Université Joseph Fourier, Grenoble
Licence d'Informatique L3

Introduction aux Systèmes et Réseaux

TP n°6 : La Bibliothèque RIO

L'objectif de ce TP est de maîtriser les principes et l'utilisation des primitives de la bibliothèque RIO, qui seront utiles pour les TP suivants.

Préambule Commencer par lire attentivement la documentation technique (*Bibliothèques d'Entrées-Sorties*) disponible dans le placard électronique.

Question 1

Récupérer et examiner le fichier `q1.c` disponible dans le placard du TP n°6. Ce programme est basé sur deux processus qui communiquent au moyen d'un tube. Le processus émetteur lit des données à partir d'un fichier d'entrée et les envoie dans le tube ; le récepteur lit ces données (d'un type et d'une taille prédéfinis) à la sortie du tube et affiche, sur la sortie standard, une chaîne de caractères correspondant à la valeur numérique lue.

Le code de l'émetteur (exécuté par le processus père) est fourni et ne doit pas être modifié. On remarquera que l'émetteur dépose des données dans le tube assez lentement (endormissement après chaque dépôt) et que la quantité de données produites à chaque pas est variable et potentiellement inférieure à la quantité consommée pour une itération du récepteur.

Il est demandé de compléter le code du récepteur (exécuté par le processus fils) en utilisant le primitive `Read` et en veillant à récupérer la bonne quantité de données. Écrire ensuite une nouvelle version du code du récepteur, en simplifiant la gestion du problème grâce à une ou plusieurs des fonctions de la bibliothèque RIO.

Vérifier que les programmes écrits fonctionnent correctement lorsque la lecture (par le récepteur) du dernier message est incomplète [1]. Cette situation peut se produire si l'émetteur envoie un nombre d'octets inférieur à la taille attendue puis se termine (par exemple, suite à une erreur inopinée). On pourra simuler ce cas de figure en utilisant le fichier d'entrée `input2.bin`.

Question 2

Récupérer le fichier `q2.txt` disponible dans le placard du TP n°6. Écrire un programme qui ouvre (une seule fois) ce fichier puis lit et affiche son contenu à l'écran en respectant les étapes suivantes :

1. initialisation du tampon avec `Rio_readinitb` ;
2. lecture d'une ligne via la fonction `Rio_readlineb` puis affichage avec `printf` ;

1. Dans le cas de cet exercice, la spécification d'un fonctionnement correct est la suivante : ne pas afficher la valeur d'un message reçu de façon tronquée (le message est simplement ignoré).

3. lecture de trois octets via la fonction `Rio_readnb` puis affichage avec (une boucle d'appels à) `putchar` ;
4. lecture de six octets via la fonction `Rio_readn` puis affichage avec (une boucle d'appels à) `putchar`.

Quel est le résultat observé ? Comment peut-on l'expliquer et que faut-il en conclure sur l'utilisation des différentes fonctions (de lecture) de la bibliothèque RIO ?

Question 3

Prendre le fichier q3.c disponible dans le placard du TP n°6. Définir un traitant (un simple affichage) pour le signal `SIGUSR1` [2] et envoyer ce signal à l'un des deux processus de l'application [3] en utilisant la commande `kill` à partir d'un autre terminal. Que constatez-vous ? Modifier le programme en utilisant une ou plusieurs fonctions de la bibliothèque RIO pour corriger le problème.

Remarque : La plupart des systèmes Unix actuels sont configurés de façon à redémarrer automatiquement un appel système interrompu par l'arrivée d'un signal. La bibliothèque RIO ne vous sera donc pas nécessairement utile à cet égard mais elle vous permet d'écrire du code d'E/S portable, avec un comportement identique quelle que soit la configuration du système sous-jacent.

Par ailleurs, on peut, dans certains cas, avoir besoin de désactiver le rédemarrage automatique des appels système suite à l'arrivée d'un ou plusieurs types de signaux (par exemple, pour débloquer et terminer proprement un processus en attente de lecture sur un tube). La primitive `siginterrupt` est prévue à cet effet.

2. **Important** : Pour les besoins de cet exercice, insérer la ligne suivante après l'appel à la fonction `Signal` : `siginterrupt(SIGUSR1, 1);`
3. Lorsque l'utilisateur n'est pas en train de taper des caractères, chaque processus est bloqué : le père est bloqué en lecture sur le terminal et le fils est bloqué en lecture sur la sortie du tube.