1 Памятка для работы в системе UNIX

1.1 Вход в систему

Для входа в систему надо набрать свой `login` (регистрационное имя) и `password` (пароль). Регистрационное имя и пароль вам выдаёт системный администратор. После первого входа в систему пароль рекомендуется сменить на свой собственный командой `passwd` (см. ниже). Хороший пароль должен состоять из латинских больших и малых букв, цифр и знаков и не должен содержать слов русского или английского языков.

Если пароль набран правильно, вы начинаете работать с командным процессором csh. Рекомендуется запускать более «продвинутый» командный процессор bash командой `bash`. Если единственное, что вы хотите сделать — это запустить графическую оболочку X Window или KDE, bash можно не запускать.

Сразу после входа в систему текущим каталогом становится ваш «домашний каталог», в котором вы можете создавать ваши файлы. Максимальный суммарный размер всех файлов, которые вы можете создать ограничен. Величину этого ограничения («размер квоты») можно узнать у системного администратора.

В терминальном классе установлена графическая оболочка KDE, достаточно удобная для использования неподготовленными пользователями. В её составе есть текстовые редакторы (Kate и др.), интегрированная среда для разработки программ (KDevelop), браузер (Konqueror), офисный пакет (KOffice). Кроме того, система помощи позволяет просматривать документацию в форматах man и info. Для запуска этой графической оболочки после входа в систему необходимо набрать команду `kde`.

Одной из основных программ, используемых при работе в графической оконной среде тем не менее всё равно остаётся эмулятор терминала (KTerm, xterm). При запуске эмулятора запустится командный процессор tcsh. Рекомендуется использовать командный процессор bash как более удобный в использовании и программировании.

1.2 Редактирование вводимых команд

Если используется командный процессор bash, при вводе команд доступны команды редактирования строки. Клавиши «курсор вправо» и «курсор влево» перемещают курсор вправо и влево по строке. Клавиша «курсор вверх» вызывает на редактирование предыдущую команду из списка уже введённых команд, клавиша «курсор вниз» — следующую команду. Комбинация «Ctrl-U» очищает текущую вводимую строку, комбинация «Ctrl-E» перемещает курсор в конец вводимой строки, комбинация «Ctrl-A» перемещает курсор в начало вводимой строки. Кроме того, при вводе команд можно использовать клавишу «Tab». При нажатии этой клавиши командный процессор пытается завершить команду, начало которой набрано в командной строке. Если такое завершение единственно, оно будет немедленно подставлено, если таких завершений несколько, командный процессор издаст звуковой сигнал и на повторное нажатие клавиши «Tab» распечатает список подходящих команд. При наборе аргументов команд по клавише «Tab» командный процессор пытается дополнить аргумент до имени существующего файла.

Аргументы команды отделяются от имени команды и от друг друга одним или несколькими символами пробела.

Для запуска процессов, которые не производят ввод/вывод на терминал, например, графических приложений или фоновых процессов, используется символ `&` («амперсенд») в конце команды. Например, команда `emacs&` запустит редактор emacs. Если не указывать сим-

вол «амперсенд» в команде, ввод и вывод на терминале будет заблокирован до окончания работы процесса.

Для прерывания текущего процесса используется комбинация клавиш «Ctrl-C». Эта комбинация клавиш не воздействует на фоновые процессы. Для прерывания фоновых процессов используется команда `kill`.

Когда ваша программа считывает данные из стандартного потока ввода, признак конца ввода подаётся нажатием «Ctrl-D». Если последняя введённая строка ещё не закончена, «Ctrl-D» нужно нажать дважды.

Если нажать «Ctrl-D» при работе с интерпретатором команд, он закончит свою работу. Если это был последний интерпретатор команд, Вы закончите работу с системой, и снова появится приглашение к вводу регистрационного имени и пароля.

1.3 Смена пароля

Для смены пароля используется команда `passwd`. Эта команда не имеет аргументов. Сначала она запрашивает текущий пароль, соответствующий вашему регистрационному имени, затем новый пароль и после этого новый пароль второй раз для подтверждения правильности. Никакой из вводимых паролей не показывается на экране.

1.4 Помощь

Для получения справочной информации по командам операционной системы, функциям библиотеки Си и пр. используется команда man.

```
man ls          # информация о команде ls
man printf      # информация о команде printf
man 3 printf    # информация о функции printf библиотеки Си
man gets        # информация о функции gets библиотеки Си
```

Графический интерфейс к документации обеспечивается программой `xman`. Кроме того, некоторая информация (например, описание компилятора GCC) доступна в гипертекстовом формате info. Для просмотра этой документации нужно набрать команду `info`.

1.5 Работа с процессами

Команда `ps` показывает список всех запущенных вами процессов, включая фоновые. В колонке `PID` указан номер процесса, который идентифицирует работающий процесс и может использоваться для его уничтожения.

Команда `kill <pid>` завершает процесс с данным идентификатором. Если процесс совсем «завис», простой команды `kill` может оказаться недостаточно. Тогда нужно использовать команду `kill -9 <pid>`. Это крайний случай, поскольку в этом случае операционная система не даёт программе шанса завершиться корректно.

1.6 Работа с файлами в файловой системе

Файловая система UNIX организована иерархически. В отличие от MS-DOS и Windows отсутствует понятие «диска», разбиение файлов по дискам прозрачно для пользователя. Поэтому файловая система имеет единственный корень: `/`. Каталоги в пути разделяются симво-

лом /, а не \, как в MS-DOS и Windows. Пути к файлам могут быть абсолютными, начинающимися от корня файловой системы /, или относительными, начинающимися от текущего каталога. В любом каталоге всегда существуют два специальных имени: . («точка») означает этот самый каталог и .. («две точки») означает родительский к данному каталог. Примеры:

`/usr/include/stdio.h`	абсолютный путь
`/etc/passwd`	другой абсолютный путь
`myprog/myfile.c`	относительный путь
`hello.c`	файл в текущем каталоге
`./hello.c`	то же самое

Заглавные и строчные буквы в имени файлов различаются. Поэтому имена файлов `x.C` и `x.c` могут означать разные файлы.

Если имя файла начинается с символа «точка», такой файл считается «скрытым» и не показывается при обычных командах просмотра каталогов.

Для работы с файлами используются следующие команды:

`ls` показывает список всех файлов в текущем каталоге, `ls -l` показывает список всех файлов с их атрибутами. `ls -l <file1> ... <filen>` показывает информацию только об указанных файлах. `ls -a` показывает информацию обо всех файлах, включая скрытые.

Команда `pwd` печатает полный путь к текущему каталогу.

Команда `cd <dir>` меняет текущий каталог. `cd` (без аргументов) устанавливает домашний каталог текущим.

Команда `mkdir <path1> ... <pathn>` создаёт новый каталоги с данным путями, причём все компоненты пути кроме последнего должны существовать и быть каталогами, а последняя компонента пути не должна существовать. Команда `rmdir <path1> ... <pathn>` удаляет каталоги с данными путями, которые должны быть пустыми, то есть не содержать имён кроме . и ..

Команда `rm <file1> ... <filen>` удаляет простые файлы, не являющиеся каталогами. Нужно быть особенно аккуратным, поскольку восстановление уже удалённого файла невозможно. Команда `rm -rf <file1> ... <filen>` удаляет перечисленные файлы и каталоги со всем их содержимым.

Команда `mcopy` позволяет копировать файлы с гибкого диска и обратно. Например,

```
mcopy file.c a:     # копирование на гибкий диск
mcopy a:prog.c .    # копирование с гибкого диска
```

Для задания нескольких файлов одновременно могут использоваться метасимволы командного процессора. Символ * обозначает произвольное количество символов, а символ ? — один произвольный символ. Обратите внимание, что в отличие от MS-DOS имя файла не обязано состоять из основной части и расширения. Поэтому конструкция `*.*` в UNIX перечисляет все файлы, имена которых содержат «точку». Для того, чтобы перечислить все файлы, кроме скрытых, используется конструкция *. Например,

```
rm *          # удаляет все файлы в текущем каталоге, кроме
              # начинающихся с точки (скрытых файлов)
rm *.o        # удаляет все объектные файлы
```

Существует командный процессор Midnight Commander, своим интерфейсом копирующий Norton Commander. Он запускается командой `mc`.

1.7 Просмотр файлов

Команда `cat <file1> ... <filen>` распечатывает содержимое файлов. Чтобы приостановить вывод на экран нужно нажать «Ctrl-S», чтобы возобновить вывод — «Ctrl-Q».

Команды `more` и `less` выводят содержимое файлов, заданных их аргументами поэкранно. Клавиша «Пробел» пролистывает один экран вперёд по файлу, клавиша «Enter» пролистывает одну строку вперёд. Программа `less` позволяет листать файлы и назад (клавиша «Стрелка вверх»).

1.8 Редактирование файлов

Мощный, но недружественный редактор — `vi` и его расширение `vim`. При запуске файла может указываться имя файла для редактирования. Если такой файл не существует, он создаётся. Редактор имеет три режима работы: командный, когда символы, вводимые с клавиатуры обозначают команды и исполняются немедленно, ввода текста, когда символы, вводимые с клавиатуры формируют вводимый текст, и режим `ex`, когда редактор принимает сложные команды, завершающиеся символом «Enter». Далее приведён совсем минимальный набор команд редактора, достаточный, чтобы редактировать файлы на базовом уровне.

При запуске редактор начинает работать в командном режиме. Команда `i` — вставка текста в текущую позицию, `a` — добавление текста после текущей позиции, `x` — удаление символа в текущей позиции, `j` — склейка двух строк в одну. В командном режиме работают команды перемещения курсора. «Enter» означает переход в начало следующей строки.

По командам `i`, `a` редактор переходит в режим ввода текста. Выход из этого режима осуществляется по клавише «Esc».

Для перехода в режим `ex` нужно в командном режиме набрать `:` («двоеточие»). Приглашением ко вводу служит символ двоеточия. Простейшие команды: `w` — записать файл, `w <name>` записать текущий буфер в файл с данным именем, `q` — выход из редактора, `q!` — выход без записи файла, `<номер строки>` — переход на строку с данным номером.

Если вы запутались, чтобы выйти из редактора нужно несколько раз нажать клавишу «Esc», затем клавиши `:` и `q`.

Очень мощный и достаточно дружественный редактор — emacs. Быстрый выход из него осуществляется по клавишам «Ctrl-X» «Ctrl-C».

1.9 Компиляция программ

Ваши файлы с текстом программы на Си должны называться `<имя>.c` (c — маленькая!). Если есть заголовочные файлы, они имеют суффикс `.h`. Исполняемые файлы не имеют никакого суффикса.

Каждое Ваше задание, которое Вы будете сдавать, должно находится в отдельном каталоге.

После того, как файл программы создан или отредактирован, программа должна быть скомпилирована. Для этого используется команда
`gcc -Wall -g <имя файла> -o <имя исп. файла>`, например

```
gcc -Wall -g prog.c -o prog -lm
```

Если ваша программа использует математические функции, например `sqrt` или `fabs`, при компиляции необходим дополнительный параметр `-lm`. После этого для запуска файла на

выполнение можно просто набрать имя исполняемого файла. Запуск Вашей программы никак не отличается от запуска системных программ (например, ls). Возможно специфицировать аргументы, если ваша программа их обрабатывает.

Обратите внимание, что при запуске программы на выполнение без указания пути к ней её исполняемый файл ищется **только** в каталогах, указанных в переменной окружения `PATH`. Если текущий каталог (обозначается `.`) не содержится в пути поиска, при запуске программы, находящейся в текущем каталоге, требуется явное указание пути, например `./myprog`.

2 Стиль кодирования программ

Стиль кодирования — это набор правил оформления программы на некотором языке. Стили кодирования существуют для всех языков. Для языка Си существует несколько стилей кодирования. Рекомендуется нижеприведённый стиль, но вы можете использовать какой-либо другой, кроме изобретённого вами самими.

Стиль кодирования не влияет на работоспособность программ, но влияет на простоту их прочтения и понимания. Несоблюдение стиля кодирования является поводом для отказа в приёме задачи про практикуму.

Редактор emacs поддерживает форматирование файлов при редактировании Си-программ. Чтобы разместить строку с правильным отступом, нажмите на клавишу «Tab». Если отступ не совпал с ожидаемым, внимательно посмотрите на программу, возможно она содержит синтаксическую ошибку.

Существует специальная программа (indent), которая переформатирует программу на Си в соответствие с указанным стилем кодирования. Для получения более подробной информации, используйте команду `man indent`.

Обратите внимание, что все примеры программ в раздаваемых Вам материалах, отформатированы в соответствие с этими правилами.

Отступы в программе следует размещать следующим образом: все директивы препроцессора начинаются с начала строки. Определение функции оформляется следующим образом:

```
<класс памяти> <возвр. тип> <имя функции>(<аргументы>)
{
}
```

фигурные скобки размещаются на отдельных строках. Пример:

```
int main(int argc, char **argv)
{
}
```

Если весь заголовок функции не умещается на одной строке, продолжение заголовка выравнивается по открывающей скобке. Например,

```
unsigned short int foo(int a,
                       double b)
{
}
```

Инициализаторы сложных объектов должны располагаться на отдельной строке. Например

```
#define MAX_CIRCLES 10
struct circle
{
  double x, y, r;
  char   name[16];
};
int ncircles = 2;
struct circle circles[MAX_CIRCLES] =
{
  { 1.0, 2.0, 0.4, "null" },
  { 3.0, 3.0, 0.1, "default" },
};
```

Отступ во вложенных блоках 2, 4, либо 8 символов, но один во всей программе. Фигурные скобки в операторах размещаются как показано на примере:

```
if (x > 0) {
} else {
}

if (y > 0) {
}

while (1) {
}

for (;;) {
}

do {
} while (x < 5);
```

Ключевые слова отделяются от последующих символов хотя бы одним пробелом. Знаки бинарных операций отделяются от своих аргументов пробелами слева и справа. Открывающие скобки не отделяются пробелами справа, а закрывающие — слева.

Рекомендуется размещать только один оператор на строке.

Размер функции не должен превышать размера одного экрана (24 строки). Если функция содержит оператор `switch`, её размер может превышать размер одного экрана. В противном случае, большие функции — почти всегда знак плохой организации программы. Нужно подумать о том, как разбить большую функцию на несколько функций меньшего размера.

Программа не должна содержать «магических констант» в теле операторов и функций. Все такие константы должны быть вынесены в отдельные определения констант. Исключением являются константы -1, 0, 1 и 2, если это номер стандартного потока ошибок.

Рекомендуется писать комментарии, описывающие назначение и параметры каждой функции (кроме main). Кроме того, рекомендуется писать поясняющий комментарий к каждому нетривиальному фрагменту программы.

Рекомендуется вставлять пустые строки между определениями переменных и началом кода функции, между функциями, между фрагментами кода, выражающими законченную мысль.

Запрещается использование макроопределений, нарушающих синтаксис языка. Например

```
#define BEGIN {
#define END   }
```

Не рекомендуется использование определений функций в старом стиле. Необходимость такого определения вы должны обосновать. Не рекомендуется использовать определения функций и переменных с типом по умолчанию.

Запрещается использование функций без прототипов, кроме системных функций. Запрещается явное определение прототипов системных функций.