Petit précis des commandes sur raspberry

Tips 1 :

* Pour avoir des infos sur commandes, faire « commande » --help (ex : cp –help)
* Avant de faire une commande, toujours voir son help, cela peut être utile pour son utilisation. Par exemple pour « rm » il y a des risques de formatage de la carte si la commande est mal rentrée.

Commandes pour manipuler des dossiers/fichiers sur raspberry :

* sudo : permet d’exécuter une commande en tant qu’administrateur sans avoir les droits
* nano : permet d’ouvrir un fichier en écriture. Si la modification est impossible, vous avez 2 options
  + vous devez être en administrateur pour ouvrir et écrire dans le fichier. Dans ce cas, faites : sudo nano monfichier
  + si cela ne marche pas, faites « ls –al » et regardez que le fichier peut être modifié (c’est-à-dire lecture et écriture possible). Cela se présentera sous la forme –wrx--… où w = writable, r = readable, x = executable .

Pour modifier ces droits, se référer à la commande « chmod » plus bas.

* cd : permet de se placer dans un dossier (ou de revenir à la base de tous les dossiers)
  + cd Documents => on se place dans le dossier Documents
  + Dans le cas où vous voulez revenir au tout début de l’arborescence, tapez juste cd
  + Si vous voulez revenir au dossier précédent, tapez : cd ..
* ls : permet d’avoir la liste des fichiers/ dossier dans le dossier dans lequel vous êtes
  + si vous faites « ls –al », vous aurez un affichage en liste avec les droits disponibles sur les fichiers (écriture, lecture, etc..) ainsi que les dates de dernières modifications
* cp : copie un fichier/dossier
* mv : déplace un fichier/dossier
  + Cette commande permet également de renommer un fichier/dossier, pour ce faire il suffit de faire : mv fichier1 fichier2

Fichier1 étant l’ancien nom et fichier2 le nouveau nom (il faut bien entendu que fichier2 n’existe pas déjà)

* rm : supprime définitivement un dossier/fichier
  + se placer dans le dossier avant celui qu’on veut supprimer
  + !!!! Attention, cette commande est très « puissante » veillez à avoir une sauvegarde du fichier (sur github par exemple) afin de le récupérer si jamais vous l’avez supprimé par erreur !!!! (sinon vous allez devoir prier que le fichier est toujours quelque part sur la raspberry et le chercher en faisant une image de cette dernière, les backups sont votre unique moyen d’éviter cela)
  + Pour éviter cela vous pouvez utiliser la commande : mv « nom du fichier/dossier » $HOME/.Trash
  + Vous pouvez également faire : echo « alias rm= ‘rm –i’ » >> $HOME/.bashrc && source $HOME/.bashrc
* chmod : permet de s’octroyer des nouveaux droits sur les fichiers (le mettre en écriture, lecture, en tant qu’exécutable) ou en retirer
  + exemple : chmod 705 « nom du fichier » permet de mettre le fichier en écriture
* sudo nano /etc/network/interfaces : permet de fixer l’adresse IP d’une raspberry
  + Pour fixer une adresse en Ethernet :

auto eth0

iface eth0 inet static

adress « l’adresses IP que vous voulez mettre »

netmask « masque correspondant à la classe de l’IP »

gateway 10.1.1.1

* + Pour fixer une adresse en wifi (wlan0) :

allow-hotplug wlan0

iface wlan0 inet static

adress « l’adresse IP que vous voulez metre »

netmask « masque correspondant à la classe de l’IP »

gateway 10.1.1.1

wpa-conf /etc/wpa\_supplicant/wpa\_supplicant.conf

* + Une fois cela fait, tapez les 3 lignes de commandes suivantes :

sudo systemctl disable dhcpcd

sudo systemctl enable networking

sudo reboot

* crontab -e : permet de configurer des exécutions de programmes à des moments spécifiques. (utiliser l’option 2 lorsque c’est demandé car c’est la plus simple à configurer)
* ps aux | grep node : cette commande permet de voir les processus encore en fonctionnement. Cela peut être intéressant si vous avez une erreur du type « events.js : … ». Cette erreur peut dans certains cas être due à un mauvais arrêt d’un programme
  + Si jamais vous aviez votre processus déjà en fonctionnement, en exécutant cette commande, vous allez obtenir une liste des processus en cours

Exemple :

root 1101 0.0 4.4 96216 42444 ? Sl 16:17 0:01 node « direction d’un fichier »

pi 1558 0.0 0.0 4372 568 pts/0 S+ 16:53 0:00 grep –color=auto node

* La 2eme colonne correspond à l’identifiant <pid> du processus. Pour l’arrêter, vous devez faire : sudo kill <pid>. !!! Attention, cette commande peut s’avérer dangereuse, donc renseignez-vous avant de l’exécuter !!!

Github :

!!!! Il est fortement conseillé de lire ça avant de vous lancer dans l’utilisation de Github !!!!

Cette partie sera dédiée aux commandes de base que j’ai dû utiliser pour faire des backups de nos différents codes sur Github.

Mais qu’est-ce que Github ?

Github est un site web de versionning. Plus concrètement, c’est un cloud à destination des développeurs (amateurs et pros) leur permettant de sauvegarder leur fichier/dossier codé dans n’importe quel langage informatique. Vous pouvez donc versionner vos codes dessus, c’est à dire, faire une sauvegarde, puis continuer à travailler sur cette sauvegarde en la téléchargeant, sans modifier celle sur Github. Pour ce faire, le logiciel utilise un système de « branche » :

* La branche master, ou branche principale. C’est l’endroit où on retrouvera tous vos codes finaux
* Les branches secondaires. Celles-ci correspondent à des versions modifiées de vos codes. Par exemple, vous avez mis sur Github un code de jeu de dé (il sera sur la branche principale). Maintenant vous voulez y intégrer des règles de jeu (par exemple le 421), vous créez une branche secondaire et faites vos modifications dessus. Cela n’impactera pas la principale. Et quand vous jugerez bon de tout fusionner, il vous suffira de « merge » (on peut voir ça comme ramener) la branche secondaire sur la branche principale.

Avant de commencer à tenter de mettre quoique ce soit sur Github, créez-vous un compte sur le site pour pouvoir y déposer vos repository (repo).

Créez donc d’abord un dossier vide qui sera votre repo. Vous mettrez dedans tous les fichiers/dossiers que vous voulez push (on reviendra sur cette commande plus tard)

Commandes Github :

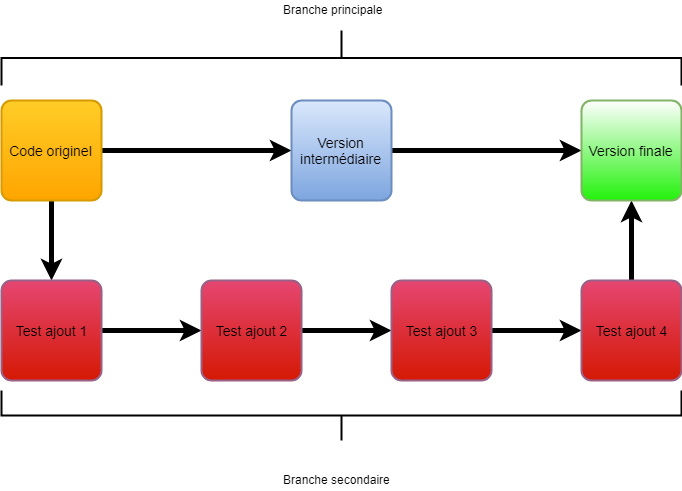
* git init : initialise la création du repo local dans lequel vous allez travailler
  + Si jamais vous avez déjà un repo sur github, vous pouvez faire :

git clone « l’url de votre repo github ». Cela le téléchargera sur votre raspberry. (petit conseil, clonez-le toujours sur votre /home/pi).

* git status : cette commande vous permet de savoir si vous avez des modifications qui ont été apportées au repo.
  + Un fichier/dossier en rouge signifie qu’il a été modifié (ajout, suppression, etc…)
  + Un fichier/dossier en vert signifie qu’il a été intégré au repo
* git add : cela permet d’ajouter toutes vos modifications au repo. Une fois cette commande effectuée, vous pouvez faire un git status. Si tout s’est bien passé, vos modifications devraient s’afficher en vert, elles sont donc prêtes à être envoyé sur le git.
  + Cette commande peut s’utiliser de 2 manières :
    - Soit en faisant : git add « nom du fichier/dossier ». Cela ajoutera uniquement le fichier/dossier spécifié. Personnellement je ne l’ai jamais utilisée car …
    - Il existe une commande bien plus simple : git add . (notez bien le point après le add, il est important). En faisant cela, vous ajouterez toutes vos modifications à votre repo local.
* git commit -m « message » : permet d’écrire un descriptif des fichiers que vous allez mettre sur votre git (par exmple : 1er push) !!!! attention, ici les guillemets sont à mettre !!!! Une fois cela fait, vous êtes prêt à publier.
* git push origin master : en entrant cette commande, vous allez envoyer votre repo local sur votre github
  + vous pouvez également utiliser : git push -u

le « -u » signifie upstream, c’est-à-dire votre repo.

* + Une fois la commande effectuée, actualisez votre page github et vérifiez que le push s’est effectué correctement.
* git remote -v : affiche tous les remote configurés. Un remote étant un repo, mais celui sur votre github. Il faut que le nom de votre repo local et de votre repo sur le git (le remote) aient le même nom sinon vous ne pourrez rien faire. (d’où l’intérêt de toujours cloner votre repo quand vous voulez y ajouter des choses)
* git remote show « nom du remote » : vous donne des infos sur un remote donné. Cela peut être utile lorsque vous avez push quelque chose mais que l’actualisation ne s’est pas effectuée.
  + Par exemple, j’ai push 2 dossiers sur notre git mais ceux-ci ne s’affichaient pas. Cela bloquait donc tout autre tentative de push. La commande git remote show permet donc de voir quel est le statut de votre remote.
* git remote add « nom du repo » « url du repo » : cela vous permet de reconfigurer le remote (repo de destination sur git). Cette commande peut être nécessaire dans le cas où la raspberry « perd de vue » le remote.

Pour ce qui est du système de « branch » et de « merge », je ne m’y suis pas penché donc ce sera à vous de vous documenter pour gérer cela. Je vous conseille de le faire afin de pouvoir faire différentes versions de vos codes tout en gardant la version originelle. Schématiquement ça donnerait ça :