université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

Projet IPF
LDD2 IM

Git

Chaque semaine, une feuille de route accompagnera la séance de projet. Elle ne contient que des suggestions pour vous aider à avancer dans le projet.

Objectif

Le but de cette feuille est de :

1. se familiariser avec `git` et ses concepts, en ligne de commande
2. organiser le dépot du projet
3. commencer à prendre en main gitlab
4. apprendre à manipuler les aspects impératifs d'OCaml

1 Prise en main de gitlab

Dans cette section, on se connecte à l'URL Gitlab de l'Université avec un navigateur Web.

1. Se connecter à `https://gitlab.dsi.universite-paris-saclay.fr` (prenom.nom/mot de passe du compte Paris-Saclay)
2. Noter son nom d'utilisateur gitlab (menu en haut à droite, `@prenom.nom` mais peut être différent)
3. Dans votre espace :
   — Crée un nouveau projet vide, privé, sans README.

2 Génération d'une clé SSH

L'authentification avec le dépôt distant se fait de préférence en SSH.

1. Exécuter dans un terminal la commande

   ```
   $ cat ~/.ssh/id_rsa.pub
   ```

   Si le fichier n'existe pas faire l'étape suivante, sinon aller directement à l'étape 4.
2. Exécuter la commande `ssh-keygen`, accepter tous les choix avec Enter.
3. Refaire la commande 1
4. Copier la ligne `ssh-rsa ...` affichée dans le terminal (surligner/bouton droit copier)
5. Repartir sur l'interface Web `gitlab`
6. Dépiler votre menu de profil (en haut à droite) → Preferences
7. Dans la barre de gauche choisir, SSH Keys
8. Coller la ligne dans la zone de texte et sauver

3 Introduction à `Git`

Dans cette section, on effectue toutes les opérations `git` en ligne de commande dans un terminal.

1. Cloner le dépôt `git` que vous venez de créer :

   ```
   git clone git@gitlab.dsi.universite-paris-saclay.fr:prenom.nom/votre-projet.git
   ```

   Vous devez **modifier** `prenom.nom` par votre nom d'utilisateur et `votre-projet` par le nom du projet. Cette URL peut être récupérée directement depuis l'interface Web (menu Clone).

2. Créer un fichier texte `README.md` (ou le modifier s'il existe). Indiquer :

```
# Projet XXXX

## Membres
  Foo Bar

## Description
  Todo
```

où `XXXX` est le nom donné à votre projet et `Foo Bar` sont vos noms et prénoms. Faire ensuite `git add README.md` puis `git commit` avec un message raisonnable. Il faut ensuite faire `git push` pour envoyer le fichier sur le serveur. Constater que dans l'interface Web, le contenu du fichier `README.md` s'affiche joliment (le format `md` est du Markdown, dans lequel `#, ##, ###` sont des titres de section, `*foo*` met du texte en `gras`, ...).

3. Télécharger l'archive du projet et y ajouter tous les fichiers dans votre dépot `git`. Ne pas oublier le fichier `.gitignore` qui est un fichier « caché ».
4. Vous assurer que vous avez fait `git add`, `git commit` et `git push`
5. Constater que les fichiers sont maintenant visibles sur l'interface Web.

4 Réflexions initiales

Ajouter au fichier `maze.ml` une fonction `dump_file : in_channel -> unit`. Cette dernière prend en argument un fichier ouvert en lecture et réaffiche son contenu dans la console.

Modifier le fichier `maze.ml` pour appeler votre fonction sur un fichier dont le nom est donné sur la ligne de commande (`Sys.argv`).

Remarque On s'attend à ce que tout le monde soit capable de faire cette feuille pour la deuxième séance.

Synchronizer le code OCaml sous `git` et vérifier que :

— le code sous `gitlab` est bien le bon

— votre binôme a bien récupéré la bonne version

Vous pouvez ensuite commencer à réfléchir à une structure de données pour les grilles de labyrinthes et l'algorithme vous permettant de charger un fichier de labyrinthe.

Un bon test est ensuite de :

— lire un fichier d'entrée, puis créer la structure de données correspondante

— afficher cette structure de données avec le même format que celui d'entrée

Votre sortie doit être identique au fichier d'entrée. Cela vous permettra en particulier de vérifier (entre les deux membres du binôme aussi) que vous avez la bonne notion de ligne et de colonne.