Задачи по ОС

02-a-0100.txt:

Направете копие на файла /etc/passwd във вашата home директория под името my_passwd.

Решение: cp /etc/passwd ~/my_passwd

02-a-0500.txt:

Направете директория practice-test в home директорията ви. Вътре направете директория test1. Можете ли да направите тези две неща наведнъж? Разгледайте нужната man страница. След това създайте празен файл вътре, който да се казва test.txt, преместете го в practice-test чрез релативни пътища.

Решение: mkdir -p ~/practice-test/test1

touch ~/practice-test/test1/test.txt

mv ~/practice-test/test1/test.txt ~/practice-test

02-a-0600.txt:

Създайте директорията /tmp/os_<ФН>/practice/01/, където <ФН> е вашият факултетен номер. Създайте 3 файла в нея - f1, f2, f3. Копирайте файловете f1, f2, f3 от директорията /tmp/os_<ФН>/practice/01/ в директория dir1, намираща се във вашата home директория. Ако нямате такава, създайте я.

Решение: mkdir -p /tmp/os_62372/practice/01; touch f1 f2 f3
mkdir dir1
cp /tmp/os_62372/practice/01/f{1,2,3} dir1

02-a-0601.txt:

Нека файлът f2 бъде преместен в директория dir2, намираща се във вашата home директория и бъде преименуван на numbers.

Решение: mv ~/dir1/f2 ~/dir2/numbers

02-a-1200.txt:

Отпечатайте имената на всички директории в директорията /home.

Решение: find /home -type d

02-a-4000.txt:

Създайте файл permissions.txt в home директорията си. За него дайте единствено - read права на потребителя създал файла, write and exec на групата, read and exec на всички останали. Направете го и с битове, и чрез "буквички".

Решение: touch permissions.txt;
chmod 435 permissions.txt
chmod u=r,g=wx,o=rx permissions.txt

02-a-4100.txt:

За да намерите какво сте правили днес: намерете всички файлове в home директорията ви, които са променени в последния 1 час.

Решение: find . -mmin -60 **(60 преди точно 60 мин, -60 за последните 60 мин)**

02-a-5000.txt:

Копирайте /etc/services в home директорията си. Прочетете го с командата cat. (Ако този файл го няма, прочетете с cat произволен текстов файл напр. /etc/passwd)

Решение: cp /etc/services .
cat services

02-a-5200.txt:

Създайте symlink на файла /etc/passwd в home директорията ви (да се казва например passwd_symlink).

Решение: ln -s /etc/passwd ~/passwd_symlink

02-a-5400.txt:

Изведете всички обикновени ("regular") файлове, които /etc и нейните преки поддиректории съдържат.

Решение: find /etc -type f

02-a-5401.txt:

Изведете само първите 5 реда от /etc/services.

Решение: head -n 5 /etc/services

02-a-5402.txt:

Изведете всички обикновени ("regular") файлове, които само преките поддиректории на /etc съдържат.

Решение: find /etc -type f -maxdepth 2 ???

02-a-5403.txt:

Изведете всички преки поддиректории на /etc.

Решение: find /etc -maxdepth 1 -type d

02-a-5500.txt:

Създайте файл, който да съдържа само последните 10 реда от изхода на 02-a-5403.

Решение: find /etc -maxdepth 1 -type d | tail -n 10 > newfile.txt

02-a-5501.txt:

Изведете обикновените файлове по-големи от 42 байта в home директорията ви.

Решение: find . -type f -size +42c -exec ls -lh {} \;

02-a-5504.txt:

Изведете всички обикновени файлове в директорията /tmp които са от вашата група, които имат write права за достъп за група или за останалите(o=w).

02-a-5505.txt:

Изведете всички файлове, които са по-нови от /tmp/os2018/practice/01/f1 (би трябвало да е създаден като част от по-ранна задача).

Решение: find -type f -newer [име на файла]

02-a-5506.txt:

Изтрийте файловете в home директорията си по-нови от /tmp/os2018/practice/01/f2 (подайте на rm опция -i за да може да изберете само тези които искате да изтриете).

Решение: find -type f -newer practice/01/f1 -exec rm -i {} \;

02-a-6000.txt:

Намерете файловете в /bin, които могат да се четат, пишат и изпълняват от всички.

Решение: find /bin -perm 777 -exec ls -lh {} \;

02-a-8000.txt:

Копирайте всички файлове от /bin, които могат да се четат, пишат и изпълняват от всички, в bin2 директория в home директорията ви. Направете такава, ако нямате.

Решение: find /bin -perm 777 -exec cp -t ~/bin2/ {} +

find /bin -perm 777 -exec cp {} ~/bin2/ \;

02-a-9000.txt:

от предната задача: когато вече сте получили bin2 с команди, архивирайте всички команди вътре, които започват с b в архив, който се казва b_start.tar. (командата, която архивира е tar -c -f <файл1> <файл2>...)

Решение: tar -c -f b_start.tar ~/bin2/[b]*

02-a-9500.txt:

Използвайки едно извикване на командата find, отпечатайте броя на редовете във всеки обикновен файл в /etc директорията.

Решение: find /etc -type f -exec wc -l {} +

02-b-4000.txt:

Копирайте най-големия файл от тези, намиращи се в /etc, в home директорията си.

03-a-0200.txt:

Сортирайте /etc/passwd лексикографски по поле UserID.

Решение: cat /etc/passwd | tr : “ ” | sort -k 5 (k – номер на поле)

03-a-0201.txt:

Сортирайте /etc/passwd числово по поле UserID. (Открийте разликите с лексикографската сортировка). Решение: cat /etc/passwd | tr : “ ” | sort -n -k 5 (k – номер на поле)

03-a-0210.txt:

Изведете само 1-ва и 5-та колона на файла /etc/passwd спрямо разделител ":".

Решение: cut -d : -f 1,5 /etc/passwd

03-a-0211.txt:

Изведете съдържанието на файла /etc/passwd от 2-ри до 6-ти символ.

Решение: cut -c 2-6 /etc/passwd

03-a-0212.txt:

Отпечатайте потребителските имена и техните home директории от /etc/passwd.

Решение: cut -d : -f 5-6 /etc/passwd

03-a-0213.txt:

Отпечатайте втората колона на /etc/passwd, разделена спрямо символ '/'.

Решение: cut -d / -f 2 /etc/passwd **?**

03-a-1500.txt:

Изведете броя на байтовете в /etc/passwd. Решение: wc -c /etc/passwd
Изведете броя на символите в /etc/passwd. Решение: wc -m /etc/passwd
Изведете броя на редовете в /etc/passwd. Решение: wc -l /etc/passwd

03-a-2000.txt:

С отделни команди, извадете от файл /etc/passwd:
- първите 12 реда Решение: head -n 12 /etc/passwd | cat -n
- първите 26 символа Решение: head -c 26 /etc/passwd ?
- всички редове, освен последните 4 Решение: head -n -4 /etc/passwd
- последните 17 реда Решение: tail -n 17 /etc/passwd | cat -n
- 151-я ред (или друг произволен, ако нямате достатъчно редове) Решение: head -n 151 /etc/passwd | tail -n +151
- последните 4 символа от 13-ти ред (символът за нов ред не е част от реда)
Решение: head -n 13 /etc/passwd | tail -n +13 | tail -c 5

03-a-3000.txt:

Запаметете във файл в своята home директория резултатът от командата `df -P` (файл с име df_result). Напишете команда, която извежда на екрана съдържанието на този файл, без първия ред (хедъра), сортирано по второ поле (numeric).

Решение: tail -n +2 df_result | sort -n -k2 (k2 – второ поле)

03-a-3100.txt:

Запазете само потребителските имена от /etc/passwd във файл users във вашата home директория.

Решение: (не е така ама горе-долу става) cut -d : -f 5 /etc/passwd | cut -d , -f 1

03-a-3500.txt:

Изпишете всички usernames от /etc/passwd с главни букви.

Решение: cat /etc/passwd | cut -d , -f 5 | grep "@" | cut -d @ -f 1 | tr a-z A-Z

03-a-5000.txt:

Изведете реда от /etc/passwd, на който има информация за вашия потребител.

Решение: cat /etc/passwd | grep "Кристияна Николова"
Изведедете този ред и двата реда преди него.

Решение: cat /etc/passwd | grep -B 2 "Кристияна Николова"
Изведете този ред, двата преди него, и трите след него.

Решение: cat /etc/passwd | grep -A 3 -B 2 "Кристияна Николова"
Изведете *само* реда, който се намира 2 реда преди реда, съдържащ информация за вашия потребител.

Решение: cat /etc/passwd | grep -A 3 -B 2 "Кристияна Николова" | head -n 1

03-a-5001.txt:

Изведете колко потребители не изпозват /bin/bash за login shell според /etc/passwd
(hint: 'man 5 passwd' за информация какъв е форматът на /etc/passwd)

Решение: cat -n /etc/passwd | cut -d : -f 7 | grep -v "/bin/bash" | cat -n

cat -n /etc/passwd | cut -d : -f 7 | grep -v "/bin/bash" | wc -l

03-a-5002.txt:

Изведете само имената на хората с второ име по-дълго от 6 (>6) символа според /etc/passwd

Решение: cat /etc/passwd | cut -d : -f 5 | cut -d , -f 1 | cut -d ' ' -f 2 | grep -E '(\w{7,})'

03-a-5003.txt:

Изведете имената на хората с второ име по-късо от 8 (<=7) символа според /etc/passwd // !(>7) = ?

?Решение: cat /etc/passwd | cut -d : -f 5 | cut -d , -f 1 | cut -d ' ' -f 2 | egrep '^.{,7}$'

cat /etc/passwd | cut -d : -f 5 | cut -d , -f 1 | grep " " | cut -d ' ' -f 2 | egrep '^.{,7}$'

03-a-5004.txt:

Изведете целите редове от /etc/passwd за хората от 03-a-5003

03-a-6300.txt:
Копирайте <РЕПО>/exercises/data/emp.data във вашата home директория.

Посредством awk, използвайки копирания файл за входнни данни, изведете:

- общия брой редове

- третия ред

- последното поле от всеки ред

- последното поле на последния ред

- всеки ред, който има повече от 4 полета

- всеки ред, чието последно поле е по-голямо от 4

- общия брой полета във всички редове

- броя редове, в които се среща низът Beth

- най-голямото трето поле и редът, който го съдържа

- всеки ред, който има поне едно поле

- всеки ред, който има повече от 17 знака

- броя на полетата във всеки ред и самият ред

- първите две полета от всеки ред, с разменени места

- всеки ред така, че първите две полета да са с разменени места

- всеки ред така, че на мястото на първото поле да има номер на реда

- всеки ред без второто поле

- за всеки ред, сумата от второ и трето поле

- сумата на второ и трето поле от всеки ред

03-b-0300.txt:

Намерете само Group ID-то си от файла /etc/passwd.

Решение: grep "Кристияна Николова" /etc/passwd | cut -d : -f 4

03-b-3400.txt:

Колко коментара има във файла /etc/services ? Коментарите се маркират със символа #, след който всеки символ на реда се счита за коментар.

Решение: grep -c ‘#’ /etc/services (-c извежда броя на редовете)

03-b-3500.txt:

Колко файлове в /bin са 'shell script'-ове? (Колко файлове в дадена директория са ASCII text?)

Решение: find /bin -name "*.sh"

03-b-3600.txt:

Направете списък с директориите на вашата файлова система, до които нямате достъп. Понеже файловата система може да е много голяма, търсете до 3 нива на дълбочина.

03-b-4000.txt:

Създайте следната файлова йерархия в home директорията ви:
dir5/file1
dir5/file2
dir5/file3

Посредством vi въведете следното съдържание:
file1:
1
2
3

file2:
s
a
d
f

file3:
3
2
1
45
42
14
1
52

Изведете на екрана:

* статистика за броя редове, думи и символи за всеки един файл –
  Решение: wc -l -w -c file1 file2 file3
* статистика за броя редове и символи за всички файлове
  Решение: wc -l -c file1 file2 file3
* общия брой редове на трите файла
  Решение: wc -l file{1,2,3} | tail -n 1

03-b-4001.txt:
Във file2 (inplace) подменете всички малки букви с главни.
Решение: sed -i -r 's/(.*)/\U\1/g' file2 (U- прави буквите от малки на главни, L – прави буквите от главни на малки)

03-b-4002.txt:
Във file3 (inplace) изтрийте всички "1"-ци.
Решение: sed -i '/^1$/d' file3

03-b-4003.txt:
Изведете статистика за най-често срещаните символи в трите файла.
Решение: cat file{1,2,3} | tr -c '[:alnum:]' '[\n*]' | sort | uniq -c | sort -nr

(Задължително сортиране преди uniq)

03-b-4004.txt:

Направете нов файл с име по ваш избор, чието съдържание е конкатенирани съдържанията на file{1,2,3}.
Решение: cat file{1,2,3} > file4

03-b-4005.txt:

Прочетете текстов файл file1 и направете всички главни букви малки като запишете резултата във file2.
Решение: sed -i -r 's/(.*)/\L\1/g' file1 >> file2

03-b-5200.txt:

Намерете броя на символите, различни от буквата 'a' във файла /etc/passwd

Решение: sed 's/\(.\)/\1\n/g' /etc/passwd | grep -v 'a' | sort | uniq | wc -l

03-b-5300.txt:

Намерете броя на уникалните символи, използвани в имената на потребителите от /etc/passwd.

Решение: cat /etc/passwd | cut -d : -f 5 | cut -d , -f 1 | grep " " > names

sed 's/\(.\)/\1\n/g' names | sort | uniq | wc -l

03-b-5400.txt:

Отпечатайте всички редове на файла /etc/passwd, които не съдържат символния низ 'ов'.

Решение: cat /etc/passwd | grep -v "ов"

03-b-6100.txt:

Отпечатайте последната цифра на UID на всички редове между 28-ми и 46-ред в /etc/passwd.

Решение: cat /etc/passwd | cut -d : -f 3 | cat -n | tail -n +28 | head -n 19 | cat /etc/passwd | cut -d : -f 3 | cat -n | tail -n +28 | head -n 19 | rev | cut -c 1

Решение2: cat -n /etc/passwd | cut -d : -f 3 | cat -n | head -n 46 | tail -n +28 | rev | cut -c 1

03-b-6700.txt:

Отпечатайте правата (permissions) и имената на всички файлове, до които имате read достъп, намиращи се в директорията /tmp. (hint: 'man find', вижте -readable).

?Решение: cd /tmp | find -readable | ls -lh

Решение2: find /tmp -readable -exec ls -lh {} \;

03-b-6900.txt:

Намерете имената на 10-те файла във вашата home директория, чието съдържание е редактирано най-скоро. На първо място трябва да бъде най-скоро редактираният файл. Намерете 10-те най-скоро достъпени файлове. (hint: Unix time)
Решение: ls -lht | head -n 10

03-b-7000.txt:

да приемем, че файловете, които съдържат C код, завършват на `.c` или `.h`.
Колко на брой са те в директорията `/usr/include`?
Решение: find /usr/include -iname "*.c" -o -iname "*.h" | wc -l **(-o стои за or)**
Колко реда C код има в тези файлове?
Решение: find /usr/include -name "*.c" | wc -l

03-b-7500.txt:

Даден ви е ASCII текстов файл - /etc/services. Отпечатайте хистограма на 10-те най-често срещани думи.
Дума наричаме непразна последователност от букви. Не правим разлика между главни и малки букви.
Хистограма наричаме поредица от редове, всеки от които има вида:
<брой срещания> <какво се среща толкова пъти>
Решение: cat /etc/services | tr [:space:] ' ' | tr ' ' '\n' | sort | uniq -c | sort -nr

03-b-8000.txt:

Вземете факултетните номера на студентите (описани във файла /srv/os/mypasswd.txt) от СИ и ги запишете във файл si.txt сортирани.
Студент е част от СИ, ако home директорията на този потребител (според /srv/os/mypasswd.txt) се намира
в /home/SI директорията.

Решение: cat /srv/os/mypasswd.txt | grep "/home/SI" | cut -c 2-6 | grep "[0-9]" | sort -n > si.txt
Решение: cat mypasswd.txt | grep "/home/SI" | grep "^s" | cut -d : -f 1 | cut -c 2- | sort -n > si.txt

03-b-8500.txt:

За всяка група от /etc/group изпишете "Hello, <група>", като ако това е вашата група, напишете "Hello, <група> - I am here!".

Решение: sed 's/^/Hello, /' /etc/group | sed -r "s/($(id -gn))/\1 - this is me\!/" | cut -d : -f 1

03-b-8600.txt:

Shell Script-овете са файлове, които по конвенция имат разширение .sh. Всеки такъв файл започва с "#!<interpreter>" , където <interpreter> указва на операционната система какъв интерпретатор да пусне (пр: "#!/bin/bash", "#!/usr/bin/python3 -u").

Намерете всички .sh файлове в директорията `/usr` и нейните поддиректории, и проверете кой е най-често използваният интерпретатор.

03-b-8700.txt:

1. Изведете GID-овете на 5-те най-големи групи спрямо броя потребители, за които съответната група е основна (primary).

2. (*) Изведете имената на съответните групи. Hint: /etc/passwd

03-b-9000.txt:

Направете файл eternity. Намерете всички файлове, намиращи се във вашата home директория и нейните поддиректории, които са били модифицирани в последните 15мин (по възможност изключете .). Запишете във eternity името (път) на файла и времето (unix time) на последната промяна.
Решение: find . -type f -mmin -15 -exec ls -lh {} \; | cut -d " " -f 6-9 > ~/eternity
Решение 2: find -mindepth 1 -mmin -15 -printf "%p %T@\n" > eternity
03-b-9050.txt:

Копирайте файл /srv/os/population.csv във вашата home директория.
Решение: cp /srv/os/population.csv ~/

03-b-9051.txt:
Използвайки файл population.csv, намерете колко е общото население на света през 2008 година. А през 2016?
Решение: grep ",2008," population.csv | sed 's/, / /' | cut -d , -f 4 | paste -sd+ | bc
grep ",2016," population.csv | sed 's/, / /' | cut -d , -f 4 | paste -sd+ | bc
Решение2: tail -n +2 population.csv | egrep "^([^,]*,){2}2008" | cut -d , -f 4 | awk 'BEGIN {cnt = 0} {cnt+=$1} END {print cnt}'
Решение3: awk -F ‘,’ ‘$3 == 2008 { sum += &4 } END { print sum }’ population.csv
Решение4: cat population.csv | grep "2008" | rev | cut -d , -f 1 | rev | paste -sd+ | bc
cat population.csv | grep "2016" | rev | cut -d , -f 1 | rev | paste -sd+ | bc

03-b-9052.txt:

Използвайки файл population.csv, намерете през коя година в България има най-много население.
Решение: grep "Bulgaria" population.csv | cut -d , -f 3-4 | tr "," " " | sort -nr -k 2 | head -n 1 | cut -d " " -f 1

03-b-9053.txt:

Използвайки файл population.csv, намерете коя държава има най-много население през 2016. А коя е с най-малко население? (Hint: Погледнете имената на държавите)

Решение: cat population.csv | sed 's/,/ /' | grep ",2016" | sort -nr -t ',' -k 3 | head -n 1

cat population.csv | sed 's/,/ /' | grep ",2016" | sort -nr -t ',' -k 3 | tail -n 1

Решение2: cat population.csv | grep "2016" | rev | cut -d , -f1,4- | sed 's/,/:/' | rev | sort -t : -nr -k2 | head -n 1 | tr ':' ' '

cat population.csv | grep "2016" | rev | cut -d , -f1,4- | sed 's/,/:/' | rev | sort -t : -nr -k2 | tail -n 1 | tr ':' ' '

03-b-9054.txt:

Използвайки файл population.csv, намерете коя държава е на 42-ро място по население през 1969. Колко е населението й през тази година?

Решение: cat population.csv | grep "1969" | rev | cut -d , -f1,4- | sed 's/,/:/' | rev | sort -t : -nr -k2 | head -n 42 | tail -n 1 | sed 's/:/ /'

03-b-9100.txt:

В home директорията си изпълнете командата `curl -o songs.tar.gz "http://fangorn.uni-sofia.bg/misc/songs.tar.gz"`

Решение: curl -o songs.tar.gz "http://fangorn.uni-sofia.bg/misc/songs.tar.gz"

03-b-9101.txt:

Да се разархивира архивът songs.tar.gz в папка songs във вашата home директорията.

Решение: mkdir ~/songs; cd songs; tar -xvf ~/songs.tar.gz

03-b-9102.txt:

Да се изведат само имената на песните.

Решение: find . -type f | sed 's/.//' | sed 's/\///' | cut -d . -f 1 | cut -d "(" -f 1 | cut -d "-" -f 2 | cut -d " " -f 2-

Решение2: find ./songs -mindepth 1 | cut -d - -f 2 | sed 's/ //' | cut -d '(' -f 1

03-b-9103.txt:

Имената на песните да се направят с малки букви, да се заменят спейсовете с долни черти и да се сортират.

Решение: find . -type f | sed 's/.//' | sed 's/\///' | cut -d . -f 1 | cut -d "(" -f 1 | cut -d "-" -f 2 | cut -d " " -f 2- | tr A-Z a-z | sed 's/ *$//' | sed 's/ /_/g' | sort

Решение2: find ./songs -mindepth 1 | cut -d - -f 2 | sed 's/ //' | cut -d '(' -f 1 | tr A-Z a-z | sed 's/ *$//' | sed 's/ /_/g' | sort

03-b-9104.txt:

Да се изведат всички албуми, сортирани по година.

Решение: find . -type f | cut -d - -f 2 | cut -d . -f 1 | cut -d "(" -f2 | cut -d ")" -f1 | sort | uniq | tr ', ' ' -' | sort -nr -k 2 | tr ' -' ', '

Решение2: find ./songs -mindepth 1 | cut -d '(' -f 2 | cut -d ')' -f 1 | sed 's/, /,/' | sort -nr -t , -k 2

03-b-9105.txt:

Да се преброят/изведат само песните на Beatles и Pink.

Решение: find . -type f | egrep 'Beatles|Pink' | sed 's/.//' | sed 's/\///' | cut -d . -f 1 | cut -d "(" -f 1 | cut -d "-" -f 2 | cut -d " " -f 2-

Решение2: find ./songs -mindepth 1 | egrep 'Beatles|Pink -' | cut -d - -f 2 | sed 's/ //' | cut -d '(' -f 1

03-b-9106.txt:

Да се направят директории с имената на уникалните групи. За улеснение, имената от две думи да се напишат слято:
Beatles, PinkFloyd, Madness

03-b-9200.txt:

Напишете серия от команди, които извеждат детайли за файловете и директориите в текущата директория, които имат същите права за достъп както най-големият файл в /etc директорията.

Решение: find . -perm $(find /etc -type f 2>/dev/null | xargs -I {} stat -c "%s %a" {} | sort -n | tail -n 1 | cut -d ' ' -f 2)| xargs ls -l

03-b-9300.txt:

Дадени са ви 2 списъка с email адреси - първият има 12 валидни адреса, а вторията има само невалидни. Филтрирайте всички адреси, така че да останат само валидните. Колко кратък регулярен израз можете да направите за целта?

Валидни email адреси (12 на брой):
email@example.com
firstname.lastname@example.com
email@subdomain.example.com
email@123.123.123.123
1234567890@example.com
email@example-one.com
_______@example.com
email@example.name
email@example.museum
email@example.co.jp
firstname-lastname@example.com
unusually.long.long.name@example.com

Невалидни email адреси:
#@%^%#$@#$@#.com
@example.com
myemail
Joe Smith <email@example.com>
email.example.com
email@example@example.com
.email@example.com
email.@example.com
email..email@example.com
email@-example.com
email@example..com
Abc..123@example.com
(),:;<>[\]@example.com
just"not"right@example.com
this\ is"really"not\allowed@example.com

Допълнителна зад:

Изведете името и пермишъните на файла съдържащ най-много редове съдържащи най-често срещания символ различен от ‘ ‘ в съдържанието на всички ‘shell script’-ове намиращи се в /bin.

Решение: find /bin -type f -exec grep -c -H $(file /bin/* | grep 'shell script' | cut -d : -f 1 | xargs cat | sed -E 's/(.)/\1\n/g' | sort | grep -v ' ' | uniq -c | sort -nr -k1 | head -n 1 | awk '{print $2}') {} \; | sort -t : -nr -k2 | head -n 1 | cut -d : -f 1 | xargs stat -c "%n %a"

Решение2: find /bin -type f | xargs file | grep 'shell script' | cut -d : -f 1 | xargs grep -c $(file /bin/* | grep 'shell script' | cut -d : -f 1 | xargs cat | sed -E 's/(.)/\1\n/g' | sort | grep -v ' ' | uniq -c | sort -nr -k1 | head -n 1 | awk '{print $2}') | sort -t : -nr -k2 | head -n 1 | cut -d : -f 1 | xargs stat -c "%n %a"

04-a-5000.txt:

Намерете командите на 10-те най-стари процеси в системата.
Решение: ps -ef --sort=start_time | head -n 11 ; ps -e -o cmd --sort=start_time | head -n 11

04-a-6000.txt:

Намерете PID и командата на процеса, който заема най-много виртуална памет в системата.
Решение: ps -o pid,cmd --sort=-vsz ax | head -n 2

ps -eo pid,cmd --sort=vsz | tail -1

04-a-6300.txt:

Изведете командата на най-стария процес
Решение: ps -e -o cmd --sort=start_time | head -n 2

ps -eo cmd= --sort=start_time | head -1

04-b-5000.txt:

Намерете колко физическа памет заемат всички процеси на потребителската група root.

Решение: sum=0;
while read line; do sum=$(($sum + $line)); done < <(ps -e -u root -o rss); echo $sum

Решение2: ps -e -g students -o drs | awk '{m+=$1}END{print m}'

04-b-6100.txt:

Изведете имената на потребителите, които имат поне 2 процеса, чиято команда е vim (независимо какви са аргументите й)

Решение: ps -aux | egrep vim | cut -d ' ' -f 1 | sort | uniq -c | egrep '^2.*'

ps -eo user,comm | awk '$2~/vim$/' | cut -d' ' -f1 | uniq -d

04-b-6200.txt:

Изведете имената на потребителите, които не са логнати в момента, но имат живи процеси

Решение: who | cut -d ‘ ‘ -f 1 | sort | uniq > onlineUsers
ps -aux | cut -d ‘ ‘ -f 1 | sort | uniq > activeProcesses
cat activeProcesses onlineUsers | sort | uniq -c | sed 's/\s*//' | egrep '^2.*'

04-b-7000.txt:

Намерете колко физическа памет заемат осреднено всички процеси на потребителската група root. Внимавайте, когато групата няма нито един процес.

Решение: ps -e -g root -o rss= | awk '{m+=$1}END{OFMT="%.3f"; if(NR==0){print 0} else {print m/NR}}'
04-b-8000.txt:

Намерете всички PID и техните команди (без аргументите), които нямат tty, което ги управлява. Изведете списък само с командите без повторения.

Решение: ps -eo tty,comm,pid | grep "^?" | tr -s ' ' | cut -d ' ' -f 2 | sort | uniq

04-b-9000.txt:

Да се отпечатат PID на всички процеси, които имат повече деца от родителския си процес.

Решение:

```
#!/bin/bash
# 04-b-9000 1st solution
# github.com/andy489

function count_children() {
        ps --ppid="${1}" | head +2 | wc -l
}

ps -eo pid=,ppid= | while read PID PPID; do
        [ $(count_children "${PID}") -ge $(count_children "${PPID}") ] &&
echo "${PID}"
done 2>/dev/null
```

05-a-2000.txt:

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

Решение: PS1=” “; PS1='${USER}@${HOSTNAME}:~$ '

05-a-2100.txt:

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.txt:

Направете си ваш псевдоним (alias) на полезна команда.

05-b-2000.txt:

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

Решение: #!/bin/bash
echo "Enter string: "

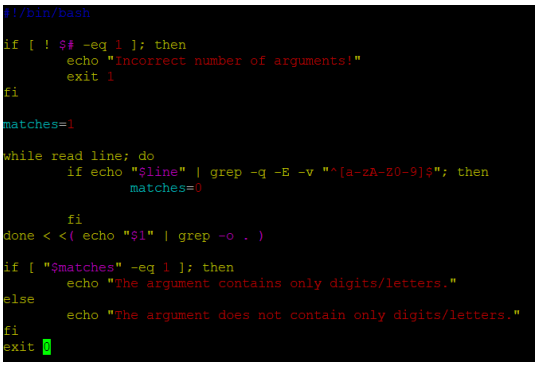
read line

echo "Hello, $line"

exit 0

05-b-2800.txt: в папка scripts имам още решения

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



```
#!/bin/bash

if [ ! $# -eq 1 ]; then
        echo "Incorrect number of arguments!"
        exit 1
fi

matches=1

while read line; do
        if echo "$line" | grep -q -E -v "^[a-zA-Z0-9]$"; then
                matches=0

        fi
done < <( echo "$1" | grep -o . )

if [ "$matches" -eq 1 ]; then
        echo "The argument contains only digits/letters."
else
        echo "The argument does not contain only digits/letters."
fi
exit 0
```

05-b-3100.txt:

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.

Решение: #!/bin/bash

echo "Enter username: "

read line

who | grep "^$line" | wc -l

exit 0

Решение:

echo "Please enter your username: "

read username

echo "Number of ${username}'s active sessions is: $(who | egrep -o ${username} | wc -l)."

exit 0

05-b-3200.txt:

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.

Решение: #!/bin/bash

echo "Enter dir path: "

read line

find $line -type f,d 2>&1 | echo "Number of files and directories: $(wc -l)"

exit 0

Решение:

```
echo "Please enter full dir name: "

read dir

echo "Number of all files and directories in ${dir} is: $(find ${dir} -type f,d 2> /dev/null | wc -l)."

exit 0
```

05-b-3300.txt:

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

Решение:

```
#!/bin/bash

echo "Enter 3 file names: "

read file1 file2 file3

paste -d '\n' $file1 $file2 | sort > $file3

exit 0
```

05-b-3400.txt:

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.

Решение :

```
#!/bin/bash

echo "Enter file name and string: "

read file string

grep -q "$string" $file && echo "yes" || echo "no"

exit 0
```

Решение:

```
read -p "Enter full path name of file: " FILE_NAME

read -p "Enter string to match: " EXP

grep -qsF "${EXP}" "${FILE_NAME}" && echo "yes" || echo "no"

exit 0
```

05-b-4200.txt:

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

Решение:
```
#!/bin/bash

awk 'BEGIN{level=0;maxLevel=0;}\
      /{/{level++}\
      /}/{level--}\
      (level>maxLevel){maxLevel=level}\
   END{print "The deepest nesting is " maxLevel ".";}' $1
```
05-b-4400.txt:

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.

?Решение:
```
#!/bin/bash

echo "Enter directory and name: "

read dir name

if [ ! -z "$name" ];

then

      mkdir ./$name

      find $dir -type f -mmin -45 -exec cp {} ./$name/ \;

else

      mkdir ./$(date +"%d-%m-%Y")
```

```
    find $dir -type f -mmin -45 -exec cp {} ./$(date +"%d-%m-%Y") \;
fi

exit 0
```

05-b-4500.txt:

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

?Решение:
```
#!/bin/bash
echo "Enter uid: "
read user
while [ false ]
do
    if who -u | grep -q "^$user "; then
    echo "$user is logged in"
    exit 0
    else
        sleep 1
fi
done
exit 0
```

05-b-4600.txt:

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
```

Решение:

```
#!/bin/bash
echo "Enter 3 numbers: "
read num left right

if ! [ "$num" -eq "$num" ] 2> /dev/null
then
	exit 3
elif ! [ "$left" -eq "$left" ] 2> /dev/null
then
	exit 3
elif ! [ "$right" -eq "$right" ] 2> /dev/null
then
	exit 3
fi

if [[ "$left" -gt "$right" ]]
then
	exit 2
fi

if [[ "$num" -gt "$right" ]] || [[ "$num" -lt "$left" ]]
then
	exit 1
fi

if [[ "$num" -ge "$left" ]] && [[ "$num" -le "$right" ]]
then
	exit 0
fi
exit 0
```

05-b-4700.txt:

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
```

Решение:

```
#!/bin/bash

echo "Enter a number and a separator: "
read num sep
```

```
if [ -z "$sep" ];
then
	echo $num | sed ':a;s/\B[0-9]\{3\}\>/ &/;ta'
	exit 0

fi

echo $num | sed ':a;s/\B[0-9]\{3\}\>/'$sep'&/;ta'

exit 0
```

05-b-4800.txt:

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.

Решение:

```
#!/bin/bash


if [ $# -ne 2 ]; then

	echo "Invalid number of arguments!"

	exit 1

fi


FILE="${1}"

DIR="${2}"


if [ ! -f "${FILE}" ]; then

	echo "First argument is not a file!"

	exit 2

elif [ ! -r "${FILE}" ]; then

	echo "File is not readable!"

	exit 3

fi
```

```
if [ ! -d "${DIR}" ]; then
	echo "Second argument is not a directory!"
	exit 4
elif [ ! -r "${DIR}" ]; then
	echo "Directory is not readable!"
	exit 5
fi

SEARCHED_HASH=$(md5sum "${FILE}" | awk '{print $1}')

CNT=0

while read CUR_HASH FILE_PATH; do
	if [ "${CUR_HASH}" = "${SEARCHED_HASH}" ]; then
		echo "$(basename ${FILE_PATH})"
		CNT=$(( ++CNT ))
	fi
done < <(find "${DIR}" -type f 2>/dev/null -print0 | xargs -0 -I {} md5sum {})

if [ $CNT -eq 0 ]; then
	echo "No copies found!"
else
	echo "Found total $CNT copies!"
fi
```

05-b-5500.txt:

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>
```

Решение:

```
#!/bin/bash

cat /etc/passwd | awk -F : 'BEGIN {printf("<table>\n <tr>\n <th>%s</th>\n <th>%s</th>\n
<th>%s</th>\n <th>%s</th>\n </tr>\n", "Username", "Group", "Login Shell", "GECKO") } { printf("
<tr>\n <td>%s</td>\n <td>%s</td>\n <td>%s</td>\n <td>%s</td>\n </tr>\n", $1, $4, $7, "GECKO") }
END { printf("</table>\n") }'
```

05-b-6600.txt:

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
```

Решение:

```
#!/bin/bash

if [ $# -ne 1 ]; then
	echo "Invalid number of arguments!"
	exit 1
fi
```

```
DIR="${1}"

if [ ! -d "${DIR}" ]; then
	echo "Argument is not a directory!"
	exit 2
fi

if [ ! -r "${DIR}" ]; then
	echo "Directory has no read permissions!"
	exit 3
fi

HASHES=$(mktemp)

find "${DIR}" -type f -maxdepth 1 -exec md5sum {} 2>/dev/null \; | sort > "${HASHES}"

cat "${HASHES}" | cut -d ' ' -f 1 | uniq -d | while read DUPLICATED_HASH; do
	grep "${DUPLICATED_HASH}" "${HASHES}" | awk '{print $2}' | tail -n +2 | xargs rm --
done

rm "${HASHES}"
```

Решение2:

```
#!/bin/bash

[ $# -eq 1 ] || exit 1

[ -d "${1}" ] || exit 2

[ -r "${1}" ] || exit 3

while read -d $'\n' LINE ; do
		find "${1}" -maxdepth 1 -type f -exec md5sum {} \; \
			| awk -v FIRST="${LINE}" '$1 == FIRST {print $2}' \
			| sort | tail -n +2 | xargs -I{} rm {}
done < <( find "${1}" -maxdepth 1 -type f -exec md5sum {} \; \
		| awk '{ print $1}' | sort | uniq -d )
```

05-b-6800.txt:

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)

Решение:

```
#!/bin/bash

echo "Enter directory name: "

read dir

find $dir -maxdepth 1 -not -path '*/\.*' -type f -exec stat -c "%n %s" {} \; | awk -F "$dir/" '{print $2}' | awk '{print $1" ""("$2" bytes)"}'

find $dir  -maxdepth 1 -not -path '*/\.*' -type d | while read -r dir2; do printf "%s " "$dir2"; find "$dir2" -type f,d | wc -l; done | tail -n +2 | awk -F "$dir/" '{print $2}' | awk '{print $1" ""("$2" entries)"}'

exit 0
```

05-b-7000.txt:

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.

Решение:

```
#!/bin/bash

if [ $# -eq 0 ];

then

   echo "Invalid number of arguments!"

   exit 1

fi

read -p "Insert searched string: " STRIN

while [ $# -ne 0 ];

do

   FILE_NAME="${1}"

   if [ ! -f "${FILE_NAME}" ];

   then

      echo "Invalid file name given as argument!"

   elif [ ! -r "${FILE_NAME}" ];
```

```
  then
    echo "File is not readable!"
  else
    CUR_MATCHES=$(grep -oc "${STRING}" "${FILE_NAME}")
    echo "In file \"${FILE_NAME}\", ${CUR_MATCHES} rows contains string \"${STRING}\""
  fi
  shift 1
done
```

05-b-7100.txt:

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.

Решение:

```
#!/bin/bash
if [ $# -ne 2 ];
then
  echo "Invalid number of arguments!"
  exit 1
fi

dir="${1}"
num="${2}"

if [ ! -d "${dir}" ];
then
  echo "First argument is not a directory!"
  exit 2
elif [ ! -r "${dir}" ];
then
  echo "Directory is not readable!"
  exit 3
fi
```

```
function validate_num
{
   grep -qE '^[+-]?[0-9]+$' <(echo "${1}")
}
if validate_num "${num}" ;
then
   while read -r -d% SIZE FILE_PATH; do
      if validate_num "${SIZE}"; then
         if [ "${SIZE}" -gt "${num}" ];
         then
            echo "${FILE_PATH}"
         fi
      fi
   done < <(find "${dir}" -type f 2>/dev/null -printf "%s %p%%")
else
   echo "Second argument is not an integer!"
   exit 4
fi
```

05-b-7200.txt:

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.

Решение:

```
#!/bin/bash

for i; do
   if [ -f "${i}" ];
   then
      if [ -r "${i}" ];
      then
         echo "$(basename "${i}") is readable file!"
      else
         echo "$(basename "${i}") is not readable file!"
      fi
   elif [ -d "${i}" ];
   then
      cnt_files=$(find "${i}" -type f 2>/dev/null | wc -l | awk '{$1=$1}1')
```

```
        echo "$(find "${i}" -type f -size -"${cnt_files}"c 2>/dev/null | xargs basename -a)"
    else
        echo "$(basename "${i}") is not a file/directory name!"
    fi
done
```

05-b-7500.txt:

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.

Решение:

```
#!/bin/bash

num=$(( (RANDOM%200)-100 ))

cntr = 0

read -p "Guess? " guess

function validate_guess
{
    grep -qE '^[+-]?[0-9]+$' <(echo "${1}")
}

while true;
do
    cntr=$((cntr+1))
    if validate_guess "${guess}";
    then
        if [ "${guess}" -gt "${num}" ];
        then
            echo "...smaller!"
        elif [ "${guess}" -lt "${num}" ];
```

```
      then
         echo "...bigger!"
      else
         echo "RIGHT! Guessed "${num}" in "${cntr}" tries!"
         exit 0
      fi
   else
      echo "Not a valid guess!"
   fi

   read -p 'Guess? ' guess
done
```

05-b-7550.txt:

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

Решение:

```
#!/bin/bash

if [ $# -ne 1 ];

then

   echo "Invalid number of arguments!"

   exit 1

fi

user="${1}"

if [ $(id -u "${user}") -eq 1 ];

then

   echo "Invalid username!"

   exit 2

fi

cntr=$(ps -u "${user}" | wc -l)

killall -15 -u "${user}"

killall -9 -u "${user}"

echo "Total: ${cntr} killed processes!"
```

Решение2:

```
#!/bin/bash
```

```
if [ $# -ne 1 ]; then
	echo "Invalid number of arguments!"
	exit 1
fi

user="${1}"

if [ $(id -u "${user}") -eq 1 ]; then
	echo "Invalid username!"
	exit 2
fi

CNT=0

while read PID; do
	kill -15 "${PID}"
	sleep 1
	kill -9 "${PID}"
	CNT=$(($CNT+1))
done < <(ps -u "${user}" -o pid=)

echo "Total: ${CNT} killed processes."
```

05-b-7700.txt:

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.

Решение:

```
#!/bin/bash
if [ $# -ne 2 ];
then
   echo "Invalid number of arguments!"
```

```
    exit 1
fi
dir="${1}"
num="${2}"
if [ ! -d "${dir}" ];
then
    echo "Invalid directory name!"
    exit 2
elif [ ! -r "${dir}" ];
then
    echo "Directory is not readable!"
    exit 3
fi
function validate_num
{
    grep -qE '^[+-]?[0-9]+$' <(echo ${num})
}
if validate_num "${num}";
then
    find "${dir}" -maxdepth 1 -type f 2>/dev/null -printf "%s\n" \
        | awk -v sz="${num}" '$1 > sz {SUM += $1} END {print SUM}'
else
    echo "Second argument is not a valid integer!"
    exit 4
fi
```

Решение2:

```
#!/bin/bash

if [ $# -ne 2 ]; then
        echo "Invalid number of arguments!"
        exit 1
```

```
fi

dir="${1}"
num="${2}"

if [ ! -d "${dir}" ]; then
	echo "Invalid directory!"
	exit 2
elif [ ! -r "${dir}" ]; then
	echo "Directory is not readable!"
	exit 3
fi

function validate_num {
	egrep -q '^[+-]?[0-9]+$' <(echo "${num}")
}

if validate_num "${num}"; then
	all_sizes=$(find "${dir}" -maxdepth 1 -type f 2>/dev/null -size +"${num}"c -printf "%s\n")
	total_size=0
	for i in ${all_sizes}; do
		(( total_size+="${i}" ))
	done
	echo "${total_size}"
else
	echo "Second argument is an invalid integer number!"
	exit 4
fi
```

05-b-7800.txt:

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 конструкцията while read -d
Решение:

```
#!/bin/bash

if [ $# -ne 1 ];
then
   echo "Invalid number of arguments!"
   exit 1
fi

dir="${1}"

if [ ! -d "${dir}" ];
then
   echo "Invalid directory name!"
   exit 2
elif [ ! -r "${dir}" ];
then
   echo "Directory is not readable!"
   exit 3
fi

cnt=0

while read -d $'\n' line;
do
   if [ -x "${line}" ];
   then
      (( cnt+=1 ))
   fi
done < <(find "${dir}" -type f 2>/dev/null)

echo "Number of executable files is ${cnt}"
```

05-b-8000.txt:

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 нампример 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.

Решение:

```
#!/bin/bash

if [ $# -ne 1 ];
then
   echo "Invalid number of arguments!"
   exit 1
fi

user="${1}"

if ! id "${user}" 2>/dev/null;
then
   echo "Invalid username!"
   exit 2
fi

ps -u "${user}" -o pid,rss,vsz | tail -n +2 | while read PID RSS VSZ;
do
   if [ $VSZ -eq 0 ];
   then
      proportion="inf"
   else
      proportion=$(echo "scale=4; $RSS/$VSZ" | bc)
   fi

   echo "${VSZ} ${PID} consume ${proportion} of RSS/VSZ memory"
done | sort -rn -t ' ' -k1 | cut -d ' ' -f2-
```

05-b-9100.txt:

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

Решение:

```
#!/bin/bash

if [ $# -ne 2 ];
then
   echo "Invalid number of arguments!"
   exit 1
fi

s="${1}"
d="${2}"
```

```
if [ ! -d "${s}" ];
then
   echo "Invalid source directory!"
   exit 2
elif [ ! -r "${s}" ];
then
   echo "Source directory is not readable!"
   exit 3
fi

if [ ! -d "${d}" ];
then
   echo "Invalid destination directory!"
   exit 4
elif [ ! -r "${d}" ];
then
   echo "Destination directory is not readable!"
   exit 5
fi

while read ext;
do
   mkdir -p "${d}/${ext}"
   find "${s}" -type f -name "*.${ext}" 2>/dev/null -print0 | xargs -0 -I {} cp {} "${d}/${ext}"
done < <(find "${s}" -type f -name "*.*" -printf "%f\n" | rev | cut -d '.' -f1 | rev | sort | uniq )
```

05-b-9200.txt:

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/

Решение:

```
#!/bin/bash

if [ $# -eq 0 ]; then
	echo "Invalid number of arguments!"
	exit 1
fi

FIRST_ARG="${1}"
FLAG_RECUR=0

if [ "${FIRST_ARG}" = -r ]; then
	FLAG_RECUR=1
	shift 1
fi

export LOGFILE=$(mktemp)

for i; do
	if [ -d "${i}" ]; then
		DIR_CONTENT=$(find "${i}" -mindepth 1 | wc -l | awk '{$1=$1}1')
		if [ "${DIR_CONTENT}" -eq 0 ]; then
			echo "[$(date +"Y-%m-%d %H:%M:%S")] Removed directory ${i}/" >> "${LOGFILE}"
			rm -di "${i}"
		elif [ "${FLAG_RECUR}" -eq 1 ]; then
			echo "[$(date +"Y-%m-%d %H:%M:%S")] Removed directory recursively ${i}/" >>
"${LOGFILE}"
			rm -Rdi "${i}"
		fi
	elif [ -f "${i} ]; then
```

```
        echo "[$(date +"Y-%m-%d %H:%M:%S")] Removed file ${i}" >> "${LOGFILE}"
        rm -di ${i}
    fi
done

echo "Log file info: "
cat ${LOGFILE}
```

05-b-9500.txt:

(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 аргумен 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".

Решение:

```
#!/bin/bash
if [ $# -ne 2 ]; then
    echo "Invalid number of arguments!"
    exit 1
fi
STR="${2}"
if [ $1 = "-r" ]; then
    echo -e "\033[0;31m${STR}"
elif [ $1 = "-g" ]; then
    echo -e "\033[0;32m${STR}"
elif [ $1 = "-b" ]; then
    echo -e "\033[0;34m${STR}"
else
    echo "First argument is invalid!"
    exit 2
fi

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

05-b-9501.txt:

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 (т.е., все едно сте извикали командата cat).

Hint: Не забравяйте да връщате цветовете в терминала.
Решение:

```
#!/bin/bash

if [ $# -ne 1 ]; then
	echo "Invalid number of arguments!"
	exit 1
fi

SC="${1}"

if ! egrep -q '^-[rgbx]$' <(echo "${SC}"); then
	echo "Invalid color argument!"
	exit 2
fi

case "${SC}" in
-r) COUNT=0
;;
-g) COUNT=1
;;
-b) COUNT=2
;;
-x) COUNT=3
;;
esac

if [ ${COUNT} -eq 3 ]; then
	while read line; do
		echo "${line}"
	done

	exit 0

fi

function change_color {
	N=${1}
	STR="${2}"
	REM=$((${N} % 3))

	case "${REM}" in
	0)
```

```
            echo -e "\033[0;31m${STR}"
        ;;
        1)
            echo -e "\033[0;32m${STR}"
        ;;
        2)
            echo -e "\033[0;34m${STR}"
        ;;
        esac
}

while read line; do
        change_color ${COUNT} "${line}"
        COUNT=$(( COUNT + 1 ))
done

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

05-b-9600.txt:

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-HH-MM-SS.{gz,tgz}", където filename е оригиналното име на файла (директорията) преди да бъде изтрит

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 detele
$ 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 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

Решение:

```
#!/bin/bash

[ $# -eq 0 ] && exit 1

FIRST_ARG="${1}"
FLAG=0

if [ "${FIRST_ARG}" = -r ];then
        FLAG=1
        shift 1
fi

function timestamp {
        date +"%Y-%m-%d-%H-%M-%S"
}

function archive_and_save {
        SRC=$1
        POSTFIX=$2
        TIMESTAMP=$(timestamp)
        tar -czf "${BACKUP_DIR}/${TIMESTAMP}_${POSTFIX}.tgz" "${SRC}"
}

function compress_and_save {
        SRC=$1
        TIMESTAMP=$(timestamp)
        gzip -c "${SRC}" > "${BACKUP_DIR}/${SRC}_${TIMESTAMP}.gz"
}

for i; do
        if [ -d "${i}" ]; then
```

```
                DIR_CONTENT=$(find "${i}" -mindepth 1 -maxdepth 1
2>/dev/null | wc -l)
                if [ "${DIR_CONTENT}" -eq 0 ]; then
                        archive_and_save "${i}" "empty_dir"
                        rm -i -d "${i}"
                elif [ "${FLAG}" -eq 1 ];then
                        archive_and_save "${i}" "full_dir"
                        rm -i -d -R "${i}"
                fi
        elif [ -f "${i}" ]; then
                compress_and_save "${i}"
                rm -i -- "${i}"
        fi
done
```

05-b-9601.txt:

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.

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 втората 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/