This document explains the memory allocation mechanisms which can be meet in MutekH.

Introduction

Memory allocation in an operating system is a large spectral domain...

Glossaire

Pour enlever toutes ambiguïtés, une liste de définition de mot technique est données.

Allocateur mémoire: On appel allocateur mémoire l'additions de deux éléments. D'une part le code executé lors d'un appel à une fonction du type malloc. Et d'autre part l'ensemble des structures de données représentant une zones mémoire allouables.

Utilisateur: L'utilisateur est le code/programe qui demande de la mémoire. Par exemple, un driver qui fait une demande de mémoire au service d'allocation mémoire est l'utilisateur de ce service.

Demande de mémoire: Correspond à l'appel d'une fonction du type malloc (la fonction ne s'appel pas forcément malloc).

Libération de mémoire: Correspond à l'appel d'une fonction du type free (la fonction ne s'appel pas forcément free).

Kernel land/User land

Kernel memory managment

Cette section explique comment le noyau gère son espace mémoire.

Region based memory allocation

Algorithm use for memory allocation

Dans une région, la mémoire est divisée en bloc. Un bloc est divisé en deux partie: -un header, contenant les informations utilisées par l'allocateur mémoire, de taille fixe. -les "données", correspondant à l'espace destiné à l'utilisateur, de taille variable.

Un bloc peut être dans un des deux états suivant: libre ou occupé. Un bloc occupé représente une allocation mémoire en cours d'utilisation. Alors qu'un bloc libre représente une zone allouable.

Schéma bloc libre/occupé

Lorsqu'un utilisateur demande de la mémoire, l'allocateur cherche un bloc dans l'état libre, dont la taille de la partie "données" est au moins égal à la taille demandé. Dans le cas d'une recherche positive, le bloc libre trouvé est utilisé pour servir la demande d'allocation mémoire. Son état passe de libre à occupé. L'adresse de début de la partie "données" est fournie à l'utilisateur.

Dans le cas où ce bloc libre possède une partie "données" de taille supérieur à la taille demandé par l'utilisateur, et que le surplus peut contenir un bloc, alors le bloc libre est scindé en deux blocs. Le premier bloc, dont la taille de la partie données est égale à la taille demandé par l'utilisateur, est dans mis dans l'état occupé, et sa partie "données" est fournie à l'utilisateur. Le second bloc est dans l'état libre, et occupe le surplus de mémoire du bloc initial.

Schéma Passage bloc libre->occupé, avec deux cas (pas scindé et scindé)

La technique utilisé par l'allocateur mémoire pour trouver un bloc libre est maintenant détaillé.

Un bloc est représenté par la structure de donnée suivante:

```
struct memory_allocator_header_s
{
  union
  {
    CONTAINER_ENTRY_TYPE(CLIST) free_entry;
    struct
    {
      struct memory_allocator_region_s  *region;
      void *free_null_marker;
    };
  };

  CONTAINER_ENTRY_TYPE(CLIST)   block_entry;

#ifdef CONFIG_MUTEK_MEMALLOC_CRC
  uint32_t crc;
#endif
};
```

Standard API (Malloc/Free?)

API overview

Virtual memory allocation

Physical page allocation

Todo

slab allocator debug memory allocator