# L3 MIAGE: Programmation système sous UNIX
# Projet

Erwan Lenormand

2019-2020

## 1 Préambule

Vous devez rendre le projet avant le début de l'examen prévu le **30 janvier 2020**. Vous pouvez réaliser ce travail soit seul, soit en binôme. Un seul groupe peut constituer un trinôme en cas de besoin. Vous devez soumettre une archive au format **.tar** (tar -rvf votre_archive.tar votre_dossier/ ). L'archive doit contenir un répertoire nommé par votre (vos) nom(s) et prénom(s) tel que NOM1_PRENOM1_NOM2_PRENOM2 et le répertoire doit contenir **uniquement** le(s) fichier(s) source(s) (.c et .h) et un Makefile. Le Makefile doit permettre de générer l'exécutable (ou les exécutables) avec la cible *all* et doit supprimer tous les fichiers générés avec la cible *clean*. Votre code doit respecter les règles vues lors du premier cours (indentation, utilisation de macros, gestion des erreurs ...).

## 2 Sujet

Vous devez développez une application qui va comparer deux fichiers et afficher si ils sont identiques ou si ils sont différents (comme la commande **diff**). L'application sera composée de trois processus. Vous pouvez soit développer un seul exécutable qui va créer deux autres processus dynamiquement, soit développer trois exécutables. En fonction de votre choix vous utiliserez des tubes anonymes ou des tubes nommés pour communiquer entre les processus. Dans tous les cas l'application doit être interactive, les chemins vers les fichiers seront transmis par l'entrée STDIN. Vous utiliserez vos connaissances sur les signaux et les tubes pour synchroniser et faire communiquer les processus.

Voici le rôle de chaque processus :

— **Le processus de contrôle :** Il est en charge de créer les tubes, de créer les processus (en fonction de votre choix) et d'initialiser des objets pouvant être communs entres les processus. Ce processus est également responsable de propager les erreurs d'un processus à l'ensemble des processus. Si une erreur provoquent la terminaison d'un processus, le processus de contrôle doit tuer tous les autres processus. Enfin il doit s'assurer de la bonne terminaison de l'application (synchronisation, désallocation des ressources).

— **Le processus d'entrées/sorties :** Ce processus demande à l'utilisateur de saisir le chemin vers un premier fichier, puis transmet la chaîne de caractères saisie au processus de traitement. Il répète l'opération en demandant la saisie d'un chemin vers un second fichier. Puis il se met en attente de la réponse du processus de traitement. Lorsqu'il reçoit la réponse, il transmet le résultat à l'utilisateur via la sortie standard.

— **Le processus de traitement :** Le processus attend la réception de la première chaîne de caractères et ouvre le fichier. Si il y a une erreur il se termine en renvoyant un code erreur. Il répète l'opération pour le second fichier. Puis il crée trois threads qui seront en charge de l'analyse des

fichiers. Deux threads ont le rôle de producteur et un thread a le rôle de consommateur. Chaque thread producteur lit ligne par ligne un des deux fichiers. Pour chaque ligne le producteur doit positionner les données lues dans un tampon accessible au thread consommateur et indiquer qu'une nouvelle ligne est disponible. Le consommateur doit attendre que les deux tampons contiennent des nouvelles données avant des les consommer, puis doit indiquer que les données sont consommer. Il est particulièrement important d'utiliser les mécanismes de synchronisation pour protéger l'accès aux ressources partagées. Le thread consommateur utilise la fonction **strcmp** pour comparer les deux lignes lues. Si une ligne diffère, le thread consommateur doit se terminer et les threads producteurs doivent être arrêtés. Le processus de traitement doit indiquer au processus d'entrées/sorties le résultat de la comparaison. Si les fichiers sont différents le numéro de la ligne où figure la première différence doit être communiquée.