



TP : PIE et Git

1 Bienvenue à EPITA !

1.1 Objectifs du TP

- Découvrir votre environnement de travail
- Apprendre à utiliser des outils d'informaticien
- Rendre un TP avec Git

1.2 Nomenclature

Fondamentaux

Les paragraphes dans les blocs verts correspondent aux fondamentaux, vous devez impérativement les maîtriser d'ici le prochain TP.

Si au terme de ce TP vous avez encore des difficultés sur ces notions, vous devez impérativement les travailler de votre côté.

Exercices

Les paragraphes dans les blocs violets correspondent à des actions que vous devez effectuer pendant le TP.

Pour aller plus loin

Les paragraphes dans les blocs bleus correspondent à des notions plus avancées. Les maîtriser vous apportera un confort supplémentaire dans l'utilisation des machines de l'école.

Power user

Vous **devez** ignorer les paragraphes dans les blocs oranges si vous n'êtes pas parfaitement à l'aise avec les notions abordées pendant ce TP. Aucune aide sur ces parties ne sera apportée par les ACDC.



2 PIE

2.1 Présentation

Le **PIE**¹ est l'environnement de travail que vous serez amené·e·s à utiliser tout le long de votre scolarité à EPITA. Il vous permettra de réaliser les différents TP, projets et devoirs qui vous seront assignés; il vous donne accès aux outils et à la configuration nécessaire au bon fonctionnement des technologies et langages qui vous seront enseignés.





Les machines de l'école tournent sous la distribution² **NixOS**. Certaines machines sous Windows ou macOS sont également disponibles.

Le PIE est administré par le **CRI**³.

En utilisant les salles machines et le PIE, vous devez vous soumettre à différentes règles. Celles-ci sont énoncées dans la [charte des salles machines](#). Sa lecture est obligatoire.

2.2 Connexion

Il est temps de se connecter ! Pour se faire :

1. si nécessaire, allumer l'ordinateur et choisir l'image nommée « **NixOS SUP** » dans le menu de démarrage ;
2. choisir **i3** comme session ;
3. sélectionner la disposition de clavier correspondant à celui du poste (**fr** pour AZERTY et **us** pour QWERTY US) ;
4. entrer son login et mot de passe CRI. En cas de difficulté, appeler un ACDC ;
5. choisir la touche  (si c'est le premier login) :  ou . En cas de doute, choisir .

2.3 i3

Tout environnement de bureau fait appel à un *gestionnaire de fenêtres* ou (*window manager*) pour positionner les fenêtres sur l'écran. Sur le PIE, le gestionnaire de fenêtres disponible se nomme **i3**. Il a la particularité d'être un *tiling window manager*, c'est-à-dire qu'il positionne lui-même les fenêtres de façon à ce qu'elles ne se recouvrent pas.

L'intérêt d'un *tiling window manager* est de libérer l'utilisateur d'une fastidieuse gestion des fenêtres et donc de gagner en efficacité. C'est tout particulièrement adapté à la programmation, qui est une activité qui nécessite d'avoir du code source et de la documentation sous les yeux en permanence.

1. Parc informatique de l'EPITA
2. Dans l'univers Linux, une distribution est un ensemble cohérent de logiciels déployables facilement.
3. Centre des ressources informatiques



Cerise sur le gâteau, ce qui reste de gestion des fenêtres peut s'effectuer à l'aide de raccourcis clavier. Ci-dessous figure un petit aide mémoire des raccourcis usuels. La touche **Mod** correspond à **Alt** ou **Windows** en fonction du choix que vous avez fait lors du premier démarrage de votre session. Ce choix peut être modifié dans le fichier de configuration d'i3.

Fondamentaux

- Mod** + + **E** Quitter i3 et revenir à l'écran de connexion (après confirmation)
- Mod** + **entrée** Ouvrir un terminal
- Mod** + + **Q** Fermer une fenêtre
- Mod** + + Déplacer une fenêtre
- Mod** + **d** Lancer **dmenu**, un lanceur d'application

Exercice

Ouvrez plusieurs terminaux à l'aide de **Mod** + **entrée** et observez comment les fenêtres se répartissent la surface d'écran disponible.

Testez une à une les commandes exposées ci-dessus, les commandes qui s'appliquent à une fenêtre se font sur celle qui est active. Par défaut, il s'agit de la fenêtre en dessous de la souris. Cette fenêtre est encadrée en bleu clair, contrairement aux inactives encadrées en noir. **dmenu** s'affichera en haut de l'écran, vous pourrez alors taper « **firefox** » puis **entrée** pour lancer un navigateur.

Pour aller plus loin

i3 offre la possibilité de répartir les fenêtres sur des bureaux virtuels. Les bureaux virtuels sont numérotés de 1 à 10 par défaut. Vous pouvez vous déplacer sur le bureau 2, par exemple, en utilisant le raccourci **Mod** + **2**^{a b}.

Il est également possible de déplacer les fenêtres d'un bureau à l'autre. En ayant le focus sur une fenêtre du bureau 1 que vous voulez déplacer sur le bureau 2, utilisez le raccourci **Mod** + + **2**.

Les bureaux virtuels en cours d'utilisation s'affichent en bas à gauche de l'écran.

Documentation d'i3 <https://i3wm.org/docs/userguide.html>

Référence i3 <https://i3wm.org/docs/refcard.html>

- ^a. Il faut utiliser les touches numériques de la rangée du haut et non pas celles du pavé numérique.
- ^b. Le bureau numéroté 10 correspond à la touche du clavier

2.3.1 i3lock

Il peut arriver que vous ayez besoin de vous absenter un court instant de votre poste alors que vous êtes en train de travailler dessus. Si vous laissez votre session accessible à n'importe



qui, vous laissez accès à cette personne à tous vos comptes, vos informations personnelles et confidentielles. Vous vous mettez également à la merci d'une *confloose*, c'est-à-dire à une série de configurations spécifiquement pensées pour maximiser l'inconfort dans l'utilisation de votre machine. Plus grave, vous vous exposez aussi à endosser la responsabilité des potentiels actes illégaux commis depuis votre session.

Pour éviter d'avoir à vous déconnecter puis reconnecter chaque fois que vous quittez votre poste, vous pouvez utiliser un *screen locker*. Le *screen locker* disponible sur le PIE se nomme **i3lock**.

Une fois **i3lock** enclenché, vous devrez taper votre mot de passe suivi de pour déverrouiller votre session.

Fondamentaux

Lancez **i3lock** à l'aide de **dmenu**, puis déverrouillez votre session.

Attention

À EPITA, il est interdit de laisser une session verrouillée plus d'une heure.

2.4 AFS

Sur votre ordinateur personnel, les fichiers que vous manipulez sont stockés localement, sur votre disque dur. Ils ne sont donc accessibles que depuis celui-ci. Cette stratégie de gestion des données ne fonctionne pas lorsque vous devez travailler sur un parc informatique tel que celui de l'EPITA, puisqu'elle impliquerait de devoir travailler sur une unique machine pour retrouver ses documents.

Pour résoudre ce problème, le CRI administre un système de fichier distribué : l'AFS⁴. Il permet d'accéder aux fichiers stockés dessus depuis n'importe quelle machine de l'école.

Fondamentaux

Chaque étudiant·e dispose d'un répertoire personnel sur l'AFS, qui n'est accessible qu'à lui-elle seul·e, et dans lequel il est possible de mettre jusqu'à 2 Go de fichiers.

Votre répertoire AFS se situe dans votre répertoire utilisateur et se nomme **afs**.

Power user

Pour vérifier l'espace disque restant dans votre AFS, vous pouvez exécuter la commande **fs listquota -human** dans votre AFS.

4. Andrew File System



Attention

Tout fichier stocké en dehors de l'AFS sera perdu au redémarrage de la machine.



3 CRI

3.1 Présentation

Le **CRI** (représenté par une chouette bleue) est à la fois un **laboratoire d'administration système** et un **service** de l'école. Il s'occupe, entre autres choses :

- de l'équipement informatique des salles machines ;
- de l'environnement de travail (**PIE**) ;
- des comptes élèves CRI.

Le **CRI** ne s'occupe **pas** :

- du Wi-Fi de l'école ;
- du réseau filaire de l'école ;
- du site internet de l'école ;
- des boîtes mails des étudiant·e·s ;
- des licences mises à disposition des étudiant·e·s ;
- de la propreté des salles machines.

3.2 Intranet CRI

3.2.1 Y accéder

Exercice

1. Ouvrez **dmenu** (Mod+d) ;
2. tapez **firefox** dans **dmenu** ;
3. appuyez sur **Entrée** ;
4. rendez-vous sur <https://cri.epita.fr> ;
5. connectez-vous en utilisant le mot de passe que vous avez choisi lors des tests de connexion.

3.2.2 Pages utiles

L'**Intranet CRI** permet de faire les choses suivantes :

Profil consulter les informations relative à votre compte CRI et modifier les clés SSH qui y sont associées ;

Promos & Classes consulter la liste des étudiant·e·s EPITA ;

Documentation chercher des informations de contact et de la documentation de base.





3.3 Moodle

Le CRI héberge une instance **Moodle**. Ce LMS (*learning management system*) sera utilisé par vos enseignants pour vous mettre à disposition vos cours. Il pourra également servir si des examens sont organisés à distance.

Vous devriez normalement tous avoir accès au cours qui contiendra les TP des ACDC, vous pourrez d'ailleurs y retrouver ce sujet : <https://moodle.cri.epita.fr/course/view.php?id=578>

Fondamentaux

Pour vous connecter à Moodle, vous devez utiliser le bouton « CRI » de la page de connexion.

Exercice

Rendez-vous sur la page des préférences de votre compte Moodle et choisissez la langue avec laquelle vous êtes censé suivre les cours : français pour la plupart d'entre-vous, anglais pour les étudiants de la section anglophone.

3.4 Machine virtuelle

En prévision du travail à distance, le CRI met à votre disposition une machine virtuelle. Celle-ci vous permettra d'avoir accès à un environnement similaire à celui des postes en salles machines. Des instructions pour installer cette machine virtuelle sont disponibles sur la [documentation du CRI](#). Il existe également [une vidéo](#) qui présente son installation. La machine virtuelle n'est plus activement maintenue par le CRI car il est attendu des étudiants de venir travailler à l'école. Vous ne pouvez donc pas demander d'aide sur son utilisation. Si vous souhaitez disposer d'un environnement similaire à celui de l'école chez vous, il est fortement recommandé de le construire soi-même en installant une distribution Linux de son choix. De cette manière, vous en apprendrez beaucoup plus sur le système que vous utilisez au quotidien à l'école.

Attention

Vous ne pourrez pas vous connecter avec votre compte CRI sur l'écran de connexion, après le lancement de la machine virtuelle puisque cela impliquerait de rendre publiquement accessible la liste des utilisateurs du CRI sur Internet. Vous devrez donc vous y connecter avec une session « epita », sans mot de passe.

4 Enseignement à distance

4.1 Teams

Teams est un logiciel fourni de base avec Office 365 et il vous est donc accessible. Il permet de faire des visioconférences, du partage d'écran, des appels audios et de converser par écrit.





Certains cours l'utiliseront, des « équipes » Teams seront alors créées pour votre classe.

Ce service n'est pas géré par le CRI, il faut donc vous y connecter avec vos identifiants Bocal (ceux de votre boîte mail).

Il est possible d'installer Teams sur votre machine ou d'y accéder via son [interface web](#).

Exercice

Lancez **teams** à l'aide de **dmenu**, connectez-vous y et familiarisez-vous avec son interface.

4.2 Discord

Les ACDC et la majorité des enseignants du cycle préparatoire avaient adopté Discord lors des confinements il y a deux ans. Il est possible que cet outil soit de nouveau utilisé cette année.

Exercice

Lancez **discord** à l'aide de **dmenu** et créez vous un compte Discord si vous n'en avez pas déjà un.

Les ACDC vous expliqueront les modalités d'utilisation de cette plateforme, si elle doit être utilisée pour vos TP (le serveur que vous devrez rejoindre, les règles à respecter, etc.).



5 Terminal

5.1 Introduction

Lorsque vous utilisez un ordinateur, vous faites appel à des programmes qui font des actions pour vous : se déplacer, créer/déplacer/modifier/supprimer des fichiers, afficher une page web, ...

Chacun de ces programmes donne une série d'instructions à exécuter au système, par exemple :

1. Afficher une fenêtre ;
2. vous demander quoi faire ;
3. exécuter des algorithmes ;
4. afficher le résultat dans la même fenêtre.

La programmation d'une interface utilisateur complète, basée sur des composants graphiques (boutons, champs de texte, label, etc) demande une quantité conséquente de travail : il faut décider où les placer, programmer les interactions, ...

Il existe heureusement une alternative aux programmes graphiques : les programmes en **ligne de commande**. Se résumant au strict minimum (texte affiché à l'écran), ils sont bien plus rapide à développer.

L'interface qui sera présentée à l'utilisateur ou l'utilisatrice pour les saisies et l'affichage prendra alors la forme d'une fenêtre sur fond noir : un terminal.

Power user

L'émulateur de terminal utilisé par défaut sur le PIE est `rxvt-unicode`. Le plus classique `xterm` est également disponible.

5.2 Le shell, un programme à votre service

Quand vous démarrez un terminal à l'aide de **Mod+Enter**, un programme, appelé *shell* a pour rôle d'attendre vos instructions et d'en afficher le résultat. Il affiche un message vous invitant à taper des commandes :

```
[prénom.nom@r01p01 ~]$
```

Dans les exemples de la suite du TP, ce texte sera remplacé par `$`. Les lignes qui ne commencent pas par ce caractère correspondent à la sortie de la commande, c'est-à-dire au texte affiché dans le terminal.

Cette ligne signifie que le shell est prêt à exécuter vos instructions. Chacune d'entre elle sera en fait le nom d'un programme à démarrer, suivi de détails permettant au programme de savoir quoi faire : les arguments.

L'instruction ainsi obtenue porte le nom de « commande ».

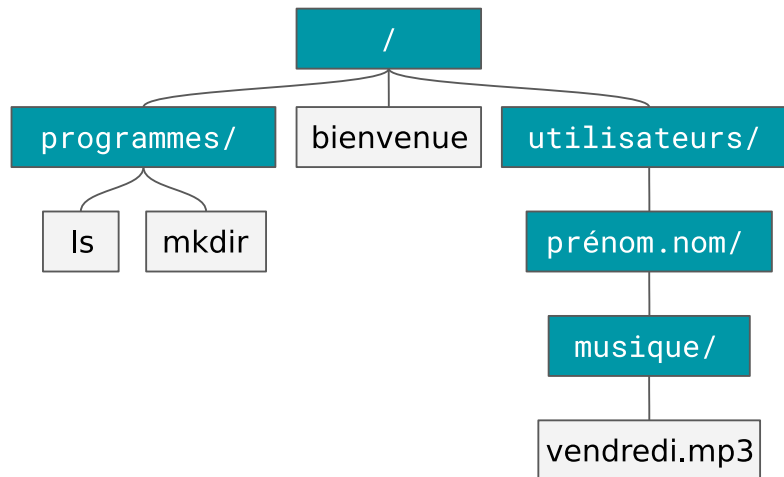
Lors de son exécution, le programme lancé pourra afficher du texte. Lorsqu'il se termine, le shell, qui l'a démarré, vous demande à nouveau d'entrer une commande.





5.3 Le système de fichiers

Sur un ordinateur, les fichiers sont organisés sous la forme d'une arborescence. Chaque fichier est désignable par le chemin qui permet de le retrouver à partir de la racine. Chaque croisement de l'arborescence est un dossier. Sous Windows, on note cette racine `C:\`, sous Linux, on la note `/`.



Pour aller plus loin

On ajoute parfois une barre oblique à la fin du nom des dossiers, pour les différencier des simples fichiers, mais ce n'est pas une obligation.

Dans cet exemple, le chemin de `ls` est `/programmes/ls`⁵. De même, le chemin de `vendredi.mp3` est `/utilisateurs/prénom.nom/musique/vendredi.mp3`⁶

Fondamentaux

On peut aussi décrire le chemin vers un dossier ou fichier à partir de l'endroit où l'on se trouve, on dit alors qu'on utilise des chemins relatifs, par opposition aux chemins absolus qui partent de la racine.

Comme leur nom l'indique, les chemins relatifs sont relatifs à un dossier particulier de l'arborescence. En fonction de l'endroit où l'on se trouve, ils peuvent ne pas désigner le même fichier !

Dans le cas où on veut désigner le dossier parent, c'est-à-dire le dossier immédiatement au dessus dans l'arborescence, on utilise « `../` ».

5. Sur le PIE, on utilisera plutôt `/bin/ls`

6. Sur le PIE, on utilisera plutôt `/home/prénom.nom/...`



Par exemple, avec le schéma ci-dessus, on désignera depuis le dossier `prénom.nom` le fichier `mkdir` par le chemin `../..programmes/mkdir`. Aussi, depuis le dossier `programmes`, on désignera la racine, « / », par « `../` ».

Lister le contenu de « `..` » depuis le dossier « `/utilisateurs/` » montre les dossiers « `utilisateurs/` » et « `programmes/` » ainsi que le fichier « `bienvenue` ».

5.4 Se déplacer dans le système de fichiers

Première étape, s'orienter : la commande `pwd`⁷ permet d'afficher le nom du dossier dans lequel vous vous trouvez actuellement.

```
$ pwd
/home/prénom.nom
```

Exercice

Entrez la commande `pwd` et observez le résultat.

En fait, `pwd` affiche le chemin absolu du dossier actuel, c'est-à-dire le chemin complet depuis la racine. Pour rappel, la racine est le dossier de plus haut niveau, qui contient tous les autres.

Si un chemin ne commence pas par un /, c'est un chemin qui débute depuis le dossier actuel, un chemin relatif donc.

Fondamentaux

La prochaine étape va être de se déplacer dans l'arborescence : nous allons changer le dossier courant. La commande qui permet d'effectuer cette action se nomme `cd`^a.

La commande `cd` requiert un argument, le chemin (relatif ou absolu) du dossier dans lequel se déplacer.

La syntaxe générale pour passer un argument à un programme est la suivante :

```
nom-du-programme argument1 argument2 argument3
```

a. Pour `change directory`

Pour aller plus loin

La commande `cd` peut également être utilisée sans arguments pour revenir dans votre répertoire utilisateur.

7. Pour *print working directory*



5.4.1 Quelques exemples

```
$ cd /  
$ pwd  
/  
$ cd /home  
$ pwd  
/home
```

Exercice

Entrez la commande `cd /home`, puis la commande `pwd`, observez le résultat puis retournez dans votre dossier personnel.

Home sweet home

Le raccourci `~` vous permet de faire référence à votre dossier personnel sans avoir à taper `/home/prénom.nom`.

La commande `ls` permet de lister le contenu d'un dossier.

Par défaut, `ls` liste le contenu du dossier courant. La commande `ls` peut également être utilisée avec des arguments pour lister le contenu d'autres dossiers.

```
$ ls /  
bin boot dev etc home lib lib64 ...  
$ ls /home  
prénom.nom
```

Ces commandes offrent des fonctionnalités proches d'explorateurs de fichiers graphiques.

Les commandes comme `ls` peuvent être utilisées avec des options (qui commencent souvent par `-`). Par exemple, l'option `-a` permet d'afficher les fichiers cachés (sous Linux, un fichier caché est un fichier commençant par un « `.` »).

```
$ cd  
$ ls  
afs  
$ ls -a  
. .. afs .bash_history .bashrc
```

Exercice

Entrez la commande `ls -a` et observez le résultat. Que contient le dossier `.config`, dans votre espace personnel ?

Entrez la commande `ls -l` et observez le résultat.





5.4.2 Créer et modifier des ressources

La commande `mkdir`⁸ permet de créer un dossier.

```
$ cd
$ mkdir dossier_test
$ ls
afs dossier_test
```

Exercice

Créez un dossier nommé `dossier_test` dans votre espace personnel.

Vous pouvez utiliser la commande `gedit` pour lancer un éditeur de texte graphique. Vous pouvez donner en argument le chemin du fichier à éditer.

```
$ gedit mon_fichier
* un éditeur apparaît *
```

La commande `cat` permet d'afficher le contenu d'un fichier.

Exercice

Créez un fichier puis affichez son contenu dans un terminal.

```
$ cat mon_fichier
du texte tapé dans gedit !
```

Beaucoup d'autres commandes sont disponibles pour manipuler les fichiers. Par exemple `mv` permet de déplacer des fichiers ainsi que des dossiers⁹.

Le premier argument de `mv` est le chemin actuel du fichier ou dossier à déplacer, et le second argument de `mv` est le nouveau chemin du fichier, **ou** le chemin du dossier dans lequel le déplacer.

```
$ mv mon_fichier autre_fichier
* Renomme mon_fichier en autre_fichier *
$ mv autre_fichier dossier_test/encore_un_autre
* Renomme autre_fichier en encore_un_autre, et le met dans le dossier dossier_test *
$ mv dossier_test/encore_un_autre ./
* Bouge sans renommer encore_un_autre dans le dossier actuel *
$ ls
afs encore_un_autre dossier_test
```

8. `make directory`

9. `move` en anglais



Exercice

La commande pour renommer un fichier est la même que pour déplacer un fichier, renommer revient effectivement à déplacer un fichier dans le même dossier avec un nom différent.

Renommez un de vos fichiers, puis déplacez-le.

Attention

Si vous déplacez un fichier en indiquant le chemin d'un autre fichier qui existe déjà, le fichier de destination sera remplacé par le fichier source sans demander de confirmation !

La commande `cp`¹⁰ permet de dupliquer un fichier.

```
$ cat encore_un_autre
du texte tapé dans gedit !
$ cp encore_un_autre heyhey
$ ls
afs encore_un_autre heyhey dossier_test
$ cat heyhey
du texte tapé dans gedit !
```

Exercice

Copiez un de vos fichiers.

La commande `rm` permet de supprimer¹¹ un fichier.

Attention

Soyez prudent·e·s, ou vous risquez de perdre des fichiers importants : contrairement à Windows, il n'y a pas de corbeille où récupérer les fichiers supprimés.

```
$ ls
afs encore_un_autre heyhey dossier_test
$ rm heyhey
$ ls
afs encore_un_autre dossier_test
```

Exercice

Copiez un fichier, affichez son contenu puis effacez le.

10. `copy`, en anglais
11. `remove`, en anglais



5.5 Manuel

Vous pouvez afficher l'aide associée à une commande en utilisant la commande `man`, suivi du nom de la commande pour laquelle vous voulez afficher la page de manuel.

```
$ man ls
```




Exercice

Affichez la page de manuel d'une commande de votre choix.

Pour quitter le manuel, appuyez sur la touche .

Pour faire défiler la page, utilisez les flèches directionnelles.



5.6 Pense-Bête

commande	fonction	exemple
<code>pwd</code>	afficher le chemin courant	<code>pwd</code>
<code>cd</code>	changer de dossier	<code>cd dossier_test</code>
<code>ls</code>	lister le contenu d'un dossier	<code>ls</code>
<code>mkdir</code>	créer un dossier	<code>mkdir dossier_test</code>
<code>cat</code>	afficher le contenu d'un fichier	<code>cat test1.txt</code>
<code>mv</code>	déplacer ou renommer un fichier	<code>mv resultplusun.txt ../testmv.txt</code>
<code>cp</code>	copier un fichier	<code>cp ../testmv.txt resplusun.txt</code>
<code>rm</code>	effacer un fichier	<code>rm ../testmv.txt</code>
	auto-complétion	<code>cat resplus</code> 
	parcourir l'historique	

5.7 Pour aller plus loin

Si le temps le permet, n'hésitez pas à explorer d'autres commandes.

Vous pouvez par exemple utiliser :

- `date` pour afficher l'heure au format de votre choix ;
- `echo` pour afficher du texte ;
- `touch` pour créer un fichier vide ;
- `grep` pour rechercher dans du texte ;
- l'opérateur `>` pour mettre ce qu'affiche le programme lancé à gauche de l'opérateur, dans un fichier dont on donne le nom à droite de l'opérateur, par exemple : `ls > liste.txt`);
- l'opérateur `|` pour chainer des commandes en passant à l'une le texte affiché par l'autre. (ex : `ls | wc`);
- le raccourci  +  pour rechercher des commandes écrites par le passé ;
- la gestion des permissions, avec `chmod` et `chown`.



6 Logiciels utiles

Votre session vous donne accès à un certain nombre de logiciels. Pour y parvenir, lancez `dmenu` et entrez le nom du logiciel que vous voulez lancer. Vous pouvez aussi le faire depuis un terminal.

Fondamentaux

Voici quelques logiciels qui vous seront nécessaires :

firefox votre navigateur préféré ;

evince un lecteur pdf, pour lire les sujets par exemple ;

i3lock si vous quittez votre poste, pensez à le verrouiller. Vous aurez besoin de votre mot de passe de session pour le déverrouiller.

Pour aller plus loin

D'autres outils pourraient vous sembler utiles, comme :

pavucontrol un outil de réglage du volume sonore ;

redshift définit la température de la couleur de votre écran, permet d'atténuer les composantes bleues le soir ;

thunderbird client email, qui peut aussi être utilisé comme client news.

Utilitaires en ligne de commande :

setxkbmap Pour changer votre disposition clavier ;

poweroff pour éteindre votre machine ;

zip crée une archive zip ;

unzip ouvre une archive zip ;

tar manipule des archives tar ;

man affiche de la documentation ;

tree affiche l'arborescence des fichiers depuis un dossier.

Power user

Les programmes suivants, utilisables en ligne de commande, ne sont destinés qu'aux plus braves d'entre vous :

weechat client de chat IRC. Vous pouvez rejoindre **#epita** sur irc.rezosup.org ;

grep recherche du texte ;

sed remplace du texte, et bien plus ;

i3 vous pouvez configurer **i3** pour en faire ce que vous voulez ; configurer des raccourcis clavier, la couleur et police d'écriture de l'interface, et bien plus ;

feh affiche des images, et peut remplacer votre fond d'écran ; Regardez du côté des options **-no-fehbg** et **-bg-center**. Vous pouvez faire en sorte qu'**i3** lance **feh**.





7 Editeurs

Il y a de nombreux éditeurs de texte disponibles dans le terminal. Cependant, deux sont utilisés couramment à EPITA : **vim** et **emacs**. Nous vous conseillons d'en choisir un, gardez juste en tête que vous utiliserez principalement **emacs** pendant votre premier semestre.

7.1 Emacs

Emacs est un **éditeur de texte**. Comme tout programme, vous pouvez l'ouvrir en utilisant **dmenu** ou directement dans votre terminal. Les commandes pour naviguer dans **emacs** suivent une notation spécifique, décrite ici :

- un **C** majuscule représente la touche **Control** ;
- un **M** majuscule représente la touche **Meta** (**Alt** par défaut) ;
- la notation '**C-x**' signifie « En maintenant **Control**, appuyer sur '**x**' » ;
- la notation '**C-x b**' signifie « En maintenant **Control**, appuyer sur '**x**', puis relâcher ces touches, et appuyer sur '**b**' ».

7.1.1 Raccourcis

Voici une liste de raccourcis clavier utiles :

- C-x C-f** ouvrir un fichier. Une fois le raccourci entré, Emacs va demander un **nom de fichier** ; tapez un nom de fichier existant pour l'ouvrir, ou tapez le nom d'un nouveau fichier pour le créer avant de l'ouvrir ;
- C-x b** changer de buffer. Un **buffer** dans Emacs est une zone dans laquelle vous pouvez éditer du texte. Vous pouvez avoir plusieurs buffers ouverts dans Emacs, au premier plan ou en arrière plan. Quand vous tapez '**C-x b**', Emacs vous demande un nom de buffer : tapez en un (souvent le nom du fichier associé au buffer que vous voulez afficher), ou appuyez sur **Enter** pour afficher le dernier buffer visité ;
- C-x s** sauvegarder les changements faits à un fichier ;
- C-x 0** fermer le buffer actuel ;
- C-x 1** fermer tous les autres buffers ;
- C-x 2** diviser le buffer actuel en deux, verticalement ;
- C-x 3** diviser le buffer actuel en deux, horizontalement ;
- C-x o** alterner entre les différents buffers affichés ;
- M-w** copier la sélection ;
- C-w** couper la sélection ;
- C-y** coller ;
- C-x C-c** fermer Emacs, en demandant de sauvegarder les fichiers auparavant.

Avec ces raccourcis vous devriez être capable d'éditer des fichiers pendant les prochains TPs. Il est fortement conseillé d'apprendre à utiliser Emacs efficacement, puisqu'il sera votre éditeur pendant tous les TPs et examens d'OCaml.





7.2 Vim

Vim est un éditeur de texte en ligne de commande très populaire parmi les étudiant·e·s de l'école. Nous ne verrons pas comment il fonctionne pendant ce TP, uniquement comment en sortir.

Exercice

Lancez la commande `vim` dans un terminal. Tapez la séquence de caractères `:` `q` puis validez avec `entrée`.

Pour aller plus loin

Vimtutor Tapez simplement `vimtutor` dans votre shell ;

Documentation officielle <https://www.vim.org/docs.php>



8 Git

8.1 Introduction

Git est un programme qui vous aide à travailler sur des projets de groupe en gardant un historique des modifications et des différentes versions de votre projet. Son utilisation vous permet de ne pas perdre votre travail lorsque vous travaillez à plusieurs ou depuis plusieurs ordinateurs.

Lorsque vous travaillez sur un projet avec **Git**, dès que le travail que vous venez d'accomplir mérite d'être sauvegardé (si vous venez d'intégrer une nouvelle fonctionnalité, vous avez corrigé un bug, ...), vous pouvez utiliser **Git** pour créer un **commit**. Un commit est une image de votre projet à un instant précis, une version de votre projet. Il contient un « snapshot » de votre dossier de travail et un lien vers le commit précédent (s'il existe). Notez que les fichiers inchangés ne sont pas dupliqués, ce qui permet de prendre moins de place sur votre disque dur. Vous aurez un historique de votre travail, et si vous souhaitez revenir à une ancienne version de votre projet, vous pourrez revenir à un commit plus ancien en utilisant la commande **checkout** de **Git** pour revenir à l'état de votre projet à cette étape.

Un projet utilisant **Git** aura, la plupart du temps, un **serveur Git**, qui représentera l'état actuel du projet, contrairement aux utilisateurs et utilisatrices qui ont des versions de développement du projet.

8.2 Git mantra

Pour utiliser **Git**, lorsque vous travaillerez sur votre projet, vous suivrez ces étapes :

1. vous clonerez le **repo Git** (votre projet) du serveur sur votre ordinateur. Tout le travail déjà présent sera téléchargé sur votre ordinateur, vous aurez accès à l'état actuel du projet, mais aussi à tous les commits précédemment effectués ;
2. vous ajouterez une nouvelle fonctionnalité ;
3. vous ajouterez (commande **add**) les fichiers modifiés ou créés pour préciser à **Git** que vous souhaitez les inclure dans votre prochain commit ;
4. vous effectuerez un **commit** avec votre ajout de fonctionnalité ou modifications. Vous expliquerez ensuite brièvement ce que vous avez fait dans un **message de commit** ;
5. vous allez ensuite effectuer un **pull** de votre projet depuis le serveur, pour vérifier si quelqu'un a modifié le projet. Si c'est le cas, **git** essaiera d'effectuer un **merge** (une fusion) des modifications automatiquement. S'il ne peut pas (si vous avez effectué des modifications sur la même partie d'un fichier), vous devrez faire le merge à la main, vous devrez choisir les modifications qui resteront dans le projet : le travail que vous venez de pull ? Le travail que vous avez effectué ? Ou un mélange des deux ;
6. vous ferez un **push** de votre travail sur le serveur. Après ça, toute personne qui fera un clone ou un pull du projet récupérera les changements que vous aurez effectués (vos commits).

Ces commandes vous seront très utiles pour débiter. Plus tard dans votre cursus vous aurez l'occasion d'en découvrir d'autres, **git** étant très riche en fonctionnalités.





8.3 Configurer git

8.3.1 Nom et email

Afin que le travail que vous produisez soit rattaché à votre nom, vous devez configurer Git pour rajouter cette information dans vos commits.

Exercice

Rajoutez dans le fichier `/home/prénom.nom/.gitconfig` :

```
[user]
name = Prénom Nom
email = prénom.nom@epita.fr
[color]
ui = true
[push]
default = simple
```

En remplaçant, bien sûr, le nom, prénom, et l'adresse mail par les vôtres.

8.3.2 Connexion au serveur

Pour pouvoir envoyer votre travail au serveur, il faut que celui-ci sache qui vous êtes. Cet élément de configuration se découpe en deux parties :

- configurer le client pour qu'il se présente correctement ;
- configurer le serveur pour reconnaître le client.

Pour y parvenir, on utilise un système de chiffrement asymétrique : le client, `git`, se présente avec une clé publique, avec lequel le serveur emballe ses réponses dans une sorte de coffre que seul le client peut ouvrir à l'aide de sa clé privée.

Pour que le serveur puisse échanger avec le client, il doit donc avoir votre clé publique, et votre client `git` doit être prêt à utiliser sa clé privée.

Créer une paire de clés Exercice

Rendez-vous sur <https://cri.epita.fr/>, identifiez-vous, cliquez sur votre login (en haut, à droite), et enfin sur « SSH keys ».

Cliquez sur « Help » et suivez les instructions.

Vous pouvez saisir un mot de passe pour votre clé. Il vous sera demandé à chaque utilisation de celle-ci.



Attention

Si la clé publique peut être librement partagée (`id_rsa.pub`), la clé privée (`id_rsa`) doit être jalousement conservée.

Donner votre clé publique au serveur Exercice

Retournez sur la page « SSH keys » et faites `Ctrl` + `V` dans le champ « Key », puis cliquez sur « Add SSH key ».

8.4 Commands

8.4.1 Clone

Pour récupérer un projet depuis un serveur, vous devez **clone** le projet. Pour ce faire, utilisez cette commande :

```
$ git clone prénom.nom@git.cri.epita.fr:p/2026-tp-pie/pie-prénom.nom
```

8.4.2 Status

Cette commande vous permet de voir l'état actuel de votre repo. Vous verrez tous les nouveaux fichiers, les fichiers modifiés et même les nouveaux fichiers qui sont prêts à être commit (cf. `$ git add`).

```
$ git status
```

8.4.3 Add

Utilisez cette commande pour rajouter des fichiers à la liste des fichiers que vous souhaitez suivre. Ces fichiers feront partie de votre prochain commit en utilisant la commande `$ git commit`.

```
$ git add settings.yml           # Ajoute les modifications du fichier settings.yml
$ git add global/settings.yml    # Ajoute le fichier settings.yml du répertoire global
$ git add .                      # Ajouter l'ensemble des fichiers du répertoire courant
```

ATTENTION

Rappelez vous que vous ne devez **pas** ajouter de fichiers exécutables, secrets, de logs ou temporaires. Vérifiez avec `$ git status` avant chaque commit.





8.4.4 Commit

Crée un **commit** avec les fichiers précédemment ajoutés.

```
$ git commit -m "Message" # Commit avec un message court  
$ git commit # Ouvre votre EDITEUR pour écrire un message de commit
```

Verbose

Utilisez -v pour montrer les changements que vous venez d'effectuer dans votre commit.

8.4.5 Pull

Télécharge les derniers commits depuis le serveur sur votre repo local.

```
$ git pull
```

8.4.6 Push

Envoie vos commits de votre ordinateur sur le **repo à distance**.

```
$ git push
```

8.5 Pour aller plus loin

Power user

Vous pouvez apprendre des notions importantes comme :

- le fichier `.gitignore` ;
- ce que sont les branches ;
- les workflows à plusieurs utilisateurs.



9 Liens Utiles

i3

- [Guide de l'utilisateur](#)

Git

- [TP de configuration Git](#)
- [Git book](#)
- [Tutoriel interactif sur l'utilisation des branches](#)

Divers

- [EPITA documentation](#)

Bash

- [Awesome Bash](#)

There is nothing more deceptive than an obvious fact