1. РЕГИСТРАЦИЯ И ПОДКЛЮЧЕНИЕ К СИСТЕМЕ UNIX. ОБЩИЕ ПРИНЦИПЫ РАБОТЫ

1.1 Общие сведения

Каждый пользователь, работающий с системой UNIX, взаимодействует с ней при помощи *терминала*. Под терминалом подразумевается не только устройство ввода-вывода данных, подсоединённое локально или удалённо к процессорной системе, исполняющей код UNIX, но и различные программы-эмуляторы терминалов, позволяющие подключиться к UNIX и использовать устройства ввода-вывода компьютера (клавиатура и экран), на котором запущена эта программа, как удалённый терминал UNIX. Устройство ввода-вывода данных, являющееся частью процессорной системы, исполняющей код UNIX, называется *системной консолью*.

Различают *графические* и *текстовые терминалы*.

Графический терминал предоставляет пользователю возможность запускать программы с графическим интерфейсом и работать с ними с помощью мыши и различных объектов экрана (кнопки, окна, полосы прокрутки, выпадающие меню и другие атрибуты GUI). Наиболее распространена реализация графического терминала X Window System. (Терминалы, поддерживающие эту спецификацию, называют *X-терминалами.*) Графический терминал может быть реализован как на системной консоли, так и удалённо. В данной работе режим графического терминала не рассматривается.

Текстовый терминал крайне нетребователен к аппаратным средствам, поэтому и аппаратные реализации текстового терминала, и программы-эмуляторы для других компьютеров очень просты. Реализации текстовых терминалов различаются по своим возможностям: одни терминалы поддерживают цвета, мышь и псевдографические примитивы, на других остаётся довольствоваться монохромным режимом с маленьким разрешением. Для программ-эмуляторов текстового терминала распространён стандарт VT100.

Основная особенность UNIX состоит в том, что пользователь может использовать ресурсы компьютера под управлением UNIX, находясь территориально в любом месте (при наличии канала связи). Более того, один и тот же пользователь может параллельно использовать ресурсы нескольких компьютеров с UNIX. И наоборот, ресурсы одного и того же компьютера могут одновременно использовать несколько пользователей. Коммерческие версии UNIX могут накладывать ограничения на число одновременно работающих пользователей, однако, для каждого пользователя количество одновременно работающих задач практически не ограничено.

Промежуток времени между подключением пользователя к системе и отключением от неё называется сеансом (session). В течение сеанса пользователь может выполнять любые доступные в системе команды, в том числе и открывать сеансы с другими системами UNIX.

Условием возможности подключения пользователя является наличие учётной записи пользователя (account) в системе. Добавление новой учётной записи осуществляет администратор. В UNIX ему сопоставляется учётная запись с именем root. При добавлении новой учётной записи в базу данных о пользователях заносится регистрационное имя нового пользователя (login) и устанавливается пароль для его входа в систему, а также некоторые дополнительные системные настройки.

Таким образом, пользователь должен знать определённое для него имя (login) и пароль, а в случае удалённого подключения – имя или IP-адрес нужной системы в сети Internet/Intranet. После запуска программы-эмулятора терминала и выбора системы для подключения происходит установление соединения и появляется приглашение для входа в систему:

```
login:
```

При наборе имени пользователя следует иметь в виду, что UNIX различает строчные и прописные буквы. После ввода регистрационного имени, завершающегося нажатием на [Enter], система предлагает ввести пароль:

```
password:
```

При вводе пароля следует быть особенно внимательным, поскольку он не отображается. Если при регистрации произошла ошибка, система выдаст сообщение:

```
Login incorrect.
login:
```

Как правило, после определённого числа неудачных попыток регистрации система разрывает соединение и работа эмулятора терминала завершается.

Системная консоль современных версий UNIX предоставляет возможность разделять клавиатуру и экран между несколькими *«виртуальными» терминалами*. Для системной консоли не требуется процедуры установки соединения с системой, поэтому каждый виртуальный терминал уже содержит приглашение входа в систему (и, как правило, некоторые сведения о системе, такие как идентификатор терминала). Переключение между виртуальными терминалами осуществляется при помощи комбинаций [Alt]+[F1], [Alt]+[F2][1], … и т. д.

После успешной регистрации система выводит некоторую актуальную информацию, например, кратко сообщает свою версию, имя компьютера и идентификатор назначенного пользователю терминала (TTY), а также

[1] Запись [Alt]+[F1] означает, что клавиши [Alt] и [F1] должны быть нажаты одновременно. Можно сначала нажать [Alt], а затем, не отпуская [Alt], нажать [F1], после чего отпустить обе клавиши.

сообщение о наличии новой почты. После этого появляется приглашение командного интерпретатора системы (оболочки) – система готова принимать команды пользователя:

```
[pupkin@somehost homedir] $
```

Вид приглашения определяется настройками пользователя. Обычно приглашение содержит имя пользователя, название системы и текущий каталог. Завершается приглашение знаком $ или # (для администратора). В последующих примерах знак $ указан для обозначения командной строки и не должен набираться.

В командной строке текстовой оболочки пользователь набирает команды. **Строчные и заглавные буквы в командной строке РАЗЛИЧАЮТСЯ!** Команды, их опции и имена файлов следует указывать в правильном регистре (обычно все строчные). Команды могут быть внешние (исполняемые файлы) или внутренние (встроенные команды оболочки). При выполнении внешних команд надо учитывать, что система должна знать путь к файлу, либо он должен находиться в одной из директорий, указанной в переменной окружения PATH. UNIX не ищет исполняемый файл в текущем каталоге (текущий каталог обозначается точкой), если только путь "./" не присутствует в PATH. Если требуется запустить программу, находящуюся в каталоге, не указанном в PATH, обязательно надо указать путь к исполняемому файлу.

Ввод команды **`exit`** или **`logout`** закончит сеанс работы с системой и приведёт к разрыву связи с выдачей соответствующего сообщения. После этого можно либо снова соединяться с удалённой системой, либо вернуться в исходную операционную среду.

Оболочка хранит историю последних набранных в ней команд. Перемещение по истории команд осуществляется при помощи клавиш вверх [↑] и вниз [↓]. Поиск в истории команд – [Ctrl]+[r]. Нажав эту комбинацию, можно набрать ключевой фрагмент искомой команды, тогда из истории будет подставлена команда, сопоставляющаяся с данным фрагментом. Чтобы просмотреть историю команд целиком, можно воспользоваться встроенной командой **`history`**.

Оболочка также предоставляет средства автозавершения имён (автоматический донабор имени). Если в командной строке набрать часть имени команды или файла и нажать [Tab], оболочка закончит имя или выведет все возможные варианты завершения имени. Если вариантов очень много, пользователю будет выдан запрос:

```
Display all 2515 possibilities? (y or n)
```

Например, если в командной строке набрать **`his`** и нажать [Tab], оболочка дополнит это имя до команды **`history`**. Если же набрать только **`hi`**, то система выдаст несколько вариантов завершения имени.

Очень многие команды работают с данными, поступающими на стандартный ввод программы (набираются на клавиатуре). Если требуется завершить ввод таких данных, следует нажать [Ctrl]+[d] в пустой строке. Например, конец стандартного ввода для оболочки приводит к завершению её работы. Если оболочка была начальной, то это эквивалентно завершению сеанса (**`exit`** или **`logout`**).

Если какая-то программа выводит большой фрагмент текста на экран, можно приостановить её вывод, нажав [Ctrl]+[s]. Возобновление вывода продолжится после нажатия [Ctrl]+[q]. Если система не реагирует на нажатые вами клавиши, проверьте, не нажали ли вы случайно [Ctrl]+[s] (нажмите [Ctrl]+[q]). Пролистать уже выведенную на текстовый терминал информацию можно при помощи комбинаций [Shift]+[PgUp] и [Shift]+[PgDn].

Прервать программу можно при помощи комбинаций [Ctrl]+[c] или [Ctrl]+[\ ]. Завершение всех программ и перезагрузка системы – [Ctrl]+[Alt]+[Del].

Если строка команды длинная, её можно разбить на несколько строк. Для этого в конце каждой строки, кроме последней, следует поместить символ обратного слэша (\ ).

Некоторые символы имеют для интерпретатора специальное значение:

| & ; < > ( ) $ ' \ " ` пробел табуляция перевод строки

Если эти символы необходимо использовать в каком-либо слове командной строки и заблокировать их специальное значение, следует применить экранирование (*quoting*). Есть три способа экранирования: обратный слэш (\ ), апостроф (') и кавычки ("). Неэкранированный обратный слэш блокирует специальное значение следующего за ним символа. Все символы, заключённые между апострофами, экранируются. Внутри такой последовательности символ апострофа недопустим. Заключение символов в кавычки экранирует их, за исключением символов: $ ' \. Первые два символа сохраняют своё специальное значение. Обратный слэш сохраняет специальное значение, только если после него находится один из символов: $ ' " \ или перевод строки.

При обработке командной строки командный интерпретатор выполняет определённые преобразования. После того, как все преобразования выполнены и все специальные символы интерпретированы, символы экранирования из командной строки удаляются.

В приведённых примерах используется встроенная команда **`echo`**, которая выводит на стандартный вывод (экран) указанные ей аргументы:

```
$ echo 1\&2\
> 345
1&2345
$ echo '123
> 45'
123
```

```
45
$ echo  1  2  3  4  5
1 2 3 4 5
$ echo " 1   2"\ \ 3'  4'   5
 1   2  3  4 5
```

Если при работе с терминалом доступна мышь, то выделение мышью фрагмента текста вызывает копирование этого текста в буфер обмена. Текст может быть вставлен в текущей позиции курсора (не курсора мыши!!!) при нажатии на правую кнопку мыши. Некоторые программы используют мышь по-своему.

1.2 Справочная система

В UNIX используются несколько видов справочных систем.

Справочная система **man** содержит информацию о программах, установленных в системе, о форматах конфигурационных файлов, о библиотечных функциях языка Си, а также о некоторых понятиях, используемых в UNIX. Формат команды **man**:

man [*опции*] [*секция*] *имя*

Файлы этой справочной системы разбиты на несколько секций. Если не указана секция, **man** ищет справочный файл с указанным именем по очереди во всех секциях, начиная с первой. В первой секции хранятся справочные файлы команд общего назначения. Во второй, третьей и девятой секциях хранятся справочные файлы для программистов (описание библиотечных функций Си, системных вызовов, функций ядра и др.). Пятая секция хранит описания конфигурационных файлов различных программ. Седьмая секция содержит справку о некоторых понятиях. В восьмой секции описаны специальные системные программы. Примеры:

- **man man** – получение подсказки по ключам команды **man**;
- **man bash** – описание командного интерпретатора bash и его встроенных команд;
- **man 1 printf** – получение подсказки по команде **printf**;
- **man 3 printf** – получение информации о функции printf из стандартной библиотеки языка Си;
- **man 7 ascii** – получение справки о коде ASCII.

Программа допускает прокрутку информации вперёд и назад при помощи клавиш управления курсором. Другие клавиши: [<] – начало файла, [>] – конец файла, [/] – поиск вперёд по шаблону, [?] – поиск назад по шаблону, [n] – продолжить поиск, [N] – продолжить поиск в обратном направлении, [h] – вывести подсказку, [q] – выход.

Чтобы получить подробную подсказку по встроенным командам оболочки (bash), можно воспользоваться **man bash**. Для получения быстрой и короткой подсказки удобнее использовать команду **help**. Вызов **help** без параметров приводит к отображению на экране всех встроенных команд оболочки. Чтобы разобраться с конкретной командой, надо указать её в качестве параметра команды:

- **help help** – получение подсказки по команде **help**;
- **help type** – получение подсказки по команде **type**.

Третий способ получения справочной информации в UNIX – использование гипертекстовой системы **info**. Эта программа предоставляет возможность навигации по ссылкам между связанными документами: переход по ссылке – [Enter], переход на уровень вверх – [u], переход на последнюю просмотренную страницу – [*l*], переход на следующую страницу в разделе – [n] и др.

Кроме того, большинство команд выводят краткую справку, если в их командной строке указать ключ *--help*. Например:

```
man --help
```

При описании синтаксиса команд используются определённые правила. Например, команда **date** выдаёт системную дату и время. Используя различные опции (ключи), можно добиться вывода даты в различных форматах. Администратору эта команда позволяет изменять системную дату и время. Синтаксис команды **date** описан так:

date [*OPTION*]… [**+***FORMAT*]
date [**-u**|**--utc**|**--universal**] [*MMDDhhmm*[[*CC*]*YY*][.*ss*]]

Первый вариант используется для вывода даты и времени в различных форматах, второй – для изменения даты и времени. При описании синтаксиса квадратные скобки [], многоточие, и вертикальная палочка (|) имеют специальный смысл и не должны набираться в командной строке.

Символы, заключённые в квадратные скобки, обозначают необязательное использование. Если применены вложенные квадратные скобки, то символы внутри вложенных квадратных скобок могут быть указаны только наряду с внешними, окружающими эти вложенные скобки символами. Например, формат даты указан в виде [MMDDhhmm[[CC]YY][.ss]] (MM – месяц, DD – день, hh – часы, mm – минуты, CC – столетие, YY – год, ss – секунды). Это означает, что столетие, год и секунды указывать необязательно. То есть дату можно задать в виде: 06181020, что будет означать 18 июня сего года 10 часов 20 минут. Можно дополнительно указать секунды: 061820.34, при этом синтаксис команды требует их отделения от остальной части при помощи точки. Можно указать год: 0618102018 (18 июня 2018 года 10 часов 20 минут). Если указывается столетие (2 цифры), то

оно обязательно должно сопровождаться последующими двумя цифрами года: 061810201994. И наконец, полный вариант: 061810202018.34 – 18 июня 2018 года 10 часов 20 минут 34 секунды. Причём весь код даты заключён в квадратные скобки – это означает, что команда может быть запущена вообще без этого параметра (тогда она просто выведет текущую системную дату на экран).

Вертикальная палочка разделяет эквивалентные или взаимно исключающие последовательности в синтаксисе. То есть команда **date** может быть запущена в одном из трёх вариантов:

```
date -u 06181020
date --utc 06181020
date --universal 06181020
```

В данном случае вертикальная палочка разделяет эквивалентные варианты. Опция *-u* (*--utc*, *--universal*) используется для задания времени по UTC (всеобщее скоординированное время, т. е. время на нулевом меридиане).

Многоточие в описании синтаксиса команды **date** показывает, что в командной строке может быть использовано несколько опций.

Как правило, команды используют опции двух типов: короткие (обозначаются одним символом, начинаются с одиночного минуса) и длинные (обозначаются словами или фразами, начинаются с двух минусов; если использовано несколько слов, слова разделяются одним минусом). Обычно одна и та же функция команды имеет обозначение и в виде короткой опции, и в виде длинной. Например, для команды **date** для короткой опции *-u* имеется два синонима – длинные опции *--utc* и *--universal*. Для опции *-R* (вывод даты в формате стандарта RFC 2822 для электронных сообщений) имеется синоним *--rfc-2822*.

Некоторые опции могут требовать дополнительных параметров. В таком случае для короткой опции её параметр отделяется пробелом, для длинной – знаком присваивания. Например, команде **date** можно указать вывести на экран не текущую дату, а заданную в командной строке (не изменяя при этом системную дату). Дату можно указать при помощи опции *-d* или *--date* (в этом случае она задаётся либо по стандарту ISO 8601 ГГГГ-ММ-ДД, либо по общепринятому в США формату ММ/ДД/ГГГГ):

```
date -d 2018-03-13
date --date=03/13/2018
```

При необходимости опции можно комбинировать, при этом порядок следования опций, как правило, роли не играет:

```
date --rfc-2822 -d 03/13/2018
date --date=2018-03-13 -R
```

Особенностью коротких опций является то, что их можно объединять за общим знаком минуса (однако, с учётом необходимых им параметров):

```
date -Rd 2018-03-13\ 13:22
```

В данном примере последовательность *-dR* будет недопустимой, поскольку опция *-d* требует обязательного параметра.

Обратите внимание, в последнем примере использован символ экранирования \, поскольку параметр для опции *-d* содержит пробел, являющийся для оболочки специальным символом (символом-разделителем параметров).

1.3 Информационные команды

Команды **who** и **w** выводят список подключённых к системе пользователей.

Чтобы получить более подробную информацию о конкретном пользователе, следует использовать команду:

finger [username[@hostname]]

Команда **uname** выводит информацию о системе. Ключ *-a* используется, чтобы получить тип системы, версию ядра, дату его последней компиляции, архитектуру процессора и т. п.

```
$ uname
FreeBSD
$ uname -a
FreeBSD students.soros.karelia.ru 2.2.6-RELEASE FreeBSD 2.2.6-RELEASE #0: Thu Jul 6
15:12:06 MSD 2000 oleg@students.soros.karelia.ru:/usr/src/sys/compile/STUDENTS i386
```

Встроенная команда **type** сообщает тип команды, указанной в качестве параметра: builtin (встроенная) или внешняя (выводит путь к исполняемому файлу):

```
$ type help
help is a shell builtin
$ type man
man is hashed (/usr/bin/man)
```

hostname – вывод доменного имени компьютера. Имя компьютера состоит из нескольких частей, отделённых точками. Слово до первой точки является собственно именем данной системы, а остальная часть – именем *домена*, к которому данный компьютер относится. Упрощённо доменом называют некоторую совокупность компьютеров. Домены выстраивают в некоторую иерархию, так что домены более верхнего уровня включают в себя домены нижних уровней. Эта иерархическая зависимость отражается в имени домена.

```
$ hostname
dims.karelia.ru
```

В приведённом примере собственное имя компьютера – *dims*. Этот компьютер входит в домен второго уровня *karelia.ru*. В свою очередь, имя этого домена говорит о том, что он входит в структуру домена верхнего уровня *ru*.

Команда **tty** выдаёт идентификатор терминала, за которым работает пользователь.

1.4 Многопользовательская защита

Пользователи, которым разрешено входить в систему (регистрироваться), перечислены в учётной базе пользователей. Пользователи объединяются в группы; последние перечислены в учётной базе групп. Каждому пользователю и каждой группе пользователей назначается целочисленный идентификатор (*UID – user identifier, GID – group identifier*). Администратору соответствует UID=0 и GID=0.

Чтобы определить UID и GID пользователя, можно воспользоваться командой **id**. Команды **whoami** и **groups** являются аналогами команды **id** с различными ключами.

Команда **chfn** позволяет пользователю изменить информацию, выдаваемую о нём командой **finger**.

Чтобы сменить пароль, пользователю следует воспользоваться командой **passwd**. При этом если пользователь не является администратором, система сначала запросит у него старый пароль, а лишь потом предложит ввести новый. При наборе пароля знаки на экране не отображаются. Чтобы уменьшить вероятность ошибки, новый пароль набирается два раза. Если слова не совпадают, система оставляет старый пароль.

Если в системе вам доступно несколько учётных записей, возможно временное изменение пользовательского идентификатора при помощи команды:

su [ключи] [пользователь]

При этом запускается оболочка с правами указанного пользователя (если вы не администратор, система запросит пароль для указанного пользователя). Если от другого пользователя надо выполнить лишь одну команду, эту команду можно указать при помощи опции *-c*. Если имя пользователя опущено, подразумевается root.

1.5 Редактор vi

В UNIX много различных текстовых редакторов, таких как **pico**, **emacs**, **ed** и т. п. Одним из самых распространённых является **vi** (visual editor) или **vim** (VI Improved – улучшенный VI). Достаточно часто **vi** используется для редактирования конфигурационных файлов операционной системы. При вызове **vi** можно в качестве аргумента указать имя редактируемого файла. В редакторе **vi** имеются три режима работы: командный, режим вставки и режим последней строки. По умолчанию редактор запускается в командном режиме. Этот режим позволяет использовать определённые команды для редактирования файлов или перехода в другие режимы. Вставка или редактирование текста осуществляется в режиме вставки. Переход из командного режима в режим вставки осуществляется с помощью клавиши [i] или клавиши [Insert]. Для завершения режима вставки и возврата в командный режим – клавиша [Esc]. В режим последней строки программа переходит по клавише [:] из командного режима. В этом режиме в последней строке экрана появляется поле для ввода многосимвольных команд. Чтобы отличать эти команды от обычных команд, перед ними записывают двоеточие.

Начните работу с редактором, вызвав команду **vi**. Клавишей [i] перейдите в режим вставки. Напечатайте краткую информацию о себе. При вставке текста можно напечатать сколько угодно строк, нажимая [Enter] после каждой строки. Можно также корректировать ошибки, используя клавишу [Backspace]. Завершение работы в режиме вставки и возврат в командный режим осуществляется клавишей [Esc]. В командном режиме можно использовать клавиши со стрелками для перемещения по файлу или для тех же целей – [h], [j], [k], [*l*], соответственно, влево, вниз, вверх, вправо.

Есть несколько способов вставки текста, отличных от использования [i]. Нажав на [o], вы начинаете вставку текста в строку ниже текущей. В командном режиме клавиша [x] удаляет символ перед курсором. Вы можете удалять целые строки, набирая команду `dd` (т. е. нажимая [d] дважды). Чтобы удалить слово, на котором находится курсор, используйте команду `dw`. Вы можете заменить фрагменты текста, используя клавишу [R]. Использование [R] для редактирования текста очень походит на клавиши [i] и [o], но [R] заменяет прежний текст вместо вставки в него. Клавиша [r] позволяет заменить один символ, отмеченный курсором. Клавиша [~] (знак тильда) изменяет регистр буквы, отмеченной курсором: заглавную делает строчной и наоборот. Клавиша [w] перемещает курсор на начало следующего слова, [b] перемещает на начало предыдущего слова. Клавиша [0] (ноль) передвигает курсор на начало текущей строки, а клавиша [$] перемещает на конец строки. Нажатием [Ctrl]+[f] курсор перемещается на экран вперёд, с помощью [Ctrl]+[b] – на экран назад. Чтобы переместить кур-

сор в конец файла – клавиша [G]. Можно переместиться также на любую строку; например, напечатав команду `10G`, вы переместите курсор на десятую строку файла. Чтобы встать на начало (на первую строку), используйте `1G`. Можно сочетать команды перемещения с другими командами, такими, как удаление. Например, команда `d$` удалит от местоположения курсора до конца строки, `dG` удалит всё от курсора до конца файла и т. д.

Для выхода из **`vi`** без внесения изменений в ранее существовавший файл используйте команду `:q!`. Когда вы нажмёте [:], курсор переместится на последнюю строку экрана, поскольку вы перейдёте в режим последней строки. В режиме последней строки вы должны нажать [Enter] после набора команды. Команда `:wq` сохраняет файл, а затем выходит из **`vi`**. Команда `ZZ` (в режиме команд) эквивалентна `:wq`. Для записи файла без выхода из **`vi`** используется просто `:w`, при этом через пробел можно указать имя файла. Для перехода к редактированию другого файла используется команда `:e` (через пробел можно указать имя). Если вы используете `:e` без предварительного сохранения файла, то сначала вы получите соответствующее предупреждение. В этот момент вы можете использовать `:w`, чтобы сохранить исходный файл, а затем использовать `:e` или сразу нажать [!] – редактировать новый файл без сохранения изменений в первом файле. Если вы используете команду `:r`, можно включить содержимое другого файла в текущий файл. Например, команда `:r foo.txt` вставит содержимое файла foo.txt в данное место текста.

Вы можете также выполнять команды прямо из **`vi`**. Команда `:r!` работает, как `:r`, но вместо чтения файла она вставляет результат выполнения данной команды в место, где находится курсор. Например:

```
:r! who
```

Также вы можете выполнить команду, находясь в редакторе **`vi,`** и вернуться в редактор после её завершения. Например:

```
:! who
```

В данном примере будет выполнена команда **`who`**, её результат выдан на экран, а не вставлен в редактируемый файл.

Можно временно запустить ещё одну копию оболочки для выполнения других команд системы при помощи команды `:shell`. Редактор запустит командный интерпретатор, который позволит временно «отложить» **`vi`** и выполнить команды. После выхода из оболочки (используя команду **`exit`**) вы вернётесь в **`vi`**.

Для получения более подробной информации следует обратиться к справочному руководству **`man vi`**, а также специальному справочнику **`vimtutor`**.

1.6 Процессы, сигналы, задания

Единицей управления и потребления ресурсов в системе служит *процесс*. Вызывая команды, пользователь тем самым порождает процессы. При создании процесс получает уникальный ненулевой целочисленный идентификатор (*PID – process identifier*), по которому система отличает данный процесс от других. Права доступа процесса определяются атрибутами UID и GID пользователя, запустившего процесс. Если эти процессы порождают другие процессы, то те наследуют права родителей. При этом для каждого процесса запоминается PID его родителя (*PPID – parent process identifier*).

Информацию о запущенных процессах можно получить, используя команду **`ps`**. Ключ *-A* предназначен для вывода всех процессов (иначе выводятся только те процессы, которые связаны с текущим терминалом). Ключ *-l* предназначен для вывода полной информации о процессе: F (флаги), S (состояние), UID, PID, PPID, C (процент использования процессора), колонки PRI и NI определяют приоритет, SZ (размер в блоках), WCHAN (состояние), TTY (терминал), CMD (строка запуска процесса).

```
$ ps
  PID TTY          TIME CMD
 4156 pts/2    00:00:00 bash
 5023 pts/2    00:00:00 ps
$ ps -l
 F  S UID  PID PPID C PRI NI SZ WCHAN  TTY    TIME  CMD
000 S 501 4156 4154 0 70  0 720 wait4 pts/2 0:00:00 bash
000 R 501 5024 4156 0 74  0 762     - pts/2 0:00:00 ps
```

Чтобы отследить процессы, интенсивно использующие процессор, удобно воспользоваться утилитой **`top`**. Она выводит информацию, аналогичную той, что выводит команда **`ps`**, но при этом работает интерактивно, периодически обновляя выводимую информацию; при этом процессы сортируются по процентному использованию процессорного времени. Клавиши управления: [пробел] – обновить информацию, [q] – выход из программы.

Сигнал – логическое взаимодействие процесса с окружающей средой, позволяющее сообщить процессу о наступлении некоторого события в системе. Сигналы различаются своими номерами – целыми числами от 1 до 32 или 64 (зависит от системы). Сигнал генерируется (т. е. посылается процессу), когда происходит определённое событие: аппаратная ошибка, результат измерения времени (сработал таймер), прерывание с терминала (пользователь нажал [Ctrl]+[c]), выполнение команды **`kill`** и т. д. На каждый сигнал, определённый в системе,

процесс должен иметь реакцию – действие, которое он выполняет при получении сигнала. Например, сигнал SIGKILL всегда завершает процесс, которому он послан.

Некоторые сигналы:

- SIGHUP (*hang up – «разрыв линии»*) – Сигнал генерируется, в частности, когда терминал, подсоединённый через последовательный порт (модем), отсоединился; часто используется для сообщения процессу о том, что его конфигурационные файлы изменились.
- SIGINT (*interrupt*) – Прерывание с терминала (пользователь нажал [Ctrl]+[c]).
- SIGKILL (*kill*) – Безусловное завершение.
- SIGSEGV (*segment violation*) – Нарушение защиты памяти (программа обратилась к области памяти, которая ей не принадлежит).
- SIGALRM (*alarm*) – Таймер процесса (заказанное процессом время истекло).
- SIGTERM (*termination*) – Условное завершение (посылается командой **`kill`**, если не указан другой сигнал).
- SIGTSTP (*terminal stop*) – Остановка с терминала (пользователь нажал [Ctrl]+[z] или [Ctrl]+[y]).
- SIGCONT (*continue*) – Продолжение после остановки.
- SIGTTIN (*trying terminal input*) – Попытка чтения фоновым процессом с активного терминала.
- SIGTTOU (*trying terminal output*) – Попытка записи фоновым процессом на активный терминал.

Пользователь может послать любой сигнал процессу, UID которого совпадает с UID пользователя, при помощи команды **`kill`**. Обычно у пользователя возникает необходимость послать процессу сигнал SIGTERM или SIGKILL, когда какой-либо процесс перестаёт реагировать (в том числе на [Ctrl]+[c]). Формат команды:

`kill` [-s *сигнал*] *PID*

Примеры:

- **`kill -l`** – получить список используемых сигналов;
- **`kill 1234`** – послать сигнал SIGTERM процессу 1234;
- **`kill -s SIGHUP 1234`**;
- **`kill -HUP 1234`**;
- **`kill -1 1234`** (сигналу SIGHUP соответствует номер 1).

Последние три команды посылают сигнал SIGHUP процессу с PID=1234.

Аналогом команды **`kill`** является команда **`killall`**. Вместо PID процесса её аргументом является имя процесса. Пользуйтесь ей осторожно, т. к. указанный сигнал посылается всем процессам с указанным именем:

```
killall httpd
killall –HUP httpd
```

Большинство систем UNIX обеспечивают механизм управления заданиями. При интерактивной работе интерпретатора каждая простая команда или конвейер (см. раздел 3) называются *заданием*. Задания, как правило, имеют отдельную нумерацию. Посмотреть список заданий можно при помощи встроенной команды **`jobs`**. Пока оболочка выполняет команды синхронно (одну за другой), механизм управления заданиями не активен и список заданий пуст. Этот механизм активируется, когда текущее задание приостанавливается или запускается в фоновом режиме. Приостановка задания – [Ctrl]+[z]:

```
$ jobs
$ vi
^Z
[1]+  Stopped                 vi
$ jobs
[1]+  Stopped                 vi
```

Для ссылок на то или иное задание можно использовать не только PID процессов, но и номера заданий в виде **`%num`**. Кроме того, на текущее задание можно сослаться комбинацией **`%%`** или **`%+`**, а на предыдущее **`%-`**. Эти обозначения можно также использовать в команде **`kill`**.

Чтобы возобновить выполнение приостановленного задания или перевести фоновое задание в приоритетный (foreground) режим, используется команда **`fg`**. Если аргумент не указан, команда оперирует текущим заданием. Примеры:

- **`fg`** или **`fg %%`** – возобновить выполнение текущего задания;
- **`fg %-`** – возобновить выполнение предыдущего задания;
- **`fg %5`** – возобновить выполнение задания № 5.

Если процессу не требуется активное взаимодействие с терминалом (вывод данных непосредственно на экран, ввод данных непосредственно с клавиатуры), то процесс может быть переведён в фоновый (background) режим. В фоновом режиме процесс выполняется параллельно с другими процессами до тех пор, пока ему не потребуется выполнить операцию ввода-вывода с терминалом, при этом он приостанавливается до тех пор, пока снова не будет переведён в приоритетный режим.

Чтобы запустить задание в фоновом режиме, в его командной строке следует использовать ограничитель **`&`**. Приостановленное задание можно перевести в фоновый режим при помощи команды **`bg`**. Пример:

```
$ while true; do date >>1.txt; sleep 5; done &
[2] 21491
$ jobs
[1]+ Stopped  vi
[2]- Running while true; do date >>1.txt; sleep 5;done &
$ kill -SIGTSTP %2
$ jobs
[1]- Stopped  vi
[2]+ Stopped while true; do date >>1.txt; sleep 5; done
$ bg %-
[1]- vi &
$ bg %2
[2]- while true; do date >>1.txt; sleep 5; done &
$ jobs
[1]+ Stopped  vi
[2]- Running while true; do date >>1.txt; sleep 5;done &
$ kill %2
[2]+ Terminated while true; do date >>1.txt;sleep 5;done
$ jobs
[1]+ Stopped  vi
```

В этом примере в фоновом режиме запускается команда, которая каждые 5 секунд сохраняет в файле 1.txt текущее время. Обратите внимание на амперсанд в конце командной строки. Система присвоила номер 2 этому заданию. Затем при помощи команды **kill** ему посылается сигнал SIGTSTP, который эквивалентен нажатию [Ctrl]+[z] для активного задания, однако, это задание не является приоритетным, т. е. не взаимодействует с терминалом, поэтому нажатие [Ctrl]+[z] никак не повлияет. В результате задание приостанавливается. Следующая команда – попытка перевести в фоновый режим остановленный ранее редактор **vi**, а затем задание № 2. Очевидно, что работа задания № 2 возобновилась в фоновом режиме, а редактор так и остался приостановленным, т. к. при возобновлении он сразу же пытается читать с терминала. Для завершения задания № 2 использовалась команда **kill**.

Следует отметить, что после выполнения команд **bg**, **fg** и **kill** оболочка не всегда сразу отображает изменение состояния задания. Иногда такое сообщение появляется только после выдачи очередного командного приглашения (нужно нажать [Enter]).

О других особенностях механизма управления заданиями можно узнать из **man bash**, раздел JOB CONTROL.

1.7 Перенаправление ввода-вывода

Каждый запущенный процесс имеет три открытых файла с дескрипторами 0, 1 и 2. Файл с дескриптором 0 соответствует стандартному вводу. Файл с дескриптором 1 соответствует стандартному выводу. Файл с дескриптором 2 соответствует стандартному выводу ошибок. По умолчанию все три файла связаны с терминалом, т. е. чтение из файла стандартного ввода приводит к вводу данных с клавиатуры, а запись в файл стандартного вывода или стандартного вывода ошибок приводит к отображению текста на экране. Обычно в файл стандартного вывода отправляется полезный результат выполнения программы, а в файл стандартного вывода ошибок – диагностические сообщения. Командный интерпретатор позволяет переназначить эти файлы, т. е. вместо терминала указанные стандартные файловые дескрипторы будут связаны с определённым файлом.

Перенаправление вывода (*n* – необязательный аргумент, номер дескриптора, по умолчанию – 1):

команда [*n*]**>***файл*

Если файл не существует, он создаётся. Если файл существует, он усекается до нулевого размера. Для дозаписи в указанный файл используется следующий синтаксис:

команда [*n*]**>>***файл*

Перенаправление ввода (*n* – необязательный аргумент, номер дескриптора, по умолчанию – 0):

команда [*n*]**<***файл*

```
$ echo :1,10j >2.txt
$ echo :wq >>2.txt
$ vi –e 1.txt <2.txt
```

В этих примерах первая команда выводит текст «:1,10j» в файл 2.txt. Вторая команда дописывает текст «:wq» в конец файла 2.txt. Третья команда запускает **vi** в пакетном режиме (ключ *-e*) – на стандартный ввод подаются команды из файла 2.txt. В результате в файле 1.txt будут объединены строки 1–10.

Если сообщения программы об ошибках неважны, стандартный вывод ошибок (дескриптор 2) можно перенаправить в специальное устройство /dev/null, как показано в нижеследующем примере:

```
$ date –s 12:30
date: cannot set date: Operation not permitted
Tue Jan 30 12:30:00 MSK 2018
$ date –s 12:30 2>/dev/null
```

```
Tue Jan 30 12:30:00 MSK 2018
```

Особый тип перенаправления стандартного ввода – встраиваемые в командную строку документы (here documents). Оболочка передаёт на стандартный ввод команды текст встроенного документа, размещаемого в том же источнике, что и команда. Встроенный документ должен оканчиваться словом-ограничителем, совпадающим с указанным после знака перенаправления.

```
команда <<ограничитель
встроенный документ
ограничитель
```

В следующем примере **vi** запускается в пакетном режиме. На стандартный ввод **vi** подаётся текст из командной строки: команды «:2,5j» (объединение строк 2–5) и «:wq» (выход с сохранением). В качестве ограничителя использовано слово «BNDR».

```
$ vi –e 1.txt <<BNDR
> :2,5j
> :wq
> BNDR
```

Специальный случай перенаправления – конвейер. *Конвейер* – это последовательность команд, соединённых управляющей операцией “|”. При этом стандартный вывод каждой команды конвейера, кроме последней, направляется в стандартный ввод следующей команды. Команды конвейера исполняются не последовательно, а параллельно. Соединение соседних команд в конвейере основано на использовании программных каналов (*pipes*). Оболочка ожидает завершения всех процессов, составляющих конвейер, если только он не выполняется в фоновом режиме. Пример:

```
who | tr a-z A-Z | vi –
```

В этом конвейере использовано три команды. Стандартный вывод команды **who** подаётся на стандартный ввод команды **tr**. Команда **tr** предназначена для замены указанного набора символов на другой набор символов, т. е. первый символ первого набора всегда заменяется на первый символ второго набора и т. д. Наборы символов задаются как аргументы командной строки, причём можно использовать минус (–) как знак диапазона. Обрабатываемый текст берётся со стандартного ввода, а результат выводится на стандартный вывод. В нашем случае первый набор задан как «a–z», что соответствует всем строчным латинским буквам. Второй набор соответствует всем заглавным латинским буквам. Результат выполнения этой команды – текст, в котором все строчные латинские буквы заменены на заглавные. Этот результат передаётся на стандартный ввод редактора **vi**. Редактор запущен с ключом “–”, означающем, что редактироваться будет текст, взятый со стандартного ввода программы.

Контрольные вопросы и задания

1. Выясните необходимую информацию для подключения и осуществите регистрацию в системе. Воспользуйтесь справочной системой UNIX и изучите синтаксис команд, упомянутых в этом разделе.
2. Используя описанные команды, получите информацию о работающих пользователях, о системе, данные о конкретном пользователе, о своей группе.
3. Создайте несколько текстовых файлов, содержащих информацию о вас и месте вашей работе (учёбы). Отредактируйте файлы, используя приведённые команды.
4. Запустите редактор **vi**, приостановите его работу. Запустите в фоновом режиме предложенную в примере команду периодического сохранения времени в файле 1.txt. Возобновите работу **vi** и загрузите в него файл 1.txt. Снова приостановите **vi**. Приостановите работу вашей команды, как показано в примере. С помощью **vi** убедитесь, что новые данные в файл не записываются, пока команда приостановлена. Завершите работу команды, вернитесь в **vi** и штатным образом выйдите из него.
5. Составьте конвейер из команд **ps** и **tr** так, чтобы информация выводилась только заглавными буквами. Вывод перенаправьте в файл ps.txt.