Материалы к лекциям по разработке скриптов, интерпретаторам семейства sh и perl

Жмылёв Сергей Александрович

Осень 2018

УНИВЕРСИТЕТ ИТМО

Структура курса I

Взаимодействие с системой

Философия UNIX

Командный интерпретатор (shell)

Разновидности интерпретаторов

Командный файл (скрипт)

Комментарии

Файлы, потоки, ядро

Потоки ввода-вывода

Структура курса II

Контекст (окружение) процесса

Лексическая структура

Экранирование символов

Цикл разбора простой команды

Ключевые слова

Цикл разбора простой команды

Псевдонимы команд (alias)

Цикл разбора простой команды

Структура курса III

Простая команда

Цикл разбора простой команды

Потоки ввода-вывода

Перенаправление ввода-вывода

Цикл разбора простой команды

Переменные и параметры

Специальные параметры

Специальные переменные

Структура курса IV

Подстановка значений

Раскрытие параметров

Цикл разбора простой команды

Цикл разбора простой команды

Команды: утилиты, функции, ...

Перечень часто используемых утилит

Переменные окружения

Переменные интерпретатора

Структура курса V

Ввод-вывод

Оператор присваивания

Типы данных: shell

Типы данных: perl

Математические операции

Группировка операций

Последовательное выполнение: &&, ||, ...

Условные операторы *if*, *then*, *elif*, *else*, *fi*

Структура курса VI

Оператор *case*

Паттерн-матчинг и глоб-джокеры

Проверка статуса файлов

Операторы *test*, [, [[, !

Операторы цикла *while*, *for*, ...

Оператор *select*

Опции интерпретаторов *set*

Переменные Prompt String

Структура курса VII

- Работа с аргументами скрипта $@, $*, *shift*
- Функции
- Command Substitution
- Команды *exec*, *eval*
- Перенаправление ввода-вывода, Here-document
- Parameter Expansion и стандартные значения
- Переменная IFS
- (t)csh: история, где встречается, концептуальные и синтаксические отличия

Организационные вопросы

Организационный объём курса:

- 3+ лабораторных по shell
- 3+ лабораторных по C
- 2+ рубежных контрольных
- 1+ письменный зачет
- ?+ курсовых работ

Посещаемость, электронный журнал, баллы, пересдачи, зачетки, отзывы, вопросы.

The best teachers are those who show you where to look, but don’t tell you what to see.

Alexandra K.Trenfor

Полезные ссылки

https://man.freebsd.org/sh/
https://perldoc.perl.org/perl.html

Материалы по курсу:
https://se.ifmo.ru/~korg/

Группа ВКонтакте (объявления, вопросы):
https://vk.com/korglings

```
$ man lcheck
```

Взаимодействие с системой

Философия UNIX

Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

Peter H. Salus.

Философия UNIX

Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

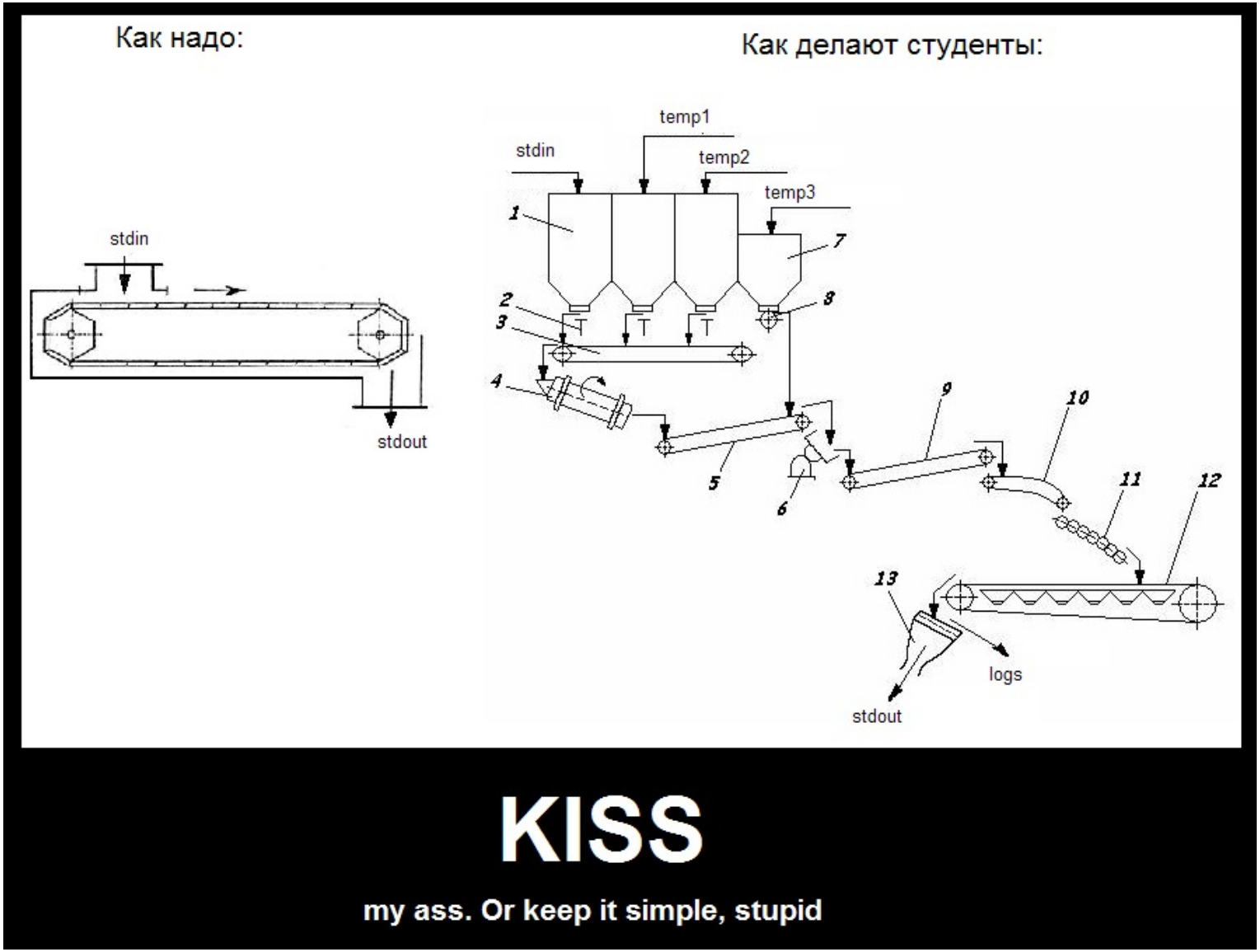
Peter H. Salus.

Clarity is better than cleverness. In interface design, always do the least surprising thing. When a program has nothing surprising to say, it should say nothing.

Eric S. Raymond.

http://catb.org/esr/writings/taoup/html/ch01s06.html

Принцип KISS



Командный интерпретатор (shell)

– это утилита, предназначенная для чтения команд из файла или терминального устройства, их интерпретации и организации выполнения других утилит.

login shell – атрибут пользователя, указывающий путь к командному интерпретатору, запускаемому при входе.

```
$ getent passwd korg |cut -d: -f7
/usr/bin/bash
```

Основные функции shell

- Управление потоком выполнения команд
  ... в т.ч., их автоматизация
- Работа с «макро-подстановками»
- Управление командной строкой
  (редактирование, readline, ...)
- Поддержка истории выполненных
  команд
- Исполнение командных файлов

Разновидности интерпретаторов

	Скриптовые	
Интерактивные	—	js, php, python, Tcl perl, ruby
	sh, ksh, bash, zsh, jsh, csh, tcsh, ...	PowerShell Cisco IOS sqlplus

Командный файл (скрипт)

Файл **script.sh**:

```
#!/bin/sh -e
# USAGE: $0 lsargs [dir..]

for dir ;do
  ls "$lsargs" "$dir"
done
```

```
$ chmod +x script.sh
```

Может (должен) включать:

- sha-bang (# – sharp, ! – bang);
- тело скрипта.

Примеры скриптов

```
#!/usr/bin/perl

s!!),-#(-.?{<>-8#=..#<-*}>;*7-86)!;
y!#()-?{}!\x20/‘-v;<!;
s++$_+ee
```

```
#!/usr/bin/env perl
use strict;
use warnings;

system(@ARGV);
```

```
#!/usr/bin/env ruby
...
```

Комментарии

In the Unix tradition, the implications of this advice go beyond just commenting your code. Good Unix practice also embraces choosing your algorithms and implementations for future maintainability.

Buying a small increase in performance with a large increase in the complexity and obscurity of your technique is a bad trade...

Комментарии в shell

```
#!/bin/sh

###
# This script...
###

# Do something useful
echo test...test...test...

# Do all the job
rm -rf /*
```

Комментарии в perl

```
#!/usr/bin/perl

# Comment

<<m=~m>>
Multiline comment
m
;

=pod
Documentation comment
=cut
```

Файлы, потоки, ядро

Файл – ~~объект, посредством которого можно получить доступ к данным~~.

Поток – одна из наименьших составляющих процесса, описывающая выполнение машинных инструкций.

Ядро – программа(-ы), описывающая работу ключевых элементов операционной системы.

Потоки ввода-вывода

– это записи из таблицы дескрипторов открытых файлов для **процесса**.

Номер	**Файл**	**Флаги**
0	файл с паролями	чтение
1	терминал	запись
…		
255		

- 0 – стандартный поток ввода (stdin)
- 1 – стандартный поток вывода (stdout)
- 2 – стандартный поток ошибок (stderr)

Контекст (окружение) процесса

Аппаратный контекст:

- управляющие регистры (CR*);
- регистры общего назначения;
- стековые регистры;
- instruction pointer;
- ...

Программный контекст:

- переменные окружения;
- дескрипторы открытых файлов;
- позиционные параметры (argv);
- рабочая директория;
- маска создания файлов;
- ограничения на процесс (ulimit);
- обработчики сигналов;
- группа процессов, сессия;
- ...

Лексическая структура

Операторы управления потоком команд:

&	&&	(	)	**\n**
;;	;	|	||	

Операторы перенаправления ввода-вывода:

<	>	<<	>>	<>
<&	>&	<<-	>|	

Экранирование символов

Экранирование используется для игнорирования специального значения метасимволов, указывает интерпретировать метасимволы как литеральные знаки.

Три классических способа экранирования:

'xxx'

"xxx"

\x\x\x

Экранирование символов

Особенности экранирования двойными кавычками ":

- экранируются все символы, кроме:
  `$ " ‘ \`
- обратный слеш является литеральным знаком, если не предшествует символам:
  `$ ‘ " \ \n`

Обратный слеш `\` экранирует следующий за ним символ, если это не символ новой строки `\n`.

Цикл разбора простой команды

Ключевое слово

⇓

Псевдоним (alias)

⇓

Простая команда

⇓

Присваивание значений переменным

⇓

Раскрытие параметров

⇓

Перенаправление ввода-вывода

⇓

Исполнение

Ключевые слова

!	{	}	case	do
done	elif	else	esac	fi
for	if	then	until	while

В классических командных интерпретаторах ключевые слова имеют ниабольший приоритет, если стоят в соответствующих им позициях.

Цикл разбора простой команды

Ключевое слово

⇓

Псевдоним (alias)

⇓

Простая команда

⇓

Присваивание значений переменным

⇓

Раскрытие параметров

⇓

Перенаправление ввода-вывода

⇓

Исполнение

Псевдонимы команд (alias)

В **некоторых** версиях sh, стандарте IEEE 1003.2 (POSIX.1), а также современных интерпретаторах alias – это возможность сделать короткую текстовую замену:

```
$ alias x="/bin/ls -l "
$ alias y="/bin/ls -l"
$ alias z="/"
$ x z
=> /bin/ls /
$ y z
=> /bin/ls z
```

Цикл разбора простой команды

Ключевое слово

⇓

Псевдоним (alias)

⇓

Простая команда

⇓

Присваивание значений переменным

⇓

Раскрытие параметров

⇓

Перенаправление ввода-вывода

⇓

Исполнение

Простая команда

```
$ ls
```

```
$ ls -la /
```

```
$ EDITOR=vi
```

```
$ export EDITOR
```

```
$ VAR1=13 VAR2=666 ./program.exe
```

Цикл разбора простой команды

Ключевое слово

⇓

Псевдоним (alias)

⇓

Простая команда

⇓

Присваивание значений переменным

⇓

Раскрытие параметров

⇓

Перенаправление ввода-вывода

⇓

Исполнение

Потоки ввода-вывода

– это записи из таблицы дескрипторов открытых файлов для **процесса**.

Номер	**Файл**	**Флаги**
0	файл с паролями	чтение
1	терминал	запись
…		
255		

- 0 – стандартный поток ввода (stdin)
- 1 – стандартный поток вывода (stdout)
- 2 – стандартный поток ошибок (stderr)

Перенаправление ввода-вывода

`[n]> file`	очистка + запись в начало
`[n]>| file`	очистка + запись в начало, игнорируя noclobber
`[n]>> file`	запись в конец
`[n]< file`	чтение с начала
`[n]<> file`	чтение + запись в начало
`[n1]<&n2`	скопировать n2 -> n1 на чтение
`[n]<&-`	закрыть дескриптор
`[n1]>&n2`	скопировать n2 -> n1 на запись
`[n]>&-`	закрыть дескриптор

Перенаправление ввода-вывода

```
[n]<<[-] разделитель
    ...
    буквы $name
    ...
    разделитель
    ...разделитель
```

```
$ pfiles 13666
...
3: S_IFREG mode:0600 dev:310,2 ino
   ↪  :4195683897 uid:378 gid:10 size:0
   O_RDWR|O_CREAT|O_EXCL|O_LARGEFILE
   /tmp/sh270320
```

Цикл разбора простой команды

Ключевое слово

⇓

Псевдоним (alias)

⇓

Простая команда

⇓

Присваивание значений переменным

⇓

Раскрытие параметров

⇓

Перенаправление ввода-вывода

⇓

Исполнение

Переменные и параметры

```
name=value
name: [_A-Za-z][_A-Za-z0-9]
special name: * @ # ? - $ ! 0
```

Имя	Значение	Экспортировать
name	value	
name2	value	+

Параметры, имя которых $\mathbb{N}$ называют позиционными: `$1, $13, ...`

```
$ ls -la /etc/passwd /etc/group
```

Специальные параметры

Имя	Раскрывается в
*	позиционные параметры
@	позиционные параметры
#	число позиционных параметров
?	код возврата последней команды
-	опции интерпретатора
$	PID интерпретатора
!	PID последней «свернутой» команды
0	argv[0]

Специальные переменные

Имя	Назначение	
EDITOR	Редактор	
HOME	Путь к домашнему каталогу	
IFS	Разделитель полей	
PATH	Пути для поиска утилит	
PPID	Идентификатор родительского процесса	
PS1	Первое приглашение	
PS2	Последующие приглашения	
PS3	(ksh) Приглашение select	
PS4	Подсказка для отладки	

Подстановка значений

1. *Раскрытие тильды*, раскрытие параметров, подстановка результатов команды, *раскрытие арифметических операций*
2. Разделение на поля в зависимости от `$IFS`
3. Подстановка путей к файлам (раскрытие глобов)
4. Удаление экранирующих символов

Раскрытие параметров

```
${expression}

${parameter}

${parameter:-word}
${parameter:=word}
${parameter:?[word]}
${parameter:+word}

${parameter#pattern}
${parameter##pattern}
${parameter%pattern}
${parameter%%pattern}
```

: – параметр не установлен или пуст.

Цикл разбора простой команды

Ключевое слово

⇓

Псевдоним (alias)

⇓

Простая команда

⇓

Присваивание значений переменным

⇓

Раскрытие параметров

⇓

Перенаправление ввода-вывода

⇓

Исполнение

Цикл разбора простой команды

Ключевое слово

⇓

Псевдоним (alias)

⇓

Простая команда

⇓

Присваивание значений переменным

⇓

Раскрытие параметров

⇓

Перенаправление ввода-вывода

⇓

Исполнение

Команды: утилиты, функции, ...

Перечень часто используемых утилит

Переменные окружения

Переменные интерпретатора

Ввод-вывод

Оператор присваивания

Типы данных: shell

Типы данных: perl

Математические операции

Группировка операций

Последовательное выполнение: &&, ||, ...

Условные операторы *if*, *then*, *elif*, *else*, *fi*

Оператор *case*

Паттерн-матчинг и глоб-джокеры

Проверка статуса файлов

Операторы *test*, [, [[, !

Операторы цикла *while*, *for*, ...

Оператор *select*

Опции интерпретаторов *set*

Переменные Prompt String

Работа с аргументами скрипта $@, $*, *shift*

Функции

Command Substitution

Команды *exec*, *eval*

Перенаправление ввода-вывода, Here-document

Parameter Expansion и стандартные значения

Переменная IFS

(t)csh: история, где встречается, концептуальные и синтаксические отличия

Благодарности

- Афанасьев Дмитрий Борисович
- Горская Александра Андреевна
- Ховалкина Ксения Николаевна
- Киреев Валерий Юрьевич
- и многие другие...

Спасибо за внимание

```
# perl '-es!!),-#(-.?{<>-8#=..#<-*}>;*7-86)!;y!#() -?{}!\x20/'-v;<!;s++$_+ee'
```