COSC 3750

Memory Management

Kim Buckner

University of Wyoming

Mar. 06, 2023

General info

- Physical memory layout is unique to each computer type.
- Can simply consider it a large array of bytes each with own address.
- Can ignore how an address is generated, only care about ordering.

Address Binding

- Scheduling input queue contains processes on disk waiting to be brought into memory.
- When programs are created, by the compiler,
  - most address will be bound to 'relocatable addresses' like x bytes from beginning of segment.
  - The linker/loader will then bind these to absolute addresses.

(more . . .)

- At compile time - can generate absolute address if you know what will happen at compile time: MSDOS .com programs.
- At load time - compiler generates relocatable code and binding is deferred to until load time.

(more . . .)

- At execution time - if process is moved, binding must be delayed until run-time. Most GP computers use this. Requires special hardware.

Logical versus Physical Address

- CPU generated addresses are logical whereas memory unit needs physical.
- This address is the one loaded into the memory address register
- Addresses bound at compile or load time are physical.
- Run time addresses are logical and are also called virtual.

(more . . .)

- Virtual address space is the set of all virtual addresses generated by a program.
  - 32 bit addresses: $2^{32} \equiv 0$ to $4,294,967,295$.
  - That is 4 Gibi bytes or 4.295 GBytes.
- Physical address space is the set of physical addresses that correspond to the virtual ones

(more . . .)

- Can use a relocation register to map one to another.
- This address is added to all process generated addresses.
- Maps logical to physical. DOS uses 4 of these ???
- User processes never know what physical address they are bound to.

Dynamic loading

- Routines are not loaded until called in the program.
- All kept on disk in relocatable form.
- Not managed by OS, up to user programs.
- Maybe DOS, maybe some others.

Dynamic linking and shared libraries

- Instead of loading being deferred, linking is.
- A stub is created which can cause the library routine to be accessed, linked and loaded when called.
- Then every time after that, the 'real' routine is accessed because the stub has overwritten itself.

(more . . .)

- Can be extended to include updates. New versions have version info and the original stubs can only access compatible versions.
- Dynamic linking requires help from the OS if processes are protected from each other (UNIX and Windows NT and up).

Overlays

- Allow processes to be larger than will fit into memory.
- When some section of code is no longer needed, or another IS needed, the needed section is loaded in the same place as the unneeded.
- Does not require OS help.
- Have to be a specially constructed program to work. DOS used overlays.

Swapping

- Processes need to be in memory to run but can be temporarily swapped out to disk (backing store) if NOT running.
- Requires that time quantum be large enough that real work is accomplished between swaps

(more . . .)

- A variant for priority scheduling is that lower priority processes can be swapped out when higher priority processes come in. (roll-in, roll-out)
- Where a process is placed when swapped in depends on when binding occurs
  - compile/link time requires that the process go back in the same space,
  - run-time means it can go anywhere.

(more . . .)

- Ready queue is all processes ready to run, whether on backing store or disk
- Dispatcher has to figure out where the process is and if on disk make arrangements to load it.
- Means that context switch time can be pretty high.

(more . . .)

- When we swap, the majority of the time involved is disk access.
- We want to swap only the minimum that a process is currently using.
- This means that if the process does dynamic memory allocation we should have memory *acquire*, memory *release* instructions for it.

(more . . .)

- Swapping requires that processes be idle, particularly any I/O is completed BEFORE we transfer it to disk
- This type of swapping we have been discussing is seldom used, too slow. But we do use modified versions.

(more . . .)

- Early PCs did not have the fancy hardware to fully support swapping. Win 3.1 supposedly supported concurrent processes but the decision was made by the user!
- Newer versions do noticeably better.

Contiguous Memory allocation

- Memory usually divided into two partitions, one for resident O/S (kernel) and one for user processes
- Major factor controlling O/S placement is interrupt vector, usually in low memory so we put kernel there too.

(more . . .)

- How to allocate memory so several processes can live there at once?
- Here each process is in a single contiguous section of memory.

Memory protection

- Provide by using a relocation register and limit register.
  - Relocation contains the starting memory address,
  - and limit contains the largest logical address.
- MMU maps from logical to physical

(more . . .)

- When the dispatcher loads a process it sets these registers and every address generated by the program is checked against them.
- This provides and effective way to let O/S change dynamically.

Memory allocation

- Multiprogramming with a Fixed number of Tasks (MFT) on IBM 360.
  - Divide memory into several fixed size partitions that can contain one process.

(more . . .)

- Multiprogramming with a Variable number of Tasks (MVT) also IBM, morphed to MFT2.
  - Allocate just enough memory to a process for it to run.
  - Keep track of remaining memory and allocate as new processes enter the system.

(more . . .)

- MVT continued:
  - As processes terminate, gather their now unused memory for reallocation.
  - If the current process on the input queue won't fit into a hole, then either wait or slide down the queue until one is found that can run.

(more . . .)

- This scheme requires that adjacent holes are coalesced.
- The dynamic storage-allocation problem.
  - first-fit - allocate the first hole that is big enough.
  - best-fit - smallest hole big enough.
  - worst fit - largest hole that is big enough.
  - *first* and *best* are better than *worst* in terms of time and storage utlization
  - *first* is not better than *best* in terms of utilization but is usually faster.

(more . . .)

- All suffer from external fragmentation.
- Which end of a free block is used?
- It is usually the case that if N blocks are allocated another .5N blocks are lost due to fragmentation. The 50 percent rule

Fragmentation

- External we just discussed.
- Internal –
  - if a process needs a block of size N and there is a block of size N+4, then the entire block may be allocated simply because it is too much trouble to keep track of such a small free space. T
  - This is internal fragmentation because the unusable space is inside an allocated block.
  - Can be fixed by paging

Paging

- Physical address space can be non-contiguous.
- Most systems today use it.
- It also helps reduce problems associated with fragmentation on the backing store.
- Traditionally handled by hardware but esp. in 64 bit O/S, is handled by software and hardware.