Big Picture – CPU, Memory, and Pages:

Page Eviction/Replacement Strategies:
When we need to remove a page from RAM and store it on disk, how do we decide which page to remove given a page access pattern?

Strategy #1:

<table>
<thead>
<tr>
<th>17</th>
<th>33</th>
<th>40</th>
<th>17</th>
<th>43</th>
<th>8</th>
<th>99</th>
<th>33</th>
<th>99</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Strategy #2:

<table>
<thead>
<tr>
<th>17</th>
<th>33</th>
<th>40</th>
<th>17</th>
<th>43</th>
<th>8</th>
<th>99</th>
<th>33</th>
<th>99</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Strategy #3:

<table>
<thead>
<tr>
<th>17</th>
<th>33</th>
<th>40</th>
<th>17</th>
<th>43</th>
<th>8</th>
<th>99</th>
<th>33</th>
<th>99</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Strategy #4:

<table>
<thead>
<tr>
<th>17</th>
<th>33</th>
<th>40</th>
<th>17</th>
<th>43</th>
<th>8</th>
<th>99</th>
<th>33</th>
<th>99</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other Strategies:

Fragmentation
As we develop various systems for storage, we want to minimize fragmentation.

- [Fragmentation]:
- [Internal Fragmentation]:
- [External Fragmentation]:

Fragmentation Example in Heap Memory:

Unallocated (3072 bytes)

Used (4096 bytes)

Free (3072 bytes)

Used Data (2048 bytes)

Start of Heap

End of Heap
Computer Peripherals

- Every other piece of hardware we consider to be a “peripheral”.
- Interface managed by the _______________________.
  
  ○ ...and managed using _____________________.
- Examples:

Threads: The Unit of Computation in an Operating System

As a programmer, the single most important construct in an Operating System is a thread.

- Every thread has a **program counter**, a pointer that stores the next instruction to be read by a program.
- A _____________ is an organization of one or more threads in the same context. A simple process has only one thread.
- In C, the initial thread is called the _______________.
  
  ○ It is what starts running your main() function!

Creating Additional Threads in C

The pthread library is the POSIX thread library allowing you to create additional threads beyond the main thread.

Creating a new thread is a complex call with four arguments:

```c
#include <pthread.h>
const int num_threads = 15;

void *thread_start(void *ptr) {
    int id = *((int *)ptr);
    printf("Thread %d running...\n", id);
    return NULL;
}

int main(int argc, char *argv[]) {
    // Create threads:
    int i;
    pthread_t tid[num_threads];
    for (i = 0; i < num_threads; i++) {
        pthread_create(&tid[i], NULL,
                        thread_start, (void *)&i);
    }
    printf("Done!\n");
    return 0;
}
```

Q1: What is the expected output of this program?

Q2: What actually happens?

Q3: What do we know about threads in C?