Общие замечания

Взять задание – значит отметить на странице https://unix2016.titanpad.com/1 свою фамилию и номера задач, например “Пупкин: A2, B7”. Имейте в виду, что в графе “задание берёт” данные отобразятся не сразу. (Затирать чужие назначения не пытайтесь, в TitanPad есть история ω .)

Конкурсное задание общее для всех, но баллы получит только победитель конкурса. В таблице будет отображаться победитель на текущий момент.

Можно предполагать, что на машине, на которой будут запускаться ваши скрипты, установлены Python 2.7, Perl 5, wget, curl и GNU Make в базовой конфигурации. В более сложных заданиях может потребоваться установка дополнительных пакетов, в этом случае требуется сопроводить решение их списком и инструкциями по их установке. Однако рекомендуется стараться не прибегать к использованию perl/python там, где достаточно средств оболочки и стандартных программ, а также не требовать установки дополнительных пакетов без обоснования.

Решения присылаются через http://www.anytask.org/ в задачи 2X, 2A и 2B. Короткие решения простых задач можно присылать в явном виде, для средних и длинных стоит использовать системы Code Review, чтобы вы могли постепенно улучшать решения в ответ на поступающие замечания (если у вас нет своего предпочтения такой системы, используйте встроенный в AnyTask Review Board).

Скрипты, в задании к которым не требуется оптимизация размера кода, стоит сопровождать справкой (usage) при вызове с параметрами -h, --help или при несоответствии параметров вызова спецификации (например, если ожидается один позиционный параметр, а вызов производится без параметров).

Сообщения об ошибках следует выводить в стандартный поток ошибок.

В тех заданиях, которые предполагается реализовывать на bash, дополнительный **1 балл** за простые и **2 балла** за сложные вы получаете в случае, если предоставляете (дополнительно) версию, корректно работающую под /bin/sh (Amquist shell). (*Дополнительно* – значит, что если версия с использованием bash-специфики оказывается короче и/или выразительнее, предоставляются обе версии.) Это следует отразить соответствующим образом в shebang. Советы по устранению “башизмов” находятся здесь.

Желательно взять одно простое и одно сложное задание. Заявки сверх этого будут приниматься лишь при условии, что никто другой не претендует на задание.

Для отладки полезно установить себе в начале скрипта set -xue -o pipefail.

Здесь и здесь перечислены некоторые ошибки начинающих, которых стоит избегать.

Автоматически проверить решение на наличие стандартных ошибок можно с помощью ShellCheck. Если на вашем решении этот инструмент выдаёт предупреждения, нужно уметь исправить их или обосновать ошибочность этих предупреждений. Для того, чтобы понять смысл строки шелл-скрипта, можно пользоваться замечательным ресурсом ExplainShell.

Короткое решение (если есть) присылается в дополнение к полному, т.е. понятному, документированному и поддерживающему опцию -h/--help.

Для укорачивания кода может оказаться полезной информация отсюда. Даже короткие решения должны корректно отрабатывать не только при первом запуске.

X. Конкурсное задание

3 балла получит тот, кто к занятию 06.10 первым пришлёт самый короткий “однострочник”, решающий следующую задачу. На вход поступают строки. Если строка не начинается с http:// или https://, следует добавить префикс http:// к её началу. Полученная (или исходная, если у неё уже был нужный префикс) строка выводится в стандартный поток. Никаких предположений о содержимом строк не делается. В отличие от остальных заданий, это могут брать все желающие, но баллы получит один.

A. Простые задания

1. **2 балла**. Напишите скрипт trans, принимающий один обязательный и один опциональный параметр. Обязательный параметр - слово или фраза, опциональный – направление перевода в формате вроде “ru-en” (с русского на английский, по умолчанию) или “en” (перевести на английский, язык оригинала определить автоматически). Скрипт использует какой-нибудь публичный HTTP API перевода и печатает одну строку - результат перевода (важно: без служебной информации API, т.е. “trans word ru” должно возвращать “слово” без кавычек).
   Задание берёт: Моргачев
2. **2 балла**. Напишите скрипт mypwd, который печатает имя текущей директории, в котором при условии, что текущий пользователь имеет имя user, часть “/home/user” отображается как “~”, а часть /home/otheruser отображается как “~otheruser”. При этом в корне домашней директории должно отображаться “~/”, а в корне домашней директории другого пользователя – “~otheruser/”. **+1 балл:** Добавьте опцию -i/--install, которая дожна вызываться так: eval $(mypwd --install) – и заменять оператор печати имени директории в $PS1 на вызов mypwd.
   Задание берёт: Блажукас
3. **2 балла.** Напишите на awk скрипт numheader, который читает только первую строку входного потока, состоящую из разделённых знаками табуляции названий колонок, и выводит столько строк, сколько обнаруживает названий. В каждой строке соответствующее название предваряется номером колонки (полезно для дальнейшего использования команды cut). Например, “col1\tcol2” превратится в “1\tcol1\n2\tcol2”. Справку по использованию (-h/--help) реализовывать необязательно (**+1 балл**).
   Задание берёт: Прудников
4. **2 балла**. Напишите скрипт tofifo, который создаёт (в директории со временными файлами) временный FIFO-канал, печатает его название, пишет в него содержимое стандартного входа, а затем удаляет канал. Назначение: упрощение использования FIFO при отладке gdb программ, ожидающих данных от другой

программы на вход: запускаем программу-генератор данных в фоновом режиме через prog | tofifo&, а в gdb прописываем в args перенаправление ввода из напечатанного при этом имени файла.

Задание берёт:

5. **3 балла.** Напишите скрипт ask, принимающий один обязательный аргумент (приглашение) и один необязательный (Y или N - значение по умолчанию). Скрипт выводит приглашение и [y/n]? – причём при указании значения по умолчанию соответствующая буква выводится в верхнем регистре – и ожидает ввода пользователя. До тех пор, пока ввод пользователя не будет начинаться с y, Y, n или N (а при указании значения по умолчанию допускается пустая строка, которая интерпретируется соответствующим образом), приглашение ко вводу повторяется. Код возврата скрипта – признак того, что пользователь согласился с содержанием приглашения. Т.е. ваш скрипт может использоваться в виде if ask ‘Do you want to continue?’ Y; then rm -fr /; fi. **+1 балл**: даже если стандартный ввод перенаправлен, ввод ответа пользователя происходит с терминала.

   Задание берёт: Назарук, Стабровский, Абрамов

6. **3 балла**. Напишите скрипт fat, по умолчанию выводящий в человекочитаемом формате список 10 файлов и папок в текущем каталоге, занимающих наибольший объём дискового пространства, и их совокупный объём. **+1 балл**: Скрипт принимает дополнительно необязательные аргументы “сколько самых больших файлов и папок пропустить” и “сколько пунктов вывести”. Пример:

```
🔊 fat 2 15
656K    gui
348K    bin
196K    include
120K    fonts
28K     .vimrc
28K     .ssh
28K     screen
28K     bcvi_plugins
28K     .bashrc
16K     .subversion_config
12K     .ycm_extra_conf.pyc
12K     .ycm_extra_conf.py
12K     .tmux.conf
12K     .screenrc
12K     README
1.5M    total
```

   Задание берёт: Кривулец, Казачёнок, Шпаковский

7. **2 балла**. Напишите скрипт fresh, оставляющий по умолчанию самый новый файл в текущем каталоге, а остальные предлагающий удалить (режим -i/--interactive команды rm). **+1 балл**: команда принимает дополнительный аргумент “число оставляемых файлов”. Пример:

```
🔊  fresh 3
Leaving: Life Dropbox Desktop
rm: remove directory ‘_’? n
rm: descend into directory ‘Documents’? n
rm: remove symbolic link ‘YandexDisk’? n
```

```
rm: descend into directory ‘Downloads’? n
rm: descend into directory ‘Work’? n
```

Задание берёт: Борисевич, Пилянский

8. **3 балла** (идея Алексея Колесова). Напишите скрипт before-grep, который принимает на вход регулярное выражение и (опционально) список файлов (в случае его отсутствия читает стандартный ввод) и выводит строки, непосредственно предшествующие содержащим регулярное выражение. Пример:

```
$ cat input.txt
abc def
zozozozo
zabc def
abc def
abc def
noooo
yess
tabc defz
$ ./before-grep 'abc def' input.txt
zozozozo
zabc def
abc def
yess
```

Задание берёт: Наливайко, Дужик

9. **3 балла** (идея Алексея Колесова). Напишите скрипт watchdog, принимающий единственный аргумент – идентификатор пользователя vk.com (например, id1 или durov), и отображающий системную нотификацию (notify-send(1)) в случае присутствия соответствующего человека онлайн.

Задание берёт: Скоклеенко, Блинов, Коновод

10. **3 балла** (идея Алексея Колесова). Напишите скрипт difference, принимающий в качестве аргументов имена двух файлов, и выводящий строки первого файла, отсутствующие во втором. **+1 балл**: относительный порядок строк первого файла сохраняется.

Задание берёт: Касияник, Лученок, Поляков

11. **3 балла** (идея Алексея Колесова). Напишите скрипт map, принимающий позиционный аргумент – имя файла – и именованный аргумент -c/--cmd – имя команды (по умолчанию echo). Скрипт должен выполнять указанную команду, передавая ей в качестве параметра по очереди каждую строку указанного файла. **+1 балл**: режим -i/--input, при котором вместо аргументов строка подаётся на стандартный вход. **+1 балл**: опция -t/--threads, которая указывает, сколько копий команды запускать параллельно (по умолчанию 1).

Задание берёт: Демченко, Мицкевич, Климук

12. **3 балла** (идея Алексея Ропана). Напишите скрипт paster, принимающий либо в виде аргументов, либо на стандартный вход (по выбору пользователя скрипта – программист должен реализовать оба варианта) текстовый сниппет и сохраняющий их на сервис https://paste.ee/. Программа печатает адрес страницы со сниппетом, возвращённый сервисом, а также открывает её в браузере по умолчанию. **+1 балл**: тип файла для подсветки синтаксиса определяется из расширения первого аргумента (нужно поддержать по крайней мере типы из

раздела Common + perl, css и html5). **+2 балла**: при невозможности определения типа по расширению тип определяется по shebang (с учётом возможности расположения интерпретатора в произвольном месте, указания дополнительной версии и опций т.е. корректно должен обрабатываться shebang вида #!/usr/bin/env python2.7 или #!/usr/bin/awk -F).

Задание берёт: Каганович

13. **3 балла.** Напишите скрипт locator.sh, который выводит абсолютный путь к директории, находящейся на один уровень выше фактического расположения скрипта (даже если скрипт был вызван через символическую ссылку). Такая задача возникает, когда необходимо обратиться из скрипта к внешнему ресурсу, расположенному по фиксированному относительному пути по отношению к скрипту.

    Задание берёт: Комар, Сатаневский

14. *(здесь может быть ваше задание)*

B. Сложные задания

1. **10 баллов**. Напишите скрипт percread.sh, экспортирующий две функции – percread и watchfile. Первая принимает один обязательный аргумент (регулярное выражение grep) и один необязательный (интервал в секундах, допускаются дробные значения). Программа руководствуется выводом команды lsof(8) или аналогичной для определения того, сколько процентов файла, соответствующего регулярному выражению, прочитано (т.е. какую долю составляет fseek-позиция в файле от его размера). Если регулярному выражению соответствует несколько файлов или ни одного, программа выходит с соответствующим сообщением об ошибке. Проценты выводятся с точностью до двух знаков после десятичного разделителя. Если задан второй параметр, программа выводит данные с указанным интервалом до тех пор, пока файл не будет закрыт. **+1 балл**: вывод программы во втором режиме происходит в пределах одной строки с затиранием предыдущего вывода. watchfile – синоним для percread с заданным по умолчанию вторым параметром.

    Задание берёт: Каганович

2. **10 баллов**. Напишите скрипт showtime.sh, предназначенный для включения в .bashrc. Скрипт настраивает bash таким образом, чтобы после выполнения каждой команды, за время работы которой изменился вывод команды date +%s, выводился временной диапазон и продолжительность (примечание: не стоит пытаться сделать /bin/sh-совместимую реализацию, вместо этого за **2 балла** можно предоставить реализацию для zsh). **+1 балл**: цветной вывод. Иллюстрация:

```
~/> sleep .01 # short operation like cd, mv, ls...
~/> sleep 200 # long operation - a super-duper calculating process
[14:30:13 - 14:33:34 (3 min 21 sec)]
~/>
```

    Задание берёт: Демченко, Абрамов

3. **10 баллов**. Выберите важную для вас директорию (например, $HOME) и однобуквенную аббревиатуру для неё (например, h). Создайте команды cdh (принимает опциональный аргумент – поддиректорию), mvh (принимает обязательный аргумент – источник – и необязательный – приёмник относительно выбранной директории), cph (параметры аналогичны mvh), mvfh (первый – обязательный – параметр – источник относительно выбранной директории, второй – по умолчанию . – приёмник относительно текущей) и cpfh (аналогично mvfh). Сделайте поддержку автодополнения созданных команд. **+1 балл**: если приёмник mv*/cp* содержит несуществующую директорию, она создаётся.
   Задание берёт: Прудников, Мицкевич
4. **10 баллов**. Напишите скрипт bookmarks.sh, экспортирующий команды s (save), g (go), p (print), d (delete) и l (list). Первые четыре принимают имя “закладки”, и соответственно: сохраняют текущую директорию в качестве закладки; переходят в директорию, соответствующую закладке; печатают и удаляют закладку. Последняя перечисляет существующие закладки. Закладки должны сохраняться между сессиями командной оболочки. **+1 балл**: содержимое закладок экспортируется в переменные вида $DIR_<имя>. **+1 балл**: цветной вывод команд p и l. **+1 балл**: автодополнение команд g, p, d, l.
   Задание берёт: Скоклеенко, Коновод
5. **8 баллов** (идея Алексея Колесова). Напишите скрипт env-editor, экспортирующий функцию, открывающую в редакторе по умолчанию пользователя (${EDITOR:-vi}) временный файл, строками которого являются элементы переменной окружения $PATH (которые в переменной разделялись двоеточием). После закрытия файла значение переменной $PATH устанавливается соответствующим образом. **+1 балл**: имя редактируемой переменной устанавливается параметром командной строки.
   Задание берёт: Пилянский, Шпаковский, Комар
6. **10 баллов**. Напишите скрипт blind-typing.sh со следующей функциональностью. При запуске скрипт произносит (через синтезатор голоса): “Type the phrases you hear. Type a single question mark character to repeat, empty line to finish”. Далее, пока пользователь не ввёл пустую строку, выбирается одна случайная строка из заранее заготовленного текстового файла и произносится. Если пользователь вводит “?”, строка произносится повторно. Задача пользователя – ввести услышанный текст. Сравнение текста должно производиться без учёта регистра, пунктуации и повторяющихся пробельных символов. Если пользователь справляется, скрипт произносит одобряющую фразу. В противном случае произносится сообщение об ошибке и выводятся строки “Expected: <ожидаемый текст>“ и “Typed: <введённый текст>”. Эти же строки сохраняются в конце файла errors.log. **+1 балл:** в конце сессии выводится процент ошибок. **+1 балл**: вводимый текст не виден пользователю. **+1 балл**: клавиша Backspace не работает, при её нажатии произносится фраза “Backspace is ignored”. **+1 балл**: если текстовый файл с фразами отсутствует, программа заготавливает его из интернет-источника.
   Задание берёт: Моргачев, Блинов

7. **7 баллов**. Напишите логику для .bashrc, которая перед показом каждой командной строки проверяет, изменилась ли дата модификации файла ~/.bashrc, и если да, то читает его.
   Задание берёт: Борисевич
8. **7 баллов**. Напишите функции updallfiles (ищет файлы в текущей директории и складывает их список в файл $HOME/.allfiles), updalldirs (то же с директориями и $HOME/.alldirs), where (перечисляет с использованием списка файлов подходящие под регулярное выражение с подсветкой) и wtf ("what the file", то же для файлов). Две последние функции должны сообщать пользователю о необходимости сначала запустить upd*, если соответствующего файла нет.
   Примечание: это учебный пример. В жизни стоит использовать locate.
   Задание берёт: Лученок
9. **10 баллов**. Напишите скрипт critical-path.sh, который принимает из стандартного потока ввода описание графа задач из строк вида
   ```
   task 10 other1 other2 other3
   ```
   которые означают "задача с именем task выполняется 10 единиц времени и требует перед своим началом выполнения задач other1, other2 и other3". Требуется вычислить минимальное время выполнения задачи с именем job с учётом всех зависимостей при условии максимального распараллеливания (гарантируется наличие описания job во входе; также гарантируется, что описана каждая из задач, указанных в качестве зависимости, и описания задач не повторяются). Названия задач состоят из маленьких латинских букв и цифр, длительности – целые числа, в сумме не превышающие 1 млн. Если граф задач содержит циклическую зависимость, следует выйти с сообщением "Cyclic dependency". Если несколько задач зависят от одной, последняя всё равно выполняется один раз.
   Пример:
   ```
   $ cat input.txt
   main 4 do1 do2
   do1 3 sleep
   sleep 10
   do2 1
   job 2 main
   $ /bin/sh critical-path.sh < input.txt
   19
   ```
   Подсказка: можно использовать make.
   +5 баллов за решение (без справки и shebang), работающее под управлением /bin/sh, размером не более 300 байт (у нас с Алексеем Ропаном получилось 195).
   Задание берёт: Кривулец
10. **9 баллов**. Скрипт total-time.sh – аналогично заданию 9, но вычислить требуется суммарную длительность выполняемых задач (следует учитывать, что некоторые может не требоваться выполнять для выполнения job). Например:
   ```
   $ cat input.txt
   job 1 task1 task2
   task2 3
   task1 4 subtask
   subtask 1
   ```

```
sleep 39
idle 8
play 3
$ /bin/sh total-time.sh < input.txt
9
```

+5 баллов за решение (без справки и shebang), работающее под управлением /bin/sh, размером не более 280 байт (у нас получилось 120).

Задание берёт: Блажукас, Стабровский

11. **10 баллов**. Напишите скрипт path.sh, который читает со стандартного ввода описание неориентированного графа вида

```
from 1 2 3
1 from
2 from 4
3 from 4
4 5 2 3
5 4 to
to 5
```

где в каждой строке описана вершина и её окружение (гарантируется, что описание корректно и что в нём присутствуют вершины from и to, названия вершин состоят из маленьких латинских букв и цифр). Требуется определить, есть ли в графе маршрут, соединяющий вершины from и to, и вывести соответственно Yes или No. **+5 баллов**: решение (без справки и shebang), работающее под управлением /bin/sh, размером не более 160 байт (у нас с Алексеем Ропаном получилось 99).

Задание берёт: Касияник, Казачёнок

12. **10 баллов**. Скрипт total-time-dup.sh – аналогично заданию 10, но каждая задача выполняется столько раз, сколько раз она указана в зависимостях для других задач. При этом во входном потоке могут встречаться строки вида

```
task 1 subtask subtask subtask
```

которые означают, что task требует выполнения subtask 3 раза.

+5 баллов за решение (без справки и shebang), работающее под управлением /bin/sh, размером не более 280 байт (у нас получилось 172).

Задание берёт: Наливайко

13. **7 баллов.** Напишите скрипт shquote, который читает построчно аргументы командной строки и собирает их в одну строку с соответствующим “закавычиванием” для правильной дальнейшей интерпретацией командной оболочкой. **+1 балл:** ‘\0’-разделитель вместо ‘\n’. Пример:

```
$ cat input.txt
echo
hello   world
don’t bother
$ squote < input.txt
echo ‘hello   world’ ‘don’\’’t bother’
```

Задание берёт: Назарук, Дужик