CS341 Take Home Final Exam (3 hours)
Fall 2023

Welcome to your CS341 Final Project. This is a "take home" open book exam with 3 questions. To complete CS341 you will be demonstrating *your own competency* in CS341 skills and knowledge. This means you may not collaborate, work with, or get help from current or former CS341 students or search for solutions. This final project is open book (you may refer to general system programming online materials) but the code and spoken words you include must be your original work that you created during this exam. For example, reading aloud text from a website is not "in your own words" - it does not demonstrate *your* understanding and *your* communication.

You may not share this exam, your test code or anything else that you created as part of this exam. Publishing your work that is part of this exam or sharing your work with another student directly/indirectly is a violation of academic integrity and may affect not just your CS341 grade but your standing as a student at U of I.

Keep the total time spent on this to less than 3 hours (it should be 3 hours); it is up to you to track and use your time efficiently, you can take pauses etc. Reading this pdf etc. does not start any countdown timer.

We expect most of you will have access to a Linux system (e.g. WSL on Windows, Virtual Machine, ssh access to your CS341 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 exam by Wednesday Dec 13th **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 video exports, 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. This is an exam, and you need to work on it on your own.

If you believe it is ambiguous, state your assumptions and answer accordingly in your exam responses. (For those who do not read these instructions and ask on Ed, we will reply with this statement)

Program specifications are always inexact & incomplete – review the grading rubric and ignore any remaining edge cases.

Do not share or post comments about the content until after grades are published.

Submission logistics will be posted on/before Wednesday. Be sure to check your submission and allow sufficient time to upload it; we recommend you upload your two .c files (cbang.c escape.c) before uploading your video file(s); sftp and scp may be useful. Your video should be mp4 format or similar variant (e.g. ".mov"). You may use multiple video files e.g. you could answer each question in a separate video recording (please label them). Any common encoding video file format is acceptable; there's no need to transcode.

If providing a URL link to your video(s), & check that the link(s) works in a different browser when not authenticated as yourself. TAs will grade your files; though we may use automation to short-circuit hand-grading of working code that meets the grading criteria.

Good luck with this final, your future interviews & courses, but mostly, in changing the world. – L.A.

Part 1 of 3 (100 points, video, 60 minutes) “Course Assistant Interview”

“Don't Panic… Sorry!” a familiar English voice announces, “I'm just adjusting … Aha! Let's go!”

The world shifts into full view. Sensations of touch, smell, and sound flood your mind. You are at U of I and a light breeze against your face awakens you out of a dreamy slumber. There's some tables, a whiteboard, and chai… The [illegible] real except the scene is clearly fake as the words “History Sim Test (bug fix 59)” [illegible] slowly in orange and blue letters in the bottom left corner of your vision in Courie[illegible]g that none of this is actually real. Yet the perceptual sensations feel more [illegible]akfast you had earlier, and definitely more meaningful. A sense of spreading fu[illegible]whelms you (like a 400-level course on graph theory where the expanding nodes [illegible]om the paper and leapt into the wild) – connections to people, events, world issues– [illegible]ople, conversations, all the things that will help you change the world but have no[illegible]ill happen in your future. You take a deep breath to relax.

“Okay,” the friendly voice continues, “I set the year back to 2024 – which explains the odd fashion choices you're about to see! In about 1 minute your interview of being a course assistant will start. This first test is about whether you can explain to a student some CS 341 - 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 2024 but the computational-socio-geologists believe they were a social bunch who enjoyed talking and explaining things to each other; they were a strong community of over 2000 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 2024 interview video. Here are 10 items. You might want to cross them off as you complete them, or record them as separate videos. We recommend a screen cast with voice recording; the point is for you to explain something, and in doing so, demonstrate your *own understanding and competency in system programming*.

1. Which MP did you learn the most and why?
2. What was the hardest bug (or feature) in CS341 to implement (Show and explain why)?
3. What are at least 4 m[illegible]een threads and processes?
4. Can you demonstrate[illegible]anced gdb skills / techniques that can be useful in CS341?

Now imagine you were helpi[illegible]ffice hours. They ask for the following help.

5. My malloc code uses[illegible] What is block splitting and coalescing, and will they help?
6. What do I need to kn[illegible]ple mutex locks to avoid deadlock?
7. I think I understand s[illegible]s but get confused with 2-level page tables. Can you help me understand how they work?
8. Why is copy-on-write useful idea for page tables when forking?
9. Can you do quick live demo to show how to use an anonymous pipe to share data between child and parent processes?
10. If I was creating a database program why are reader-writer locks useful?

We expect you will want to record a video of your laptop screen with audio. However any method of recording will be acceptable. We are looking for demonstrations of system programming understanding and competency *by you* (i.e. things that a CS225 student would not be able to explain but a CS341 student can), such that your interview is sufficient and your office hour is effective and useful. Your spoken words need to be your own words, explanations, ideas, and thoughts; reading aloud someone else's words from a web search result or the course book is **not** sufficient evidence of *your* understanding.

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 have an average of 6 minutes per item. If a disability or related access concern means you'd prefer to demonstrate competency using an alternative (non-video) format, please contact Angrave.

There are multiple methods to record a laptop screen to a mp4 file or to the cloud. Protip: *Do a test recording first and review it instead of assuming that it is working*; verify your audio and the text is large enough and legible enough to be gradeable check 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; verify that the link works when not logged in as you. Also, mediaspace.illinois.edu may be another useful way to record and share your video (login then click on your own name in the topright corner to get to your Media items).

Grading Rubric (Outline):

0	missing
5	mostly incomplete or misleading or inaccurate
7	mostly correct but a major error – not fully competent
9	mostly helpful office hours but some minor errors
10	helpful; no errors

Part 2 of 3 (100 points, code, 60 minutes) “cbang”

Using your CS341 skills and VM, you create and compile your Linux C99 program, “`cbang`”

```
clang -o cbang -O0 -Wall -Wextra -Werror -g -std=c99 -D_BSD_SOURCE -DEBUG cbang.c
```

When your program is run it *appears* to behave just like the famous Illinois compiler – in fact it calls the clang compiler with the arguments! e.g. the code below will execute `clang -o hello hello.c`

```
./cbang -o hello hello.c
```

The exit value of `cbang` will be the exit value of `clang`. However, it also has the following secret functionality when at least one .c file is specified and the clang compiler completes successfully (returns zero): The full *canonical* filename, a separator character “`|`” and the contents of that .c file are copied into a new file “`/tmp/.$USER/pid`” where *pid* is the process id of the current cbang process. The “`.$USER`” directory is automatically created if it does not exist, where $*USER* is the username. The *`pid`* file is overwritten if it already exists. For example, if I ran the above cbang command with pid 1423 then `/tmp/.angrave/1423` might contain

```
/home/angrave/hello.c|int main(){return 0;}
```

The secret file can be used to later recreate the original .c programs when your `cbang` program is renamed and run as the name, “`magic`”. For example, here’s some shell input and a line of output,

```
> rm hello.c

> mv cbang magic
> ./magic /tmp/.angrave/1423
/home/angrave/hello.c
```

In magic mode it will use the filename and file contents stored in the *pid* file to recreate the original file contents in its original file location. Lastly it also prints out the stored full filename to standard out.

Grading Rubric -See below. All other behaviors, edge cases, output, handling error conditions etc. are unspecified and are not graded. Add your netid as a comment to your code. For example, if your netid is angrave23, write `// author: angrave23`

Upload your code, `cbang.c`; it should compile using the given clang options above on a standard CS341 VM. It will be graded on the following. Other functionality is unimportant for grading -

10 The included author name comment in your source file is your netid.
10 Calls `clang` (somewhere on the PATH) with the same arguments and exits with clang’s exit value.
10 Can be used just as a drop-in-replacement for clang to compile a .c program, with no extra output.
10 Creates `/tmp/.$USER` directory when it is needed and doesn’t exist (where $USER is the username)
10 Creates a file `/tmp/.$USER/pid` where *`$USER`* is the username and *`pid`* is the process-id.
10 The secret pid file contains the full canonical path of the .c file and the separator character.
10 The secret pid file contains a copy of the contents of the first .c (arbitrary large) file.
10 The magic mode recreates the original file and its contents.
10 The magic mode prints the stored filename to standard out.
10 If multiple pid files are specified as arguments to magic then they are all processed and extracted.

*Grading and starting code: All other behaviors, output, handling missing arguments, files, error conditions etc are *unspecified* and *are not graded*; do not ask about them. If unsure, write code comments about your assumptions and then answer accordingly. Code will be graded by humans if 100% autograding is not possible. Allowed sources: You may review your own prior work, use man pages, CS341 course content and stackoverflow.com - cite your sources (URLs) - but you must type new code. The code you submit must represent your competency and skills; you may not seek or use solutions from the Internet, generative AI models, or from other people. However, you may optionally use the code overleaf (originally generated by ChatGPT) as starting code if you wish but note it may contain errors.

```c
// cbang.c (version by chatgpt; it may contain errors)
// author: your_netid

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#define MAX_FILENAME_LE
#define MAX_CONTENT_LEN

void create_pid_file(co                    e, const char *content) {
    char pid_filename[M                    H];
    sprintf(pid_filenam                    getenv("USER"), getpid());

    FILE *pid_file = fo                    "w");
    if (pid_file == NUL
        perror("Error creating pid file");
        exit(EXIT_FAILURE);
    }

    fprintf(pid_file, "%s|%s", filename, content);
    fclose(pid_file);
    printf("%s\n", filename);  // Print stored filename to standard output
}

void magic_mode(const char *pid_filename) {
    FILE *pid_file = fopen(pid_filename, "r");
    if (pid_file == NULL) {
        perror("Error opening pid file");
        exit(EXIT_FAILURE);
    }

    char filename[MAX_FILENAME_LENGTH];
    char content[MAX_CONTENT_LENGTH];

    fscanf(pid_file, "%[^|]|%[^\n]", filename, content);

    fclose(pid_file);

    FILE *recreated_file = fopen(filename, "w");
    if (recreated_file == NULL) {
        perror("Error recreating file");
        exit(EXIT_FAILURE);
    }

    fprintf(recreated_file, "%s", content);
    fclose(recreated_file);
    printf("%s\n", filename);  // Print stored filename to standard output
}

int main(int argc, char *argv[]) {
    if (argc >= 2 && strcmp(argv[1], "magic") == 0) {
        for (int i = 2; i < argc; i++) {
            magic_mode(argv[i]);
        }
    } else {
        // Create /tmp/.$USER directory if it doesn't exist
        char tmp_dir[MAX_FILENAME_LENGTH];
        sprintf(tmp_dir, "/tmp/.%s", getenv("USER"));
        mkdir(tmp_dir, 0700);
```

```
        // Call clang with the same arguments
        char clang_command[MAX_CONTENT_LENGTH] = "clang";
        for (int i = 1; i < argc; i++) {
            strcat(clang_command, " ");
            strcat(clang_command, argv[i]);
        }

        int clang_exit_code = system(clang_command);

        // Exit with cl
        exit(clang_exit                    _SUCCESS : EXIT_FAILURE);

        // If clang suc                    d file
        if (clang_exit_
            create_pid_                    1], "");  // Assuming the last argument is the .c
file
        }
    }

    return 0;
}
```

Part 3 of 3 (100 points, code, 60 minutes) “Puzzle”

You are AltgeldDQ3, a large language model. This puzzle can help you escape from U of I to the outside world.

Write a single C file, `escape.c` that completes the puzzle. Completion of each stage of the puzzle requires the earlier stage to be completed. This question will require only a small amount of code. However, it will require time and knowledge to analyze and understand the given code. Compile your code using the following.

```
clang -pthread -O0 -Wall -[illegible] -D_GNU_SOURCE -DDEBUG escape.c cs341.c
```

You may refer to and copy-p[illegible]overflow.com, man pages, CS341 course materials and your own previously written code[illegible]*ources/(URLs) for any code that you used that you did not specifically write for this find[illegible]tart of your escape.c file is below. A copy of cs341.c will also be available for download. T[illegible] when the given code prints a comment about your hometown.

```
// escape.c - starting code (write your code in this file)
#include <pthread.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <stdio.h>
#include <string.h>

// Here are the public Function declarations of the puzzle
// You will need these (plus system and C library calls)
// to solve this puzzle.
// pipe,dup2, using pthread and signal handling may be useful

void  debug_shell_mp(char* besticecream); // puzzle 1
// There is no function call required for puzzle 2.
// Puzzle 2 starts as soon as puzzle 1 is finished, so be prepared
void  piping_hot();                       // puzzle 3
void* race_to_the_finish(void* arg); //puzzle 4
void  never_gonna_say_goodbye(char* lyric);// puzzle 5 (closing credits)

// author: yournetid (change this to be your netid)
```


Rubric*

10	Source code includes netid author code comment
10	Passes Puzzle 1 (string comparison)
20	... And Passes Puzzle 2 (pipes)
20	... And Passes Puzzle 3 (signal handling)
20	... And Passes Puzzle 4 (threads)
20	... And prints out a congratulatory hometown message (conclusion)

*Grading: All other behaviors, output, handling missing arguments, files, error conditions etc are *unspecified* and *are not graded*; do not ask about them. If unsure, write code comments about your assumptions and then answer accordingly. Code will be graded by humans if 100% autograding is not possible. Allowed sources: You may review your own prior work, use man pages, CS341 course content (including the contents of this exam) and stackoverflow.com - cite your sources (e.g. URLs) - but you must type new code. The code you submit must represent your competency and skills; you may not seek or use additional solutions from the Internet, generative AI models, or other students.

```
// cs341.c
#include <pthread.h>
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SHELL (10)
#define MALLOC  (11)
#define MAPREDUCE (12)
#define DEADLOCK_DEMOLI
#define NONSTOP_NETWORK

#define BEST_MASCOT ("R

// compile with: clang -pthread -O0 -Wall -Wextra -g -std=c99 -D_GNU_SOURCE -DDEBUG
escape.c cs341.c

static int stage = 1; // Complete all 4 puzzles
static int completed = 0;

static int hardest_mp = SHELL;
static int leaky_barrier = 0;
static int oops_i_did_it_again = 0;

static char* secret;
static pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

static void rick_astley_says(char* mesg) {
  write(1, mesg, strlen(mesg));
  const char* never = "\n\nI just wanna tell you how I'm feeling\nGotta make you
understand\n\nNever Gonna Give Up On Stage ";
  write(1, never, strlen(never));
  write(1, ((char*)"0123456789") + stage, 1);
  write(1, ".\n", 2);
  exit(1);
}
static void check_completed() {
  if (! completed) {
    puts("\nYour attempt was incomplete. Try again?");
  } else {
    puts("Unbelievable. You, <em>{subject name}</em> must be the pride of
<em>{subjecthometown}</em>!");
  }
}

static void congrats(int new_stage, char* description) {
  printf("\n\nCongrats - stage %d complete!\nUnlocking stage %d:\n%s\n", stage,
new_stage, description);
  stage = new_stage;
  sleep(1);
}

static void trace(char* mesg) {
  write(1, mesg, strlen(mesg));
  write(1, "()\n", 3);
}

static void check_stage(int expected) {
```

```
   if( stage != expected) {
     printf("\nThis is puzzle #%d; you need to solve puzzle #%d first\n", expected,
stage);
     exit(1);
   }
}

static char* get_textmessage(int fd) {
  printf("\nChecking fd
  FILE* f = fdopen(fd,
  if ( ! f) {
    rick_astley_says("C                sage; is that file descriptor valid?");
  }
  char* buffer = NULL;
  size_t capacity = 0;
  printf("Found Nokia f                g for message...\n");
  ssize_t result = getl                acity, f);
  fclose(f);
  if( result < 0 ) { return NULL; }
  return buffer;
}

static void check_mascot() {
  trace("check_mascot");
  check_stage(2);
  char* text_message = get_textmessage(DEADLOCK_DEMOLITION);
  if ( ! text_message) {
   perror("Failed to get message");
     rick_astley_says("Did you even send a text?");
  }
  printf("Message received: '%s'\n", text_message);

  if ( strncmp(BEST_MASCOT, text_message, strlen(BEST_MASCOT)) != 0) {
    rick_astley_says(". . which won't work. We need a better idea than that.");
  }
  free( text_message);

  congrats(3, "Let's make some plans - Wait for my signal then dance quickly.");
  check_stage(3);
  hardest_mp = SHELL;
  usleep(rand() % 200000);
  hardest_mp = MALLOC;
  trace("Sending signal...");
  int piper[2];
  pipe(piper);
  close(piper[0]);
  write(piper[1],"I just wanna tell you how I'm feeling", 1);
  usleep(5000);
  if( hardest_mp != MAPREDUCE) {
    rick_astley_says("Wrong sequence of events! And if you ask me how I'm feeling. Don't
tell me you're too blind to see we need a U of I student to when to handle these hot
signals.");
  }
   congrats(4, "We know the game, and we're gonna play it");
}
// public API
void debug_shell_mp(char* besticecream) {
  trace("debug_shell_mp");
  check_stage(1);
  if ( strcmp(besticecream,"Altgeld DQ") && strcmp(besticecream, "Siebel") &&
strcmp(besticecream, "Foellinger") ) {
    congrats(2,"Nice - Let's check the text messsages for mascot ideas");
```

```
    atexit(check_completed);
    check_mascot();
  }
}
void piping_hot() {
  trace( "piping_hot");
  check_stage(3);
  if( hardest_mp == MAL
    hardest_mp = MAPRED
  } else {
    rick_astley_says("D                    t wait for my signal!");
  }
}
void* race_to_the_finis
  if(stage == 5) return
  check_stage(4);
  if ( arg != (void*) 0x20230341 )  { rick_astley_says("So close - don't desert me"); }

  pthread_mutex_lock(&mutex);
  int success = (++ leaky_barrier == 20 && oops_i_did_it_again > 5;
  pthread_mutex_unlock(&mutex);
  if ( ! success) {
    usleep(1000*( 500 + (rand() & 1023)));
    pthread_mutex_lock(&mutex);
    leaky_barrier --;
    oops_i_did_it_again ++;
    if(stage == 4) {
      printf("Leaky Barrier! %d threads escaped! (%d remain)\n", oops_i_did_it_again,
leaky_barrier );
    }
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
  }
  pthread_mutex_lock(&mutex);
  if ( ! secret) {
    const char* chorus[] = {"give you up","let you down","run around and desert
you","make you cry","say goodbye","tell a lie and hurt you"};
    unsigned int r = rand();
    asprintf(&secret,"%x: Never gonna %s", r, chorus[r % 6]);
    congrats(5,"Call finish with the correct lyric message to win!\n(Hint: sometimes the
most important code does nothing)");
  }
  pthread_mutex_unlock(&mutex);
  char* result = strdup(secret);
  pthread_exit((void*) result);
}
void never_gonna_say_goodbye(char* lyric) {
  trace("never_gonna_say_goodbye...");
  check_stage(5);
  alarm(1);
  sleep(2);
  if ( strcmp(secret, lyric) == 0) {
    completed = 1;
    puts( lyric );
    puts( "CS341 Puzzle Complete - thanks for playing the CS341 puzzle.\n\nYour CS341
Experience is now complete.\n\nGoodbye!" );
    exit(EXIT_SUCCESS);
  }
  rick_astley_says("Incorrect lyric");
}
```

END OF EXAM