Welcome baaaaack to CS439H!

Wake me up when September ends

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

Quiz everybody say WAWAWAWAWAWA

```
fs->read_all(
   “feedback.txt”,
   n,
   buffer
);
```

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

P5

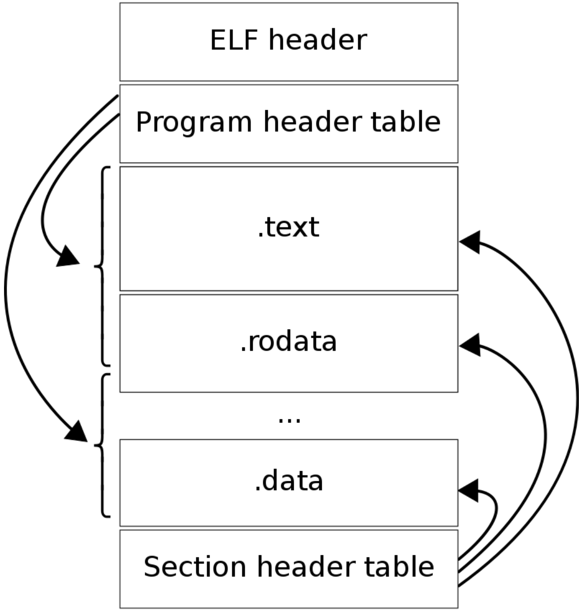
```
fs->read_all(
    “feedback.txt”,
    n,
    buffer
);
```

How is p5 going?

A. that’s a thing?
B. Cloned the project.
C. Looked through the starter code.
D. Started planning/writing code
E. Done with at least one part of the project
F. Done with the whole project but still failing a couple test cases
G. Any% p5 Speedrun glitched

ELF?

- **E**xecutable and **L**inkable **F**ormat
- Binary file that represents something you can run (i.e. a program)
  - Analogous to exe on Windows or dmg on Mac
- Two main parts: header and program



ELF header

Program header table

.text

.rodata

...

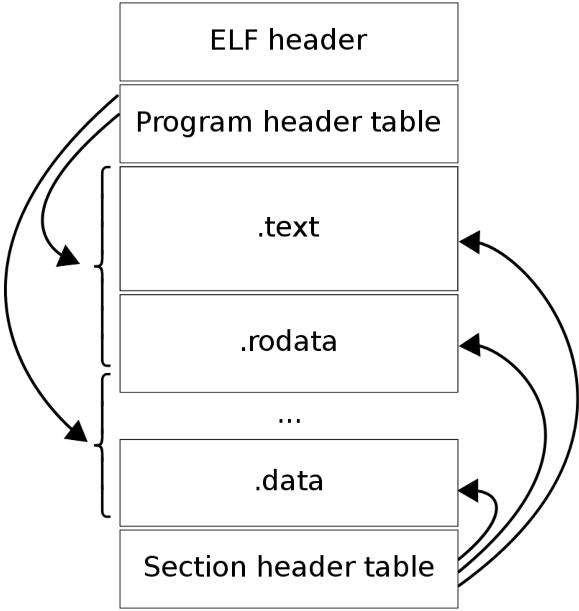
.data

Section header table

ELF?

- How to run a program?
  a. Read the program from the filesystem
  b. Load the program into memory (where?)
  c. Jump to the entry point of the program (how?)

https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

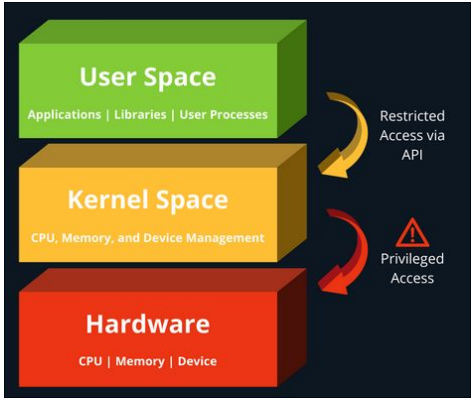


More ELF!

- Relocatable ELFs
  a. Sometimes we want our ELFs to be loadable at different addresses but still work the same
     - e.g. position independent executables, shared libraries
  b. Less vulnerable to attacks/breaches (why?)
  c. Dynamic Libraries!
     - Code sharing (less memory usage!), easy to update
     - Addresses are determined at runtime, so we can load multiple in different spots (think libc)

“User Mode”

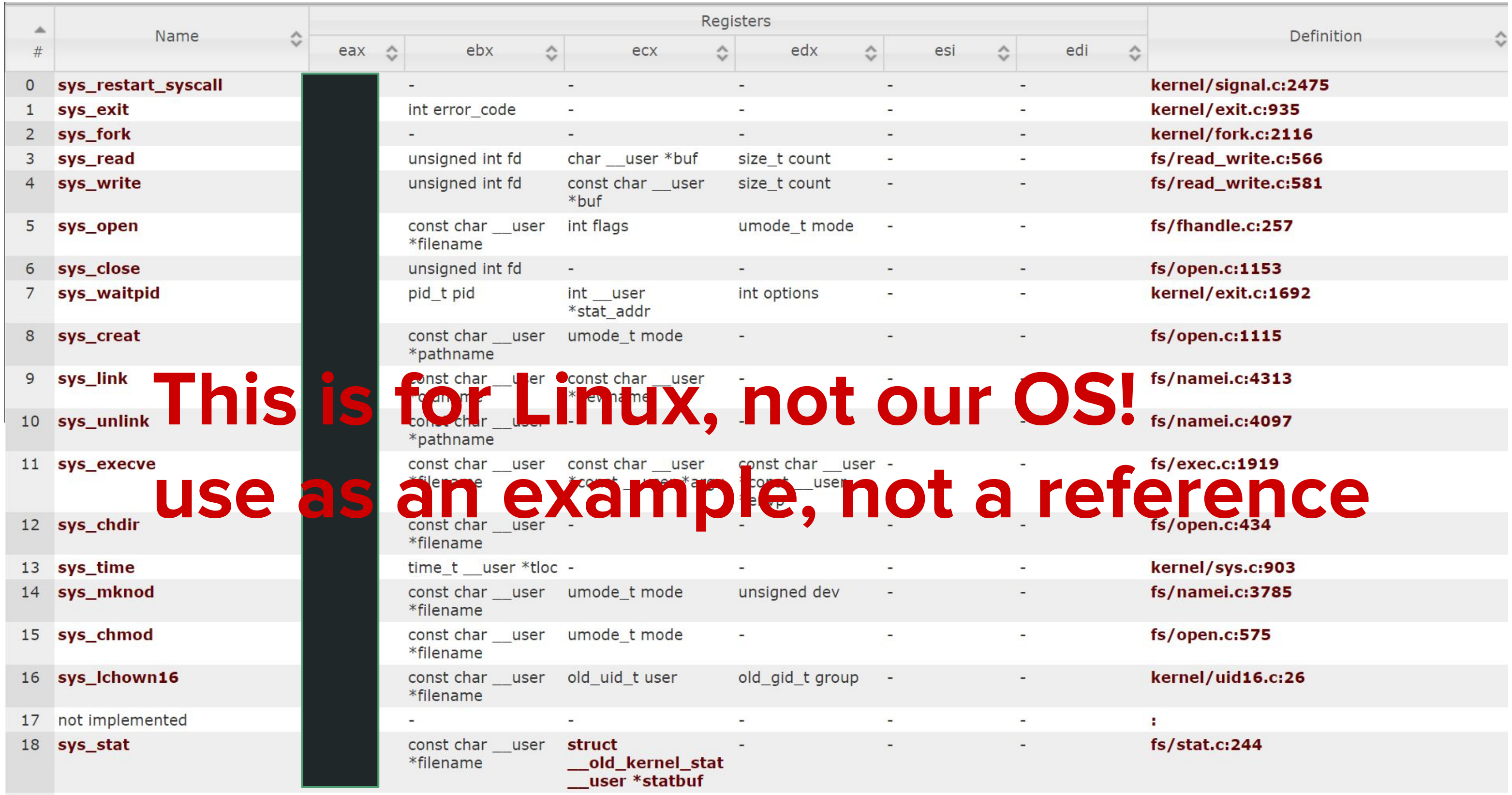
- What’s the difference between a kernel and a user?
- Why do we need user mode?
- User programs - could be anything…
  - Imagine a malicious usermode test case, your kernel should be able to defend against it
  - So far, tests have been in kernel mode, but now you write kernelMain yourself
- We put protections around potentially dangerous things like messing with the filesystem
- But user programs still need to do things like read files?


User Space
Applications | Libraries | User Processes
Restricted Access via API
Kernel Space
CPU, Memory, and Device Management
Privileged Access
Hardware
CPU | Memory | Device

Syscalls

- Used to ask the kernel for restricted tasks
- Eax → eax register in x86
- Corresponds to the type of sys call
- Parameter 1 is stored in userEsp[1], parameter 2 is userEsp[2], etc etc

Syscalls (register packing)



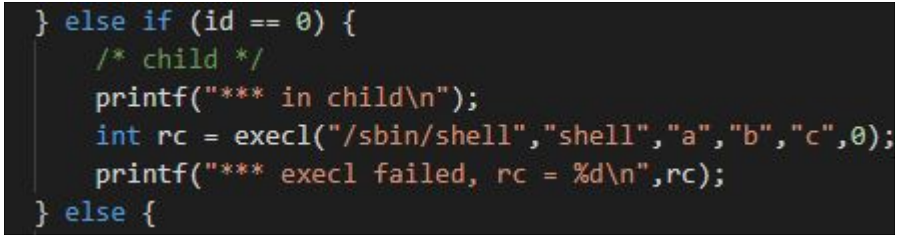
#	Name	Registers						Definition
		eax	ebx	ecx	edx	esi	edi	
0	**sys_restart_syscall**	[redacted]	-	-	-	-	-	**kernel/signal.c:2475**
1	**sys_exit**	[redacted]	int error_code	-	-	-	-	**kernel/exit.c:935**
2	**sys_fork**	[redacted]	-	-	-	-	-	**kernel/fork.c:2116**
3	**sys_read**	[redacted]	unsigned int fd	char __user *buf	size_t count	-	-	**fs/read_write.c:566**
4	**sys_write**	[redacted]	unsigned int fd	const char __user *buf	size_t count	-	-	**fs/read_write.c:581**
5	**sys_open**	[redacted]	const char __user *filename	int flags	umode_t mode	-	-	**fs/fhandle.c:257**
6	**sys_close**	[redacted]	unsigned int fd	-	-	-	-	**fs/open.c:1153**
7	**sys_waitpid**	[redacted]	pid_t pid	int __user *stat_addr	int options	-	-	**kernel/exit.c:1692**
8	**sys_creat**	[redacted]	const char __user *pathname	umode_t mode	-	-	-	**fs/open.c:1115**
9	**sys_link**	[redacted]	const char __user *oldname	const char __user *newname	-	-	-	**fs/namei.c:4313**
10	**sys_unlink**	[redacted]	const char __user *pathname	-	-	-	-	**fs/namei.c:4097**
11	**sys_execve**	[redacted]	const char __user *filename	const char __user *const __user *argv	const char __user *const __user *envp	-	-	**fs/exec.c:1919**
12	**sys_chdir**	[redacted]	const char __user *filename	-	-	-	-	**fs/open.c:434**
13	**sys_time**	[redacted]	time_t __user *tloc	-	-	-	-	**kernel/sys.c:903**
14	**sys_mknod**	[redacted]	const char __user *filename	umode_t mode	unsigned dev	-	-	**fs/namei.c:3785**
15	**sys_chmod**	[redacted]	const char __user *filename	umode_t mode	-	-	-	**fs/open.c:575**
16	**sys_lchown16**	[redacted]	const char __user *filename	old_uid_t user	old_gid_t group	-	-	**kernel/uid16.c:26**
17	not implemented	[redacted]	-	-	-	-	-	**:**
18	**sys_stat**	[redacted]	const char __user *filename	**struct __old_kernel_stat __user *statbuf**	-	-	-	**fs/stat.c:244**

This is for Linux, not our OS!

use as an example, not a reference

exec(l)

- `int execl(const char* pathname, const char* arg,…, (char*)NULL)`
- Switches to different executable
- Never returns



```
} else if (id == 0) {
    /* child */
    printf("*** in child\n");
    int rc = execl("/sbin/shell","shell","a","b","c",0);
    printf("*** execl failed, rc = %d\n",rc);
} else {
```

Path will tell us where the elf file to load is

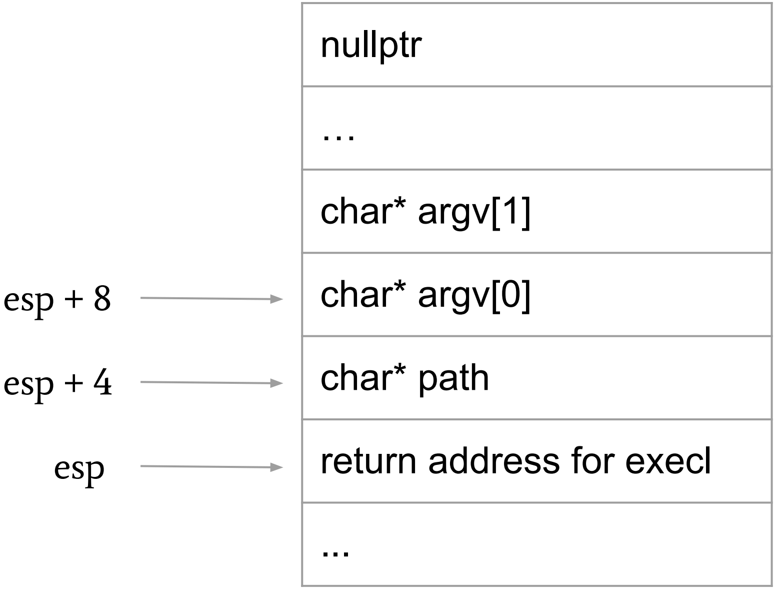
Argc should be 4

Argv should be a pointer to an array containing the other arguments

With this example we are passing in 6 arguments:

- path - /sbin/shell
- arg1 - shell
- arg2 - a
- arg3 - b
- arg4 - c
- arg5 - null

exec(l) - The layout of arguments when CALLED



nullptr

…

char* argv[1]

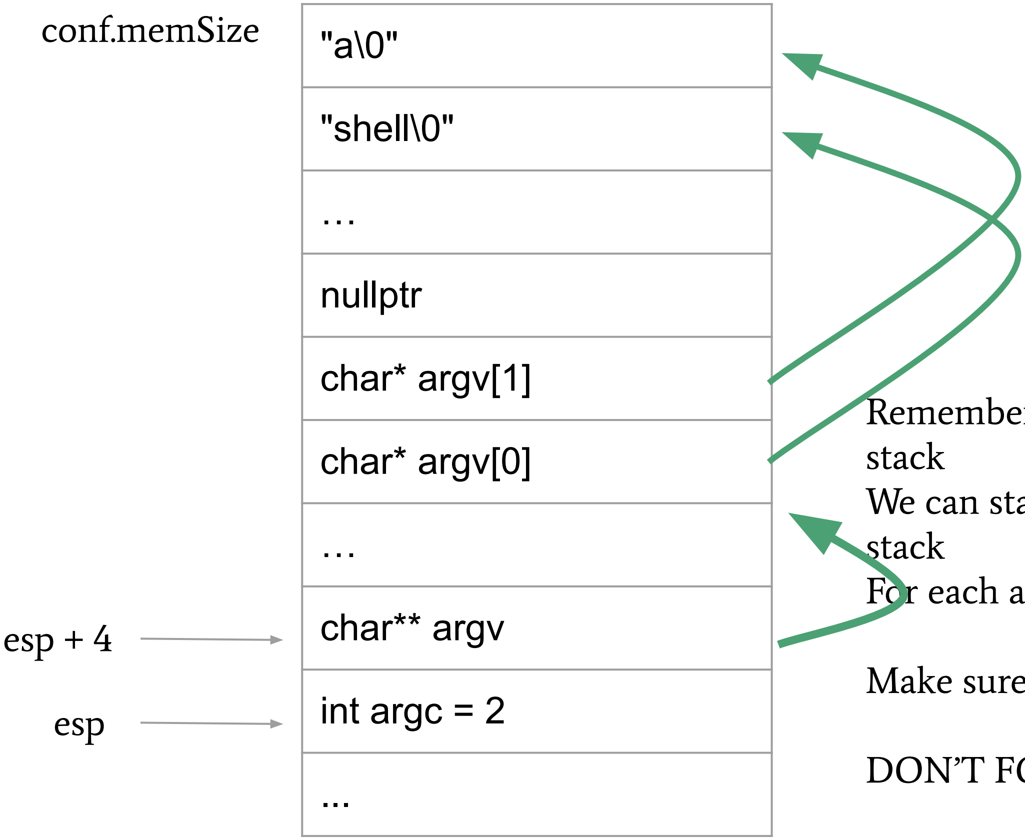
esp + 8 → char* argv[0]

esp + 4 → char* path

esp → return address for execl

...

exec(l) - How to set up parameters on the NEW stack



conf.memSize

"a\0"

"shell\0"

…

nullptr

char* argv[1]

char* argv[0]

…

esp + 4 → char** argv

esp → int argc = 2

...

```
execl("/sbin/shell", "shell", "a", nullptr);
```

Remember - The "top" of the stack is the lowest address in the stack
We can start with userEsp=kConfig.memsize as the top of the stack
For each argument we push on, we subtract from the esp.

Make sure to keep arguments 4 byte aligned by the end.

DON'T FORGET NULL TERMINATORS!!!

exec(l) - cont

- After setting up the Kernel Stack to mimic the new process we need to:
- Load in the desired file and entry point

- To fully transform into a new process we call switch to user.
- What should entry be?
  - Comes from loading the elf file
- What should be the user stack?
  - The top of the stack that we altered from the previous slide

exit

- `void exit(int rc);`
  - Ends the process with exit code rc.
  - Also, in this project, shuts down the whole system (obviously not true for real systems)

write

- `ssize_t write(int fd, void* buf, size_t nbyte);`
    - Attempts to write `nbyte` bytes of data starting at the pointer `buf` into the file descriptor `fd`
    - If successful, returns the number of bytes written
        - **Not guaranteed to be equal to `nbyte`**
        - Must be at least 1 if successful (guarantee some progress)
    - We only support writing to standard output
        - stdout is represented by fd=1 by default

Error behavio(u)r

- User programs can be invalid
  - what should we do?
  - how should we guard against malicious user programs?
  - what if user programs try to access kernel?

p5 structure

- `kernel.cc`
  - The starting code that runs right before you launch your very first user process
  - `kernelMain` should execute the ELF file /sbin/init and not return
    - Called right after `init.cc` finishes setting up
- `sys.h/sys.cc`
  - Kernel handlers for system calls
- `elf.h/elf.cc`
  - ELF loader for an ELF file given a Node
  - You should reject invalid ELFs or non-ELF files
- None of this is publicly visible to the test case - you can feel free to mix things up as you please
  - You can add/remove functions that you want (except `kernelMain`, which has to be the main entry point of your kernel)
  - But if you do really crazy things that we can’t understand, we can’t help

p5 structure

- Test cases look different, *again*
    - (This is a report question, so it is your job to figure out how the test system works) :D

Questions?

```
                                *** Don't panic
                                      ***
                                      ***
               ***                          oooo$$$$$$$$$$$$oooo
             ***                       oo$$$$$$$$$$$$$$$$$$$$$$$$o
  ***                    oo$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$o         o$   $$ o$
 ***    o $ oo        o$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$o       $$ $$ $$o$
  *** oo $ $ "$      o$$$$$$$$$    $$$$$$$$$$$$$    $$$$$$$$$o       $$$o$$o$
  *** "$$$$$$o$     o$$$$$$$$$      $$$$$$$$$$$      $$$$$$$$$$o    $$$$$$$$
  ***   $$$$$$$    $$$$$$$$$$$      $$$$$$$$$$$      $$$$$$$$$$$$$$$$$$$$$$$
   ***   $$$$$$$$$$$$$$$$$$$$$$$    $$$$$$$$$$$$$    $$$$$$$$$$$$$$   """$$$
   ***    "$$$""""$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$     "$$$
  ***     $$$   o$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$     "$$$o
 ***    o$$"   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$        $$$o
***    $$$    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$"  "$$$$$$ooooo$$$$o
***   o$$$oooo$$$$$  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$   o$$$$$$$$$$$$$$$$$
   ***   $$$$$$$$"$$$$   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$     $$$$""""""""
       ***  """"       $$$$    "$$$$$$$$$$$$$$$$$$$$$$$$$$$$"       o$$$
        ***             "$$$o     """$$$$$$$$$$$$$$$$$$$"$$"         $$$
        ***                $$$o          "$$""$$$$$$""""            o$$$
         ***                $$$$o                                 o$$$"
          ***                 "$$$$o       o$$$$$$o"$$$$o        o$$$$
          ***                    "$$$$$oo     ""$$$$o$$$$$o   o$$$$""
            ***                       ""$$$$$oooo  "$$$o$$$$$$$$$"""
              ***                          ""$$$$$$$oo $$$$$$$$$$
              ***                                  """"$$$$$$$$$$$
             ***                                      $$$$$$$$$$$$
             ***                                       $$$$$$$$$$"
               ***                                        "$$$""
                                      ***
```