EPFL CS212 : Semaine 01

J.-C. Chappelier, E. Bugnion EPFL

% Rev. 2021.02.17 / 3

Table des matières

Projet programmation système W01

Introduction

Bienvenue dans ce « cours-projet » !
Il sera composé de **deux parties indépendantes** :

1. une prise en main, sur trois semaines, de l'environnement et des outils, constituée de trois rendus individuels ;
2. le développement, sur neuf semaines, du projet propremement dit, constitué trois rendus de groupe (de deux étudiants) ; ce projet consiste cette année en la création d'un système de gestion de fichiers images composé de deux grosses parties :

(1) un utilitaire en ligne de commande pour gérer des images dans une base de donnée à un format spécifique, inspiré de celui utilisé par Facebook ;
(2) un serveur Web pour gérer ces mêmes images.

(et deux semaines supplémentaires : une pour souffler un peu en milieu de projet et une pour tout terminer correctement à la fin.)

Le détail hebdomadaire se trouve ici : http://progos.epfl.ch/projet/

Les objectifs de ce cours-projet sont de vous permettre :

1. de développer une application concrète en C incluant des pointeurs et une gestion simple de fichiers ;
2. d'avoir un premier contact expérimental avec plusieurs outils usuels de « développement système », tels que le contrôle de versions (`git`), les pages de documentation (manpages), les Makefiles et les outils de débogage ;
3. d'installer et utiliser des bibliothèques logicielles (« *libraries* » en anglais) ;
4. de « *refactoriser* » du code au fur et à mesure des contraintes.

A partir de la 4e semaine, vous devrez travailler par groupes de deux. Chaque groupe aura son propre dépôt (« *github repository* »). Vous aurez chaque semaine de nouvelles instructions sur ce site et recevrez parfois du nouveau code via `git`, un outil permettant la gestion du travail en commun (« outil de gestion de versions »).

Concrètement, cette semaine

Objectifs

Cette semaine, nous allons procéder à la mise en place et la découverte des outils qui vous seront utiles dans votre projet :

1. mise en place de GitHub ;
2. découverte de Git ;
3. découverte des manpages Unix ;
4. compilation, modification et soumission d'un code C (sera approfondi la semaine prochaine) ;
5. documentation du code avec Doxygen.

Ce que vous aurez à rendre pour cette semaine (délai : dimanche 7 mars 23:59) sera (expliqué plus loin, mais résumé ici) :

— une version traduite en français de `hello_world.c` ;
— une version opérationnelle de `sha.c` (calcul de SHA256 de chaînes) ;
— une version opérationnelle de `thumbify.c` (production d'une image réduite d'une image existante) ;
— un `Makefile` simple compilant deux cibles, « démos » des bibliothèques utilisées : `sha.c` et `thumbify.c`.

Environnement de travail

Pour ce cours-projet, vous pouvez travailler :

— en accédant à distance aux machines virtuelles IC-CO-IN-SC (Ubuntu 18.04) depuis votre propre machine via vdi.epfl.ch [plus de détails sous ce lien si nécessaire, ou sur cette page d'un autre cours de M. Chappelier] ;

— ou sur votre propre machine si elle est sous Linux ;
(**Note :** sous Windows, depuis Windows 10, il existe aussi un « terminal Ubuntu » que vous pouvez peut être [pas testé] utiliser pour ce cours ; plus de détails ici en html, ici en PDF et ici en Markdown, changez simplement `g++` en `gcc` et `.cc` en `.c`) ;
— ou via un container Docker (voir cette page) ;
— ou sur une machine virtuelle Ubuntu que vous aurez créée (voir cette page pour VirtualBox ou cette page pour VMWare).

Aucun autre système ne sera supporté (ni accepté). Si vous travaillez sur votre propre environnement (autre que ceux cités ci-dessus), c'est donc à vos propres risques et périls, **sans garantie de soutien de notre part**. (Nous fournissons néanmoins ici un bref guide pour l'installation d'un environnement de développement sur macOS.)

En plus d'un environnement de développement C usuel (compilateur, éditeur, débogueur), il vous faudra les outils suivants (`sudo apt install <package>` pour les installer sur une Debian/Ubuntu):

— `git` pour la gestion du projet (cf ci-dessous) ;
— un client SSH comme par exemple `openssh-client` pour communiquer avec GitHub (voir ci-dessous) ;
— les pages de documentation (*manpages*) : `manpages` et `manpages-dev` ;
— `doxygen` pour produire automatiquement de la documentation à partir de votre code ;
— `libssl-dev` une bibliothèque de cryptographie que nous utiliserons pour calculer des « clés » ;
— `libvips-dev` une bibliothèque de gestion d'images ;
— `libjson-c-dev` une bibliothèque pour la communication JSON avec le webserver.

Nous utiliserons également la bibliothèque Mongoose pour créer le server Web, mais le code source (de cette bibliothèque) vous sera fourni directement.

Remarque spécifique aux VM fournies IC-CO-IN-SC (Ubuntu 18.04) : pour de (mauvaises) raisons techniques, il n'est pas possible de compiler n'importe où (= dans n'importe quel répertoire) sur ces VMs. **Seuls deux endroits** permettent de compiler :

— (conseillé) soit dans le répertoire `posixfs` (mais ce répertoire n'est pas visible en tant que tel dans votre myfiles sur d'autres OS ; ce qui n'est pas un inconvénient pour ce projet-ci...) ;
— soit dans le répertoire `myfiles/Programmation` qui est à créer, une fois pour toute en début de semestre, en lançant la commande : `create_prog`.

GitHub

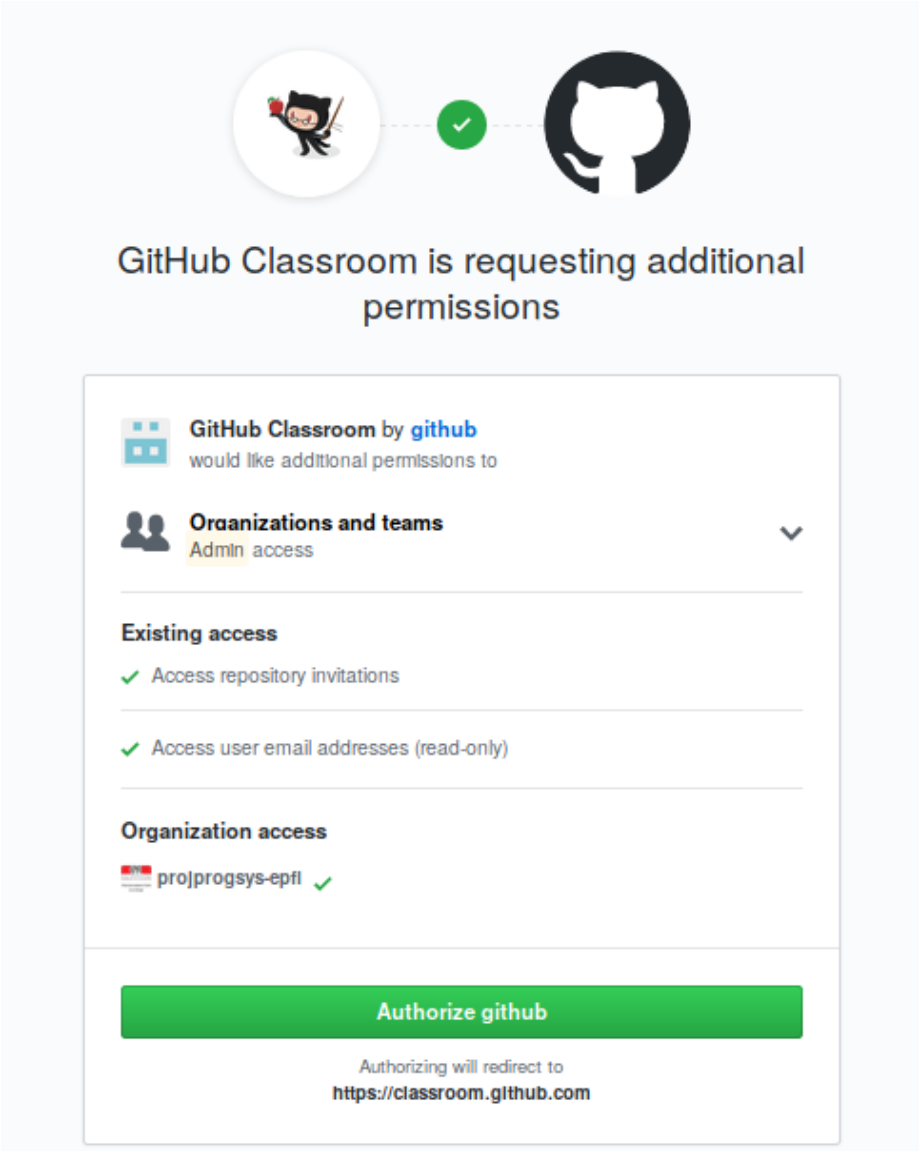
GitHub est un dépôt Git public accessible sur le Web. Chaque étudiant va recevoir au départ un dépôt personnel sur GitHub pour le premier devoir de ce

cours (inscription individuelle ci-dessous). Plus tard, vous recevrez des dépots GiHub supplémentaires (individuel ou par groupe) pour chaque rendu pour lequel ce sera nécessaire. Tout cela sera précisé ultérieurement.

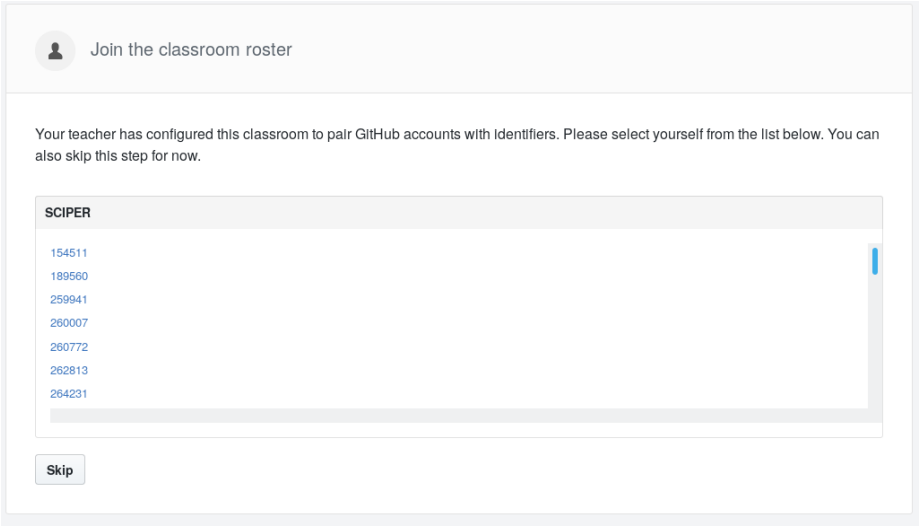
La première chose dont vous avez besoin est donc un compte GitHub. Si vous n'en avez pas encore, veuillez en créer un en vous enregistrant ici. Un compte gratuit suffit pour ce cours, il n'est pas du tout nécessaire d'avoir un compte payant.
(et si vous avez déjà un compte GitHub, utilisez le pour ce cours).

Ensuite, une fois que vous avez un compte GitHub, il vous faut rejoindre le premier devoir de ce cours en suivant ce lien : https://classroom.github.com/a/mDG3zUvw.

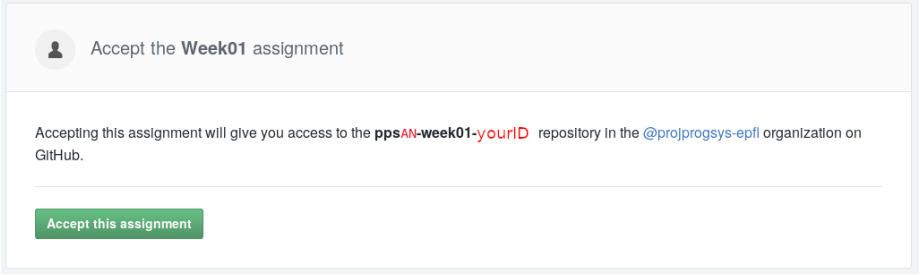
GitHub Classroom va (peut être) alors vous demander le droit d'accéder à vos dépôts :



puis de rejoindre le premier devoir (cliquez bien sur **votre numéro SCIPER** ; *ne* prennez *pas* celui de quelqu'un d'autre !) :



et enfin de créer un dépôt GitHub pour ce premier devoir :



Une fois ces deux invitations acceptées, vous devriez recevoir un message de GitHub vous indiquant que vous avez rejoint le devoir « Week01 » et vous avez dans vos dépôts GitHub, un dépôt `pps21-week01-VOTRE_GITHUB_ID` dont l'URI devrait ressembler à ceci :

```
git@github.com:projprogsys-epfl/pps21-week01-GITHUBID.git
```

Pour que cet URI fonctionne, vous devez enregistrer votre clé publique SSH dans GitHub. Vous pouvez le faire ici.

Si vous n'avez pas encore de clé SSH, vous pouvez en générer une sous Linux en faisant

```
ssh-keygen
```

puis copiez le contenu du fichier `~/.ssh/id_rsa.pub` dans GitHub à l'endroit mentionné ci-dessus.
Demandez de l'aide aux assistants si vous en avez besoin sur ce point (SSH).

NOTE : vous pouvez aussi utiliser `https` pour votre `REPO_URI` :

`https://github.com/projprogsys-epfl/pps21-week01-YOURID.git`

mais vous aurez à vous authentifier à chaque fois.

Git

Ce n'est pas le but de ce cours-projet que de vous *enseigner* les gestionnaire de versions, ni tous les détails de `git` ; cela sera fait dans votre cours de « Software Engineering » l'an prochain. Le but ici est de vous faire prendre un premier contact avec ce genre d'outils et vous permettre de travailler efficacement en utilisant une partie miminale de ce qu'il offre.

Pour ce cours, vous ne devriez pas avoir besoin de connaître plus de quelques commandes simple de `git`:

- `git clone [REPO_URI]`
- `git pull`
- `git add [FILE]`
- `git commit -m "Commit message"`
- `git push`
- `git status`
- `git tag`

Pour chaque commande, vous pouvez obtenir de l'aide de la part de `git` en tapant :

```
git help <COMMAND>
```

Pour découvrir git : voyez cette page complémentaire.

Et ensuite...

Allez maintenant récupérer la suite de ce TP dans votre repo GitHub (si tant est que vous ayez reçu l'email de confirmation !). Pour cela, exécutez la commande suivante :

```
git clone REPO_URI
```

Cela doit créer un répertoire local sur votre machine, de même nom : `pps21-week01-YOURID` (avec *votre* identifiant GitHub à la place).

Allez dans ce répertoire :

```
cd pps21-week01-YOURID
```

Vous devriez y trouver un répertoire `done` contenant :

- un fichier MarkDown `what_to_do_next.md` ;
- trois fichiers C : `hello_world.c`, `sha.c` et `thumbify.c` ;
- un `Makefile` (partiel) ;

— un fichier `libvips.supp` optionnel de configuration pour `valgrind` (pour ceux qui savent ce que c'est ; nous l'utiliserons bien plus tard dans le projet) ;
— une image `papillon.jpg` ;
— deux répertoires : `ex_single` et `ex_multiple` servant d'exemple pour la suite (Makefiles).

Ouvrez le `what_to_do_next.md` (soit dans un éditeur de texte, soit dans un viewer markdown ; sur les VMs du cours vous pouvez utilisez ReText (`retext`) qui fait les deux) et continuez les exercices de cette semaine depuis là-bas.

NOTE : pour voir du MarkDown directement dans Firefox, vous pouvez ajouter un plugin comme ceux-ci :

— https://addons.mozilla.org/fr/firefox/addon/markdown-viewer-chrome/
— https://addons.mozilla.org/fr/firefox/addon/markdown-viewer-webext/

ou similaires.

Conclusion

Nous reproduisons ici la conclusion que vous devez trouver en fin du fichier `what_to_do_next.md` (**ATTENTION !!** Ici, ce **N'EST PAS** la fin de la série ; vous **devez** continuer par le fichier `what_to_do_next.md` ; nous reproduisons ici sa conclusion uniquement pour information et rappel) :

Les quatre choses que vous ayez à rendre, d'ici au dimanche 7 mars 23:59, sont les fichiers `hello_world.c`, `sha.c`, `thumbify.c` et `Makefile` corrigés/complété comme demandé, ainsi que le fichier `time.csv` tel que décrit ci-dessus et dans le barème du cours ; ils sont à mettre dans votre dépôt GitHub dans le répertoire `done/` (et à `commit` + `push`). Nous vous recommandons de faire un `make feedback` avant de faire votre rendu définitif.

Le rendu se fait simplement avec le `commit + push`. Nous clônerons de notre coté l'état de votre dépôt le lundi matin dès la première heure.

Par ailleurs, vous pouvez commencer à rechercher un binôme pour la partie travail par groupe de deux qui commencera dès le 17 mars. Nous vous indiquerons le moment venu comment inscrire votre groupe.