Announcements

- **Project #2 is available on the web**
  - Atoi.c has been posted
  - Reference solution to project #1, delete loop in Print_String function inside libuser.c

- **Midterm #1 was returned**
  - Solution on web
  - Must submit requests for re-grades in writing by 3/19/02

<table>
<thead>
<tr>
<th>Grade breakdown</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>12</td>
<td>8</td>
<td>5</td>
<td>13</td>
<td>9</td>
<td>9</td>
<td>56.2</td>
</tr>
<tr>
<td>Min</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>Max</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>84</td>
</tr>
</tbody>
</table>
Sharing Memory

- **Pages can be shared**
  - several processes may share the same code or data
  - several pages can be associated with the same page frame
  - given read-only data, sharing is always safe

- **when writes occur, decide if processes share data**
  - operating systems often implement “copy on write” - pages are shared until a process carries out a write
    - when a shared page is written, a new page frame is allocated
    - writing process owns the modified page
    - all other sharing processes own the original page
  - page could be shared
    - processes use semaphores or other means to coordinate access
What Happens when a virtual address has no physical address?

- **called a page fault**
  - a trap into the operating system from the hardware
- **caused by: the first use of a page**
  - called *demand paging*
  - the operating system allocates a physical page and the process continues
  - read code from disk or init data page to zero
- **caused by: a reference to an address that is not valid**
  - program is terminated with a “segmentation violation”
- **caused by: a page that is currently on disk**
  - read page from disk and load it into a physical page, and continue the program
- **caused by: a copy on write page**
OS Protection attributes (Win32)

- NOACCESS: attempts to read, write or execute will cause an access violation
- READONLY: attempts to write or execute memory in this region cause an access violation
- READWRITE: attempts to execute memory in this region cause an access violation
- EXECUTE: Attempts to read or write memory in this region cause an access violation
- EXECUTE_READ: Attempts to write to memory in this region cause an access violation
- EXECUTE_READ_WRITE: Do anything to this page
- WRITE_COPY: Attempts to write will cause the system to give a process its own copy of the page. Attempts to execute cause access violation
- EXECUTE_WRITE_COPY: Attempts to write will cause the system to give a process its own copy of a page. Can’t cause an access violation
Handling a page fault

1) Check if the reference is valid
   - if not, terminate the process
2) Find a page frame to allocate for the new process
   - for now we assume there is a free page frame.
3) Schedule a read operation to load the page from disk
   - we can run other processes while waiting for this to complete
4) Modify the page table entry to the page
5) Restart the faulting instruction
   - hardware normally will abort the instruction so we just return from the trap to the correct location.