

02B : Se repérer et remettre un travail

420-3A6-ST : Programmation système

Dany Gagnon

Cégep de Saint-Jean-sur-Richelieu

Mercredi 2 septembre 2026

Table des matières

- ① Lire le terminal
- ② Arborescence et chemins
- ③ Trouver de l'aide
- ④ Relier la VM à GitLab
- ⑤ Pratique guidée
- ⑥ Synthèse

La question de la séance

Je viens d'ouvrir un terminal.
Comment retrouver mes fichiers
et remettre mon travail sans le perdre ?

Nous allons avancer une commande à la fois, puis vérifier le résultat.

Ce que nous allons faire

- lire une commande avant de l'exécuter ;
- nous déplacer avec `pwd`, `ls` et `cd` ;
- créer et manipuler quelques fichiers sans travailler au hasard ;
- relier la VM à GitLab ;
- remettre l'exercice 01 et confirmer sa réception.

À la fin, chaque étape laissera une preuve visible dans le terminal ou dans GitLab.

Lire le terminal

Notre point de départ

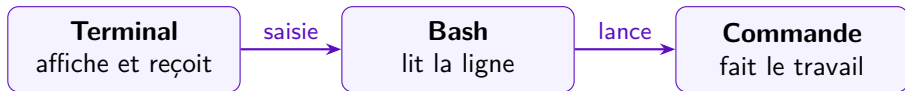
```
da1234567@3a6-arch:~$
```

Que pouvons-nous déjà lire ?

- `da1234567` : le compte connecté ;
- `3a6-arch` : le nom de la VM ;
- `~` : le répertoire personnel ;
- `$` : l'endroit où écrire une commande.

Le texte avant le curseur est **l'invite**. Il ne faut pas le retaper.

Terminal, shell et commande



Dans ce cours, nous écrivons les commandes dans un terminal et Bash les interprète.

Lire une commande de gauche à droite



```
ls -la /tmp
```

`ls` le nom de la commande ;

`-la` des options qui changent son comportement ;

`/tmp` la cible donnée à la commande.

Avant d'appuyer sur Entrée : **où suis-je, quelle est la cible, que fera la commande ?**

Trois touches utiles dès maintenant

Tabulation complète un nom de fichier ou de commande ;

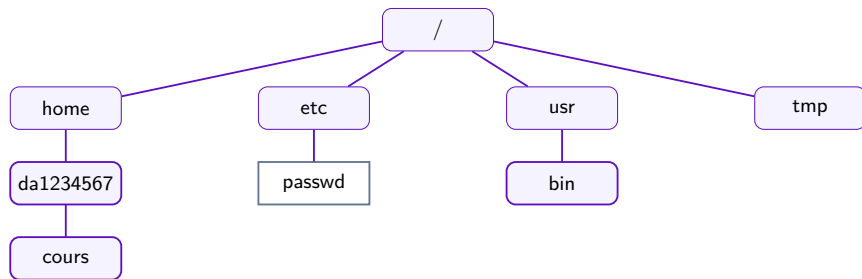
Flèche vers le haut rappelle une commande précédente ;

Ctrl+C interrompt une commande qui attend ou qui dure trop longtemps.

Utilisez la tabulation. Elle évite beaucoup de fautes de frappe dans les chemins.

Arborescence et chemins

Une seule arborescence



Sous Unix, tous les chemins partent de la racine /.

Toujours savoir où l'on se trouve



Le terminal travaille depuis un **répertoire courant**.

- `pwd` affiche ce répertoire ;
- `ls` montre ce qu'il contient ;
- `cd chemin` change de répertoire.

Après chaque `cd`, lancez `pwd`. Le résultat doit correspondre à ce que vous aviez prévu.

Deux façons de nommer le même fichier

Si le répertoire courant est `/home/da1234567/cours` :

Forme	Exemple	Point de départ
absolue	<code>/home/da1234567/cours/notes.txt</code>	la racine <code>/</code>
relative	<code>notes.txt</code>	le répertoire courant

Le fichier visé est le même. Seul le point de départ change.

Trois raccourcis à comprendre

- . le répertoire courant ;
- .. le répertoire parent ;
- ~ le répertoire personnel.

`cd` sans argument revient aussi au répertoire personnel.

Naviguer en gardant ses repères

```
$ pwd
/home/da1234567
$ ls
$ mkdir -p cours/3a6
$ cd cours/3a6
$ pwd
/home/da1234567/cours/3a6
$ cd ..
$ pwd
/home/da1234567/cours
```

Lisez la sortie après chaque déplacement. Une commande à la fois suffit.

Les gestes de base sur les fichiers

Commande	Effet
<code>touch notes.txt</code>	crée un fichier vide
<code>cat notes.txt</code>	affiche le contenu d'un fichier
<code>cp notes.txt copie.txt</code>	copie un fichier
<code>mv copie.txt archive.txt</code>	déplace ou renomme
<code>rm -i archive.txt</code>	demande avant de supprimer

Pour apprendre, travaillez dans `~/cours/3a6/essai`. Vérifiez avec `ls` après chaque opération.

Trouver de l'aide

Commencer par l'aide courte

Si vous connaissez le nom de la commande :

`ls -help` rappelle les options de `ls` ;

`man ls` ouvre sa documentation complète ;

`help cd` explique `cd` dans Bash.

Il n'est pas nécessaire de mémoriser toutes les options. Il faut savoir où les retrouver.

Lire juste ce dont on a besoin

Ouvrez le manuel de `mkdir` :

```
$ man mkdir
```

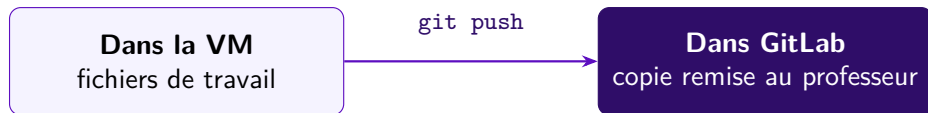
Cherchez l'option `-parents` avec `/parents`.

Espace	page suivante
/mot	chercher vers l'avant
n	prochaine occurrence
q	quitter

Fermez ensuite le manuel avec `q`.

Relier la VM à GitLab

Le dépôt existe à deux endroits



Enregistrer un fichier dans la VM ne l'envoie pas automatiquement dans GitLab.

1. Accepter les invitations

Dans votre courriel du Cégep :

- ① acceptez l'invitation au modèle de l'exercice 01 ;
- ② acceptez l'invitation à votre remise ;
- ③ connectez-vous à GitLab avec la même adresse ;
- ④ ouvrez le projet dont le nom est votre DA.

Vérification

Le chemin du projet doit finir par votre propre DA. N'écrivez pas dans le modèle.

2. Dire à Git qui crée les commits

Remplacez les valeurs par votre nom et votre courriel :

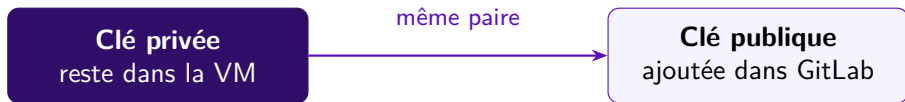
```
$ git config --global user.name "Votre prénom et votre nom"  
$ git config --global user.email "1234567@cstjean.qc.ca"
```

Vérifiez :

```
$ git config --global --list
```

Cette configuration se fait une fois dans la VM.

3. Une clé SSH possède deux parties



SSH prouve que la clé privée correspond à la clé publique sans envoyer la clé privée.

4. Créer la clé publique

Utilisez votre adresse du Cégep :

```
$ ssh-keygen -t ed25519 -C "1234567@cstjean.qc.ca"
```

Acceptez le chemin proposé. Choisissez ensuite une phrase secrète que vous pouvez retenir.

Affichez la partie publique :

```
$ cat ~/.ssh/id_ed25519.pub
```

Copiez toute la ligne, de `ssh-ed25519` jusqu'à votre adresse.

5. Ajouter la clé dans GitLab

Dans GitLab :

- ① ouvrez votre avatar, puis *Edit profile* ;
- ② choisissez *Access*, puis *SSH keys* ;
- ③ collez la clé publique et donnez-lui le titre VM 3A6 ;
- ④ ajoutez la clé.

Testez ensuite la connexion :

```
$ ssh -T git@gitlab.com
```

Au premier essai, confirmez l'identité de l'hôte avec *yes*. GitLab doit ensuite vous souhaiter la bienvenue.

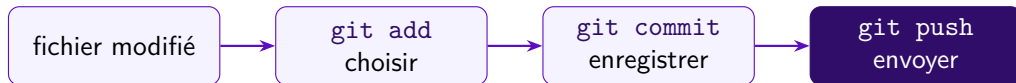
6. Cloner votre remise

Dans votre projet personnel, ouvrez *Code* et copiez l'adresse *Clone with SSH*.

```
$ mkdir -p ~/cours/3a6  
$ cd ~/cours/3a6  
$ git clone ADRESSE_COPIEE  
$ cd 1234567  
$ ls
```

Remplacez 1234567 par votre DA. Le dossier doit contenir `enonce.pdf`, `verifier-installation.sh` et `Makefile`.

Ce que Git enregistre



Un commit reste dans la VM tant que `git push` ne l'a pas envoyé.

7. Vérifier la VM

Depuis le dépôt cloné :

```
$ bash verifier-installation.sh
```

Le résultat attendu se termine par :

```
RÉSULTAT : PRÊT POUR LE COURS
```

Prenez une capture de tout l'écran. Le bureau, le terminal et le résultat complet doivent rester visibles.

8. Placer la capture dans le dépôt

Avec l'application *Fichiers* :

- ① ouvrez le dossier affiché par `pwd` ;
- ② déplacez votre capture dans ce dossier ;
- ③ nommez-la `preuve-DA.png` en remplaçant `DA` ;
- ④ revenez au terminal et lancez `ls`.

Le nom de la capture doit maintenant apparaître à côté des fichiers de l'exercice.

9. Tester, enregistrer et envoyer

Exemple pour le DA 1234567 :

```
$ make test  
$ git status  
$ git add preuve-1234567.png  
$ git commit -m "Remettre exercice 01"  
$ git push
```

Si la capture est en format JPEG, utilisez son vrai nom dans `git add`.

10. Confirmer la remise dans GitLab



Rechargez la page de votre projet personnel.

- le dernier commit porte le message `Remettre exercice 01`;
- la capture apparaît dans la liste des fichiers;
- la pipeline se termine avec un crochet vert;
- le projet affiché porte votre DA.

La remise est reçue seulement après le `git push`. L'échéance est le 2 septembre à 23 h 59.

Pratique guidée

Parcours 1 : se déplacer sans deviner

Dans `~/cours/3a6/essai` :

- ① créez `depart.txt` ;
- ② copiez-le sous le nom `copie.txt` ;
- ③ renommez la copie `archive.txt` ;
- ④ supprimez `archive.txt` avec confirmation ;
- ⑤ revenez dans votre répertoire personnel.

Après chaque déplacement, utilisez `pwd`. Après chaque modification, utilisez `ls`.

Parcours 2 : remettre l'exercice 01

Suivez les étapes de la section GitLab dans l'ordre.

- ❶ la connexion SSH vous reconnaît ;
- ❷ votre propre dépôt est cloné dans la VM ;
- ❸ le script confirme l'installation ;
- ❹ `make test` accepte la capture ;
- ❺ le commit et la pipeline sont visibles dans GitLab.

Arrêtez-vous au premier résultat inattendu. Conservez le message affiché au lieu de recommencer l'installation.

Autoévaluation formative sur 5

Niveau	État du travail
0	Rien d'observable.
1	Un début existe, sans résultat sérieusement exploitable.
2	Le travail est très incomplet, incorrect ou ne compile pas.
3	La base est présente, mais un problème majeur demeure.
4	Le résultat fonctionne; il reste de petits problèmes.
5	Le résultat est complet, clair et presque sans amélioration nécessaire.

Client usager

La capture lisible se trouve dans le projet GitLab qui porte le bon DA, et la pipeline réussit.

Client programmeur

Les commandes ont été exécutées depuis le dépôt cloné; le commit contient seulement la preuve demandée.

Synthèse

Deux réflexes à conserver

Dans le terminal

`pwd` pour se situer,
`ls` pour observer,
puis une seule modification.

Dans un dépôt

`git status` pour observer,
un `commit` pour enregistrer,
`git push` pour remettre.

Une commande est terminée quand son résultat a été vérifié.

À la prochaine séance

Nous lirons un petit programme C en réutilisant les mêmes habitudes :

- retrouver le bon dossier ;
- modifier une petite partie ;
- compiler ;
- observer le résultat ;
- enregistrer le travail dans Git.