Semestre 1 – Piscine PHP

Rush 3

Bienvenue dans ce troisième Rush de piscine ! ☺

Ce rush va vous demander d'approfondir vos connaissances sur git et d'appliquer ce que vous avez appris pendant la piscine.

Informations quant au sujet

Il ne vous est **PAS** demandé de faire de la comparaison de fichiers / merge ou de push comme le fait git. Git est un outil très puissant ayant demandé plusieurs années de développement à une grande équipe de professionnels. Il s'agit ici de faire une version très simplifiée où vous vous contenterez de compresser et de stocker les fichiers dans le dossier ciblé.

Pour toute commande inconnue et pour tout mauvais usage de commande (mauvais arguments / pas assez d'arguments, …) afficher votre man (infos plus loin dans le sujet) et retourner 1.

à moins qu'il ne soit précisé autrement, toute commande réussie return 0 et affiche ce qui est demandé. Tout échec / erreur devra retourner 1 et afficher ce qui est demandé.

Tout affichage (réussite, erreur, man, …) sera suivit par un retour à la ligne.

Le sujet étant vaste, il est tout à fait possible de faire des bonus (affichage en couleurs, flag sur une commande, commande additionnelle …).
Pour être validés, les bonus ne peuvent être effectués **QU'APRÈS** les commandes de bases (sans quoi ils ne seront pas notés).
Les bonus ne doivent en **AUCUN CAS** interférer avec les parties obligatoires (ex : ne pas ajouter des lignes de texte supplémentaires à celles qui sont demandés dans les fonctions).
Tout bonus (commandes en plus ou flags) doit être inclus dans votre man où il sera détaillé.

Sommaire

Etape 1 : Un bon début

Rendu : Piscine_PHP_Rush_3/MyGitLight.php

Écrire un programme exécutable « MyGitLight.php » qui prend en ligne de commande, l'argument "init" suivit d'un path. MyGitLight devra copier son propre code source et le placer dans un dossier. MyGitLight que vous aurez créé dans le path.

Exemple :

```
piscine@samsung:~# ls
MyGitLight.php
piscine@samsung:~# php MyGitLight.php init /cheminImaginaire
piscine@samsung:~# ls
MyGitLight.php
piscine@samsung:~# cd /cheminImaginaire
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight
piscine@samsung:/cheminImaginaire# cd .MyGitLight
piscine@samsung:/cheminImaginaire/.MyGitLight# ls -a
. .. MyGitLight.php
```

Affichage demandé pour les erreurs :

- mauvaises permissions: "could not access folder : bad permissions"
- présence d'un .MyGitLight dans le dossier : "this folder already has a MyGitLight"
- le dossier n'existe pas : "could not access $path"

Réussite : ne rien afficher

Vous pouvez avoir plus d'un fichier pour ce sujet mais dans quel cas vous DEVREZ aussi les copier avec la commande init. Seuls les fichiers présents dans le dossier .MyGitLight que vous aurez créé seront évalués.

Vous devez avoir un fichier MyGitLight.php, c'est lui qui sera appelé lors des corrections.

Etape 2 : Passons aux choses sérieuses

Rendu : Piscine_PHP_Rush_3/MyGitLight.php

Vous devez ajouter les commandes “add”, “commit”, “rm” et “log”.

- La commande “add” appelée sans argument, copie tout le répertoire courant ainsi que les sous dossiers (pensez à utiliser la récursivité) et les place dans un dossier que vous aurez créé dans votre dossier .MyGitLight (ne copiez pas le dossier .MyGitLight !!!). Il devra être possible de sélectionner spécifiquement certains fichiers ou dossiers en les passant en argument sous forme de liste à la commande add.

Exemple :

```
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php add file1 folder1/
```

- La commande “commit” prend pour argument un message et crée une tarball avec le contenu de votre dossier. Vous devez attribuer un id à chaque commit de façon à pouvoir y retourner par la suite.

- La commande “rm” prend pour argument un ou plusieurs fichiers/dossiers. Elle supprime ces derniers du dossier de suivi et du répertoire de travail. Attention la suppression ne se fait que si les fichiers sont à la fois présents dans le dossier de suivi et le répertoire de travail.

- La commande “log” affiche tous les commits sous forme “$id $message” triés par id et par ordre décroissant.

Exemple :

```
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight
piscine@samsung:/cheminImaginaire# touch test
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php rm test
File not found
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php add
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php commit “adding test”
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php log
1 adding test
piscine@samsung:/cheminImaginaire# touch test2
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test test2
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php add
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php commit “adding test2”


piscine@samsung:/cheminImaginaire# ls -a
```

```
. .. .MyGitLight test test2
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php log
1 adding test
2 adding test2
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php rm test2
rm test2 ok
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test
```

C'est vous qui êtes en charge de décider comment vous stockez vos informations, vos choix seront regardés en soutenance. Les id de commits commencent toujours à 1 et chaque nouveau commit ajoute 1 à la valeur de façon à ce que 1er = 1, 2nd = 2, …

Affichage demandé pour les erreurs :

- commit :
    - pas de message : "A commit message is needed"
    - rien à ajouter / rien de modifié (pas de **add** avant) : "nothing to add"

Réussite : ne rien afficher (sauf pour le log qui affiche les commits).

Etape 3 : Toujours plus !

Rendu : Piscine_PHP_Rush_3/MyGitLight.php

Vous devez ajouter la commande **status**.

La commande status affiche quels sont les fichiers qui ont été modifiés. Une syntaxe différente de celle de git est demandée ici.

Exemple d'affichage demandé pour **status** :

```
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php status
modified :
untracked :
deleted :
piscine@samsung:/cheminImaginaire# touch test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php status
modified :
untracked :
test
deleted :
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php add
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php status
modified :
untracked :
deleted :
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php commit “adding test”
piscine@samsung:/cheminImaginaire# rm test
piscine@samsung:/cheminImaginaire# touch test2
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test2
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php status
modified :
untracked :
test2
deleted :
test
```

status possède la syntaxe suivante :
“modified:\n”
“- $file_name\n” * nombre de fichiers modifiés (créé avec le flag add)
“untracked:\n”
“- $file_name\n” * nombre de fichiers pas présents dans le dossier de suivi
“deleted:\n”
“- $file_name\n” * nombre de fichiers ayant été supprimés

Important :

- Trier les fichiers par ordre alphabétique pour chaque catégorie
- Le nom des fichiers inclut le path jusqu'à la racine du projet (là où est le .MyGitLight)
- Même vides, les titres des catégories doivent tout de même être affichés.

La commande status de git est complexe, il vous est demandé ici de faire quelque chose de plus simple. Si vous souhaitez faire du bonus ici, assurez-vous que cela n'interfère pas avec ce qui est demandé. (vous pouvez modifier l'affichage mais ce doit être avec une commande de plus ou un flag)

Autre exemple :

```
php .MyGitLight/MyGitLight.php status
modified:
dossierTest/fichierModifié1.php
dossierTest/fichierModifié2.php
z_file.txt
untracked:
nouveauFichier.php
unToutNouveauFIchier.php
deleted:
FichierSupprimé1
```

Faites attention aux valeurs de retour et au "\n" en fin d'affichage !!!!

Etape 4 : Encore un effort !

Rendu : Piscine_PHP_Rush_3/MyGitLight.php

Vous devez ajouter la commande **checkout**.

La commande **checkout** est utilisée pour passer d'un commit à l'autre. Elle prend comme argument l'ID d'un commit.

Dès que la commande est appelée, vous devez supprimer tout ce qui se trouve dans le dossier de travail et le remplacer par ce qui se trouvait dans le commit demandé.

Exemple :

```
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight
piscine@samsung:/cheminImaginaire# touch test
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php add
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php commit “adding test”
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php log
1 adding test
piscine@samsung:/cheminImaginaire# rm test
piscine@samsung:/cheminImaginaire# touch test2
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test2
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php add
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php commit “adding test2”
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test2
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php log
1 adding test
2 adding test2
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php checkout 1
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php checkout 2
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test2
```

Affichage demandé pour les erreurs :

- pas d'id en paramètre : “checkout needs an ID”

- l'id n'existe pas / est incorrect : “bad id”

En cas de réussite ne rien afficher.

Etape 5 : Un dernier effort !

Rendu : Piscine_PHP_Rush_3/MyGitLight.php

Vous devez ajouter la commande **diff**.

diff est une commande complexe, vous la ferez avec la librairie externe Diff que vous trouverez ici : http://code.stephenmorley.org/php/diff-implementation/

Le code de cette librairie **DOIT** être inclus dans votre code source et être copiée avec la commande **init** sans quoi elle ne pourra pas marcher.

Quand le flag diff est utilisé, faire usage de la librairie Diff sur **TOUS** les fichiers possédant une différence (ignorez les fichiers n'ayant pas été modifiés) dans l'ordre alphabétique (le path est à inclure dans le nom, comme pour **status**).Ne rien afficher si aucune différence est présente.

Les fonctions de la librairie à utiliser sont : Diff::toString et Diff::compareFiles.

Exemple :

```
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight
piscine@samsung:/cheminImaginaire# touch test
piscine@samsung:/cheminImaginaire# ls -a
. .. .MyGitLight test
piscine@samsung:/cheminImaginaire# echo -e “hello\n1000\a” > test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php add
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php commit “adding test”
piscine@samsung:/cheminImaginaire# echo -e “hello\n2\na” > test
piscine@samsung:/cheminImaginaire# php ./.MyGitLight/MyGitLight.php diff
test
  hello
- 1000
+ 2
  a
```

Syntaxe demandée :
“$filename”
sortie de Diff pour les fichiers
séparer tous les fichiers par un retour à la ligne supplémentaire

Etape 6 : Enfin fini !

Rendu : Piscine_PHP_Rush_3/MyGitLight.php

Vous devez ajouter la commande **help**.

Vous êtes ici libres de créer le man pour votre MyGitLight.

“libres de créer” != “bâcler le travail”

Pensez à regarder comment sont fait les mans pour faire le votre

C’est dans cette partie que vous devrez détailler tous les bonus que vous aurez fait dans ce projet s’il y en a (flags, arguments en plus, …).