Segmentation and Paging

CSC 213 – September 10, 2021
Agenda for Today

1. Q&A
2. Segmentation and Paging
3. Hexadecimal Practice
4. Wrap Up
Assignment Questions

It appears that main.c never calls sorted_list_destroy. Should we add a call?

Yes, that's a good idea. This is just for testing. You don't have to turn in the new version of main.c.

Do we have to use the stack demo code from class for our sorted list?

No! You can implement the sorted list any way you like. What I don't recommend doing is implementing a sorted list on top of the stack (e.g. don't call push, pop, and peek inside your sorted list functions).

Does efficiency matter?

No, but it's a good idea to make things efficient when you can. For example, try not to walk through the whole list when counting values.
Assignment Questions

Do we have access to the ngram solution?

   Not right at this moment, but I'll make it available and post on teams.
Lab Questions

How should mysh respond to an invalid command?
   The shell should print an error (maybe with perror) and then display another prompt. It's okay if your shell prints exit status for a child that didn't run.

Why doesn't cd work?
   The cd program doesn't really do anything useful. You'll implement a special cd command in part C.

What if mysh prints exit status of 174 instead of 0?
   You are using the exit status incorrectly. Use the WEXITSTATUS macro!
Lab Questions

In part E, how do we even create background processes?

A background process is the default state of a process. Running something in the "foreground" requires extra work: the parent waits for the child. At some point, you'll need to check if it's exited, but not block. You can do that with the `waitpid` function using the `WNOHANG` flag.

Why and how do we call `waitpid` with `WNOHANG` in a loop?

You need to call `waitpid` repeatedly to collect all of the exited background commands. Call it over and over again until it tells you there are no exited children left.
Lab Questions
Segmentation and Paging
Differences

The reading for today describes two mechanisms to implement address spaces.

What are the differences between segmentation and paging?

Segments can be variable length, but pages are all the same size.

Paging may waste space: we get a whole page even if we need one byte (*internal fragmentation*)

We have just a few segments (code, stack, data) but an address space may need many pages.

Segmentation may waste space: we can have little gaps between segments (*external fragmentation*)

Segmentation generally requires large contiguous regions of memory.

Address translation is more complicated with paging.

Paging makes it easier to share code and data in multiple places.
Whiteboard: Segmentation
Segmentation

<table>
<thead>
<tr>
<th>id</th>
<th>base</th>
<th>size</th>
<th>perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5MB</td>
<td>3MB</td>
<td>r-x</td>
</tr>
<tr>
<td>1</td>
<td>2MB</td>
<td>2MB</td>
<td>rw-</td>
</tr>
<tr>
<td>2</td>
<td>0MB</td>
<td>3MB</td>
<td>rw-</td>
</tr>
</tbody>
</table>
Whiteboard: Paging
Paging

32 bit pointers => 4GB address space

4KB page size

Pages begin at multiples of 4KB
Hexadecimal Practice
Hexadecimal

<table>
<thead>
<tr>
<th>hex</th>
<th>dec.</th>
<th>bin.</th>
<th>hex</th>
<th>dec</th>
<th>bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0</td>
<td>0000</td>
<td>0x8</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>0x1</td>
<td>1</td>
<td>0001</td>
<td>0x9</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>0x2</td>
<td>2</td>
<td>0010</td>
<td>0xA</td>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>0x3</td>
<td>3</td>
<td>0011</td>
<td>0xB</td>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>0x4</td>
<td>4</td>
<td>0100</td>
<td>0xC</td>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>0x5</td>
<td>5</td>
<td>0101</td>
<td>0xD</td>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>0x6</td>
<td>6</td>
<td>0110</td>
<td>0xE</td>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>0x7</td>
<td>7</td>
<td>0111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Write a four-digit hex number with bits 4 and 12 set, and the rest unset:

\[ 0x1010 = 0b 0001\_0000\_0001\_0000 \]

2. Which of these numbers are multiples of 32 (0x20)?

- 0x10
- 0x24
- 0x20
- 0xF0

   - \( \checkmark \)
   - \( x \)
Reminders

Reading
Read about memory errors for Monday.

Assignment
The sorted-list assignment is due next Monday.

Lab
The shell lab is due next Wednesday (9/15) for full credit.
Turn the lab in by the following Wednesday (9/22) for up to 80% credit.

Syllabus Reminder
You need at least 70% on every lab (before any late penalty) to pass this class.