Memory

Physical Memory

So what happens when we can't find a suitable free space for a new process (even if we have GC)?

Memory

Physical Memory

So what happens when we can't find a suitable free space for a new process (even if we have GC)?

We may choose not to admit the process in the first place

Memory

Physical Memory

So what happens when we can't find a suitable free space for a new process (even if we have GC)?

We may choose not to admit the process in the first place

Another possibility is the option of killing existing processes: we usually don't want to and only if the new allocation is for a process that is sufficiently important

Memory

Physical Memory

So what happens when we can't find a suitable free space for a new process (even if we have GC)?

We may choose not to admit the process in the first place

Another possibility is the option of killing existing processes: we usually don't want to and only if the new allocation is for a process that is sufficiently important

Better is to *preempt* memory: take it away from one process and give it to another

Memory

Physical Memory

Remember that preemption takes a resource away from a process and returns it later in the same state

Memory

Physical Memory

Remember that preemption takes a resource away from a process and returns it later in the same state

For memory this means the bits in the memory when it is returned are unchanged from what they were when it was taken away

Memory

Physical Memory

Remember that preemption takes a resource away from a process and returns it later in the same state

For memory this means the bits in the memory when it is returned are unchanged from what they were when it was taken away

Even though that memory has been used by some other process and written its own data or code into it

Memory

Physical Memory

We can preempt a process and copy the contents of the memory it occupies to somewhere else: usually disk

Memory

Physical Memory

We can preempt a process and copy the contents of the memory it occupies to somewhere else: usually disk

Note that only the data need to be saved: the code is already on disk in the file that contains the program

Memory

Physical Memory

We can preempt a process and copy the contents of the memory it occupies to somewhere else: usually disk

Note that only the data need to be saved: the code is already on disk in the file that contains the program

Copying to disk is a (relatively) very slow operation: even the fastest disks are quite slow

Memory

Physical Memory

We can preempt a process and copy the contents of the memory it occupies to somewhere else: usually disk

Note that only the data need to be saved: the code is already on disk in the file that contains the program

Copying to disk is a (relatively) very slow operation: even the fastest disks are quite slow

Even solid state disks

Memory

Physical Memory

We can preempt a process and copy the contents of the memory it occupies to somewhere else: usually disk

Note that only the data need to be saved: the code is already on disk in the file that contains the program

Copying to disk is a (relatively) very slow operation: even the fastest disks are quite slow

Even solid state disks

So this kind of memory preemption has a large overhead

Memory

Physical Memory

We can preempt a process and copy the contents of the memory it occupies to somewhere else: usually disk

Note that only the data need to be saved: the code is already on disk in the file that contains the program

Copying to disk is a (relatively) very slow operation: even the fastest disks are quite slow

Even solid state disks

So this kind of memory preemption has a large overhead

This is a tradeoff of speed (time spent copying to and from disk) against process size (memory allocation)

Memory

Physical Memory

Swapping

The simplest case is preemption of the memory of an entire process

Memory

Physical Memory

Swapping

The simplest case is preemption of the memory of an entire process

When a process makes a request for an allocation that the OS cannot immediately satisfy the OS can try *swapping*

Memory

Physical Memory

Swapping

The simplest case is preemption of the memory of an entire process

When a process makes a request for an allocation that the OS cannot immediately satisfy the OS can try *swapping*

This is where one or more other processes are selected by the OS and they are copied out to disk to make space

Memory

Physical Memory

Swapping

The simplest case is preemption of the memory of an entire process

When a process makes a request for an allocation that the OS cannot immediately satisfy the OS can try *swapping*

This is where one or more other processes are selected by the OS and they are copied out to disk to make space

The best choice is usually a blocked process that couldn't have been run right now anyway

Memory

Physical Memory

When a swapped process is scheduled again it must be copied back by the OS into memory first

Memory

Physical Memory

When a swapped process is scheduled again it must be copied back by the OS into memory first

Which might require swapping out something else to make room

Memory

Physical Memory

When a swapped process is scheduled again it must be copied back by the OS into memory first

Which might require swapping out something else to make room

Data is retrieved from where it was saved, while code is copied back from the original program file — this is why some OS's don't like you deleting programs while they are running

Memory

Physical Memory

This differs from overlays in that it is the OS that does the swapping, not the process itself

Memory

Physical Memory

This differs from overlays in that it is the OS that does the swapping, not the process itself

This makes it transparent to the process and the programmer doesn't have to think about it

Memory

Physical Memory

This differs from overlays in that it is the OS that does the swapping, not the process itself

This makes it transparent to the process and the programmer doesn't have to think about it

. . . but they should as swapping is very time consuming, and slows down the speed of execution of programs immensely

Memory

Physical Memory

This differs from overlays in that it is the OS that does the swapping, not the process itself

This makes it transparent to the process and the programmer doesn't have to think about it

. . . but they should as swapping is very time consuming, and slows down the speed of execution of programs immensely

A good programmer will try to avoid the need for swapping by requesting memory allocations carefully

Memory

Physical Memory

This differs from overlays in that it is the OS that does the swapping, not the process itself

This makes it transparent to the process and the programmer doesn't have to think about it

. . . but they should as swapping is very time consuming, and slows down the speed of execution of programs immensely

A good programmer will try to avoid the need for swapping by requesting memory allocations carefully

Something that often is forgotten these days!

Memory

Physical Memory

The OS will take swapping into account when scheduling

Memory

Physical Memory

The OS will take swapping into account when scheduling

There is a clear interaction of scheduling and swapping processes: each will affect the other

Memory

Physical Memory

Variants:

Memory

Physical Memory

Variants:

- Only one process ever in memory, swapped as a whole when scheduled: simple, and used on very early systems

Memory

Physical Memory

Variants:

- Only one process ever in memory, swapped as a whole when scheduled: simple, and used on very early systems
- Swapping of processes: only marginally harder, and fits well with a partitioning system and fits well with scheduling

Memory

Physical Memory

Variants:

- Only one process ever in memory, swapped as a whole when scheduled: simple, and used on very early systems
- Swapping of processes: only marginally harder, and fits well with a partitioning system and fits well with scheduling
- Swapping *parts* of a process: not so easy as the OS has to work harder to determine which parts of a process's code or data might not be needed in the near future

Memory

Physical Memory

The OS still has the difficult task of deciding which process or processes to swap to make room: e.g., one large one or two small ones? I/O intensive or CPU intensive?

Memory

Physical Memory

The OS still has the difficult task of deciding which process or processes to swap to make room: e.g., one large one or two small ones? I/O intensive or CPU intensive?

An I/O intensive process is less likely to need to be scheduled soon; but it would a like fast response when it is needed and not wait for a slow swap back into memory

Memory

Physical Memory

The OS still has the difficult task of deciding which process or processes to swap to make room: e.g., one large one or two small ones? I/O intensive or CPU intensive?

An I/O intensive process is less likely to need to be scheduled soon; but it would a like fast response when it is needed and not wait for a slow swap back into memory

A CPU intensive process would like to be scheduled often; but is not so sensitive to a delay through being swapped