

Manuel de l'utilisateur

1. Manuel de l'utilisateur
 1. Organisation des bibliothèques système
 2. API du système
 3. Description de la plateforme

1. Organisation des bibliothèques système

La figure suivante illustre la modalisation du système :



Les bibliothèques constituant le système sont :

- `pthread` : contient l'interface système des Threads POSIX.
- `libc` : contient l'interface des services système tel que `malloc`, `printf`, `read`, `memcpy` ..etc.
- `mwmr` : contient l'interface système des FIFO 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 de processeurs de la plate-forme.
- `arch` : contient le code C qui dépend de la plate-forme, tel que les configurations système vis-à-vis des composants de la plate-forme, les ISR d'interruption des différents types de cibles?, etc.

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

2. 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`: initialise la structure d'un attribut.
 - ◆ `pthread_create`: crée un Thread
 - ◆ `pthread_exit`: met fin à une tâche avec une valeur de retour
 - ◆ `pthread_self`: récupère l'identité du Thread appelant.
 - ◆ `pthread_equal`: teste l'égalité entre deux identificateurs.
 - ◆ `pthread_yield`: cède le processeur pour un autre Thread.
 - ◆ `pthread_join`: attend la fin d'un Thread.
 - ◆ `pthread_spin_init`: initialise un verrou à une attente active.
 - ◆ `pthread_spin_destroy`: détruit un verrou.
 - ◆ `pthread_spin_lock`: verrouille le verrou à une attente active.
 - ◆ `pthread_spin_trylock`: version non bloquante de `pthread_spin_lock`.
 - ◆ `pthread_spin_unlock`: déverrouille le verrou à une attente active.
- Les fonctions implémentant le protocole MWMR :
 - ◆ `mwmr_read`: lit une FIFO MWMR.
 - ◆ `mwmr_write`: écrit dans une FIFO MWMR.
 - ◆ `mwmr_init`: crée et initialise une FIFO MWMR.
- Quelques fonctions de la bibliothèque `libc` :
 - ◆ `printf`: affiche une chaîne de caractères formatée sur le terminal utilisateur (`tty1`).

- ◆ `sprintf?`: écrit une chaîne de caractères formatée dans un buffer.
 - ◆ `fprintf?`: affiche une chaîne de caractères formatée sur un terminal donné.
 - ◆ `puts?`: affiche une chaîne de caractères non formatée sur le terminal utilisateur (tty1).
 - ◆ `strlen?`: calcule la longueur d'une chaîne de caractères.
 - ◆ `malloc?`: alloue de mémoire dynamique.
 - ◆ `read?`: lit un nombre fixe d'octets à partir du buffer système.
 - ◆ `write?`: écrit le contenu d'un buffer sur un terminal.
 - ◆ `memset?`: remplit une zone mémoire par une valeur donnée.
 - ◆ `memcpy?`: copie une zone mémoire source vers une autre zone mémoire destination.
- Appels propres à l'implémentation Mutekp :
 - ◆ `pthread_attr_setprocid_np` : permet d'affecter un numéro de processeur à un attribut.
 - ◆ `pthread_profiling_np?` : permet d'afficher, sur le terminal système (tty0), des statistiques sur le déroulement de l'application et la réactivité du système.

3. Description de la plateforme

La figure suivante illustre la plateforme matérielle



Composants de la plateforme :

- Type MPSoCs 1 seul cluster
 - ◆ 4 processeurs de type MIPS R3000 avec cache Instruction et Data (16 lignes / 8 mots).
- Un interconnect VCI (Virtual Component Interconnect)
- Une RAM multi segments
 - ◆ 8 segments
 - ◆ taille de 256 octets à 64ko
- Un contrôleur TTY
 - ◆ 4 terminaux
- Un contrôleur de timers programmables.
 - ◆ 5 timers
- Un ICU : concentrateur d'interruptions
 - ◆ relié au processeur 0 pour 1 timer et les TTY
- Un contrôleur de verrous matériels