INF600A: Laboratoire #3
Scripts Unix
Solution

Remarque : Les commentaires expliquant l'interface et le fonctionnement du script au début de chacun des fichiers ont été omis.

1. tf.sh

```
#!/bin/bash -

set -o nounset  # Pour signaler une erreur si une variable non definie est utilis
set -o errexit  # Pour signaler une erreur si un exit status non nul est produit

function usage {
    echo "usage:"; echo "$0 nomDeFonction [fichier]"; exit 1
}

[[ $# == 0 || $# -gt 2 ]] && usage

nom_de_fonction="$1"
if [[ $(uname) == "Darwin" ]]; then
  readonly debut_function=\
    "^[[:blank:]]*function[[:blank:]]*${nom_de_fonction}[[:blank:]]"
else
  readonly debut_function=\
    "^[[:blank:]]*function\b[^[:alnum:]]*\b${nom_de_fonction}\b"
fi

readonly fin_function="^[[:blank:]]*}"


if [[ $# == 1 ]]; then
    # Un seul argument: on va examiner tous les fichiers *.sh.
    liste_de_fichiers=$( find . -name "*.sh" | sort )
    plusieurs_fichiers=1
elif [[ $# == 2 ]]; then
    # Un fichier est explicitement indique: c'est cet unique fichier qu'on fouill
    liste_de_fichiers="$2"
    plusieurs_fichiers=0
fi

# La redefinition de IFS est necessaire pour traiter les noms de
# fichiers avec des espaces!
OIFS="$IFS"
IFS=$'\n'
for fich in $liste_de_fichiers; do
    if grep -E -q "$debut_function" "$fich"; then
        [[ $plusieurs_fichiers == 1 ]] && echo "$fich"
        sed -n "/$debut_function/,/$fin_function/p" "$fich"
    fi
done
IFS=$OIFS
```

2. num.sh

```
#!/bin/bash -

set -o nounset  # Pour signaler une erreur si une variable non definie est
set -o errexit  # Pour signaler une erreur si un exit status non nul est p

# Fonctions auxiliaires.
function num {
    # Fonction qui recoit un nom de fichier et, optionnellement, une
    # largeur *positive*. Elle affiche ensuite le contenu de chacune des
    # lignes avec le numero et l'indentation appropriee selon la largeur
    # indiquee. Si aucune largeur n'est specifiee, alors les numeros ne
    # seront pas necessairement alignes.
    #
    local fich="$1"
    local largeur=""
    if [[ $# == 2 ]]; then
        largeur="$2"
    fi

    local format="%${largeur}d. %s\n"

    awk "{ printf \"$format\", NR, \$0 }" "$fich"
}

function usage {
    echo "usage: "
    echo "  $0 [-k] fichier"
    echo "  Avec k > 0"
    exit 1
}

#
```

```
###################################
# PROGRAMME PRINCIPAL
###################################

[[ $# == 0 || $# -gt 2 ]] && usage

# On identifie et on valide les arguments.
largeur=""
if [[ $# == 2 ]]; then
    # Deux arguments, donc largeur specifie: on valide l'option.
    [[ ! $1 =~ ^-[0-9]+$ ]] && usage
    largeur="${1#-}"

    [[ $largeur == 0 ]] && usage # Doit etre un nombre superieur a 0.
    shift # Pour acceder au nom de fichier.
fi

fichier="$1"

# Les arguments sont valides, donc:
# - fichier = nom du fichier a traiter
# - largeur = largeur du champ decimal a utiliser pour les numeros de lign
#             (peut etre vide, auquel cas on utilise la largeur requise pa
# On appelle la fonction auxiliaire qui fait le travail.
num "$fichier" $largeur
```

3. map.sh

```
#!/bin/bash -

set -o nounset  # Pour signaler une erreur si une variable non definie est
set -o errexit  # Pour signaler une erreur si un exit status non nul est p

# On charge les fonctions qui pourront etre utilisees par map
for fich in MapFonctions/*.f; do
    . "$fich"   # Il y a un "." devant "$fich" == source!
done

function usage {
    echo "usage:"
    echo "$0 fonction [fichier...]"
    exit 1
}

[[ $# == 0 ]] && usage

fonction="$1"
shift

if [[ $# == 0 ]]; then
    cat | while read ligne; do
        $fonction $ligne
    done
else
    while [[ $# != 0 ]]; do
        cat $1 | while read ligne; do
            $fonction $ligne
        done
        shift
    done
fi
```