

-- 05-a-2000

Сменете вашия `prompt` с нещо по желание. После върнете оригиналния обратно.

-- 05-a-2100

Редактирайте вашия `.bash_profile` файл, за да ви поздравява (или да изпълнява някаква команда по ваш избор) всеки път, когато влезете в системата.

-- 05-a-2200

Направете си ваш псевдоним (`alias`) на полезна команда.

-- 05-b-2000

Да се напише `shell` скрипт, който приканва потребителя да въведе низ (име) и изпечатва "Hello, низ".

-- 05-b-2800

Да се напише `shell` скрипт, който приема точно един параметър и проверява дали подаденият му параметър се състои само от букви и цифри.

-- 05-b-3100

Да се напише `shell` скрипт, който приканва потребителя да въведе низ – потребителско име на потребител от системата – след което извежда на стандартния изход колко активни сесии има потребителят в момента.

-- 05-b-3200

Да се напише `shell` скрипт, който приканва потребителя да въведе пълното име на директория и извежда на стандартния изход подходящо съобщение за броя на всички файлове и всички директории в нея.

-- 05-b-3300

Да се напише `shell` скрипт, който чете от стандартния вход имената на 3 файла, обединява редовете на първите два (`map paste`), подрежда ги по азбучен ред и резултата записва в третия файл.

-- 05-b-3400

Да се напише `shell` скрипт, който чете от стандартния вход име на файл и символен низ, проверява дали низа се съдържа във файла и извежда на стандартния изход кода на завършване на командата с която сте проверили наличието на низа.

NB! Символният низ може да съдържа интервал (' ') в себе си.

-- 05-b-4200

Имате компилируем (a.k.a няма синтактични грешки) source file на езика C. Напишете shell script, който да показва колко е дълбоко най-дълбокото nest-ване (влагане).

Примерен .c файл:

```
#include <stdio.h>

int main(int argc, char *argv[]) {

    if (argc == 1) {
        printf("There is only 1 argument");
    } else {
        printf("There are more than 1 arguments");
    }

    return 0;
}
```

Тук влагането е 2, понеже имаме main блок, а вътре в него if блок.

Примерно извикване на скрипта:

```
./count_nesting sum_c_code.c
```

Изход:

```
The deepest nesting is 2 levels
```

-- 05-b-4300

Напишете shell script, който по подаден като аргумент файл с map между <nickname> ↔ <реален username> и nickname, ще ви улесни да изпращате съобщения на хората.

Пример за файл указател:

```
tinko s61966
minko s881234
ginko s62000
dinko s77777
```

Примерно извикване на програмата:

```
./send_message myAddressBook dinko
```

И вече може да изпращате съобщения на човека с username s77777

NB! Можете да създавате нови потребители използвайки 'sudo useradd username'. За да проверите дали се пращат съобщенията отворете 2 сесии към виртуалката ви, създайте новият потребител, логнете се с него ('sudo su username' от едната сесия) и от другата сесия пратете съобщението.

-- 05-b-4301

Напишете shell script, който автоматично да попълва файла указател от предната задача по подадени аргументи: име на файла указател, пълно име на човека (това, което очакваме да е в /etc/passwd) и избран за него nickname.

Файлът указател нека да е във формат:

<nickname, който лесно да запомните> <username в os-server>
// може да сложите и друг delimiter вместо интервал

Примерно извикване:

./populate_address_book myAddressBook "Ben Dover" uncleBen

Добавя към myAddressBook entry-то:

uncleBen <username на Ben Dover в os-server>

***Бонус: Ако има няколко съвпадения за въведеното име (напр. има 10 човека Ivan Petrov в /etc/passwd), всички те да се показват на потребителя, заедно с пореден номер ≥ 1 , след което той да може да въведе някой от номерата (или 0 ако не си хареса никого), и само избраният да бъде добавен към указателя.

-- 05-b-4400

Напишете shell script, който да приема параметър име на директория, от която взимаме файлове, и опционално експлицитно име на директория, в която ще копираме файлове. Скриптът да копира файловете със съдържание, променено преди по-малко от 45 мин, от първата директория във втората директория. Ако втората директория не е подадена по име, нека да получи такова от днешната дата във формат, който ви е удобен. При желание новосъздадената директория да се архивира.

-- 05-b-4500

Да се напише shell скрипт, който получава при стартиране като параметър в командния ред идентификатор на потребител.

Скриптът периодично (`sleep(1)`) да проверява дали потребителят е `log-нат`, и ако да – да прекратява изпълнението си, извеждайки на стандартния изход подходящо съобщение.

NB! Можете да тествате по същият начин като в `05-b-4300.txt`

-- 05-b-4600

Да се напише `shell` скрипт, който валидира дали дадено цяло число попада в целочислен интервал.

Скриптът приема 3 аргумента: числото, което трябва да се провери; лява граница на интервала; дясна граница на интервала.

Скриптът да връща `exit status`:

- 3, когато поне един от трите аргумента не е цяло число
- 2, когато границите на интервала са обърнати
- 1, когато числото не попада в интервала
- 0, когато числото попада в интервала

Примери:

```
$ ./validint.sh -42 0 102; echo $?
```

1

```
$ ./validint.sh 88 94 280; echo $?
```

1

```
$ ./validint.sh 32 42 0; echo $?
```

2

```
$ ./validint.sh asdf - 280; echo $?
```

3

-- 05-b-4700

Да се напише `shell` скрипт, който форматира големи числа, за да са по-лесни за четене.

Като пръв аргумент на скрипта се подава цяло число.

Като втори незадължителен аргумент се подава разделител. По подразбиране цифрите се разделят с празен интервал.

Примери:

```
$ ./nicenumber.sh 1889734853
```

1 889 734 853

```
$ ./nicenumber.sh 7632223 ,
```

7,632,223

-- 05-b-4800

Да се напише shell скрипт, който приема файл и директория. Скриптът проверява в подадената директория и нейните под-директории дали съществува копие на подадения файл и отпечатва имената на намерените копия, ако съществуват такива.

NB! Под 'копие' разбираме файл със същото съдържание.

-- 05-b-5500

Да се напише shell script, който генерира HTML таблица съдържаща описание на потребителите във виртуалката ви. Таблицата трябва да има:

- заглавен ред с имената на колоните
- колони за username, group, login shell, GECKO field (man 5 passwd)

Пример:

```
$ ./passwd-to-html.sh > table.html
```

```
$ cat table.html
```

```
<table>
  <tr>
    <th>Username</th>
    <th>group</th>
    <th>login shell</th>
    <th>GECKO</th>
  </tr>
  <tr>
    <td>root</td>
    <td>root</td>
    <td>/bin/bash</td>
    <td>GECKO here</td>
  </tr>
  <tr>
    <td>ubuntu</td>
    <td>ubuntu</td>
    <td>/bin/dash</td>
    <td>GECKO 2</td>
  </tr>
</table>
```

-- 05-b-6600

Да се напише shell скрипт, който получава единствен аргумент директория и изтрива всички повтарящи се (по съдържание) файлове в дадената директория. Когато има няколко еднакви

файла, да се остави само този, чието име е лексикографски преди имената на останалите дублирани файлове.

Примери:

```
$ ls .
f1 f2 f3 asdf asdf2
# asdf и asdf2 са еднакви по съдържание, но f1, f2, f3 са
уникални
```

```
$ ./rmdup .
$ ls .
f1 f2 f3 asdf
# asdf2 е изтрит
```

-- 05-b-6800

Да се напише shell скрипт, който получава единствен аргумент директория и отпечатва списък с всички файлове и директории в нея (без скритите).

До името на всеки файл да седи размера му в байтове, а до името на всяка директория да седи броят на елементите в нея (общ брой на файловете и директориите, без скритите).

а) Добавете параметър -a, който указва на скрипта да проверява и скритите файлове и директории.

Пример:

```
$ ./list.sh .
asdf.txt (250 bytes)
Documents (15 entries)
empty (0 entries)
junk (1 entry)
karh-pishtov.txt (8995979 bytes)
scripts (10 entries)
```

-- 05-b-7000

Да се напише shell скрипт, който приема произволен брой аргументи – имена на файлове. Скриптът да прочита от стандартния вход символен низ и за всеки от зададените файлове извежда по подходящ начин на стандартния изход броя на редовете, които съдържат низа.

NB! Низът може да съдържа интервал.

-- 05-b-7100

Да се напише shell скрипт, който приема два параметъра – име на директория и число. Скриптът да извежда на стандартния изход имената на всички обикновени файлове във директорията, които имат размер, по-голям от подаденото число.

-- 05-b-7200

Да се напише shell скрипт, който приема произволен брой аргументи – имена на файлове или директории. Скриптът да извежда за всеки аргумент подходящо съобщение:

- дали е файл, който може да прочетем
- ако е директория – имената на файловете в нея, които имат размер, по-малък от броя на файловете в директорията.

-- 05-b-7500

Напишете shell script guess, която си намисля число, което вие трябва да познате. В зависимост от вашия отговор, програмата трябва да ви казва "надолу" или "нагоре", докато не познате числото. Когато го познаете, програмата да ви казва с колко опита сте успели.

./guess (програмата си намисля 5)

```
Guess? 22
...smaller!
Guess? 1
...bigger!
Guess? 4
...bigger!
Guess? 6
...smaller!
Guess? 5
RIGHT! Guessed 5 in 5 tries!
```

Hint: Един начин да направите рандъм число е с `$(((RANDOM % b) + a))`, което ще генерира число в интервала `[a, b]`. Може да вземете `a` и `b` като параметри, но не забравяйте да направите проверката.

-- 05-b-7550

Да се напише shell скрипт, който приема параметър – име на потребител. Скриптът да прекратява изпълнението на всички текущо работещи процеси на дадения потребител, и да извежда колко са били те.

NB! Може да тествате по същият начин като описаният в 05-b-4300

-- 05-b-7700

Да се напише shell скрипт, който приема два параметъра – име на директория и число. Скриптът да извежда сумата от размерите на файловете в директорията, които имат размер, по-голям от подаденото число.

-- 05-b-7800

Да се напише shell скрипт, който намира броя на изпълнимите файлове в PATH.

Hint: Предполага се, че няма спейсове в имената на директориите

Hint2: Ако все пак искаме да се справим с този случай, да се разгледа IFS променливата и констуркцията while read -d

-- 05-b-8000

Напишете shell script, който получава като единствен аргумент име на потребител и за всеки негов процес изписва съобщение за съотношението на RSS към VSZ. Съобщенията да са сортирани, като процесите с най-много заета виртуална памет са най-отгоре.

Hint:

Понеже в Bash няма аритметика с плаваща запетая, за смятането на съотношението използвайте командата bc. За да сметнем например 24/7, можем да: echo "scale=2; 24/7" | bc

Резултатът е 3.42 и има 2 знака след десетичната точка, защото scale=2.

Алтернативно, при липса на bc ползвайте awk.

-- 05-b-9100

Опишете поредица от команди или напишете shell скрипт, които/който при известни две директории SOURCE и DESTINATION:

- намира уникалните "разширения" на всички файлове, намиращи се някъде под SOURCE. (За простота приемаме, че в имената на файловете може да се среща символът точка '.' максимум веднъж.)

- за всяко "разширение" създава по една поддиректория на DESTINATION със същото име

- разпределя спрямо "разширението" всички файлове от SOURCE в съответните поддиректории в DESTINATION

-- 05-b-9200

Да се напише shell скрипт, който получава произволен брой аргументи файлове, които изтрива.

Ако бъде подадена празна директория, тя бива изтрита. Ако подадения файл е директория с поне 1 файл, тя не се изтрива.

За всеки изтрит файл (директория) скриптът добавя ред във log файл с подходящо съобщение.

а) Името на log файла да се чете от shell environment променлива, която сте конфигурирали във вашия .bashrc.

б) Добавете параметър -r на скрипта, който позволява да се изтриват непразни директории рекурсивно.

в) Добавете timestamp на log съобщенията във формата:

```
2018-05-01 22:51:36
```

Примери:

```
$ export RMLOG_FILE=~/.logs/remove.log
$ ./rmlog -r f1 f2 f3 mydir/ emptydir/
$ cat $RMLOG_FILE
[2018-04-01 13:12:00] Removed file f1
[2018-04-01 13:12:00] Removed file f2
[2018-04-01 13:12:00] Removed file f3
[2018-04-01 13:12:00] Removed directory recursively mydir/
[2018-04-01 13:12:00] Removed directory emptydir/
```

-- 05-b-9500

(Цветно принтиране) Напишете shell script color_print, който взима два параметъра.

Първият може да е измежду "-r", "-g" "-b", а вторият е произволен string.

На командата "echo" може да се подаде код на цвят, който ще оцвети текста в определения цвят.

В зависимост от първия аргумент, изпринтете втория аргумент в определения цвят:

"-r" е червено. Кодът на червеното е '\033[0;31m' (echo -e "\033[0;31m This is red")

"-g" е зелено. Кодът на зеленото е '\033[0;32m' (echo -e "\033[0;32m This is green")

"-b" е синьо. Кодът на синьото е '\033[0;34m' (echo -e "\033[0;34m This is blue")

Ако е подадена друга буква изпишете "Unknown colour", а ако изобщо не е подаден аргумент за цвят, просто изпишете текста.

Hint:

В края на скрипта си напишете:

```
echo '\033[0m'
```

,за да не се прецакат цветовете на терминала. Това е цветът на "няма цвят".

-- 05-b-9501

Този път програмата ви ще приема само един параметър, който е измежду ("-r", "-b", "-g", "-x").

Напишете shell script, който приема редовете от stdin и ги изпринтва всеки ред с редуващ се цвят. Цветовете вървят RED-GREEN-BLUE и цветът на първия ред се определя от аргумента.

Ако е подаден аргумент "-x", то не трябва да променят цветовете в терминала (т.е., все едно сте извикали командата cat).

hint: Не забравяйте да връщате цветовете в терминала.

-- 05-b-9600

Да се напише shell скрипт, който получава произволен брой аргументи файлове, които изтрива.

Ако бъде подадена празна директория, тя бива изтрита. Ако подадения файл е директория с поне 1 файл, тя не се изтрива.

Да се дефинира променлива BACKUP_DIR (или друго име), в която:

- изтритите файлове се компресират и запазват
- изтритите директории се архивират, компресират и запазват
- имената на файловете е "filename_yyyy-mm-dd-NN-MM-SS.{gz,tgz}", където filename е оригиналното име на файла (директорията) преди да бъде изтрита

а) Добавете параметър -r на скрипта, който позволява да се изтриват непразни директории рекурсивно и съответно да се запазят в BACKUP_DIR

Примери:

```
$ export BACKUP_DIR=~/.backup/
```

```
# full-dir/ има файлове и не може да бъде изтрита без параметър -r
```

```
$ ./trash f1 f2 full-dir/ empty-dir/
```

```
error: full-dir/ is not empty, will not delete
```

```
$ ls $BACKUP_DIR
```

```
f1_2018-05-07-18-04-36.gz
```

```
f2_2018-05-07-18-04-36.gz  
empty-dir_2018-05-07-18-04-36.tgz
```

```
$ ./trash -r full-dir/
```

```
$ ls $BACKUP_DIR  
f1_2018-05-07-18-04-36.gz  
f2_2018-05-07-18-04-36.gz  
full-dir_2018-05-07-18-04-50.tgz  
empty-dir_2018-05-07-18-04-36.tgz
```

можем да имаме няколко изтрети файла, които се казват по един и същ начин

```
$ ./trash somedir/f1
```

```
$ ls $BACKUP_DIR  
f1_2018-05-07-18-04-36.gz  
f1_2018-05-07-18-06-01.gz  
f2_2018-05-07-18-04-36.gz  
full-dir_2018-05-07-18-04-50.tgz  
empty-dir_2018-05-07-18-04-36.tgz
```

-- 05-b-9601

Да се напише shell скрипт, който възстановява изтрети файлове, които имат запазено копие в BACKUP_DIR (от предната задача).

При възстановяването файловете да се декомпресират, а директориите да се декомпресират и разархивират.

а) Да се дефинира параметър -l, който изрежда всички файлове, които могат да бъдат възстановени и датата на тяхното изтриване.

б) Скриптът да приема 2 параметъра. Първият е името на файла, който да се възстанови, а вторият е директорията, в която файлът да бъде възстановен. Ако вторият аргумент липсва, файлът да се възстановява в сегашната директория, където скриптът се изпълнява.

в) Когато има $n > 1$ запазени файла със същото име, да се изпише списък с n реда на потребителя и да се изиска той да въведе цяло число от 1 до n , за да избере кой файл да възстанови.

Примери:

```
# BACKUP_DIR трябва да е дефинирана преди използването на скрипта
```

```
$ echo $BACKUP_DIR
```

```
~/backup
```

```
$ ./restore.sh -l
```

```
f1 (2018/05/07 18:04:36)
```

```
f1 (2018/05/07 18:06:01)
```

```
f2 (2018/05/07 18:04:36)
```

```
full-dir (2018/05/07 18:04:50)
```

```
empty-dir (2018/05/07 18:04:36)
```

```
$ ls restored-dir/
```

```
# възстановяване на файл в подадена директория
```

```
$ ./restore.sh f2 target-dir/
```

```
$ ls restored-dir/
```

```
f2
```

```
# възстановяване на дублиран файл в сегашната директория
```

```
$ ./restore.sh f1
```

```
(1) f1 (2018/05/07 18:04:36)
```

```
(2) f1 (2018/05/07 18:06:01)
```

```
choose file (1, 2):
```

```
# потребителят въвежда 2
```

```
$ ls
```

```
f1
```

```
$ ./restore.sh -l
```

```
f1 (2018/05/07 18:04:36)
```

```
full-dir (2018/05/07 18:04:50)
```

```
empty-dir (2018/05/07 18:04:36)
```

```
# възстановяване на директория в сегашната директория
```

```
$ ./restore.sh full-dir
```

```
$ ls
```

```
f1 full-dir/
```