Занятие 2.
Bourne Shell aka POSIX sh.

Влад 'mend0za' Шахов
Linux & Embedded Team Leader

Linux & Embedded Department

Что такое Unix shell?

Что такое Unix shell? (Назойливый повтор)

- Обычная программа, запускающаяся после входа в систему
- Интерактивный командный интерпретатор
- Язык программирования
- Платформа интеграции (для утилит)
- Сотни разных реализаций (bash, ksh, zsh, tcsh, . . .)
- Масса различных диалектов

Shell. Ключевые понятия - 1

Определения

- Приглашение командной строки (CMD PROMPT):

Shell. Ключевые понятия - 1

Определения

- Приглашение командной строки (CMD PROMPT):
  $, #, user@host:~$

Shell. Ключевые понятия - 1
Определения

- Приглашение командной строки (CMD PROMPT):
  $, #, user@host:~$

- Команда:
  whoami; top; exit

Shell. Ключевые понятия - 1
Определения

- Приглашение командной строки (CMD PROMPT):
  $, #, user@host:~$
- Команда:
  whoami; top; exit
- Параметр:
  man bash; who am i

Shell. Ключевые понятия - 1
Определения

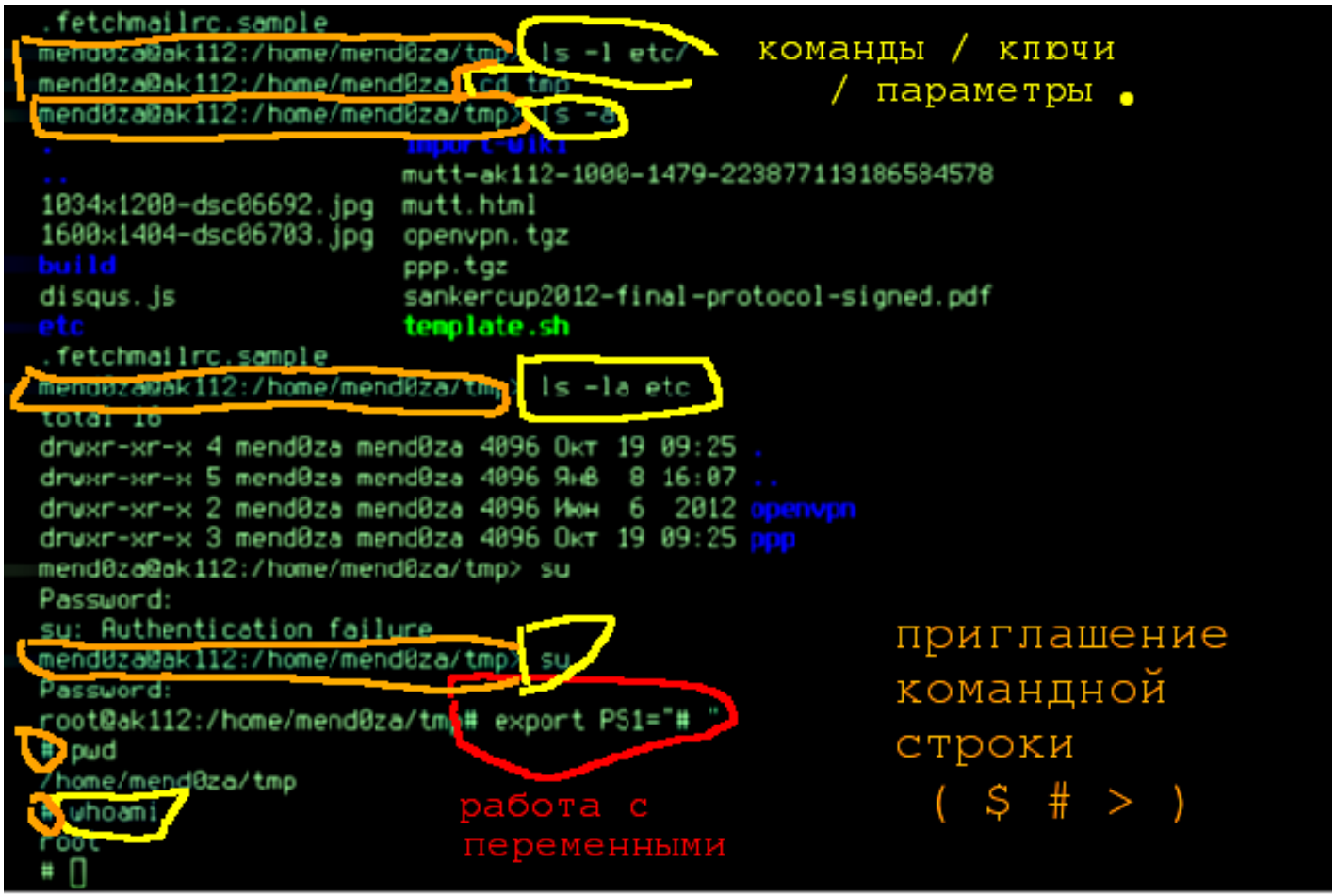
- Приглашение командной строки (CMD PROMPT):
  $, #, user@host:~$
- Команда:
  whoami; top; exit
- Параметр:
  man bash; who am i
- Ключ (1 символ):
  ls -a; ls -al; ls -a -l /tmp/

Shell. Ключевые понятия - 1
Определения

- Приглашение командной строки (CMD PROMPT):
  $, #, user@host:~$
- Команда:
  whoami; top; exit
- Параметр:
  man bash; who am i
- Ключ (1 символ):
  ls -a; ls -al; ls -a -l /tmp/
- Длинный ключ (GNU-style:
  ls --version

Shell. Ключевые понятия - 2

Картинка для закрепления



Приёмы эффективной работы

Как в Shell работать быстро?

Приёмы эффективной работы

Как в Shell работать быстро?

1. автодополнение путей и команд
2. история команд
3. редактирование командной строки

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26
- Пути и имена файлов

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26
- Пути и имена файлов
  Пример: ls /u[TAB]lo[TAB]sh[TAB]/ca[TAB]

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26
- Пути и имена файлов
  Пример: ls /u[TAB]lo[TAB]sh[TAB]/ca[TAB]
  Результат: ls /usr/local/share/ca-certificates/
  16 vs 36

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26
- Пути и имена файлов
  Пример: ls /u[TAB]lo[TAB]sh[TAB]/ca[TAB]
  Результат: ls /usr/local/share/ca-certificates/
  16 vs 36
- Параметры и ключи [1]

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26
- Пути и имена файлов
  Пример: ls /u[TAB]lo[TAB]sh[TAB]/ca[TAB]
  Результат: ls /usr/local/share/ca-certificates/
  16 vs 36
- Параметры и ключи [1]
  Пример: apti[TAB]--a[TAB]sh[TAB]core[TAB][ENTER]

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26
- Пути и имена файлов
  Пример: ls /u[TAB]lo[TAB]sh[TAB]/ca[TAB]
  Результат: ls /usr/local/share/ca-certificates/
  16 vs 36
- Параметры и ключи [1]
  Пример: apti[TAB]--a[TAB]sh[TAB]core[TAB][ENTER]
  Результат: aptitude --assume-yes show coreutils
  16 vs 37

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 1

Волшебная кнопка - TAB

- Имя команды
  Пример: mys[TAB]_co[TAB]
  Результат: mysql_convert_table_format
  8 vs 26
- Пути и имена файлов
  Пример: ls /u[TAB]lo[TAB]sh[TAB]/ca[TAB]
  Результат: ls /usr/local/share/ca-certificates/
  16 vs 36
- Параметры и ключи [1]
  Пример: apti[TAB]--a[TAB]sh[TAB]core[TAB][ENTER]
  Результат: aptitude --assume-yes show coreutils
  16 vs 37

[1] Только у BASH и ZSH (если настроены)

Приёмы эффективной работы
Автодополнение путей и команд - 2

Единственный вариант подстановки:
TAB дополняет сразу

Приёмы эффективной работы
Автодополнение путей и команд - 2

Единственный вариант подстановки:

TAB дополняет сразу

Несколько вариантов подстановки?

Ещё больше волшебства - 2 кнопки TAB!
2xTAB - список вариантов подстановки

Приёмы эффективной работы
Автодополнение путей и команд - 3

Примеры

- apt[TAB][TAB]
- aptitude --[TAB][TAB][2]
- ls /[TAB][TAB] [3]

[2] Только для BASH и ZSH
[3] Можно использовать вместо команды “ls”

Приёмы эффективной работы
История команд

Просмотр истории

- “Up” и “Down” - вперёд-назад

Приёмы эффективной работы
История команд

Просмотр истории

- “Up” и “Down” - вперёд-назад
- “Ctrl+R” - интерактивный поиск в истории

Приёмы эффективной работы
История команд

Просмотр истории

- “Up” и “Down” - вперёд-назад
- “Ctrl+R” - интерактивный поиск в истории
- повторно ”Ctrl+R“ - искать дальше

Приёмы эффективной работы
Редактирование командной строки

Emacs editing mode [4]

- “Left” и “Right” - вперёд-назад по текущей строке

[4] Только KSH-совместимые: bash, zsh, pdksh, mksh, etc

Приёмы эффективной работы
Редактирование командной строки

Emacs editing mode [4]

- “Left” и “Right” - вперёд-назад по текущей строке
- “Ctrl+a” и “Ctrl+e” - перейти в начало и конец строки

[4] Только KSH-совместимые: bash, zsh, pdksh, mksh, etc

Приёмы эффективной работы
Редактирование командной строки

Emacs editing mode [4]

- “Left” и “Right” - вперёд-назад по текущей строке
- “Ctrl+a” и “Ctrl+e” - перейти в начало и конец строки
- “Ctrl+u” - удалить от курсора до начала строки

[4] Только KSH-совместимые: bash, zsh, pdksh, mksh, etc

Приёмы эффективной работы
Редактирование командной строки

Emacs editing mode [4]

- “Left” и “Right” - вперёд-назад по текущей строке
- “Ctrl+a” и “Ctrl+e” - перейти в начало и конец строки
- “Ctrl+u” - удалить от курсора до начала строки
- “Ctrl+w” - удалить слово (от курсора до разделителя, влево)

[4] Только KSH-совместимые: bash, zsh, pdksh, mksh, etc

Условное выполнение команд

Код возврата (RETURN CODE):

результат выполнения у любой команды Shell

Shell return code:

- 0 - выполнень успешно
- не 0 - ошибка

Условное выполнение команд

Код возврата (RETURN CODE):

результат выполнения у любой команды Shell

Shell return code:

- 0 - выполнень успешно
- не 0 - ошибка

Операции над кодом возврата:

- “&&” - логическое И
- “||” - логическое ИЛИ

Условное выполнение команд

Код возврата (RETURN CODE):

результат выполнения у любой команды Shell

Shell return code:

- 0 - выполнень успешно
- не 0 - ошибка

Операции над кодом возврата:

- “&&” - логическое И
- “||” - логическое ИЛИ

Примеры:

- cat /proc/1/environ || echo fail
- find /usr/share/doc -name “*.txt” && echo ok

Скрипты

Shell Script, определение

Последовательность команд Shell.

Разделитель: перевод строки, “;”

Скрипты

Shell Script, определение

Последовательность команд Shell.
Разделитель: перевод строки, “;”

shebang

#!something или чем мы запускаем скрипт.
По умолчанию : #!/bin/sh
Всегда первая строка скрипта.
Фактически: /bin/sh scriptname

Скрипты

Shell Script, определение

Последовательность команд Shell.
Разделитель: перевод строки, “;”

shebang

#!something или чем мы запускаем скрипт.
По умолчанию : #!/bin/sh
Всегда первая строка скрипта.
Фактически: /bin/sh scriptname

Парадоксальные примеры

#!/bin/rm
#!/bin/awk -f
#!/bin/less

Запуск скриптов

1. sh scriptname
2. chmod +x script
   ./script
3. из каталогов в переменной PATH
   echo $PATH
   ~/bin (если есть)
   /usr/local/bin
4. в текущей копии shell[5]
   . ./script
   source script[6]

[5] Остальные способы - запускают новый shell

[6] Несовместимо с POSIX. Происходит из ksh. Добавляет текущий каталог к списку путей

Потоки ввода-вывода

Особенности архитектуры[7]:

У каждой запущенной программы 3 потока I/O:

0. ввода
1. вывода
2. ошибок

Связаны с экраном и клавиатурой терминала.

[7] См документацию языка программирования Си

Потоки ввода-вывода

Особенности архитектуры[7]:

У каждой запущенной программы 3 потока I/O:

0. ввода
1. вывода
2. ошибок

Связаны с экраном и клавиатурой терминала.

Связаны с терминалом только по умолчанию

shell позволяет переопределить весь ввод и вывод программы

[7] См документацию языка программирования Си

Базовый синтаксис перенаправления

- Ввод “<”

  sort <.bash_history

[8] Файл затрёт новым содержанием, если он существовал ранее

Базовый синтаксис перенаправления

- Ввод “<”
  sort <.bash_history
- Вывод “>” [8]
  find /usr/share/doc -name “*.txt” >txt-docs
- Вывод “1>”
  find /usr/share/doc -name “*.txt” 1>txt-docs

[8] Файл затрёт новым содержанием, если он существовал ранее

Базовый синтаксис перенаправления

- Ввод “<”
  sort <.bash_history
- Вывод “>” [8]
  find /usr/share/doc -name “*.txt” >txt-docs
- Вывод “1>”
  find /usr/share/doc -name “*.txt” 1>txt-docs
- Ошибки “2>”
  find /tmp 2>find.errors

[8] Файл затрёт новым содержанием, если он существовал ранее

Базовый синтаксис перенаправления

- Ввод “<”
  sort <.bash_history
- Вывод “>” [8]
  find /usr/share/doc -name “*.txt” >txt-docs
- Вывод “1>”
  find /usr/share/doc -name “*.txt” 1>txt-docs
- Ошибки “2>”
  find /tmp 2>find.errors
- Вывод (дописать в конец) “1>>”
  find /usr/share/doc -name “*.txt” >>txt-docs
- Ошибки (дописать в конец) “2>>”
  find /tmp 2>>find.errors

[8] Файл затрёт новым содержанием, если он существовал ранее

Базовый синтаксис перенаправления

- Ввод “<”
  sort <.bash_history
- Вывод “>” [8]
  find /usr/share/doc -name “*.txt” >txt-docs
- Вывод “1>”
  find /usr/share/doc -name “*.txt” 1>txt-docs
- Ошибки “2>”
  find /tmp 2>find.errors
- Вывод (дописать в конец) “1>>”
  find /usr/share/doc -name “*.txt” >>txt-docs
- Ошибки (дописать в конец) “2>>”
  find /tmp 2>>find.errors

[8] Файл затрёт новым содержанием, если он существовал ранее

Расширенный синтаксис перенаправления

- Pipe [9] “cmd1 | cmd2 ”
  Вывод cmd1 направляется на ввод cmd2.
  man bash|grep ksh

[9] Классика Unix
[10] In Real Life (IRL) используется только в скриптах

Расширенный синтаксис перенаправления

- Pipe [9] “cmd1 | cmd2 ”
  Вывод cmd1 направляется на ввод cmd2.
  man bash|grep ksh
- Склеить потоки “N>&M”
  В примере: просмотреть одновременно и вывод и ошибки
  find /tmp 2>&1 | less

[9] Классика Unix
[10] In Real Life (IRL) используется только в скриптах

Расширенный синтаксис перенаправления

- Pipe [9] “cmd1 | cmd2 ”
  Вывод cmd1 направляется на ввод cmd2.
  man bash|grep ksh
- Склеить потоки “N>&M”
  В примере: просмотреть одновременно и вывод и ошибки
  find /tmp 2>&1 | less
- ”Ввод здесь“[10] ”<<END_MARKER“

```
sort <<EOF
oieu
ak
zf
EOF
```

[9] Классика Unix
[10] In Real Life (IRL) используется только в скриптах