Зад. 1
Да се напише shell скрипт, който приканва потребителя да въведе низ (име) и изпечатва "Hello, низ".

Зад. 2
Да се напише shell скрипт, който приема точно един параметър и проверява дали подаденият му параметър се състои само от букви и цифри.

Зад. 3
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.

Зад. 4
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.

Зад. 5
Да се напише shell скрипт, който чете от стандартния вход имената на 3 файла, обединява редовете на първите два, подрежда ги по азбучен ред и резултата записва в третия файл.

Зад. 6
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. Символният низ може да съдържа интервал (' ') в себе си.

Зад. 7
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

Зад. 8
Напишете shell script, който автоматично да попълва файл указател 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:
./pupulate_address_book myAddressBook "Ben Dover" uncleBen uncleBen <username на Ben Dover в os-server>
Добавя към myAddressBook entry-то:
Ако има няколко съвпадения за въведеното име, всички те да се покажат 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.

Зад. 9
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.

Зад. 10
Да се напише shell скрипт, който получава при стартиране като параметър в командния ред идентификатор на потребител. Скриптът периодично да проверява дали потребителят е log-нат, и ако да - да прекратява изпълнението си, извеждайки на стандартния изход подходящо съобщение.

Зад. 11
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
```

Зад. 12
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
```

Зад. 13
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. Под 'копие' разбираме файл със същото съдържание.

Зад. 14
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, GECOS field.

Пример:

```
$ ./passwd-to-html.sh > table.html
$ cat table.html
<table>
        <tr>
                <th>Username</th>
                <th>group</th>
                <th>login shell</th>
                <th>GECOS</th>
        </tr>
        <tr>
                <td>root</td>
                <td>root</td>
                <td>/bin/bash</td>
                <td>GECOS here</td>
        </tr>
        <tr>
                <td>ubuntu</td>
                <td>ubuntu</td>
                <td>/bin/dash</td>
                 <td>GECOS 2</td>
        </tr>
</table>
```

Зад. 15
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

Зад. 16
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).
Добавете параметъра, 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)

Зад. 17
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. Низът може да съдържа интервал.

Зад. 18
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 директориятa, които имат размер, по-голям от подаденото число.

Зад. 19
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.

Зад. 20
Напишете shell script guess, който си намисля число, което вие трябва да познаете. В зависимост от вашия отговор, програмата трябва да ви казва "надолу" или "нагоре", докато не познаете 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!

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.

Зад. 21
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.

Зад. 22
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.

Зад. 23
Да се напише shell скрипт, който намира броя на изпълнимите файлове в PATH. Може да има спейсове в имената на директориите.

Зад. 24
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.

Зад. 25
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. (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.

Зад. 26
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.

- Името на log файла да се чете от shell environment променлива, която сте конфигурирали във вашия .bashrc
- Добавете параметър -r на скрипта, който позволява да се изтриват непразни директории рекурсивно
- 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/
```

Зад. 27
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, изпринтите 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. В края на скрипта си напишете: echo '\033[0m' ,за да не се прецакат цветовете на терминала. Това е цветът на "няма цвят".

Зад. 28
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 променяте цветовете в терминала. Не забравяйте да връщате цветовете в терминала.

Зад. 29
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, в която:

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

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 detele
$ ls $BACKUP_DIR f1_2018-05-07-18-04-36.gz
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
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 изтрити 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

Зад. 30
Да се напише shell скрипт, който възстановява изтрити файлове, които имат запазено копие в BACKUP_DIR (от предната задача). При възстановяването файловете да се декомпресират, а директориите да се декомпресират и разархивират.
Да се дефинира параметър -l, който изрежда всички файлове, които могат да бъдат възстановени и датата на тяхното изтриване.
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.
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/
```

Зад. 31
Даден е текстов файл с име philip-j-fry.txt. Напишете shell script, които извеждат броя редове, съдържащи поне една четна цифра и несъдържащи малка латинска буква от a до w.
Примерно съдържание на файла:
123abv123

123zz123
MMU_2.4

Примерен изход:

Броят на търсените редовете е 2

Зад. 32

Напишете shell скрипт, който по подаден един позиционен параметър, ако този параметър е директория, намира всички symlink-ове в нея и под-директориите и` с несъществуващ destination.

Зад. 33

Напишете shell скрипт който, ако се изпълнява от root, проверява кои потребители на системата нямат homedir или не могат да пишат в него.

Примерен формат:

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin

Зад. 34

В текущата директория има само обикновени файлове (без директории). Да се напише bash script, който приема 2 позиционни параметъра – числа, който мести файловете от текущата директория към нови директории (a, b и c, които трябва да бъдат създадени), като определен файл се мести към директория 'a', само ако той има по-малко редове от първи позиционен параметър, мести към директория 'b', ако редове са между първи и втори позиционен параметър и в 'c' в останалите случаи.

Зад. 35

Напишете скрипт, който извежда името на потребителския акаунт, в чиято home директория има най-скоро променен обикновен файл и кой е този файл.

Зад. 36

Както знаете, при отваряне на файл с редактора vi, той създава в същата директория временен файл с име в следния формат: точка, името на оригиналния файл, точка, swp. Например, при редактиране на файл с име foo.txt ще се създаде временен файл с име .foo.txt.swp.

Напишете shell скрипт, който приема два задължителни позиционни аргумента – имена на директории.

Примерно извикване: ./foo.sh ./dir1 /path/to/dir2/

В dir1 може да има файлове/директории, директорията dir2 трябва да е празна.

Скриптът трябва да копира всички обикновенни файлове от dir1 (и нейните под-директории) в dir2, запазвайки директорийната структура, но без да копира временните файлове, създадени от редактора vi (по горната дефиниция).

Примерни обекти:

dir1/
dir1/a
dir1/.a.swp
dir1/b
dir1/c/d
dir1/c/.bar.swp

Обекти след изпълнението:

dir2/
dir2/a
dir2/b
dir2/c/d dir2/c/.bar.swp

Зад. 37

Напишете скрипт, който получава задължителен първи позиционен параметър – директория и незадължителен втори – число. Скриптът трябва да проверява подадената директория и нейните под-директории и да извежда имената на:

- при подаден на скрипта втори параметър – всички файлове с брой hardlink-ове поне равен на параметъра
- при липса на втори параметър – всички symlink-ове с несъществуващ destination (счупени symlink-ове).

Зад. 38

Напишете shell script, който получава задължителен първи позиционен параметър – директория и незадължителен втори – име на файл. Скриптът трябва да намира в подадената директория и нейните под-директории всички symlink-ове и да извежда (при подаден аргумент файл – добавяйки към файла, а ако не е – на стандартния изход) за тях следната информация:

- ако destination-а съществува – името на symlink-а -> името на destination-а
- броя на symlink-овете, чийто destination не съществува.

Примерен изход:

lbaz -> /foo/bar/baz
lqux -> ../../../qux lquux -> /foo/quux

Broken symlinks: 34

Зад. 39
Вашите колеги от съседната лаборатория ползват специализиран софтуер за оптометрични изследвания, който записва резултатите от всяко измерване в отделен файл. Файловете имат уникално съдържание, по което се определя за кое измерване се отнася файла. За съжаление, тъй като колегите ви ползват бета версия на софтуера, той понякога записва по няколко пъти резултатите от дадено измерване в произволна комбинация от следните варианти:

- нула или повече отделни обикновенни файла с еднакво съдържание
- нула или повече групи от hardlink-ове, като всяка група съдържа две или повече имена на даден файл с измервания.

Помогнете на колегите си, като напишете shell скрипт, който приема параметър – име на директория, съдържаща файлове с измервания. Скриптът трябва да извежда на стандартния изход списък с имена на файлове, кандидати за изтриване, по следните критерии:

- ако измерването е записано само в отделни файлове, трябва да остане един от тях
- ако измерването е записано само в групи от hardlink-ове, всяка група трябва да се намали с едно име
- ако измерването е записано и в групи, и като отделни файлове, за групите се ползва горния критерий, а всички отделни файлове се премахват.

Зад. 40
Файловете във вашата home директория съдържат информация за музикални албуми и имат специфична структура. Началото на всеки ред е годината на издаване на албума, а непосредствено, след началото на всеки ред следва името на изпълнителя на песента. Имената на файловете се състоят от една дума, която съвпада с името на изпълнителя.
Примерно съдържание на файл с име "Bonnie":

2005г. Bonnie - "God Was in the Water" (Randall Bramblett, Davis Causey) – 5:17
2005г. Bonnie - "Love on One Condition" (Jon Cleary) – 3:43
2005г. Bonnie - "So Close" (Tony Arata, George Marinelli, Pete Wasner) – 3:22
2005г. Bonnie - "Trinkets" (Emory Joseph) – 5:02
2005г. Bonnie - "Crooked Crown" (David Batteau, Maia Sharp) – 3:49
2005г. Bonnie - "Unnecessarily Mercenary" (Jon Cleary) – 3:51
2005г. Bonnie - "I Will Not Be Broken" - "Deep Water" (John Capek, Marc Jordan) – 3:58

Да се напише shell скрипт приемащ два параметъра, които са имена на файлове от вашата home директория. Скриптът сравнява, кой от двата файла има повече на брой редове, съдържащи неговото име (на файла). За файлът победител изпълнете следните действия:

- извлечете съдържанието му, без годината на издаване на албума и без името на изпълнителя
- сортирайте лексикографски извлеченото съдържание и го запишете във файл с име ’изпълнител.songs’ .

Примерен изходен файл (с име Bonnie.songs):

"Crooked Crown" (David Batteau, Maia Sharp) – 3:49
"God Was in the Water" (Randall Bramblett, Davis Causey) – 5:17
"I Will Not Be Broken" - "Deep Water" (John Capek, Marc Jordan) – 3:58
"Love on One Condition" (Jon Cleary) – 3:43
"So Close" (Tony Arata, George Marinelli, Pete Wasner) – 3:22
"Trinkets" (Emory Joseph) – 5:02
"Unnecessarily Mercenary" (Jon Cleary) – 3:51

Зад. 41
Имате текстов файл със следното съдържание (всяка книга е на един ред):

1979 г. - „Синият тайфун“ (сборник съветски научнофантастични разкази за морето)
1979 г. - „Двойната звезда“ - Любен Дилов
1979 г. - „Завръщане от звездите“ - Станислав Лем (Превод: Веселин Маринов)
1979 г. - „Среща с Рама“ - Артър Кларк (Превод: Александър Бояджиев)
1979 г. - „Алиби“ - Димитър Пеев (криминален роман)
1979 г. - „Тайнственият триъгълник“ (сборник НФ разкази за морето)
1979 г. - „Второто нашествие на марсианците“ - Аркадий и Борис Стругацки
1979 г. - „Гробищен свят“ - Клифърд Саймък (Превод: Михаил Грънчаров)
1979 г. - „Чоки“ - Джон Уиндъм (Превод: Теодора Давидова)
1979 г. - „Спускане в Маелстрьом“ - Едгар Алан По (Превод: Александър Бояджиев)
1980 г. - „Допълнителна примамка“ - Робърт Ф. Йънг (Превод: Искра Иванова, ...)
1980 г. - „Кристалното яйце“ - Хърбърт Уелс (Превод: Борис Миндов, ...)
1980 г. - „Онирофилм“ (сборник италиански НФ разкази) (Превод: Никола Иванов, ...)

Напишете shell script (приемащ аргумент име на файл) който извежда:

- всеки ред от файла с добавен пореден номер във формат "1. ", "2. ", ... "11. " ...
- махат данните за годината на издаване
- сортират изхода по заглавие (лексикографски, възходящо).

Примерен изход (показани са само първите 4 реда):

5. „Алиби“ - Димитър Пеев (криминален роман)
7. „Второто нашествие на марсианците“ - Аркадий и Борис Стругацки

8. „Гробищен свят“ - Клифърд Саймък (Превод: Михаил Грънчаров)
2. „Двойната звезда“ - Любен Дилов

Зад. 42

Напишете скрипт, който приема два позиционни аргумента – име на текстови файл и директория. Директорията не трябва да съдържа обекти, а текстовият файл (US-ASCII) е стенограма и всеки ред е в следния формат:

ИМЕ ФАМИЛИЯ (уточнения): Реплика

където:

- ИМЕ ФАМИЛИЯ присъстват задължително
- ИМЕ и ФАМИЛИЯ се състоят само от малки/главни латински букви и тирета
- (уточнения) не е задължително да присъстват
- двоеточието ‘:’ присъства задължително
- Репликата не съдържа знаци за нов ред
- в стринга преди двоеточието ‘:’ задължително има поне един интервал между ИМЕ и ФАМИЛИЯ
- наличието на други интервали където и да е на реда е недефинирано.

Примерен входен файл:

John Lennon (The Beatles): Time you enjoy wasting, was not wasted.
Roger Waters: I’m in competition with myself and I’m losing.
John Lennon:Reality leaves a lot to the imagination.
Leonard Cohen:There is a crack in everything, that’s how the light gets in.

Скриптът трябва да:

- създава текстови файл dict.txt в посочената директория, който на всеки ред да съдържа:
  ИМЕ ФАМИЛИЯ;НОМЕ
  където:
  - ИМЕ ФАМИЛИЯ е уникален участник в стенограмата (без да се отчитат уточненията)
  - НОМЕР е уникален номер на този участник, избран от вас
- създава файл НОМЕР.txt в посочената директория, който съдържа всички (и само) редовете на дадения участник.

Зад. 43

Напишете скрипт, който приема два аргумента - имена на директории. Първата (SRC) съществува, докато втората (DST) трябва да бъде създадена от скрипта. Директорията SRC и нейните поддиректории може да съдържат файлове, чиито имена завършат на .jpg. Имената на файловете може да съдържат интервали, както и поднизове, оградени със скоби, например:

A single (very ugly) tree (Outdoor treks) 2.jpg
Falcons.jpg
Gorgonzola (cheese).jpg
Leeches (two different ones) (Outdoor treks).jpg
Pom Pom Pom.jpg

За даден низ ще казваме, че е почистен, ако от него са премахнати leading и trailing интервалите и всички последователни интервали са сведени до един.

За всеки файл дефинираме следните атрибути:

- заглавие – частта от името преди .jpg, без елементи оградени в скоби, почистен
  Примери:
  A single tree 2 Falcons Gorgonzola Leeches
  Pom Pom Pom
- албум - последният елемент от името, който е бил ограден в скоби, почистен. Ако албум е празен стринг, ползваме стойност по подразбиране misc
  Примери:
  Outdoor treks
  misc
  cheese
  Outdoor treks
  misc
- дата - времето на последна модификация на съдържанието на файла, във формат YYYY- MM-DD
- хеш - първите 16 символа от sha256 сумата на файла. Забележка: приемаме, че в тази идеална вселена първите 16 символа от sha256 сумата са уникални за всеки файл от тези, които ще се наложи да обработваме.

Скриптът трябва да създава в директория DST необходимата структура от под-директории, файлове и symlink–ове, така че да са изпълнени следните условия за всеки файл от SRC :

- DST/images/хеш.jpg - копие на съответния файл
- следните обекти са относителни symlink–ове към хеш.jpg:
  - DST/by-date/дата/by-album/албум/by-title/заглавие.jpg
  - DST/by-date/дата/by-title/заглавие.jpg
  - DST/by-album/албум/by-date/дата/by-title/заглавие.jpg
  - DST/by-album/албум/by-title/заглавие.jpg
  - DST/by-title/заглавие.jpg

Зад. 44
Напишете shell скрипт, който приема един позиционен параметър - число. Ако скриптът се изпълнява като root, да извежда обобщена информация за общото количество активна памет (RSS - resident set size, non-swapped physical memory that a task has used) на процесите на всеки потребител. Ако за някой потребител обобщеното число надвишава подадения параметър, да изпраща подходящи сигнали за прекратяване на процеса с най-много активна памет на потребителя. Приемаме, че изхода в колоната RSS е число в същата мерна единица, като числото, подадено като аргумент.

Зад. 45
Напишете скрипт, който приема задължителен позиционен аргумент - име на потребител FOO. Ако скриптът се изпълнява от root:

- да извежда имената на потребителите, които имат повече на брой процеси от FOO, ако има такива
- да извежда средното време (в секунди), за което са работили процесите на всички потребители на системата (TIME, във формат HH:MM:SS)
- ако съществуват процеси на FOO, които са работили над два пъти повече от средното време, скриптът да прекратява изпълнението им по подходящ начин.

Зад. 46
Напишете скрипт, който ако се изпълнява от root потребителя:

- извежда обобщена информация за броя и общото количество активна памет (RSS - resident set size, non-swaped physical memory that a task has used) на текущите процеси на всеки потребител
- ако процесът с най-голяма активна памет на даден потребител използва два пъти повече памет от средното за потребителя, то скриптът да прекратява изпълнението му по подходящ начин.

Зад. 47
Напишете скрипт, който ако се изпълнява от root потребителя, намира процесите на потребителите, които не са root потребителя и е изпълнено поне едно от следните неща:

- имат зададена несъществуваща home директория
- не са собственици на home директорията си
- собственика на директорията не може да пише в нея.

Ако общото количество активна памет (RSS - resident set size, non-swaped physical memory that a task has used) на процесите на даден такъв потребител е по-голямо от общото количество активна памет на root потребителя, то скриптът да прекратява изпълнението на всички процеси на потребителя.

Зад. 48
Напишете скрипт, който приема три задължителни позиционни аргумента: име на файл; низ1; низ2
Файлът е текстови, и съдържа редове във формат:

ключ =стойност

където стойност може да бъде:

- празен низ, т.е. редът е ключ=
- низ, състоящ се от един или повече термове, разделени с интервали, т.е., редът е ключ=t1 t2 t3

Някъде във файла:

- се съдържа един ред с ключ първия подаден низ (низ1)
- и може да се съдържа един ред с ключ втория подаден низ (низ2).

Скриптът трябва да променя реда с ключ низ2 така, че обединението на термовете на редовете с ключове низ1 и низ2 да включва всеки терм еднократно.
Примерен входен файл:

```
$ cat z1.txt
        FOO=73
        BAR=42
        BAZ=
        ENABLED_OPTIONS=a b c d
        ENABLED_OPTIONS_EXTRA=c e f
```

Примерно извикване:

```
$ ./a.sh z1.txt ENABLED_OPTIONS ENABLED_OPTIONS_EXTRA
```

Изходен файл:

```
$ cat z1.txt
        FOO=73
        BAR=42
        BAZ=
        ENABLED_OPTIONS=a b c d ENABLED_OPTIONS_EXTRA=e f
```

Зад. 49
Напишете скрипт, който приема три задължителни позиционни параметра - директория SRC, директория DST (която не трябва да съдържа файлове) и низ ABC. Ако скриптът се изпълнява от root потребителя, то той трябва да намира всички файлове в директорията SRC и нейните под-директории, които имат в името си като под-низ ABC, и да ги мести в директорията DST, запазвайки директорийната структура (но без да запазва мета-данни като собственик и права, т.е. не

ни интересуват тези параметри на новите директории, които скриптът би генерирал в DST).
Пример:

- в SRC (/src) има следните файлове:
  /src/foof.txt /src/1/bar.txt
  /src/1/foo.txt
  /src/2/1/foobar.txt
  /src/2/3/barf.txt
- DST (/dst) е празна директория
- зададения низ е foo

Резултат:

- в SRC има следните файлове:
  /src/1/bar.txt
  /src/2/3/barf.txt
- в DST има следните файлове:
  /dst/foof.txt
  /dst/1/foo.txt
  /dst/2/1/foobar.txt

Зад. 50
Напишете скрипт, който получава два задължителни позиционни параметъра – директория и низ. Сред файловете в директорията би могло да има такива, чиито имена имат структура vmlinuz-x.y.z-arch където:

- vmlinuz е константен низ;
- тиретата “-” и точките “.” присъстват задължително
- x е число, version
- y е число, major revision
- z е число, minor revision
- наредената тройка x.y.z формира глобалната версия на ядрото
- arch е низ, архитектура (платформа) за която е съответното ядро.

Скриптът трябва да извежда само името на файла, намиращ се в подадената директория (но не и нейните поддиректории), който:

- спазва гореописаната структура
- е от съответната архитектура спрямо параметъра-низ, подаден на скрипта
- има най-голяма глобална версия.

Пример:

- Съдържание на ./kern/:
  vmlinuz-3.4.113-amd64
  vmlinuz-4.11.12-amd64
  vmlinuz-4.12.4-amd64
  vmlinuz-4.19.1-i386
- Извикване и изход:
  $ ./task1.sh ./kern/ amd64
  vmlinuz-4.12.4-amd64

Зад. 51
Нека съществува програма за моментна комуникация (Instant messaging), която записва логове на разговорите в следния формат:

- има определена директория за логове (LOGDIR)
- в нея има директорийна структура от следния вид:
- LOGDIR/протокол/акаунт/приятел/
  като на всяко ниво може да има няколко екземпляра от съответния вид, т.е. няколко директории протокол, във всяка от тях може да има няколко директории акаунт, и във всяка от тях – няколко директории приятел
- във всяка от директориите приятел може да има файлове с имена от вида yyyy-mm-dd-hh-mm-ss.txt – година-месец-ден и т.н., спрямо това кога е започнал даден разговор
- всеки такъв файл представлява лог на даден разговор със съответния приятел, като всяка разменена реплика между вас е на отделен ред
- даден идентификатор приятел може да се среща няколко пъти в структурата (напр. през различни ваши акаунти сте водили разговори със същия приятел).

Напишете скрипт, който приема задължителен позиционен аргумент - име на лог директория (LOGDIR). Скриптът трябва да извежда десетимата приятели, с които имате най-много редове комуникация глобално (без значение протокол и акаунт), и колко реда имате с всеки от тях.

Зад. 52
Напишете скрипт, който приема два позиционни аргумента – имена на текстови файлове в CSV формат:
8,foo,bar,baz

2,quz,,foo
12,1,3,foo
3,foo,,
5,,bar,
7,,,
4,foo,bar,baz

Валидни са следните условия

- CSV файловете представляват таблица, като всеки ред на таблицата е записан на отделен ред;
- на даден ред всяко поле (колона) е разделено от останалите със запетая
- броят на полетата на всеки ред е константа
- в полетата не може да присъства запетая, т.е., запетаята винаги е разделител между полета
- ако във файла присъстват интервали, то това са данни от дадено поле
- първото поле на всеки ред е число, което представлява идентификатор на реда (ID).

Примерно извикване: ./foo.sh a.csv b.csv

Скриптът трябва да чете a.csv и на негова база да създава b.csv по следния начин:

- някои редове във файла се различават само по колоната ID, и за тях казваме, че формират множество A_i
- за всяко такова множество A_i да се оставя само един ред - този, с най-малка стойност на ID-то
- редовете, които не са членове в някое множество A_i се записват в изходния файл без промяна.

Зад. 53

Напишете два скрипта (по един за всяка подточка), които четат редове от STDIN. Скриптовете трябва да обработват само редовете, които съдържат цели положителни или отрицателни числа; останалите редове се игнорират. Скриптовете трябва да извежда на STDOUT:

- всички уникални числа, чиято абсолютна стойност е равна на максималната абсолютна стойност сред всички числа
- всички най-малки уникални числа от тези, които имат максимална сума на цифрите си.

Примерен вход:

We don’t
n11d n0
educat10n
12.3
6
3
-42
-42
111
111
-111

Примерен изход за първия скрипт: -111
111

Примерен изход за втория скрипт: -42

Зад. 54

Напишете шел скрипт, който приема множество параметри. Общ вид на извикване: ./foo.sh [-n N] FILE1...

В общия случай параметрите се третират като имена на (.log) файлове, които трябва да бъдат обработени от скрипта, със следното изключение: ако първият параметър е стрингът -n, то вторият параметър е число, дефиниращо стойност на променливата N, която ще ползваме в скрипта. Въвеждаме понятието идентификатор на файл (ИДФ), което се състои от името на даден файл без разширението .log. За удобство приемаме, че скриптът:

- ще бъде извикван с аргументи имена на файлове, винаги завършващи на .log
- няма да бъде извикван с аргументи имена на файлове с еднакъв ИДФ.

Лог файловете са текстови, като всеки ред има следния формат:

- време: timestamp във формат YYYY-MM-DD HH:MM:SS
- интервал
- данни: поредица от символи с произволна дължина.

За удобство приемаме, че редовете във всеки файл са сортирани по време възходящо.

Примерно съдържание на даден лог файл:

2019-05-05 06:26:54 orthanc rsyslogd: rsyslogd was HUPed
2019-05-06 06:30:32 orthanc rsyslogd: rsyslogd was HUPed
2019-05-06 10:48:29 orthanc kernel: [1725379.728871] Chrome_~dThread[876]: segfault

Скриптът трябва да извежда на STDOUT последните N реда (ако N не е дефинирано - 10 реда) от всеки файл, в следния формат:

- timestamp във формат YYYY-MM-DD HH:MM:SS
- интервал
- ИДФ
- интервал

- данни.

Изходът трябва да бъде глобално сортиран по време възходящо.

Зад. 55

За удобство приемаме, че разполагате със системен инструмент sha256sum, който приема аргументи имена на файлове като за всеки файл пресмята и извежда уникална хеш стойност, базирана на съдържанието на файла. Изходът от инструмента е текстови, по един ред за всеки подаден като аргумент файл, в следния формат:

- хеш стойност с дължина точно 64 знака
- два интервала
- име на файл.

Примерна употреба и изход:

```
$ sha256sum /var/log/syslog /var/log/user.log README.md
    b2ff8bd882a501f71a144b7c678e3a6bc6764ac48eb1876fb5d11aac11014b78 /var/log/syslog
    e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 /var/log/user.log
    e4702d8044b7020af5129fc69d77115fd4306715bd678ba4bef518b2edf01fb9 README.md
```

Напишете скрипт, който приема задължителен параметър име на директория (ДИР1). Някъде в директорията ДИР1 може да съществуват архивни файлове с имена NAME_report-TIMESTAMP.tgz, където:

- NAME е низ, който не съдържа символ '_'
- TIMESTAMP е във формат Unix time (POSIX time/UNIX Epoch time)

На всяко пускане на скрипта се обработват само новосъздадените или модифицираните по съдържание спрямо предното пускане на скрипта архивни файлове от горния тип. За всеки такъв архивен файл се изпълнява следното:

- ако архивният файл съдържа файл с име meow.txt, то този текстови файл да бъде записан под името /extracted/NAME_TIMEST AMP .txt, където NAME и TIMEST AMP са съответните стойности от името на архивния файл.

Зад. 56

Напишете shell скрипт, който получава два задължителни позиционни параметъра - име на файл (bar.csv) и име на директория. Директорията може да съдържа текстови файлове с имена от вида foobar.log, всеки от които има съдържание от следния вид:

Пример 1 (loz-gw.log):

```
Licensed features for this platform:
    Maximum Physical Interfaces     : 8
    VLANs                           : 20
    Inside Hosts                    : Unlimited
    Failover                        : Active/Standby
    VPN-3DES-AES                    : Enabled
    *Total VPN Peers                : 25
    VLAN Trunk Ports                : 8
This platform has an ASA 5505 Security Plus license.
Serial Number: JMX00000000
Running Activation Key: 0e268e0c
```

Пример 2 (border-lozenets.log):

```
Licensed features for this platform:
    Maximum Physical Interfaces     : 4
    VLANs                           : 16
    Inside Hosts                    : Unlimited
    Failover                        : Active/Active
    VPN-3DES-AES                    : Disabled
    *Total VPN Peers                : 16
    VLAN Trunk Ports                : 4
This platform has a PIX 535 license.
Serial Number: PIX5350007
Running Activation Key: 0xd11b3d48
```

Имената на лог файловете (loz-gw, border-lozenets) определят даден hostname, а съдържанието им дава детайли за определени параметри на съответният хост.

Файлът bar.csv, който трябва да се генерира от вашия скрипт, е т.н. CSV (comma separated values) файл, тоест текстови файл - таблица, на който полетата на всеки ред са разделени със запетая. Първият ред се ползва за определяне на имената на колоните.

Скриптът трябва да създава файла bar.csv на база на log файловете в директорията. Генерираният CSV файл от директория, която съдържа само loz-gw.log и border-lozenets.log би изглеждал така:

```
hostname,phy,vlans,hosts,failover,VPN-3DES-AES,peers,VLAN Trunk Ports,license,SN,key
loz-gw,8,20,Unlimited,Active/Standby,Enabled,25,8,ASA 5505 Security Plus,JMX00000000,0e268e0c
border-lozenets,4,16,Unlimited,Active/Active,Disabled,16,4,PIX 535,PIX5350007,0xd11b3d48
```

Полетата в генерирания от скрипта CSV файл не трябва да съдържат излишни trailing/leading интервали. Приемете, че всички whitespace символи във входните файлове са символа "интервал".

Зад. 57
Напишете shell скрипт, който приема задължителен параметър - име на файл. Файлът е log файл на HTTP сървър, в който се записват всички получени от сървъра request-и, които клиентите са изпратили. Файлът е текстови, като на всеки ред има информация от следния вид:

35.223.122.181 dir.bg - [03/Apr/2020:17:25:06 -0500] GET / HTTP/1.1 302 0 "-" "Zend_Http_Client"
94.228.82.170 del.bg - [03/Apr/2020:17:25:06 -0500] POST /auth HTTP/2.0 400 153 "foo bar" "<UA>"

Всеки ред на файла се състои от полета, разделени с интервал. Описание на полетата с пример спрямо първият ред от горните:

- адрес на клиент - 35.223.122.181
- име на виртуален хост (сайт) - dir.bg
- име на потребител - -
- timestamp на заявката - [03/Apr/2020:17:25:06 -0500]
- заявка - GET / HTTP/1.1 - състои се от три компонента, разделени с интервал: метод на заявка- та (за удобство
- приемаме, че може да има само GET и POST заявки), ресурсен идентификатор, и протокол (приемаме, че може
- да има само HTTP/1.0, HTTP/1.1 и HTTP/2.0 протоколи)
- код за статус на заявката - 302
- брой байтове - 0
- referer - "-" - ограден в двойни кавички, подава се от HTTP клиента, произволен низ
- user agent - "Zend_Http_Client" - ограден в двойни кавички, подава се от HTTP клиента, произволен низ

За всеки от top 3 сайта, към които има най-много заявки, скриптът трябва да изведе в долния формат:

- брой на HTTP/2.0 заявките
- брой на не-HTTP/2.0 заявките
- top 5 клиента, направили най-много заявки, завършили с код, по-голям от 302 (и броя на съответните им зявки)

dir.bg HTTP/2.0: 0 non-HTTP/2.0: 5
del.bg HTTP/2.0: 5 non-HTTP/2.0: 0
5 94.228.82.170 2 34.73.112.204 1 185.217.0.138

Зад. 58
Под пакет ще разбираме директория, която има следната структура:

<name>
|-- version
|-- tree
|...

Където <name> е името на пакета, version е текстов файл, който съдържа низ от вида 1.2.3-4 и нищо друго, а tree е директория с произволно съдържание.
За да получим архив на пакет, архивираме (tar) и компресираме (xz) съдържанието на директорията tree.
Под хранилище ще разбираме директория, която има следната структура:

<repo name>
|-- db
|-- packages
|...

Където <repo name> е името на хранилището, db е текстов файл, чиито редове имат вида <package name>-<package version> <package checksum> и са сортирани лексикографски. Директорията packages съдържа архиви с имена <package checksum>.tar.xz, които съответстват на редове в db. Под <package checksum> имаме предвид sha256 сумата на архива на пакета.
Напишете скрипт repo_add.sh, който приема два аргумента - път до хранилище и път до пакет, който добавя пакета в хранилището. Ако същата версия на пакет вече съществува, архивът се заменя с новата версия. В противен случай, новата версия се добавя заедно с другите.
Примерно хранилище:

myrepo
|-- db
|-- packages
|-- 6e3549438bc246b86961b2e8c3469321ca22eabd0a6c487d086de7a43a0ef766.tar.xz
|-- 66b28e48161ba01ae25433b9ac4086a83b14d2ee49a62f2659c96514680ab6e8.tar.xz
|-- 99c934ad80bd9e49125523c414161e82716b292d4ed2f16bb977d6db7e13d9bc.tar.xz

Със съдържание на db:

glibc-2.31-2 6e3549438bc246b86961b2e8c3469321ca22eabd0a6c487d086de7a43a0ef766
zlib-1.1.15-8 66b28e48161ba01ae25433b9ac4086a83b14d2ee49a62f2659c96514680ab6e8
zlib-1.2.11-4 99c934ad80bd9e49125523c414161e82716b292d4ed2f16bb977d6db7e13d9bc

Примерен пакет:

zlib
|-- version # contains '1.2.11-3'
|-- tree
|...

Съдържание на хранилището след изпълнение на ./repo-add.sh myrepo zlib

```
myrepo
        |-- db
|-- packages
        |-- 6e3549438bc246b86961b2e8c3469321ca22eabd0a6c487d086de7a43a0ef766.tar.xz
        |-- 66b28e48161ba01ae25433b9ac4086a83b14d2ee49a62f2659c96514680ab6e8.tar.xz
        |-- b839547ee0aed82c74a37d4129382f1bd6fde85f97c07c5b705eeb6c6d69f162.tar.xz
|-- 99c934ad80bd9e49125523c414161e82716b292d4ed2f16bb977d6db7e13d9bc.tar.xz
```

Със съдържание на db:

```
glibc-2.31-2 6e3549438bc246b86961b2e8c3469321ca22eabd0a6c487d086de7a43a0ef766
zlib-1.1.15-8 66b28e48161ba01ae25433b9ac4086a83b14d2ee49a62f2659c96514680ab6e8
zlib-1.2.11-3 b839547ee0aed82c74a37d4129382f1bd6fde85f97c07c5b705eeb6c6d69f162
zlib-1.2.11-4 99c934ad80bd9e49125523c414161e82716b292d4ed2f16bb977d6db7e13d9bc
```

Зад. 59

Напишете shell скрипт, който приема 3 позиционни аргумента – две имена на файлове и име на директория. Примерно извикване: $ ./foo.sh foo.pwd config.cfg cfgdir/

В директорията cfgdir/ и нейните под-директории може да има файлове с имена завършващи на .cfg. За да са валидни, тези файлове трябва да съдържат редове само в следните формати (редовете започващи с # са коментари):

```
# internal laboratory
{ no-production };

{ volatile };

# meow configs
{ run-all; };
```

Във файла foo.pwd има описани потребителски имена (username) и MD5 хеш суми на паролите им, с по един запис на ред, в следният формат: username:password_hash

Също така, разполагате с команда pwgen, която генерира и извежда на STDOUT случайни пароли, и знаете, че поддържа следните два аргумента: $ pwgen [password_length] [number_of_passwords]Вашият скрипт трябва да валидира cfg файловете в директорията, и за тези, които не са валидни, да извежда на STDOUT името на файла и номерирани редовете, които имат проблем, в следния формат:

```
Error in filename.cfg:
Line 1:XXXX
Line 37:YYYY
```

където XXXX и YYYY е съдържанието на съответния ред.

За валидните файлове, скриптът трябва да:

- генерира config.cfg като обединение на съдържанието им
- името на файла, без частта .cfg дефинира потребителско име. Ако във файла с паролите не съществува запис за този потребител, то такъв да се добави и на стандартния изход да се изведе потребителското име и паролата (поне 16 символа) разделени с един интервал.

Зад. 60

Под конфигурационен файл ще разбираме файл, в който има редове от вида key=value, където key и value могат да се състоят от букви, цифри и знак “долна черта” (“_”). Освен това, във файла може да има празни редове; може да има произволен whitespace в началото и в края на редовете, както и около символа “=”. Също така са допустими и коментари в даден ред: всичко след символ “#” се приема за коментар.

Под <date> ще разбираме текущото време, върнато от командата date без параметри; под <user> ще разбираме името на текущият потребител.

Напишете shell скрипт set_value.sh, който приема 3 позиционни аргумента – име на конфигурационен файл, ключ (foo) и стойност (bar). Ако ключът:

- присъства във файла с друга стойност, скриптът трябва да:
  - да закоментира този ред като сложи # в началото на реда и добави в края на реда #edited at <date> by <user>
  - да добави нов ред foo = bar # added at <date> by <user> точно след стария ред
- не присъства във файла, скриптът трябва да добави ред от вида foo = bar # added at <date> by <user> на края на файла.

Примерен foo.conf:

```
# route description

from = Sofia
to = Varna          # my favourite city!
type = t2_1
```

Примерно извикване:

```
./set_value.sh foo.conf to Plovdiv
```

Съдържание на foo.conf след извикването:

```
# route description
```

from = Sofia
# to = Varna # my favourite city! # edited at Tue Aug 25 15:48:29 EEST 2020 by human
to = Plovdiv # added at Tue Aug 25 15:48:29 EEST 2020 by human
type = t2_1

Зад. 61
Разполагате с машина, на която е инсталиран специализиран софтуер, който ползва два потребителски акаунта – oracle и grid. Всеки от потребителите би трябвало да има environment променлива ORACLE_HOME, която указва абсолютен път до директория във формат /path/to/dir. В под- директория bin на зададената директория би трябвало да има изпълним файл с име adrci. Всеки от двата потребителя има собствена директория diag_dest, която е във вида /u01/app/потребител. Когато някой от потребителите изпълни неговото копие на командата adrci с параметър exec="show homes" може да получи на STDOUT един от следните два изхода:

- вариант 1: (неуспех):
  No ADR homes are set
- вариант 2: (успех):
  ADR Homes:
  diag/rdbms/orclbi/orclbi1
  diag/rdbms/orclbi/orclbi2

И в двата случая командата приключва с exit code 0. Ако командата се изпълни успешно, тя връща списък с един или повече ADR Homes, които са релативни имена на директории спрямо diag_dest на съответният потребител.
Напишете скрипт, който може да се изпълнява само от някой от тези два акаунта, и извежда на STDOUT размера в мегабайти и абсолютният път на всеки ADR Home.
Примерен изход:
0 /u01/app/oracle/diag/rdbms/orclbi/orclbi1
389 /u01/app/oracle/diag/rdbms/orclbi/orclbi2

Зад. 62
Разполагате с машина, на която е инсталиран специализиран софтуер, който ползва два потребителски акаунта – oracle и grid. За всеки от двата акаунта съществува директория, която ще наричаме diag_dest и тя е от вида /u01/app/потребител. Всеки от потребителите би трябвало да има environment променлива ORACLE_HOME, която указва абсолютен път до директория във формат /път/до/дир. В поддиректорията bin на зададената директория би трябвало да има изпълним файл с име adrci.
Всеки от потребителите може да подаде серия от подкоманди, които неговото копие на adrci да изпълни, като го извика с параметър exec="cmd1; cmd2; cmd3". Отделните подкоманди се разделят с точка и запетая (;). Командата adrci винаги приключва с exit code 0.
Нека дефинираме следните подкоманди:

- SET BASE – за да се гарантира правилна работа на командата adrci, при всяко нейно извикване първата подкоманда трябва да бъде от вида SET BASE diag_dest, където diag_dest е абсолютният път на съответната директория за дадения потребител
- SHOW HOMES – подкоманда SHOW HOMES извежда на STDOUT един от следните два изхода:
  - вариант 1: (неуспех): No ADR homes are set
  - вариант 2: (успех):
    ADR Homes:
    diag/rdbms/orclbi/orclbi1
    diag/rdbms/orclbi/orclbi2

Ако командата се изпълни успешно, тя връща списък с един или повече ADR Homes, които са релативни имена на директории спрямо diag_dest на съответният потребител.
От полученият списък с релативни пътища интересни за нас са само тези, които за име на директория на второ ниво имат една от следните директории: crs, tnslsnr, kfod, asm или rdbms.

- SET HOMEPATH – подкоманда SET HOMEPATH път задава активна работна директория, спрямо която ще се изпълняват последващи подкоманди в рамките на същото изпълнение на adrci; път е релативен път до директория, спрямо изхода на SHOW HOMES
- PURGE – подкоманда PURGE -AGE минути изтрива определени файлове в текущо активната работна директория, по-стари от дефинираното време в минути. Забележка: изтриват се само безопасни файлове, т.е. такива, чието изтриване няма да доведе до проблем. Дефиницията на безопасни файлове е извън обхвата на тази задача.

Напишете shell скрипт, който може да се изпълнява само от някой от указаните два акаунта и приема задължителен първи позиционен аргумент число (в часове, минимална стойност 2 часа). Скриптът трябва да почиства безопасните файлове по-стари от зададените часове в интересните ADR Home-ове на съответния потребител.

Зад. 63
Един от често използваните DNS сървъри е BIND9, при който описанието на DNS зоните обикновено стои в текстови файлове, наричани зонални файлове. За улеснение, в рамките на задачата, ще ползваме опростено описание на зоналните файлове.
Под whitespace разбираме произволна комбинация от табове и интервали.

Под FQDN разбираме низ, който има допустими символи малки латински букви, цифри и точка; не може да започва с точка, не може да има две или повече съседни точки, задължително завършва с точка.
Зоналните файлове съдържат ресурсни записи, по един на ред. Общият вид на даден ресурсен запис е <ключ> <TTL> <клас> <тип> <RDATA>, разделени с whitespace, например: astero.openfmi.net. 3600 IN A 185.117.82.99
Където:

- ключ (astero.openfmi.net.) – FQDN
- L (3600) – цифри; полето може да липсва
- клас (IN) - главни латински букви; класът винаги е IN
- тип (A) - главни латинки букви; някое от SOA, NS, A, AAAA
- RDATA (185.117.82.99) - данни на записа; различни за различните типове ресурсни записи; всичко след типа до
- края на реда.

Знакът точка-и-запетая ; е знак за коментар, и всичко след него до края на реда се игнорира.
Във всеки зонален файл трябва да има точно един SOA запис, и той трябва да е първият запис във файла. Пример за едноредов SOA запис:

```
openfmi.net. 3600 IN SOA nimbus.fccf.net. root.fccf.net. 2021041901 86400 7200 3024000 3600
```

RDATA-та на SOA запис се състои от два FQDN-а и пет числа, разделени с whitespace.
Въпреки, че горното е валиден SOA запис, за прегледност в зоналните файлове често се ползва следният синтаксис (многоредов SOA запис, еквивалентен на горния):

```
openfmi.net. 3600 IN SOA nimbus.fccf.net. root.fccf.net. (
2021041901 ; serial
86400 ; refresh
7200 ; retry
3024000 ; expire
3600 ; negative TTL
)
```

т.е., поредицата от числа се разбива на няколко реда, оградени в обикновенни скоби, и за всяко число се слага коментар какво означава.
Първото от тези числа (serial) представлява серийният номер на зоната, който трябва да се увеличава всеки път, когато нещо в зоналният файл се промени. Изключително важно е това число само да нараства, и никога да не намалява.
Един от често използваните формати за сериен номер, който показва кога е настъпила последната промяна в зоналния файл представлява число във вида YYYYMMDDTT, т.е., четири цифри за година, две цифри за месец, две цифри за дата и още две цифри за поредна промяна в рамките на описания ден. За последните две цифри (TT) има ограничение да са от 00 до 99 (естествено, така не може да има повече от 100 промени в рамките на един ден).
Приемаме, че конкретен сериен номер (точната поредица цифри) се среща само на едно място в зоналния файл.
Напишете шел скрипт, който по подадени имена на зонални файлове променя серийният номер в SOA записа на всеки файл по следният алгоритъм:

- ако датата в серийният номер е по-стара от днешната, новият сериен номер трябва да е от вида днешнадата00
- ако датата в серийният номер е равна на днешната, серийният номер трябва да се увеличи с единица.

Важат следните условия:

- скриптът трябва да може да обработва и едноредови, и многоредови SOA записи
- за всеки зонален файл, който не е успял да обработи, скриптът трябва да вади съобщение за грешка, което включва и името на зоналния файл.

Зад. 64
Напишете shell скрипт, който приема два позиционни параметъра – имена на файлове. Примерно извикване:

```
$ ./foo.sh input.bin output.h
```

Файлът input.bin е двоичен файл с елементи uint16_t числа, създаден на little-endian машина. Вашият скрипт трябва да генерира C хедър файл, дефиниращ масив с име arr, който:

- съдържа всички елементи от входния файл
- няма указана големина
- не позволява промяна на данните.

Генерираният хедър файл трябва да:

- съдържа и uint32_t променлива arrN, която указва големината на масива
- бъде валиден и да може да се #include-ва без проблеми от C файлове, очакващи да “виждат” arr и arrN.

За да е валиден един входен файл, той трябва да съдържа не повече от 524288 елемента. За справка, dump на съдържанието на примерен input.bin:

```
00000000: 5555 5655 5955 5a55 6555 6655 6955 6a55 UUVUYUZUeUfUiUjU
00000010: 9555 9655 9955 9a55 a555 a655 a955 aa55 .U.U.U.U.U.U.U.U
```

Зад. 65
Описание на формат на CSV (текстови) файл:

- CSV файлът представлява таблица, като всеки ред на таблицата е записан на отделен ред
- на даден ред всяко поле (колона) е разделено от останалите със запетая
- в полетата не може да присъства запетая, т.е. запетаята винаги е разделител между полета
- броят на полетата на всеки ред е константа

- първият ред във файла е header, който описва имената на колоните.

Разполагате с два CSV файла със следното примерно съдържание:

- base.csv:

```
unit name,unit symbol,measure
second,s,time
metre,m,length
kilogram,kg,mass
ampere,A,electric current
kelvin,K,thermodynamic temperature
mole,mol,amount of substance
candela,cd,luminous intensity
```

- prefix.csv:

```
prefix name,prefix symbol,decimal t
era,T,1000000000000
giga,G,1000000000
mega,M,1000000
mili,m,0.001 nano,n,0.000000001
```

Където смисълът на колоните е:

- за base.csv
  - unit name – име на мерна единица
  - unit symbol – съкратено означение на мерната единица
  - measure – величина
- за prefix.csv
  - prefix name – име на представка
  - prefix symbol – означение на представка
  - decimal – стойност

Във файловете може да има и други редове, освен показаните в примера. Приемаме, че файловете спазват описания по-горе формат, т.е. не е необходимо да проверявате формата.
Напишете shell скрипт, който приема три задължителни параметъра: число, prefix symbol и unit symbol.
Скриптът, ако може, трябва да извежда числото в основната мерна единица без представка, добавяйки в скоби величината и името на мерната единица.
Примерен вход и изход: $ ./foo.sh 2.1 M s
2100000.0 s (time, second)

Зад. 66

Съвременните компютри могат да влизат в различни режими за енергоспестяване (suspend) и излизат от този режим (wake-up) при настъпването на определени събития. Linux kernel предоставя специален виртуален файл /proc/acpi/wakeup, чрез който може да се проверява или променя настройката за "събуждане" при настъпване на различни събития. Тъй като този файл е интерфейс към ядрото, "четенето" от файла и "писането" във файла изглеждат различно.
За улеснение на задачата ще ползваме опростено описание на този файл. Под whitespace разбираме произволна комбинация от табове и интервали.
При четене от файла изходът представлява четири колони, разделени с whitespace, в полетата не може да има whitespace; първият ред е header на съответната колона.
Примерно съдържание на файла:

Device	S-state	Status	Sysfs node
GLAN	S4	*enabled	pci:0000:00:1f.6
XHC	S3	*enabled	pci:0000:00:14.0
XDCI	S4	*disabled	
LID	S4	*enabled	platform:PNP0C0D:00
HDAS	S4	*disabled	pci:0000:00:1f.3
RP01	S4	*enabled	pci:0000:00:1c.0

където:

- първата колона дава уникално име на устройство, което може да събуди машината, като името е ограничено до четири знака главни латински букви и цифри
- третата колона описва дали това устройство може да събуди машината. Възможните стойности са enabled/disabled, предхождани от звездичка
- втората и четвъртата колона ги игнорираме в рамките на задачата.

При записване на име на устройство във файла /proc/acpi/wakeup, неговото състояние се променя от disabled на enabled или обратно.
Например, ако /proc/acpi/wakeup изглежда както примера по-горе, при изпълнение на командата:
echo XHC > /proc/acpi/wakeup, третият ред ще се промени на:

XHC	S3	*disabled	pci:0000:00:14.0

При изпълнение на командата:
echo HDAS > /proc/acpi/wakeup, шестият ред ще се промени на:

HDAS S4 *enabled pci:0000:00:1f.3

Напишете shell скрипт, който при подаден първи параметър име на устройство (напр. LID) настройва съответното устройство да не може да събуди машината.

Зад. 67

Съвременните компютри могат да влизат в различни режими за енергоспестяване (suspend) и излизат от този режим (wake-up) при настъпването на определени събития. Linux kernel предоставя специален виртуален файл /proc/acpi/wakeup, чрез който може да се проверява или променя настройката за "събуждане" при настъпване на различни събития, в общия случай - при активност на някое устройство. Тъй като този файл е интерфейс към ядрото, "четенето" от файла и "писането" във файла изглеждат различно.

За улеснение на задачата ще ползваме опростено описание на този файл.

Под whitespace разбираме произволна комбинация от табове и интервали. При четене от файла изходът представлява четири колони, разделени с whitespace, в полетата не може да има whitespace; първият ред е header на съответната колона.

Примерно съдържание на файла:

Device	S-state	Status	Sysfs node
GLAN	S4	*enabled	pci:0000:00:1f.6
XHC	S3	*enabled	pci:0000:00:14.0
XDCI	S4	*disabled	
LID	S4	*enabled	platform:PNP0C0D:00
HDAS	S4	*disabled	pci:0000:00:1f.3
RP01	S4	*enabled	pci:0000:00:1c.0

където:

- първата колона дава уникално име на устройство, което може да събуди машината, като името е ограничено до четири знака главни латински букви и цифри
- третата колона описва дали това устройство може да събуди машината .Възможните стойности са enabled/disabled, предхождани от звездичка
- втората и четвъртата колона ги игнорираме в рамките на задачата.

При записване на име на устройство във файла /proc/acpi/wakeup, неговото състояние се променя от disabled на enabled или обратно. Например, ако файлът изглежда както примера по-горе, при запис на XHC в него, третият ред ще се промени на:

XHC S3 *disabled pci:0000:00:14.0

При запис на HDAS, шестият ред ще се промени на:

HDAS S4 *enabled pci:0000:00:1f.3

Дефиниран е формат на конфигурационен файл, който описва желан комплект от настройки на wakeup събития.

Примерен файл:

```
# comment bar
GLAN disabled
LID enabled # comment foo
```

където:

- знакът диез (#) е знак за коментар до края на реда
- редовете би трябвало да са комбинация от име на устройство и желаното състояние на настройката за събуждане при събития от това устройство, разделени с whitespace.

Напишете скрипт, който при подаден първи параметър име на конфигурационен файл в горния формат осигурява исканите настройки за събуждане. Ако във файла има ред за устройство, което не съществува, скриптът да извежда предупреждение.