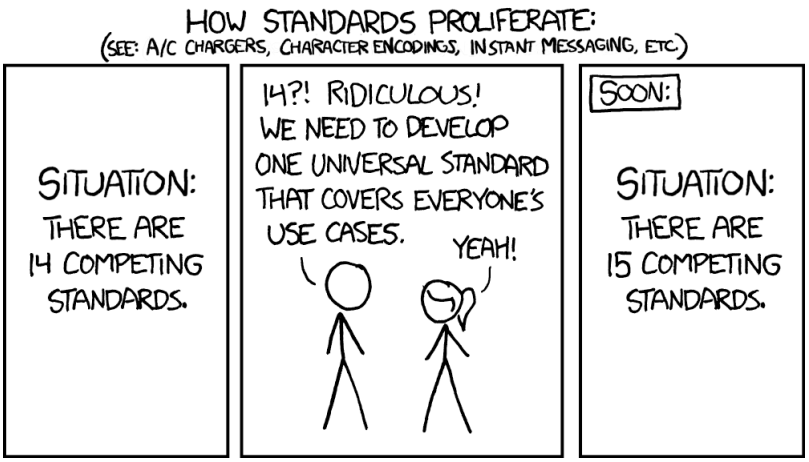
MPI* Info

Lecteurs/rédacteurs

- TD14 -



Olivier Caffier

Question 1 Une erreur s'est glissée dans la fonction `rwlock_init`. L'identifier et la corriger.

Corrigé : Dans l'attribution `nb_readers = 0`, il faut remplacer par `l->nb_readers = 0` car sinon on ne modifie pas la structure.

Question 2 Aurait-on pu utiliser un autre type de variable à la place du sémaphore `write_lock`?

Corrigé : On ne peut pas utiliser le type `mutex` car la personne qui ouvre en écriture n'est pas la personne qui lit. (tous les lecteurs n'ont pas la mission de verrouiller).

Question 3 Expliquez, en français, les rôles des lignes 4 et 5 ainsi que 12 et 13.

Corrigé :

- Les lignes 4 et 5 permettent au *premier lecteur* (i.e celui qui lit le doc. en premier) de "verrouiller" le doc, le protéger des rédacteurs et permettre aux autres lecteurs de lire le doc sans problème.
- En ce sens, les lignes 12 et 13 permettent au dernier lecteur de "déverrouiller" le doc. vis-à-vis des rédacteurs.

Si le problème ne présentait que des rédacteurs, on utiliserait un verrou.

Question 4 Expliquez pourquoi deux rédacteurs ne pourront pas écrire simultanément.

Corrigé : La fonction qui permet de verrouiller l'accès s'exécute en une opération, elle bloque tout jusqu'à ce que ce dernier déverrouille. Ainsi, deux rédacteurs ne pourront pas écrire simultanément.

Question 5 Pourquoi est-ce que tous les agents (lecteurs ET rédacteurs) utilisent le même sémaphore?

Corrigé : On utilise le même sémaphore sinon il n'y a pas de corrélation entre l'idée de bloquer le fichier et de débloquer (chez les lecteurs et rédacteurs).

Question 6 Quel problème de concurrence apparait dans le code proposé? Expliquer comment modifier le code pour résoudre ce problème.

Corrigé : La variable `nb_readers` n'est pas du tout protégée, elle peut posséder une valeur > 1 si deux lecteurs incrémentent etc... Pour corriger ce problème, on rajoute un `mutex m` en rajoutant aux lignes

— 2 et 10 : `lock(m)`
— 6 et 14 : `unlock(m)`

Question 7 On considère maintenant les fonctions `reader` et `writer` (cf. énoncé). Que représenter la variable `counter`?

Corrigé : La variable `counter` joue ici le "rôle" de fichier, les lecteurs lisent sa valeur et les rédacteurs l'incrémentent.

Question 8 Recopier et compléter le `main` pour lancer les fonctions précédentes.

Corrigé : On a le code suivant :

```
1  int main(int argc, char* argv[]){
2    if (argc != 3){
3      fprintf(stderr, "usage: rwlock readloops writeloops \n");
4      exit(1);
5    }
6    read_loops = atoi(argv[1]);
7    write_loops = atoi(argv[2]);
8
9    rwlock_init(&mutex);
10   pthread_t p1,p2;
11   pthread_create(&p1,NULL,writer, NULL);
12   pthread_create(&p2,NULL,reader, NULL);
13   pthread_join(p1,NULL);
14   pthread_join(p2,NULL);
15   return 0;
16 }
```