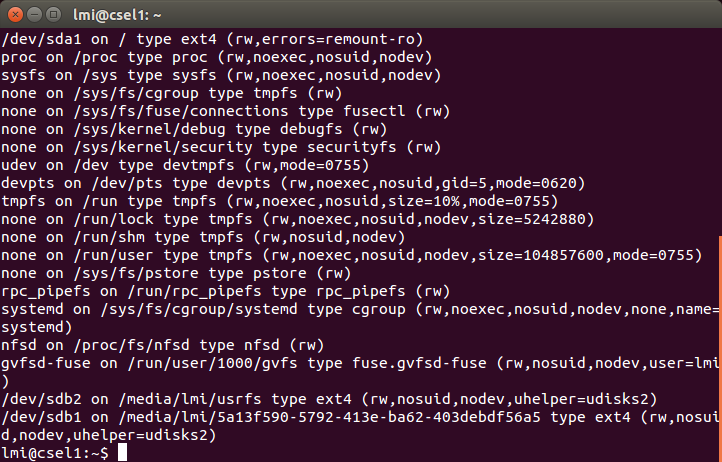
# Environnement Linux Embarqué

## Gravure de la carte SD :

Pour graver la carte SD, il faut trouver le nom du périphérique de la carte. Il peut être obtenu avec la commande mount :

$mount

La commande va lister tous les périphériques connectés, dans notre cas, la carte SD se nomme lmi et est liée à /dev/sdb2 et /dev/sdb1.

Un script a ensuite été écrit, regroupant les différentes commandes à exécuter pour la gravure de la carte.

Code du script gravure

Une fois la carte gravée et insérée dans la cible, le ventilateur se met à tourner si tout s’est bien passé. Si rien ne se passe, il faut également contrôler que le switch de l’Odroid est en position pour booter sur la carte SD.

N.B. exécution d’un script :

* Aller dans le répertoire du script. Puis :
  + ./nom\_du\_script.sh
* Si le script n’est pas placé à la racine du répertoire sur lequel il faut travailler, il faut changer ses droits avec la commande chmod+x

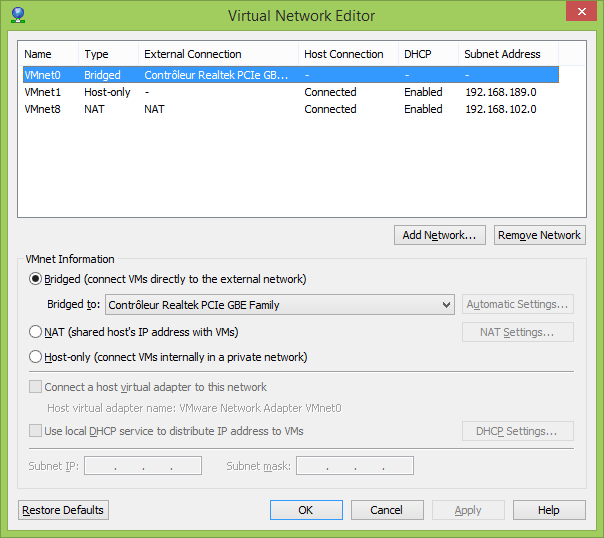
## Mise en place de l’infrastructure

La cible ODROID-XU3 doit avoir l’adresse IP 192.168.0.11 et la machine hôte l’IP 192.168.0.4.

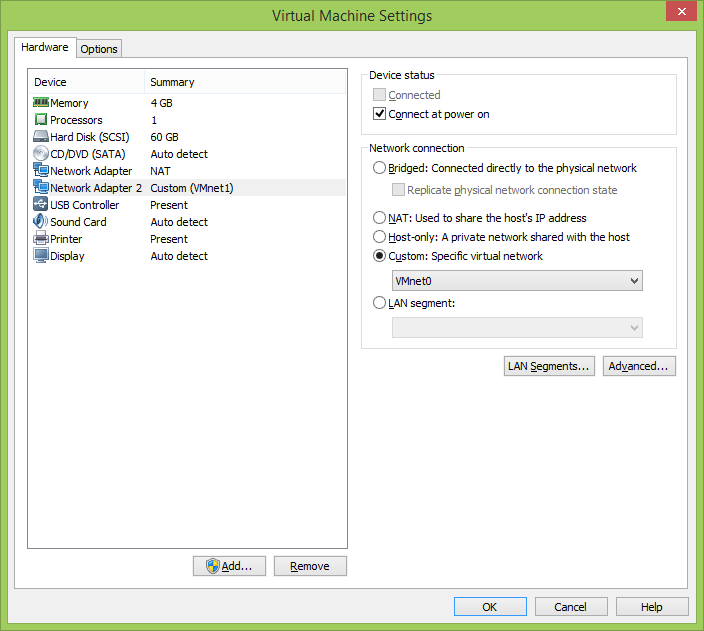
### Configuration de la machine virtuelle

Pour associer la carte réseau de l’ordinateur à la machine virtuelle, il faut suivre les étapes ci-dessous :

* Éteindre la vm, aller dans *edit->Virtual Network Editor…->change settings*
* Dans la fenêtre des réseaux, changer la configuration en bridged to : Contrôleur PCIe (propre à l’ordinateur), carte réseau de l’ordinateur



Puis dans les settings de la VM, il faut aller changer le réseau pour utiliser celui que l’on vient de configurer.



La dernière étape est d’activer le réseau dans la machine virtuelle (icône réseau -> enabling networking).

### Accès ssh sans mot de passe

Il faut aller modifier le fichier /etc/ssh/sshd\_config sur la cible pour autoriser l’accès sans mot de passe. Pour accéder au fichier, il faut entrer sur la cible par la connexion série :

$ sudo minicom

On peut ensuite recherche le fichier et l’éditer avec vi ou vim. Une fois la configuration faite, il faut redémarrer la cible :

# reboot

Normalement, on devrait avoir accès à la cible en ssh en entrant la commande suivante dans la machine hôte :

$ ssh [root@192.168.0.11](mailto:root@192.168.0.11)

Pour valider la connexion Ethernet/IP, on peut également taper les commandes suivantes dans la cible.

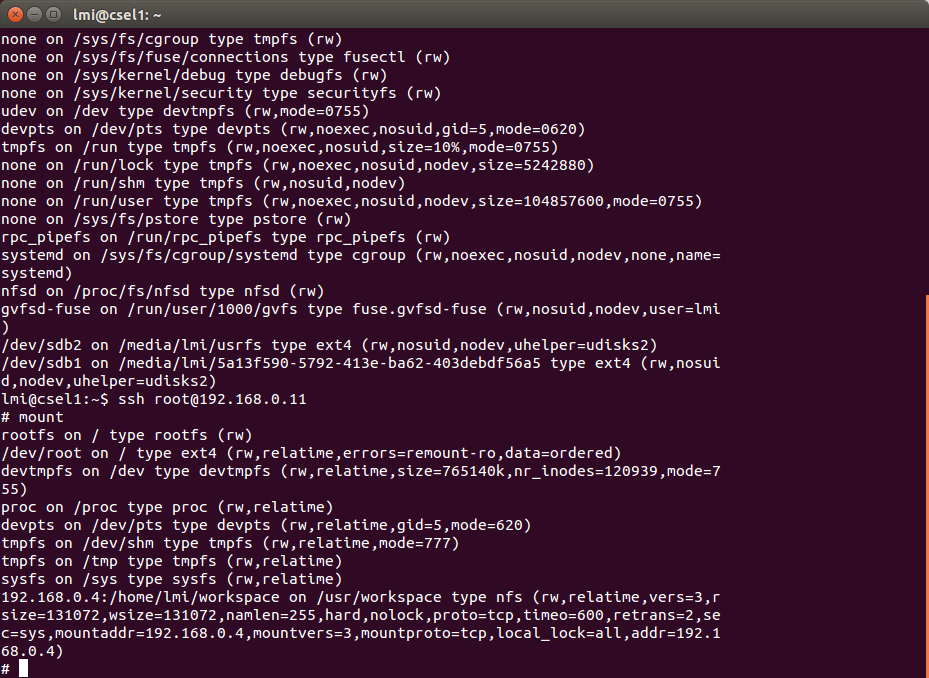
> Usb start

> ping 192.168.0.4

Il faut exécuter ces commandes après avoir arrêté la cible dans son U-Boot (carriage return au démarrage). Si la machine hôte répond au ping, tout a donc bien été configuré.

# Création de l’espace de travail

Le but est de configurer le noyau Linux afin de mounter un usrfs sous ext4 depuis la carte SD. En d’autres termes, partager un espace de travail entre la cible et l’hôte. Pour cela, il faut accéder à la cible via le port série ou par ssh et de taper les commandes indiquées dans la donnée.

Pour contrôler que le répertoire est bien partagé avec la cible, on peut y entrer la commande mount et normalement on voit le répertoire partagé.

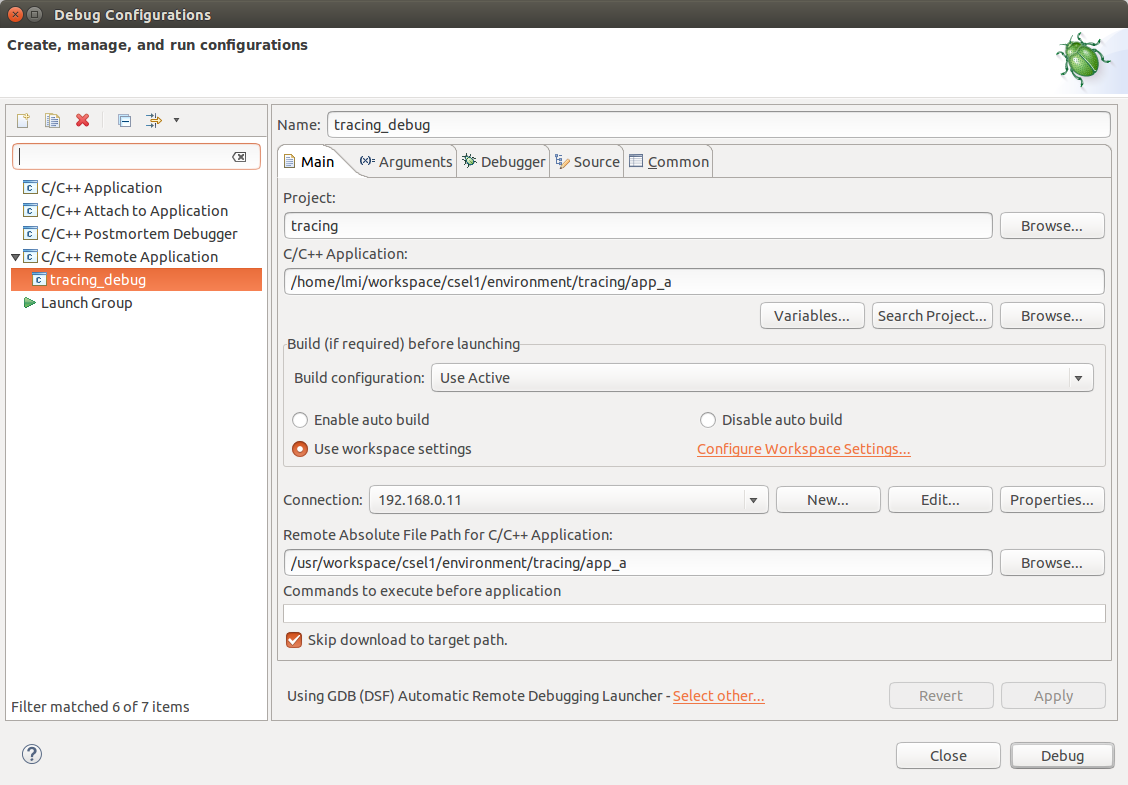
## Debugging de l’application

Cette section présente les différentes étapes à réaliser pour configurer Eclipse pour qu’il utilise une connexion ssh entre l’hôte et la cible.

Pour cela, il faut commencer par charger un projet dans Eclipse :

* File -> Import… (si le projet existe déjà)
* C/C++ ->Existing Code as Makefile Project (configure pour utiliser le Makefile du projet et non un de Eclipse)
* Il suffit ensuite de définir le nom du projet et le path jusqu’au code source

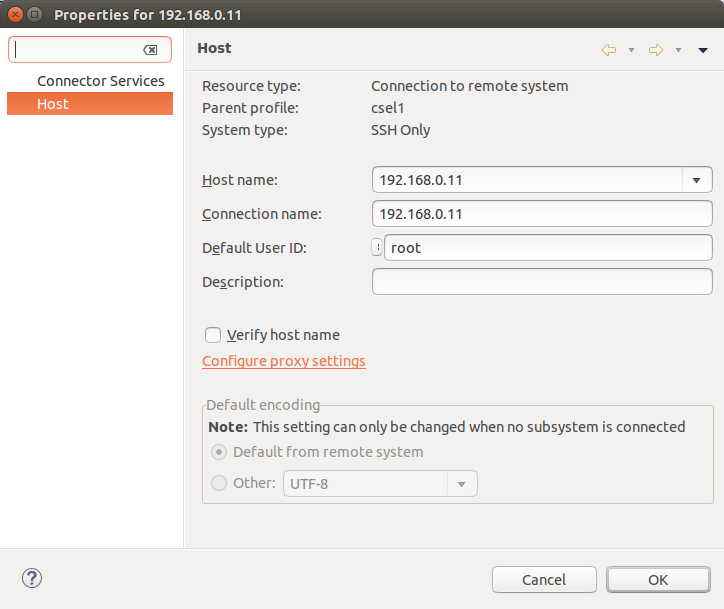
Une fois le projet importé dans l’espace de travail, il faut configurer le debugger:

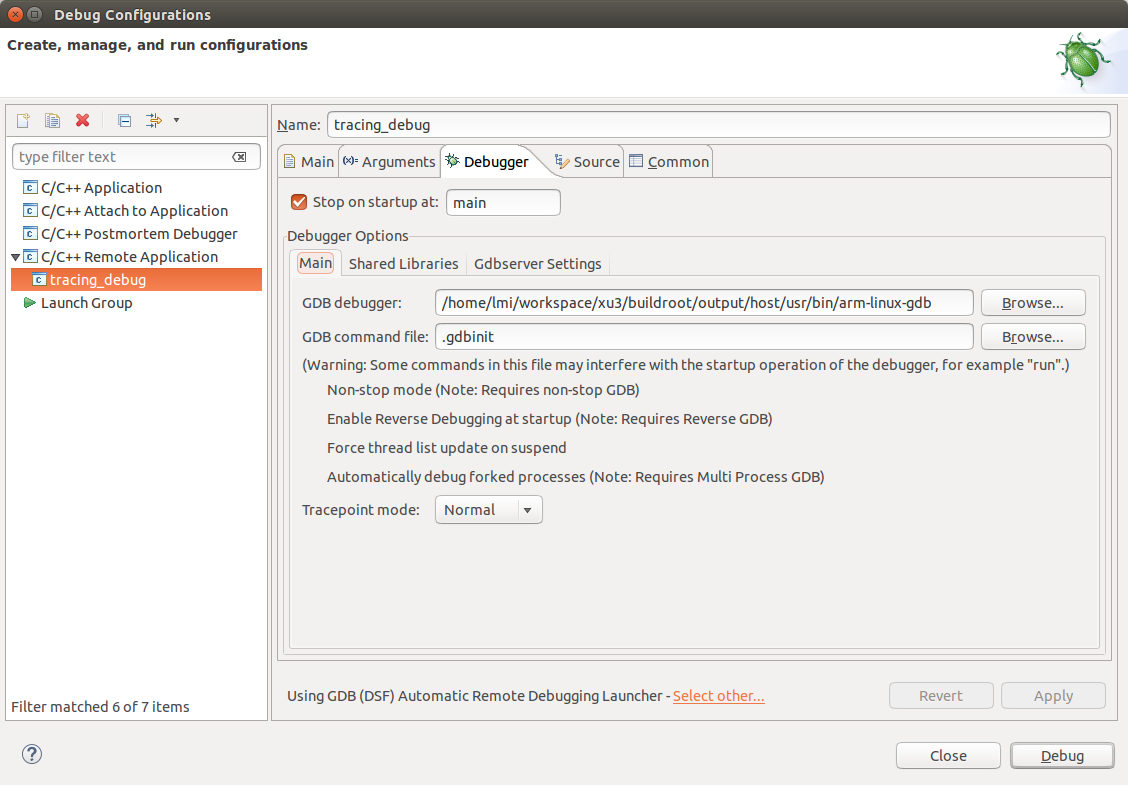


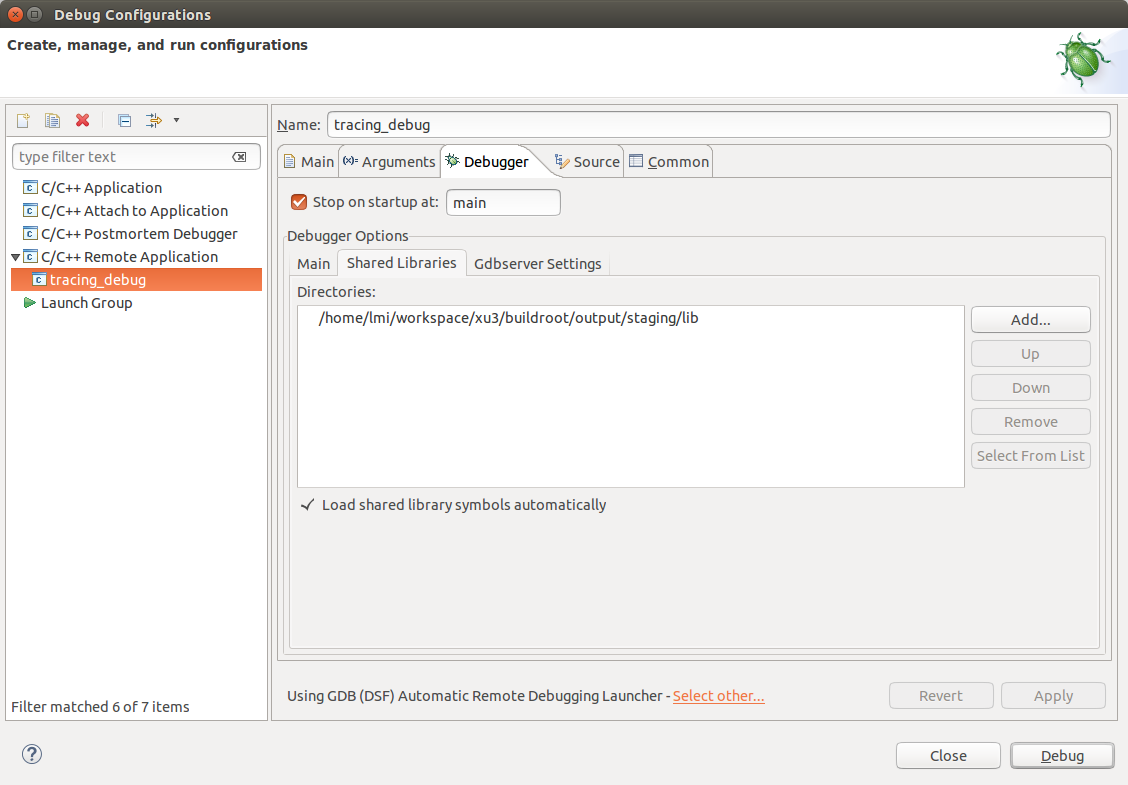
Path de la cible

Path de l’hôte

Connection ssh

Pour pouvoir entrer le path de la cible, il faut impérativement que la connexion ssh soit configurée comme ci-dessous :

Il faut encore aller configurer le debugger gdb :



Avec cette configuration, on peut ensuite debugger pas à pas les exercices d’exemples. Il faut tout de même changer le path du debugger pour qu’il corresponde à l’application à debugger.

# Réponses aux questions

1. aucune idée

2.

3.

4.

5.