IUT - Département Informatique

*M1.101 - Utilisation des Systèmes Informatiques - 2019-2020*

# M1.101 TP6

## TP noté

## Table des matières



# 1 Consignes

Le travail est à faire en **binôme** et à rendre **avant** le **20/12/2019** sur la plateforme d'enseignement (par un seul membre du binôme) sous la forme d'un fichier `.tar.gz` contenant **uniquement** les deux fichiers suivants :

1. Le script Shell nommé `blogue.sh`
   - aéré et indenté;
   - écrit en utilisant des fonctions;
   - avec le nom des auteurs dans le cartouche du fichier;
   - commenté et documenté (présence d'un cartouche pour chaque fonction, et code facile à comprendre);
   - avec peu (idéalement pas) d'erreurs recensées par `schellcheck`.
2. Un rapport **PDF** qui précise :
   1. les auteurs du travail;
   2. ce que fait le script;
   3. la syntaxe d'appel du script;
   4. des traces d'exécution;
   5. les difficultés rencontrées lors du projet;
   6. le travail qui a été réalisé;
   7. les améliorations éventuellement faites.

Remarques supplémentaires :

- Toute remise **hors délais** ne sera pas corrigée (**la note est de 0**).
- Tout rendu fait en monôme ou trinôme ne sera pas corrigé (**la note est de 0**).
- Le plagia sera sévèrement sanctionné (la note est de 0 aussi bien pour le binôme source que le binôme destination et la procédure administrative adéquate est enclenchée).

— Le **strict** respect des consignes sera pris en compte dans la notation (vous codez au minimum ce que l'on demande, pas ce que vous voulez faire)
— Vérifiez immédiatement que vous avez accès à la plateforme d'enseignement avec votre groupe(les clés sont `S1A1`, `S1A2`, `S1B1`, `S1B2`, `S1C1`, `S1C2`, `S1D1`, `S1D2`).

Les semaines de travail sur ce projet sont celles du 02/12/2019, 09/12/2019, 16/09/2019.

# 2 Présentation du sujet : le gestionnaire de blogue

Il s'agit d'un script `bash` de gestion de blogue ou site statique quelconque. Il permet de contrôler l'ajout/suppression/modification des pages ainsi que la génération d'une version statique HTML et PDF du contenu. Les pages sont écrites à l'aide du langage `Markdown`[1].

Le script prend en entrée un nom de commande pour effectuer l'action correspondante (éventuellement suivit de ses paramètres). Si le nom de commande n'est pas précisé, le script affiche le message suivant (vous en déduirez aisément les actions à effectuer) :

```
$ ./blogue.sh
[ERREUR] 1 action est attendue
Usage : ./blogue.sh COMMANDE [PARAMETRE]

La liste des commandes est editer, supprimer, lister, construire

 - Éditer un document : ./blog.sh editer [PAGE]
 - Supprimer un document : ./blog.sh supprimer [PAGE]
 - Lister les documents : ./blog.sh lister
 - Générer les fichiers PDF/HTML : ./blog.sh construire
 - Visualiser le site : ./blog.sh visualiser [pdf]

La commande usage affiche cette documentation.
```

Le script est insensible à la casse du nom de la commande.

Les fonctionnalités à implémenter sont les suivantes :

— **Éditer**. L'application est capable de lancer l'édition d'une page. Le nom de la page (sans extension) est fourni en argument et l'application lance un éditeur sur le fichier `markdown/<nom_page>.md`. Si aucun nom n'est fourni, l'application affiche la liste des possibilités et permet à l'utilisateur de spécifier le fichier qui l'intéresse. Le choix de l'éditeur se fait de la façon suivante :
  — Si la variable d'environnement `EDITOR` est spécifiée, sa valeur est utilisée comme éditeur.
  — Sinon, si le programme `/usr/bin/editor` existe, ce programme est utilisé comme éditeur.
  — Sinon, le script doit tester la présence de différents éditeurs standards et sélectionner automatiquement le premier existant.
  — Une erreur est générée si aucun nom de page n'est indiqué ou si le nom est `index`.
— **Supprimer**. L'application supprime la page (sélectionnée de la même façon que pour éditer). Une demande de confirmation est effectuée avant la suppression.
— **Lister**. L'application est capable de lister l'ensemble des pages existantes (i.e., les pages du dossier `markdown`). Si il n'y a pas de pages, l'application doit afficher «`Il n'y a pas de fichier`».

1. https://michelf.ca/projets/php-markdown/syntaxe/

— **Construire**. L'application est capable de générer une version `HTML` (de plusieurs pages) et une version `PDF` (de un fichier) de son contenu. Le code `HTML` (1 fichier `html` par page) est généré dans le dossier `web` et la version `PDF` est générée dans le fichier `blog.pdf`. C'est l'outil `pandoc`[2] qui se charge de faire les conversions. Pour la version `HTML`, une première étape consiste à générer le fichier `markdown/index.md` qui contient la liste des pages du site sous forme de liens `HTML` ainsi que la mention suivante : *Contenu actualisé le par <nom de l'utilisateur> sur* .
— **Visualiser**. L'application est capable de lancer (`xdg-open`) un navigateur pour afficher le contenu de `web/index.html` qui agit comme un sommaire. Elle est capable de visualiser le fichier pdf si l'argument optionnel `pdf` est fourni.

Si vous le souhaitez, vous pouvez utiliser `blogue.sh` pour éditer votre compte rendu (le sujet de ce projet a été produit avec la correction des enseignants).

# 3 Travail facultatif

Si les tâches demandées sont toutes réalisées, vous avez la possibilité d'effectuer des tâches facultatives pour obtenir des points bonus (attention, la correction ne les prend pas en compte si le reste n'est pas fait).

— Une erreur est générée si le nom de la page comporte une extension.
— Le programme fonctionne très bien, même si les chemins à utiliser comportent des espaces.
— De nombreuses variables d'environnement peuvent être spécifiées, de façon optionnelle, pour paramétrer le rendu des documents.
— Il est possible de spécifier un thème pour produire un résultat visuellement différent avec contenu identique.

# 4 Contraintes

— Le script doit être composé de plusieurs fonctions ayant chacune une petite responsabilité claire. Il doit y avoir entre autre une fonction main qui lance le programme et une fonction usage qui explique comment utiliser l'application.
— La langue anglaise doit être utilisée pour nommer les variables/fonctions et rédiger les commentaires.
— Les différents messages d'erreurs doivent être affichés sur la sortie d'erreur.
— shellcheck indique que le code est parfait.
— Le début du script shell doit contenir les commandes suivantes :

```
#!/bin/bash
set -o errexit # Exit if command failed.
set -o pipefail # Exit if pipe failed.
set -o nounset # Exit if variable not set.
# Remove the initial space and instead use '\n'.
IFS=$'\n\t'
```

2. http://pandoc.org/MANUAL.html