1. ***Написать скрипт определяющий запущен ли скрипт пользователем root.***

#!/bin/bash

if ! [ "$(id -u)" = 0 ];

then

echo "I am not root!"

exit 1

fi

1. ***Написать скрипт который после запуска ожидает ввод строи от пользователя, а затем выводит ее в стандартный вывод. Используйте команду read.***

#!/bin/bash

exec 6<&0

exec 0< myfile

count=1

while read -r line

do

echo "Line #$count: $line"

count=$(( count + 1 ))

done

exec 0<&6

read -r -p"Are you done now? " answer

case $answer in

y) echo "Goodbye";;

n) echo "Sorry, this is the end.";;

esac

1. ***Написать скрипт который после запуска ожидает ввод строки от пользователя, а затем выводит ее в стандартный вывод, но ограничить время ожидания пользовательского ввода 5 секундами. В случае если пользователь не успел ввести строку, остановить программу и напечатать в стандартный вывод сообщение об этом.***

#!/bin/bash

if read -t 5 -p "Enter your name: " name

then echo "Hello $name, welcome to my script"

else echo "Sorry, too slow! "

fi

1. ***Напишите скрипт, возвращающий***
   1. ***количество дней, прошедших с начала года***

#!/bin/bash

echo "Количество дней, прошедших с начала года: $(date +%j.)"

* 1. ***количество секунд, прошедших с начала "эпохи UNIX"***

#!/bin/bash

echo "Количество секунд, прошедших с 01/01/1970 : $(date +%s.)"

1. ***Написать скрипт, который выводит в стандартный вывод все аргументы который были ему переданы. Используйте команду shift.***

#!/bin/bash

count=1

while [ -n "$1" ]

do

echo "Параметр №$count = $1"

count=$(( count + 1 ))

shift

done

1. ***Напишите скрипт, который ожидает пользовательского ввода одного символа с клавиатуры, проверяет что символ был введен один. Далее скрипт определяет, что за символ был введен - буква в нижнем регистре, буква в верхнем регистре, цифра, знак пунктуации, пробел или что-то другое. Используйте команду case.***

#!/bin/bash

echo; echo "Нажмите клавишу"

read -r Keypress

case "$Keypress" in

[a-z] ) echo "буква в нижнем регистре";;

[A-Z] ) echo "Буква в верхнем регистре";;

[0-9] ) echo "Цифра";;

\* ) echo "Знак пунктуации, пробел или что-то другое";;

esac

1. ***Написать скрипт преобразующий метры в мили. В качестве входящего аргумента должна быть цифра - метры. В стандартный вывод вывести количество миль.***

#!/bin/bash

read -r "Enter metres: " var1

var2=$(( var1 / 1609 ))

echo $var2

1. ***Напишите скрипт который будет проводить симуляцию 700 бросков 6 гранного кубика.***

#!/bin/bash

RANDOM=$$

PIPS=6

MAXTHROWS=700

throw=0

ones=0

twos=0

threes=0

fours=0

fives=0

sixes=0

print\_result ()

{

echo

echo "единиц = $ones"

echo "двоек = $twos"

echo "троек = $threes"

echo "четверок = $fours"

echo "пятерок = $fives"

echo "шестерок = $sixes"

echo

}

update\_count()

{

case "$1" in

0) ((ones ++));;

1) ((twos ++));;

2) ((threes ++));;

3) ((fours ++));;

4) ((fives ++));;

5) ((sixes ++));;

esac

}

echo

while [ "$throw" -lt "$MAXTHROWS" ]

do

((die1 = RANDOM % PIPS))

update\_count "$die1"

((throw ++))

done

print\_result

exit 0

1. ***Написать скрипт, который рандомно сгенерирует на диске структуру файлов и папок. Входным параметром будут: директория, в которой нужно генерировать данные, глубину вложенности поддиректорий, максимальный размер файлов, которые будут создаваться в этих директориях и максимальное количество итераций по созданию объектов в директории (т.е. общее количество файлов и директорий, которые будут созданы). Максимальная длина имени - 8 символов, содержимое файлов - рандом или нули. Проверку на доступное место делать не нужно.***

#!/bin/bash

mkdir -p test/dir--{000..100}

touch test/dir--{000..100}/file-{A..Z}

fallocate -l 16K 16K.txt

1. ***Написать скрипт, который принимает 2 входных параметра: source\_dir и destination\_dir - две директории на диске, сравнивает, что они ни одна из них не является родительской для другой, у них не совпадают имена и т.п. Далее скрипт должен подсчитать место, занимаемое source\_dir, и место, доступное на диске, где находится директория destination\_dir, если есть доступное место - скопировать (с наследованием всех атрибутов: времени создания, доступа и модификации, владельца и т.п.), если места нет - вывести предупреждение на экран и предложить пользователю продолжить (C или Y) или прервать (N или A).***
2. ***Модернизировать скрипт, чтобы он упаковывал данные из source\_dir и складывал в destination\_dir. При этом в самом начале работы скрипта нужно предусмотреть диалог с пользователем: предложить дописывать текущую дату и время в формате YYMMDD\_HHSS.gz к имени файла или выполнять ротацию архивов, т.е. дописывал порядковый номер архива (0.gz, 1.gz, 2.gz). В случае выбора второго варианта - так же нужно запросить максимальное количество копий, при этом номер 0 присваивается самому последнему созданному архиву, а номер N-1 - самому старому, на каждой итерации архив с номером N - удаляется.***
3. ***Модернизировать скрипт, чтобы все сообщения об ошибках подавлялись, а стандартные - выводились на консоль + записывались в лог-файл в текущей директории с именем out\_YYMMDD\_HHSS.log. Т.к. сообщения об ошибках подавляются, то в случае возникновения ошибок в процессе работы скрипта (проблемы с пермишенами, нехватка места, отсутствие какой-либо утилиты и т.п.) необходимо в конце работы скрипта вывести сообщение в консоль красным цветом "Warning: X error(s) occurred!", где X - количество перехваченных ошибок или количество вызовов команд, которые повлекли ошибки (как удобнее, на выбор).***
4. ***Написать скрипт, который бы выводил на экран наиболее выгодный курс валют конвертации USD->BYR и EUR->BYR. Способ работы скрипта: либо передавать входным параметром "USD" или "EUR" и выводить курс, либо, если входных параметров нет - а консольном диалоге запросить на выбор желаемый курс. Получать можно с любого сайта (в идеале - с finance.tut.by).***
5. ***Написать скрипт, который мог бы пройти авторизацию на Яндексе, сохранить сессию в cookie-файл, и показать кол-во писем в папке Inbox.***

#!/bin/bash

BACKUP\_DIR='/home/www/backup'

PROJECT='neblog.info'

MAX\_BACKUPS='20'

DATE=$(date '+%Y-%m-%d')

DIRS='/home/www/projects/neblog'

TOKEN=''

LOGFILE='backup.log'

sendLog='your-mail@yandex.ru'

sendLogErrorsOnly='true'

function mailing()

{

if [ ! $sendLog = '' ];then

if [ "$sendLogErrorsOnly" == true ];

then

if echo "$1" | grep -q 'error'

then

echo "$2" | mail -s "$1" $sendLog > /dev/null

fi

else

echo "$2" | mail -s "$1" $sendLog > /dev/null

fi

fi

}

function logger()

{

echo "$(("$(date "+%Y-%m-%d %H:%M:%S")")) File $BACKUP\_DIR: $1" >> $BACKUP\_DIR/$LOGFILE

}

function parseJson()

{

local output

regex="(\"$1\":[\"]?)([^\",\}]+)([\"]?)"

[[ $2 =~ $regex ]] && output=${BASH\_REMATCH[2]}

echo "$output"

}

function checkError()

{

echo parseJson 'error' "$1"

}

function getUploadUrl()

{

json\_out = $(curl -s -H Authorization: "OAuth $TOKEN https://cloud-api.yandex.net:443/v1/disk/resources/upload/?path=app:/ "$backupName" &overwrite = true)"

json\_error = $(checkError "$json\_out")

if [[ $json\_error != '' ]];

then

logger "$PROJECT - Yandex.Disk error: $json\_error"

mailing "$PROJECT - Yandex.Disk backup error" "ERROR copy file $FILENAME. Yandex.Disk error: $json\_error"

echo ''

else

output=$(parseJson 'href' "$json\_out")

echo "$output"

fi

}

function uploadFile

{

local json\_out

local uploadUrl

local json\_error

uploadUrl=$(getUploadUrl)

if [[ $uploadUrl != '' ]];

then

echo "$uploadUrl"

json\_out = "$(curl -s -T $1 -H "Authorization: OAuth $TOKEN" $uploadUrl)"

json\_error=$(checkError "$json\_out")

if [[ $json\_error != '' ]];

then

logger "$PROJECT - Yandex.Disk error: $json\_error"

mailing "$PROJECT - Yandex.Disk backup error" "ERROR copy file $FILENAME. Yandex.Disk error: $json\_error"

else

logger "$PROJECT - Copying file to Yandex.Disk success"

mailing "$PROJECT - Yandex.Disk backup success" "SUCCESS copy file $FILENAME"

fi

else

echo 'Some errors occured. Check log file for detail'

fi

}

function backups\_list() {

curl -s -H "Authorization: OAuth $TOKEN" "https://cloud-api.yandex.net:443/v1/disk/resources?path=app:/&sort=created&limit=100" | tr "{},[]" "\n" | grep "name[[:graph:]]\*.tar.gz" | cut -d: -f 2 | tr -d '"'

}

function backups\_count() {

local bkps = "$(backups\_list | wc -l)"

expr $bkps / 2

}

function remove\_old\_backups() {

bkps=$(backups\_count)

old\_bkps=$((bkps - MAX\_BACKUPS))

if [ "$old\_bkps" -gt "0" ];then

logger "Удаляем старые бекапы с Яндекс.Диска"

for i in $(eval "echo {1..((old\_bkps \* 2))})"; do

curl -X DELETE -s -H "Authorization: OAuth $TOKEN" "https://cloud-api.yandex.net:443/v1/disk/resources?path=app:/$(backups\_list | awk '(NR == 1)')&permanently=true"

done

fi

}

logger "--- $PROJECT START BACKUP $DATE ---"

logger "Создаем архив каталогов $BACKUP\_DIR/$DATE-files-$PROJECT.tar.gz"

tar -czf "$BACKUP\_DIR/$DATE-files-$PROJECT.tar.gz $DIRS"

FILENAME=$DATE-files-$PROJECT.tar.gz

logger "Выгружаем на Яндекс.Диск архив с файлами $BACKUP\_DIR/$DATE-files-$PROJECT.tar.gz"

backupName=$DATE-files-$PROJECT.tar.gz

uploadFile "$BACKUP\_DIR/$DATE-files-$PROJECT.tar.gz"

logger "Удаляем архивы с диска"

find $BACKUP\_DIR -type f -name "\*.gz" -exec rm '{}' \;

if [ $MAX\_BACKUPS -gt 0 ];then remove\_old\_backups; fi

logger "Завершение скрипта бекапа"