Chapitre 2 : Fils d'exécution

2.1 Comment pourriez-vous évaluer ou mesurer la taille requise par un programme pour sa pile?

Statiquement, on peut regarder l'arbre d'appel (non récursif, sans pointeur de fonction) et la taille de pile prise par chaque fonction. Dynamiquement, on peut lire le pointeur de pile de temps en temps. Il est aussi possible de mettre un quota sur la pile et de le diminuer jusqu'à ce qu'il y ait un problème.

2.2 Vous avez un vieil ordinateur ayant appartenu à votre grand-père qui ne contient qu'un seul processeur et un seul disque. Vous voulez y effectuer la compilation d'un gros logiciel (e.g. noyau Linux). La compilation de chaque fichier demande un mélange de calcul et d'accès disque. Combien de fils d'exécution devriez-vous demander à parallel make? Combien de fils d'exécution demanderiez-vous pour un serveur Web?

L'idée est d'utiliser le CPU en parallèle avec le disque. L'un ou l'autre sera le goûlot d'étranglement et donc au mieux, avec deux fils, si la charge est égale CPU et disque, cela pourrait aller deux fois plus vite. Plus de fils ne feraient qu'encomber l'ordonnanceur et la mémoire cache. Pour un serveur Web, les attentes après le réseau et les clients sont importantes et il est donc intéressant d'avoir de nombreux fils, assez pour éviter qu'ils soient tous pris en attente, non disponibles pour servir des requêtes.

2.3 Deux fils d'exécution communiquent entre eux par un tuyau bi-directionnel. Le premier lit du disque et envoie au second une certaine quantité de donnée puis se met en attente de la réponse et l'écrit sur disque. Le second traite les données au fur et à mesure de leur réception et envoie le résultat aussi au fur et à mesure. Est-ce correct? Peut-on procéder autrement?

Le problème est que le premier ne lit rien avant la fin alors que le second écrit au fur et à mesure. Ceci causera problème à moins que le tampon associé au tuyau soit plus grand que la quantité de données à traiter. Le second bloquera rapidement en écriture et ne pourra plus lire, ce qui bloquera aussi le premier en écriture.

2.4 Un débogueur insère des points d'arrêt en remplaçant l'instruction cible par une instruction d'interruption (INT3). Une fois cet emplacement atteint, le programme est interrompu et le INT3 retiré. Pour poursuivre, le débogueur repart l'exécution en mode une instruction à la fois, replace le INT3 et repart l'exécution en mode normal. On vous demande de changer le débogueur pour fonctionner en mode non-stop, multi-fils: un point d'arrêt peut ne cibler qu'un seul fil d'exécution. Comment feriez-vous?

La difficulté serait d'enlever et remettre le INT3 alors que d'autres fils continuent leur exécution et doivent ou non s'arrêter sur cette instruction. La solution est de recopier et d'exécuter l'instruction remplacée par le INT3 ailleurs (out of line) et de compenser pour sa position modifiée (e.g. saut relatif au compteur de programme).