Operating Systems - Chapter 14 Summary

Types of Memory

• Stack memory: managed automatically by compiler, short-lived, for local variables.
• Heap memory: managed manually by programmer using malloc() and free(). Long-lived but error-prone.

The malloc() Call

• Allocates requested number of bytes on the heap and returns a pointer.
• Usage usually combined with sizeof() operator to request correct size.
• Returns NULL on failure; programmer must check.

The free() Call

• Frees memory allocated by malloc().
• Library keeps track of allocated block sizes; user only passes pointer.

Common Errors in Memory Management

• Forgetting to allocate memory before use (segfaults).
• Not allocating enough memory (buffer overflow, potential security risk).
• Forgetting to initialize allocated memory (uninitialized reads).
• Forgetting to free memory (memory leaks).
• Freeing memory too early (dangling pointers).
• Freeing memory twice (double free, undefined behavior).
• Calling free() on invalid pointers.

Underlying OS Support

• malloc/free are library calls, not system calls.
• Implemented with system calls like brk/sbrk (adjust heap end) and mmap (create new memory regions).
• Users should not call brk/sbrk directly; stick to malloc/free.

Other Calls

• calloc(): like malloc but also zeroes the memory.
• realloc(): resizes an allocated block, copying contents if needed.

Summary

The memory API in C provides stack memory (automatic) and heap memory (manual). Programmers use malloc() and free() to manage heap memory, but errors are common: leaks, overflows, dangling pointers, etc. Underlying support comes from system calls like brk, sbrk, and mmap. Additional calls like calloc and realloc add flexibility. Mastering this API is essential for building robust systems and avoiding memory-related bugs.