**Introduction**

The concept of virtual memory in operating systems refers to extending the physical memory (RAM) to the hard drive so that running processes behave as if there is unlimited address space. Paging means dividing the running process into equally sized logical pieces called pages so that only requested pages at the current execution time are only loaded into psychical memory. A page fault happens when a page is requested but it is not yet in physical memory. In this case the page has to be brought from disk into RAM. If there is no enough room for the page then an existing page in memory has to be flushed out back to disk. If the page that need to be removed is unmodified then we can just overwrite it but if it is modified (called dirty) then we need to copy it back to disk. It is always better not to remove an often used page that is when page replacement algorithms come into play. Here is a summary of some well known page replacement algorithms:

**Optimal Page Replacement Algorithm**

1. Replace the page that will be used furthest in the future
2. It is hard or impossible to implement

**Not Recently Used Page Replacement Algorithm (NRU)**

1. Classify pages into the following
    1. Not referenced Not dirty
    2. Not referenced and dirty
    3. Referenced Not dirty
    4. Referenced and dirty
2. Periodically clear reference bit for all pages
3. Remove a page from the lowest non empty class
4. Easy to implement and performance adequate

**First In First Out Page Replacement Algorithm (FIFO)**

1. As the name indicates first page in first out
2. It is easy to implement using a linked list
3. Bad, forces pages out regardless of usage

**Second Chance Page Replacement Algorithm**

1. It is easy to implement and better version of FIFO
2. If the page is not referenced then throw the page out
3. If the page is referenced then clear the reference bit and move the page to the tail of the list
4. Continue searching for a free page

**Clock Page Replacement Algorithm**

1. Simpler implementation compared to second chance algorithm
2. Clock hand points to the next page to replace
3. If the page is not referenced then we replace it
4. If the page is referenced then we advance the clock hand until a non referenced page is encountered

**Least Recently Used Page Replacement Algorithm (LRU)**

1. Throw a page that has been unused for longest time
2. In a linked list implementation put the most recently used pages at the front of the list and the least recently used pages at the rear
3. Update the list on every memory reference but that is an expensive operation
4. Near optimal performance in theory but rather expensive to implement in practice

**Not Frequently Used Page Replacement Algorithm (NFU)**

1. An approximation of LRU
2. At each clock interrupt scan the page table
3. If a page is referenced then increment the page reference counter
4. On replacement pick the page with the lowest counter value
5. Does not support page aging and tend to keep pages with high counter values

**Aging Page Replacement Algorithm**

1. Similar to NFU but page reference counter is reduced over time
2. Replace the page with lowest counter value

**Working Set Page Replacement Algorithm**

1. Bring a page into RAM when it is requested (demand paging)
2. A process references a small set of pages at any given time (Locality of reference)
3. The working set of a process is the set of pages used by the process in a given interval of time
4. So the goal of this algorithm is to keep most of the working set in memory in order to reduce the number of page faults

Please use the comments section below for questions, corrections or feedback. Thanks for reading.