EE445M/EE380L.12
Embedded and Real-Time Systems/ Real-Time Operating Systems

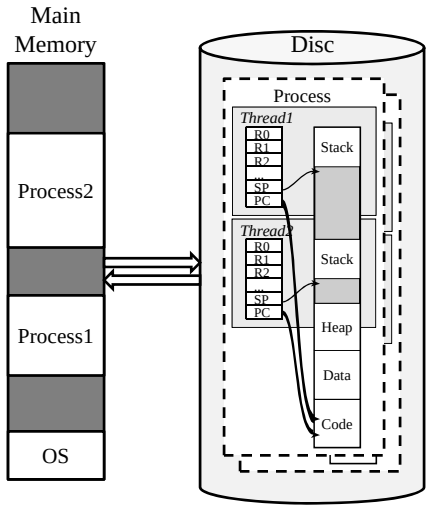
Lecture 9:
Memory Management, Heap

Operating System

- Manage computer system resources
  - CPU, processors
    - Threads
  - Storage, flash/disc
    - Files
  - Memory, RAM
    - Heap, processes

Memory Management

- Sharing
  - Per-thread: stack
  - Per-program/-process: heap, code, data
- Allocation
  - Static, permanent
    - Globals, OS code
  - Dynamic, temporary
    - Stack, heap, process swapping
- Protection
  - Access control

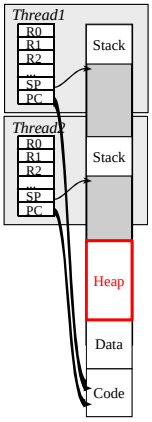


Fragmentation

- Internal
  - Wasted space inside allocated region
  - Convenience of the operating system
  - Contains no information
  - Wasted in order to improve speed or provide for a simpler implementation
- External
  - Unusable storage is outside the allocated regions
  - 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/OS allocates/deallocates blocks of storage of varying sizes

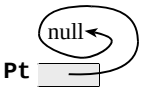
Heap

- Separate piece of main memory
  - "Memory region" in µCOS-II
- Managed by the operating system
  - Initialization **Heap_Init** called by OS during the initialization phase
- Used for temporary allocation
  - Allocation **Heap_Malloc** called by user or OS
  - Deallocation **Heap_Free** called by user or OS

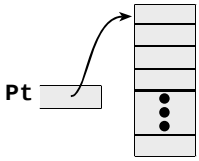


Dynamic Memory Allocation

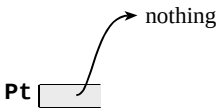
Before **malloc**



After **malloc**



After **free**




```
void Function(void){
  int i;
  int *pt;  int pt[20];
  // allocate 20 words
  pt = (*int)Heap_Malloc(4*20);
  for(i = 0; i < 20; i++)
    // put data into array
    pt[i] = i;
  Heap_Free(pt);
}
```

```
int *Pt;
void Begin(void){
  // allocate 20 words
  Pt = (*int)Heap_Malloc(4*20);
}
void Use(void){ int32_t i;
  for(i = 0; i < 20; i++)
    // put data into array
    Pt[i] = i;
}
void End(void){
  Heap_Free(Pt);
}
```

Heap Manager

- Heap_Init
  - Allocate & initialize heap memory
    - Statically allocated storage assigned by compiler

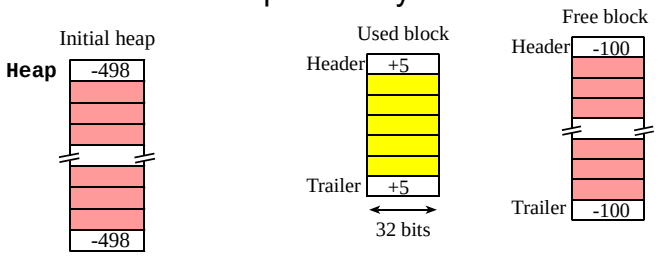
```
static long Heap[500]; // 2000 byte heap
```

- Heap_Malloc
  - Allocate block in heap free space
    - Must use contiguous allocation
    - First fit, best fit, worst fit
- Heap_Free
  - Reclaim block into heap free space

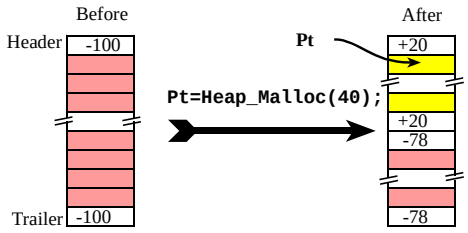
Heap_4C123.zip

Heap Manager Example

- Blocks of variable size
  - Size counter at beginning/end of each block
    - Positive if used (allocated), negative if free
  - Internal fragmentation
    - Overhead for size header/trailer
    - Allocated in multiple of 4 byte words



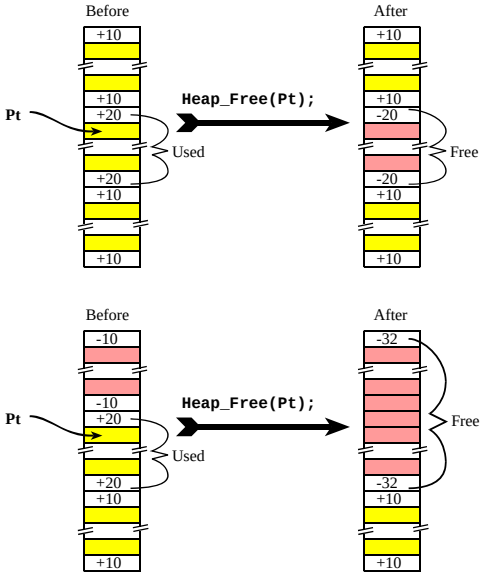
Heap_Alloc



- Allocate block
  - Find a free block
    - Uses first fit
  - Free block is divided into 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 fragmentation

Heap_Free

- Four cases
  - No merge
  - Merge above
  - Merge below
  - Merge both above and below



Knuth's Buddy Allocation

- Maintain heap as collection of blocks each with a size of 2^m
- When user requests a block of size **n**
  - Find smallest block with $2^m \geq$ **n**
  - Split block into half until best fit
- When user releases a block
  - Merge with other half (buddy block of same order), if possible

Final Exam 2010