CS39002 OPERATING SYSTEMS LABORATORY
SPRING 2025

LAB ASSIGNMENT: 9
DATE: 26–MAR –2025

Demand paging with swapping (no page replacement)

The system

Think of an embedded computing system having 64 MB memory and supporting 4 KB pages. The OS reserves 16 MB of the memory, so 48 MB / 4 KB = 12288 frames are available for user processes. The processes in the device are of a specific type. Each process uses 10 pages for storing the *essential segments* (code, global variables, and stack). There is an *additional data segment* that is meant for storing a read-only array A of s integers, where $s \in [10^6, 2 \times 10^6]$. Assuming 32-bit integers, the frame requirement of this data segment lies in the range [977, 1953]. The OS uses a virtual memory with 2048 4-KB pages for each process, that is, the page table of each process has 2048 entries. The machine has a total of 64 MB / 4 KB = 16384 = 2^{14} frames. Moreover, we also maintain a valid/invalid bit in each entry of the page table. Consequently, 15 bits suffice for each page-table entry. However, computers do not like sizes that are not powers of 2, so each page-table entry is a 16-bit unsigned short int. The most significant bit is used as the valid/invalid bit.

When a process arrives, the 10 pages in the essential segments are loaded to memory. The array A to be stored in the additional segment resides in the hard disk. Pages from this array are loaded to memory on demand. Pages once loaded continue to stay in the memory unless the process terminates or is swapped out. In order to make the swapping operation efficient, only the 10 essential pages are swapped out. The array A is assumed to be read-only, so the pages of A do not change during the run of the process. After a swap-in event, however, all the pages of A that are needed in future must be reloaded from the disk.

The computer starts with n user processes. Each process carries out a sequence of m binary searches using the following algorithm (x is the search key). Assume that m is constant over all the processes. The scheduler dispatches the processes in a round-robin fashion. Each time quantum consists of a single binary search.

```
L = 0; R = s - 1;
while (L < R) {
       M = (L + R) / 2;
       if (x <= A[M]) R = M;
       else L = M + 1;
}
```

The simulation

In this assignment, you make a single-process (single-thread) simulation of the computer in order to measure the performance of this memory-management scheme.

To start with, load all the n processes to memory, giving each of them only the 10 essential frames. As searches are carried out by the processes, the relevant pages of A are loaded to memory frames. Assume that the array A is different for different processes, that is, each individual process needs to maintain the pages of its own private array A. If there are many processes, the loaded pages of the arrays A eventually eat up all the frames in the main memory. When an ongoing search needs to load a new page from its A and the memory is already full, that process is swapped out, and all the frames (essential and additional) allocated to that process are returned to the list of free frames. Alongside a list of free frames, a list of swapped out processes too is to be maintained by the simulation.

When a process P finishes all of its m searches, it quits. All the frames allocated to P are returned to the list of free frames. After that, a check is done whether the list of swapped-out processes contains any entries. If

not, the simulation continues to the next search (of the next ready process). Otherwise, a *single* waiting process Q from the list of swapped-out processes is taken. Q is swapped in by allocating to it only the 10 essential frames. The search of Q, which swapped it out, is restarted before any other search of a ready process. Since all the frames allocated to P are now available, it is expected that this search of Q will succeed without a need of a swap-out event again. Notice that the memory freed by P may be temporarily sufficient to restart multiple swapped-out processes. But this is perhaps not a good idea, because a second restarted process Q' (possibly Q too) may soon encounter the memory-full situation and will have to be swapped out again. It is safer to restart Q' when another process P' terminates.

The input and the simulation data

The input is supplied in a text file *search.txt*. The file starts with n (the number of processes) and m (the number of searches of each process). You may assume that $50 \leqslant n \leqslant 500$ and $10 \leqslant m \leqslant 100$. The input file then contains n lines, one for each process. Each line contains (exactly) $m + 1$ integers. The first entry in a line is the size s of the array A for the process. This is followed by m indices $k_0, k_1, k_2, \ldots, k_{m-1}$ in the array A (so each k_j is in the range $[0, s - 1]$).

The simulation data consists of the contents of the file *search.txt* along with some other information (like how many searches have already been carried out by each of the n processes).

A program *gensearch.c* for generating random input samples is provided to you. Supply n and m as its two command-line parameters. The default values are $n = 200$ and $m = 100$. Notice that you are not going to initiate n processes or threads for simulating the behavior of the n processes. So a large value of n is not a burden to your system. However, it is not recommended to take very large values of m, because we do not want all or almost all pages of each A to be loaded to memory.

Simulating the binary search

What about the array A and its pages? Well, you actually do not need a real array for this simulation. Each k in the line of a process in the input file indicates an index in A, where the search terminates. This search must follow the binary-search algorithm mentioned on the previous page. You may pretend that $A[i] = i$ for all i, and you are searching for k, so the condition `x <= A[M]` is simulated by `k <= M`. However, for the sake of the simulation, pretend that `A[M]` is accessed. If the page containing `A[M]` is already loaded to the memory (look at the valid/invalid bit in the page table of the process), proceed to the next iteration of the binary-search loop. Otherwise, this simulates a page fault. Try to *load* the desired page from A by allocating a free frame, and updating the corresponding entry in the page table of the process. Then go to the next iteration of the binary-search loop. If no free frames are available, simulate a swap-out operation.

Maintaining kernel data

In addition to the simulation data, you need to maintain a set of items for simulating the working of the kernel. First, you need to maintain the page tables of each process. Use your own implementation of this table. Recall that each page table is an array of 2048 unsigned short integers. The MSB (15-th bit) of each element of this array is to be used as the valid/invalid bit. Use bit-wise operations to set/clear/retrieve these bits. No other implementation is allowed.

The kernel data also consists of three lists: the ready queue, the list of free frames, and the list of swapped-out processes. Implement each of these lists as a FIFO queue. You may use ready-made library implementations. The ready queue is served in a round-robin fashion, so the next process to be scheduled is extracted from the front of the queue, and after a successful search, that process is added to the back of the queue. The other two lists need not be maintained as FIFO queues. But follow this strictly, particularly, for the list of swapped-out processes. This means that processes are swapped in in the same order as they are swapped out, a natural strategy indeed.

In addition, you maintain some items as kernel data in order to print certain performance figures at the end of the simulation. This includes the number of page accesses (accesses of `A[M]`), the number of page faults encountered during these accesses, and the number of swaps used in the simulation.

You also compute the *degree of multiprogramming* achieved by the simulation. Without demand paging, this is 12288 / 2048 = 6. With demand paging, this increases by a significant factor. Of course, the degree of multiprogramming is a function of m (the number of searches per process), and should decrease with increasing m. If m is too large, then nearly all pages of A will be loaded to the memory. For an average size of 1.5×10^6 of A, this will give a degree of multiprogramming close to 9.

In your simulation, keep track of the number of active (running and not swapped out) processes. Take a minimum of these counts at all times when the memory is full (that is, a swap-out operation is simulated). This minimum will be the degree of multiprogramming for the simulation.

Output

In the non-verbose mode, print only the swap-out and swap-in operations, and the final statistics. In the verbose mode, additionally print the searches carried out in the simulation (process numbers and search numbers). Use a compile-time flag `VERBOSE` to switch between the two modes. You may use the following makefile.

```
run: demandpaging.c
        gcc -Wall -o runsearch demandpaging.c
        ./runsearch

vrun: demandpaging.c
        gcc -Wall -DVERBOSE -o runsearch demandpaging.c
        ./runsearch

db: gensearch.c
        gcc -Wall -o gensearch gensearch.c
        ./gensearch

clean:
        -rm -f runsearch gensearch
```

Submit a single C/C++ source file demandpaging.c(pp).

Sample Output

The non-verbose output for a sample run with $n = 128$ and $m = 64$ is given below (in a two-column format). The corresponding input file (*search.txt*) and the verbose output are provided to you in a separate archive.

```
$ make run
gcc -Wall -o runsearch demandpaging.c
./runsearch
+++ Simulation data read from file
+++ Kernel data initialized
+++ Swapping out process   57  [127 active processes]
+++ Swapping out process   75  [126 active processes]
+++ Swapping out process   93  [125 active processes]
+++ Swapping out process  107  [124 active processes]
+++ Swapping out process  122  [123 active processes]
+++ Swapping out process   12  [122 active processes]
+++ Swapping out process   36  [121 active processes]
+++ Swapping out process   53  [120 active processes]
+++ Swapping out process   74  [119 active processes]
+++ Swapping out process   95  [118 active processes]
+++ Swapping out process  114  [117 active processes]
+++ Swapping out process    7  [116 active processes]
+++ Swapping out process   31  [115 active processes]
+++ Swapping out process   56  [114 active processes]
+++ Swapping out process   76  [113 active processes]
+++ Swapping out process   92  [112 active processes]
+++ Swapping out process  118  [111 active processes]
+++ Swapping out process   13  [110 active processes]
+++ Swapping out process   33  [109 active processes]
+++ Swapping out process   55  [108 active processes]
+++ Swapping out process   86  [107 active processes]
+++ Swapping out process  111  [106 active processes]
+++ Swapping out process    5  [105 active processes]
+++ Swapping out process   29  [104 active processes]
+++ Swapping out process   60  [103 active processes]
+++ Swapping out process   87  [102 active processes]
+++ Swapping out process  117  [101 active processes]
+++ Swapping out process   20  [100 active processes]
+++ Swapping out process   58  [99 active processes]
+++ Swapping out process   91  [98 active processes]
+++ Swapping out process  125  [97 active processes]
+++ Swapping out process   32  [96 active processes]
+++ Swapping out process   66  [95 active processes]
+++ Swapping out process  102  [94 active processes]
+++ Swapping out process    4  [93 active processes]
+++ Swapping out process   44  [92 active processes]
+++ Swapping out process   82  [91 active processes]
+++ Swapping out process  126  [90 active processes]
+++ Swapping out process   38  [89 active processes]
+++ Swapping out process   71  [88 active processes]
+++ Swapping out process  109  [87 active processes]
+++ Swapping out process   22  [86 active processes]
+++ Swapping out process   62  [85 active processes]
+++ Swapping out process  105  [84 active processes]
+++ Swapping out process   37  [83 active processes]
+++ Swapping out process   96  [82 active processes]
+++ Swapping out process   23  [81 active processes]
+++ Swapping out process   79  [80 active processes]
+++ Swapping out process    6  [79 active processes]
+++ Swapping out process   61  [78 active processes]
+++ Swapping out process  119  [77 active processes]
+++ Swapping out process   41  [76 active processes]
+++ Swapping out process  103  [75 active processes]
+++ Swapping out process   28  [74 active processes]
+++ Swapping out process   88  [73 active processes]
+++ Swapping out process   15  [72 active processes]
+++ Swapping out process   78  [71 active processes]
+++ Swapping out process   14  [70 active processes]
+++ Swapping out process   83  [69 active processes]
+++ Swapping out process   39  [68 active processes]
+++ Swapping out process  123  [67 active processes]
+++ Swapping out process   80  [66 active processes]
+++ Swapping out process   30  [65 active processes]
+++ Swapping out process  116  [64 active processes]
+++ Swapping out process   72  [63 active processes]
+++ Swapping out process   40  [62 active processes]
+++ Swapping out process    3  [61 active processes]
+++ Swapping out process   99  [60 active processes]
+++ Swapping out process   69  [59 active processes]
+++ Swapping out process   68  [58 active processes]
+++ Swapping out process   49  [57 active processes]
+++ Swapping out process   42  [56 active processes]
+++ Swapping out process   43  [55 active processes]
+++ Swapping out process   48  [54 active processes]
+++ Swapping out process   50  [53 active processes]
+++ Swapping out process   64  [52 active processes]
+++ Swapping out process  104  [51 active processes]
+++ Swapping out process   11  [50 active processes]
+++ Swapping out process   46  [49 active processes]
+++ Swapping out process  106  [48 active processes]
+++ Swapping out process   24  [47 active processes]
+++ Swapping out process   81  [46 active processes]
+++ Swapping out process   45  [45 active processes]
+++ Swapping out process    8  [44 active processes]
+++ Swapping out process  112  [43 active processes]
+++ Swapping out process   26  [42 active processes]
+++ Swapping out process  120  [41 active processes]
+++ Swapping out process  121  [40 active processes]
+++ Swapping in process    57  [40 active processes]
+++ Swapping in process    75  [40 active processes]
+++ Swapping in process    93  [40 active processes]
+++ Swapping in process   107  [40 active processes]
+++ Swapping in process   122  [40 active processes]
+++ Swapping in process    12  [40 active processes]
+++ Swapping in process    36  [40 active processes]
+++ Swapping in process    53  [40 active processes]
+++ Swapping in process    74  [40 active processes]
+++ Swapping in process    95  [40 active processes]
+++ Swapping in process   114  [40 active processes]
+++ Swapping in process     7  [40 active processes]
+++ Swapping in process    31  [40 active processes]
+++ Swapping in process    56  [40 active processes]
+++ Swapping in process    76  [40 active processes]
+++ Swapping in process    92  [40 active processes]
+++ Swapping in process   118  [40 active processes]
+++ Swapping in process    13  [40 active processes]
+++ Swapping in process    33  [40 active processes]
+++ Swapping in process    55  [40 active processes]
+++ Swapping in process    86  [40 active processes]
+++ Swapping in process   111  [40 active processes]
+++ Swapping in process     5  [40 active processes]
+++ Swapping in process    29  [40 active processes]
+++ Swapping in process    60  [40 active processes]
+++ Swapping in process    87  [40 active processes]
+++ Swapping in process   117  [40 active processes]
+++ Swapping in process    20  [40 active processes]
+++ Swapping in process    58  [40 active processes]
+++ Swapping in process    91  [40 active processes]
+++ Swapping in process   125  [40 active processes]
+++ Swapping in process    32  [40 active processes]
+++ Swapping in process    66  [40 active processes]
+++ Swapping in process   102  [40 active processes]
+++ Swapping in process     4  [40 active processes]
+++ Swapping in process    44  [40 active processes]
+++ Swapping in process    82  [40 active processes]
+++ Swapping in process   126  [40 active processes]
+++ Swapping in process    38  [40 active processes]
+++ Swapping in process    71  [40 active processes]
+++ Swapping in process   109  [40 active processes]
+++ Swapping in process    22  [40 active processes]
+++ Swapping in process    62  [40 active processes]
+++ Swapping in process   105  [40 active processes]
+++ Swapping in process    37  [40 active processes]
+++ Swapping in process    96  [40 active processes]
+++ Swapping in process    23  [40 active processes]
+++ Swapping in process    79  [40 active processes]
+++ Swapping in process     6  [40 active processes]
+++ Swapping in process    61  [40 active processes]
+++ Swapping in process   119  [40 active processes]
+++ Swapping in process    41  [40 active processes]
+++ Swapping in process   103  [40 active processes]
+++ Swapping in process    28  [40 active processes]
+++ Swapping in process    88  [40 active processes]
+++ Swapping in process    15  [40 active processes]
+++ Swapping in process    78  [40 active processes]
+++ Swapping in process    14  [40 active processes]
+++ Swapping in process    83  [40 active processes]
+++ Swapping in process    39  [40 active processes]
+++ Swapping in process   123  [40 active processes]
+++ Swapping in process    80  [40 active processes]
+++ Swapping in process    30  [40 active processes]
+++ Swapping in process   116  [40 active processes]
+++ Swapping in process    72  [40 active processes]
+++ Swapping in process    40  [40 active processes]
+++ Swapping in process     3  [40 active processes]
+++ Swapping in process    99  [40 active processes]
+++ Swapping in process    69  [40 active processes]
+++ Swapping in process    68  [40 active processes]
+++ Swapping in process    49  [40 active processes]
+++ Swapping in process    42  [40 active processes]
+++ Swapping in process    43  [40 active processes]
+++ Swapping in process    48  [40 active processes]
+++ Swapping in process    50  [40 active processes]
+++ Swapping in process    64  [40 active processes]
+++ Swapping in process   104  [40 active processes]
+++ Swapping in process    11  [40 active processes]
+++ Swapping in process    46  [40 active processes]
+++ Swapping in process   106  [40 active processes]
+++ Swapping in process    24  [40 active processes]
+++ Swapping in process    81  [40 active processes]
+++ Swapping in process    45  [40 active processes]
+++ Swapping in process     8  [40 active processes]
+++ Swapping in process   112  [40 active processes]
+++ Swapping in process    26  [40 active processes]
+++ Swapping in process   120  [40 active processes]
+++ Swapping in process   121  [40 active processes]
+++ Page access summary
          Total number of page accesses  =  169315
          Total number of page faults    =  42727
          Total number of swaps          =  88
          Degree of multiprogramming     =  40
$
```