# Manuel de développement :



# I. Architecture du MUTEKP

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, read, 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 différant types de cibles ..etc.

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. Introduction au noyau MUTEKP

## II.1 Le concept d?un thread dans le système

Un Thread est un fil d?exécution d?un programme.
Tous les Threads de l?application partagent le même espace d?adressage, où chaque Thread possède :

- Son propre contexte d?exécution (le PC, un pointeur de pile et d?autres registres de travail du processeur).

- Deux piles.
    - ♦ Plie utilisateur
    - ♦ Pile système

Quelques avantages :

- Création et gestion plus rapide (vs processus).
- Partage des ressources par défaut.
- Communication entre les threads plus simple via la mémoire (les variables globales).
- Déploiement plus efficace de l?application sur des architectures multi-processeurs.

## II.2 États d?un thread

La duré de vie d?un Thread peut être divisée en un ensemble d?états.
Les différents états d?un Thread sont :

- Run : quand le Thread s?exécute sur le processeur en faisant son calcul.
- Kernel : quand le thread « tombe » dans le noyau suite à un appel aux services noyau ou une interruption matérielle.
- Wait : lors que le thread demande une ressource qui n?est pas disponible.
- Ready : quand le thread est en attente de gagner le processeur pour poursuivre son exécution.
- Zombie : quand le thread se termine en attendant que un autre thread pren acte de sa terminaison.
- Create : état spécial dans le quel le Thread vient d?être crée. Il n?a pas encore été chargé sur un processeur et attend de le gainer pour la première fois.
- Dead : état spécial dans le quel le Thread est déclarer définitivement mort. Toute tentative de joindre un Thread dans cet état échouera.

## II.3 Ordonnancement des threads

## II.4 Organisation et gestion de la mémoire

### II.4.1 L?organisation mémoire

### II.4.2 La gestion mémoire

## II.6 Le buffer système

# III. Détaille du noyau et les structures de données système

## III.1 Gestion des threads

### III.1.1 La structure de donnée du thread

### III.1.2 La structure de donnée Ordonnanceur et la table d?ordonnancement

## III.2 Gestion de la mémoire

### III.2.1 La structure gestionnaire mémoire

## III.3 Gestion des périphériques

### III.3.1 La structure gestionnaire des verrous

### III.3.2 Représentation des cibles (TTY, Timer et ICU)

## III.4 Gestion des interruptions