Section 6

Eric Wu (ericwu@cs)
Topics for Today

- Project 1 & 2 Recap
- More Project 3
- Virtual Memory
- Deadlocks
- Midterm
Project 1 & 2 Recap

• Quick recap of design considerations
  – What went well?
  – What should be improved?
Project 1 Designs

OK
NtReadFile(...) {
    ...  
    recordValue(retVal1);
    return retVal1;
    ...  
    record Value(retVal2);
    return retVal2;
}

Better
NtReadFile(...) {
    int retVal = _NtReadFile(...);
    recordValue(retVal);
    return retVal;
}

_ NtReadFile(...) {
    // Old NtReadFile
    ...
}
Project 1 Designs

OK
sysinfo.c

ULONG readInfo = 0;
ULONG readWarning = 0;
ULONG readSuccess = 0;
...

Better
sysinfo.c

struct CSE451Info {
    ULONG read[4];
    ULONG write[4];
    ULONG open[4];
    ULONG create[4];
};
Project 2 Designs

OK
• Using buffer as a contiguous block and resizing when full.
• Storing entire output string of the history entry into the buffer.

Better
• Using linked list of buffers and removing from front, adding to back.
  – Also used circularly linked list
• Storing enumerations of each history item into the buffer.
Project 2 Designs

• Overall good design decisions
  – Attaching mutex pointers to CSE451Info structs
  – Making critical sections as small as possible
  – Placing header files in `base/ntos/inc/` and modular implementations in `base/ntos/ex/`
  – Placing initialization code in files in `base/ntos/init/`
More Project 3

• Due Friday, Feb 17 at 11:59 pm
• If your shared space or SVN has issues, let me know ASAP!
• Please describe your changes in your write up!
  – You can use more than 1 page...if necessary.
More Project 3

• Check your group membership with the following on attu:

  group <username>

  – This is your directory in
    /projects/instr/12wi/cse451/<dir>

  – E.g. membership to group cse451x maps to
    /projects/instr/12wi/cse451/x
More Project 3

- Copy by multiple chunks, not necessarily multiple files.
  - Break files into chunks of work (use chunkSize == BufferSize)
  - Schedule chunks to threads (each thread copies one chunk at a time)
More Project 3

- Performance?
  - What to do when there is only one small file?
  - What to do when there are multiple large files?
• Task scheduling approaches?
  – Place chunks in a FIFO queue
  – Work stealing (Google it!)
More Project 3 (Hints)

• Asynchronous I/O needs to keep track of status of operations.
  – E.g. open file, read file, write file, close file
  – State tables may be helpful

• Remember that threads run single functions.
  – Threads terminate after function returns, so figure out how to keep threads alive (if necessary)

• Think carefully about what needs to be locked.
  – Reading and writing a file requires disk seeks.
More Project 3 (Testing)

- Single small file
- Multiple small files
- Single large file
- Multiple large files
- Files in different directories
- Be creative!
Virtual Memory

• Recap
  – Abstracts physical memory
  – Uses a page table and offset to find a real address.
  – Addresses seen in code are actually virtual memory addresses.
Virtual Memory

Here is a pointer:

\[ p: 0x0041ab8fe023ecd5 \]

Process address space:

0 \quad \quad 2^{64}-1

Page table:

<table>
<thead>
<tr>
<th>Virtual Address</th>
<th>Physical Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0041ab…</td>
<td></td>
</tr>
</tbody>
</table>

Physical memory
Virtual Memory

$P_1$ address space

<table>
<thead>
<tr>
<th>code</th>
<th>DLL</th>
<th>data</th>
<th>stack</th>
</tr>
</thead>
</table>

$P_2$ address space

<table>
<thead>
<tr>
<th>code</th>
<th>DLL</th>
<th>data</th>
<th>stack</th>
</tr>
</thead>
</table>

physical memory

page table

$2^{64}-1$
Virtual Memory

• Processes are protected from each other via virtual memory
• But, how is the kernel memory protected?
Virtual Memory

• Processes are protected from each other via virtual memory
• But, how is the kernel memory protected?
  – Kernel memory is part of the process memory!
Deadlocks

• What is it?
Deadlocks

• What is it?
  – An irreducible circular dependence.
Spot the deadlock!

```c
foo(x, y) {
    lock(&x);
    lock(&y);
    ...
    unlock(&y);
    unlock(&x);
    ...
}
```
Spot the deadlock!

Thread 1

```c
foo(B, A) {
    lock( &B );
    lock( &A );
    ...
    unlock( &A );
    unlock( &B );
    ...
}
```

Thread 2

```c
foo(A, B) {
    lock( &A );
    lock( &B );
    ...
    unlock( &B );
    unlock( &A );
    ...
}
```
Spot the deadlock!

Thread 1

```c
foo(B, A) {
  lock( &B );
  lock( &A );
  ...
  unlock( &A );
  unlock( &B );
  ...
}
```

Thread 2

```c
foo(A, B) {
  lock( &A );
  lock( &B );
  ...
  unlock( &B );
  unlock( &A );
  ...
}
```
Midterm

• Anything up to Wednesday’s lecture (Feb. 8th)

• Conceptual questions
  – Not much computation or math required

• Questions?