CS 340
Computer Systems

#8: malloc, Page Eviction, and Threads

Sep. 14, 2023 · G Carl Evans

Data Structures for Heap Management

When we manage heap memory, we need to use memory to help us store memory:

- Overhead:
- Allocated Memory:

Metadata-based Approach to Memory Storage

06/heap.c

```
int *a = malloc(4096);
printf("a = %p\n", a);
free(a);

int *b = malloc(4096);
printf("b = %p\n", b);

int *c = malloc(4096);
printf("c = %p\n", c);

int *d = malloc(4096);
printf("d = %p\n", d);

free(b);
free(c);

int *e = malloc(5000);
printf("e = %p\n", e);

int *g = malloc(10);
printf("g = %p\n", g);

int *g = malloc(10);
printf("g = %p\n", g);
```

Heap w/ Data Structures:

Pages in Cache – Eviction/Replacement Strategies:

We know that memory is divided into pages, a page table provides a translation between virtual page numbers and physical pages, and that we allocate memory via malloc. How do we decide what pages to cache?

Strategy #1:

	17	33	40	17	43	8	99	33	99	17
CACHE										

Strategy #2:

	17	33	40	17	43	8	99	33	99	17
CACHE										

Strategy #3:

	17	33	40	17	43	8	99	33	99	17
CACHE										

Strategy #4:

	17	33	40	17	43	8	99	33	99	17
CACHE										

Other Strategies:

Fragmentation

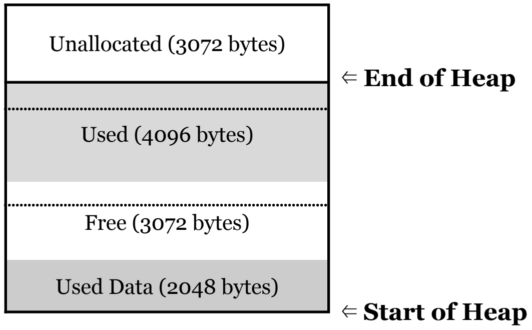
As we develop various systems for storage, we want to minimize **fragmentation**.

- [Fragmentation]:

- [Internal Fragmentation]:

- [External Fragmentation]:

Fragmentation Example in Heap Memory:



Abstraction #4: Computer Peripherals

- Every other piece of hardware we consider to be a “peripheral”.
- Interface managed by the ______________________________.
    - ...and managed using ___________________________.
- Examples:

Threads: The Unit of Computation in an Operating System

As a programmer, the single most important construct in an Operating System is a thread.

- Every thread has a **program counter**, a pointer that stores the next instruction to be read by a program.
- A _______________ is an organization of one or more threads in the same context. A simple process has only one thread.
- In C, the initial thread is called the _______________________.
    - It is what starts running your main() function!

Example: Launching Fifteen Threads

07/fifteen-threads.c

```c
#include <pthread.h>

const int num_threads = 15;

void *thread_start(void *ptr) {
  int id = *((int *)ptr);
  printf("Thread %d running...\n", id);
  return NULL;
}

int main(int argc, char *argv[]) {
  // Create threads:
  int i;
  pthread_t tid[num_threads];
  for (i = 0; i < num_threads; i++) {
    pthread_create(&tid[i], NULL,
                               thread_start, (void *)&i);
  }

  printf("Done!\n");
  return 0;
}
```