Memory Management

CS 342 – Operating Systems

Ibrahim Korpeoglu
Bilkent University
Department of Computer Engineering

Memory Management

- *Memory* is an important *resource* that must be carefully managed.
- *Cost*, *size*, and *volatility* are important parameters of memory chips.
  - There is no single memory technology that is good in all these parameters.
- Therefore we have a memory hierarchy
  - Registers, Cache, Main Memory, Disk, Tape…
- The part of the OS that manages the memory hierarchy is called the *Memory Manager*.

Memory Manager Jobs

- Its is job is to keep track of which parts of memory are used by which programs and which parts are free.
- Allocate memory when programs need it, and de-allocate it when programs finished.
- Manage swapping of programs between memory and disk.

Basic Memory Management

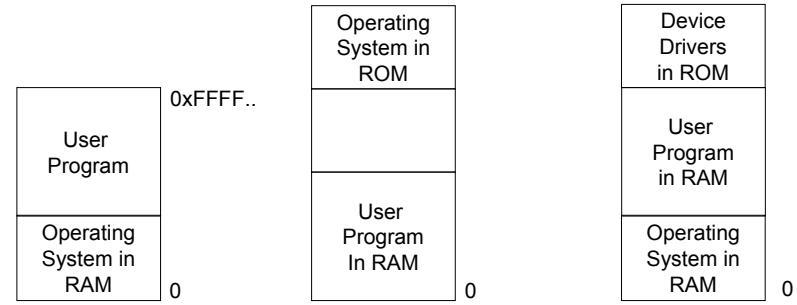
- Two classes of memory management systems
  - That move processes back and forth between memory and disk
  - That have all processes in memory during execution until termination.
    - We will look to this first.

Basic Memory Management

- Memory management that has a process in memory until termination.
  - Mono-programming without Swapping or Paging
  - Multiprogramming with Fixed Partitions
  - Modeling Multiprogramming
    - Analysis of Multiprogramming System Performance
  - Relocation and Protection

Mono-programming

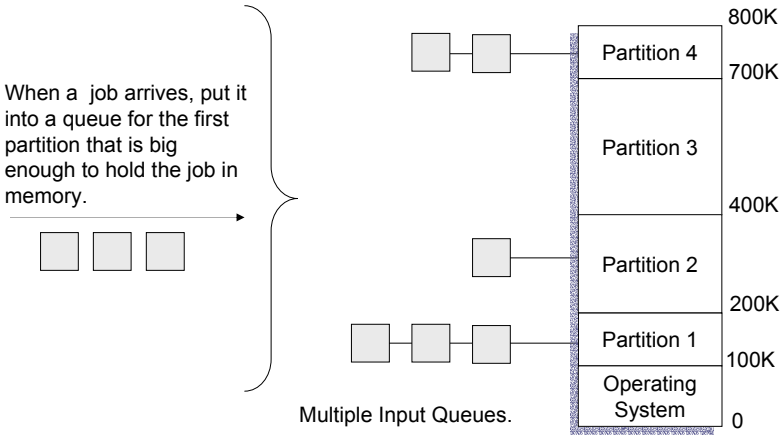
- Run one program at a time
- OS and one program will exists in the memory



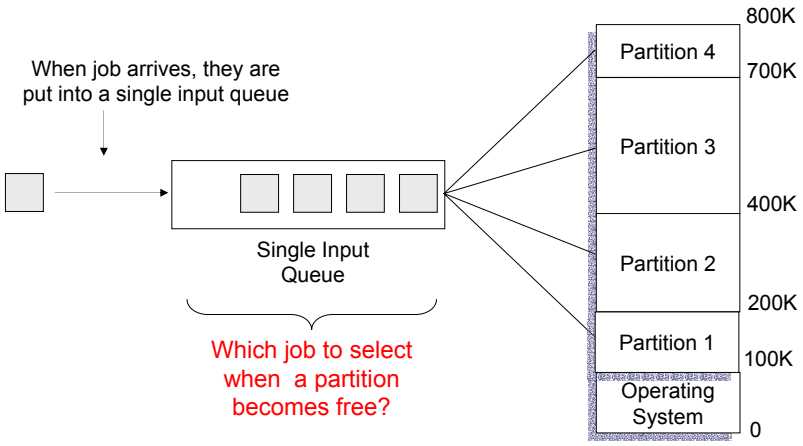
Multiprogramming with Fixed Partitions

- Modern systems allow multiple programs to run at the same time.
- This allows high CPU utilization:
  - While a process in blocked for I/O, an other one can use CPU.
- Divides the memory into n partitions (possible each with different sizes)
- The partitioning can be done at system boot up time.

Ways to use fixed partitions - 1



Ways to use fixed partitions - 2

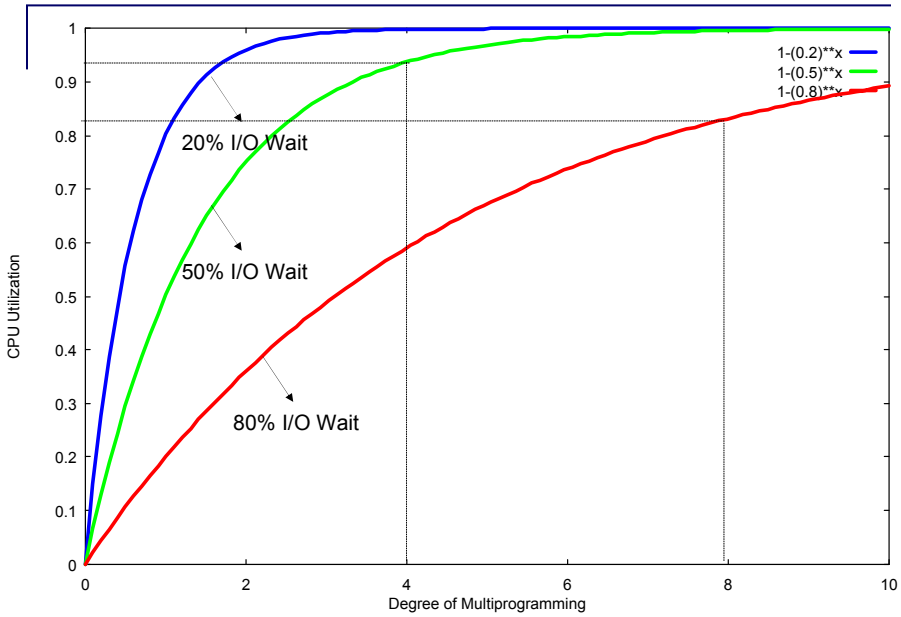


Ways to use fixed partitions - 2

- Selecting a Job when a partition become free!
  - 1) Select the first job from the start of queue that fits in to the available partition.
  - 2) Search all the jobs and find the largest sized job that fits into the available partition.
    - Discriminates against small jobs.
    - Solve by counting the *skip overs* for the fitting jobs. If a job is skipped *k* times, then don't skip it in the next search

Modeling Multiprogramming

- Multiprogramming improves CPU utilization.
- Assume a process spends a fraction p (0<=p<=1) of its waiting for I/O to complete.
- With n process in memory (n: degree of multiprogramming):
  - The chance that all n processes are waiting for I/O is: p^n
  - The CPU utilization is then: $1-p^n$.



80% average I/O wait

Memory Size	OS Size	program Size	#programs that can fit (degree of multiprogramming)	CPU utilization	Percent Increase in CPU utilization
32MB	16MB	4MB	4	60%	-
48MB	16MB	4MB	8	83%	38%
64MB	16MB	4MB	12	93%	12%

Analysis of Multiprogramming Systems

- Assume 4 jobs arrive to a batch system ayt 10:00 am in the morning precisely at the following moments:
  - Job 1: 10:00am (requires 4 minutes of CPU)
  - Job 2: 10:10am (requires 3 minutes of CPU)
  - Job 3: 10:15am (requires 2 minutes of CPU)
  - Job 4: 10:20am (requires 1 minute of CPU)
- When does each job completes?

Analysis of Multiprogramming Systems

p = 0.8 and n is in {1, 2, 3, 4}

	Process 1	Process 2	Process 3	Process 4	
CPU idle	.80	.64	.51	.41	p^n
CPU busy	.20	.36	.49	.59	$1-p^n$
CPU/process	.20	.18	.16	.15	$(1-p^n)/n$

Analysis of Multiprogramming Systems