Systèmes informatiques
Projet 2 - Support pour l'interview de design

Justine Doutreloux
Nicolas Breels

10 avril 2015

1 Architecture

Il y a 6 fonctions de bases.

- `void take_number(int number, char *file)` : Cette fonction prend un nombre, elle le factorise en nombre premier. Ensuite elle écrit les différents nombres premier dans un fichier les uns à la suite de l'autres et elle termine en écrivant le nom du fichier. Si l'argument fichier est NULL, elle écrit : " noFile".
- `void only_number(char *file)` : Prend le fichier qui a été écrit par les différents threads, affiche le nombre unique à la première ligne, le nom du fichier à la deuxième et le temps d'exécution à la troisième.
- `void take_fichier(char *file)` : Prend un fichier, converti chaque nombre avec la librairie endian(3) et aplique `take_number`
- `int max_threads()` : retourne le nombre maximum de thread rentré par l'utilisateur.
- `void add_number()` : Applique `take_number`
- `void add_file()` : Télécharge le fichier à l'adresse URL rentré par l'utilisateur et l'enregistre dans le dossier courrant du projet. Applique `take_fichier`

2 Threads

Pour ce projet, nous pensons utilisé un thread par fichier à analyser. Chaque thread effectuera la même chose sur un fichier différent. Les fonctions expliquées ci-dessus seront appliquées sur les fichiers et le thread écrira les réponses, suivies du nom du fichier, dans un fichier central. Nous utiliserons les sémaphores pour régler le problème de synchronisation. Les threads seront ensuite terminés et le programme reprendra son exécution normale.

3 Synchronisation

Nous utilisons les sémaphores car nous les trouvons plus faciles à appliquer dans le cas qui nous occupe. Il faudra faire attention aux différentes utilisations de post et wait. Deux threads ne pourront pas écrire dans le fichier central en même temps.

4 Structures de données

Les facteurs premiers seront échangés via un fichier central dans lequel chaque thread écrira (synchronisation via les sémaphores).

5 Algorithme

```
#include <stdio.h>
unsigned int liste_facteurs[32];
int i = 0;
void f(unsigned int n, unsigned int d)
{
  if (n != 1)
    {
      if (n % d)
        {
          d++;
          f(n, d);
        }
      else
        {
          liste_facteurs[i] = d;
          i++;
          f(n / d, d);
        }
    }
}
int main(void)
{
  unsigned int n;
  int k;
  scanf("%u", &n);
  f(n, 2);
  for (k =0; k < i; k++)
  printf("%d␣", liste_facteurs[k]);
  printf("\n");
  return 0;
}
```

Source : http ://openclassrooms.com/forum/sujet/produit-de-facteur-premier-58167.