Computer Engineering 571: Embedded Operating Systems

Programming Assignment 4: Filesystems

Skye Russ

Introduction:

An ideal page replacement algorithm will minimize the number of page faults, disk accesses and dirty page writing. In this assignment, five different page replacement algorithms were tested. These include, first in first out, random, least recently used, periodic reference reset, and future prediction. The future prediction algorithm was designed and developed by Skye Russ.

Implementation:

The main function takes command line inputs for the input file and the page replacement algorithms. Available page replacement algorithms are: Random (RAND), First in First Out (FIFO), Least Recently Used (LRU), Periodic Reference Reset (PER), and Future Prediction (FUT). The script initializes an array of main memory pages that store the process number, virtual page number, offset, dirty bit, reference bit, and last used time unit. These values are used for page replacement decisions.

The function then goes into the main control loop where it reads one line of the file, checks if it has the corresponding page loaded, and replaces the page as needed. The replacement is decided by a helper function, with each replacement algorithm having a corresponding helper function. Once the page to replace is selected, the function checks whether the page to be replaced is dirty and writes it back to the main memory as necessary.

Once all operations have been completed, the program outputs the number of operations completed, number of page faults, number of page hits, number of disk references, and number of dirty page writes. Below is the output from running both data1.txt and data2.txt for all scheduling algorithms.

Replacement Algorithm Descriptions:

First In First Out (FIFO):

The FIFO algorithm was implemented with a simple index counter that points to the index of the next page to be replaced. After the page is replaced, the index is incremented by one. When the index reaches the size of the main memory, the counter resets to zero. This allows the main memory to fill and stores the pages in order to allow for the oldest page to be replaced when any new page needs to be loaded.

Random (RAND):

The random algorithm uses the C "rand()" function which returns a random integer between 0 and 65535. The modulus of this value and the number of main memory pages is taken to allow for a random index between 0 and the number of main memory pages minus one. That value is then returned as the page to be replaced.

Least Recently Used (LRU):

The least recently used algorithm checks each page's last used time unit and finds the minimum. The page with the lowest last used time unit is selected for replacement. On the event of a tie, the page with the lower index value is replaced.

Periodic Reference Reset (PER):

Every 200 time units (200 pages referenced), all pages in the main memory have their reference bits set to 0. The algorithm chooses a page to be replaced based on five rules in ascending order. If any page fits the description of a lower-level rule, it is replaced with priority over the other pages. The rules are as follows:

1. An unused page (this will only happen early in your simulation).
2. An unreferenced page (reference bit is 0) where the dirty bit is 0.
3. An unreferenced page (reference bit is 0) where the dirty bit is 1.
4. A referenced page (reference bit is 1) where the dirty bit is 0.
5. A page that is both referenced (reference bit is 1) and dirty (dirty bit is 1).

Future Prediction (FUT):

The future prediction algorithm reads 100 pages in the future and stores their process number, virtual page number, and offset. The current set of main memory pages is then compared to these future pages, and each page is given a score based on the number of times it will be referenced in the next 100 pages. The page with the lowest number of references in the next 100 pages is selected for replacement. On the event of a tie, the page that was least recently used is selected to be replaced.

Experimental Data:

All algorithms were executed for the two data sets provided. They yielded these results.

Table 1: Data 1 Output

Algorithm	Page Faults	Disk Accesses	Dirty Page Writes
FIFO	3248	6309	3061
RAND	3552	6879	3327
LRU	3268	6328	3060
PER	3629	6951	3322
FUT	3057	5940	2883

Table 2: Data 2 Output

Algorithm	Page Faults	Disk Accesses	Dirty Page Writes
FIFO	2878	5625	2747
RAND	3202	6216	3014
LRU	2876	5596	2720
PER	3279	6318	3039
FUT	2714	5297	2583

Conclusion:

While many of the algorithms performed similarly, the Future Prediction algorithm was the clear winner. In both data sets, the future prediction algorithm had less page faults, less disk accesses, and less dirty page writes when compared to the other algorithms. In data set 1, the future prediction algorithm had 191 less page faults, 369 less disk accesses, and 178 less dirty page writes when compared to First in First Out (the next best algorithm for that data set). Interestingly, First in First Out minorly outperformed Least Recently Used and majorly outperformed Random and Periodic Reference Reset.

In the second data set, the future prediction algorithm also outperforms all other algorithms. It has 162 less page faults, 299 less disk accesses, and 137 less dirty page writes when compared to the Least Recently Used algorithm.

It is notable that the random replacement algorithm could potentially outperform the future prediction algorithm. This would only be the case if the random algorithm chose the exact same pages to replace as the future prediction algorithm. The algorithms would then have the same number of page faults, disk accesses and dirty page writes, but the random replacement algorithm would have taken less overhead time to calculate which page to replace.