FreeBSD USB Device Mode/USB OTG

Lors de l'exécution sur du matériel prenant en charge le mode périphérique USB ou USB OTG, comme celui intégré à de nombreuses cartes embarquées, la pile USB FreeBSD peut fonctionner en mode périphérique. Le mode Périphérique permet à l'ordinateur de se présenter sous la forme de différents types de classes de périphériques USB, y compris les ports série, les adaptateurs réseau et le stockage de masse, ou une combinaison de ceux-ci. Un hôte USB comme un ordinateur portable ou un ordinateur de bureau peut y accéder tout comme les périphériques USB physiques. Le mode appareil est parfois appelé “mode gadget USB”.

Le matériel peut fournir la fonctionnalité de mode périphérique de deux manières de base : avec un « port client » séparé, qui ne prend en charge que le mode périphérique, et avec un port USB OTG, qui peut fournir à la fois le mode périphérique et hôte. Pour les ports USB OTG, la pile USB bascule automatiquement entre le côté hôte et le côté périphérique, en fonction de ce qui est connecté au port. La connexion d'un périphérique USB comme une clé USB au port fait basculer FreeBSD en mode hôte. La connexion d'un hôte USB comme un ordinateur fait basculer FreeBSD en mode périphérique. Les “ports clients” à usage unique fonctionnent toujours en mode périphérique.

Ce que FreeBSD présente à l'hôte USB dépend du sysctl hw.usb.template. Certains modèles fournissent un seul périphérique, tel qu'un terminal série ; d'autres en fournissent plusieurs, qui peuvent tous être utilisés en même temps. Un exemple est le modèle 10, qui fournit un périphérique de stockage de masse, une console série et une interface réseau.

USB Virtual Serial Ports

Configuration des ports série en mode périphérique USB

La prise en charge du port série virtuel est fournie par les modèles numéro 3, 8 et 10. Le modèle 3 fonctionne avec Microsoft Windows 10 sans avoir besoin de pilotes spéciaux et de fichiers INF. D'autres systèmes d'exploitation hôtes fonctionnent avec les trois modèles. Les modules du noyau **usb_template** et **umodem** doivent être chargés.

Pour activer les ports série en mode périphérique USB, ajouter ces lignes à /etc/ttys :

```
ttyU0   "/usr/libexec/getty 3wire"    vt100    onifconsole secure
ttyU1   "/usr/libexec/getty 3wire"    vt100    onifconsole secure
```

Ajouter ensuite ces lignes à /etc/devd.conf :

```
notify 100 {
    match "system"        "DEVFS";
    match "subsystem"    "CDEV";
    match "type"        "CREATE";
    match "cdev"        "ttyU[0-9]+";
    action "/sbin/init q";
```

```
};
```

Recharger la configuration si devd(8) est déjà en cours d'exécution :

```
# service devd restart
```

Assurer-se que les modules nécessaires sont chargés et que le modèle correct est défini au démarrage en ajoutant ces lignes à /boot/loader.conf, en le créant s'il n'existe pas déjà :

```
umodem_load="YES"
hw.usb.template=3
```

Pour charger le module et définir le modèle sans redémarrer, utiliser :

```
# kldload umodem
# sysctl hw.usb.template=3
```

Connexion aux ports série en mode périphérique USB depuis FreeBSD

Pour se connecter à une carte configurée pour fournir des ports série en mode périphérique USB, connecter l'hôte USB, tel qu'un ordinateur portable, aux cartes USB OTG ou au port client USB. Utiliser pstat -t sur l'hôte pour lister les lignes de terminal. Vers la fin de la liste, se devrier voir un port série USB, par exemple “ttyU0”. Pour ouvrir la connexion, utiliser :

```
# cu -l /dev/ttyU0
```

Après avoir appuyé plusieurs fois sur la touche Entrée, se verrer une invite de connexion.

Connexion aux ports série en mode périphérique USB à partir de macOS

Pour se connecter à une carte configurée pour fournir des ports série en mode périphérique USB, connecter l'hôte USB, tel qu'un ordinateur portable, aux cartes USB OTG ou au port client USB. Pour ouvrir la connexion, utiliser :

```
# cu -l /dev/cu.usbmodemFreeBSD1
```

Connexion aux ports série en mode périphérique USB à partir de Linux

Pour se connecter à une carte configurée pour fournir des ports série en mode périphérique USB, connecter l'hôte USB, tel qu'un ordinateur portable, aux cartes USB OTG ou au port client USB. Pour ouvrir la connexion, utiliser :

```
```

```
# minicom -D /dev/ttyACM0
```

Connexion aux ports série en mode périphérique USB à partir de Microsoft Windows 10

Pour se connecter à une carte configurée pour fournir des ports série en mode périphérique USB, connecter l'hôte USB, tel qu'un ordinateur portable, aux cartes USB OTG ou au port client USB. Pour ouvrir une connexion, se aurer besoin d'un programme de terminal série, tel que PuTTY. Pour vérifier le nom du port COM utilisé par Windows, exécuter le Gestionnaire de périphériques, développer “Ports (COM & LPT)”. se verrer un nom similaire à “Périphérique série USB (COM4)”. Exécuter le programme de terminal série de votre choix, par exemple PuTTY. Dans la boîte de dialogue PuTTY, définisser “Type de connexion” sur “Série”, taper le COMx obtenu à partir du Gestionnaire de périphériques dans la boîte de dialogue “Ligne série” et cliquer sur Ouvrir.

Interfaces réseau en mode périphérique USB

La prise en charge des interfaces réseau virtuelles est fournie par les modèles numéro 1, 8 et 10. Noter qu'aucun d'entre eux ne fonctionne avec Microsoft Windows. D'autres systèmes d'exploitation hôtes fonctionnent avec les trois modèles. Les modules du noyau usb*template (4) et if*cdce (4) doivent être chargés.

S'assurer que les modules nécessaires sont chargés et que le modèle correct est défini au démarrage en ajoutant ces lignes à /boot/loader.conf, en le créant s'il n'existe pas déjà :

```
if_cdce_load="YES"
hw.usb.template=1
```

Pour charger le module et définir le modèle sans redémarrer, utiliser :

```
# kldload if_cdce
# sysctl hw.usb.template=1
```

Configuration du stockage de masse USB

Cette section fournit une description détaillée de la définition de la cible sans utiliser le fichier cfumass rc. Ceci est nécessaire si, par exemple, on veut fournir un LUN accessible en écriture.

Le stockage de masse USB ne nécessite pas l'exécution du démon **ctld**, bien qu'il puisse être utilisé si on le souhaite. Ceci est différent de l'iSCSI. Ainsi, il existe deux manières de configurer la cible : **ctladm** ou **ctld**. Les deux nécessitent le chargement du module noyau cfumass.ko. Le module peut être chargé manuellement :

```
# kldload cfumass
```

Si **cfumass.ko** n'a pas été intégré au noyau, `/boot/loader.conf` peut être configuré pour charger le module au démarrage :

```
cfumass_load="YES"
```

Un LUN peut être créé sans le démon **ctld**:

```
# ctladm create -b block -o file=/data/target0
```

Cela présente le contenu du fichier image `/data/target0` en tant que LUN vers l'hôte USB. Le fichier doit exister avant l'exécution de la commande. Pour configurer le LUN au démarrage du système, ajouter la commande à `/etc/rc.local`.

ctld peut également être utilisé pour gérer les LUN. Créer `/etc/ctl.conf`, ajouter une ligne à `/etc/rc.conf` pour s'assurer que **ctld** est automatiquement lancé au démarrage, puis démarrer le démon.

Ceci est un exemple de fichier de configuration `/etc/ctl.conf` simple:

```
target naa.50015178f369f092 {
    lun 0 {
        path /data/target0
        size 4G
    }
}
```

L'exemple crée une cible unique avec un seul LUN. Le **naa.50015178f369f092** est un identifiant de périphérique composé de 32 chiffres hexadécimaux aléatoires. La ligne de chemin définit le chemin complet d'un fichier ou zvol supportant le LUN. Ce fichier doit exister avant de démarrer **ctld**. La deuxième ligne est facultative et spécifie la taille du LUN.

Pour s'assurer que le démon **ctld** est démarré au démarrage, ajouter cette ligne à `/etc/rc.conf`:

```
ctld_enable="YES"
```

Pour démarrer **ctld** maintenant, exécuter cette commande :

```
# service ctld start
```

Au démarrage du démon **ctld**, il lit `/etc/ctl.conf`. Si ce fichier est modifié après le démarrage du démon, recharger les modifications afin qu'elles prennent effet immédiatement :

```
# service ctld reload
```