Memory Manager

http://users.ece.utexas.edu/~valvano/arm/Heap_4F120.zip

Definitions

Internal fragmentation

convenient of the operating system
contains no information
wasted in order to improve speed or provide for a simpler implementation.
wasted storage is inside the allocated region.

External fragmentation

largest block that can be allocated is less than the total amount of free space
occurs because memory is allocated in contiguous blocks
occurs over time as free storage becomes divided into many small pieces.
Worse when application allocates/deallocates blocks of storage of varying sizes.
unusable storage is outside the allocated regions.

Explain what happens if

```
char v1;
short v2;
char v3;
long v4;
```

Think of the four storage categories, give examples of each:

Private scope, temporary allocation
Private scope, permanent allocation
Public scope, temporary allocation
Public scope, permanent allocation

heap

large piece of memory
managed by the operating system
used for temporary allocation
initialization **`Heap_Init`** called by OS during the initialization process
allocation **`Heap_Malloc`** called by user or OS
deallocation **`Heap_Free`** called by user or OS
statically allocated storage assigned by the compiler (2000-byte heap)

```
static long Heap[500];
```

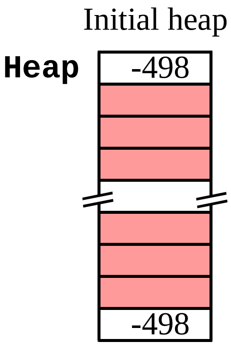

Figure 3.7. An initial heap of 2000 bytes is one block of 498 words (each box is 32 bits).

The user or OS itself calls **Heap_Malloc** when it needs a contiguous blocks of memory. If the memory is needed for a long time, a pointer to it should be stored in permanent memory. For example, if a 20-byte buffer is needed, we could call

```
char *Pt;
void UserStart(void){  // called at the beginning
  Pt = Heap_Malloc(20);
}
```

When the program is finished with the block, it is released by calling **Heap_Free**.

```
void UserFinish(void){ // called at the end
  Heap_Free(Pt);
}
```

> **Checkpoint 3.7:** What happens if a function allocates a block, stores a pointer to the block in a local variable, and then returns from the function without deallocating the block?

Saving the pointer to an allocated block in a local variable does not make sense. If the memory is needed for the duration of one function call, the block should be allocated on the stack. For example, if a 20-byte buffer is needed, we could call

```
void User(void){ char buffer[20];
// use 20-byte buffer
}
```

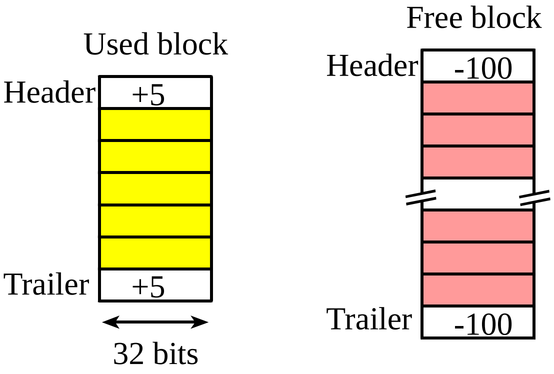

Figure 3.8. Each block has a header and a trailer.

When allocating blocks we can use a number of algorithms to choose which block to allocate. Let **n** be the number of bytes requested by **`Heap_Malloc`**.

- **First fit** uses the first free block with a size greater than or equal to **n**.
- **Best fit** uses the smallest free block with a size greater than or equal to **n**.
- **Worst fit** uses the largest free block with a size greater than or equal to **n**.

Depending on the allocation pattern of the user program, these three allocation methods will have differing levels of external fragmentation. The implementation on the book web site as **Heap_xxx.zip** uses first fit.

Checkpoint 3.8: How would you change the way free blocks are organized to implement best fit?

block is allocated with **`Heap_Malloc`**

find a free block
free block is divided to two parts
new free block is smaller,
a pointer to the allocated block is returned
block may not be large enough to split (allocate the big block, internal frag)

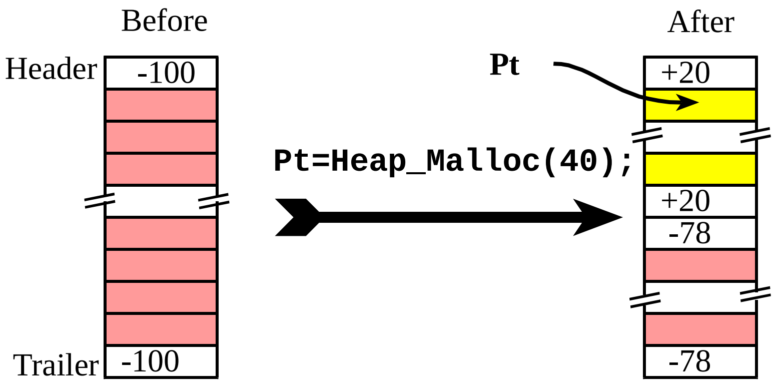

Figure 3.9. Example, the user calls Pt=Heap_Malloc(80).

Checkpoint 3.9: In Figure 3.9, why does the sum of the parts not equal the whole? In particular, 20+78 does not equal 100.

When deallocating a block with **`Heap_Free`**, there are four cases:

no merge,
merge above,
merge below and
merge both above and below.

.

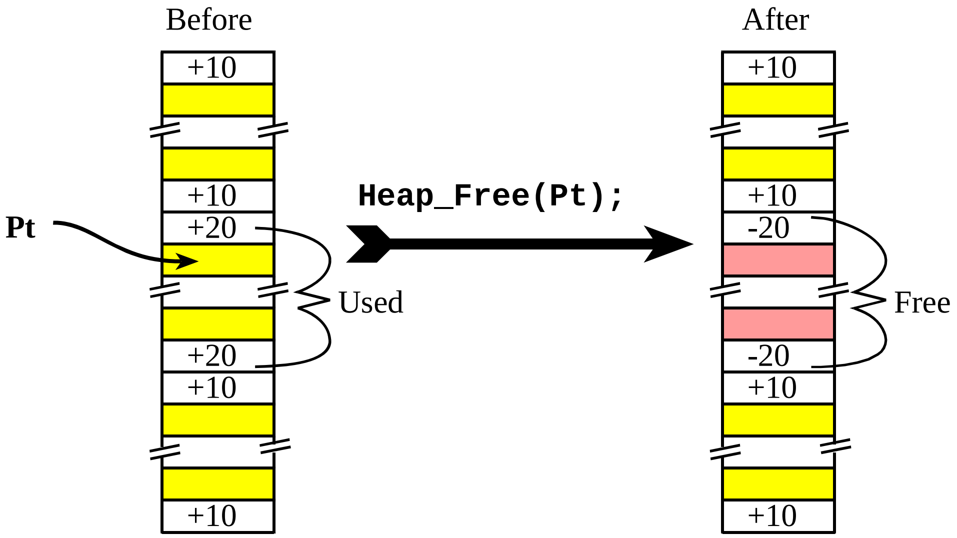

Figure 3.10. Example, the user calls Heap_Free(Pt) (no merge).

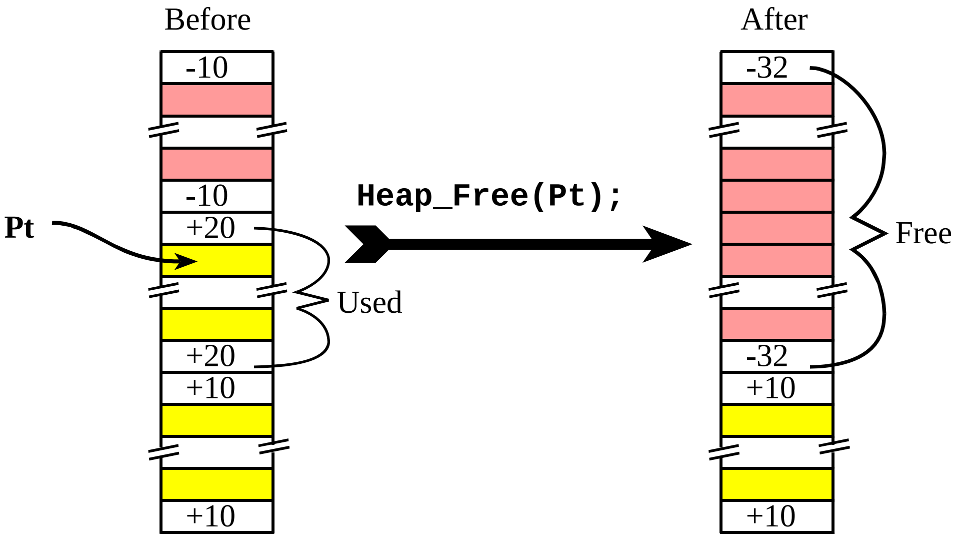

Figure 3.11. Two blocks are merged during a call to Heap_Free.

Checkpoint 3.10: What happens if you continue to access a memory block after the block is deallocated?

The Knuth *buddy* allocation maintains the heap as a collection of blocks each with a size of $2^{\mathbf{m}}$. When the user requests a block of size **n**, it will find the smallest block with $2^{\mathbf{m}}$ greater than or equal to **n**. For example, if the smallest block is size 1024, and the user requests a block of 100 bytes, the 1024-byte block will be divided into two 128-byte blocks, one 256-byte block and one 512-byte blocks. The user will be given the 128-byte block. The 28 extra bytes allocated to the user is internal fragmentation.

Final exam 2010

(20) Question 8. Implement a memory manager for fixed sized blocks. Let the block size be

```
#define SIZE 100   // size in bytes
```

Let the number of blocks be

```
#define NUM 10     // number of blocks
```

For these two definitions, 1000 bytes of memory will be managed. Create three functions: initialization, allocate and deallocate.