Institut Galilée — Systèmes d'exploitation et Réseaux
Licence 2

TP noté

Threads

Le but de ce TP est d'écrire un programme qui lit le contenu d'un répertoire, dont le chemin est passé en argument du programme. Pour chaque fichier contenu dans ce répertoire, le programme va vérifier s'il contient votre numéro d'étudiant. Là où cela va se compliquer, c'est que nous allons utiliser des threads.

Ce TP est découpé en plusieurs exerices, afin de vous aider à réaliser le tp pas à pas.

1 Mise en place

— Téléchargez sur l'ENT les fichiers `tp_sujet.c` et `liste.txt`
— Renommez le fichier `tp_sujet.c` au format `NOM_PRENOM_NUMERO_ETUDIANT.c`.
— Modifiez la première ligne du fichier afin d'y ajouter vos noms prénoms et numéro d'étudiants.

2 Fonction de manipulation de structure

Cet exerice a pour but de vous rafraîchir la mémoire sur votre cours de programmation impérative. C'est aussi l'occasion de grapiller facilement des points.

1. Écrire la fonction `argument_texte_t creer_argument(char *, int )` ; qui initialise les champs de la structure `argument_texte_t` aux valeurs passées en argument.
2. Écrire la fonction `argument_fichier_t * creer_argument_fichier(char *)` ; qui alloue l'espace mémoire nécessaire pour une structure `argument_fichier_t`, initialise son champs `resultat` à 0 et **recopie** la chaîne de caractère passée en argument dans le champs `chemin`.
3. Écrire la fonction `void liberer_argument_fichier(argument_fichier_t *)` ; qui libère l'espace mémoire utilisé par une structure `argument_fichier_t`.

3 Recherche dans une chaîne de caractère

Écrire une fonction `void * recherche_texte(void *)` qui prend en entrée un pointeur sur une structure `argument_texte_t`.

La fonction renvoie 1 si le texte contient votre numéro d'étudiant et 0 sinon.

4 Attributs de thread

Écrire les fonctions `pthread_attr_t attributs_recherche_texte()` et `pthread_attr_t attributs_recherche_fichier()`.
La première renvoie des attributs de threads où la taille de la pile d'appel est initialisé `PTHREAD_STACK_MIN`.
La seconde renvoie des attributs de threads où la taille de la pile d'appel est initialisé à votre numéro d'étudiant.

5 Recherche dans un fichier

1. Écrire une fonction qui prend en entrée le chemin d'un fichier, une chaine de caractère (on suppose que l'allocation mémoire a déjà été faite), et la taille du fichier. La fonction recopie le contenu du fichier dans la chaîne de caractère.
2. Écrire une fonction qui prend en entrée un tableau de 10 structures `argument_texte_t`, une chaîne de caractères et la longueur de cette chaîne, que l'on nommera `taille`. La fonction initialise le tableau en utilisant la fonction `argument_texte_t creer_argument(char *, int ) ;`. Chaque `argument_texte_t` pointe vers un morceau de la chaîne de caractères, chaque morceau étant de taille `taille/10`, sauf le dernier dont la taille peut-être légèrement plus petite.
3. Écrire une fonction `void * recherche_fichier(void *) ;` qui reçoit en entrée un pointeur sur une structure `argument_fichier_t`. Cette fonction :
   — Crée une chaîne de caractère dans la pile d'appel, de taille 10 Mo et charge le contenu du fichier dans la chaîne de caractère à l'aide de la première fonction de cet exercice.
   — Découpe le contenu du fichier en 10 parts égales (sauf la dernière, dont la taille peut) et charge ces 10 parts dans un tableau de 10 structures `argument_texte_t` en utilisant la dexuième fonction de cet exercice.
   — Pour chaque part du fichier, la fonction y cherche **votre numéro d'étudiant** à l'aide d'un thread, dont la taille de la pile d'appel est égal à `PTHREAD_STACK_MIN` et utilisant la fonction `void * recherche_texte(void *)`.
   — Si l'un des threads renvoie 1, initialise le résultat de la structure `argument_fichier_t` à 1.
   — la fonction renvoie le pointeur qu'on lui avait passé en argument.

6 Programme principal

Écrire un programme qui reçoit en argument un répertoire à parcourir.

— pour chaque fichier, le programme appelle à l'aide un thread, dont la taille de la pile d'appel **est égale à votre numéro d'étudiant**, la fonction `void * recherche_fichier(void *) ;`.
— à la fin du programme, si votre numéro d'étudiant est apparu dans un fichier, le programme l'affiche en précisant le nom de ce fichier.
— (Bonus) : si un des fichiers passé en argument fait plus de 10 Mo, il est ignoré.

7 Impératifs pour le rendu

1. Vous devez rendre un unique fichier sur l'ENT, dont le nom est `NOM_PRENOM_NUMERO_ETUDIANT.c`.

2. Votre code sera commenté, comme vous avez appris à le faire dans la matière programmation orienté objet. On fera particulièrement attention aux paramètres `@param`, `@requires` et `@result` pour les fonctions `recherche_texte` et `recherche_fichier`
3. Un programme qui ne compile pas vaut 0, il vaut mieux rendre un programme qui ne répond pas à toutes les questions qu'un programme qui ne compile pas. N'hésitez donc pas à mettre les zones problématiques en commentaires.
4. Tout **soupçon** de fraude sera immédiatement sanctionné d'un 0 pour **l'ensemble** des personnes impliquées, voir d'un conseil de discipline. Toute justification du type "on en a discuté ensemble" ne sera pas admise. Discuter n'est pas interdit. Rendre des devoirs dont le code est identique l'est. Changer le nom des variables est un très mauvais moyen de camoufler une fraude.