

# Manuel d'utilisateur :

## 1. Manuel d'utilisateur :

1. I. Organisation des bibliothèques système
2. II. API du système
3. III. Description de la plateforme
4. IV. L'Organisation des fichiers systèmes
5. V. La génération d'une application

## I. Organisation des bibliothèques système

La figure suivante illustre la modalisation du système :



Les bibliothèques constituant le système sont :

- pthread : contient l'implémentation d'un sous-ensemble des thread POSIX.
- libc : contient l'implémentation des services système tel que malloc, printf, pipe, memcpy ..etc.
- mwmr : contient l'implémentation du protocole MWMR.
- sys : contient le code système qui ne dépend pas de l'architecture de la plate-forme ou de type des processeurs utilisés.
- cpu : contient le code système en assembleur qui dépend de type des processeurs de la plate-forme.
- arch : contient le code C qui dépend de la plate-forme tel que des configurations système vis-à-vis des composants de la plate-forme, les ISR d'interruption des différents types de cibles ..etc.

Les bibliothèques pthread, libc et mwmr sont indépendantes de la spécification de la plate-forme.

En cas de modification au niveau de la configuration matériel, il suffit d'adapter le code système des deux bibliothèques cpu et arch pour pouvoir déployer MUTEKP sur la nouvelle plate-forme.

## II. API du système

MUTEKP fournit trois bibliothèques pour les threads de l'application :

- Un sous-ensemble de l'API des threads POSIX :
  - ◆ pthread\_attr\_init: initialiser la structure d'un attribut.
  - ◆ pthread\_create: créer une tâche
  - ◆ pthread\_exit: mettre fin à une tâche avec une valeur de retour
  - ◆ pthread\_self: Donner l'identifiant du thread appelant.
  - ◆ pthread\_equal: tester l'égalité entre deux identificateurs.
  - ◆ pthread\_yield: céder le processeur pour un autre thread.
  - ◆ pthread\_join: attendre la fin d'un thread.
  - ◆ pthread\_spin\_init: initialiser un verrou à une attente active.
  - ◆ pthread\_spin\_destroy: détruire un verrou.
  - ◆ pthread\_spin\_lock: verrouiller le verrou à une attente active.
  - ◆ pthread\_spin\_trylock: version non bloquante de pthread\_spin\_lock.
  - ◆ pthread\_spin\_unlock: déverrouiller le verrou à une attente active.
- Les fonctions implémentant le protocole MWMR :
  - ◆ mwmr\_read: lecture d'un FIFO MWMR.
  - ◆ mwmr\_write: écriture dans un FIFO MWMR.
  - ◆ mwmr\_init: création et initialisation d'un FIFO MWMR.

- Quelques fonctions de la bibliothèque libC :
  - ◆ printf?: afficher une chaîne de caractère formatée.
  - ◆ malloc?: allocation de mémoire dynamique.
  - ◆ pipe?: créer un tube de communication (par flux d'octets) entre deux threads.
  - ◆ read?: pour lire d'un buffer ou un tube.
  - ◆ write?: pour écrire dans un buffer ou un tube.
  - ◆ memset?: remplir une zone mémoire par une valeur donnée.
  - ◆ memcpy?: copie une zone mémoire source vers une autre zone mémoire destination.

La norme POSIX ne propose pas dans son API pthread aucun appel permettant d'affecter ou de spécifier un thread à un processeur donné.

MUTEKP propose l'appel `pthread_attr_setprocid_np` qui permet d'affecter un numéro de processeur à un attribut. Cela permettra de préciser sur quel processeur le nouveau thread va-t-il s'exécuter.

### **III. Description de la plateforme**

### **IV. L'Organisation des fichiers systèmes**

### **V. La génération d'une application**