

Page Management

- In a demand paging system ...
- Page fault occurs
 - A frame must be allocated
- Source of “new” frame
 - From free list, if available
 - Otherwise, replace a frame in use

Paging Algorithms

Two primary policy decisions

- Which page to replace?
 - Must unmap prior virtual page
 - And write to pagefile?
- Process frame allocation
 - How distribute among processes?
 - How many should be competing?

Evaluating Algorithms

- Minimize page-fault rate
- Test against reference stream
 - Sequence of memory addresses
 - Also called “reference string”
- Consider algorithm overhead

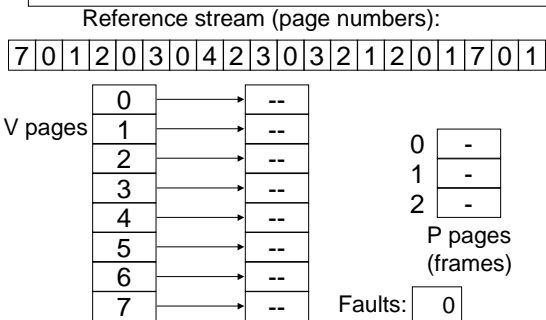
Reference Stream

- **Generation**
 - Random (not typical?)
 - Traces from actual programs
- **Resolution**
 - Don't need every memory address
 - Just sequence of virtual page numbers

FIFO Algorithm

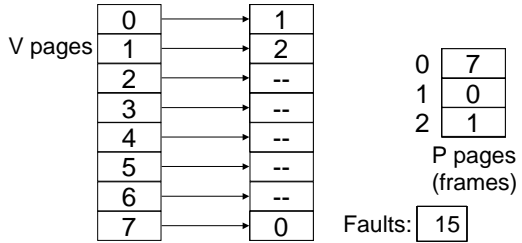
- **Allocate fixed # of frames**
 - To a particular process
- **If needed page present, OK**
- **If not, replace the "oldest"**
 - Write prior data to pagefile, if dirty
 - Remap frame to needed page & load

FIFO Example

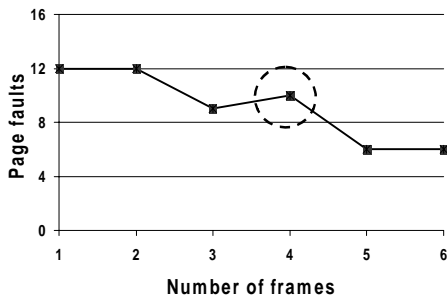


FIFO Example

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



Belady's Anomaly



Optimal Algorithm

- **Lowest page-fault rate**
- **Simple**
 - Replace the page that will not be used for the longest time
- **Small problem**
 - Requires predicting the future



LRU Algorithm

- **Least-recently used**
 - Replace it with new page
 - Optimal, except backwards in time!
- **Tracking**
 - Time stamp each page on reference
 - Keep list; move to head on reference

LRU Approximation

- **Housekeeping problem**
 - On every reference!
 - Can't do it in software
 - Hardware too expensive?
- **How about something close?**
 - Less difficult to implement

Reference Bit+



- **Keep page reference bit**
 - Set on every page access
 - Not just write like "dirty" bit
- **Periodically update history**
 - Set of bits in memory for each page
 - Shift in new bit, discard oldest

Second Chance

- **Variation of FIFO algorithm**
- **Select page to replace**
- **Test reference bit**
 - If clear, this is the victim
 - If set, clear and go to next in FIFO
 - If no new reference, victim next time

Counting Algorithms

- **Keep count of page accesses**
 - Since page was read in
- **Least frequently used (LFU)?**
- **Most frequently used (MFU)?**
- **Not very good choices in general**

Page Buffering

- **Keep pool of free frames**
- **On fault, choose victim**
 - New page gets pool page
 - Victim moved to pool after write
- **Variation**
 - Remember pool pages
 - Needed page may be there

Frame Allocation

- **Second paging algorithm**
- **How allocate frames?**
 - Among competing processes
- **How many processes?**
 - In active competition

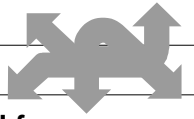
Allocation Strategies

- **One frame per process**
 - Fault continuously
- **All frames for one process**
 - No multiprogramming
- **Compromise**
 - Choose #processes, frames allocated

Allocation Options

- **Equal allocation**
- **Proportional allocation**
- **Global versus local**
 - Allocate frames from own process
 - Or from any process
 - Difficult to maintain predictability

Thrashing



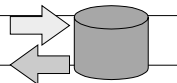
- If process lacks needed frames
- Fault removes “old” page
- Next fault needs it back
- Process stuck in page wait
- What if OS senses idle time?

Working Set

Relates to page fault frequency

- “Natural size” of a process
 - Optimum number of pages
- Too many page faults ($ws > F$)
 - Waste time in paging
- Too few page faults ($ws < F$)
 - Some pages not heavily used?

Swapping Revisited



- Need memory
 - For working sets of all processes
- What if not enough frames?
 - Sum of WS is greater than memory
- Remove some processes
 - Swap out to free pages for others

Swapping Criteria

- **Idle process**
 - More than 20 seconds? (BSD)
- **Large process**
 - Find largest few
 - Select longest resident

Page Locking



- **What if process has disk I/O?**
 - With DMA buffer in process memory
- **But buffer pages are replaced**
 - Because of other page faults
- **Need to lock pages in “core”**
 - For duration of I/O operation
