

Welcome
CS439H!

| | | |
|------|------|------|
| back | back | back |
| back | back | back |
| back | back | back |

to



Stress

- 439H is **not an easy class**
 - Lots of new material
 - Unfamiliar programming environments
 - Fast, often relentless pace
- Struggling in this course is normal
 - There will be times you won't know the answer or solution
 - This is expected - we want everyone to succeed, but the only way we can help is if you ask for it
- If you find yourself overwhelmed or spending more time on this class than you think you should be, **please reach out** to Dr. Gheith or the TAs
 - We can help out as far as the class goes
 - We can provide other resources if we are not able to help

[Mental health resources available at UT](#)

poll

Anonymous 8/22/19
80 👁

What does OS stand for?

Wrong answers only

📁 other

An [instructor](#) thinks this is a good question

Good question | 2 Edit

poll

Anonymous 8/22/19
80 👁

What does OS stand for?

Wrong answers only

📁 other

An [instructor](#) thinks this is a good question

Good question | 2 Edit

S Students' Answer

Anonymous 8/22/19

Omae wa mou Shindeiru

Thanks | 11 Edit

poll

Anonymous 8/22/19
80 👁

What does OS stand for?

Wrong answers only

📁 other

An [instructor](#) thinks this is a good question

Good question | 2 Edit

S Students' Answer

Anonymous 8/22/19

Omae wa mou Shindeiru

Thanks | 11 Edit

i Instructors' Answer

Ahmed Gheith 8/25/19

Nani

Undo thanks | 14

everybody Quiz say YAY!

```
while (true) yield();  
// someone else will  
// take care of it
```

How was the quiz?

- A. easy
 - B. mostly fine
 - C. mostly fine, but not enough time
 - D. too hard, but finished mostly in time
 - E. too hard and not enough time
 - F. too hard regardless of time
-

```
int fd =  
open( "feedback.txt" );  
  
mmap( NULL,  
      64*FEEDBACK_SIZE,  
      PROT_READ,  
      MAP_PRIVATE,  
      fd, 0 );
```

How is p8 going?

- A. that's a thing?
 - B. I've heard/talked about it
 - C. Cloned the project. **(How?)**
 - D. Looked through the starter code. **(huh.)**
 - E. Started planning/writing code
 - F. Done with at least one part of the project
 - G. Done with the whole project but still failing a couple test cases
 - H. Fully syscalling
-

P8

(So, uh, we don't actually have the project yet...)

General Stuff

- Will be released tomorrow morning
- Dr. Gheith will decide what the project is when he wakes up tomorrow

2-week project (-ish)

- Both p8 and p9's specs will be released tomorrow
- p8 test cases due Wed, p8 code due Fri
- p9 test cases due next Wed, p9 code due next Fri
- p8 tests can only test the p8 parts of the spec (but should be valid TCs for p9)
- p9 test cases can test everything

Probable Parts of P8/P9

- mmap
 - User control over virtual memory
- files
 - open()-ing, read()-ing, and close()-ing files
 - maybe also len(), seek()
- user signal handling
 - registering a signal handler

mmap

- Allows us to have memory-mapped files (or zeroed regions)
- Sections off a region of virtual memory to point to some file
- File is lazily loaded (e.g. memory access in mapped area triggers a read)
 - We don't want to be reading entire large files
- Also allows for shared mappings
 - Forked children can access the same file from the same region of memory
- Anonymous mappings
 - Maps all zeros, useful when used in conjunction with other flags
 - Shared memory?

File syscalls (probably)

- `int open(const char* path)`
 - Returns a “file descriptor”, which is just some integer representing a file the kernel has open
 - Similar to what `sem()` did
- `ssize_t len(int fd)`
 - Returns the length of a file in bytes, given the file’s descriptor
- `ssize_t read(int fd, void* buf, size_t n)`
 - Reads up to `n` bytes into `buf` from the file represented by `fd`, starting from the offset
 - Also updates the offset after the read, based off how much of the file was read
 - Returns how much of the file was read
- `off_t seek(int fd, off_t offset, int whence)`
 - Sets the current file location to `offset`
- `int close(int fd)`
 - Closes the file associated with `fd` (also clean up any internal structures you may have created)

Signals

- Like interrupts, but handled by the user
- `kill` command in Linux sends a signal to a program
 - Don't be confused by the name - SIGKILL (signal #9) is only one of the many signals you can send
- Some signal handlers exist by default, but you can also register your own
 - Useful for if a program wants to catch various signals, like SIGINT (ctrl+c)
- Signal handler runs when receiving a signal, then returns back to previous execution
- How to implement?

