

Краткая шпаргалка по работе в дисплейных классах

Вход в систему

В дисплейных классах установлены рабочие станции, работающие под управлением OS Linux. Таким образом, вы можете повысить свою грамотность чтением соответствующей справочной литературы. Здесь же излагаются самые начальная и простая информация о том как можно начать работу в классах.

Каждый пользователь должен быть зарегистрирован с системе. Это означает, что ему выдается login (входное имя) и password (пароль). В какой-то момент системный администратор регистрирует всех студентов и назначает им login и password. Кроме этого в системе есть универсальный пользователь student с паролем mexmat. Этот вход можно использовать в некоторых исключительных ситуациях для временной работы, но при этом есть существенное ограничение — все ваши рабочие файлы сохраняются только на время сеанса, после выключения или перезагрузки компьютера они пропадают.

Сохраняйте свой пароль в тайне во избежание неприятностей с вашим аккаунтом, файлами и доступом в систему.

После включения компьютера в течении некоторого времени происходит загрузка операционной системы. Если она прошла успешно, то на экране появляется картинка с полями ввода входного имени и пароля. Их надо ввести. Если система вас опознала, то откроется основное рабочее окно графической оболочки.

По окончании работы **НЕ ЗАБУДЬТЕ ВЫЙТИ ИЗ СИСТЕМЫ**. Помимо того, что кто-то может деструктивно “поработать” с вашим аккаунтом в ваше отсутствие, при вечернем обесточивании компьютера могут пропасть или исказиться файлы, которые вы по каким-то причинам не закрыли или не сохранили после редактирования.

Для выхода из системы нужно использовать системное меню, которое расположено в левой нижней части основного рабочего окна графической оболочки.

Простейшая схема работы в консоли

Если нажать правую кнопку мыши на свободном месте основного рабочего окна, то откроется меню, в кором будет пункт **konsole**. Этот пункт запускает на выполнение рабочую консоль (терминал) с интерпретатором командной строки. В консоли можно вводить строки-команды и получать ответ системы на данные команды.

Для подробного знакомства с возможностями командной строки и скриптами можно посмотреть справочную информацию по ключевому слову **bash**.

Здесь мы упомянем только несколько простейших команд.

Получить справочную информацию о команде, функции и т.п.

`man имя команды, функции и т.д.`

(здесь `man` от `MANnual pages`) — справочная база Linux/Unix. Команда `man` выдает справочную информацию из специальной базы, в которой описываются команды `bash`, библиотечные функции и т.п. Обычно в этой базе записана исчерпывающая информация по конкретному ключевому слову, но также по некоторым командам/функциям информация там может и отсутствовать (все зависит от конфигурации и наполненности этой базы). В некоторых случаях одно и то же ключевое слово может относиться к разным записям в базе (например, есть и команда, и функция с одним и тем же именем). В этом случае нужно указать раздел базы, из которого надо брать соответствующее описание. Раздел указывается как целое число в первом аргументе команды `man`, т.е., например,

`man 2 time`

Номер раздела можно просто подобрать как 1, 2, 3 и т.д. и посмотреть, что выдает man. Данная команда очень удобна, если вы забыли как конкретно работает та или иная команда или функция, какие у нее параметры и т.п. Однако некоторые описания могут оказаться очень объемными (наберите, например, man gcc и посмотрите, что вам выдадут).

Выйди из чтения описания можно с помощью команды **q** (от слова quit).

Сделать каталог *имя-каталога* текущим

```
cd имя-каталога
```

(здесь cd от Change Directory). Команда cd без параметров переместит вас в ваш стартовый (home) рабочий каталог (полезно, если вы вдруг запутались в файловой системе и не знаете где находитесь). Имейте в виду, что некоторые каталоги вам могут быть недоступны (системные, от других пользователей и т.п.). В таком случае вас либо просто туда не пустят, либо формально пустят, но вы там ничего не сможете увидеть. e

Просмотреть список файлов

в текущем каталоге:

```
ls
```

в каталоге с именем *dir*:

```
ls dir
```

в каталоге с именем *dir* с подробной информацией о файлах:

```
ls dir -la
```

Создать каталог *имя*

```
mkdir имя
```

Копировать файл *file1* в *file2*

```
cp file1 file2
```

здесь и далее можно указывать не просто имена файлов, а также относительные или абсолютные пути к файлам.

Переименовать (переместить) файл *file1* в *file2*

```
mv file1 file2
```

Удалить файл *file*

```
rm file
```

ВНИМАНИЕ! Восстановить удаленные файлы нельзя!

Удалить каталог *dir*

пустой:

```
rmdir dir
```

непустой со всем его содержимым:

```
rm -rf dir
```

ВНИМАНИЕ! Восстановить удаленные файлы нельзя!

Просмотреть файл *file*

```
less file
```

При просмотре можно пользоваться стрелками, клавишами Page-UP и Page-Down. Выход из программы: нажать q. Более подробную информацию о возможностях утилиты less можно получить, нажав h после ее запуска.

запустить файл *file* на выполнение

Запустить на выполнение можно так называемые “исполняемые файлы”, т.е. готовые программы, полученные, например, в результате компиляции и сборки вашего программного C/C++ кода. Для этого достаточно набрать имя исполняемого файла в командной строке и нажать enter. Иногда система требует указать не только имя, но и каталог, в котором расположен это файл. В таком случае надо перед именем файла набрать имя каталога. В большинстве случаев вы будете запускать файл либо из текущего каталога, либо где-то относительно своего домашнего (home) каталога. Например, если ваш исполняемый файл называется a.out и лежит в каталоге task1, находящемся в вашем home каталоге, то запустить этот файл на выполнение можно так. Если текущий каталог есть task1:

```
./a.out
```

Если текущий каталог любой другой:

```
~/task1/a.out
```

Т.е. “.” есть обозначение текущего каталога, а ~ — ваш домашний (home) каталог.

“Убить” запущенный процесс

Если ваша программа после запуска “повисла”, т.е. ничего не выдает, ни на что не реагирует, то вам надо как-то ее завершить. Если вы ее запускали из консоли, то можно попробовать нажать ctrl-c или ctrl-z. Если это не помогает, то надо “убить” этот процесс. Процесс (работающая программа) идентифицируется своим названием (именем исполняемого файла) и текущим номером в системе. Так как повисшая программа обычно блокирует консоль (терминал), из которого она запущена, то нужно открыть вторую консоль на основном рабочем окне графической оболочки и в этой консоли выполнить команду

```
ps x
```

для выдачи списка всех запущенных процессов, или для сокращения выдачи

```
ps x | grep name
```

где теперь вы указываете имя вашего исполняемого файла. В этом списке нужно найти строчку с именем вашей программы и в ней найти номер N процесса (первая колонка — pid — process identifier). Далее можно собственно ликвидировать процесс с найденным номером

```
kill -9 N
```

Программ с одним именем может быть запущено несколько, различаются они по имени терминала, с которым связаны, это вторая колонка в списке процессов.

Редактирование файла

Для редактирования файла может применяться множество редакторов, каждый из которых имеет свой собственный набор операций и свою идеологию. Выбирайте тот, который вам проще или больше нравится. Информацию по конкретным редакторам здесь приводить не будем, а сразу отсылаем к соответствующим руководствам или справочникам.

Единственное замечание. Если вы вдруг случайно открыли файл каким-то редактором, который вам не знаком, то главное, что надо уметь делать — это выйти из него с сохранением вашего имеющегося текста.

Большинство редакторов предоставляет справочную информацию либо по кнопке F1, либо по другим комбинациям клавиш, например,

ctrl-k потом h — редактор joe

ctrl-? — редактор mim

:help — редактор vim

и т.п.

Кстати, выход с сохранением из редактора vim выполняется командой **:wq**

Вход из редактора mim выполняется последовательным нажатием кнопок ESC и “стрелка вверх”.

Файловые менеджеры

Для удобства и наглядности файловых операций (в противовес командной строке в консоли) можно использовать файловые менеджеры — специальные программы, позволяющие видеть списки файлов и удобно с ними работать. В графической оболочке Linux есть различные файловые менеджеры (похожие на проводник windows), и также есть более удобный (на наш взгляд) файловый менеджер с двумя рабочими окнами (аналогичный Norton Commsnder, Total Commander, Far для windows). Здесь он называется Midnight Commander и запускается из консоли командой

```
mc
```

После запуска MC сама консоль также останется активной и доступной, и переключаться между окном консоли и окнами MC можно по клавише ctrl-o.

Работа с MC достаточно проста и наглядна, основные файловые операции “посажены” на функциональные клавиши F1 — F10 и легко осваиваются.

Вкратце:

стрелки вверх/вниз — перемещение по списку файлов;

TAB — переход в соседнюю панель;

ENTER — выполнение действия, ассоциированного с файлом, в частности

для файла .C или .CPP — запустить редактор vim

для файла .. — переход в родительский каталог

для каталога — переход в этот каталог

для исполняемого файла — запуск этого файла

Клавиши F1–F10 в разных ситуациях могут означать разные действия, при этом в нижней строке есть подсказка. Например, при режиме показа файловых панелей

F1 — вызов справки

F3 — просмотр текущего файла

F4 — редактирование текущего файла встроенным редактором

shift-F4 — редактировать новый файл, при выходе потребуется дать ему имя

F5 — копирование текущего файла

F6 — перемещение (переименование) текущего файла

F7 — создать новый каталог

F8 — удалить текущий файл

F9 — перейти в командное меню MC

F10 — выйти из MC (завершить его работу)

При копировании и перемещении файла по умолчанию это происходит из текущего окна файлового менеджера в другое окно с тем же именем файла (точнее, в каталоги, которые показаны в этих окнах). Если вы хотите сменить имя файла при такой операции, то его надо ввести в открывшееся при копировании диалоговое окно.

При просмотре или редактировании файлов смысл этих клавиш изменяется (см. подсказки в нижней строке).

Кроме этого в нижней строке MS можно также набирать консольные команды, правда, сообщения об их работе будут выводиться на консоль, перекрытую файловыми панелями, посмотреть можно переключая панели по `ctrl-o`.

Основные команды компиляции

Подготовка программы из кода на C/C++ в исполняемый файл проходит два этапа — компиляция файлов (получаются так называемые объектные файлы) и сборка или линковка (от английского `link`) — получается исполняемый файл, который уже можно запускать на выполнение.

В разных ситуациях эти действия могут объединяться или разделяться, автоматизироваться и т.п.

Рассмотрим, например, вариант, когда наша программа записана в двух файлах `fmain.cpp` и `fcalc.cpp`.

Можно выполнить вышеописанные этапы и запустить полученную программу, последовательно выполнив следующие консольные команды

```
g++ -c fmain.cpp
g++ -c fcalc.cpp
g++ fmain.o fcalc.o
./a.out
```

Первые два команды — это компиляция соответствующих файлов, в результате создаются объектные файлы `fmain.o` `fcalc.o`. Третья команда — сборка исполняемого файла вместе с системными библиотеками по умолчанию. Результат — исполняемый файл — по умолчанию будет называться `a.out`. Последняя команда — запуск на выполнение файла `a.out`.

В простых случаях можно объединить первые три команды в одну строку

```
g++ fmain.cpp fcalc.cpp
```

Здесь сначала будет выполнена компиляция всех перечисленных файлов, а потом их сборка. Если программа состоит из нескольких файлов, то их все надо упоминать либо в отдельной команде по первому образцу, либо перечисляя их все в команде второго образца.

Для файлов, написанных на языке C вместо `g++` надо использовать команду `gcc`, т.е., например

```
gcc fmain.c fcalc.c
```

По сути, `g++` и `gcc` — это вызов одного и того же компилятора, но для разных языковых конструкций и разных системных библиотек.

Команды компиляции могут снабжаться дополнительными параметрами, которые обычно выглядят как некоторое ключевое слово (или буква) со знаком “-” перед ним и последующими параметрами. Полный список таких инструкций можно получить по справке

```
man gcc
```

Но это огромный текст, который с непривычки читать очень сложно.

Вот некоторые из таких дополнительных параметров:

- g — включить отладочную информацию
- o *name* — дать указанное имя исполняемому (выходному) файлу (вместо `a.out`)
- I *dir* — искать включаемые файлы также и в директории `dir`

`-std=c++11` — компилировать в соответствии с указанным стандартом (с++98, с++03, с++14, с++17, с++20 и т.д.)

`-lname` — подключить указанную библиотеку

`-lm` — подключить математическую библиотеку (это для gcc), g++ подключает ее автоматически

Место записи этих параметров в строке обычно может быть где угодно

Например, вы хотите компилировать файл `fff.cpp` в стандарте с++11 для последующей отладки в отладчике `gdb` и также назвать исполняемый файл именем `myprog`

```
g++ fff.cpp -std=c++11 -o myprog -g
```

Вот еще пример для языка C

```
gcc myfile.c -lm -g -o prog.exe
```

На самом деле, писать `.exe` для имен исполняемых файлов здесь совершенно излишне, но вам никто не запрещает это делать, если вы привыкли работать в windows и хотите это продемонстрировать и при работе в Linux.

Ошибки компиляции

Как правило, первая компиляция вашего кода завершится с ошибками. Сообщения компилятора при этом содержат имя файла, номер строки и позиции, а также краткую цитату вашего кода в том месте, где компилятор решил, что ваш код неверен.

Обычно эта информация позволяет сразу понять и исправить обнаруженную ошибку. Но иногда компилятор может указать на другое место в вашем коде, поскольку ваша ошибка не сразу дала отрицательный эффект, а только после ее влияния на последующий код. Часто это случается при нарушении баланса открывающих и закрывающих скобок, когда это несоответствие обнаруживается уже ближе к концу вашего текста. Кроме этого, ошибки также бывают “наведенными”, когда первая ошибка разрушает логику в последующих местах вашего кода. Поэтому всегда надо исправлять ошибки, начиная с самой первой. Возможно, после ее исправления пропадут и некоторые последующие, смысл которых был вам совершенно непонятен.

Если полный список предупреждений и ошибок не поместился на экране, то его можно сохранить в текстовом файле и потом просмотреть любыми средствами. Для сохранения надо выполнить компиляцию с перенаправлением вывода стандартного канала ошибок

```
<команда компиляции> 2>имя
```

например,

```
gcc file.c 2>err.txt  
less err.txt
```

Отладчик gdb

```
gdb name
```

`name` — имя исполняемого файла, программа должна быть скомпилирована с ключем `-g`).

Основные команды отладчика

`q` (от `quit`) — выйти из отладчика

`r` (от `run`) — запустить программу

`p var` (от `print`) — посмотреть значение переменной `var`

`s` — (от `step`) сделать шаг программы, если данная строчка содержит вызов функции, то перейти к отладке этой функции

`n` (от `next`) — сделать шаг программы, перейдя к следующей строчке текста, если данная строчка содержит вызов функции, то исполнить ее “как единое целое”

`br name` (от `breakpoint`) — установить точку останова на функции *name*, если эта функция будет вызвана, то отладчик остановит программу и перейдет к отладке этой функции
`c` (от `continue`) — запустить программу после остановки
`bt` (от `back trace`) — посмотреть стек вызовов
`up` или `down` — перемещаться вверх-вниз по стеку вызовов
`help` — посмотреть справочную информацию

Несмотря на примитивный интерфейс и текстовый режим управления, отладчик `gdb` является мощным инструментом для отладки. Он позволяет устанавливать точки останова на конкретные строки программы, активировать эти остановы только при выполнении заданных условий, исследовать ваш код в разных местах программы и т.д. Но для использования подобных возможностей придется внимательно изучать инструкцию.