Segmentation

CSE 2431: Introduction to Operating Systems
Reading: §8.6, [OSC]
Contents

• Page Replacement (Review)
  – Working Set
  – WSClock
• Segmentation
• Segmentation + Paging
Miss Ratio Curve

Graph showing the relationship between Page Fault Rate and Number of Page Frames. The graph indicates that as the working set becomes too small or too large, the page fault rate increases. A reasonable working set is shown to produce a low and stable page fault rate.

Page Rate for Single Process
Reasonable Working Set
Working Set too large
WSClock Page Replacement Algorithm

- Carr and Hennessey, 1981
- Used very widely (Linux)
- Circular list as in the clock algorithm.
  - Initially, list is empty.
  - As pages are loaded into memory, pages are added to the list
  - Each entry contains *Time of last use* as well as reference and dirty bits.

**Algorithm:**
- At each page fault, the page pointed to by the clock hand is examined.
- Repeat the following:
  - If reference bit is 1, then the page has been reference during the current tick, so it is in working set and not ideal candidate to remove. Clear the reference bit and update the time of last use.
  - If reference bit is 0, then check if it is in working set window (i.e., if *Current Time* minus *Time of last use* is less than the working set window size time).
  - If page is not in the working set and the page is clean, then replace it.
  - If the page is not in working set and page is dirty, request write to disk, and move on to the next page in the circular list.
Virtual Memory So Far

• One-dimensional address space

• An example: the compiling process

• Need a way of freeing programmers from managing expanding and contracting

<table>
<thead>
<tr>
<th>Call Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse Tree</td>
</tr>
<tr>
<td>Constant Table</td>
</tr>
<tr>
<td>Source Text</td>
</tr>
<tr>
<td>Symbol Table</td>
</tr>
</tbody>
</table>
Segmentation

• Segment is a logical unit
  – Programmer is aware of and uses as a logical unit
  – E.g., Code, Heap, Stack

• Advantages:
  – Easy to handle dynamically expanded/shrunked data structures
  – Linking is simplified if each procedure occupies a separate segment
  – Facilitating sharing procedures/data
  – Different protection for different segments
User’s View of Program
Logical View of Segmentation

- Similar to variable partition
- First fit, best fit, or worst fit.
- External fragmentation
# Paging vs. Segmentation

<table>
<thead>
<tr>
<th>Consideration</th>
<th>Paging</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Must the programmer be aware that this technique is being used?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>How many linear address spaces are there?</td>
<td>1</td>
<td>Many</td>
</tr>
<tr>
<td>Can the total address space exceed the size of physical memory?</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can procedures and data be distinguished and separately protected?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can tables whose size fluctuates be accommodated easily?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Is sharing of procedures between users facilitated?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Why was this technique invented?</td>
<td>To get a large linear address space without having to buy more physical memory</td>
<td>To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection</td>
</tr>
</tbody>
</table>
Segmentation Architecture

- Logical address: 
  \(<\text{segment-number, offset}>\)
- Each process has one segment table with each entry for one segment
- Each table entry has
  - Base: contains the starting physical address where the segment reside in memory
  - Limit: specifies the length of the segment
- Segment-table base register (STBR)
  - Points to the segment table’s location in memory
- Segment-table length register (STLR)
  - Indicates number of segments used by a program
- Segment number $s$ is valid if $s < \text{STLR}$
An Example of Segmentation
Segmentation Address Mapping

```
CPU  s  d  \rightarrow  \begin{array}{cc}
\text{limit} & \text{base} \\
\end{array}
\rightarrow  \quad \text{segment table}
\rightarrow  \quad <
\rightarrow  \quad \begin{array}{c}
\text{yes} \\
\text{no} \\
\end{array}
\rightarrow  \quad +
\rightarrow  \quad \text{trap; addressing error}
\rightarrow  \quad \text{physical memory}
```
Sharing of Segments

- **Logical memory process $P_1$**
  - Segment 0
  - Segment 1
  - Data 1

- **Logical memory process $P_2$**
  - Segment 0
  - Segment 1
  - Data 2

**Segment Table**

- **Process $P_1$**
  - Limit: 25286
  - Base: 43062
- **Process $P_2$**
  - Limit: 25286
  - Base: 43062

**Physical Memory**

- Editor: 43062
- Data 1: 68348
- Data 2: 90003
- Process $P_1$: 72773
- Process $P_2$: 98553
Intel IA-32 Address Translation

logical address

selector

offset

descriptor table

segment descriptor

+

linear address

directory

page

offset

page frame

physical address

page directory

directory entry

page table

page table entry

page directory base register
Summary

- WSClock Page Replacement
- Segmentation
- Segmentation + Paging