CS241 Final Exam
Fall 2021

Welcome to your CS241 Final Project. This is a "take home" open book exam. To complete CS241 you will be demonstrating *your own competency* in CS241 skills and knowledge. This means you may not collaborate, work with, or get help from current or former CS241 students or search for solutions. This final project is open book (you may access and reuse general system programming online materials) but the code and spoken words you include must be your own original work that you created during this exam.

You may not share code, hints, test code or anything else that you created as part of this exam. Publishing your work that is part of this final exam or sharing your work with another student directly/indirectly is a violation of academic integrity and may affect not just your CS241 grade but your standing as a student at UIUC.

Keep the total time spent on this to less than 5 hours (it should be 3 hours); it is up to you to track this.

We expect most of you will have access to a Linux system (e.g., WSL on Windows, Virtual Machine, ssh access to your CS241 VM, or the VM-in-a-browser). However in principle you could complete this using only a smart phone, pen and paper. Complete this final project by Wednesday December 15th **11:00pm CT**. Submission details will be posted on Ed and/or email. There will be an approximate 59 minute grace period after 11pm to allow for slow uploads and IT issues.

Questions? Other than submission questions, we're not going to respond to questions about the actual content of the exam; use your best efforts to interpret the instructions.

If you believe it is ambiguous state your assumptions and answer accordingly in your exam responses. We may answer questions about logistics on the forum but not about the exam content.

Out of fairness, do not share or post comments about the content until after grades are published.

Be sure to check your submission and allow sufficient time to upload it; we recommend you upload small files before uploading your video file. Your video should be mp4 format or similar variant (e.g. ".mov"). It may be compromised of multiple files.

Part 1 of 3 (100 points, video, 60 minutes) “My First Office Hours”

> *"Um. Don't Panic. Um. Sorry!" a familiar English voice announces, "I'm just adjusting the ... Aha! There we go."*
>
> *The world shifts into full view. Sensations of touch, smell, and sound flood your mind. You are desk area in Siebel and a light breeze against your face awakens you out of a dreamy slumber. There's some tables, a whiteboard, bookcase and chairs. The sensations feel real except the scene is clearly fake the words "History Simulation Test (bug fix 53)" scroll in and blinks slowly in orange and blue letters in the bottom left corner of your vision in Courier New font, confirming that none of this is actually real. Yet the perceptual sensations feel more tangible than the breakfast you had earlier, and definitely more meaningful. A sense of spreading future connections overwhelms you (like a 400-level course on graph theory where the expanding nodes and edges escaped from the paper and leapt into the wild) – connections to people, events, things – future interviews, people, conversations, things that will help you change the world but have not yet happened but will happen in your future.*
>
> *"Okay," the friendly voice continues, "I set the year back to 2022 – which explains the odd fashion choices you're about to see! In about 1 minute your office hours will start and the first student will walk in. This first test is about whether you can explain to a student some 241 topics, to demonstrate that you actually understand these topics and can be a valuable member of the course staff. So be ready to do some live programming demos, explain the concepts, whatever you need to help the student thrive in this class.*
>
> *We're not sure how students learned back in 2022 but the computational-socio-geologists believe they were a social bunch who really enjoyed talking and explaining things to each other; they were a strong community that wanted a meaningful life by changing the world and by their connections with each other. Don’t forget to pay it forward!*

Continued...

Record your CS241 office hour video. Here are the 10 things the students asks about. (You might want cross them off as you complete them)

1. Why does `asprintf` take a pointer to a pointer, can you help me understand why and how to use `asprintf`?
2. I'm confused about threads; should I use `pthread_join` or `pthread_exit`, do they do the same thing?
3. I've heard of '`fork-exec-wait`' can you write some code to explain what is going on each step?
4. I'm stuck; my heap allocator code keeps crashing; can you give me three suggestions on how to find the problem?
5. What is block coalescing and why do Knuth's boundary tags help?
6. Why do I need condition variables; can't I just use mutex locks for all synchronization problems?
7. I read the course book but can you help me understand the conditions for deadlock in terms of code that I might write?
8. Conceptually, what is basic idea of the reader-writer problem?
9. The Critical Section Problem talks about "Mutual Exclusion" "Progress" and "Bounded Wait"; what do these mean?
10. When you took CS241 what was i) your favorite and ii) most challenging parts and iii) why?

We expect you will want to record a video of your laptop screen with audio and use a text editor as a whiteboard. However any method of recording will be acceptable (e.g. phone pointed at piece of paper). We are looking for *demonstrations of system programming understanding and competency* (i.e. things that a CS225 student would not be able to explain but a CS241 student can), such that your office hour is effective and useful. Words need to be your own words,ideas and thoughts; reading aloud a Google search result or just the course book is not acceptable.

You can record multiple parts of a video if you need a break. However don't worry about fluffs, mistakes and restarts; imagine this was a **real office hours** – just say oops and carry on! We expect most students will create approximately 30- 60(max) minutes of content. We will only grade the first 60 minutes of video content; which means you only have 6 minutes per item.

There are multiple methods to record a laptop screen to mp4 file or cloud (e.g. Protip: *Do a test recording first and review it instead of assuming that it is working*; make sure your text is large enough and legible enough to be gradeable and that it is a recording of your screen not your laptop camera!). Ultimately you will be sharing the mp4/mov file or providing a URL to your cloud recording (be sure to check that the link works when not logged in as you). The university website, https://mediaspace.illinois.edu may be another useful way to record and share your video

Grading Rubric Outline: For each of the 10 student questions,

0	missing
5	mostly incomplete or misleading or inaccurate
9	mostly helpful for the student; some minor errors
10	helpful; no errors

Part 2 of 3 (100 points, code, 60 minutes) "Memory mapped secrets"

It's no longer safe and you have only minutes to spare before *BigMichicanCorp* finds you; time for a fast exit. You've put all of your valuable secrets into a single file (it could be a text file, zip file etc), now you just need to get out of here without anyone being able to discover or decrypt its valuable contents. Fortunately you have your CS241 skills and a VM. You quickly create and use a Linux C99 program `helloworld.c` using the following -

```
clang -O0 -Wall -Wextra -Werror -Wno-error=unused-parameter -g -std=c99
-D_GNU_SOURCE -DDEBUG helloworld.c
```

When your program is run it seems to be the simplest program; it just prints `Hello World`

```
./a.out
```

(prints "`Hello World`" without the quotes to standard-out and exits with value 0)

However, later if `a.out` is renamed to `encrypt` it can create an encrypted version of the contents of your file with every byte xor'd with a random value and store it in a new output file

```
mv a.out encrypt
./encrypt mystuff.zip cats.jpg
```

It opens the existing file `mystuff.zip` in binary (using `open` and `mmap`) for reading and writing, and creates one new file `cats.jpg`. As it processes the bytes of `mystuff.zip` it also overwrites as soon as possible the contents of `mystuff` with `0xff` bytes so that the file's original content is no longer recoverable. The first half of `cats.jpg` is a sequence of random bytes (read from `/dev/urandom`), the same length of the input file. The second half contains the bytes from the original file encrypted by xor-ing each byte with the corresponding byte stored at the first half of the file. Upon completion the output file will be the twice length as the original file while the length of the original file is unchanged but all of original contents will have been overwritten.
For example if `mystuff.zip` contained two bytes, `0x51,0x52` and the first two random values read from `/dev/urandom` were `0x41 0x03` then the new `cats.jpg` would contain with `0x41,0x03,0x10,0x51` and `mystuff.txt` would become `0xff,0xff`.

You head out of town, pausing only to donate your old laptop. Three days, two flights and a row boat later, you rename your program to `decrypt`, and recover your secrets -

```
mv encrypt decrypt
./decrypt cats.jpg output.zip
```

This reads the two file `cats.jpg` (using `mmap`) and creates the output file `output.zip` (with contents identical to the original `mystuff.zip` file).

Upon successful completion it deletes `cats.jpg` from the filesystem.

For example, using the file contents above it could recover the original two byte file (0x51 0x52)

Write one program that uses the process name to determine behavior when executed. If the program name is `./encrypt` then encrypt the file, if it is `./decrypt` then decrypt the given files, as described above. If neither is true, or 2 filename arguments are not provided, then innocently print "`Hello World`" and exit with value 0.

Use `argv[0]` to check the process name.
Use `open` and `mmap` to read and modify the contents of original file (encryption mode) and to read the input file in decryption mode.
Use `fopen` and `fputc` to create the output files in both encrypt and decrypt modes.
The `unlink` and `stat` calls may also be useful.
The `hexdump` utility may be useful.

Grading Rubric & Submission

Add your netid as a comment to your code. For example, if your netid was angrave, write

```
// author: angrave
```

Upload your code, `helloworld.c`; it should compile using the above clang options on a standard CS241 VM. It will be graded on (subject to minor modifications)

10	Correctly uses the program’s arguments to change behavior between hello-world, encrypt and decrypt modes based on the process’ name
10	Overwrites the original file contents with `0xff` bytes as soon as each byte has been read.
10	Random bytes are sourced from `/dev/urandom`
10	Does not crash; exits normally with value 0 when given valid arguments
10	Encoding generates desired encrypted output file
10	The original file’s content can be recreated after encrypting and decrypting
10	Encrypted file is removed from the filesystem after decryption processing is complete
10	Uses `mmap` to read the given input file
10	Uses `fopen` and `fputc` for output
10	Includes your netid in the author comment at the top of the code

All other behaviors, output, missing arguments, missing files, handling error conditions etc. are *unspecified and are not graded.*

Part 3 of 3 (100 points, code, 60 minutes) "Saving Demo Day (my partner owes me big time)"

It's 1 hour to demo time but your partner has not shown up. Their custom web server is running on their VM but you don't know the port number that it is listening on.
Create a C99 Linux program `search.c` using the same clang options as part 2 that automatically tries to connect on every port number for all TCP ports in a given range for a given host. Print out the first port that allows a connection and responds like a web server (returns status code 200).

In the example below, your search program attempts an IPv4 scan for open TCP ports between 1000(inclusive) and 2000(exclusive) on the host `mydemo.cs.illinois.edu`

```
./search mydemo.cs.illinois.edu 1000 2000
```

If a connection is found send a simple, valid HTTP request `GET / HTTP/1.0\r\n\r\n` and wait for a response (you can assume the service will respond quickly or terminate the connection). If the response starts with "`200`" then print the port number and exit. Write out a single unbuffered "." for every port number check that failed (though this not graded). For example, the above program might print the following before exiting.

```
............................1028
```

Grading Rubric & Submission

Add your netid as a comment to your code. For example, if your netid was angrave, write

```
// author: angrave
```

Upload your code `search.c`; it should compile using the clang options in Part 2 on a standard CS241 VM. It will be graded on the following functionality -

20	Attempts connections between the start and end(exclusive) port numbers for the given host using IPv4
20	Ignores closed ports
20	Ignores open ports that do not respond like the expected HTTP server
20	Prints the port number and exits when the webserver is found
20	Includes your netid in the author comment at the top of the code

Hints: A python web server will be useful for testing. Don't forget to close fds when `connect` succeeds. Your implementation does not need to be fast. You can assume a valid hostname and network connection. You should not print anything else to stdout; output to stderr will not be graded. You may start with textbook networking example (please cite your source) but the bulk of the code should be your own.

That's it - Thank you! We will provide submission details on or before Wednesday. Expect to upload your mp4 video file(s) (or provide a working link), plus `search.c` `helloworld.c` files. If providing a URL link to your video, check that the link works in a browser when not authenticated as yourself. Humans (TAs) will grade your files; though we may use automation to short-circuit handgrading of working code that meets the grading criteria. Good luck in 2022!