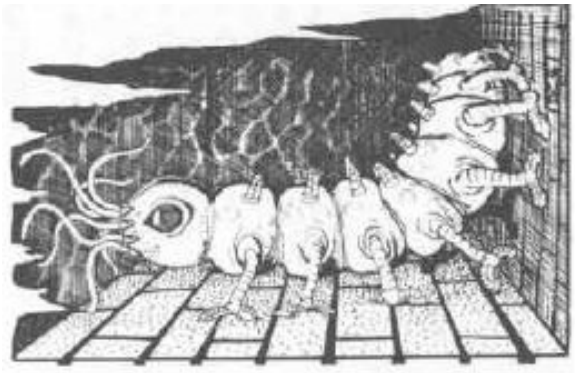
Travail pratique #03

Objectifs

— Expérimenter la multiprogrammation dans un cas d'utilisation réel
— Utiliser le multi-thread pour obtenir un gain de performance
— Convertir un programme standard pour le rendre parallèle
— Utiliser les primitives de synchronisation
— Tirer profit de la liaison dynamique pour favoriser l'extensibilité d'un programme

Règles importantes

1. Chaque classe doit avoir son propre fichier *.h* et *.cpp*
2. N'oubliez pas les *"include guard"*
3. Prenez grand soin de respecter la casse pour les noms de classe, méthode, etc
4. Veuillez commenter votre code *intelligemment*
5. Votre code doit être de qualité exemplaire
6. **Votre code doit compiler sans erreurs pour être corrigé**



Remarques

ATTENTION : En aucunes circonstances ce travail ne doit être utilisé et/ou testé sur de vrais sites web. Un environnement de test vous est fourni et il est fortement recommandé de déconnecter votre ordinateur d'internet lors de vos tests pour éviter qu'une erreur ne se produise et aie des répercussions indésirables.
De plus, puisque votre dévoué prof aime bien sa job et espère la conserver le plus longtemps possible, il serait judicieux de travailler d'un endroit ou il est impossible pour votre ordinateur de communiquer avec le monde extérieur, et ce peu importe le moyen de communication possible (wifi, internet, pigeon voyageur, télégraphe, signaux de fumée). L'idéal serait de travailler dans une cage de faraday, construite dans une caverne, dans un pays du tier-monde (ou à St-Marcel). Méfiez-vous de la *Pbzzvffvba Fpbynver qr Fg-Ulnpvagur* !

(SVP lire l'ensemble de ce document avant de commencer le travail)

Programme de base fourni

Un programme de base (nommé *crawler*) est fourni avec ce travail. Il s'agit d'un robot dont la principale tâche est de parcourir automatiquement des pages web pour en extraire de l'information. Le programme lit une liste d'urls à partir d'un fichier texte (*urls.txt*) et pour chacune d'entre elles il télécharge le code HTML et extrait les courriels trouvés (liens de type mailto).

Environnement de test

Pour tester votre programme avec le jeu de test fourni (répertoire *tests*) vous devez démarrer l'exécutable nommé httpd.exe contenu dans ce répertoire. Celui-ci crée un serveur web local qui écoute sur le port 8000 et qui permet d'accéder aux pages web contenues dans le répertoire *tests*. Bien qu'un fichier *urls.txt* est déjà fourni avec le projet, d'autres sont aussi fournis avec les jeux de tests. Au besoin, vous pouvez tester avec n'importe quel sous-ensemble des tests en utilisant les urls de votre choix parmis celles fournies dans ce répertoire.

L'environnement de test fourni essaie de recréer le plus fidèlement possible les conditions réelles d'un crawler parcourant internet, incluant la latence réseau, la conformance au protocole HTTP, le type et le format des documents retournés, etc. Si votre crawler fonctionne bien avec les jeux de tests fournis, il est très raisonnable de penser qu'il fonctionnerait aussi bien s'il utilisait une liste d'url provenant de vrais site web dans le fichier *urls.txt*.

C'est la première fois que ce travail est fait dans un environnement de test plutôt que sur de vrais sites web, en conséquences il se peut qu'il y aie des ajustements à faire en cours de travail et que de nouveaux jeux de tests vous soient fournis si nécessaire.

Instructions

Le programme fourni a deux lacunes :

1. On aimerait que le programme ne récupère pas seulement la liste des courriels contenus dans chaque page, mais qu'il puisse être possible d'ajouter des fonctionnalités pour récupérer d'autres informations à l'aide de plugins. Encore mieux, on aimerait être capable d'ajouter des plugins sans avoir à recompiler le crawler, simplement en déposant une ou des DLLs dans le répertoire du programme.
2. Vu qu'une seule page est téléchargée à la fois, ce n'est pas optimal et plusieurs facteurs (le temps de latence du réseau, le délais de réponse du serveur distant, etc) peuvent causer des ralentissements.

Amélioration 1 : Système de plugins

Votre programme doit pouvoir charger et gérer des plugins qui auront la tâche de traiter l'information téléchargée. Chaque fois que le contenu HTML d'une page web est téléchargé, le contenu de la page (code HTML stocké dans un *std :: string*) doit être envoyé à chacun des plugins. La façon de procéder et la logique du gestionnaire de plugins sera très semblable à la démonstration faite en classe et il est suggéré de vous en inspirer.

Voici la classe de base à partir de laquelle les plugins devront construire un objet et en donner un pointeur lorsque demandé (déjà fournie dans le projet de base) :

Listing 1 – Classe abstraite PageHandler

```
class PageHandler
{
public:
  virtual ~PageHandler() {}

    // Méthode appelée pour chaque page, le contenu html de la page est passé en paramamêtre
  virtual void HandlePage(std::string& html) = 0;

    // AfficherResultat sera appelée une seule fois lorsque le crawler aura terminé de s'exécuter et affichera l'information demandée
    virtual void AfficherResultat() = 0;
}
```

Chaque plugin devra exporter une fonction nommée *GetHandler* non-décorée (extern "C") qui retournera un pointeur vers une instance d'un objet ayant comme type parent *PageHandler* :

Listing 2 – Fonction exportée GetHandler

```
extern "C"
{
  __declspec(dllexport) PageHandler* GetHandler()
  {
        // à compléter
  }
}
```

Une fois votre système de plugins en place, veuillez créer les deux plugins (projet de type DLL dans la même solution que le crawler) suivants :

1. Un plugin qui extrait toutes les adresses courriels présentes dans chaque page et les conserve en mémoire. Une fois que le crawler aura terminé de s'exécuter, la méthode *AfficherResultat* du plugin sera appelée et devra afficher la liste des courriels **uniques** récupérés (un par ligne) suivi d'une ligne contenant le nombre total de courriels **uniques** récupérés. Veuillez utiliser (et enlever) le code qui fait déjà la majeure partie de plugin du crawler et le déplacer dans le plugin.
2. Un plugin qui extrait tous les titres principaux de niveau 1 de chaque page (texte entre balises HTML `h1`). La méthode *AfficherResultat* de votre plugin devra afficher tous les titres (incluant les doublons) trouvés, un par ligne, suivi d'une ligne qui indique le nombre de titre trouvés.

Il est important que les plugins n'affichent rien en console, sauf lorsque leur méthode *AfficherResultat* sera appelée.

Comme dans la démonstration faite en classe, votre crawler doit charger automatiquement au démarrage toute DLL qui se trouve dans le même répertoire que l'exécutable et l'utiliser comme un plugin. Outre les 2 plugins demandés, n'importe qui devrait pouvoir ajouter un plugin sous forme de DLL qui respecte les contraintes demandées et celui-ci devrait être utilisé pour ajouter des fonctionnalités supplémentaires au besoin sans que cela n'implique de modification au code de votre crawler et une recompilation.

Veuillez conserver l'affichage en console du programme original lorsque possible, plus particulièrement la séquence de points qui s'affichent pour donner un aperçu de la vitesse de traitement

Amélioration 2 : traitement parallèle (multi-thread)

En continuant avec le programme que vous avez amélioré de la section précédente, vous devez le rendre plus rapide en respectant les règles suivantes :

1. Vous ne devez pas changer l'affichage du programme. Suite à vos optimisations, le seul changement apparent sera la vitesse d'exécution. Les différentes informations (points, liste de email, nombre de email, liste de titres, nombre de titre, etc) doivent être affichées dans le même ordre (il est normal que les emails ou les titres soient dans un ordre différent par contre).
2. Vous devez modifier le programme pour qu'il utilise 20 threads.
3. Vous devez diviser le nombre d'url à traiter **le plus également possible** entre les threads.
4. Vous devez protéger l'accès concurrent à la liste de courriels ainsi qu'a toute autre variable utilisée par plus d'un thread à l'aide d'un mutex.
5. Lorsque la liste des urls à parcourir est vide, chaque thread doit s'arrêter proprement. Le programme doit ensuite attendre qu'ils soient tous terminés avant d'afficher la liste des courriels et le temps d'exécution.
6. Votre code doit être robuste et ne doit planter sous aucunes considérations. Vous pouvez assumer que le fichier d'urls (*urls.txt*) sera toujours valide.
7. **Votre crawler doit s'auto-nourrir.** Vous devez trouver et extraire tout les liens contenus dans chaque pages (les balises *a href="..."*) pour les ajouter à la liste des adresses à traiter.
8. Vous ne devez pas parcourir plus d'une fois la même url.

Remise

Vous devez me remettre sur vortex votre solution *Visual studio* complète **une fois la solution nettoyée** :

1. Menu Générer, Nettoyer la solution
2. Effacer manuellement le répertoire .vs et les autres fichiers inutiles
3. Ne pas me remettre de jeux de tests