

КПО

Скрипты, Makefile, Configure

Лекция №8 (версия 1.0)

```
resent"/>  
fish.web.present
```

```
<!-- do not forg
```

```
oot}" else="{gfv
```

```
app.context-root]
```

```
resent">
```

```
b]"/>
```

Работа с проектом

- Создать каталог / структуру каталогов
- Создать файл / множество файлов
- Написать код
- Проверить код
- Откомпилировать код
- Запустить и проверить модуль / систему
- Собрать модули в исполняемые файлы
- Создать дистрибутив

```
sent"/>  
fish.web.present
```

```
<!-- do not forg
```

```
oot}" else="$ {gfv
```

```
app.context-root]
```

```
resent">
```

```
b]"/>
```

Скрипты

- Windows
 - BAT (Batch, DOS)
 - CMD (Command, Windows NT)
 - PowerShell (Windows 2003)
- Linux / Unix
 - sh
 - bash
 - И т.п.

```
resent"/>  
fish.web.present
```

```
<!-- do not forg
```

```
oot}" else="$gfv
```

```
app.context-root]
```

```
resent">
```

```
b]"/>
```

BAT

- REM (goto mark & :mark)
- ECHO ON/OFF
- PAUSE
- @
- CALL
- IF, FOR

COMMAND.COM / CMD.EXE

```
sent"/>  
fish.web.present
```

```
<!-- do not forg
```

```
oot}" else="{gfv
```

```
app.context-root]
```

```
resent">
```

```
b]"/>
```

BAT

```
sent"/>  
fish.web.present  
<!-- do not forg
```

```
2 <project def  
3   <target r  
4     <proj  
5     <ava  
6     <ava  
7     <ava  
8     <temp  
9     <ech  
10  </target>  
11  
12  <target r  
--
```

Пример пакетного файла, вычисляющего выражения:

```
:start  
@echo off  
cls  
title Калькулятор  
color 71  
echo Введите уравнение:  
set /p Exp=  
set /a Result=%Exp%  
cls  
echo Вычислено  
echo Ваше уравнение: %Exp%  
echo Результат: %Result%  
echo.  
echo Нажмите любую клавишу . . .  
pause > nul  
goto start
```

1. Метка, создающая цикличность программы.
2. Выключение эха (вывода выполняющихся строк на экран).
3. Очистка экрана.
4. Изменение заголовка окна Windows на строку «Калькулятор».
5. Изменение цвета шрифта и фона (тёмно-синий на светло-сером).
6. Вывод строки «Введите уравнение».
7. Создание переменной `Exp` для хранения ввода пользователя.
8. Вычисление результата выражения и помещение его в переменную `Result`.
9. Очистка экрана.
10. Вывод строки «Вычислено».
11. Вывод строки «Ваше уравнение» и значения переменной `Exp`.
12. Вывод строки «Результат» и значения переменной `Result`.
13. Остановка выполнения кода до нажатия любой клавиши.
14. Переход на метку `start`, выполнение кода начнётся со следующей после метки команды.

```
resent">  
b}"/>
```

```
33 </target>  
34 <target n  
35 <temp  
36 <copy
```

Bash

- #
- Преамбула (#!)
- echo
- let / set

```
resent"/>  
fish.web.present
```

```
<!-- do not forg
```

```
oot}" else="{gfv
```

```
app.context-root]
```

```
resent">
```

```
b]"/>
```

PowerShell

- **Командлеты (англ. cmdlets)** — это специализированные команды PowerShell, которые реализуют различную функциональность. Это встроенные в PowerShell команды. Командлеты именуются по правилу Глагол-Существительное, например Get-ChildItem, благодаря чему их предназначение понятно из названия. Командлеты выводят результаты в виде объектов или их коллекций. Дополнительно, командлеты могут получать входные данные в такой же форме и, соответственно, использоваться как получатели в конвейере. Хотя PowerShell позволяет передавать по конвейеру массивы и другие коллекции, командлеты всегда обрабатывают объекты поочередно. Для коллекции объектов обработчик командлета вызывается для каждого объекта в коллекции по очереди.

PowerShell

- В PowerShell, как и в оболочках UNIX/Linux, присутствует конвейер. Этот конвейер служит для передачи выходных данных одного командлета во входные данные другого командлета.
- PowerShell включает язык сценариев с динамическими типами, на котором можно реализовывать сложные операции с использованием командлетов. Язык сценариев поддерживает переменные, функции, конструкции ветвления (if-then-else) циклы (while, do, for и foreach), структурированную обработку ошибок и множество других возможностей, включая интеграцию с .NET.

PowerShell

```
Windows PowerShell
PS C:\> Get-ChildItem 'MediaCenter:\Music' -rec |
>> where { -not $_.PSIsContainer -and $_.Extension -match 'wma|mp3' } |
>> Measure-Object -property length -sum -min -max -ave
>>
Count          : 1307
Average        : 5491276.09563887
Sum            : 7177097857
Maximum        : 22905267
Minimum        : 3235
Property       : Length

PS C:\> Get-WmiObject CIM_BIOSElement | select biosv*, man*, ser* | Format-List

BIOSVersion    : <TOSCPPL - 6040000, Ver 1.00PARTBL>
Manufacturer   : TOSHIBA
SerialNumber    : M821116H

PS C:\> <[wmiSearcher]@'
>> SELECT * FROM CIM_Job
>> WHERE Priority > 1
>> 'E).get() | Format-Custom
>>
class ManagementObject#root\cimv2\Win32_PrintJob
{
    Document = Monad Manifesto - Public
    JobId = 6
    JobStatus =
    Owner = User
    Priority = 42
    Size = 1027088
    Name = Epson Stylus COLOR 740 ESC/P 2, 6
}

PS C:\> $rssUrl = 'http://blogs.msdn.com/powershell/rss.aspx'
PS C:\> $blog = [xml](new-object System.Net.WebClient).DownloadString($rssUrl)
PS C:\> $blog.rss.channel.item | select title -first 3

title
-----
MMS: What's Coming In PowerShell U2
PowerShell Presence at MMS
MMS Talk: System Center Foundation Technologies
```

```
sent"/>
fish.web.present
```

```
<!-- do not for
```

```
oot}" else="$gfv
```

```
app.context-root
```

```
resent">
```

```
b]"/>
```

Make

make — утилита, автоматизирующая процесс преобразования файлов из одной формы в другую. Чаще всего это компиляция исходного кода в объектные файлы и последующая компоновка в исполняемые файлы или библиотеки.

До создания make системы сборки (компиляции) ПО Unix обычно состояли из shell-скриптов сборки, сопровождавших исходный код программ. *make* была создана Стюартом Фельдманом (*Stuart Feldman*) в 1977 году в Bell Labs.

Make

```
make [ -f make-файл ] [ цель ] ...
```

Файл ищется в текущем каталоге. Если ключ `-f` не указан, используется имя по умолчанию для make-файла — `Makefile`. `make` открывает make-файл, считывает правила и выполняет команды, необходимые для создания указанной цели. Стандартные цели для сборки дистрибутивов GNU:

- `all` — выполнить сборку пакета;
- `install` — установить пакет из дистрибутива (производит копирование исполняемых файлов, библиотек и документации в системные каталоги);
- `uninstall` — удалить пакет (производит удаление исполняемых файлов и библиотек из системных каталогов);
- `clean` — очистить дистрибутив (удалить из дистрибутива объектные и исполняемые файлы, созданные в процессе компиляции);
- `distclean` — очистить все созданные при компиляции файлы и все вспомогательные файлы, созданные утилитой `./configure` в процессе настройки параметров компиляции дистрибутива.

Make

Программа make выполняет команды согласно правилам, указанным в специальном файле. Этот файл называется make-файл (makefile, мейкфайл). Как правило, make-файл описывает, каким образом нужно компилировать и компоновать программу. make-файл состоит из правил и переменных. Правила имеют следующий синтаксис:

цель1 цель2 ...: рекуизит1 рекуизит2 ...

команда1

команда2

```
sent"/>  
fish.web.present
```

```
<!-- do not for
```

```
oot}" else="$}{gfv
```

```
app.context-root]
```

```
resent">
```

```
b}"/>
```

Make

Для ускорения длительных задач используют распараллеливание вычислений внутри make-программы. Распараллеливание делается автоматически make-интерпретатором. Для запуска make-файла в многопоточном режиме необходимо передать опцию `-j <число потоков>`.

`make [-j <число потоков>] [цель]`

```
sent"/>  
fish.web.present
```

```
<!-- do not forg
```

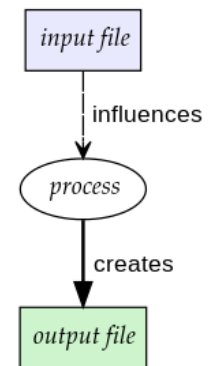
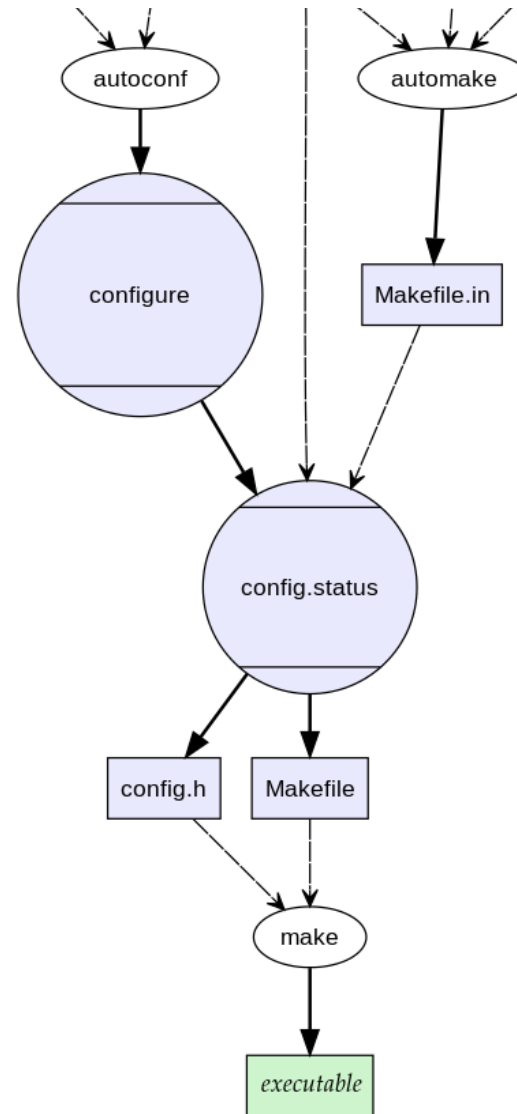
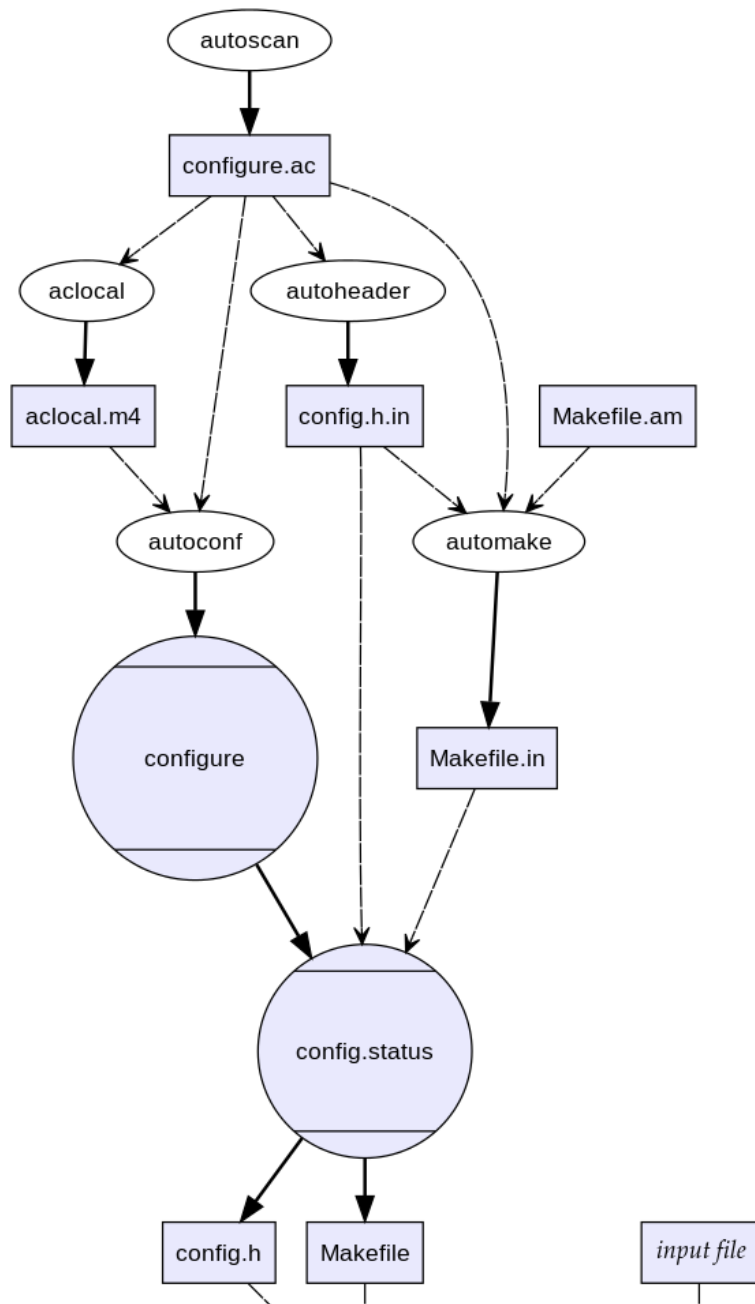
```
oot}" else="$ {gfv
```

```
app.context-root]
```

```
resent">
```

```
b]"/>
```

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36



sent
forg

{gfv
root}

b)"/>

ИСТОЧНИКИ

- https://ru.wikipedia.org/wiki/%CF%E0%EA%E5%F2%ED%FB%E9_%F4%E0%E9%EB
- <https://ru.wikipedia.org/wiki/Bash>
- https://ru.wikipedia.org/wiki/Windows_PowerShell
- <https://ru.wikipedia.org/wiki/Autotools>
- <http://www.gnu.org/software/autoconf/>
- <http://www.gnu.org/software/automake/>
- <http://squadette.ru/autotools-ru/>

```
sent"/>  
fish.web.present
```

```
<!-- do not forg
```

```
oot}" else="$gfv
```

```
app.context-root
```

```
resent">
```

```
b]"/>
```