TD 1 : Entrées/sorties caractère sous interruptions

Objectif

L'objectif de cet exercice est de programmer un pilote de périphérique caractère par attente active puis demande d'interruptions. Le contrôleur de périphérique est très simple pour se libérer des contraintes d'accès aux registres du contrôleur.

Spécification du matériel

Une ligne série est utilisée pour communiquer entre deux ordinateurs. Les opérations d'envoi et de réception d'un caractère sont deux opérations indépendantes pouvant se dérouler en parallèle.

Le contrôleur d'entrées/sorties comprend quatre registres internes :

- Un registre d'entrée pour la réception d'un caractère
- Un registre de sortie pour l'émission d'un caractère
- Un registre d'état
- Un registre de contrôle (commande)

Le contrôleur (en *émission* ou *réception*) peut être géré en mode "test d'état" (attente active), ou avec demande d'interruption. Les niveaux d'interruption *it_e* (entrée) et *it_s* (sortie) sont distincts. On dispose des primitives suivantes pour la programmation du contrôleur :

- *void lire(char *c)* : range dans *c* le contenu du registre d'entrée
- *void écrire(char c)* : range *c* dans le registre de sortie
- *char état_sortie* : rend vrai quand le registre de sortie est vide (on peut envoyer un caractère)
- *char état_entrée* : rend vrai quand le registre d'entrée est plein (un caractère a été reçu et peut être lu)
- *void autoriser_it_s()* : une interruption sur le niveau it_s est postée quand le registre de sortie *devient* vide. L'interruption est acquittée en écrivant un caractère dans le registre de sortie
- *void autoriser_it_e()* : une interruption sur le niveau it_e est postée quand le registre d'entrée *devient* plein. L'interruption est acquittée en lisant un caractère.
- *void interdire_it_e()* : le contrôleur ne poste pas d'interruption sur le niveau *it_e*
- *void interdire_it_s()* : le contrôleur ne poste pas d'interruption sur le niveau *it_s*
- *void initialiser()* : Initialise le contrôleur. Le contrôleur ne poste pas d'interruption sur les niveaux *it_e* et *it_s*.

Dans un premier temps, on suppose qu'un seul processus utilise la ligne série. On souhaite réaliser les trois primitives suivantes :

- void tty_init(void)
- *void tty_em_ligne (char *l);* : émet la ligne contenue dans le tableau l. Une ligne se termine par un zéro.
- *void tty_rec_ligne (int *lg, char *l, int maxl)* : reçoit une ligne de caractères terminée par 0 et retourne sa longueur dans lg. Pour éviter les débordements de tampon, la ligne est tronquée si elle est plus longue que *maxl* caractères.

1. Ecrire les trois opérations *tty_init, tty_em_ligne* et *tty_rec_ligne*, le contrôleur étant programmé en mode test d'état (attente active). L'émission et la réception dans cette question sont des opérations synchrones.

2. Ecrire de nouveau les trois opérations en programmant maintenant le contrôleur avec demande d'interruption en entrée et en sortie. La réception est gérée avec *anticipation* : les caractères reçus sont mémorisés dans un *tampon de réception* jusqu'à leur consommation (appel à *tty_rec_ligne*). Un tampon est également affecté à l'émission. Dès que la ligne à émettre est recopiée dans le tampon émission, le processus suppose que l'entrée/sortie est terminée, c'est alors la routine de traitement d'interruptions qui prend le relai. Vous écrirez également les traitants d'interruption associés aux niveaux *it_e* et *it_s*.

3. Le partage de la ligne série entre plusieurs processus avec le code des deux questions précédentes pose t-il des problèmes de cohérence des lignes émises/reçues ? Si oui, modifier le code.