Travaux Pratiques de Système n°7

Exercice 24 : Copie de fichiers

Le but de cet exercice est de réaliser une copie de fichier à l'aide de deux processus : un qui va lire le fichier à copier et un autre qui va écrire le nouveau fichier. Les deux processus vont communiquer à l'aide d'un tube. La commande prendra donc en paramètre deux noms de fichier : la source et la destination.

Question 24.1 Vérifier que la commande a bien deux paramètres et écrire un message sur l'erreur standard sinon.

Question 24.2 Vérifier que le fichier source est un fichier régulier, et que les liens source et destination ne pointent pas sur le même inode. Terminer le programme et écrire un message sur l'erreur standard sinon.

Question 24.3 Créer les deux processus et le tube et faire la copie comme indiqué. On veillera à ce que tous les descripteurs ouverts soient correctement fermés dans tous les processus, en contrôlant les erreurs pouvant se produire lors de ces fermetures de descripteurs.

Question 24.4 Utiliser les macros adéquates (WIFEXITED, WEXITSTATUS, WIFSIGNALED, WTERMSIG) pour afficher dans le processus père comment les processus fils se sont terminés.

Question 24.5 À votre avis, est-ce que cette méthode permet d'accélérer la copie par rapport à un seul processus ? Pourquoi ?

Question 24.6 On veut à présent pouvoir arrêter la copie en cours et supprimer le fichier destination à l'aide de `CTRL+C`. Modifier le programme pour mettre en place ce comportement. Indice : `unlink(2)`.

Exercice 25 : Calcul de π

Une méthode pour obtenir une approximation de π est de tirer aléatoirement n points (x, y) avec $x, y \in [0, 1]$. Parmi ces n points, p points appartiennent au disque unité (de centre O et de rayon 1), c'est-à-dire que $x^2 + y^2 < 1$. Or, la probabilité de tomber dans le disque unité est de $\frac{\pi}{4}$. Une approximation de π est donc $4 * \frac{p}{n}$. On utilisera le type `unsigned long` pour n et p.

Question 25.1 En utilisant `random(3)` et `INT_MAX`, écrire une fonction C qui renvoie un `double` compris dans l'intervalle $[0, 1]$.

Correction de la question 25.1

```
double get_double(void) {
  return (double) random() / INT_MAX;
}
```

Question 25.2 Écrire une commande `seq_pi` qui fait le calcul de π suivant la méthode indiquée en utilisant un seul processus. On fixera n à 10^7.

Correction partielle de la question 25.2

```
void compute_pi(void) {
  for (unsigned long i = 0; i < N; ++i) {
    double x = get_double();
    double y = get_double();
    if (x*x + y*y < 1.0) {
      ++p;
    }
  }
}
```

À partir de maintenant, on veut lancer plusieurs processus en même temps pour accélérer le calcul. Pour faire le calcul final de π, on va propager les résultats [5] avec un tube entre chacun des processus, le dernier processus réalisant le calcul final de π.
On prendra garde à initialiser le générateur aléatoire en utilisant `srand(3)` et `getpid(2)` (de manière à ce que chaque processus ait une graine différente).

Question 25.3 Écrire une commande `biprocess_pi` qui utilise deux processus et un tube. On fixera n à 10^7.

Question 25.4 Écrire une commande `interrupted_pi` qui utilise deux processus et qui calcule tant qu'elle n'est pas interrompue (par un `CTRL+C`).

Question 25.5 Écrire une commande `parallel_pi` qui utilise q processus, q étant passé en paramètre de la commande et qui calcule tant qu'elle n'est pas interrompue (par un `CTRL+C`).

5. p pour la **question 25.3**, (n et p) pour les suivantes.