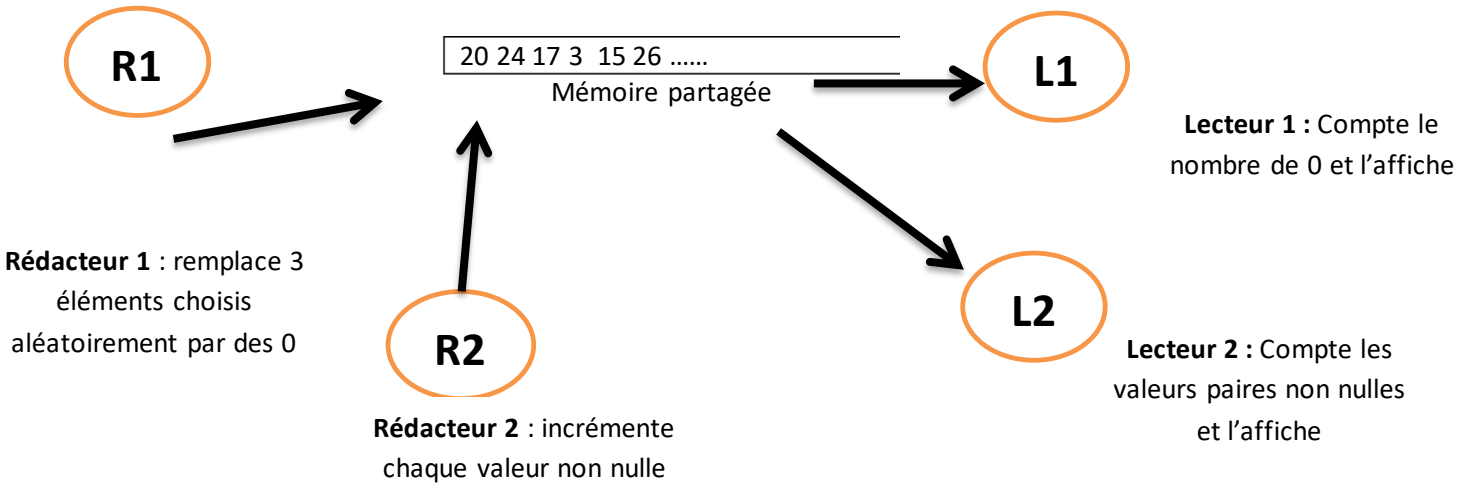
TP : LECTEURS - REDACTEURS



Objectif : **Simuler le modèle Lecteur-Rédacteur (2 Lecteurs et 2 rédacteurs)**

On désire simuler le modèle Lecteur-Rédacteur. En partant d'une **mémoire partagée** contenant un tableau de 50 valeurs aléatoires comprises entre 1 et 50. Le 1er rédacteur remplace 3 éléments d'une façon aléatoires par des 0. A chaque modification il s'endort entre 1 à 3 secondes. A la fin de son temps de sommeil, il doit afficher un message montrant sa relance. Le 2ème rédacteur incrémente chaque valeur non nulle. Comme pour le 1er rédacteur, à chaque modification il s'endort entre 1 à 3 secondes. A la fin de son temps de sommeil, il doit afficher un message montrant sa relance.

Par ailleurs, 3 processus Lecteurs tentent de lire le contenu de la mémoire partagée. Le premier lecteur compte le nombre de 0 du tableau et l'affiche et le 2ème lecteur compte toutes les valeurs paires non nulles et l'affiche.

A chaque lecture, les processus s'endorment entre 1 et 2 secondes.

Afin de vous aider dans la programmation de ce modèle « Lecteur-Rédacteur » voici globalement l'algorithme :

NL : nombre de lecteurs (**mémoire partagée**) initialisé au nombre de lecteurs

Mutex_NL : sémaphore d'exclusion mutuelle pour l'accès à NL

Mutex_A : sémaphore d'exclusion mutuelle entre les rédacteurs entre eux ainsi qu'entre les lecteurs et les rédacteurs. C'est-à-dire :

- S'il y a une écriture, aucune autre écriture ne pourra se faire.
- S'il y a au moins une lecture, aucune écriture ne pourra se faire.

LECTEUR	REDACTEUR
while(1) { **//début lecture** P(mutex_NL); NL=NL+1; if (NL==1) P(mutex_A);**// Se bloque en cas où il y a un autre lecteur ou un rédacteur** V(mutex_NL); ***lectures***; **//fin lecture** P(mutex_NL); NL=NL-1; If(NL==0) V(mutex_A);**// débloque un rédacteur ou un lecteur** V(mutex_NL); }	while(1) { **//début écriture** P(mutex_A); ***écritures***; **//fin écriture** V(mutex_A); }

➔ ***Ecrire le programme avec 4 processus qui réalisent les opérations des 2 lecteurs et des 2 rédacteurs présentés ci-dessus en respectant la synchronisation décrite dans l'algorithme.***

➔ ***Prévoir un handler qui permet de capturer le signal SIGINT afin de détruire les sémaphores et supprimer les mémoires partagées avant de quitter le programme.***

➔ Rappel :

- srand(getpid()) ; //initialise le générateur de valeurs aléatoires
- rand()%N ; //renvoie une valeur comprise entre 0 et N non comprise