Mutek*p*

Bienvenue sur le site du projet.
Mutek*P* est un noyau de système d?exploitation pour des architectures multiprocesseur intégrées sur puce de type MPSoCs.

Muni d'un code claire et facile à lire, Mutek*P* a été crée dans le but de disposer d?un noyau embarqué à vocation pédagogique
implémentant l?API des threads POSIX sur des architectures MPSoCs en mettant en avant les concepts système utilisés.
Il servira comme support pédagogique dans différents modules du MASTER ACSI.

(Ce site est en construction)

Manuel d'utilisation

UserManuel : tout ce qu'il faut savoir pour pouvoir programmer des applications au dessus de Mutek*p*

(this Site is under construction)

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.
- 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.

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 fourni trois bibliothèques pour les threads de l?application :

- Un sous-ensemble de l'API des threads POSIX :
  - pthread_attr_init: initier 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?indenté du thread appelant.
  - pthread_equal?: tester l?égalité entre deux identificateur.
  - 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) enter 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 moire source vers une autre zone mémoire distination.

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. L?Organisation des fichiers systèmes

IV. La génération d?une application

V. Description de la plateforme

Manuel de développement

I. La plate-forme matérielle

II. Architecture du MUTEKP

III. Introduction au noyau MUTEKP

III.1 Le concept d?un thread dans le système

III.2 États d?un thread

III.3 Les transitions d?états

III.4 Ordonnancement des threads

III.5 Organisation et gestion de la mémoire

III.5.1 L?organisation mémoire

III.5.2 La gestion mémoire

III.6 Le buffer système

IV. Détaille du noyau et les structures de données système

IV.1 Gestion des threads

IV.1.1 La structure de donnée du thread

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

IV.2 Gestion de la mémoire

IV.2.1 La structure gestionnaire mémoire

IV.3 Gestion des périphériques

IV.3.1 La structure gestionnaire des verrous

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

IV.4 Gestion des interruptions