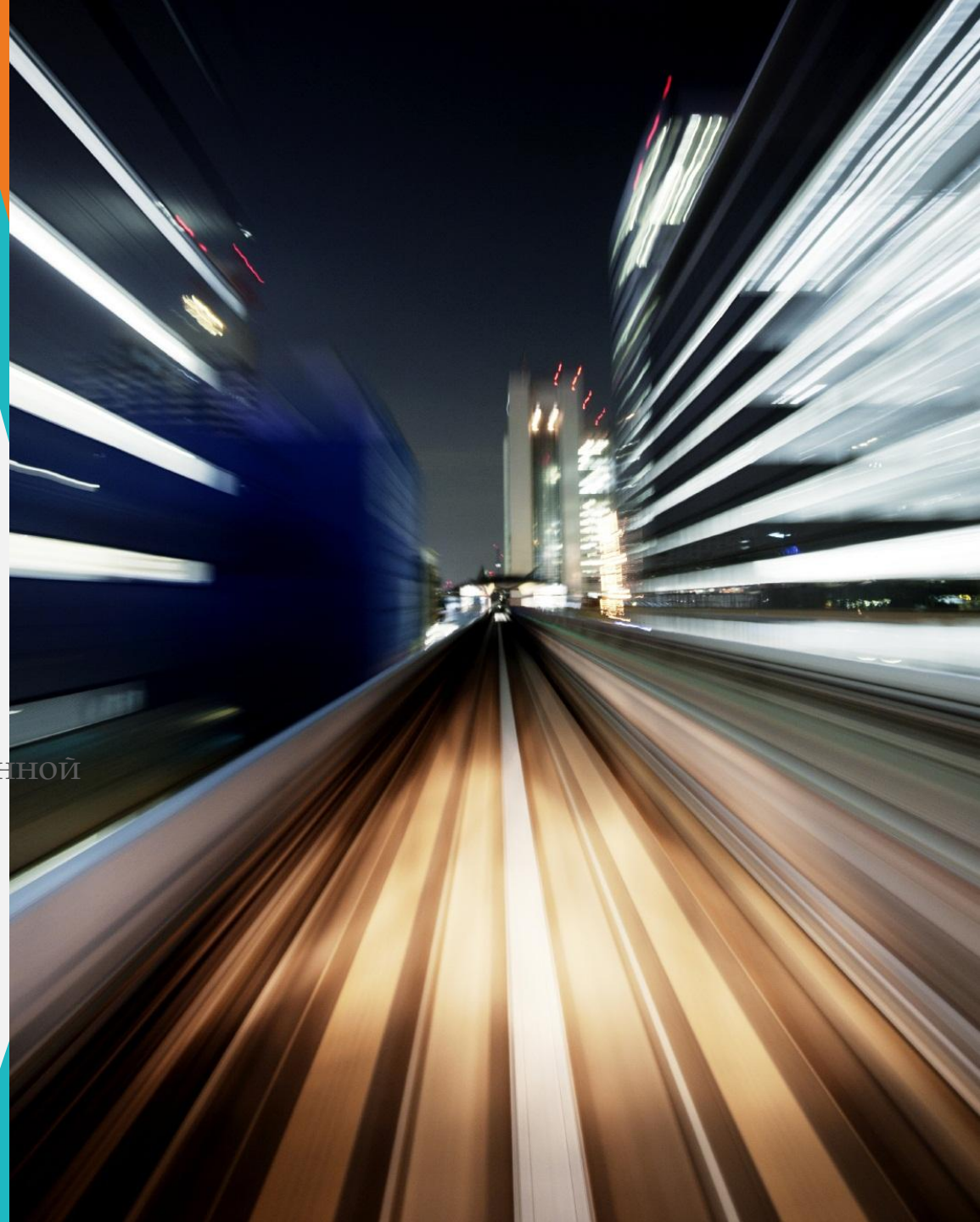
Задания

Доцент кафедры информационной безопасности, к.т.н.
Прилуцкий А.М.

Безопасность операционных систем (Безопасность ОС)

Специальность– 10.05.03 – Информационная безопасность автоматизированных систем



Вопросы по Лекции *

1. ***

Вопросы лекции

1. Язык управления заданиями
2. Пакетная обработка
3. Задания в среде UNIX

Язык управления заданиями

Командный интерпретатор

Командный интерпретатор — программа, работающая в течение всего сеанса работы пользователя с ОС. Ее основная функция — выполнение команд пользователя, записанных на языке данного командного интерпретатора и выполнение этих команд либо непосредственно (встроенными в интерпретатор средствами), либо путем вызова других программ.

Основным способом взаимодействия командного интерпретатора с пользователем является интерфейс командной строки. При таком способе взаимодействия в качестве основного устройства для работы с системой используется терминал. Терминал служит для отображения информации и ввода информации пользователем. Физически терминал — это монитор и клавиатура. Логически с точки зрения ОС терминал — набор из двух файлов. Один из этих файлов служит для ввода информации (которая поступает с клавиатуры), другой — для вывода информации (которая выводится на экран).

Командный интерпретатор

Сигналом для начала работы служит выводимое на экран приглашение командной строки или просто приглашение — последовательность символов, указывающая на то, что интерпретатор ожидает ввода команды. Типичное приглашение в Unix-системах имеет следующий вид: **$**

Ввод команды завершается нажатием клавиши Enter, после чего интерпретатор начинает выполнение команды.

Например, можно вывести имя файла, соответствующего терминалу. Это осуществляется при помощи команды tty:

```
$ tty <Enter>
/dev/console
```

Здесь /dev/console — имя файла, который используется для вывода данных и считывания ввода пользователя.

Язык управления заданиями

Язык управления заданиями должен иметь средства:

- определения последовательности выполнения программ в задании и средства определения используемых ресурсов;
- определения типа выполнения программ (основной режим, фоновый режим, определение приоритета и т. п.);
- определения условий выполнения частей задания и ветвения задания;
- проверки состояния ресурсов ОС (файлов и процессов) и их свойств (например, прав доступа).

Пакетная обработка

Режимы выполнения заданий

Работа с заданиями может выполняться в двух режимах: диалоговом или пакетной обработки.

Работа в **Диалоговом режиме** подразумевает постоянный диалог с пользователем во время выполнения задания. В диалоге с пользователем по мере необходимости определяются параметры выполнения команд и программ, входящих в задание.

В пакетном режиме все данные и параметры, необходимые для выполнения задания, готовятся до начала выполнения задания в виде пакетного файла. Само задание выполняется без участия оператора, а все диагностические сообщения и сообщения об ошибках накапливаются в файлах протокола выполнения. Диагностические сообщения в пакетном режиме также могут выводиться на терминал по мере их появления, но при этом пользователю после завершения выполнения задания будут доступны только последние сообщения, которые были выведены на экран. Если количество диагностических сообщений превышает количество строк терминала, то первые сообщения будут потеряны.

Передача параметров пакетному заданию

Для передачи заданию данных до выполнения используются параметры задания. Если задание обрабатывается командным интерпретатором, предоставляющим интерфейс командной строки, то параметры задания определяются как параметры командной строки. Командная строка представляет собой текстовую строку, в которой указано имя вызываемого задания и передаваемые ему параметры. Все параметры отделяются друг от друга символами-разделителями, например пробелами. Параметры — строки, которые могут представлять собой числовые константы, последовательности символов, имена файлов и устройств и т. п. Интерпретация параметров позиционная — параметры различаются по разделителям и поэтому нельзя определить третий параметр, не определив первый и второй. Передаваемые параметры, вообще говоря, не должны содержать символов-разделителей и различных управляющих кодов. Однако возможно использование пробела в тексте параметра, для этого его необходимо заключить в двойные кавычки

Результат выполнения задания

Результат выполнения задания заключается в изменении информационного окружения задания — содержимого и состояния файлов и каталогов, запуске или останове других заданий и программ пользователя.
Для облегчения автоматической обработки результата при завершении задания формируется код возврата — числовое значение, характеризующее успешность выполнения. Конкретные значения кодов возврата определяются в тексте задания. Доступ к коду возврата можно получить непосредственно после завершения задания при помощи специальных средств командного интерпретатора.

Задания в среде UNIX

Командный интерпретатор BASH

Синтаксис языка управления заданиями определяется используемым командным интерпретатором. В настоящем издании в качестве базового используется командный интерпретатор BASH (Bourne-Again Shell). При работе в диалоговом режиме команды BASH вводятся с клавиатуры в ответ на приглашение командной строки.

Задания, оформляемые в виде файлов, состоят из двух частей — заголовка, определяющего имя командного интерпретатора и путь к нему, и собственно текста задания.

Заголовок начинается с первого символа первой строки файла задания и деля интерпретатора BASH выглядит обычно следующим образом:

! /bin/bash

Здесь # — символ комментария. Все символы, которые находятся на строке после символа и не воспринимаются интерпретатором как команды, а также все, что находится после этого символа, игнорируется при выполнении задания.

Заголовок является комментарием особого вида за счет того, что сразу за символом помещен символ ! . Конструкция ! будучи помещенной в начало файла задания, указывает на то, что после нее записывается полное имя исполняемого файла командного интерпретатора.

Командный интерпретатор BASH

После заголовка следует основная часть скрипта — последовательность команд. Одна строка скрипта может содержать одну или более команд, комментарии или быть пустой. Как правило, одна строка скрипта содержит одну команду, при большем количестве команд на одной строке они разделяются точкой с запятой.

Синтаксически вызов большинства команд интерпретатора BASH состоит из двух частей:

<имя команды> «параметры>

В качестве имени команды может выступать либо внутренняя команда BASH, либо имя файла, содержащего код программы или текст задания.

Если команда и ее параметры слишком длинны, то можно, воспользоваться символом переноса «\». После указания этого символа можно продолжить запись команды на следующей строке, а командный интерпретатор будет воспринимать все, что записано после символа переноса, как продолжение команды.

Переменные

Типы переменных. При написании заданий существует возможность определения и использования переменных (аналогично тому, как это делается при программировании на языках высокого уровня). Существует два типа переменных: **внутренние переменные и переменные окружения** — т. е. переменные, заданные в специальной области ОС и доступные всем выполняемым программам. Переменные окружения — часть того информационного окружения, которое влияет на выполнение программ пользователя.

Присвоение значений переменным осуществляется при помощи конструкций:

<имя переменной>=<значение>

или

let <имя переменной>=<значение>

При присвоении значения нужно обратить внимание на то, что пробелы между именем переменной и знаком равенства, а также между знаком равенства и значением не допускаются.

Переменные

Чтобы при выполнении задания можно было получить доступ к значению переменной, необходимо воспользоваться операцией подстановки $. При прямом указании в тексте задания имени переменной это имя будет воспринято как строка. Если имя предварить операцией подстановки, то при выполнении задания будет произведена подстановка значения переменной с указанным после символа $ именем.

```
# ! /bin/bash
variable=Hello
echo variable
echo $variable
```

После выполнения такого задания на экран будет выведено

```
variable
Hello
```

Учитывая все сказанное выше, нетрудно догадаться, что присвоение значения одной переменной другой переменной будет записано как var1=$var2

Системные переменные

Системные переменные имеют стандартные имена и фиксированную трактовку, при этом их значения присваиваются самим командным интерпретатором, пользователь может только считать значения встроенных переменных, но не может явно изменить их значение.

В BASH определены следующие встроенные переменные:

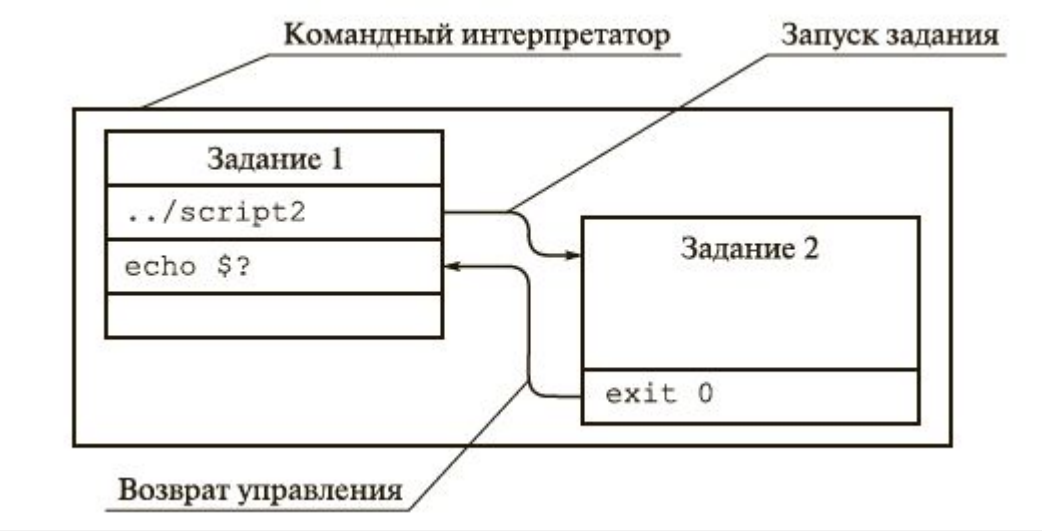
1) `$?` — код возврата последней команды. Возврат из скрипта производится командой `exit` с указанием кода возврата (`exit <код возврата>`). Код возврата может служить для управления выполнением задания, например, при последовательном поиске строк в файле. В зависимости от кода возврата программы поиска подстроки в файле `grep` можно либо выдать сообщение об ошибке, если нужная строка не найдена, либо продолжить поиск следующей строки;

2) `$#` — число параметров командной строки вызова задания;

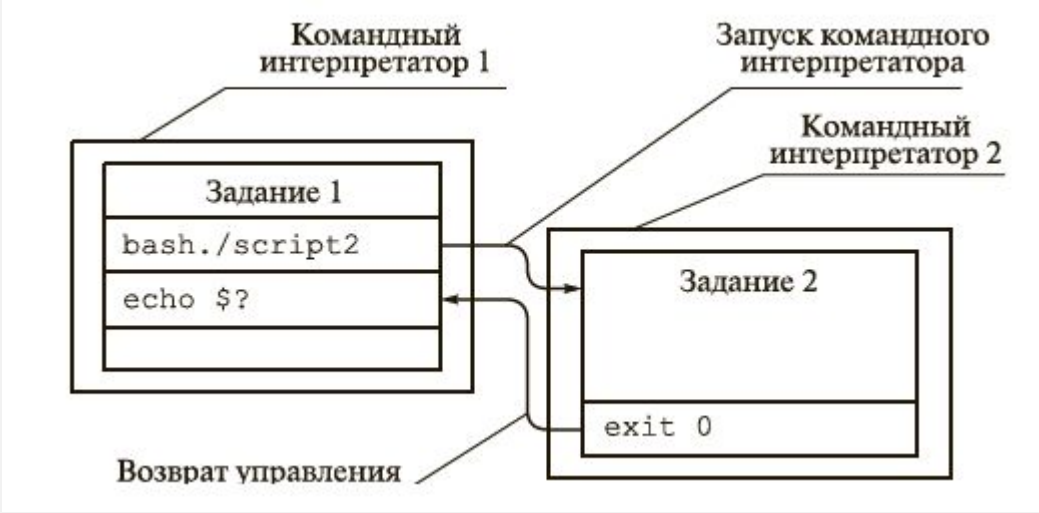
3) `$1,…,$9` — при помощи этих переменных производится подстановка значений параметров, указанных в командной строке, при помощи которой было вызвано задание. Переменной `$1` соответствует первый параметр, переменной `$9` — девятый.

Запуск задания на исполнение

Для запуска задание на исполнение, его нужно передать командному интерпретатору, который будет анализировать текст задания и трансформировать команды задания в системные вызовы ОС. При этом командный интерпретатор будет поддерживать информационное окружение задания, т. е. хранить все временные данные, необходимые для выполнения задания, например, привязки к текущему терминалу, локальные переменные, файловые дескрипторы.



Запуск задания в контексте текущего командного интерпретатора



Запуск задания в новом командном интерпретаторе

Запуск задания на исполнение

Задание может быть запущено на исполнение как непосредственно пользователем из командной строки, так и из другого задания. Во втором случае имя запускаемого задания представляет собой команду в тексте запускающего задания. Условимся при этом называть запускающее задание родительским по отношению, к запускаемому - Дочернему. Аналогично в случае создания нового командного интерпретатора: создающий интерпретатор будем называть родительским по отношению к порождаемому - дочернему.
Возможны следующие варианты запуска задания:

- путем указания имени файла задания (возможно с полным или относительным путем к файлу):

  /check/ scripts /teacher /gather . Sh

Чтобы воспользоваться этим методом запуска, файл с заданием должен иметь атрибут «исполнимый». При таком запуске задания на исполнение запускается новая копия командного интерпретатора, путь к которому указан в заголовке задания. После завершения задания управление возвращается родительскому командному интерпретатору. В случае если задание было запущено из другого задания, родительское задание продолжает свое выполнение со следующей команды. Если задание было запущено пользователем в командной строке, происходит возврат к приглашению командной строки родительского интерпретатора.

Запуск задания на исполнение

- путем запуска командного интерпретатора с указанием имени файла задания в качестве параметра:

/bin/bash /check/ scripts/teacher/gather . sh

При таком запуске файл с заданием необязательно должен иметь атрибут «исполнимый». Также допускается отсутствие заголовка в тексте задания — выбор командного интерпретатора осуществляется пользователем, запускающим задание на выполнение, поэтому при такой форме запуска заголовок игнорируется.

В отличие от предыдущего метода запуска при отсутствии абсолютного или относительного пути к файлу задания его поиск осуществляется в текущем каталоге. Это связано с тем, что в данном случае имя файла передается в качестве параметра командному интерпретатору, а не используется как имя исполняемого файла. В остальном метод запуска аналогичен предыдущему.

Запуск задания на исполнение

- путем запуска при помощи команды exec:

exec /check/scripts/teacher/gather.sh

При данном запуске управление передается дочернему командному интерпретатору безвозвратно — родительский командный интерпретатор полностью заменяется дочерним, который выполняет запущенное задание. Возврата к выполнению родительского С, задания после завершения дочернего не происходит. Файл задания при такой форме запуска должен иметь атрибут «исполнимый»;

- путем запуска в том же командном интерпретаторе:

/check/ scripts /env.sh

Если перед именем запускаемого задания указать точку, отделенную от имени пробелом, то выполнение задания продолжится уже запущенной копией командного интерпретатора, при этом для нового задания будет использоваться та же самая среда времени выполнения.

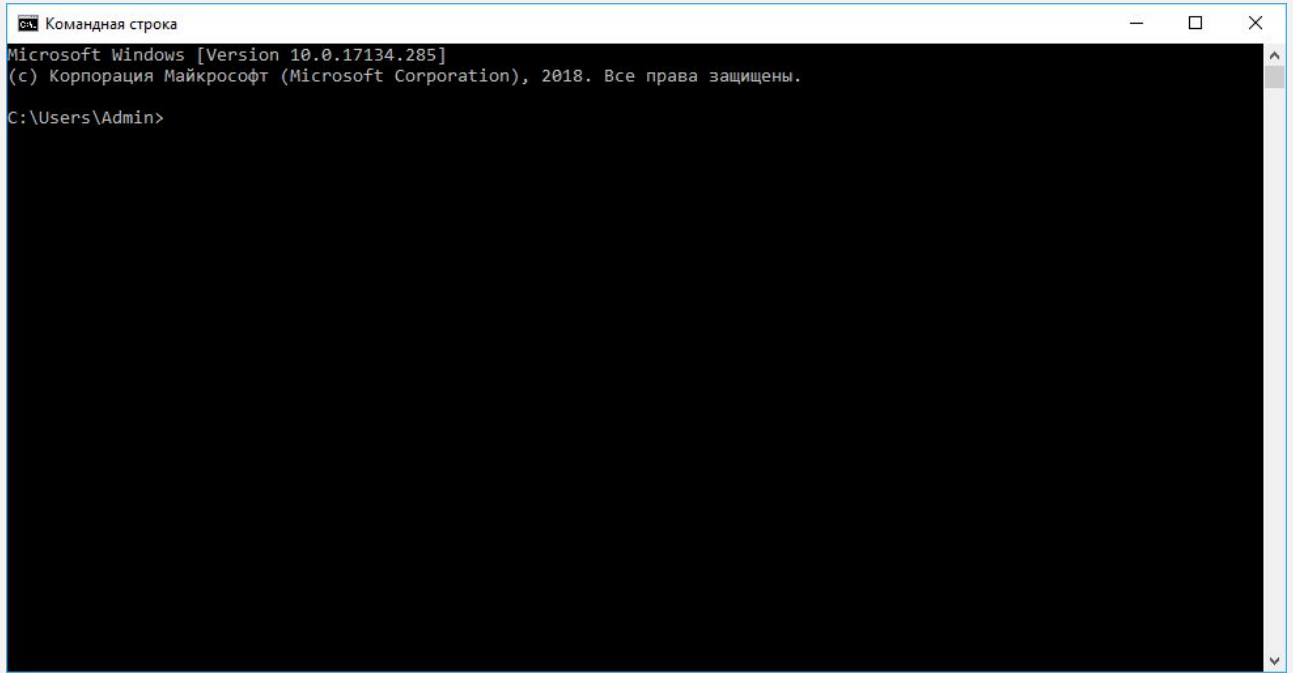
При таком запуске новое задание получит в свое распоряжение те же самые переменные, что и вызвавшее его задание — при сохранении копии командного интерпретатора сохраняется и его внутреннее состояние, в том числе и переменные. Все переменные, объявленные и инициализированные в вызванном задании, будут иметь те же значения и после его завершения, когда управление будет передано вызвавшему заданию.

Задания в Windows

Командный интерпретатор в Windows

Для запуска командного интерпретатора в ОС Windows, основанных на ядре Windows NT, используется программа cmd.exe. Для запуска интерфейса командной строки необходимо открыть меню Start (Пуск), выбрать пункт меню Run (выполнить) и запустить программу cmd.exe.

После запуска командного интерпретатора появляется окно, содержащее стандартное приглашение к работе.



Пакетная обработка в Windows

Задания также оформляются в виде текстовых файлов, содержащих перечень команд, которые должны быть выполнены командным интерпретатором. Сами файлы должны иметь расширения .bat или . cmd. А исполнение, механизмы передачи параметров и результат работы командных скриптов в Windows такие же, что и в Linux.

В качестве имени команды может использоваться либо внутренняя команда командного интерпретатора, либо имя исполняемого файла, содержащего код внешней программы.

Переменные

При создании командных сценариев пользователь имеет возможность оперировать переменными окружения. Переменные могут поступать от нескольких источников. В соответствии с источником определения той или иной переменной окружения их можно подразделить на встроенные системные, встроенные пользовательские и локальные.

Встроенные системные переменные определяются на уровне ОС и доступны всем процессам независимо от пользователя. Встроенные пользовательские переменные определяются на этапе входа пользователя в систему и существуют все время, пока продолжается сеанс работы данного пользователя. Для получения списка всех переменных окружения (как системы, так и пользователя) и их текущего значения используется команда **set**.

Спасибо за внимание

Вопросы