Rapport du projet Cesar

Clément Caumes 21501810

30 avril 2017

1 Description des structures de données

1.1 Structure THREAD

Cette structure correspond au type de l'argument dans la fonction de travail des employés. Elle contient le numéro du rang de la première lettre du mot à chiffrer/déchiffrer dans le buffer. Il y a également un pointeur sur ce buffer ainsi que sur le pas et le sens de chiffrement. On a aussi le thread associé à cet élément THREAD, avec des pointeurs sur un mutex et une condition afin de synchroniser les threads lors du chiffrement/déchiffrement du buffer.

1.2 Structure ANALYSE

Cette structure contient les informations sur un message : le path du message, le pas et le sens de chiffrement, le pid du processus associé à ce message, le tube de communication (entre le chef d'équipe et le directeur). Il y a également le buffer qui sera alloué par le chef d'équipe ainsi que la liste de THREAD associés à chaque mot du buffer.

2 Implémentation générale

Le projet est subdivisé en trois procédures principales qui correspondent au travail des trois différents acteurs : processus_directeur, processus_chef_dequipe et threads_employes.

processus_directeur est la procédure du directeur (processus du programme ./cesar) qui va lire le fichier principal dans lequel se trouvent les informations des messages à chiffrer/déchiffrer.
Une liste est créée dans laquelle pour chaque élément il y a les informations pour chaque message. Le processus directeur va alors créer n processus fils pour les n messages. Ces processus fils vont faire la procédure de processus_chef_dequipe avec comme argument la structure ANALYSE contenant les informations du message attribué.

processus_chef_dequipe du message va alors stocker dans le buffer le message (lu grâce au path). Ensuite, ce processus va compter le nombre de mots dans le buffer. Pour chaque mot, un thread va être créé dans lequel on envoie comme argument un THREAD contenant le pointeur du buffer, le numéro de la case du début du mot attribué, les pointeurs du pas et du sens de chiffrement, ainsi que des pointeurs sur le mutex et la condition du message (utiles pour la synchronisation des threads).

threads_employes du mot va alors chiffrer/déchiffrer (selon le pas et le sens de chiffrement) chaque lettre du buffer à partir du rang auquel on l'a attribué, jusqu'à rencontrer un espace ou la fin de la chaîne de caractères de buffer. Tous les threads du message seront synchronisés (grâce à barriere). Lorsque chaque mot a été chiffré/déchiffré, on ferme les threads et on revient sur le processus_chef_dequipe du message. Si le message était un chiffrement alors le processus chef dequipe va créé un nouveau fichier-cypher dans lequel sera écrit le message-buffer chiffré. On utilise alors un tube anonyme qui va envoyer le buffer à son père (processus_directeur).
Quand on arrive dans le père (processus_directeur), si le message était à déchiffrer, alors on lit dans le tube anonyme le buffer (envoyé par son fils) puis on l'affiche dans le terminal.

3 Alternatives du projet

Pour le projet, nous n'avons pas utilisé de mutex pour la sécurisation du buffer. En effet, en vue de l'implémentation du projet, les threads manipulent directement le buffer grâce au pointeur sur celui-ci et chaque thread manipule un intervalle particulier. Du coup, il n'y a pas vraiment de variable critique.