Virtual Memory

COMP755 Advanced Operating Systems

Virtual Memory Performance

- Size of the Page
- Fetch Policy
- Placement Policy
- Cleaning policy
- Resident Set Management
- Replacement Policy
- Multiprogramming Level

Resident Set Management

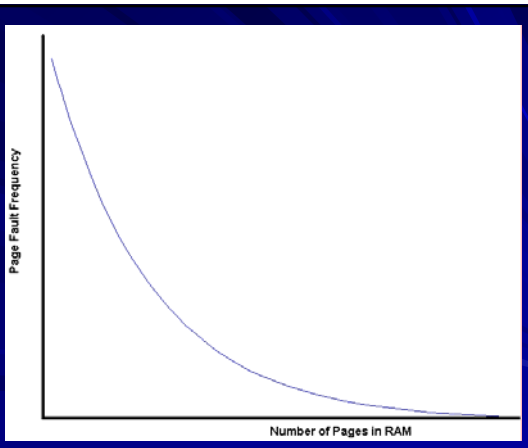
- Should all programs have an equal number of pages? If there are N programs running, should each program have 1/N of the total RAM?
- When a program gets a page fault and needs another page, should the new page replace only a page in that user's program or can it replace any page from any program?

Resident Set Management

- Fixed or variable number of pages per process.
- Pages replaced globally or within a process.
- Working Set Algorithm looks at the number of pages recently accessed to determine how many pages a program should have.

Allocating Pages

- If free pages are available, it is usually advantageous to give more pages to a program.
- When few or no free pages are available, giving pages to one program means taking pages from another.
- Taking pages from another program will likely increase that program's page fault rate.
- The goal is to reduce the total number of page faults in the system.



Page Fault Frequency Algorithm

- If a program is generating more page faults than some limit, give the program more pages in RAM.
- If the OS needs to take pages from a program, remove them from a program that is generating less than some minimum number of page faults.

Working Set Strategy

- For each program, the OS keeps resident all pages that have been accessed in the last Δ time units.
- Increasing the parameter Δ increases the number of pages.
- Usually implemented so that at a page fault the system keeps those pages that have been accesses in Δ time units.

Thrashing

- A system is said to be thrashing when it is spending so much of its resources maintaining the virtual memory that little actual work gets accomplished.
- As the number of running programs increases, each program gets fewer pages in RAM.
- With fewer pages in RAM, each program creates more page faults.

Replacement Policy

- What page should be overwritten?
- **Optimal** - requires looking forward in time. Useful for evaluating other algorithms.
- **LRU** - works well.
- **FIFO** - denies the concept of locality
- **Clock** - used by Mac OS.
- **Page Buffering** - System keeps a list of available and released pages.

Multiprogramming Level

- How many programs can run simultaneously?
- Swap out idle programs

Locked Pages

- Some pages cannot be paged out.
- Certain parts of the OS must always be in RAM (If the memory manager gets paged out, what will page it back in.)
- If a user is doing I/O to or from a page, that page must stay in RAM until the I/O is complete. The I/O controller won't know that another program is using the RAM.

Simulation

- Everybody gets a program *(pieces of paper)* of 16 pages labeled in hexadecimal.
- Select 4 places on your desk to represent four frames of RAM.
- As addresses are called, that page must be loaded into RAM (put on your desk) if not already there.
- Keep track of the number of page faults you generate.
- Assume RAM is initially empty.

The Program Accesses Page

5

- Since this is the first reference, it is sure to cause a page fault.
- Keep track of the page faults that occur.

The Program Accesses Page

5 4

The Program Accesses Page

5 4 7

The Program Accesses Page

5 4 7 5

The Program Accesses Page

5 4 7 5 6

The Program Accesses Page

5 4 7 5 6 8

The Program Accesses Page

5 4 7 5 6 8 A

The Program Accesses Page

5 4 7 5 6 8 A 5

The Program Accesses Page

5 4 7 5 6 8 A 5 7

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7 A

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7 A 6

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F

The Program Accesses Page

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

Results

- How many page faults did you get?
- Different replacement algorithms will cause page faults at different times.

LRU Algorithm

- The page that has not been referenced for the longest period of time is replaced.
- Locality of reference suggests that recently referenced pages will be referenced again.
- Implementation may be difficult since systems do not record the time of last reference.

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5			

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	4		

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	4	7	

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	4	7	

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	4	7	6

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	8	7	6

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	8	A	6

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	8	A	6

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	8	A	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	6	A	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	6	F	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	6	F	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	6	F	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	6	F	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	6	F	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	6	5	7

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	6	5	4

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	6	5	4

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	F	5	4

LRU Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

A	F	5	4

13 total page faults

Optimal Algorithm

- The optimal algorithm looks forward in time and replaces the page that will not be referenced again or will be referenced the farthest in the future.
- Cannot be implemented.
- Provides a theoretical minimum number of page faults

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5			

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

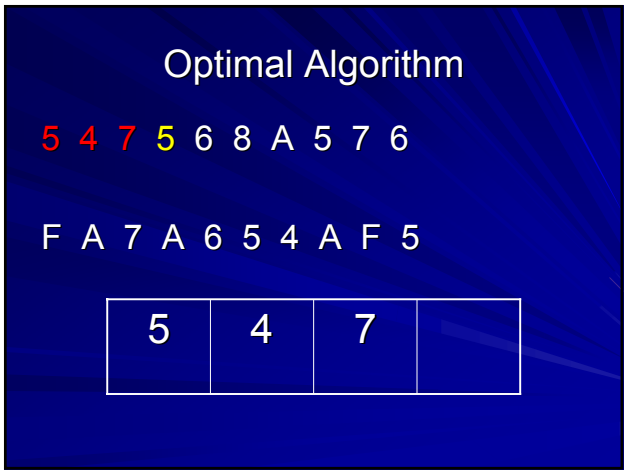
5	4		

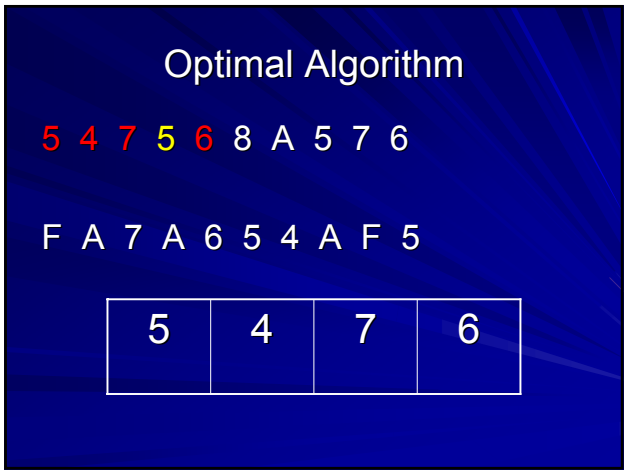
Optimal Algorithm

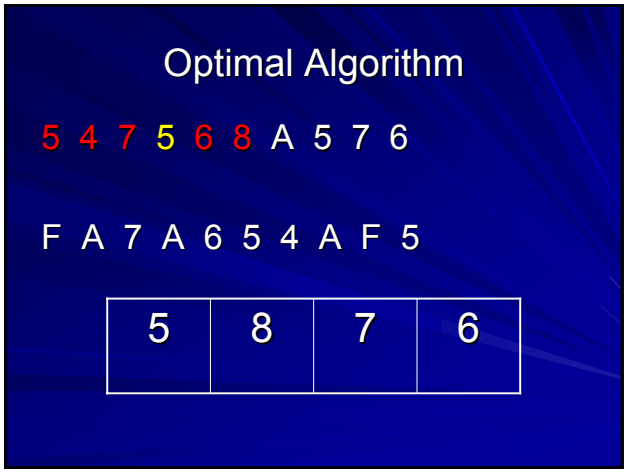
5 4 7 5 6 8 A 5 7 6

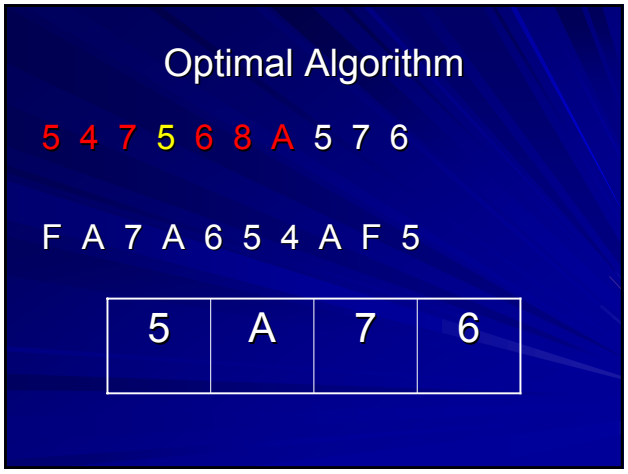
F A 7 A 6 5 4 A F 5

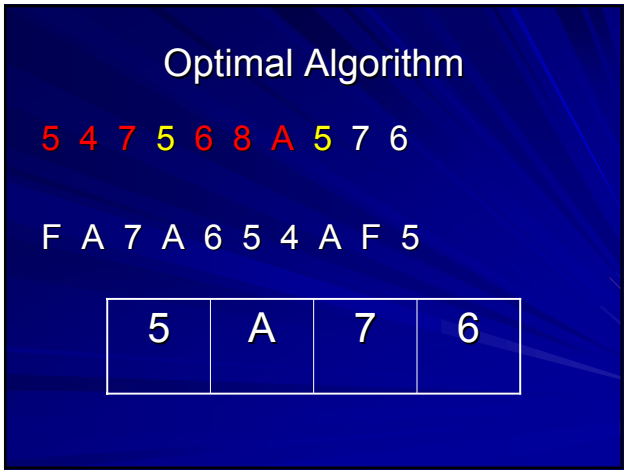
5	4	7	

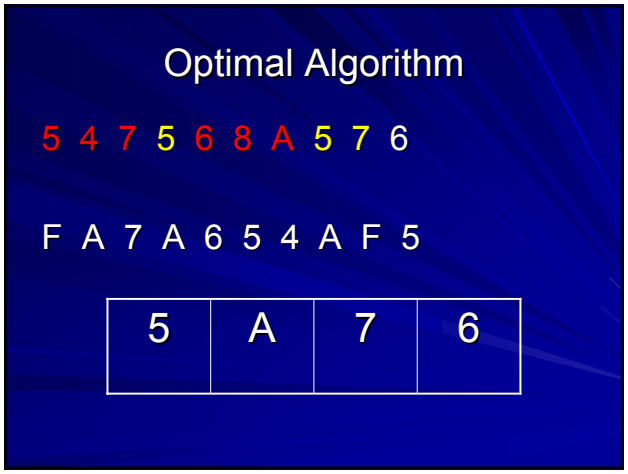

Optimal Algorithm
5 4 7 5 6 8 A 5 7 6
F A 7 A 6 5 4 A F 5
5 4 7


Optimal Algorithm
5 4 7 5 6 8 A 5 7 6
F A 7 A 6 5 4 A F 5
5 4 7 6


Optimal Algorithm
5 4 7 5 6 8 A 5 7 6
F A 7 A 6 5 4 A F 5
5 8 7 6


Optimal Algorithm
5 4 7 5 6 8 A 5 7 6
F A 7 A 6 5 4 A F 5
5 A 7 6


Optimal Algorithm
5 4 7 5 6 8 A 5 7 6
F A 7 A 6 5 4 A F 5
5 A 7 6


Optimal Algorithm
5 4 7 5 6 8 A 5 7 6
F A 7 A 6 5 4 A F 5
5 A 7 6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

5	A	7	6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	7	6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	7	6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	7	6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	7	6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	7	6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	5	6

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	5	4

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	5	4

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	5	4

Optimal Algorithm

5 4 7 5 6 8 A 5 7 6

F A 7 A 6 5 4 A F 5

F	A	5	4

9 Total Page Faults

Clock Algorithm

- Considers if a page has been referenced and if it has been modified.
- Unmodified pages are replaced before modified pages to avoid having to write the old page back to disk.
- Pages are kept in a circular queue with a pointer to the page after last replaced page.

Clock Algorithm

- Scan 1 – Beginning with the pointer, the pages are scanned for one that has not been referenced or modified. The first page found is replaced.
- Scan 2 – Starting from the pointer, look for a page that has not been referenced but has been modified. As the pages are checked, clear the referenced bit.

Clock Algorithm

- Scan 3 - Starting from the pointer, the pages are scanned for one that has not been referenced or modified.
- Scan 4 - Starting from the pointer, look for a page that has not been referenced but has been modified.

Page Buffering

- The OS maintains a list of free pages.
- Pages are taken from the free list whenever a new page is needed.
- When a page is replaced, it is put on the end of the free list if it has not been modified.
- If the page was modified, it is put on a modified list.

Page Buffering

- Pages on the modified list are written to disk when the system is idle. Cleaned pages are put on the free list.
- When a page fault occurs, the system checks these lists to see if the page is still in RAM.