Relatório Trabalho 1 Software Básico

IMPLEMENTAÇÃO DE UM SISTEMA DE GERENCIAMENTO DE ALOCAÇÃO DINÂMICA DE MEMÓRIA (HEAP)

Pedro Amaral Chapelin - GRR20206145
Arthur dos Santos Lima Gropp - GRR20211769

Para este trabalho, criamos uma API denominada “memalloc.s”, a qual é capaz de realizar operações de gerenciamento dos blocos de memória na seção “heap”.

A primeira operação que a API em questão realiza é denominada “setup_brk”, cuja função é determinar o início da seção “heap”. Para isso, simplesmente realizamos uma chamada ao sistema, visando receber em “%rax” o valor do “brk”, e salvamos o endereço retornado em “TOPO_HEAP”. Como essa função é utilizada antes de a “heap” sofrer qualquer alteração (desconsiderando a que ocorre em função do “printf”), podemos garantir que “TOPO_HEAP” guardará o endereço do início desta seçâo, tendo em vista que a “heap” cresce para baixo.

A função “get_brk”, em contrapartida, apesar de similar com a “setup_brk” possui uma única diferença: após encontrar o valor do “brk” atual, não salva esse valor em nenhuma variável. Esta função é utilizada apenas internamente, para que saibamos até onde a “heap” vai.

Ademais, a API possui a função “dismiss_brk”, que basicamente restaura o valor do “brk” por meio de uma chamada ao sistema, onde o argumento passado é o próprio endereço do topo da “heap”.

A cereja do bolo está na função “memory_alloc”, responsável por realizar a alocação de um determinado espaço na memória. A maneira como faz isso é a seguinte: inicialmente guarda em “%r15” o tamanho requisitado para a alocação, em “%r12” o endereço do “brk” atual (retornado pela função “get_brk”), e em “%rbx” o endereço do início da “heap” (armazenado em “TOPO_HEAP”). A partir disso entra em um “loop”, onde verifica se está no fim da seção “heap”, se o bloco em que se encontra está ou não ocupado, e se o tamanho é o suficiente para que a alocação ocorra com sucesso. Nesse sentido, caso se encontre no fim da seção heap, realiza um “jump” para fora do loop, onde modifica o valor da “brk” abrindo o espaço necessário para um novo bloco capaz de armazenar o tamanho de bytes requisitados. Caso encontre um bloco ocupado, simplesmente aponta para o próximo. Em relação ao tamanho do bloco encontrado, caso seja igual ao requisitado, simplesmente o marca como ocupado. Caso seja maior que o requisitado, verifica se existe a possibilidade de criar um novo bloco advindo da diferença entre ambos. Caso seja menor que o requisitado, verifica se o bloco imediatamente a frente está disponível para que possa se unir ao atual, e em seguida realiza um novo teste de tamanho. Ao fim de tudo, retorna o endereço referente ao início do bloco de dados.

Por fim, a API possui a função “memory_free”, responsável por desalocar o bloco cujo endereço é passado por parâmetro. Após verificar se o endereço realmente pertence à seção “heap”, marca o bloco como livre e retorna “1” indicando sucesso. Caso fracasse, retorna “0”.

O código responsável por implementar e testar as funções da nossa biblioteca é o que nos foi disponibilizado pelo professor, assim como o “makefile”. A API possui, também, um arquivo de cabeçalho “memalloc.h”, onde podem ser encontradas descrições breves sobre a funcionalidade de cada função.