

Manuel de développement :

1. Manuel de développement :

1. I. Architecture du MUTEKP

2. II. Introduction au noyau MUTEKP

1. II.1 Le concept d'un thread dans le système

2. II.2 États d'un thread

3. II.3 Les transitions d'états

4. II.4 Ordonnancement des threads

5. II.5 Organisation et gestion de la mémoire

1. II.5.1 L'organisation mémoire

2. II.5.2 La gestion mémoire

6. II.6 Le buffer système

3. III. Détail du noyau et les structures de données système

1. III.1 Gestion des threads

1. III.1.1 La structure de donnée du thread

2. III.1.2 La structure de donnée Ordonnanceur et la table d'ordonnement

2. III.2 Gestion de la mémoire

1. III.2.1 La structure gestionnaire mémoire

3. III.3 Gestion des périphériques

1. III.3.1 La structure gestionnaire des verrous

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

4. III.4 Gestion des interruptions

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 différents 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.
 - ◆ Pile 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



II.3 Les transitions d'états

II.4 Ordonnancement des threads

II.5 Organisation et gestion de la mémoire

II.5.1 L'organisation mémoire

II.5.2 La gestion mémoire

II.6 Le buffer système

III. Détail 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 Ordonnancement 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