Lecture 5 – Memory Management

Karl R. Wilcox
Karl@cs.rhul.ac.uk

Royal Holloway University of London

Administration

- **There WILL be a lecture this Friday**
  - **In MFLT, 11:00**
- **Lecture Notes**
  - **Are available at http://www.cs.rhul.ac.uk/~karl**
  - **Formats**
    - **Original Powerpoint 2000 slides**
    - **PDF files, A4 2 slides per page, colour**
    - **Let me know if you would like other formats**

Royal Holloway University of London

Objectives

- In this class we will discuss:
  - Why manage memory
  - Memory allocation
  - Virtual memory
  - Paged memory
  - Page replacement algorithms

Royal Holloway University of London

The Memory Manager

- In our discussion of processes we have assumed that all concurrent processes fit simultaneously into memory
- This is not typically the case
  - (It may be on desktop or personal systems, but still need a memory manager)
- The memory manager is responsible for:
  - Recording which parts of memory are in use or free
  - Where to allocate memory for new processes
  - How to manage swapping between main memory and disk storage

Royal Holloway University of London

Memory Considerations

- **Swapping processes out to disk may be slow**
  - **therefore frequent swaps are inefficient**
- **We aim to keep the CPU busy**
  - **But if in-memory processes have a lot of I/O wait time CPU utilisation may be poor**
- **The longer the I/O wait time, the more processes should be in memory so at least one is ready to run**
  - **Assume processes spend a fraction *p* of time in I/O wait**
  - **There are *n* processes in memory at once**
  - **The probability that <u>all</u> processes are waiting is p^n**
  - **CPU utilisation is $1 - p^n$**

Royal Holloway University of London

Examples

- **Assume 100M of RAM, 20M for OS, user programs of 20M each**
  - **If each process spends 80% of its time in I/O wait:**
    - **CPU Utilisation = $1 - 0.8^4 = 59\%$**
  - **What if we had 90% I/O wait?**
    - **CPU Utilisation = $1 - 0.9^4 = 34\%$**
  - **If we add another 100M of memory (at 80% I/O)?**
    - **CPU Utilisation = $1 - 0.9^8 = 86\%$**
  - **If we add another 100M of memory (at 90% I/O)?**
    - **CPU Utilisation = $1 - 0.9^9 = 61\%$**
  - **A further 100M of memory?**
    - **80% I/O = 95%** **90% I/O = 77%**

Royal Holloway University of London

Memory Allocation Algorithms

- **The process of allocating main memory to processes**
  - **New processes and existing processes which need additional stack or data space**
- **Fixed size memory allocation**
  - **May not make best use of memory**
  - **Easy to manage**
- **Variable size memory allocation**
  - **Makes better use of memory**
  - **More complicated to manage**
- **In either case we need to track what is allocated and what is free**

Royal Holloway University of London

Bit Maps For Memory Allocation

- **Memory is divided in *allocation units***
  - **Can be a few words or several kilobytes**
- **For each allocation unit there is a bit in the map**
  - **The bit is 1 if the unit is occupied, 0 if it is free**
- **Simple way to keep track of memory**
  - **The size of the bit map depends only the size of the memory and the size of the allocation unit**
- **Main problem is searching the bit map for *k* consecutive bits**
  - **These may cross CPU word size boundaries**

Royal Holloway University of London

Linked Lists For Memory Allocation

- Maintain a linked list of memory segments
- Each segment is either allocated to a process
- List maintained in address order
- Updating list is straightforward
- Adjacent holes must be merged (to combine them into a single larger hole)
- Therefore it is easier if the list is doubly linked

Royal Holloway University of London

Link List Algorithms

- It is easy to find all available holes in the list
  - Which of them should be allocated?
- First Fit
  - Scan the list until one is found sufficiently large
- Next Fit
  - As first fit, but starting from where you left off
- Best Fit
  - Allocate smallest hole that the process fits in
- Worst Fit
  - Allocate largest available hole
- Quick Fit
  - Maintain separate lists for common sizes

Royal Holloway University of London

Buddy System For Memory Allocation

- The buddy system takes advantage of computers using binary numbers for addressing
- The manager maintains a list of free blocks of size 1, 2, 4, 8, 16, etc. bytes up to size of memory
- Initially, one hole the size of the whole memory
- To allocate space for a process which needs *x* bytes, find a hole of size 2^k, where $2^{k-1} < x <= 2^k$
- If necessary bigger holes are divided into two holes half as big

Royal Holloway University of London

Not Enough Memory

- **The previous discussion assumed that processes would fit into the available memory**
  - **Not always the case**
- **An early solution – Overlays**
  - **Overlay 0 runs first**
  - **When done calls next overlay into memory**
  - **Overlay splitting was done by the programmer**
  - **Performance depended on how well (how modular) the split was**
- **Replaced by the concept of virtual memory**

Royal Holloway University of London

Virtual Memory

- The combined size of the program, data, and stack may exceed the amount of physical memory available
- The operating system keeps those parts of the program currently in use in the main memory
- The rest is kept on disk
- Applicable to all the processes in a multi-programming system

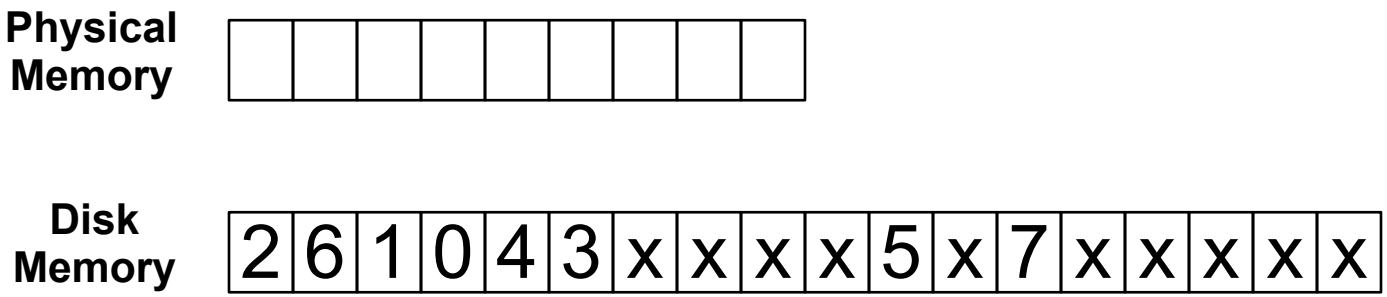
Royal Holloway University of London

Paging

- Virtual memory systems use the technique Paging
- On a computer there exists a set of memory addresses that programs can produce
- They are called virtual addresses
- They form the virtual address space
- These addresses go to a memory management unit (MMU)
- The MMU hardware maps the virtual addresses onto the physical memory address

Pages and Page Frames

- **The virtual space is divided into fixed sized pages**
- **The physical memory is divided into page frames of the same size**
- **Transfers between the two memories are always in units of a page**



Royal Holloway University of London

Page Faults

- Since the virtual memory is bigger than the physical memory not every page is mapped onto a page frame
- A present / absent bit in each entry keeps track of whether a page is mapped or not
- When the program tries to use an unmapped page it causes a trap to the operating system
- This trap is called a page fault
- The OS chooses one page frame, writes the contents back to disk and fetches the called page into the page frame, then updates the MMU map

Royal Holloway University of London

Page Tables

- **Page Tables are used for the mapping (virtual address → physical address) in the MMU**
- **The virtual address is split into a virtual page number and offset**
  - **Virtual page number = index into page table**
  - **Offset = last part of (virtual and physical) address**

- **The major issues:**
  - **The page table can be extremely large**
  - **The mapping must be fast**

Royal Holloway University of London

Multilevel Page Tables

- To avoid keeping all page tables in memory
  - Uses a top level page table
  - And second level page tables
- The virtual address is now partitioned into:
  - PT1 field (e.g. 10 bits)
  - PT2 field (e.g. 10 bits)
  - Offset field (e.g. 12 bits)
- The PT1 field indexes the top level page table
- There we get the address of the second level page table (indexed by the PT2 field)
- This table contains the physical address

Royal Holloway University of London

Associative Memory

- Most programs tend to make a large number of references to a small number of pages
- Therefore computers are equipped with a small hardware device for mapping virtual addresses to physical addresses without using page tables
- This device is called an associative memory
- Has a small number of entries, each about 1 page
- The OS looks up the page in the associative memory first
- If not present then the normal page tables are consulted

Royal Holloway University of London

Summary

- **We wish to have multiple processes in memory**
  - **So that at least one is ready to run**
- **Therefore memory must be shared between processes**
  - **Need to keep track of which memory is used by what**
- **We would also like to let processes run that need more memory than physically available**
  - **We use virtual paged memory to achieve this**

Royal Holloway University of London

Next Lecture

- **On Friday, MFLT**
- **We will talk about about page replacement algorithms**
- **Lecture Notes: http://www.cs.rhul.ac.uk/~karl**