Le port série

Le port série est souvent le seul moyen de communication disponible sur un système embarqué (surtout en phase de configuration).

Références :

http://tldp.org/HOWTO/Serial-Programming-HOWTO/x56.html#AEN88
http://tldp.org/HOWTO/Serial-HOWTO.html
http://www.nullmodem.com
http://www.easysw.com/~mike/serial/serial.html

Considérations matérielles :
DTE signifie Data Terminal Equipment et correspond en général au PC, alors que DCE signifie Data Communication Equipement et correspond au périphérique.
Pour relier deux éléments de même nature il faut toujours un câble croisé.
Pour relier un DCE à un DTE un câble droit suffit.
Si le contrôle de flux matériel n'est pas utilisé, il suffit de câbler Txd,Rxd et la masse.
Sur un système embarqué on peut avoir soit un port à la norme RS232 soit un port fonctionnant directement avec les tensions du microprocesseur (3.3 ou 5V). Dans ce dernier cas un adaptateur vers RS232 est nécessaire (circuit type MAX232).
Si votre pc ne comporte pas de port série un adaptateur USB/RS232 est nécessaire.

I/ Utilisation en ligne de commande

Les ports série apparaissent comme des fichiers spéciaux dans /dev.
Généralement /dev/ttyS0, /dev/ttyS1,... et /dev/ttyUSB0,/dev/ttyUSB1,... si vous utiliser un adaptateur USB.
Listez les ports série de votre machine (ls -l) et déterminez ceux qui sont réellement présents (par exemple avec stty). Afficher leurs configurations.
Configurez un port série pour 19200 bit/s parité impaire (stty).
Reliez deux pc entre eux et comminuquez (echo et cat).
Sauvez dans un fichier les données reçues, envoyer le contenu d'un fichier texte.
Installer le programme gtkterm et utiliser le pour communiquer. Avec ce programme vérifier que vous savez sauvegarder une configuration, envoyer un fichier texte, sauvegarder dans un fichier les données reçues.

II/ Utilisation en C

Lire le **serial programming HOWTO** et expliquer les différences entre les modes canonique et non-canonique ainsi qu'entre le fonctionnement synchrone et asynchrone.
En vous inspirant de l'exemple écrivez un programme (en mode canonique synchrone) qui transmet une chaîne de caractères toutes les cinq secondes et un programme qui lit et affiche les chaînes reçues. Vos programmes prendront le périphérique et la vitesse de transmission en argument sur la ligne de commande : readtty -d /dev/ttyUSB0 -b 2400 (sans option on utilise /dev/ttyS0 à 9600bd). (utilisez getopt() voir le manuel de la libc à http://www.gnu.org/software/libc/manual/)
Pour la vitesse, vous pouvez reprendre une partie du code libre à cette adresse :
http://cvs.sourceforge.jp/cgi-bin/viewcvs.cgi/*checkout*/uclinux-h8/userland/stty/stty.c?content-type=text%2Fplain&rev=1.1

Intérêt du mode asynchrone :

Souvent le problème ce pose en ces termes :

- on envoie une commande à un appareil sur le port série et attend sa réponse sur ce même port.

ou

- on souhaite récupérer des trames qui sont émises à des intervalles irréguliers.

Dans ces deux cas, le programme ne peut pas rester bloquer indéfiniment dans la fonction read. Il faut donc que l'appel ne soit pas bloquant pour pouvoir reprendre la main au bout d'une durée maximale (timeout).

– Écrire un programme qui lit en permanence des données de manière asynchrone et les affichent. Si aucune donnée n'est reçue au bout d'un timeout (en µs) paramétrable, le programme affiche un message d'alerte.

Ex : $./interrogation -d /dev/ttyUSB0 -b 57600 -t 5000000
waiting for data...
Received : bonjour
waiting for data...
Timeout while waiting for data 5000228 uSecs passed
waiting for data...
Received : allo
…...

Utiliser la fonction gettimeofday dans <sys/time.h> pour la gestion du timeout.

Écrire une fonction réutilisable **int ReadResponse(char * response , int timeout_us);** qui attend une réponse sur le port série (présumé ouvert avec le descripteur global fd). La fonction retourne le nombre de caractères reçus ou une valeur négative en cas d'erreur (timeout, erreur de lecture).