СИ и СИ++
в LINUX

По заказу фирмы Элтекс

Дидактические единицы

- Знакомство с Linux (команды, утилиты, оболочки, компиляторы, сценарии)
- СИ (СИ++) в Linux (функции, адресная арифметика, структуры, работа с файлами, параметры командной строки, управление процессами, создание библиотек)
- Системное программирование в Linux (порожденные процессы, средства их синхронизации – каналы, очереди, семафоры, потоки, мьютексы, сокеты).
- Взаимодействие программ с внешними устройствами, написание драйверов.
- Основы программирования в Qt.

Темы лекций

- **Всего 30 час:**
- Знакомство с Linux (оболочки, компиляторы, отладчики, make) (2 час)
- Создание и применение статических и динамических библиотек (2 час)
- Процессы и управление ими (2 час)
- Взаимодействие процессов (8 час)
- Многопоточное программирование (4 час)

Темы лекций

- Программирование сокетов (2 час)
- Программное взаимодействие с внешними устройствами (4 час)
- Кросс-компиляция (2 час)
- Написание драйверов (2 час)
- Основы программирования в Qt (2 час)

Темы лабораторных работ

- **Всего 70 час:**
- Знакомство с Linux (компиляторы, отладчики, make) (4 час)
- Функции, адресация, адресная арифметика (4 час)
- Структуры в СИ (4 час)
- Работа с файлами, параметры командной строки (4 час)
- Процессы и управление ими (4 час)

Темы лабораторных работ

- Взаимодействие процессов через каналы (4 час)
- Взаимодействие процессов через очереди сообщений (4 час)
- Взаимодействие процессов через разделяемую память и семафоры (4 час)
- Многопоточное программирование (4 час)

Темы лабораторных работ

- Программирование сокетов (4 час)
- Программное взаимодействие с внешними устройствами (8 час)
- Кросс-компиляция (8 час)
- Написание драйверов (8 час)
- Основы программирования в Qt (4 час)
- Итоговое занятие (прием задолженностей), выставление рейтингов (4 час)

Правила выставления рейтингов

- Общая сумма баллов в рейтинге не может превышать 100 баллов для согласования с системой оценок ECTS.
- Посещение лекции оценивается в 1 балл, отсутствие – в 0 баллов.
- Выполнение каждой лабораторной работы (за исключением ознакомительной) в срок (демонстрация работоспособной программы на текущем занятии) оценивается в 6 балла, на следующем занятии – в 3 балла.

Правила выставления рейтингов

- Возможно получение 1 дополнительного балла за досрочное (в течение 1 часа занятий) выполнение лабораторной работы, нестандартное решение задачи.
- Рейтинги передаются заказчику курса после половины занятий и по окончании курса.

Знакомство с Linux

- В большинстве Linux-систем, а во встраиваемых системах – всегда, работа с операционной системой ведется через какой-либо интерпретатор командной строки, shell. Наиболее популярным интерпретатором является bash (GNU Bourne-Again Shell).
- Для входа пользователя в систему можно использовать либо терминал компьютера, на котором установлен Linux (поддерживается до 6 сеансов одновременно, переключение производится клавишами <ALT>+F1…F6), либо терминальную программу на другом компьютере (обычно putty или shellguard в Windows-системах, в состав ОС не входят). В Linux-системах всегда имеется программа ssh (Secure Shell, защищенная оболочка) для шифрованного подключения по сети к серверу ssh.

Знакомство с Linux

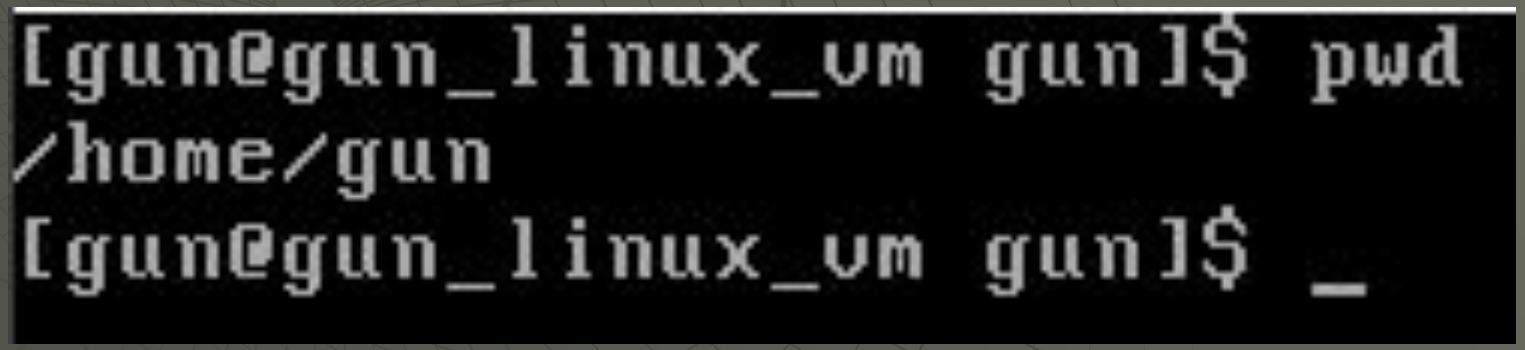
- Строка подключения имеет вид:
- **ssh [-l логин] имя_сервера или**
- **ssh логин@имя_сервера**
- Вместо имени сервера может быть указан его IP-адрес. Большинство администраторов Linux-систем запрещают подключение по сети суперпользователю (с логином root). В этом случае необходимо подключиться к серверу от имени обычного пользователя, а потом дать команду su. Если пользователь не указан, то предполагается суперпользователь, если указан (su user1), то предполагается вход от имени указанного пользователя.
- После входа пользователя в ОС (заметим, что при этом пароль не отображается вообще, даже звездочками) интерпретатор формирует приглашение вида [пользователь@имя_компьютера текущая_папка], например:
- **[gun@toshiba gun]$**

Знакомство с Linux

- Интерпретатор анализирует вводимые строки (после нажатия пользователем клавиши Enter) и определяет, содержат ли они команды операционной системы, утилиты, имена исполняемых файлов или сценариев, которые могут быть интерпретированы.
- Данные рассуждения не относятся к графическим оболочкам (GNOME, KDE), в которых интерпретируются также перемещения и клики мышью.
- Рассмотрим кратко основные команды, программы, утилиты ОС Linux и опции компиляторов языка С.

Команды

- Основными командами являются команды перемещения по файловой системе, команды создания, копирования (перемещения) и удаления файлов, папок и ссылок.
- Как известно, имя текущего каталога указано в приглашении интерпретатора. Но чтобы получить полный путь к текущему каталогу, используется команда **pwd**:



Команды

- Для просмотра содержимого текущего каталога используется команда **ls**.
- Данная команда выделяет цветом папки (голубым), исполняемые файлы и файлы сценариев (бледно-зеленым) и текстовые файлы (белым). Существуют также расцветки для ссылок и временных файлов.
- Команда **ls** имеет массу опций (ключей). Полный список можно получить через утилиту справки **man ls**.

```
[gun@gun_linux_vm gun]$ pwd
/home/gun
[gun@gun_linux_vm gun]$ ls
ipc   media    pipes      process  svshm    user.sh  XF86Config
ls.c  pass.sh  print_pid  svmsg    threads  work
[gun@gun_linux_vm gun]$ _
```

Команды

- Один из наиболее популярных ключей **–l**, обеспечивающий развернутый формат вывода с указанием размера и даты модификации файла, его принадлежности пользователю и группе пользователей, а также права доступа для пользователя, членов его группы и всех остальных.
- Зная, какие подкаталоги находятся в текущем каталоге, можно производить переходы по ним командой **cd** имя_каталога. Для выхода в каталог более высокого уровня используется команда **cd ..** (двоеточие).
- Для создания нового каталога используется команда **mkdir**, а для удаления - **rmdir**. Получить описания этих команд также можно с помощью утилиты **man**.
- Богата опциями команда копирования **cp**, позволяющая не только копировать и перемещать файлы, но и создавать на них ссылки.

Команды

- Часто после копирования файла (каталога), созданного другим пользователем, возникает проблема недостаточности прав на его использование. В этом случает создатель-владелец файла или каталога может передать принадлежность файла другому пользователю командой **chown** или изменить права доступа к нему командой **chmod**. Права доступа задаются либо символами rwx (где r – read (чтение), w – write (запись), x- eXecute (выполнение) для пользователя, его группы и всех остальных, либо четырьмя восьмеричными цифрами, которые складываются из битовых масок 4, 2 и 1. Все пустые места заполняются нулями.
- Первая цифра отвечает за установку идентификатора создателя (4), группы создателя (2) или бита владения (1).

Команды

- Вторая цифра обозначает права доступа для владельца: чтение (4), запись (2) и выполнение (1). Третья цифра указывает права доступа тех пользователей, которые входят в данную группу. Четвертая цифра обозначает права доступа остальных пользователей. Типичный пример: **chmod 666** или **chmod 777**.
- Обе данных команды поддерживают рекурсивный режим (ключ **–R**) для задания прав (принадлежности) на каталог и все вложенные в него файлы и подкаталоги.
- Для создания текстовых файлов служит команда **tee**. Для их просмотра – **cat**, для удаления **– unlink**.

Программы

- В системах программирования особую роль имеют такие программы, как текстовые редакторы, компиляторы и отладчики.
- Среди текстовых редакторов наиболее популярен редактор **vi**. Его особенностью является наличие нескольких режимов – замены, вставки, командный. По умолчанию редактор открывает текстовый файл в режиме замены, для перехода в режим вставки необходимо нажать клавишу **“i”**, для выхода из режима вставки используется клавиша **<Esc>**, для перехода в командный режим необходимо нажать **<Shift>+:**, а затем ввести команду: **“w”** – для записи файла, **“q”** – для выхода из редактора.

Программы

- Стандартным компилятором во всех Linux-системах является **gcc**. У данного компилятора много ключей и опций, получить список которых можно утилитой man. Наиболее часто применяется строка следующего вида:
- **gcc –Wall –o output input.c**
- Здесь **–Wall** означает вывод сообщений о всех предупреждениях и ошибках компиляции, **–o** задает имя исполняемого файла, после которого перечисляются через пробел все входные файлы, которые могут быть текстовыми, объектными и файлами библиотек.
- Минимально допустимая строка вызова компилятора включает только имя входного файла. В этом случае исполняемый файл получает имя **a.out**.

Программы

- Заметим, что при запуске любой программы предполагается, что она находится в одной из папок, перечисленных в переменной окружения PATH, как правило, это папки **/usr/bin** и **/usr/sbin**. Поскольку домашние папки пользователей Linux-систем в этот список не входят, то запуск программ пользователей производится с указанием полного пути, например, **/home/user/a.out**. Для запуска программ в текущей папке перед именем исполняемого файла указывают сокращенный путь вида **./**.

Программы

- Для компиляции программ написанных на языке C++ используется компилятор **g++**. Он отличается от gcc тем, что по умолчанию подключает не стандартную библиотеку C, а стандартную библиотеку C++. Все флаги и опции у G++ такие же точно, как и у gcc. Особенностью программ на языке C++ является то, что у include-файлов не надо указывать расширение .h, зато необходимо явно задавать пространство имен: **using namespace std;**, что позволит использовать стандартные потоки ввода-вывода.

Программы

- Для отладки программ рассмотрим отладчик gdb, входящий в комплект программ GNU. Для того, чтобы им пользоваться, нужно сначала скомпилировать программу так, чтобы её двоичный файл содержал отладочную информацию. Эта информация включает в себя, в частности, описание соответствий между адресами исполняемого кода и строками в исходном коде. Такая компиляция достигается путём добавления флага -g к команде на компиляцию. Например, для сборки программу kalkul, то дали такую команду:
- **g++ main.cpp -o kalkul –g**

Программы

- Запустим отладчик GDB, загрузив в него программу для отладки:
- **gdb ./kalkul**
- Чтобы запустить программу внутри отладчика, даётся команда run:
- **run**
- Чтобы посмотреть исходный код, даётся команда :
- **list**
- Если дать эту команду без параметров, то она первые девять строк исходного кода главного файла (то есть такого, в котором имеется функция main). Чтобы просматривать файл дальше, надо снова набирать list. Чтобы посмотреть конкретные строки, надо указать два параметра: с какой строки начинать просмотр, и с какой строки заканчивать.
- **list 12,15**

Программы

- Чтобы просмотреть другие файлы проекта, надо перед номерами строк указать название нужного файла и отделить его от номеров строк двоеточием.
- **list problem.cpp:20,29**
- Поставим точку останова на строке номер 21. Точка останова – это метка, указывающая, что программа, дойдя до этого места, должна остановиться.
- **break 21**
- Посмотреть, где вы поставили точки останова, можно с помощью команды **info breakpoints.**
- **info breakpoints**
- (При желании можно вместо номера строки указать название функции, тогда программа остановится перед входом в функцию.)

Программы

- Запустим программу.
- **run**
- Программа дойдёт до точки останова и остановится, выведя нам строку, у которой эта точка расположена. Нам, конечно, интересно знать, в каком именно месте мы остановились, и что программа уже успела выполнить. Даём команду **backtrace**.
- **backtrace**
- Отладчик выдаст, например, следующую информацию:
- **#0 CProblem::Calculate (this=0x804b008) at problem.cpp:21**
- **#1 0x08048e00 in CProblem::Solve (this=0x804b008) at problem.cpp:93**
- **#2 0x08048efc in main () at main.cpp:15**

Программы

- Это означается, что мы находимся внутри выполняющейся функции Calculate, являющейся функцией-членом класса CProblem. Она была вызвана из функции Solve того же класса, а та, в свою очередь, из функции **main**. Таким образом, команда **backtrace** показывает весь стек вызываемых функций от начала программы до текущего места. Посмотрим, чему же равно на этом этапе значение переменной Numeral.
- **print Numeral**
- Если мы вместо **print** будем пользоваться командой **display,** то величина этой переменной будет показываться каждый раз, когда программа останавливается, без специального указания**.**
- **display Numeral**
- Добавим ещё одну точку останова на строке 25 файла problem.cpp.
- **break problem.cpp:25**

Программы

- Продолжим выполнение программы:
- **continue**
- Команда **continue** продолжает выполнение программы с текущего адреса. Если бы мы набрали run, программа начала бы выполняться сначала. Посмотрим, чему равны значения переменных Numeral, SecondNumeral и Operation:
- **print Numeral**
- **print Operation**
- **print SecondNumeral**
- Отладчик **gdb** позволяет прямо во время выполнения программы изменить значение любой переменной.
- **set SecondNumeral=4**
- Если не верим, что её значение изменилось, можно проверить.
- **print SecondNumeral**

Программы

- Теперь уберём точки останова. Их было создано две. Но это можно проверить:
- **info breakpoints**
- Удалим их**:**
- **delete 1**
- **delete 2**
- Не должно остаться ни одной точки останова. Проверяем:
- **info breakpoints**
- Действительно не осталось ни одной. Теперь пошагово пройдём всю программу. Поставим точку останова на десятой строке главного файла.
- **break main.cpp:10**
- Запустим программу
- **run**

Программы

- Дойдя до десятой строчки, она остановится. Теперь проходим её, останавливаясь на каждой строчке, с помощью команды **step.**
- **step**
- Чтобы не набирать каждый раз **step**, можно просто вводить букву s. Чтобы при вызове функции, программа не входила в неё, а продолжала дальше выполняться только на текущем уровне стека, вместо **step** даётся команда **next** или просто n.
- **next**
- Если мы вошли в функцию, но не хотим дальше проходить её по шагам, а хотим, чтобы она отработала и вернула нас на предыдущий уровень стека (то есть, обратно в функцию, вызвавшую её), мы пользуемся командой **finish**.
- **finish**

Программы

- Таким образом, можно просмотреть, как выполняется вся программа или любой участок программы. На любом шаге можно проверять значение любой переменной. Чтобы перестать проходить программу по шагам и запустить её до конца, надо дать команду **continue.**
- Дадим короткий список наиболее часто встречающихся команд отладчика **gdb**. За более подробной информацией вы, конечно, всегда можете обратиться к встроенному описанию программы (**info gdb**) или руководством по пользованию (**man gdb**).
- **Команды отладчика gdb**:
- **backtrace** – выводит весь путь к текущей точке останова, то есть названия всех функций, начиная от **main()**; иными словами, выводит весь стек функций;
- **break** – устанавливает точку останова; параметром может быть номер строки или название функции;

Программы

- **clear** – удаляет все точки останова на текущем уровне стека (то есть в текущей функции);
- **continue** – продолжает выполнение программы от текущей точки до конца;
- **delete** – удаляет точку останова или контрольное выражение;
- **display** – добавляет выражение в список выражений, значения которых отображаются каждый раз при остановке программы;
- **finish** – выполняет программу до выхода из текущей функции; отображает возвращаемое значение, если такое имеется;
- **info breakpoints** – выводит список всех имеющихся точек останова;
- **info watchpoints** – выводит список всех имеющихся контрольных выражений;

Программы

- **list** – выводит исходный код; в качестве параметра передаются название файла исходного кода, затем, через двоеточие, номер начальной и конечной строки;
- **next** – пошаговое выполнение программы, но, в отличие от команды step, не выполняет пошагово вызываемые функции;
- **print** – выводит значение какого-либо выражения (выражение передаётся в качестве параметра);
- **run** – запускает программу на выполнение;
- **set** – устанавливает новое значение переменной
- **step** – пошаговое выполнение программы;
- **watch** – устанавливает контрольное выражение, программа остановится, как только значение контрольного выражения изменится;
- **quit** – выход из отладчика.

Программы

- Еще одной важной программой для разработчика программного обеспечения, работающего с сервером удаленно, является программа ftp. Если программа написана и отлажена на удаленном компьютере (по сети) то перенос исходного текста может быть произведен по протоколу ftp. Клиентские программы, реализующие подключение по данному протоколу, имеются в комплекте практически всех ОС, а синтаксис их вызова идентичен синтаксису подключения по ssh. Впрочем, с целью копирования программ с удаленного сервера справляются и программы-браузеры.

Утилиты

- К важным для программиста утилитам Linux относятся утилиты **ps**, **kill**, и файловый менеджер **mc**.
- Утилита **ps** позволяет получить список процессов в системе (аналог Диспетчера задач Windows) и их программные идентификаторы. Зная программный идентификатор (PID) задачи можно эту задачу принудительно завершить (снять с выполнения) утилитой **kill**.
- Утилита **ps** имеет немало опций, перечень которых можно получить из справочной системы **man**. Наиболее актуальна опция **ps –u логин_пользователя**, позволяющая получить список задач, запущенных от имени указанного пользователя.

Утилиты

- В случае зацикливания, «зависания» или приостановки прикладной программы пользователя рекомендуются следующие действия. Необходимо еще раз войти в систему (открыв новую консоль, используя сочетания клавиш **<ALT>+<F2>…<F6>** при работе за терминалом, или выполнив новое сетевое подключение), получить утилитой **ps** список процессов текущего пользователя, выяснить PID задачи, которая выполняется (running) или приостановлена (stopped), и вызвать утилиту **kill PID**, где PID – идентификатор задачи, которую требуется завершить. Данные действия особенно актуальны для систем, в которых у пользователей имеются ограничения (квоты) на использование дискового пространства и оперативной памяти, а также для встраиваемых (embedded) систем, в которых оперативная память ограничена аппаратно.

Утилиты

- Заметим, что для выполнения указанных действий для задач, запущенных от имени другого пользователя требуются соответствующие права доступа. Полные права доступа имеет суперпользователь с логином root.
- Еще одна полезная для программиста программа – файловый менеджер и текстовая оболочка **mc** (Midnight Commander), содержащаяся в дистрибутивах многих версий Linux-систем, но зачастую не устанавливаемая по умолчанию. Данный файловый менеджер позволяет выполнять многие системные команды, используя функциональные клавиши и сочетания клавиш. В **mc** удобно перемещаться по дереву каталогов, выполнять различные операции с файлами и папками (создавать, копировать, перемещать, удалять), для которых применяются практически те же клавиши и сочетания клавиш, как и в популярном файловом менеджере FAR для Windows.

Утилиты

- Менеджер **mc** содержит встроенный текстовый редактор, открывающий файл по нажатию клавиши <F4>, с подсветкой ключевых слов и знаков препинания. Интересна возможность сворачивания окна редактирования (сочетанием клавиш <Ctrl>+O) для отображения консоли, содержащей, например, сообщения об ошибках компиляции программы.
- Основные команды редактора выполняются клавишами <F4>, <Shift>+<F4> (создание нового файла), <F2> (сохранение изменений), <F10> (выход).
- Особенностью редактора являются операции с блоками текста (начало и окончание выделения текста производится клавишей <F3>, собственно выделение - <Shift>+<стрелка>, копирование блока - <F5>, перемещение - <F6>, отмена последнего действия - <Ctl>+U.