внутренние команды shell – это команды, которые выполняются непосредственно самим командным процессором. ***Внешние команды*** shell – остальные команды, доступные пользователю не только через командный процессор. Внешние команды подразделяются на ***системные*** (реализованные в текущей версии операционной системы и доступные всем пользователям) и ***пользовательские*** (разработанные самим пользователем). Например, команда справки "help" является внутренней (дает справку только по shell), а команда "man sh" является внешней, так как дает справку по любой команде ОС.
Как уже сказано выше, о внутренних командах shell можно узнать с помощью системы справки:
>help
Из выведенной командой help информации можно узнать о конструкциях языка shell. По команде
>man sh
выводится информация, из которой видно, что shell это тоже команда с ключами и аргументами.
Примерами других внешних команд служат:
pwd, chdir, ls, ... – команды навигации пользователя по файловой системе и обзора содержимого конкретных каталогов;
expr используется для вычисления арифметических выражений;
test используется для сравнения чисел и строк, а также для определения наличия или атрибутов файлов;
sed, tr, awk, head, tail, cut и другие – для работы с текстом.
Краткая сводка часто используемых внешних команд дана на Рис. 6. Более полная информация доступна, например, по ссылке: https://putty.org.ru/articles/unix-linux-ref.html

Файловые команды

`ls` – список файлов и каталогов
`ls -al` – форматированный список со скрытыми каталогами и файлами
`cd dir` – сменить директорию на *dir*
`cd` – сменить на домашний каталог
`pwd` – показать текущий каталог
`mkdir dir` – создать каталог *dir*
`rm file` – удалить *file*
`rm -r dir` – удалить каталог *dir*
`rm -f file` – удалить форсированно *file*
`rm -rf dir` – удалить форсированно каталог *dir* *
`cp file1 file2` – скопировать *file1* в *file2*
`cp -r dir1 dir2` – скопировать *dir1* в *dir2*; создаст каталог *dir2*, если он не существует
`mv file1 file2` – переименовать или переместить *file1* в *file2*. если *file2* существующий каталог - переместить *file1* в каталог *file2*
`ln -s file link` – создать символическую ссылку *link* к файлу *file*
`touch file` – создать *file*
`cat > file` – направить стандартный ввод в *file*
`more file` – вывести содержимое *file*
`head file` – вывести первые 10 строк *file*
`tail file` – вывести последние 10 строк *file*
`tail -f file` – вывести содержимое *file* по мере роста, начинает с последних 10 строк

Управление процессами

`ps` – вывести ваши текущие активные процессы
`top` – показать все запущенные процессы
`kill pid` – убить процесс с id *pid*
`killall proc` – убить все процессы с именем *proc* *
`bg` – список остановленных и фоновых задач; продолжить выполнение остановленной задачи в фоне
`fg` – выносит на передний план последние задачи
`fg n` – вынести задачу *n* на передний план

Права доступа на файлы

`chmod octal file` – сменить права *file* на *octal*, раздельно для пользователя, группы и для всех добавлением:

- 4 – чтение (r)
- 2 – запись (w)
- 1 – исполнение (x)

Примеры:
`chmod 777` – чтение, запись, исполнение для всех
`chmod 755` – rwx для владельца, rx для группы и остальных.
Дополнительные опции: `man chmod`.

SSH

`ssh user@host` – подключится к *host* как *user*
`ssh -p port user@host` – подключится к *host* на порт *port* как *user*
`ssh-copy-id user@host` – добавить ваш ключ на *host* для *user* чтобы включить логин без пароля и по ключам

Поиск

`grep pattern files` – искать *pattern* в *files*
`grep -r pattern dir` – искать рекурсивно *pattern* в *dir*
`command | grep pattern` – искать *pattern* в выводе *command*
`locate file` – найти все файлы с именем *file*

Системная информация

`date` – вывести текущую дату и время
`cal` – вывести календарь на текущий месяц
`uptime` – показать текущий аптайм
`w` – показать пользователей онлайн
`whoami` – имя, под которым вы залогинены
`finger user` – показать информацию о *user*
`uname -a` – показать информацию о ядре
`cat /proc/cpuinfo` – информация ЦПУ
`cat /proc/meminfo` – информация о памяти
`man command` – показать мануал для *command*
`df` – показать инф. о использовании дисков
`du` – вывести "вес" текущего каталога
`free` – использование памяти и swap
`whereis app` – возможное расположение программы *app*
`which app` – какая *app* будет запущена по умолчанию

Архивация

`tar cf file.tar files` – создать tar-архив с именем *file.tar* содержащий *files*
`tar xf file.tar` – распаковать *file.tar*
`tar czf file.tar.gz files` – создать архив tar с сжатием Gzip
`tar xzf file.tar.gz` – распаковать tar с Gzip
`tar cjf file.tar.bz2` – создать архив tar с сжатием Bzip2
`tar xjf file.tar.bz2` – распаковать tar с Bzip2
`gzip file` – сжать *file* и переименовать в *file.gz*
`gzip -d file.gz` – разжать *file.gz* в *file*

Сеть

`ping host` – пропинговать *host* и вывести результат
`whois domain` – получить информацию whois для *domain*
`dig domain` – получить DNS информацию *domain*
`dig -x host` – реверсивно искать *host*
`wget file` – скачать *file*
`wget -c file` – продолжить остановленную закачку

Установка пакетов

Установка из исходников:
`./configure`
`make`
`make install`
`dpkg -i pkg.deb` – установить пакет (Debian)
`rpm -Uvh pkg.rpm` – установить пакет (RPM)

Клавиатурные сочетания

`Ctrl+C` – завершить текущую команду
`Ctrl+Z` – остановить текущую команду, продолжить с `fg` на переднем плане или `bg` в фоне
`Ctrl+D` – разлогиниться, тоже самое, что и `exit`
`Ctrl+W` – удалить одно слово в текущей строке
`Ctrl+U` – удалить строку
`!!` - повторить последнюю команду
`exit` – разлогиниться

Рисунок 6. Краткая сводка команд UNIX/Linux.

6. Пример работы пользователя в терминальном режиме.