Fonctions bash : demander choix parmi liste

- Objet : Fonctions bash pour demander un choix parmi une liste d'options
- Niveau requis :
  avisé
- Commentaires : *Création de scripts bash interactifs.*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.

Introduction

Collections de fonctions bash pour demander à l'utilisateur de faire un choix parmi une liste et rendre vos scripts plus interactif!

Les fonctions permettent à chaque fois de définir un choix par défault (si réponse vide ou incorrecte)

Vous trouverez:

1. Question binaire type Oui/Non avec exécution immédiate lors de la réponse de l'utilisateur.
2. Choix textuel avec un petit contrôle sur l'affichage
3. Choix dans liste numéroté

Finalement, dans la dernière partie (exemple), vous trouverez un script compilant toutes les fonctions à des fins de démonstration.

bonus: pour dé-commenter les scripts:

```
 cat script.sh | sed '/^\s*#/d' | sed 's/#.*$//' > script_sans
commentaires.sh
```

Question binaire

Utilisation

askYesNo

```
askYesNo "Voulez vous continuer ?" true
```

```
Voulez vous continuer ? [O/n] o
```

Script

askyesno.sh

```
#!/bin/bash
# Yes/No question with default
askYesNo () {
        QUESTION=$1
        DEFAULT=$2
        if [ "$DEFAULT" = true ]; then                  # Valeur par
défaut définie en paramètre
                OPTIONS="[O/n]"
                DEFAULT="o"
            else
                OPTIONS="[o/N]"
                DEFAULT="n"
        fi
        read -p "$QUESTION $OPTIONS " -n 1 -s -r INPUT    # Execute au
premier caractère (-n 1)
        INPUT=${INPUT:-${DEFAULT}}                      # Si $INPUT
vide => remplace par $DEFAULT
        echo ${INPUT}
        if [[ "$INPUT" =~ ^[yYoO]$ ]]; then             # True si
y,Y,O ou o
            ANSWER=true
        else                                            # Faux pour
le reste
            ANSWER=false
        fi
}
```

Choix parmi liste

Utilisation

fonction questionWithDefaultSimple

```
questionWithDefaultSimple "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6
arg7 arg8" arg6
```

```
arg1            arg2            arg3            arg4
arg5            arg6            arg7            arg8
Votre choix? [arg6] :
```

Script

question_with_default_simple.sh

```
#!/bin/bash

questionWithDefaultSimple () {

    # Positional argument should strictly come before named arguments
    # 1er argument : Question à poser
    # 2e argument : Liste des options
    # 3e argument (optionnel) choix par défaut (sinon le premier item)

    QUESTION=$1
    read -ra OPTIONS  <<< "$2"                  # Transforme en array

    if [ -n "$3" ]; then                        # Si non donnée en
entrèe => on prend le premier élément
        DEFAULT="$3"
    else
        DEFAULT="${OPTIONS[0]}"
    fi

    # Create pattern for printf
    COL_SIZE=15
# nombre de caractère pour chaque colonne
    FORMAT_STRING="%-${COL_SIZE}s %-${COL_SIZE}s %-${COL_SIZE}s %-
${COL_SIZE}s\n"  # 4 colonnes de $COL_SIZE caractère (méthode bourrin)

    # Show
    printf "$FORMAT_STRING" "${OPTIONS[@]}"    # Mise en forme des
options selon $FORMAT_STRING et affichage

    # Ask user
    printf "\x1b[1;32m$QUESTION\x1b[0m"         # Question en gras +
vert
    read -p " [$DEFAULT] : " -r INPUT           # Demande d'input sans
aller à la ligne
    INPUT=${INPUT:-${DEFAULT}}                  # Si l'input est vide,
on rempli avec la valeur par défaut

    # Collect result
    if [[ "${OPTIONS[@]}" =~ "$INPUT" ]]; then   # Si la réponse est
une réponse valide, c-à-d que la valeur est présente dans $OPTIONS
        ANSWER=$INPUT                             # On garde la valeur
dans ANSWER
    else
        ANSWER=$DEFAULT                           # Sinon on prend celle
par défaut
    fi
}
```

Choix parmi liste : complet

Utilisation

nom.sh

```
questionWithDefaultComplet "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8" -d 4 -c 3
```

```
 arg1    arg2    arg3
 arg4    [arg5]  arg6
 arg7    arg8
Votre choix? [arg5] :
```

Script

question_with_default_complete

```
#!/bin/bash

questionWithDefaultComplet () {

    # Positional argument should strictly come before named arguments
    # 1er argument : Question à poser
    # 2e argument : Liste des options
    # -c|--columns (optionnel) Nombre de colonnes pour l'affichage
    # -d|--default (optionnel) position dans la liste de l'option à mettre par défaut

    QUESTION=$1
    read -ra OPTIONS  <<< "$2"          # Transforme la liste d'options en array
    read -ra OPTIONS_SHOW <<< "$2"      # Même liste, mais destinée à être modifié pour l'affichage

    POS_DEFAULT=0                       # Initialisation : position du choix par défaut à défaut de -d|--default
    NB_COLUMNS=4                        # Initialisation ; nombre de colonnes à défaut de paramètre -c|--columns

    while [[ $# -gt 0 ]] ; do           # Tant que le nombre d'argument ($#) n'est pas épuisé
        key="$1"                        # on cherche les clés correspondant aux options (-c et -d)
        case $key in
```

```
            -d|--default)               # Une fois la clé  touvée
            POS_DEFAULT="$2"            # On sélectionne l'argument qui
lui succède
            shift                       # on "consomme"/retire un
argument de la liste $@
            ;;
            -c|--columns)               # Même chose pour l'autre clé
            NB_COLUMNS="$2"
            shift
            ;;
        esac
        shift # past value              # On consomme un argument dans
tout les cas
    done

    # Get new default option and format display
    OPTIONS_SHOW[$POS_DEFAULT]="[${OPTIONS_SHOW[$POS_DEFAULT]}]"   # On
entoure de [crochet] l'option par défault
    DEFAULT="${OPTIONS[$POS_DEFAULT]}"                             # On
sauve sa valeur dans une variable

    # Get columns character length
    max_length=0                                          # Calcul de la
taille minimum d'une colonne (toutes les options doivent tenir)
    for opt in ${OPTIONS[@]}; do                          # On itère sur
les options
        opt_length=$(echo $opt | wc -c)                   # On compte le
nombre de caractères
        if [ $opt_length -ge $max_length ]; then          # Si on trouve
une chaîne plus grande que la précédente "plus grande",
            max_length=$opt_length                        # on enregistre
celle nouvelle taille
        fi
    done
    max_length=$(($max_length+1))                         # 'tite marge
supplémentaire

    # Create pattern for printf
    FORMAT_STRING=""                                        # préparation
de la mise en forme de l'affichage
    for i in $(seq $NB_COLUMNS); do                         # On
concatène "le nombre de colonnes" *
        FORMAT_STRING="$FORMAT_STRING %-${max_length}s "  # la taille
min d'une colonne
    done
    FORMAT_STRING="$FORMAT_STRING \n"

    # Show
    printf "$FORMAT_STRING" "${OPTIONS_SHOW[@]}"    # On affiche

    # Ask user
```

```
    printf "\x1b[1;32m$QUESTION\x1b[0m"              # Un peu de couleur
    read -p " [$DEFAULT] : " -r INPUT                # On demande une
entrée à l'utilisateur
    INPUT=${INPUT:-${DEFAULT}}                       # Si il n'a rien
rentré on substitue la valeur avec celle par défaut.

    # Collect result
    if [[ "${OPTIONS[@]}" =~ "$INPUT" ]]; then       # Si la réponse est
valide (dans la liste)
        ANSWER=$INPUT                                # On garde la
valeur
    else
        ANSWER=$DEFAULT                              # Sinon, on prend
celle par défaut
    fi
}
```

Choix parmi liste numérotée : build-in select

la build-in select effectue une boucle infinie demandant de choisir parmi une liste, vous pouvez insérer une instruction **case** ou un nouveau **select** pour faire des sous menus.

Utilisation

```
numberWithSelect "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6"
```

```
Votre choix?
 1) arg1
 2) arg2
 3) arg3
 4) arg4
 5) arg5
 6) arg6

#? 5
Vous avec choisi arg5
```

Script

numberWithSelect

```
numberWithSelect () {

    QUESTION=$1
    read -ra OPTIONS  <<< "$2"

    echo $QUESTION
    select ANSWER in ${OPTIONS[@]}; do
        if [ -n "$ANSWER" ]; then
            echo "Vous avec choisi $ANSWER"
            break
        else
            echo "Il n'y a pas de tel index!"
        fi
    done

}
```

Choix parmi liste numérotée avec valeur par défaut

Utilisation

nom.sh

```
numberedMenu "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6" 2
```

```
 1) arg1
 2) arg2
 3) arg3
 4) arg4
 5) arg5
 6) arg6

Votre choix? [2] :
```

Script

numbered_menu.sh

```
#!/bin/bash

numberedMenu () {

    # $1 = Question à poser
    # $2 = liste des éléments séparés par un espace
```

```
    # S3 (optionnal) = index de l'élément par défaut

    QUESTION=$1
    read -ra OPTIONS  <<< "$2" # Obtention de la liste de options

    if [ -n "$3" ]; then       # Si donné par l'utilisateur (c-à-d
variable non vide)
        DEFAULT=$3
    else                       # Sinon, valeur par défaut
        DEFAULT=1
    fi

    # Show
    printf "\n"
    for i in $(seq ${#OPTIONS[@]}); do                          # on
compte le nombre d'options
        printf '    %3s %s\n' "$i)" "${OPTIONS[$(($i-1))]}"   # On
affiche la position dans la liste et la valeur de l'option
    done
    printf '\n'

    # ask user
    printf "\x1b[1;32m$QUESTION\x1b[0m"          # Un peu de couleur
(gras + vert)
    read -p " [$DEFAULT] : " -r INPUT            # Demande valeur sans
retour à la ligne
    INPUT=${INPUT:-${DEFAULT}}                   # Substitut par
$DEFAULT si réponse vide

    # Collect result
    if [[ "$(seq ${#OPTIONS[@]})" =~ "$INPUT" ]]; then
        ANSWER="${OPTIONS[$(($INPUT-1))]}"
    else
        ANSWER="${OPTIONS[$(($DEFAULT-1))]}"
    fi
}
```

Exemples

Ce fichier réunit toutes les fonctions définies plus haut et lance une série d'exemples

toutes_les_fonctions_et_exemple.sh

```
#!/bin/bash

# Yes/No question with default
askYesNo () {
```

```
        QUESTION=$1
        DEFAULT=$2
        if [ "$DEFAULT" = true ]; then
                OPTIONS="[O/n]"
                DEFAULT="o"
            else
                OPTIONS="[o/N]"
                DEFAULT="n"
        fi
        read -p "$QUESTION $OPTIONS " -n 1 -s  -r INPUT
        INPUT=${INPUT:-${DEFAULT}}
        echo ${INPUT}
        if [[ "$INPUT" =~ ^[yYoO]$ ]]; then
            ANSWER=true
        else
            ANSWER=false
        fi
}


questionWithDefault () {

    # Positional argument should strictly come before named arguments

    QUESTION=$1
    read -ra OPTIONS  <<< "$2" # Keep options list
    read -ra OPTIONS_SHOW <<< "$2" # Same list but highlight [default
value]

    POS_DEFAULT=0 # position of default argument
    NB_COLUMNS=4 # nummber of element by line to display

    while [[ $# -gt 0 ]] ; do
        key="$1"
        case $key in
            -d|--default)
            POS_DEFAULT="$2"
            shift # past argument
            ;;
            -c|--columns)
            NB_COLUMNS="$2"
            shift
            ;;
        esac
        shift # past value
    done

    # Get new default option and format display
    OPTIONS_SHOW[$POS_DEFAULT]="[${OPTIONS_SHOW[$POS_DEFAULT]}]"
    DEFAULT="${OPTIONS[$POS_DEFAULT]}"
```

```
    # Get columns character length
    max_length=0
    for opt in ${OPTIONS[@]}; do
        opt_length=$(echo $opt | wc -c)
        if [ $opt_length -ge $max_length ]; then
            max_length=$opt_length
        fi
    done
    max_length=$(($max_length+1))

    # Create pattern for fstring
    FORMAT_STRING=""
    for i in $(seq $NB_COLUMNS); do
        FORMAT_STRING="$FORMAT_STRING %-${max_length}s "
    done
    FORMAT_STRING="$FORMAT_STRING \n"

    # Show
    printf "$FORMAT_STRING" "${OPTIONS_SHOW[@]}"

    # Ask user
    printf "\x1b[1;32m$QUESTION\x1b[0m"
    read -p " [$DEFAULT] : " -r INPUT
    INPUT=${INPUT:-${DEFAULT}}

    # Collect result
    if [[ "${OPTIONS[@]}" =~ "$INPUT" ]]; then
        ANSWER=$INPUT
    else
        ANSWER=$DEFAULT
    fi
}


questionWithDefaultSimple () {

    # Positional argument should strictly come before named arguments

    QUESTION=$1
    read -ra OPTIONS  <<< "$2" # Keep options list

    if [ -n "$3" ]; then
        DEFAULT="$3"
    else
        DEFAULT="${OPTIONS[0]}"
    fi

    # Create pattern for fstring
    COL_SIZE=15
    FORMAT_STRING="%-${COL_SIZE}s %-${COL_SIZE}s %-${COL_SIZE}s %-
```

```
${COL_SIZE}s\n"

    # Show
    printf "$FORMAT_STRING" "${OPTIONS[@]}"

    # Ask user
    printf "\x1b[1;32m$QUESTION\x1b[0m"
    read -p " [$DEFAULT] : " -r INPUT
    INPUT=${INPUT:-${DEFAULT}}

    # Collect result
    if [[ "${OPTIONS[@]}" =~ "$INPUT" ]]; then
        ANSWER=$INPUT
    else
        ANSWER=$DEFAULT
    fi
}

numberedMenu () {

    # $1 = question to ask
    # $2 = elements separted by space
    # S3 (optionnal) = index of defaut element

    QUESTION=$1
    read -ra OPTIONS  <<< "$2" # Keep options list

    if [ -n "$3" ]; then
        DEFAULT=$3
    else
        DEFAULT=1
    fi

    # Show
    printf "\n"
    for i in $(seq ${#OPTIONS[@]}); do
        printf '    %3s %s\n' "$i)" "${OPTIONS[$(($i-1))]}"
    done
    printf '\n'

    # ask user
    printf "\x1b[1;32m$QUESTION\x1b[0m"
    read -p " [$DEFAULT] : " -r INPUT
    INPUT=${INPUT:-${DEFAULT}}

    # Collect result
    if [[ "$(seq ${#OPTIONS[@]})" =~ "$INPUT" ]]; then
        ANSWER="${OPTIONS[$(($INPUT-1))]}"
    else
        ANSWER="${OPTIONS[$(($DEFAULT-1))]}"
    fi
```

```
}


echo "execution de:"
echo 'askYesNo "Voulez vous continuer ?" true'
echo ''
askYesNo "Voulez vous continuer ?" true
echo $ANSWER
echo ""
echo '-------------'
echo ""
askYesNo "Voulez vous continuer ?" false
echo 'askYesNo "Voulez vous continuer ?" false'
echo $ANSWER

echo ""
echo '-------------'
echo ""
echo "execution de:"
echo 'split_args -n named_1 pos_1 -p_1 --named_long named pos_2'
echo ""
split_args -n named_1 pos_1 -p_1 --named_long named pos_2
echo "positional arguemnts : $POSITIONAL_ARGS"
echo "named arguements : $NAMED_ARGS"
echo "parameter arguments: $PARAM_ARGS"

echo ""
echo '-------------'
echo ""
echo "execution de:"
echo 'questionWithDefault "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6
arg7 arg8 arg9 arg10 arg11 arg12 arg13 arg14'
echo ""
questionWithDefault "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6 arg7
arg8 arg9 arg10  arg11 arg12 arg13 arg14"
echo $ANSWER

echo ""
echo '-------------'
echo ""
echo "execution de:"
echo 'questionWithDefault "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6
arg7 arg8 arg9 arg10 arg11 arg12 arg13 arg14" -d 8 -c 6'
echo ""
questionWithDefault "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6 arg7
arg8 arg9 arg10 arg11 arg12 arg13 arg14" -d 8 -c 6
echo $ANSWER

echo ""
echo '-------------'
```

```
echo ""
echo "execution de:"
echo 'questionWithDefaultSimple "Votre choix?" "arg1 arg2 arg3 arg4
arg5 arg6 arg7 arg8 arg9 arg10 arg11 arg12 arg13 arg14' arg6
echo ""
questionWithDefaultSimple "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6
arg7 arg8 arg9 arg10 arg11 arg12 arg13 arg14" arg6
echo $ANSWER

echo ""
echo '-------------'
echo ""
echo "execution de:"
echo 'numberedMenu "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6 arg7
arg8 arg9 arg10 arg11 arg12 arg13 arg14" 2'
echo ""
numberedMenu "Votre choix?" "arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8
arg9 arg10 arg11 arg12 arg13 arg14" 2
echo $ANSWER

echo ""
echo '-------------'
echo ""
```