Function Samples

?????? *.gz ??

??: KEEP=100 CleanUp /path/to/dir

```
CleanUp() {
    dest="$1"
    if [ $dest ];then
        CheckDIR $dest
        cd $dest
        echo "-> Cleaning up the directory $dest [$(date +'%Y-%m-%d %T')] ..."
        total=$(ls -lt *.gz 2>/dev/null | grep -v "^d" | grep -v "^total" | wc -l)
        echo " [!] The numbers of the existing files: $total"
        tail_num=$(( $total - $KEEP ))
        if [ $tail_num -gt 0 ];then
            for f in $(ls -lt *.gz | grep -v "^d" | grep -v "^total" | tail -$tail_num | awk -F ' ' '{print $9}')
            do
                #echo " [!] Deleted file: $f"
                rm -vf $f
            done
        else
            echo " [!] No need of deletion"
        fi
    fi
}
```

????

??: CheckDIR /path/to/dir

```
CheckDIR() {
    echo -n "-> Checking the directory <$1> ...... "
    if [ ! -d  $1 ]
    then
        echo "[Failed]"
```

```
    exit 1
  else
    echo "[OK]"
  fi
}
```

ip ??

```
# Test an IP address for validity:
# Usage:
#      valid_ip IP_ADDRESS
#      if [[ $? -eq 0 ]]; then echo good; else echo bad; fi
#   OR
#      if valid_ip IP_ADDRESS; then echo good; else echo bad; fi
#
function is_ip()
{
   local  ip=$1
   local  stat=1

   if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
      OIFS=$IFS
      IFS='.'
      ip=($ip)
      IFS=$OIFS
      [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 \
         && ${ip[2]} -le 255 && ${ip[3]} -le 255 ]]
      stat=$?
   fi
   return $stat
}
```

???????

```
Is_number() {
   # Usage: Is_number ${your-number} 0
   # Type: Number with decimal
   # If the number is invalid, return 0.
   # If not specify, return NULL
   local input err_num re
```

```
    input=$1
    err_num=$2
    re="^[0-9]+([.][0-9]+)?$"
    if [[ $input =~ $re ]]; then
        echo $input
    else
        echo $err_num
    fi
}


log_utilization_percent=$(Is_number $raw 0)


# Validate for integer
[[ $port =~ ^-?[0-9]+$ ]] && dosomething
# No minus character
[[ $port =~ ^[0-9]+$ ]] && dosomething
```

???(??????)

```
CL () {
WORDS=$@; termwidth="$(tput cols)"; padding="$(printf '%0.1s' ={1..500})"; printf '%*.*s %s %*.*s\n' 0
"$(((termwidth-2-${#WORDS})/2))" "$padding" "$WORDS" 0 "$(((termwidth-1-${#WORDS})/2))" "$padding";
}


CL "This is Seperator Line"
```

?????

```
ToUpper() {
    echo $1 | tr "[:lower:]" "[:upper:]"
}


ToLower() {
    echo $1 | tr "[:upper:]" "[:lower:]"
}
```

URL ????

```
function urlencode() {
    local data
```

```
    if [[ $# != 1 ]]; then
        echo "Error: No string to urlencode"
        return 1
    fi
    data="$(curl -s -o /dev/null -w %{url_effective} --get --data-urlencode "$1" "")"
    if [[ $? != 3 ]]; then
        echo "Unexpected error" 1>&2
    else
        echo "${data##/?}"
    fi
    return 0
}


urlencode "$1"


function urldecode() {
    # urldecode <string>

    local url_encoded="${1//+/ }"
    printf '%b' "${url_encoded//%/\\x}"
}
```

??????

- https://linuxhint.com/return-string-bash-functions/
- https://www.linuxjournal.com/content/return-values-bash-functions

Example-1: Using Global Variable

```
function F1()
{
    retval='I like programming'
}

retval='I hate programming'
echo $retval
F1
echo $retval
```

Example-2: Using Function Command

```
function F2()
{
    local  retval='Using BASH Function'
    echo "$retval"
}


getval=$(F2)
echo $getval
```

Example-3: Using Variable

```
function F3()
{
    local arg1=$1

    if [[ $arg1 != "" ]];
    then
        retval="BASH function with variable"
    else
        echo "No Argument"
    fi
}


getval1="Bash Function"
F3 $getval1
echo $retval
getval2=$(F3)
echo $getval2
```

is_root

???????? root ????

```
is_root()
{
    current_user="`id`"
    case "$current_user" in
        uid=0\(root\)*)
        ;;
        *)
```

```
			echo "$0: You must have root privileges in order to install ${product_name}"
			echo "su as root and try again"
			echo
			exit 1
		;;
	esac
}
```

With if-then

```
if [ ${EUID} -ne 0 ]
then
	exit 1 # this is meant to be run as root
fi


Run_MyCodes
```

check_os

????? OS ??
???

```
check_os()
{
	if [ -f /etc/redhat-release ];then
		rhel=`grep "^Red Hat Enterprise Linux" /etc/redhat-release`
		centos=`grep "^CentOS" /etc/redhat-release`

		release=`cat /etc/redhat-release | cut -d'.' -f1 | awk '{print $NF}'`

		if [ ! -z "${rhel}" ];then
			sys="rhel"
		fi

		if [ ! -z "${centos}" ];then
			sys="rhel"
		fi
	fi

	if [ -z "${sys}" -o -z "${release}" ];then
```

```
    echo "Can not determine Linux distribution."
    exit 127
  fi
  
  if [ -f /proc/vz/veinfo ];then
  # we are under Virtuozzo
    virtuozzo=1
  else
    virtuozzo=0
  fi

  if [ "${VNDEBUG}" ];then
    debugrpm='--rpmverbosity=debug'
  else
    debugrpm=''
  fi
}
```

???

```
# Detect the platform for different environment.
OS="$(uname)"
case $OS in
   "Linux")
     MAIL_CMD="/bin/mail"
     ;;
   "AIX")
     MAIL_CMD="/usr/bin/mail"
     ;;
    *)
     MAIL_CMD="/bin/mail"
     ;;
esac
```

???

```
get_linux_distribution ()
{
   if [ -f /etc/debian_version ]; then
       DIST="DEBIAN"
```

```
    HTTP_DIR="/etc/apache2/"
    HTTP_CONFIG=${HTTP_DIR}"apache2.conf"
    MYSQL_CONFIG="/etc/mysql/mariadb.conf.d/50-server.cnf"
  elif [ -f /etc/redhat-release ]; then
    DIST="CENTOS"
    HTTP_DIR="/etc/httpd/"
    HTTP_CONFIG=${HTTP_DIR}"conf/httpd.conf"
    MYSQL_CONFIG="/etc/my.cnf"
  else
    DIST="OTHER"
    echo 'Installation does not support your distribution'
    exit 1
  fi
}
```

IsRunning

?????????????????????????????????? script ??????

```
IsRunning() {
  if [ ! -e $script_pid ];then
    RUNCOUNT=0
  else
    RUNPID=$(cat $script_pid)
    RUNCOUNT=$(ps -fp $RUNPID | grep -c `basename $0`)
  fi
  #echo $RUNCOUNT

  if [ $RUNCOUNT -eq 0 ];then
    echo $$ > $script_pid
    return 1  # return False
  else
    return 0  # return True
  fi
}


script_pid="/tmp/$(basename $0).$(whoami).pid"
# Check if the program is already running.
if IsRunning
then
```

```
    echo "Abort: The program is already running!"
    exit 1
fi
```

getopts ????

```
function usage ()
{
⌷cat <<- EOT

  Usage :  ${0##/*/} [options] -t application -q 1024,1055,1077 -m dpggen -u kroybal -r ods [--]

  Options:
  -d|debug      Bash Debugging Info
  -m|machine⌷Database host machine
  -t|table⌷Starting table
  -u|user⌷User to connect to DB
  -q|quals⌷Qualifying records to delete
  -r|relation⌷Database name
  -s|schema     Logical Partition
  -h|help       Display this message
  -v|version    Display script version

⌷EOT
}   # ----------  end of function usage  ----------


#-------------------------------------------------------------------
#  Handle command line arguments
#-------------------------------------------------------------------

[[ $# -eq 0 ]] && {
⌷# no arguments
⌷usage
⌷exit 1
}

while getopts ":dhm:q:r:s:t:u:v" opt
do
  case $opt in
```

```bash
    d|debug   )  set -x;;
    h|help    )  usage; exit 0   ;;
    t|table   )  table=$OPTARG;;
    q|quals   )  quals=$OPTARG;;
    m|machine )  host=$OPTARG;;
    u|user    )  user=$OPTARG;;
    r|relation )  db=$OPTARG;;
    s|schema  )  schema=$OPTARG;;
    v|version )  echo "$0 -- Version $ScriptVersion"; exit 0   ;;


    \? )  echo -e "\n  Option does not exist : $OPTARG\n"
          usage; exit 1   ;;

  esac    # --- end of case ---
done
shift $(($OPTIND-1))
```

```bash
Usage() {
    echo "Usage: $0 [-t <TEXT-Message>] [-n <phone-number>]";
    echo "For example: $0 -t \"Hello, SuperMan\" -n \"085713 457199\" "
}

while getopts "t:n:" o; do
    case "$o" in
       t)
          t=$OPTARG
          ;;
       n)
          n=$OPTARG
          ;;
       \?)
          echo "Invalid option: -$OPTARG"
          Usage
          exit 1
          ;;
        :)
          echo "Option -$OPTARG requires an argument !"
          Usage
          exit 1
          ;;
```

```
      esac
  done


  if [ $OPTIND -ne 5 ]; then
      echo "Invalid options entered !"
      Usage
      exit 1
  fi
```

??????

```
[Option] [Typical meaning]
-a All, append
-b Buffer,block size, batch
-c Command, check
-d Debug, delete, directory
-D Define
-e Execute, edit
-f File, force
-h Headers, help
-i Initialize
-I Include
-k Keep, kill
-l List, long, load
-m Message
-n Number, not
-o Output
-p Port, protocol
-q Quiet
-r Recurse, reverse
-s Silent, subject
-t Tag
-u User
-v Verbose
-V Version
-w Width, warning
-x Enable debugging, extract
-y Yes
-z Enable compression
```

Without getopts

```
while [[ "${1}" != "" ]]; do
  case "${1}" in
    --vertical)    verticle="true" ;;
      --add-txt)    add_txt="true" ;;
      --replace)     replace="true" ;;
      --verify)     verify="true" ;;
    esac


    shift 1
done
```

Pause ??

??: pause

> " ??????????

```
pause()
{

  local force_pause=$1

  if [ -z $force_pause ]; then
    if test $NONINTERACTIVE; then
      return 0
    fi
  else
    shift
  fi

  [ $# -ne 0 ] && echo -e $* >&2
  echo -e "Press [Enter] to continue...\c " >&2
  read tmp
  return 0
}
```

Log ??

```
log() {  # classic logger
    local prefix="[$(date +%Y/%m/%d\ %H:%M:%S)]: "
    echo "${prefix} $@" >&2
}


log "INFO" "a message"
```

Error ????

?????????

```
error()
{
 echo -e "Error: $*!" >&2
 return 0
}
```

```
die()
{
 echo
 echo "ERROR: $*"
 echo
 echo "Check the error reason, fix and try again."
 echo "You are welcome to open a support ticket at https://help.4psa.com!"
 echo
 rm -f ${repolocal} local$$
 rm -f /var/lock/subsys/vninstaller
 exit 1
}

die() {
    printf "ERR: %s\n" "$1" >&2
    exit 1
}
[ -e "$SRC" ] || die "Wal colors not found, exiting script. Have you executed Wal before?"


check_deps()
```

```
{
	which yum >/dev/null 2>&1
	if [ $? -gt 0 ];then
		echo "yum binary can not be found."
		die "It's not in the system PATH or in a standard location."
	fi

	which ifconfig >/dev/null 2>&1
		if [ $? -gt 0 ];then
		yum -y install net-tools >/dev/null 2>&1
		if [ $? -gt 0 ];then
			echo "ifconfig binary can not be found and package net-tools could not be installed."
			die "It's not in the system PATH or in a standard location."
		fi
	fi
}
```

?? Yes/No

??: getyn "Would you like to install WANPIPE now? [y]"

> $NONINTERACTIVE ??????????? user ??????????? 1
> NONINTERACTIVE=1 ??? = ?????????????

```
prompt()
{
	if test $NONINTERACTIVE; then
		return 0
	fi

	echo -ne "$*" >&2
	read CMD rest
	return 0
}

getyn()
{
	if test $NONINTERACTIVE; then
```

```
	return 0
	fi

	while prompt "$* (y/n) "
	do	case $CMD in
		[yY])	return 0
			;;
		[nN])	return 1
			;;
		*)	echo -e "\nPlease answer y or n" >&2
			;;
	esac
	done
}
```

???? (hash/encryption)

* Hashing, Encryption, Encoding ??????????: hashing_encryption_encoding.jpg

```
_Decrypt() {
    # Encrypt the string by following the command.
    # echo "your password" | openssl enc -base64
    # NOTE: The openssl requires to be installed.
    #
    # Decrypt the hash code by following the command.
    # _Decrypt "your-hash-key"
    #
    local hash
    hash="$1"
    if which openssl >/dev/null 2>&1;then
        if [ -n $hash ];then
            echo -n "$hash" | openssl enc -base64 -d
        fi
    fi
    return 0
}
```

```
# Encrypt
echo -n "123456" | openssl enc -aes-256-cbc -a -pass pass:ThisIsPassword
```

U2FsdGVkX188Af3yumHOyI19WPbltgRQnPDACmtRniQ=

```bash
# Decrypt
echo -n "U2FsdGVkX188Af3yumHOyI19WPbltgRQnPDACmtRniQ=" | openssl enc -aes-256-cbc -a -pass pass:ThisIsPassword -d

123456
```

```bash
# Hashing with openssl
echo -n "Hello, World" | openssl dgst -sha256
```

???????? (hash + salt)

```bash
#!/bin/bash
## Create:
salt=12345_

protocol=sha1sum

read -p "Enter login: " username
read -p "Password: " -s pass1
read -p "Repeat: " -s pass2

if [ "$pass1" != "$pass2" ]; then echo "Pass missmatch"; exit 1; else password=$pass1; fi

echo -en "$username " >> ./mypasswd
echo -e "${salt}${password}" | $protocol | awk '{print $1}' >> ./mypasswd
```

```bash
#!/bin/bash
## Read:
salt=12345_ #(samesalt)
protocol=sha1sum

read -p "Enter username: " username
read -p "Enter password: " -s password

if [ `grep $username ./mypasswd | awk '{print $2}'` != `echo "${salt}${password}" | $protocol | awk '{print $1}'` ]; then echo -e "wrong username or password"; exit 127; else echo -e "login successfull"; fi
```

```
# With openssl
echo 'mypassword' | openssl passwd -stdin
echo 'mypassword' | openssl passwd -salt "sAlT" -stdin


# Toe see mode details
openssl passwd -h
```

????

```
# set q here
q=""
helper(){
    local p="$*"
    for f in $p
    do
        echo "Working on $f"
        ffmpeg -nostdin -vaapi_device /dev/dri/renderD128 -i "$f" -vf format=nv12,hwupload -c:v hevc_vaapi -f mkv
-rc_mode 1 -qp "$q" "${f%.*}.HEVC.mkv"
    done
}


helper '/tmp/r/*.mp4'
helper '/tmp/r/*.avi'
helper '/path/to/*.mkv'
```

????

```
genpasswd()
{
    length=$1
    [ "$length" == "" ] && length=16
    tr -dc A-Za-z0-9_ < /dev/urandom | head -c ${length} | xargs
}
password=$(genpasswd)


if [ -e "/root/passwordMysql.log" ] && [ ! -z "/root/passwordMysql.log" ]
then
    password=$(awk '{print $1}' /root/passwordMysql.log)
fi
```

```
touch /root/passwordMysql.log
echo "$password" > /root/passwordMysql.log
```

??? `whiptail`

????

```
whiptail --title "Example Dialog" --msgbox "This is an example of a message box. You must hit OK to continue." 8
78
```

Yes/No

```
if (whiptail --title "Example Dialog" --yesno "This is an example of a yes/no box." 8 78); then
    echo "User selected Yes, exit status was $?."
else
    echo "User selected No, exit status was $?."
fi
```

????

```
COLOR=$(whiptail --inputbox "What is your favorite Color?" 8 39 Blue --title "Example Dialog" 3>&1 1>&2
2>&3)

exitstatus=$?
if [ $exitstatus = 0 ]; then
    echo "User selected Ok and entered " $COLOR
else
    echo "User selected Cancel."
fi

echo "(Exit status was $exitstatus)"
```

A file

```
echo "Welcome to Bash $BASH_VERSION" > test_textbox
#              filename height width
whiptail --textbox test_textbox 12 80
```

????

```
PASSWORD=$(whiptail --passwordbox "please enter your secret password" 8 78 --title "password dialog" 3>&1 1>&2 2>&3)

exitstatus=$?
if [ $exitstatus == 0 ]; then
    echo "User selected Ok and entered " $PASSWORD
else
    echo "User selected Cancel."
fi

echo "(Exit status was $exitstatus)"
```

??

```
whiptail --title "Menu example" --menu "Choose an option" 25 78 16 \
"<-- Back" "Return to the main menu." \
"Add User" "Add a user to the system." \
"Modify User" "Modify an existing user." \
"List Users" "List all users on the system." \
"Add Group" "Add a user group to the system." \
"Modify Group" "Modify a group and its list of members." \
"List Groups" "List all groups on the system."
```

????

```
whiptail --title "Check list example" --checklist \
"Choose user's permissions" 20 78 4 \
"NET_OUTBOUND" "Allow connections to other hosts" ON \
"NET_INBOUND" "Allow connections from other hosts" OFF \
"LOCAL_MOUNT" "Allow mounting of local devices" OFF \
"REMOTE_MOUNT" "Allow mounting of remote devices" OFF
```

????

```
whiptail --title "Radio list example" --radiolist \
"Choose user's permissions" 20 78 4 \
"NET_OUTBOUND" "Allow connections to other hosts" ON \
"NET_INBOUND" "Allow connections from other hosts" OFF \
"LOCAL_MOUNT" "Allow mounting of local devices" OFF \
"REMOTE_MOUNT" "Allow mounting of remote devices" OFF
```

???

```bash
#!/bin/bash
{
    for ((i = 0 ; i <= 100 ; i+=5)); do
        sleep 0.1
        echo $i
    done
} | whiptail --gauge "Please wait while we are sleeping..." 6 50 0
```

???????

```
function color
{
    # □□□□□□□□
    reset="\033[0m"
    bold="\033[1m"
    underline="\033[4m"
    inverse="\033[7m"

    # □□□□□□□
    redx="\e[1;31m"
    black="\033[30m"
    red="\033[1;31m"
    green="\033[32m"
    yellow="\033[33m"
    orange="\033[1;93m"
    blue="\033[1;34m"
    purple="\033[35m"
    cyan="\033[36m"
    white="\033[1;37m"

    # □□□□□□□
    bg_black="\033[40m"
    bg_red="\033[41m"
    bg_green="\033[42m"
    bg_yellow="\033[43m"
    bg_blue="\033[44m"
```

```
    bg_purple="\033[45m"
    bg_cyan="\033[46m"
    bg_white="\033[47m"


}


function os_name
{
    if [ -e /etc/os-release ]; then
        # Get the name of the current Linux distribution
        os_name=$(grep PRETTY_NAME /etc/os-release | cut -d= -f2 | tr -d '"')
        if [[ "$os_name" == *"Debian"* ]]; then
            OSNAME="Debian"
            OSTYPE="T_Debian"
        elif [[ "$os_name" == *"CentOS"* ]]; then
            OSNAME="CentOS"
            OSTYPE="T_RedHat"
        elif [[ "$os_name" == *"Ubuntu"* ]]; then
            OSNAME="Ubuntu"
            OSTYPE="T_Debian"
        elif [[ "$os_name" == *"Kali"* ]]; then
            OSNAME="Kali"
            OSTYPE="T_Debian"
        else
            OSNAME="Unknown distribution"
            OSTYPE="T_RedHat"
        fi
    fi
}


# NOTE: 这个区分 command 和 package 还真的不好弄
function i
{
    color
    os_name
    # os
    if [[ $OSTYPE == "T_Debian" ]]; then
        OS_APP="apt-get"
    else
        OS_APP="yum"
```

```
        fi
    local package=$1
    if ! command -v "$package" &> /dev/null; then
        printf "$package   ${redx} uninstalled ${reset} \n"
        read -p "□□□□ $package□ (Y/n): " choice
        choice=${choice:-y}
        case "$choice" in
            y|Y )
                echo "□□□□ $package..."
                sudo $OS_APP update
                sudo $OS_APP install -y "$package"
                ;;
            * )
                printf "exit\n"
                exit 1
                ;;
        esac
    else
        printf "$package   ${green} installed ${reset} \n"
    fi
}


i "rsync"
```

```
# 'which' □ 'command' □□□□
function chk_commands
{
    if ! which curl &> /dev/null; then
        printf "WARNING curl □□□□□□\n"
    elif ! command -v rsync &> /dev/null; then
        printf "WARNING rsync □□□□□□\n"
    fi
}

function chk_command
{
    local cmd
    cmd=$1
    if ! command -v $cmd &> /dev/null; then
        printf "WARNING $cmd □□□□□□\n"
```

```
    fi
}


chk_command "rsync"
```

Learning

- [How to write a function in bash](#)