Relatório Trabalho Prático

Software Básico
Heloisa Benedet Mendes
GRR20221248

Este trabalho foi feito individualmente.
A implementação das funções exigidas pelo trabalho se deu da seguinte forma:

setup_brk

void setup_brk();

Esta função realiza a syscall para brk(0), ou seja, obtém o endereço do fim da Heap, e atualiza os ponteiros original_brk e current_brk para este endereço.

dismiss_brk

void dismiss_brk();

Alinha o valor de current_brk com original_brk, para depois fazer a syscall para brk(original_brk), o que alinha o fim da Heap com os ponteiros.

memory_alloc

void memory_alloc(unsigned long int bytes);*

Primeiro, o registrador %rdx, que será usado nesta função como um ponteiro para percorrer os endereços da Heap, é alinhado com o início da Heap e o parâmetro da função passado por meio do registrador %rdi é armazenado em %rbx.
Depois é iniciado um laço for, cuja condição é o %rdx apontar para uma posição válida de memória na Heap, existem 3 possibilidades neste laço:

1. Se o %rdx aponta para o fim da Heap, significa que não existe um bloco livre adequado ao requerimento da função, assim, um novo bloco deve ser alocado ao fim da Heap e o programa é desviado.
   Nesta nova seção, é alocado o número de bytes requerido mais 16 bytes, destinados ao indicador de uso e ao tamanho, essa alocação é feita usando a syscall para brk(tamanho + 16).
   O bloco criado é marcado como ocupado, current_brk é atualizado para o fim do bloco e o início do bloco editável é definido como retorno da função, feito isso o programa termina.
2. O bloco para o qual %rdx está apontando está livre, o programa então é desviado para outra seção que dá continuídade a esta opção.
   A seção primeiro confere se o tamanho disponível é o suficiente para a exigência da função, em caso negativo, o programa retorna para o laço for e continua a iteração, já em caso positivo, marca o bloco como ocupado e se divide em duas vertentes:
   - O espaço disponível é suficiente apenas para o bloco que se deseja alocar. Neste caso, o programa define o retorno da função e termina.
   - É possível dividir o bloco em dois blocos funcionais, ou seja, depois de separar o espaço requerido, sobram no mínimo 17 bytes para a formação de um novo bloco livre.
     Para isso, é sobrescrito o tamanho do bloco utilizado e o ponteiro é movido para o fim do bloco, para então inserir o valor 0 na posição seguinte e o tamanho restante disponível na próxima.
3. O bloco atual está ocupado, então o programa itera pela Heap e aponta para o próximo bloco na memória, voltando para o início do laço.

memory_free

*int memory_free(void *pointer);*

Esta função recebe um ponteiro para o início do bloco editável do bloco que deseja desalocar como argumento, assim, o ponteiro é movido para o bloco de indicador de ocupação desse bloco, e então o programa verifica se essa é uma posição válida na Heap (maior que o início e menor que o fim). Se não for, o programa retorna 0 e encerra, se for, confere se o bloco indica ocupação ou não. Caso indique não ocupação, também é retornado 0, caso indique ocupação, insere o valor 0 no indicador de ocupação e retorna 1, em ambos os casos o programa encerra logo após definir o retorno.