Pour configurer la vitesse et le format du port série, il faut une commande un peu fastidieuse, que j'ai donc mise dans un petit fichier :

```
[yg@Pavilion]$ cat 921600.sh
stty 921600 cs8 raw -echo -echoe -echonl -echok -echoprt -echoctl
-parenb -F $TTY
[yg@Pavilion]$ ./921600.sh
[yg@Pavilion]$ stty -F $TTY
speed 921600 baud; line = 0;
min = 1; time = 0;
-brkint -icrnl -imaxbel
-opost
-isig -icanon -echo -echoe -echok -echoctl
```

On devine que la commande **stty** va changer la vitesse à 921600 baud et les caractères sont sur 8 bits. Le reste est bien plus obscur et cela n'a rien à voir avec la liaison série elle-même, mais comment le terminal est géré. En fait on enlève tous les échos, sinon c'est la zizanie assurée, chaque caractère entré sera affiché avant même d'être envoyé. Et si la liaison série ne fonctionne pas, on sera dupé par cet affichage qui nous fait croire qu'on reçoit bien ce qu'on envoie... Donc gardez bien cette commande dans un coin, cela vous sera toujours utile.

La dernière commande vérifie que les paramètres (en particulier la vitesse) sont correctement configurés, et on peut commencer à jouer. Par exemple, on peut envoyer des caractères directement d'un terminal vers un autre en passant par le port série, si on branche ses broches d'envoi et de réception ensemble. Le système d'exploitation et les outils en ligne de commande ont parfois des comportements un peu étrange lorsqu'on tente de lire et d'écrire en même temps par deux programmes différents. Gardez patience, on finit toujours par y arriver.

Par exemple, dans un terminal, tapez cette commande :

```
[yg@Pavilion]$ od -wl -t xl $TTY
```

L'ordinateur va attendre que des octets arrivent sur **$TTY** et il affichera la valeur hexadécimale de chacun au fur et à mesure.

Pour envoyer des octets, c'est très simple, puisque **$TTY** est vu comme un fichier. Cette commande suffit pour envoyer les caractères entrés dans le terminal, vers le port série :

```
[yg@Pavilion]$ cat > $TTY
```

L'envoi n'est pas direct car le terminal attend l'appui sur la touche ENTRÉE pour envoyer un paquet d'octets. Mais cela suffit dans la plupart des cas pour s'assurer que le port série fonctionne.

2 Méthode de test

Pour chaque dongle USB, nous allons tester à la fois la vitesse de transmission et l'intégrité des données. Les outils de base fournis avec GNU/Linux suffisent largement pour cela.

En premier lieu, nous allons transmettre un fichier d'un terminal à un autre en passant par le port série (toujours avec TX et RX connectés ensemble). L'utilitaire **dd** fait l'affaire, car il permet de transférer des blocs de taille arbitraire et il indique le débit.

Ce fichier transmis pourrait être n'importe quoi mais pour nous assurer de la qualité de la transmission, j'ai choisi des octets aléatoires. Ainsi, des types d'erreurs non triviales ont plus de chance d'être détectés. Et pour créer ce fichier, nous utilisons de nouveau **dd**, qui va lire un million d'octets à partir du générateur de nombres pseudo-aléatoires du noyau :

```
[yg@Pavilion]$ dd if=/dev/urandom of=rnd bs=1000000 count=1
1+0 enregistrements lus
1+0 enregistrements écrits
1000000 octets (1,0 MB) copiés, 0,325864 s, 3,1 MB/s
```

Ensuite vient le test de la transmission du fichier. Dans un premier terminal, qui servira pour la réception, tapez ceci :

```
[yg@Pavilion]$ dd if=$TTY of=rnd2 bs=10000000
```

Remarquez la taille de bloc supérieure à celle du nôtre. C'est pour s'assurer que la lecture est faite en continu, d'un coup, car sinon **dd** a une taille de bloc par défaut de 512 octets et les appels système répétés peuvent réduire la performance.

Ensuite, dans un autre terminal, du côté émission, on lance ceci :

```
[yg@Pavilion]$ dd if=rnd of=$TTY bs=1000000 ; sync
1+0 enregistrements lus
1+0 enregistrements écrits
1000000 octets (1,0 MB) copiés, 10,7876 s, 92,7 kB/s
```

A la fin de la commande, il ne faut pas oublier **sync** pour vider les tampons de transmission et de réception, sinon le transfert risque de perdre des données sur la fin.

Une fois la transmission complète, on interromp le récepteur avec CTRL+C :

```
^C0+10662 enregistrements lus
0+10662 enregistrements écrits
1000000 octets (1,0 MB) copiés, 28,1837 s, 35,5 kB/s
```

Enfin on compare les deux fichiers :

```
[yg@Pavilion]$ ls -al rnd*
-rw-rw-r--. 1 yg yg 1000000 2 avril 11:23 rnd
-rw-rw-r--. 1 yg yg 1000000 2 avril 11:29 rnd2
[yg@Pavilion]$ diff rnd rnd2
[yg@Pavilion]$
```

Si la commande n'affiche rien, alors les deux fichiers sont identiques et le test a réussi !