Memory Management

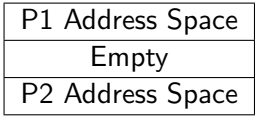
44-550: Operating Systems

Memory Manager

- Manages allocation and deallocation of main memory
- Quality is key
  - So much deals with memory
  - Significantly impacts performance
- Primitive in non-multiprogramming systems
- Uses small blocks of storage located in physical memory in non-contiguous places
- Virtual memory management
  - Abstracts away from how memory is implemented on the device
  - Applications can use memory regardless of implementation

Process Address Space

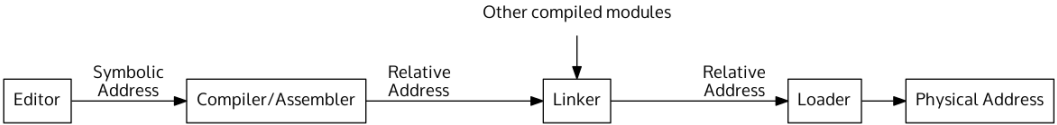
- Set of logical addresses a process references in its code
  - Assumes a mechanism to map logical addresses to physical through the OS
- Logical addresses are bound to physical addresses upon allocation



  - Symbolic addresses: used in source program (variable names, etc)
  - Relative addresses: memory addresses relative to the process space
  - Physical addresses: the actual address in RAM

Binding

With a compiled language like C/C++, Fortran, etc., the compiler translates the source with symbolic addresses to object code in the machine language with relative/relocatable addresses (or offsets)
The linker combines program objects wkithother required object modules into a single program with logical addresses.

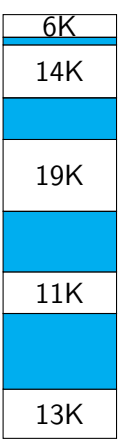


Memory Placement Strategies

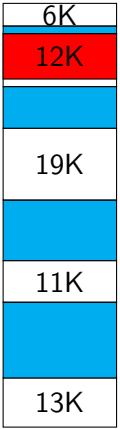
- Fetch strategies: When to move program/data into main memory
  - On demand
  - Anticipatory
- Placement strategies: where does incoming data go?
  - First Fit
  - Best Fit
  - Worst Fit

First Fit

The allocator moves memory into the first spot big enough to handle the memory. Since there may be many holes in the memory, this is an efficient mechanism. Say we want to place a 12KB chunk of data into memory:



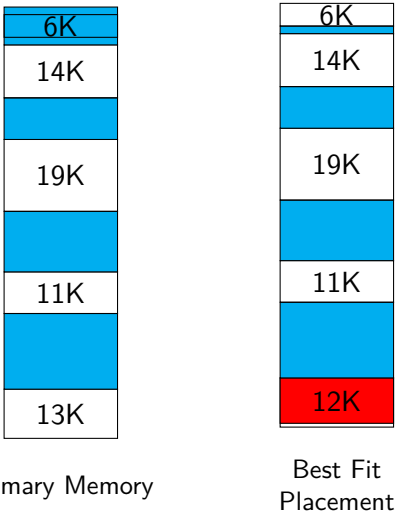
Primary Memory



First Fit Placement

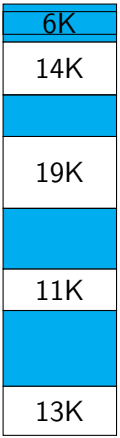
Best Fit

The allocator places a process in the smallest block of unallocated memory in which it will fit. Say we want to place a 12KB chunk of data into memory:
Note: both best fit and worst fit have small pieces of memory left over in the chunks where the data was placed; it is unlikely that any program data will be able to fit there (resulting in wasted memory)



Worst Fit

The allocator places a process in the largest block of unallocated memory in which it will fit. Say we want to place a 12KB chunk of data into memory:
The idea here is that there will (hopefully) be enough remaining space to allocate more data there.



Primary Memory

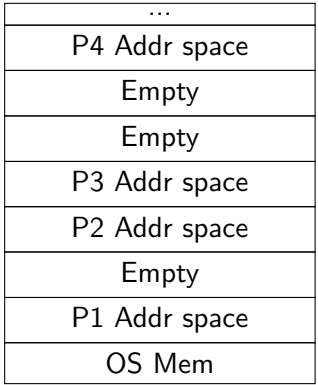
Worst Fit Placement

Memory Placement Strategies

- Replacement strategies: when memory is too full or fragmented, need to remove some of the program or data
- Difficult to use in multiprogramming environment
  - Multiple processes/threads can access the same data; how do we know who is going to use it when?
- Must keep track of contiguous allocations (fragmentation is possible), compact memory, etc.
- Need virtual memory, segmentation, and paging
- What we have: physical and virtual addresses

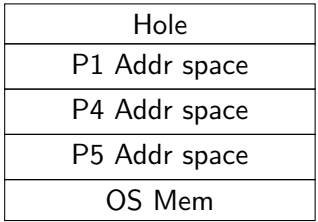
Memory Partitioning

- Can be done contiguously or non-contiguously
- Contiguous allocation has many issues
  - Internal Fragmentation
  - Fixed size holes
- We'll focus on dynamic partitioning



Dynamic Contiguous Partitioning

- Use variable size partitions
- Create when sufficient memorny is available
- Number of partitions is variable
- Blocks of available memory is called holes
- Still have fragmentation, but it is external
  - Fragmentation doesn't happen within process memory



Dynamic Contiguous Partitioning

- Can compact memory
- Can use dynamic relocation to move processes in memory even after starting the program
- Swapping may be necessary
- When a process is blocked (e.g. for I/O) it can be swapped to disk and the memory used for something else
  - Being blocked must be more expensive than swapping
  - If the process is blocked for 10 ms, and it takes 6ms to transfer to and from the disk...

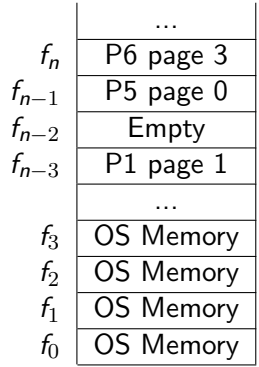
Non-contiguous Allocation

- Helps to remove fragmentation
- Uses two core concepts to accomplish this
  - Paging
  - Segmentation

Paging

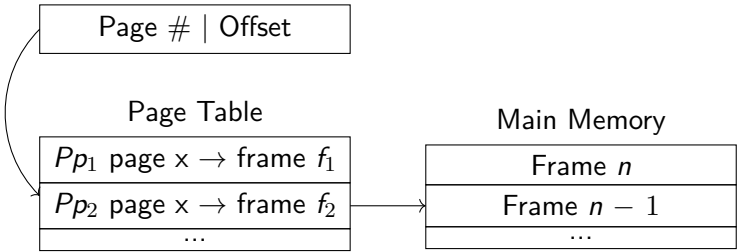
- Divides the process address space into small fixed sized blocks of logical memory called *pages*
- Process size is then measured in number of pages
- Physical memory is divided into frames
- Page size is a power of 2
- Typically, one page maps to one frame of physical memory

Paging Memory Layout



Logical Addressing with Paging

- Addressing requires the page number and an offset



Logical Addresses

- Logical address of a process consists of a page number and an offset
  - Physical address is determined by the frame number and the offset
- Page table keeps track of which frame corresponds to which page
  - One table entry per page (and one page per frame, frequently)
- Most significant bits are the page number
- Least significant bits are the offset
  - Example: 20 bit addressing, have 1024 pages with 1024 bytes each
    - first 10 bits: page
    - next 10 bits: offset

0000000010 0111011110

Page 2 Offset: 478 bytes

Advantages and Disadvantages of Paging

Advantages

- No external fragmentation
- Simple memory management algorithm
- Swapping is easy
- Transparent to programmer (system allocates memory)

Disadvantages

- Internal fragmentation on last page of process memory
- Page tables may consume more memory
- Can protect memory by marking whether or not a frame is valid or invalid