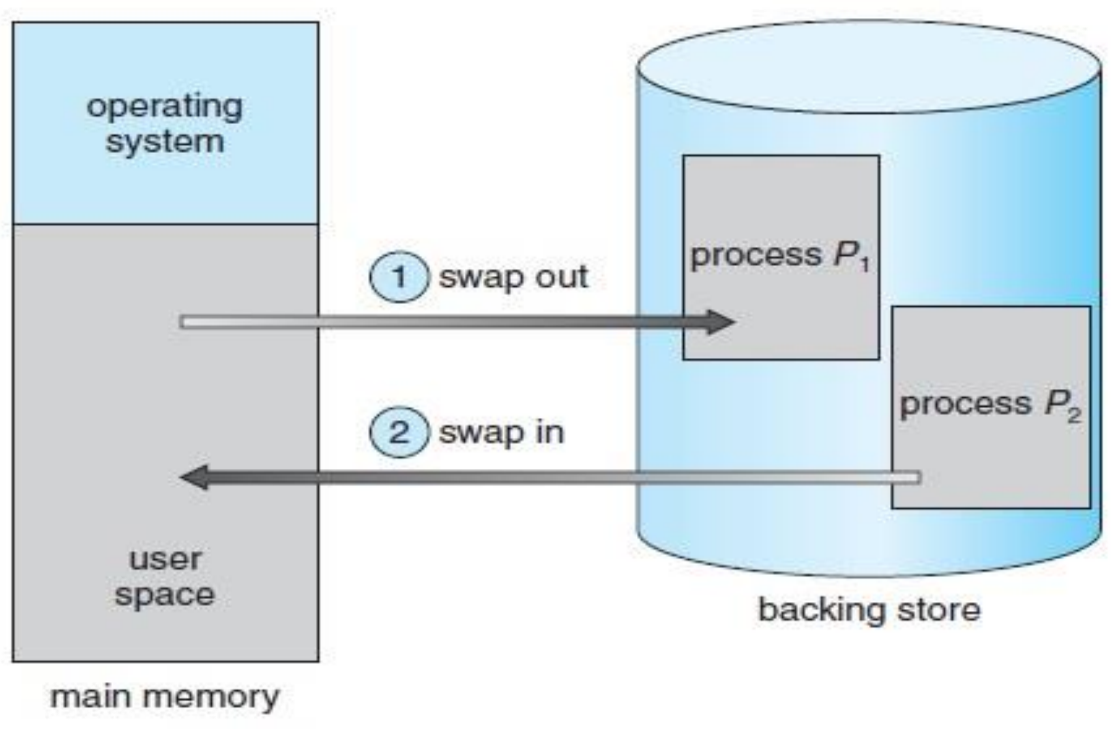
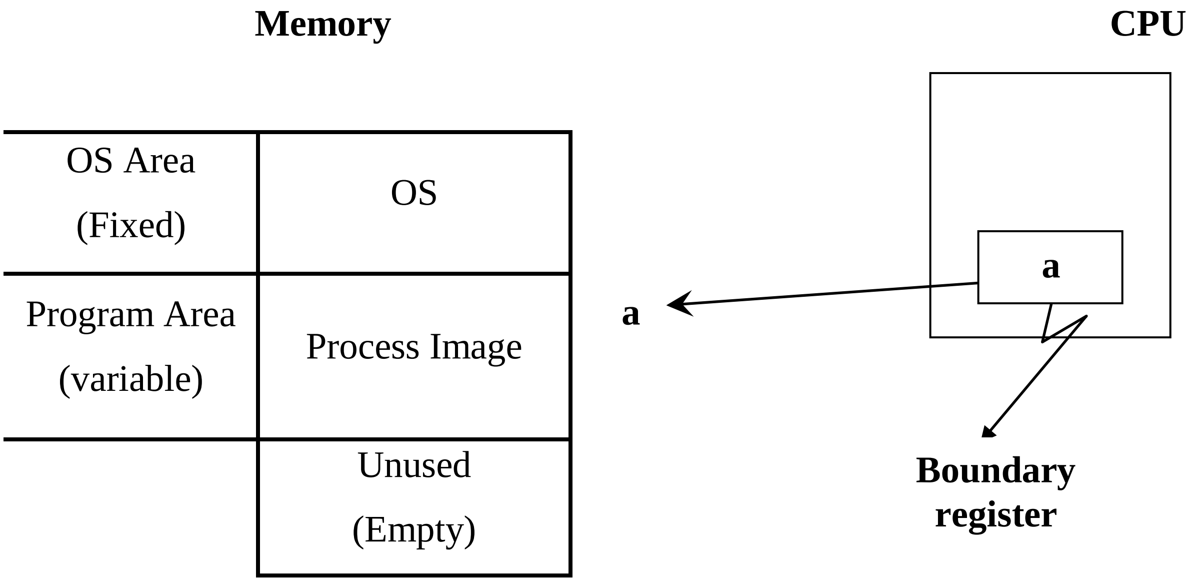
Swapping:

A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution

- Backing store – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- Roll out, roll in – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped
- Modified versions of swapping are found on many systems (i.e., UNIX, Linux, and Windows)
- System maintains a ready queue of ready-to-run processes which have memory images on disk



Protection in a Single-User System



-Without protection, the process may alter the operating system. The protection can be implemented with a single boundary register built into the processor which can be modified only by a privileged instruction.

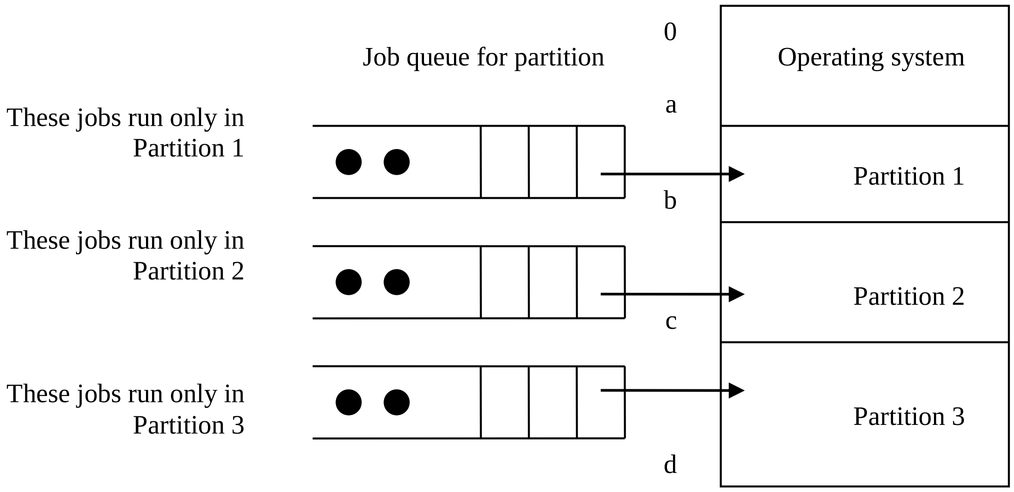
-The boundary register contains the memory address at which the user's program begins. Each time a process references a memory address, the system determines if the request is for an address greater than or equal to that stored in the boundary register.

- If so, the system services the request. If not, then the program is trying to access the operating system. The system intercepts the request and terminates the process with an appropriate error message.

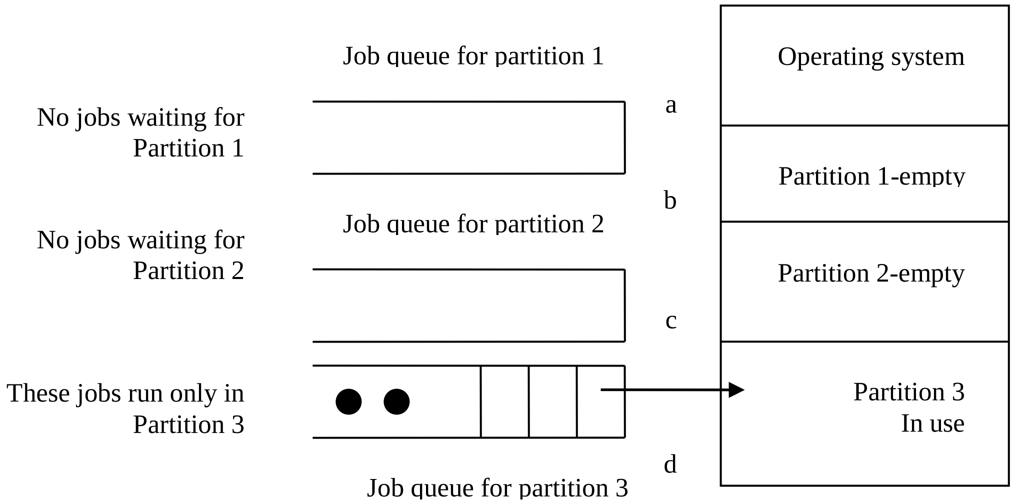
Fixed-Partition Multiprogramming

-The earliest multiprogramming systems used fixed partition multiprogramming. The system divides main memory into a number of fixed size partitions. Each partition holds a single job.

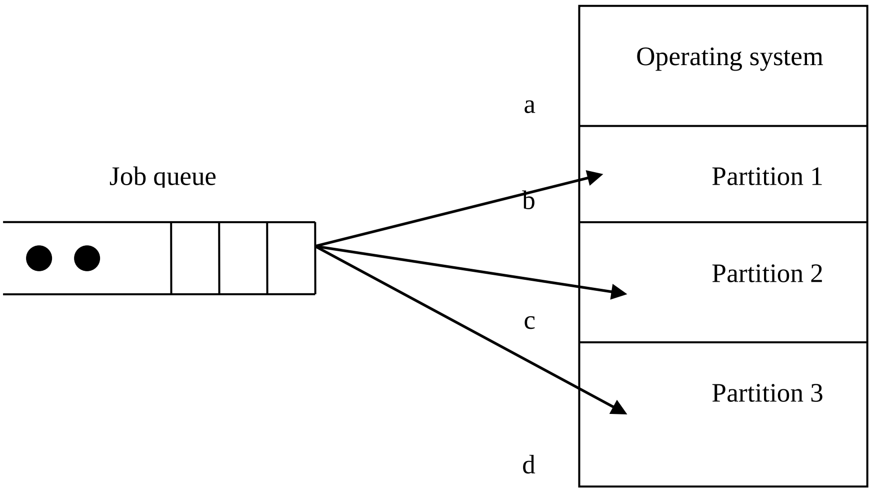
-In the earliest multiprogramming systems, the programmer translated a job using an absolute assembler or compiler. It meant that a job had its precise location in memory determined before it was launched and could run only in a specific partition. If the programs partition was occupied then that job had to wait even if other partitions were available.


0
Job queue for partition
Operating system
a
These jobs run only in Partition 1
Partition 1
b
These jobs run only in Partition 2
Partition 2
c
These jobs run only in Partition 3
Partition 3
d

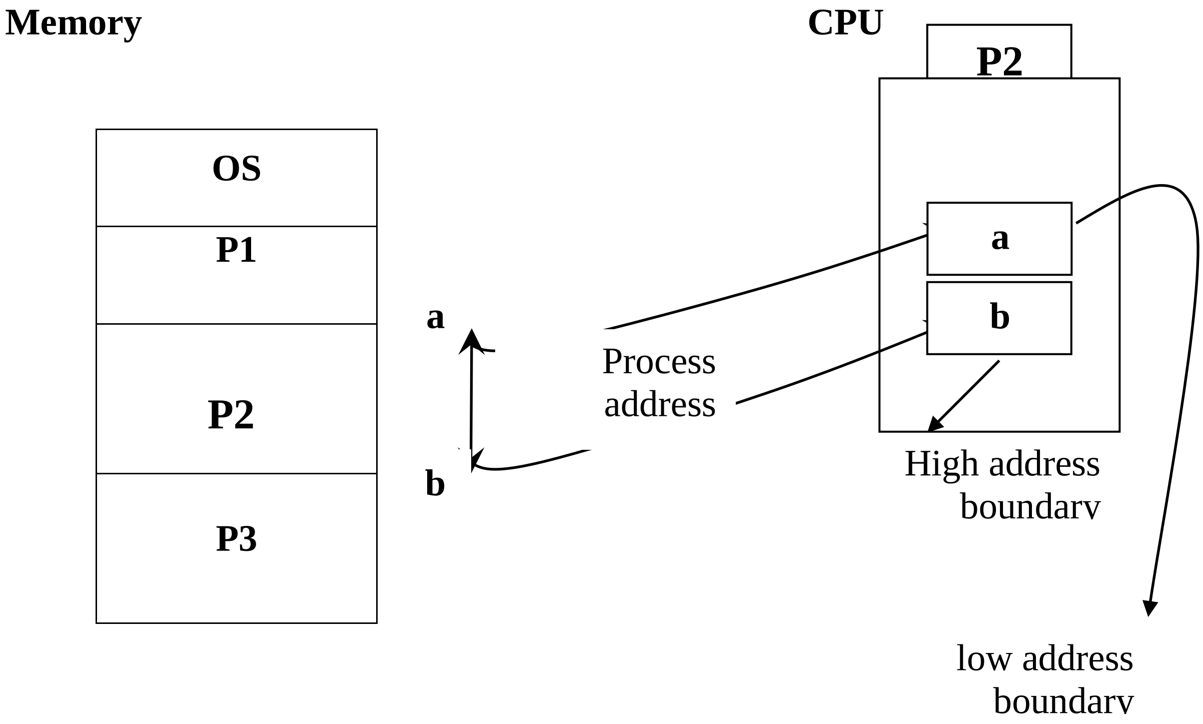
-In the following figure, all the jobs in the system must run in partition 3. Because this partition currently is in use, all other jobs are forced to wait, even though the system has two other partitions in which the jobs could run.


Job queue for partition 1
Operating system
a
No jobs waiting for Partition 1
Partition 1-empty
b
Job queue for partition 2
No jobs waiting for Partition 2
Partition 2-empty
c
These jobs run only in Partition 3
Partition 3
In use
d
Job queue for partition 3

-To overcome the problem, the developers created relocating compilers, assemblers and loaders. These tools produce a Relocatable program that can run in any available partition that is large enough to hold that program.

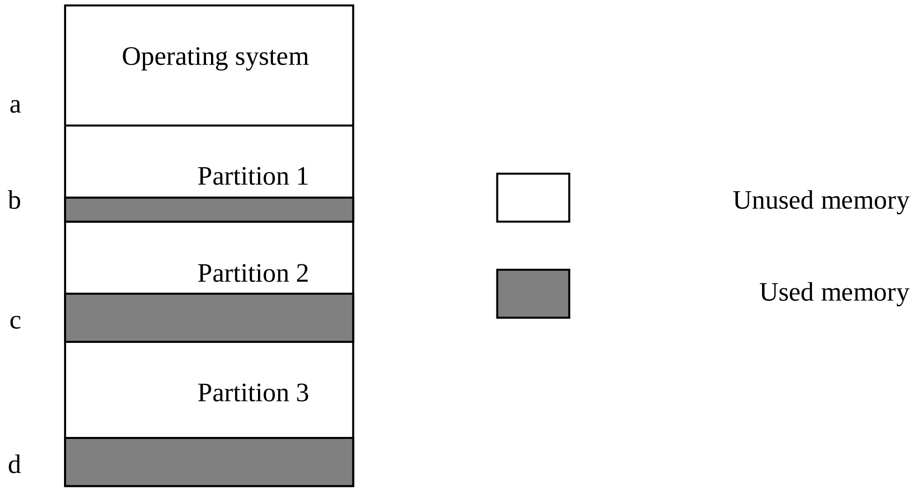


-Protection often is implemented with multiple boundary register. The system can delimit each partition with two boundary registers low and high, also called base and limit registers.



-When a process issues a memory request, the system checks whether the requested address is greater than or equal to the process's low boundary register value and less than the process's high boundary register value. If so, the system honors the request, otherwise, the system terminates the program with an error message.

-Fixed partition multiprogramming suffers from internal fragmentation, which occurs when the size of a process's memory and data is smaller than that of the partition in which the process executes.



-The system's three user partitions are occupied but each program is smaller than its corresponding partition. Consequently, the system may have enough main space in which to run another program but has no remaining partitions in which to run the program. Thus, some of the system's memory are wasted.