Prova II - Apresentação

PCS 3732

Eduardo Hiroshi Ito NºUSP: 11806868

Gabriel Coutinho Ribeiro NºUSP: 11803437

Ricardo Tamay Honda NºUSP: 11803778

Malloc-eiro

Gerenciador de memória dinâmica

O que é?

- Gerenciador de memória dinâmica

- Aloca e libera páginas na região de memória heap

- Opera em nível do sistema operacional
  - Lida diretamente com endereços físicos
  - Semelhante ao gerenciador de memória dinâmica interno do Linux
  - Normalmente, seria usado somente por outros programas em nível de sistema operacional, mas, no contexto o projeto, pode ser utilizado por qualquer programa

Algoritmo para o gerenciador

- Existem inúmeros algoritmos diferentes para lidar com memória dinâmica
  - **Chunk Allocator** - usado pela newlib
  - **Slab allocator** - capaz de alocar tamanhos menores que uma página
  - **Buddy allocator** - usado internamente pelo Linux

- No projeto, foi escolhido o buddy allocator, por conta de sua simplicidade

Buddy allocator, o cerne do projeto

- Conceitos básicos
  - Memória dividida em blocos de tamanho 2^k páginas, diz-se , então, que o bloco é de ordem k
  - Se um bloco do tamanho desejado não está disponível, um bloco maior é dividido na metade e os dois novos blocos são *buddies* um do outro
  - Uma metade é usada para a alocação e a outra fica livre
  - Os blocos são continuamente divididos na metade até que um bloco do tamanho desejado esteja disponível
  - Quando um bloco é liberado, seu *buddy* é examinado e, se também estiver livre, os dois são unidos

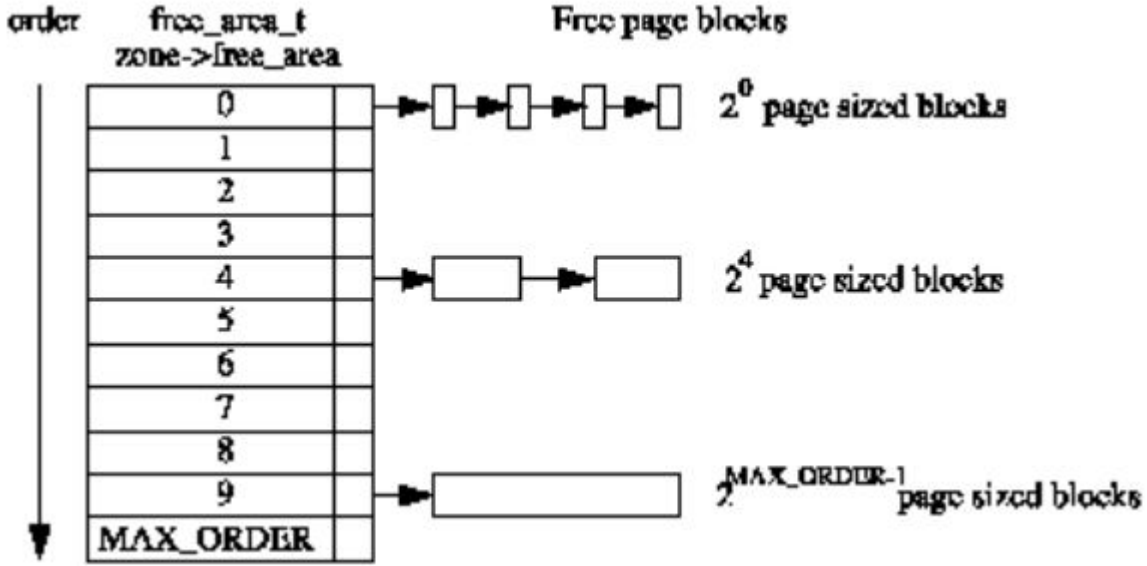
Gerenciamento dos blocos livres

- Um vetor de estruturas `free_area_t` são mantidos para cada ordem
- Cada estrutura contém uma lista de blocos (lista ligada) de páginas livres da mesma ordem
- Definição da estrutura `free_area_t`:

```
typedef struct free_area_struct {
        struct list_head        free_list;
        unsigned long           *map;
} free_area_t;
```

- **free_list** é a lista ligada de blocos
- **map** é um mapa de bits representando o estado de um par de *buddies*

Gerenciamento dos blocos livres



Gerenciamento dos blocos livres

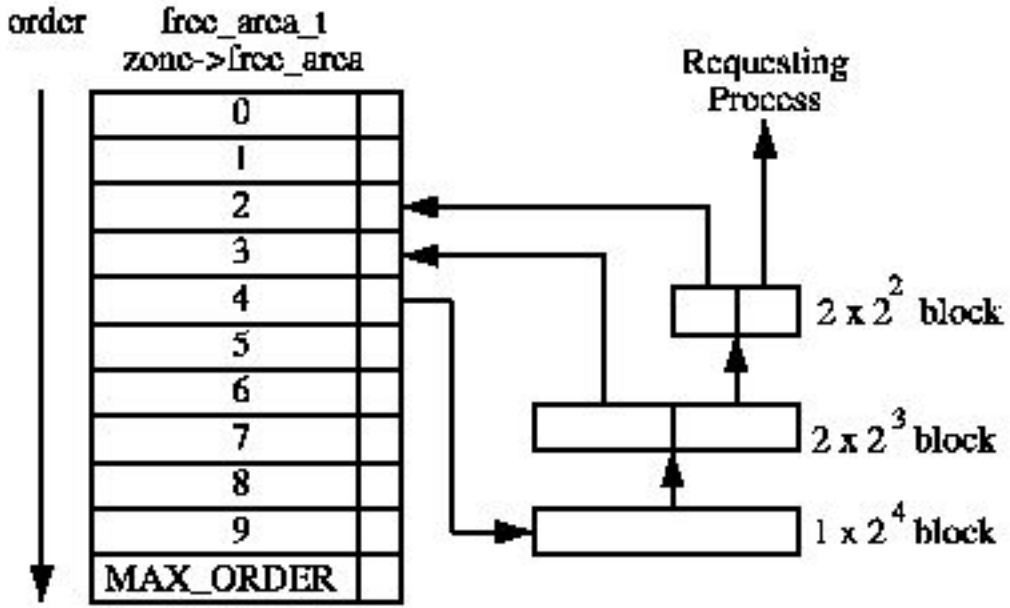
- Quanto ao mapa de bits
  - Um bit representa o estado de um par de *buddies*
  - Cada vez que um *buddy* é alocado ou liberado, o bit representando o par é setado
    - com valor 0 se ambos os blocos estão livres ou alocados
    - com 1 se somente um deles está alocado

Alocando blocos

- Alocações são sempre para uma ordem específica

- Se um bloco da ordem requisitado não é encontrado, um bloco de maior ordem é dividido em dois *buddies*
  - Um deles é alocado e o outro é posto na lista de blocos livres para a sua ordem

Alocando blocos



Alocando blocos

- Quando um bloco é liberado posteriormente, seu *buddy* é checado
  - Se ambos estão livres, são unidos em um bloco de ordem mais alta
    - O *buddy* do novo bloco é checado, repetindo o processo
  - Se o *buddy* não está livre, o bloco liberado é adicionado à lista de livres da sua ordem

Liberando blocos

- Uma desvantagem do *buddy allocator* é que o gerenciador deve lembrar do tamanho original da alocação
  - No projeto isso é feito reservando a primeira palavra do bloco para a ordem dele ao aloca-lo
- Quando um bloco é liberado, tenta-se uni-lo com seu *buddy*
  - Para detectar se eles podem ser unidos, o bit que os representa no mapa de bits é checado
  - Como um dos blocos foi liberado, é sabido que ao menos um *buddy* está livre
  - Se, após inverter o bit no mapa de bits, seu valor é
    - 0, é sabido que o outro *buddy* também está livre e eles podem ser unidos
    - 1, o outro *buddy* está alocado e o bloco vai para a lista de blocos livres de sua ordem
  - Quando os *buddies* são unidos, o processo se repete para o novo bloco

Disponível em:

github.com/GabrielCout/Projeto-de-Laboratorio-de-Processadores