Operating Systems

Christopher Kruegel
Department of Computer Science
UC Santa Barbara
http://www.cs.ucsb.edu/~chris/

Memory Management

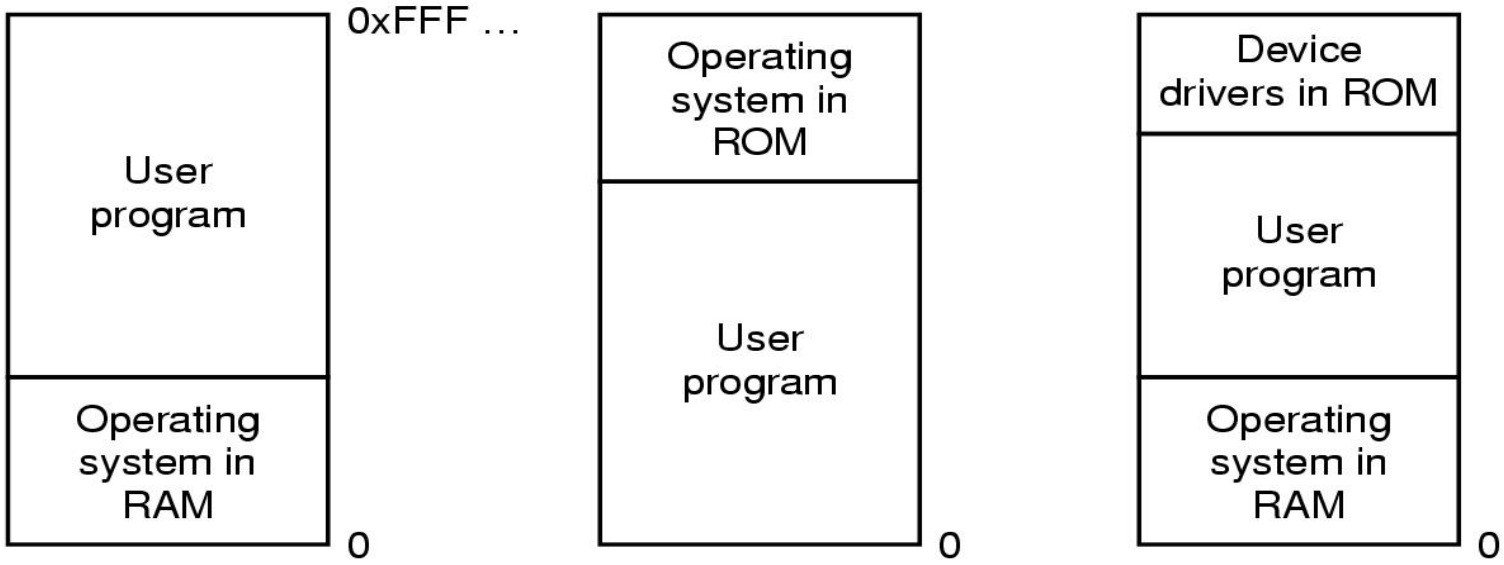
- Ideally memory would be
  - Large
  - Fast
  - Non volatile
- In practice, a memory hierarchy is used
  - Small amount of fast, expensive memory cache
  - Some medium-speed, medium price main memory
  - Gigabytes of slow, cheap disk storage
- The memory manager handles the memory hierarchy

Memory Management Approaches

- Mono-programming vs. multi-programming
- Fixed program set vs. swapping
- Complete image vs. partial image (virtual memory)
- Modern systems
  - multi-programmed, with swapping and virtual memory

Simplest Memory Management

- Run only one user process at a time
- Operating/system and device drivers resident or in ROM



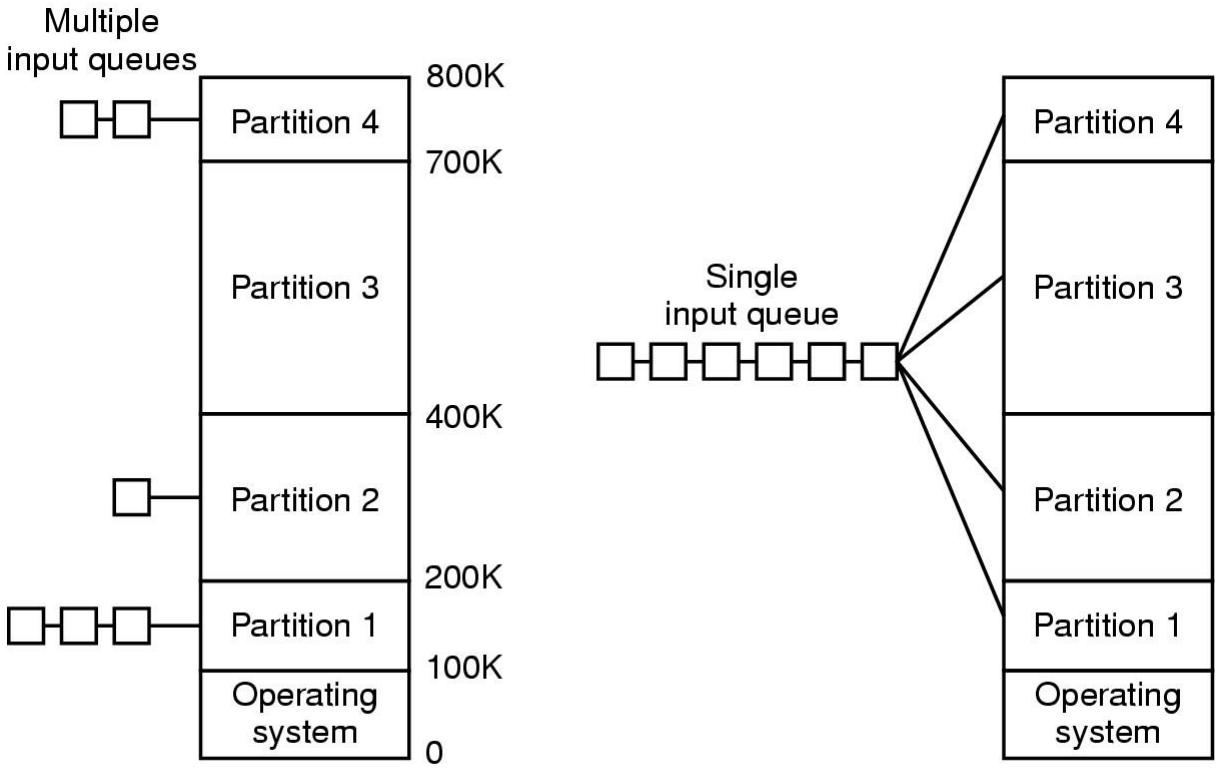
Dealing With Multiprogramming

- Available memory is divided into a number of fixed partitions
- Partitions can be of different sizes
- When process needs to be executed, put in one of available partitions

- Problem: if a small program gets a big partition memory gets wasted

- Solution
  - Use different queues for different partitions
  - Use one queue and choose the largest job that fits an available partition and use some aging mechanisms to ensure that small jobs are not left to starve (e.g., an “ignored” counter)

Multiprogramming with Fixed Partitions



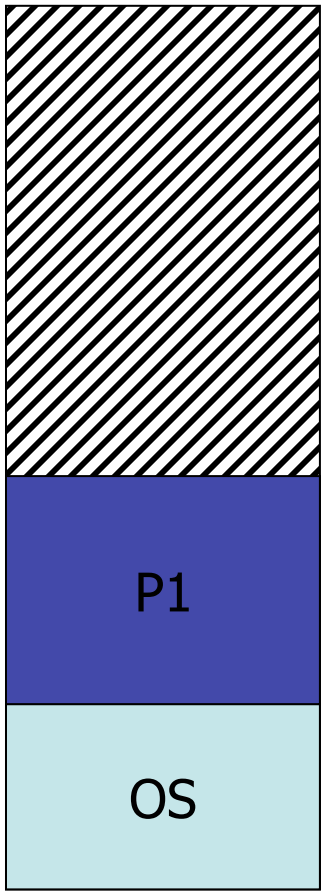
Relocation and Protection

- The programmer cannot be sure where program will be loaded in memory
  - Address locations of variables, code routines cannot be absolute
  - Must keep a program out of other processes' partitions
- Relocation can be done at loading time
  - Maintain information about the addresses that need to be relocated
  - Does not solve the protection problem (protection bits)
- Use base and limit registers
  - Address locations added to base value to map to physical address
  - Attempts to access address locations beyond the limit value generates an error

Swapping

- Most of the time there is not enough memory to hold all the active processes at the same time
- Swapping is the process of saving a task to disk
- In swapping processes are loaded to and discarded from memory following the needs of execution
- Address must be relocated each time either by hardware or by software
- After a while memory can get fragmented and may need compaction, which is computationally expensive
- A process may also try to get bigger and bigger and bigger

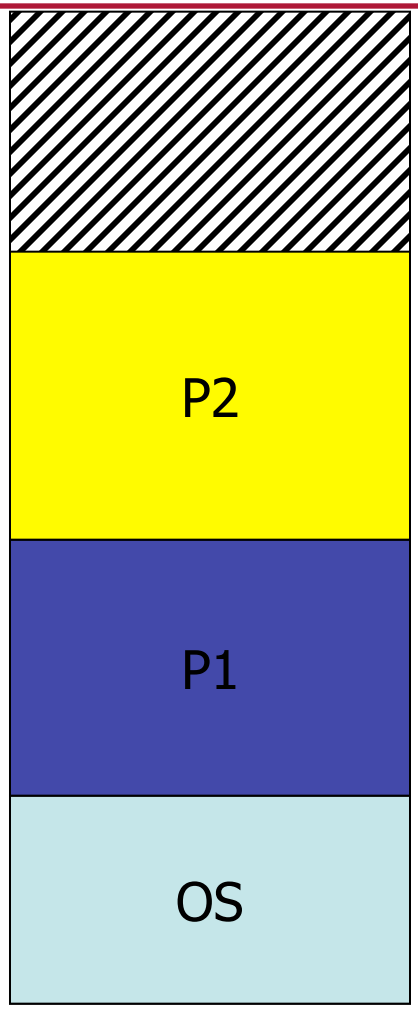
An Example



P1	600k	10s
P2	1000k	5s
P3	300k	10s
P4	700k	8s
P5	500k	15s

P2 can be loaded

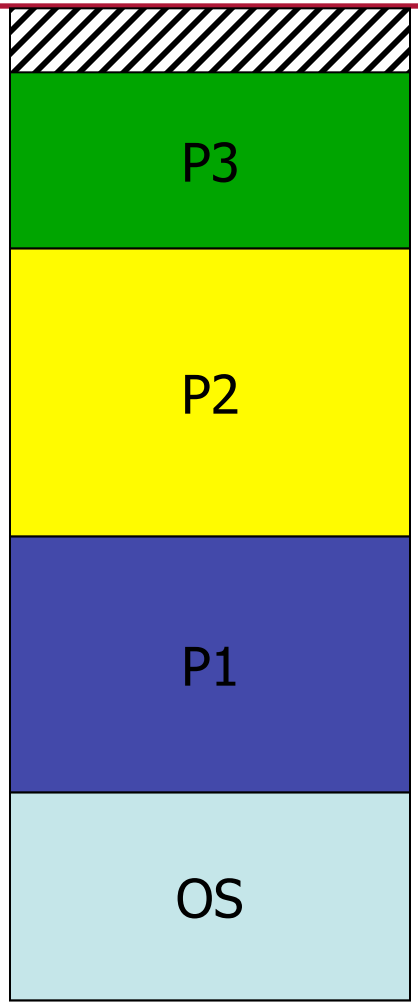
An Example



P1	600k	10s
P2	1000k	5s
P3	300k	10s
P4	700k	8s
P5	500k	15s

P3 can be loaded

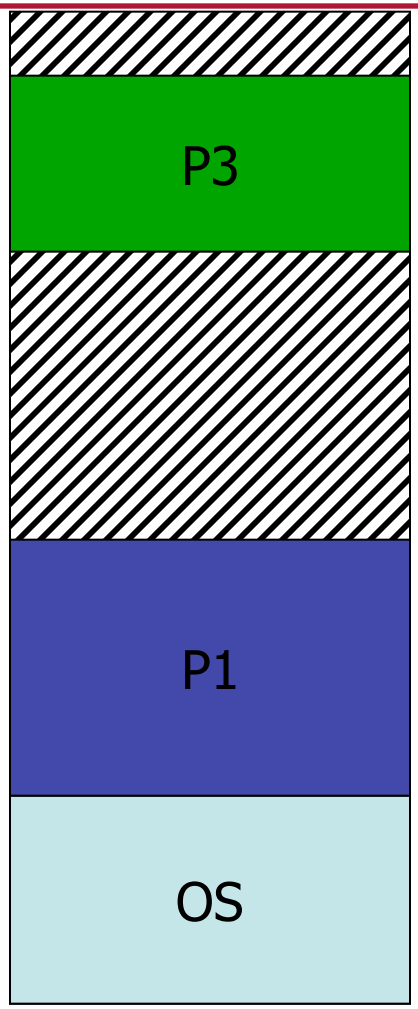
An Example



P1	600k	10s
P2	1000k	5s
P3	300k	10s
P4	700k	8s
P5	500k	15s

P4 & P5 need to wait!

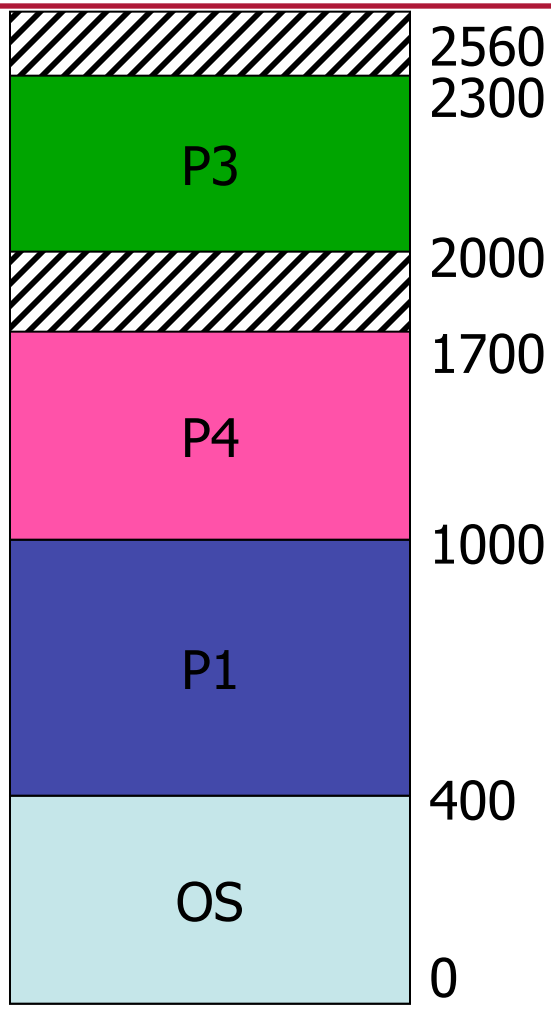
An Example



P1	600k	10s
P2	1000k	5s Terminates
P3	300k	10s
P4	700k	8s
P5	500k	15s

P4 can be loaded

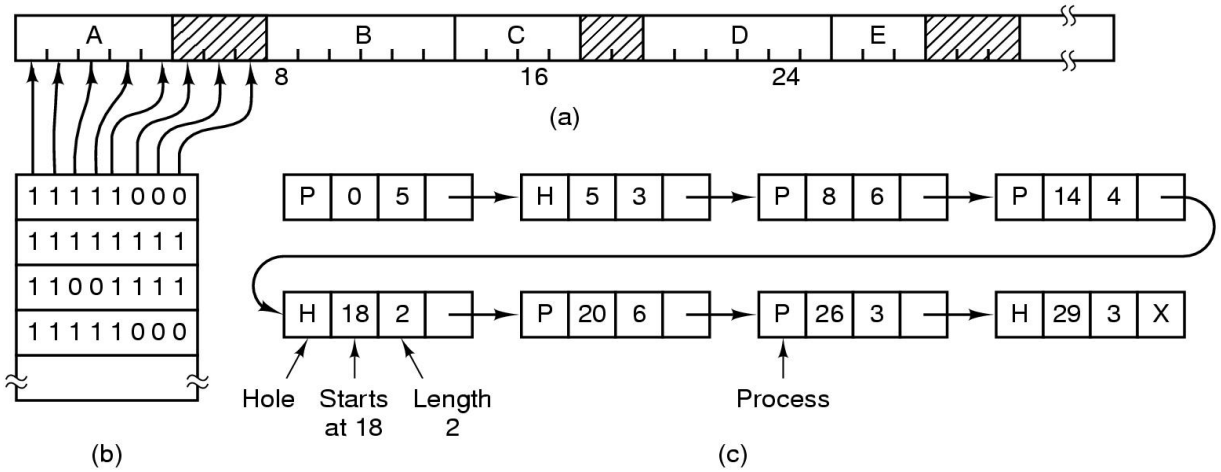
An Example



P1	600k	10s
P2	1000k	5s
P3	300k	10s
P4	700k	8s
P5	500k	15s

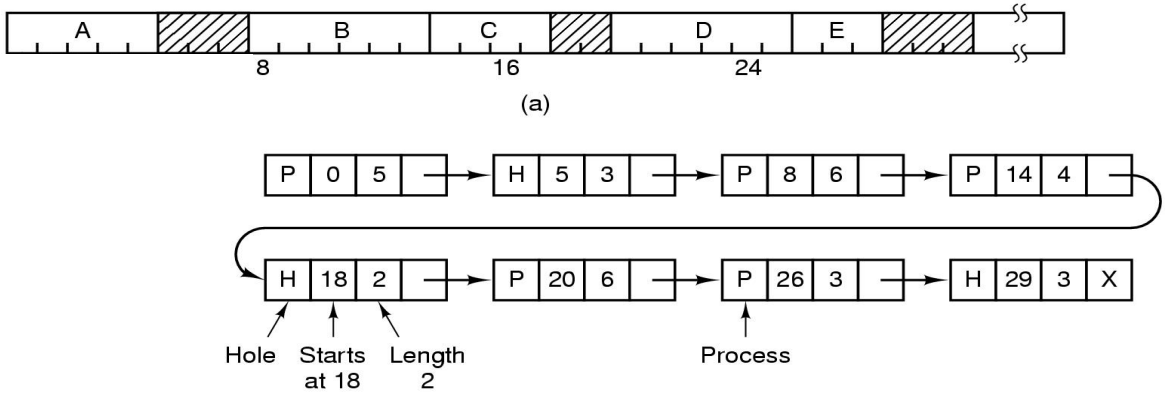
Memory Management with Bitmaps

- Divide memory in allocation units
- Keep track of which units have been used and which ones are free using a bitmap
- Tradeoff:
  - Big allocation units: +small bitmap -may waste memory
  - Small allocation units: -better “fit” +big bitmap



Memory Management with Linked Lists

- Maintain a linked list where each element
  - May represent a process (P) or a piece of free memory (“hole”, H)
  - Contains number of initial unit
  - Contains length of memory block
  - Maintains pointer to each element (single- or double-linked)
- List is usually keep ordered



List Management

- Two possible events
  - A process gets out of the memory
    - Compaction
  - A new process wants to be in memory
    - Allocation algorithms

- First fit
  - Search the list until a suitable hole is found
  - Split the hole in a P and an H

- Best fit
  - Search the entire list and use the smallest hole that fits the program
  - Slow (requires complete scan through the list)

List Management

- Separate lists
  - Maintains two separate lists for processes and holes
  - Holes are ordered by size
  - Speeds up search
  - Makes compaction difficult

- Quick fit
  - Separate lists, hashed by size or size ranges
  - Speeds up search
  - Makes compaction difficult

- Techniques still very relevant for heap management (`malloc`)

Virtual Memory

- What if a program is too big to be loaded in memory?
- What if a higher degree of multiprogramming is desirable?
- Physical memory is split into *page frames*
- Virtual memory is split into pages
- OS (with help from the hardware) manages the mapping between pages and page frames